LibreOffice

Equipo de documentación de LibreOffice

Guía de Base

6.2

Derechos de autor

Colaboradores

Este libro está adaptado de versiones anteriores del mismo.

De esta edición

Pulkit Krishna
Alain Romedenne
Dan Lewis
Jean-Pierre Ledure
Jean Hollis Weber
Randolph GAMO

De ediciones previas

Jochen Schiffers
Martin Fox
Dan Lewis
Andrew Pitonyak
Robert Großkopf
Hazel Russman
Peter Schofield
Jost Lange
Jean Hollis Weber
Steve Schwettman

De la edición en español

Juan Peramos
Celia Palacios
Juan C. Sanz Cabrero
Ignacio Alcalá
B. Antonio Fernández
Jonatán Perren

Comentarios y sugerencias

Puede dirigir cualquier clase de comentario o sugerencia acerca de este documento a: documentation@es.libreoffice.org.

Nota

Todo lo que envíe a la lista de correo, incluyendo su dirección de correo y cualquier otra información personal que escriba en el mensaje se archiva públicamente y no puede ser borrada

Fecha de publicación y versión del programa

Versión en español publicada el 4 de agosto de 2021. Basada en la versión 6.2 de LibreOffice.

Contenido

Guía de Base

Preámbulo

¿Para quién es este libro?

Cualquiera que quiera ponerse al día rápidamente con LibreOffice Base encontrará valioso este libro. Ya sea que nunca antes haya trabajado con bases de datos, o que haya trabajado con ellas en un DBMS (Sistema de gestión de bases de datos), o que esté acostumbrado a otro sistema de base de datos desde una suite ofimática o a un sistema de base de datos independiente como MySQL, este libro es para él.

Es posible que desee leer primero el Capítulo 8, *Primeros pasos con Base*, en la *Guía de primeros pasos*.

¿Qué hay en este libro?

Este libro presenta Base, el componente de base de datos de LibreOffice. Base utiliza el motor de base de datos HSQLDB[1] para crear documentos de base de datos. Puede acceder a bases de datos creadas por muchos programas de bases de datos, incluidos Microsoft Access, MySQL, Oracle y PostgreSQL. Base incluye una funcionalidad adicional que le permite crear aplicaciones completas basadas en datos.

- Este libro presenta las características y funciones de Base, utilizando un conjunto de bases de datos de muestra.
- Crear una base de datos
- Acceso a bases de datos externas.
- Crear y usar tablas en bases de datos relacionales.
- Creación y uso de formularios para la entrada de datos.
- Usar consultas para reunir datos de diferentes tablas, calcular resultados cuando sea necesario y filtrar rápidamente un registro específico de una masa de datos
- Crear informes usando el Generador de informes
- Vinculación de bases de datos a otros documentos y formularios externos, incluido el
- uso en la combinación de correspondencia
- Filtrado y búsqueda de datos.
- Uso de macros para evitar errores de entrada, simplificar tareas y mejorar la usabilidad de los formularios.
- Mantenimiento de bases de datos.

Bases de datos de muestra

Se ha creado un conjunto de bases de datos de muestra para acompañar este libro. Se pueden descargar aquí: https://wiki.documentfoundation.org/images/5/52/Sample-databases.zip

- Media_without_macros.odb
- Media_with_macros.odb
- Example_Sport.odb (Chapter 1, "Introduction to Base")
- Example_CSV_included.odb (Chapter 2, "Creating a Database")
- Example_jump_Cursor_Subform_Mainform.odb (Chapter 4, "Forms")
- Example_Report_conditional_Overlay_Graphics.odb (Chapter 6, "Reports")
- Example_Report_Bill.odb (Chapter 6, "Reports")
- Example_Report_Rows_Color_change_Columns.odb (Chapter 6, "Reports")

1 Una base de datos interna de Firebird también está disponible, bajo "características experimentales".

- Example_PDFForm_Import.odb (Chapter 7, "Linking to Databases")
- Example_Autotext_Searchmark_Spelling.odb (Chapter 8, "Database tasks")
- Example_Documents_Import_Export.odb (Chapter 8, "Database tasks")
- Example_Search_and_Filter.odb (Chapter 9, "Macros")
- Example_direct_Calculation_Form.odb (Chapter 9, "Macros")
- Example_Combobox_Listfield.odb (Chapter 9, "Macros")
- Example_serial_Number_Year.odb (Chapter 9, "Macros")
- Example_Database_Formletter_direct.odb (Chapter 9, "Macros")
- Example_Mail_File_activate.odb (Chapter 9, "Macros")
- Example_Dialogs.odb (Chapter 9, "Macros")

Dónde obtener más ayuda

Este libro, las otras guías de usuario de LibreOffice, el sistema de ayuda incorporado y los sistemas de asistencia al usuario suponen que está familiarizado con su computadora y las funciones básicas, como iniciar un programa, abrir y guardar archivos.

Sistema de ayuda

LibreOffice viene con un extenso sistema de ayuda. Esta es su primera línea de soporte para usar LibreOffice. Los usuarios de Windows y Linux pueden elegir descargar e instalar la Ayuda sin conexión para usarla cuando no esté conectado a Internet; la Ayuda sin conexión se instala con el programa en macOS.

Para mostrar el sistema de ayuda, presione *F1* o seleccione **Ayuda > Ayuda de LibreOffice** en el menú Ayuda. Si no tiene la ayuda sin conexión instalada en su computadora y está conectado a Internet, su navegador predeterminado abrirá las páginas de Ayuda en línea en el sitio web de LibreOffice. El menú Ayuda también incluye enlaces a otra información de LibreOffice y servicios de soporte.

Puede colocar el puntero del ratón sobre cualquiera de los iconos de la barra de herramientas para ver un pequeño cuadro *información sobre herramientas* con una breve explicación de la función del icono. Para una explicación más detallada, seleccione **Ayuda> ¿Qué es esto?** desde la barra de menú y mantenga el puntero sobre el icono.

Además, puede elegir si desea activar Descripciones emergentes ampliadas usando **Herramientas > Opciones > LibreOffice > General**.

Soporte en línea gratuito

La comunidad de LibreOffice no solo desarrolla software, sino que también ofrece soporte gratuito y voluntario. Además de los enlaces del menú de *Ayuda* anteriores, hay otras opciones de soporte comunitario disponibles en línea, consulte la tabla a continuación.

Free LibreOffice support	
FAQs	Respuestas a preguntas frecuentes https://wiki.documentfoundation.org/Faq/es
Listas de correo	Apoyo gratuito de la comunidad proporcionado por una red de usuarios experimentados. https://www.libreoffice.org/get-help/mailing-lists/
Preguntas y respuestas y Base de conocimientos	Proporciona asistencia comunitaria gratuita en un servicio web con formato de preguntas y respuestas. Busque temas similares o abra uno nuevo en: https://ask.libreoffice.org/es/questions/ El servicio está disponible en varios idiomas; simplemente reemplace /es/ con: de, en, fr, ja, ko, nl, pt, tr y muchos otros en la dirección web anterior.

Free LibreOffice support	
Soporte en Lengua nativa	El sitio web de LibreOffice en varios idiomas. https://www.libreoffice.org/community/nlc/ Listas de correo para lenguas nativas https://wiki.documentfoundation.org/Local_Mailing_Lists Información sobre redes sociales https://wiki.documentfoundation.org/Website/Web_Sites_services
Accessibility options	Información sobre las opciones de accesibilidad disponibles. https://www.libreoffice.org/get-help/accessibility/
Foro OpenOffice	Otro foro que brinda soporte para LibreOffice, entre otras suites ofimáticas de código abierto https://forum.openoffice.org/es/forum/

Soporte y formación, de pago

También puede pagar por el soporte a través de contratos de servicio de un proveedor o firma consultora especializada en LibreOffice. Para obtener información sobre el soporte profesional certificado, consulte el sitio web de The Document Foundation:
https://www.documentfoundation.org/gethelp/support/

Lo que está viendo podría ser diferente

Ilustraciones

LibreOffice se ejecuta en los sistemas operativos Windows, Linux y macOS, cada uno de los cuales tiene varias versiones y los usuarios pueden personalizarlo (fuentes, colores, temas, administradores de ventanas). Las ilustraciones de esta guía pueden haber sido tomadas en distintos sistemas operativos, por lo tanto, algunas ilustraciones puede que no se vean exactamente como las que ve en la pantalla de su computadora.

Además, algunos de los cuadros de diálogo pueden diferir debido a la configuración seleccionada en LibreOffice. En algunos sistemas, puede usar cuadros de diálogo del sistema operativo de su computadora (predeterminados) o cuadros de diálogo proporcionados por LibreOffice. Para cambiar qué diálogos se usan, vaya a: **Herramientas > Opciones > LibreOffice > General** y seleccione o anule la selección de la opción **Usar diálogos de LibreOffice**.

Iconos

Los iconos utilizados para ilustrar algunas de las muchas herramientas disponibles en LibreOffice pueden diferir de las utilizadas en esta guía. Los iconos de esta guía se han tomado de una instalación de LibreOffice que se configuró para mostrar el conjunto de iconos de Colibre.

Para cambiar el conjunto de iconos utilizado, vaya a: **Herramientas > Opciones > LibreOffice > Ver**. En la sección *Estilo de iconos*, elija uno de la lista desplegable.

Uso de LibreOffice en Mac

Algunas teclas y elementos de menú en macOS son diferentes de los utilizados en Windows y Linux.

La tabla siguiente ofrece algunas sustituciones comunes para las instrucciones de este capítulo. Para obtener una lista más detallada, consulte la *Ayuda* de la aplicación.

Windows o Linux	*Equivalente en Mac*	*Efecto*
Herramientas > Opciones opción de menú	**LibreOffice > Preferencias**	Acceso a las opciones de configuración
Clic con el botón derecho	*Control+clic* o *clic derecho* depende de la configuración del equipo	Abre menú contextual
Ctrl (Control)	⌘ *(Comando)*	Utilizado con otras teclas
F5	*Mayúscula+⌘+F5*	Abre el navegador
F11	*⌘+T*	Abre la ventana de estilos y formato

¿Cómo se denominan todas estas cosas?

Los términos utilizados en LibreOffice para la mayoría de las partes de la interfaz de usuario (las partes del programa que ve y usa, en contraste con el código detrás de escena que realmente lo hace funcionar) son las mismas que para la mayoría de los otros programas.

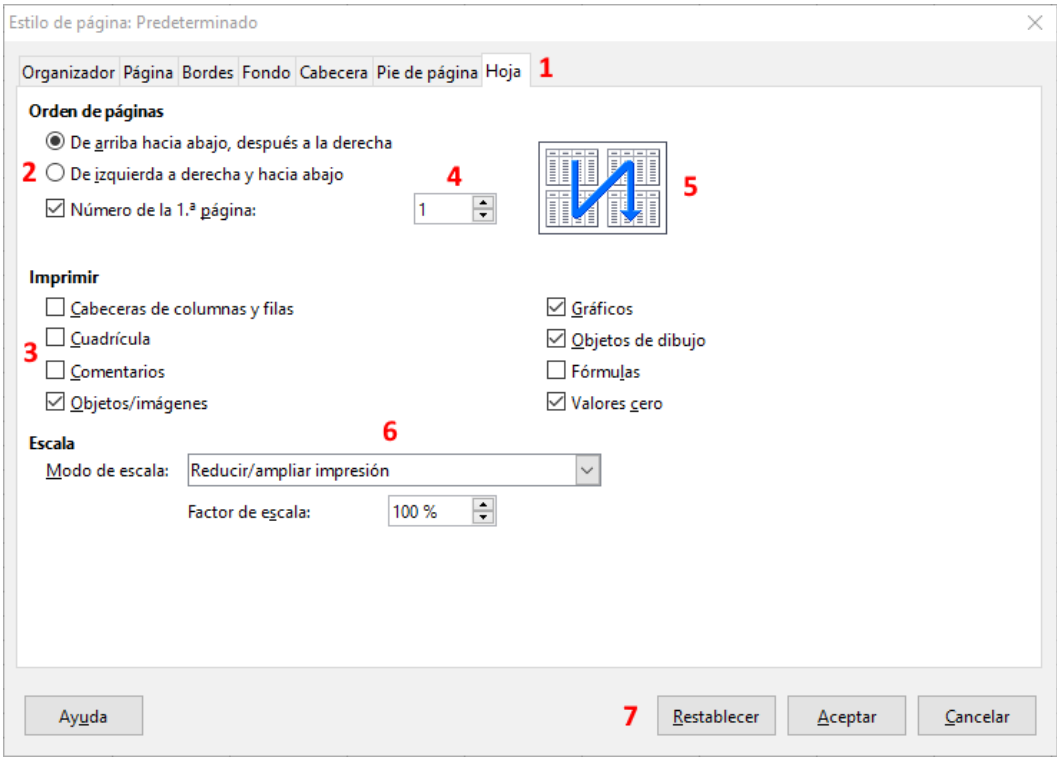


Figura 1: Diálogo (de Calc) que muestra controles comunes

1) Página con pestañas (estrictamente hablando no es un control).
2) Botones de radio (solo se puede seleccionar uno a la vez).
3) Casilla de verificación (se puede seleccionar más de una a la vez).
4) Cuadro de número (haga clic en las flechas hacia arriba y hacia abajo para cambiar el número que se muestra en el cuadro de texto al lado, o escríbalo en el cuadro de texto).
5) Miniatura o vista previa.
6) Lista desplegable para seleccionar un elemento.
7) Botones.

Un *diálogo* es un tipo especial de ventana. Su propósito es informarle de algo, o solicitar su opinión, o ambos. Proporciona controles que puede usar para especificar cómo llevar a cabo una acción. Los nombres técnicos de los controles comunes se muestran en la Figura 1. En la mayoría de los casos, no usamos los términos técnicos en este libro, pero es útil conocerlos porque la Ayuda y otras fuentes de información a menudo los usan.

En la mayoría de los casos pueden interactuar solo con el cuadro de diálogo (no con el documento en sí) mientras que el cuadro de diálogo permanezca abierto. Cuando cierre el cuadro de diálogo después de usarlo (generalmente, al hacer clic en **Aceptar** u otro botón se cierra el cuadro de diálogo), puede volver a trabajar con el documento.

Algunos cuadros de diálogo se pueden dejar abiertos mientras trabaja, para que pueda alternar entre el cuadro de diálogo y el documento. Un ejemplo de este tipo es el cuadro de diálogo *Buscar y Reemplazar*.

Reconocimientos

Este libro fue traducido del manual alemán *LibreOffice 6.2 Base Handbuch*, disponible aquí: (ODT) https://wiki.documentfoundation.org/images/6/6c/Base_Gesamtband_einseitig_V62.odt (PDF) https://wiki.documentfoundation.org/images/1/1a/Base_Gesamtband_einseitig_V62.pdf

Preguntas frecuentes

¿Cómo se licencia LibreOffice?

LibreOffice se distribuye bajo the Open Source Initiative (OSI) aprobada por Mozilla Public License (MPL). Vea https://es.libreoffice.org/acerca-de/licencias/

Se basa en el código de Apache OpenOffice disponible bajo la Licencia Apache 2.0, pero también incluye software, que difiere de una versión a otra, bajo una variedad de otras licencias de Código Abierto. El nuevo código está disponible bajo LGPL 3.0 y MPL 2.0.

¿Puedo distribuir LibreOffice a cualquiera?¿Puedo venderlo?¿Puedo usar LibreOffice en mi negocio?

Sí.

¿En cuántas computadoras puedo instalarlo?

En tantas como quiera.

¿LibreOffice está disponible en mi idioma?

LibreOffice se ha traducido (localizado) a más de 40 idiomas, por lo que su idioma probablemente sea compatible. Además, hay más de 70 diccionarios de ortografía, separación silábica y sinónimos disponibles para idiomas y dialectos que no tienen una interfaz de programa localizada. Los diccionarios están disponibles en el sitio web de LibreOffice en: www.libreoffice.org.

¿Por qué necesito Java para ejecutar LibreOffice? ¿Está escrito en Java?

LibreOffice no está escrito en Java; Está escrito en el lenguaje C ++. Java es uno de los varios lenguajes que se pueden usar para extender el software. Java JDK / JRE solo es necesario para algunas funciones; la más notable es el motor de base de datos relacional HSQLDB.

¿Cómo puedo contribuir a LibreOffice?

Puede ayudar con el desarrollo y la asistencia al usuario de LibreOffice de muchas maneras, y no necesita ser un programador. Para comenzar, visita esta página web: https://es.libreoffice.org/comunidad/involucrate/

¿Puedo distribuir el PDF de este libro o imprimir y vender copias?

Sí, siempre y cuando cumpla con los requisitos de una de las licencias en la declaración de derechos de autor al comienzo de este libro. No tiene que solicitar un permiso especial.

Además, le pedimos que comparta con el proyecto algunas de las ganancias que obtiene de la venta de libros, en consideración a todo el trabajo que hemos realizado para producirlos.

Donaciones a LibreOffice: https://www.libreoffice.org/donate/

LibreOffice

Guía de Base

Capítulo 1
Introducción a Base

Contenido

Introducción

En la operación diaria de la oficina, las hojas de cálculo se usan regularmente para agregar conjuntos de datos y realizar algún tipo de análisis sobre ellos. Como los datos en una hoja de cálculo se presentan en una vista de tabla, claramente visibles y se pueden editar o agregar, muchos usuarios preguntan por qué deberían usar una base de datos en lugar de una hoja de cálculo. Este libro explica las diferencias entre los dos.

Nota

En lenguaje técnico, el "archivo de documento de base de datos" se utiliza para una base de datos desde una única interfaz y el "sistema de base de datos" abarca el sistema de gestión de base de datos (DBMS) y la base de datos real(database).

Base ofrece acceso a varios sistemas de bases de datos a través de una interfaz gráfica de usuario. Base funciona de manera predeterminada con el motor de base de datos incorporado HSQLDB.

Este capítulo presenta dos bases de datos de muestra y todo este libro se basa en ellas: Media_without_macros.odb y Media_with_macros.odb, ampliado con la inclusión de macros.

Ambas bases de datos son para operar sobre una biblioteca: captura de medios, captación de usuarios, alquiler de medios y todo lo relacionado con ella, como el retiro para lectores.

Más adelante en este capítulo se da un ejemplo más detallado (comenzando en la página 29). Este utiliza la base de datos Example_Sport.odb para organizar una competición deportiva.

Nota

Como cualquier software, LibreOffice Base no está completamente libre de errores. Particularmente molestas son las regresiones, que reintroducen un error de una versión anterior en la versión actual. El siguiente enlace conduce a las regresiones pendientes actualmente:

https://bugs.documentfoundation.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=REOPENED&bug_status=NEEDINFO&bug_status=PLEASETEST&component=Database&keywords=regression&order=Importance&product=LibreOffice

Una mirada a la lista de errores puede ayudarlo a comprender las diferencias entre la documentación y su propia versión del programa.

Base – un contenedor para el contenido de la base de datos

Un archivo Base es una carpeta comprimida que contiene información para las diferentes áreas de trabajo de Base. En el uso diario, Base se abre inicialmente con la vista que se muestra en la Figura 2.

El entorno Base contiene cuatro áreas de trabajo: *Tablas*, *Consultas*, *Formularios* e *Informes*. Dependiendo del área de trabajo seleccionada, se pueden llevar a cabo varias tareas: crear contenido nuevo o recuperar elementos existentes.

En las áreas de trabajo *Formularios* e *Informes*, los elementos respectivos se organizan dentro de una estructura de directorio (Figura 3). Esto se hace directamente al guardar en el diálogo *Guardar* o mediante la creación de nuevas carpetas usando **Insertar > Carpeta**.

Aunque la esencia de una base de datos está formada por tablas, Base comienza con la vista *Formularios* porque los formularios son los elementos más utilizados cuando se trabaja con bases de datos. Con los formularios, puede hacer entradas en las tablas y analizar el contenido de las mismas.

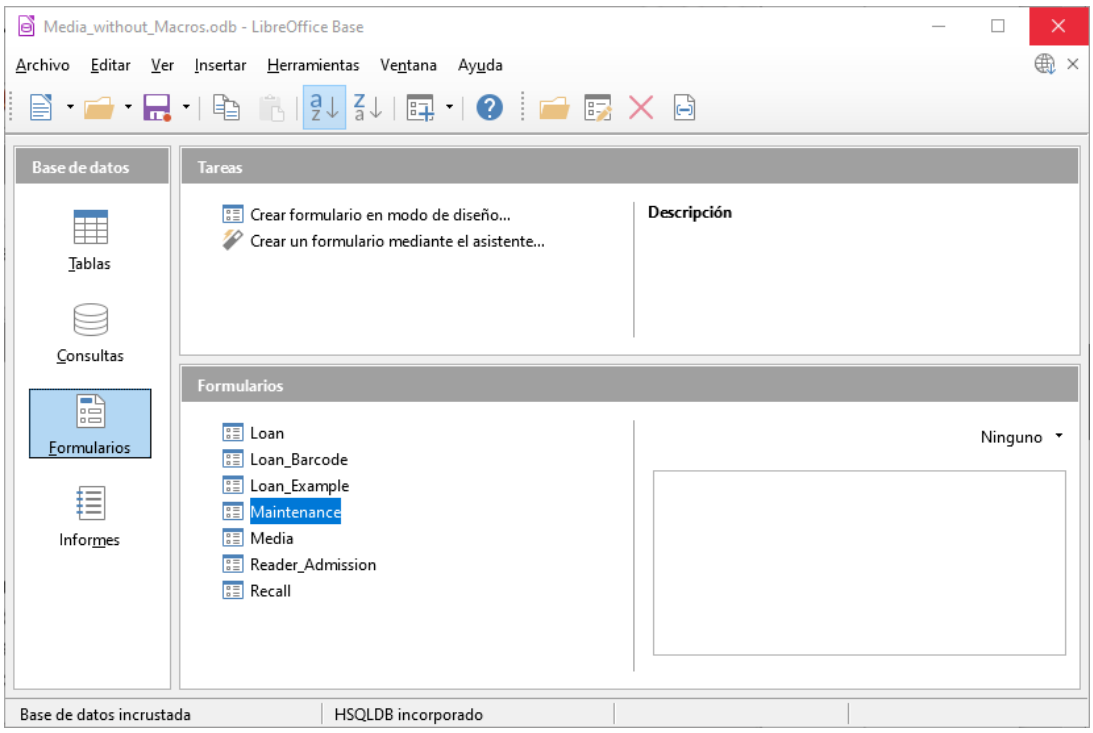


Figura 2: Vista de Base cuando se abre

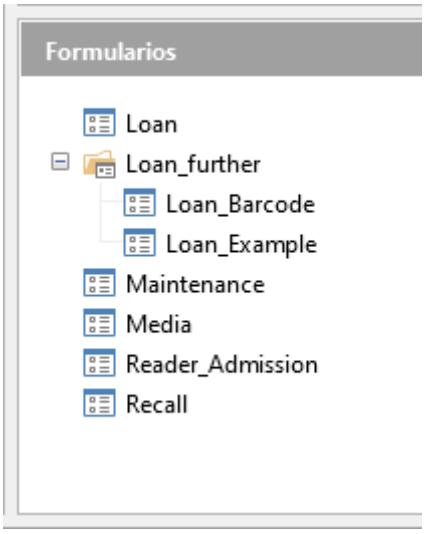


Figura 3: Estructura de directorios en un área de trabajo.

Entrada de datos usando formularios

Los formularios simples muestran solo una tabla, como en la parte superior del formulario *Loan*. El formulario *Loan* (Figura 4) se ha ampliado para mostrar información adicional:

- El rango de personas que se muestra se puede filtrar por apellido para limitar los detalles que se muestran. Si un usuario ingresa la letra "G" en el campo *Filter (Last Name)* a la derecha de la tabla de *Loan*, solo se mostrarán las personas cuyo apellido comience con "G".
- La información del nuevo prestatario se puede ingresar directamente en los campos de la tabla del formulario.
- Los detalles de los artículos que se tomarán prestados se ingresan y se muestran en el área central del formulario. El nombre del usuario también se destaca claramente. Si un

artículo prestado anteriormente está vencido y debe devolverse, esta área está bloqueada (no es posible ingresar datos) y el título indicará "Préstamo bloqueado temporalmente". Los artículos prestados se muestran en el área inferior del formulario.

- La fecha de préstamo (*Loan Date*) se establece como la fecha actual. En el campo desplegable a la izquierda del botón *Refresh* se encuentran los elementos multimedia que se pueden pedir prestados. Los artículos que ya están prestados al prestatario seleccionado no están disponibles para su selección.
- Los elementos multimedia seleccionados para el préstamo se agregan a los detalles del préstamo actual haciendo clic en el botón *Refresh*.
- En la sección inferior del formulario (*Return*) no es posible eliminar filas de datos. Solo se pueden editar los campos *Return Date* y *Extension*. Si un prestatario anteriormente estaba bloqueado y posteriormente ha devuelto los artículos vencidos, el área de préstamo puede desbloquearse haciendo clic en el botón *Refresh*.

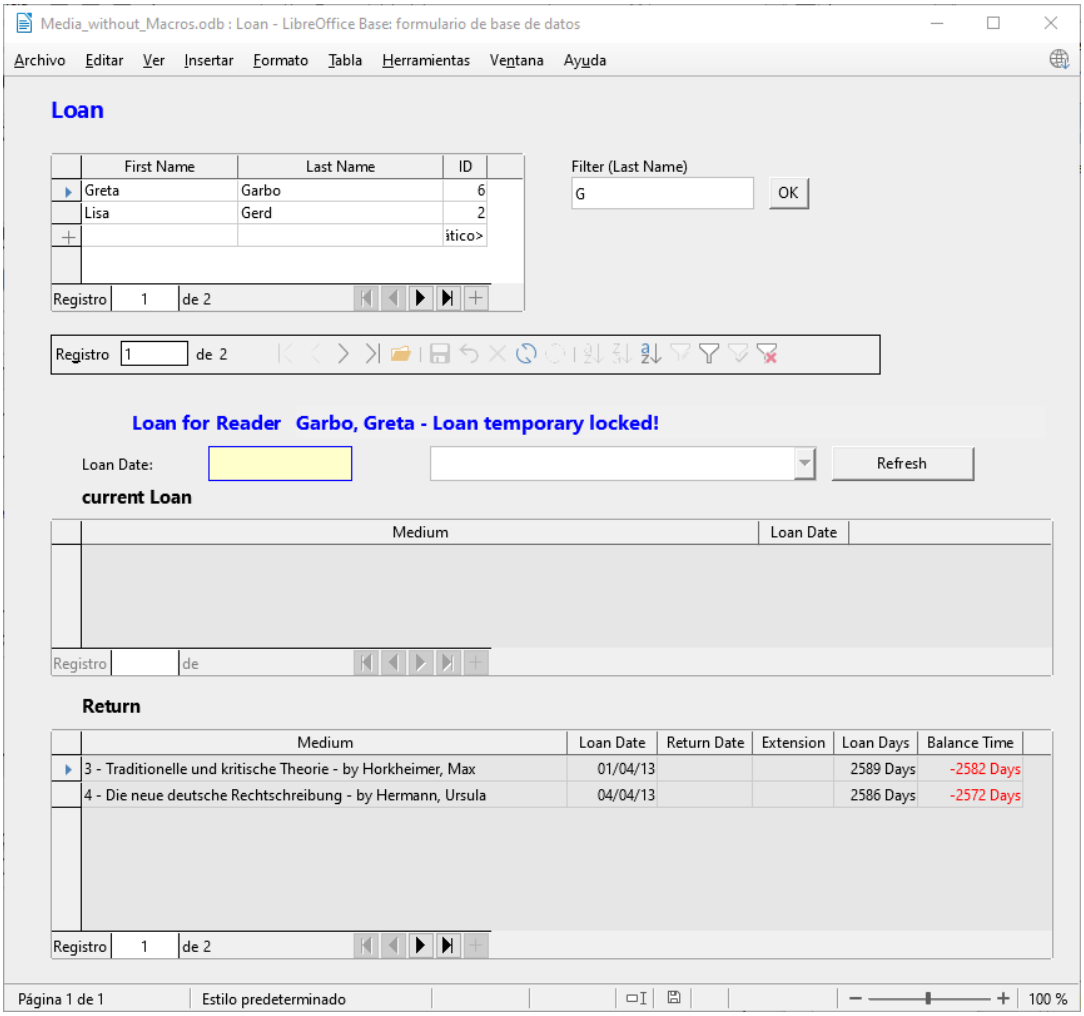


Figura 4: El formulario Loan

Todas estas funciones pueden llevarse a cabo sin usar macros, cuando el formulario se configura y se completa de la manera descrita.

Entrada directa de datos en una tabla: conceptos básicos para la entrada de datos

Las tablas en una base de datos están relacionadas de manera muy similar a una red. Una tabla recibe información de otras tablas o se la proporciona. Esto se conoce como la relación y se muestra mediante una línea entre las tablas que conectan un campo de cada una.

Nota

La tabla *Reader* tiene una relación con otra tabla que involucra *Gender_ID*. De manera similar, la tabla *Media* tiene una relación con cuatro tablas más, cada una de las cuales involucra uno de estos campos: *Category_ID*, *Mediastyle_ID*, *Town_ID* y *Publisher_ID*.

La tabla *Loan* está directamente relacionada con las tablas *Media* y *Reader*, como se muestra en la Figura 5.

Cuando se toma prestado un libro, en lugar de guardar su título en la tabla *Loan*, solo se guarda un número en el campo *Media_ID*. El campo *ID* de la tabla *Media* almacena el identificador único para cada registro de esta tabla.

Este campo es un campo clave de la tabla *Media*: la clave principal o clave primaria.

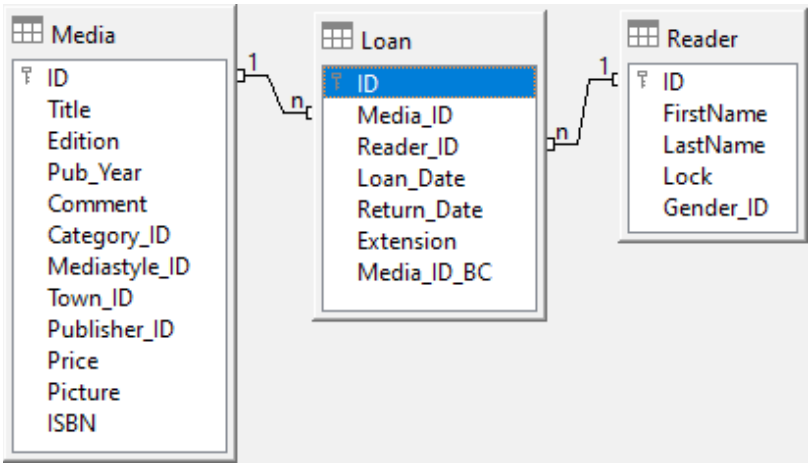


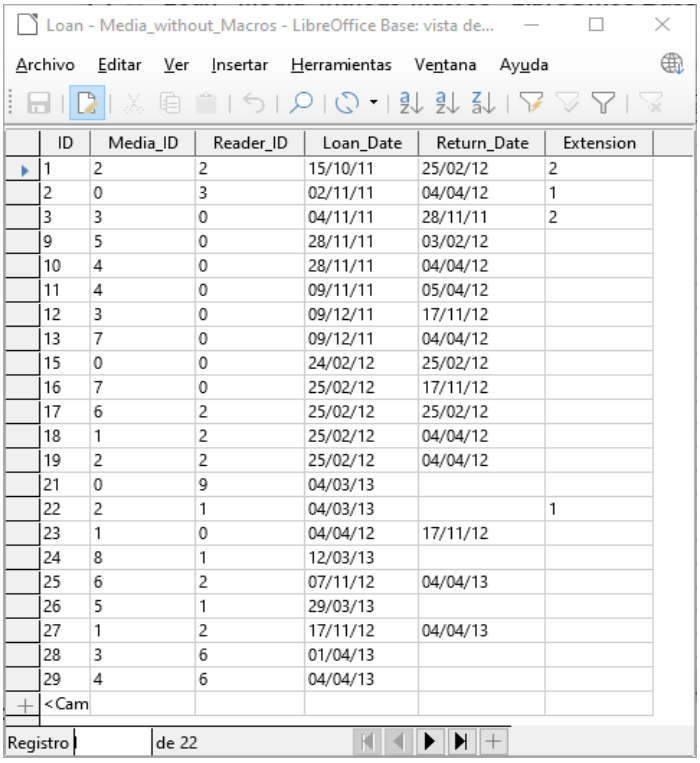
Figura 5: Relación entre las tablas Loan y Reader

Consejo

La clave primaria determina de forma exclusiva los valores para cada campo en cada registro de una tabla. Por lo que, cuando se toma prestado un artículo, el número ingresado en el campo *Media_ID* coincide con el número en el campo *ID* de la tabla *Media* que identifica el registro que contiene la información sobre el artículo prestado.

El nombre del lector no se ingresa en la tabla *Loan* cada vez. Esta información se guarda en la tabla *Reader*. También tiene un campo de clave principal que identifica a cada persona que toma prestado un artículo. El valor de este campo se puede introducir en la tabla *Loan* con el campo *Reader_ID* que identifica a la persona específica.

Las relaciones entre las tablas tienen la ventaja de que el trabajo de escritorio en el formulario se reduce considerablemente. En lugar de tener que ingresar el título de medios y los nombres y apellidos sin ningún error, estos se pueden ingresar seleccionando los números correctos para los campos *Media_ID* y *Reader_ID*, lo que permite la selección de los elementos de medios correctos y los nombres y apellidos. Finalmente, el mismo medio puede ser prestado nuevamente más tarde y el mismo lector puede pedir prestado varios medios más en el primer evento de *Loan*.



ID	Media_ID	Reader_ID	Loan_Date	Return_Date	Extension
1	2	2	15/10/11	25/02/12	2
2	0	3	02/11/11	04/04/12	1
3	3	0	04/11/11	28/11/11	2
9	5	0	28/11/11	03/02/12	
10	4	0	28/11/11	04/04/12	
11	4	0	09/11/11	05/04/12	
12	3	0	09/12/11	17/11/12	
13	7	0	09/12/11	04/04/12	
15	0	0	24/02/12	25/02/12	
16	7	0	25/02/12	17/11/12	
17	6	2	25/02/12	25/02/12	
18	1	2	25/02/12	04/04/12	
19	2	2	25/02/12	04/04/12	
21	0	9	04/03/13		
22	2	1	04/03/13		1
23	1	0	04/04/12	17/11/12	
24	8	1	12/03/13		
25	6	2	07/11/12	04/04/13	
26	5	1	29/03/13		
27	1	2	17/11/12	04/04/13	
28	3	6	01/04/13		
29	4	6	04/04/13		
<Cam					

Figura 6: Estructura de datos de la tabla

La estructura de la tabla para tal formulario es relativamente básica y fácil de configurar.

En la tabla que se muestra en la Figura 6, los datos se pueden ingresar directamente en las filas y columnas de la tabla como cuando se usa el formulario. Las relaciones de esta tabla con las otras tablas de la base de datos se utilizan en el formulario.

- El campo más importante, la clave primaria (*ID*) que se añade automáticamente, muestra el contenido indispensable y único para la mayoría de las bases de datos. Para obtener más información sobre este tema, consulte la sección "*Relaciones entre tablas*" en el *Capítulo 3, Tablas*.
- El segundo campo, *Media_ID*, almacena valores de la clave primaria de la tabla *Media*. Se refiere al número en el campo correspondiente, *ID*, en la tabla *Media*. Dicha referencia a una clave primaria se denomina clave externa. En el formulario, se mostrarán el título y el autor en un cuadro de lista en lugar de la clave externa. El cuadro de lista transmite, en segundo plano, el valor a la clave externa de la tabla.
- El tercer campo, *Reader_ID*, almacena los valores de la clave primaria de la tabla *Reader*. En este ejemplo, esta clave es solo un número que se refiere al lector que toma prestados elementos multimedia. En el formulario, se muestran el apellido y el nombre del lector. Como se ve en la tabla, el lector con el número de clave principal '0' ha tomado prestados muchos medios. La tabla puede guardar la clave primaria única de la tabla *Reader* como una clave externa *Reader_ID* muchas veces. Pero en ningún caso puede un lector, que figura en la clave externa de la tabla *Loan*, ser eliminado en la tabla *Reader*. De lo contrario, ya no sería comprensible que alguien hubiera prestado medios. La base de datos realiza la configuración predeterminada para que sea imposible eliminarla. El término técnico para esto es el requisito de *integridad referencial*.

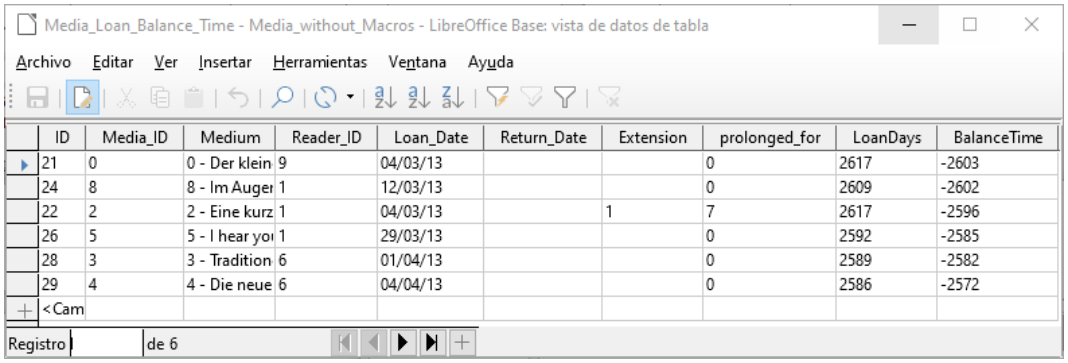
La fecha del préstamo (*Loan_Date*) se almacena en el cuarto campo. Si esta fecha está presente y es posterior a la fecha actual, el conjunto de datos correspondiente para el lector se muestra en la tabla inferior del formulario debajo del botón *Return*.

El campo *Extension* contiene información sobre extensiones del préstamo para un artículo. El significado de los valores 1, 2 ... se explica más adelante. La base de datos contiene una tabla separada con la etiqueta *Settings*.

La entrada de estos datos permite la gestión de una biblioteca simple.

Consultas: obtener información sobre datos en tablas

Las consultas muestran una vista de las tablas. Reúnen contenido de varias tablas en una vista general. Las consultas se almacenan en el lenguaje de consulta SQL. Las consultas no son tablas, aunque en Base parecen ser lo mismo que una tabla para nosotros.



ID	Media_ID	Medium	Reader_ID	Loan_Date	Return_Date	Extension	prolonged_for	LoanDays	BalanceTime
21	0	0 - Der klein	9	04/03/13			0	2617	-2603
24	8	8 - Im Auge	1	12/03/13			0	2609	-2602
22	2	2 - Eine kurz	1	04/03/13		1	7	2617	-2596
26	5	5 - I hear yo	1	29/03/13			0	2592	-2585
28	3	3 - Tradition	6	01/04/13			0	2589	-2582
29	4	4 - Die neue	6	04/04/13			0	2586	-2572
<Cam									

Figura 7: Ejemplo de consulta

La consulta que se muestra en la Figura 7 enumera todos los medios que están actualmente en préstamo. Calcula cuánto tiempo ha estado prestado cada artículo y el saldo del período del préstamo. Cuando *Media_ID*, el campo de clave externa, lee la clave primaria, el título y el autor en el campo *Media* se combinan en un solo texto. Este campo será necesario en el formulario bajo el subtítulo *Return*. Los campos combinados en la consulta también sirven como campos de conexión desde el formulario de préstamo real a la tabla *Loan*, es decir, a través de los campos *Media_ID* y *Reader_ID*.

- Se muestran todos los medios en los cuales la fecha de devolución no se ha introducido en la tabla *Loan*. Como descripción general adicional, el nombre del medio se incluye en la consulta junto con el *Media_ID*.
- La referencia al lector se establece con la clave primaria de la tabla Lector.
- La diferencia horaria en días se especifica como *LoanDays* entre la fecha del préstamo (*Loan_Date*) y la fecha actual.
- El número de días de préstamo se resta del tiempo de préstamo para dar el número restante de días del período del préstamo. El tiempo de préstamo puede variar con diferentes tipos de medios.
- En la tabla *Settings*, un valor de '1' para *Extension* corresponde a una extensión del período de préstamo de 7 días. En el conjunto de datos anterior, la línea con *'2' como Media_ID,* muestra una extensión de 7 días.

Informes - presentación de datos

En los informes, los datos se procesan para que puedan imprimirse en un formato útil. Los formularios como el de la Figura 8 no son adecuados para una carta con formato limpio.

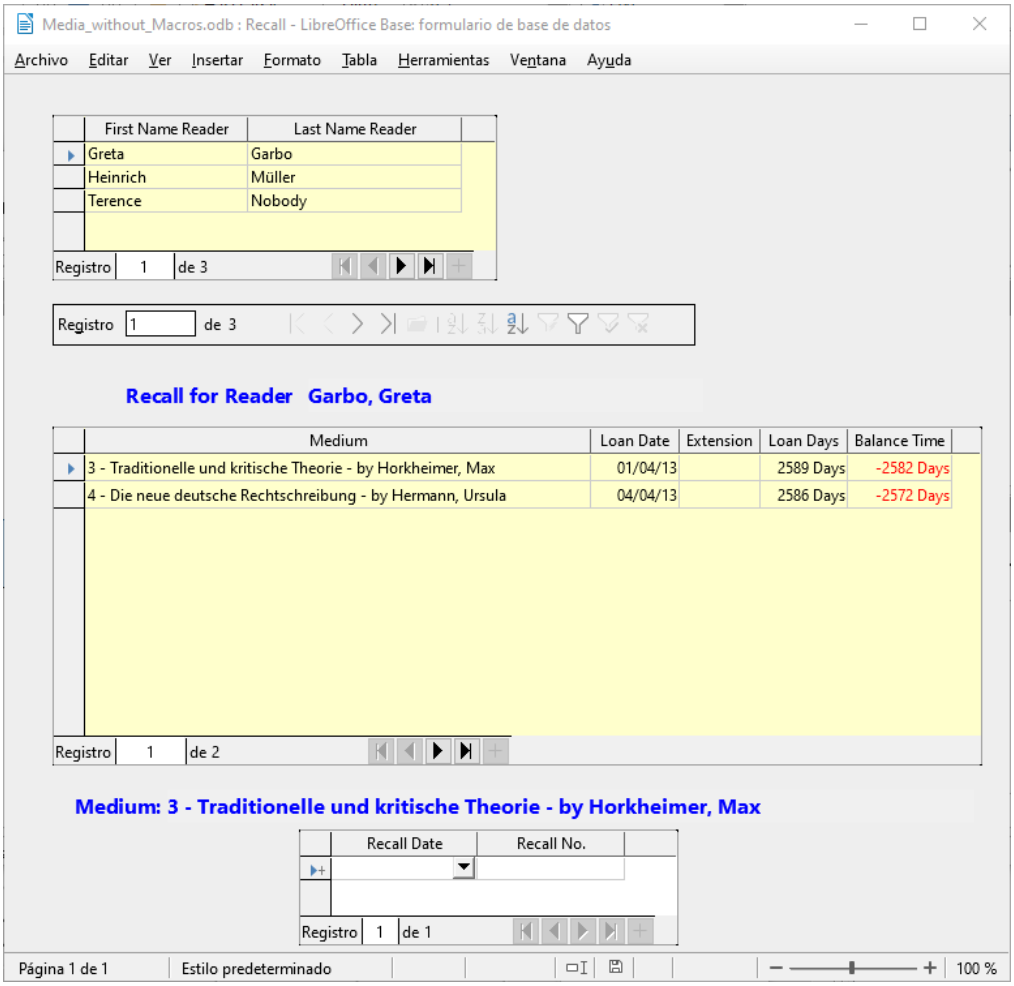


Figura 8: Formulario que contiene información para un aviso de retirada

Antes de que se pueda imprimir un informe real en forma de aviso de retiro, la información de retiro debe ingresarse en el formulario *Recall*. La tabla en el formulario muestra todas las personas que tomaron prestados artículos con un tiempo de préstamo restante negativo.

La fecha *Recall_Date* y el número *Recall No.* de aviso de recuperación se ingresan para cada elemento de medios a recuperar. Esta fecha podría ser la fecha actual en el proceso de advertencias. La fecha de recuperación por defecto es la fecha actual. El *Recall No* es un número entero incrementado en 1 con cada aviso de recuperación sucesivo para un prestamista o medio en particular.

Este formulario, en el ejemplo de esta base de datos sin macros, requiere la entrada del usuario para crear avisos de recuperación.

En la versión con macros, la fecha se ingresa automáticamente y se imprime el aviso de retiro.

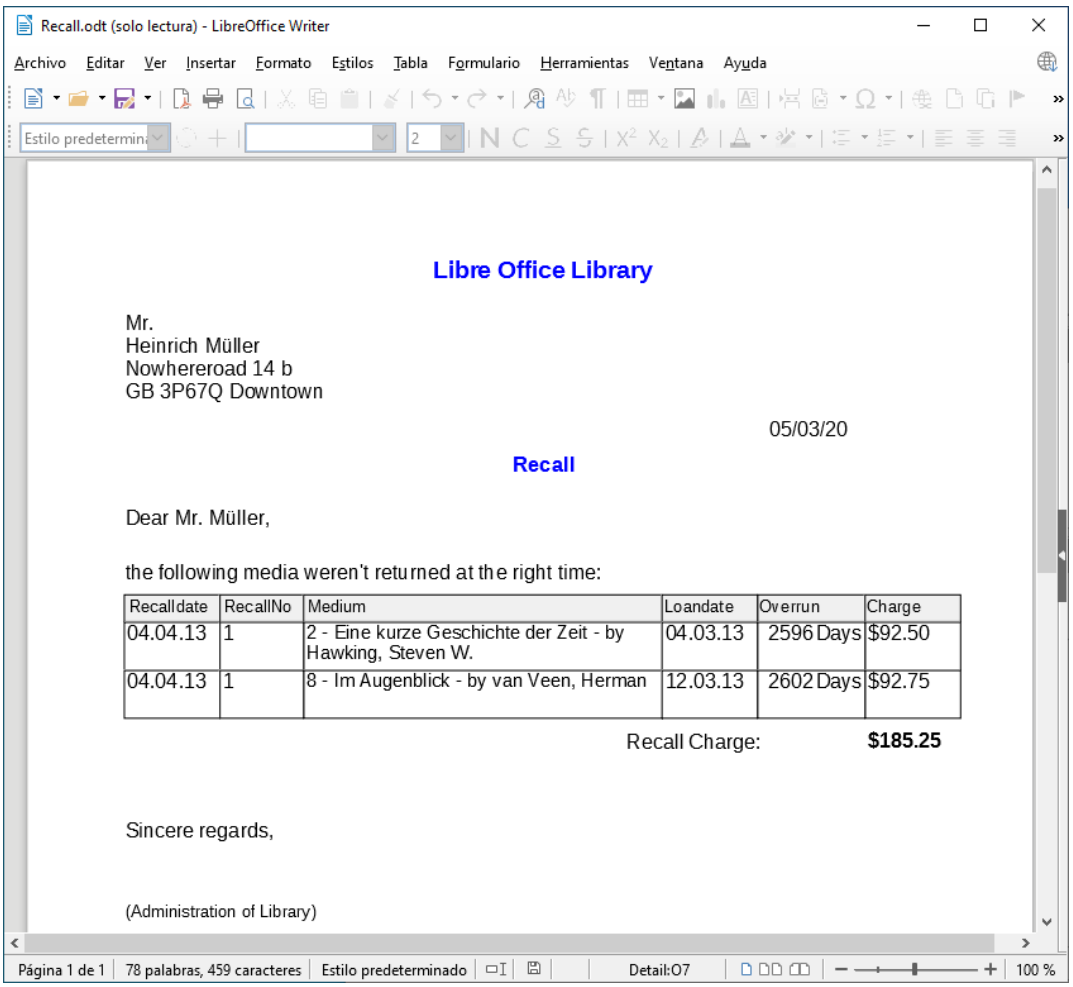


Figura 9: Muestra de Aviso de recuperación

El aviso de recuperación (Figura 9) se genera mediante una consulta a partir de los datos ingresados previamente. El usuario de la base de datos solo necesita seleccionar el informe de recuperación y se puede imprimir y enviar una carta de recuperación a todas las personas que tienen una entrada de recuperación en el formulario de la página anterior.

En dicho informe puede haber múltiples entradas (elementos vencidos) para una persona en particular. Si la tabla que contiene los elementos para esta persona excede el espacio en una página, se extiende para cubrir una página siguiente.

El informe abarca más que una carta de combinación de correspondencia producida con Writer. Reúne automáticamente los conjuntos de datos para imprimir y organiza el texto de acompañamiento necesario en consecuencia.

Una carta similar a la de la figura anterior solo se puede implementar con macros, como se describe en “Crear un informe” en la página 50.

Manejo seguro de un archivo de Base

Las tablas, consultas, formularios e informes de la base de datos interna HSQLDB se almacenan en un archivo de Base. Debido a que el archivo de base de datos se escribe en la memoria, los múltiples objetos que contiene requieren que se trate con cuidado. Los informes de errores dejan en claro que un archivo de base de datos requiere un tratamiento un poco más cuidadoso que, por ejemplo, un archivo de texto escrito en Writer.

Por eso deben tenerse en cuenta las siguientes instrucciones al tratar con un archivo Base:

- Un archivo de base de datos abierto no debe guardarse con un nombre diferente utilizando *Guardar como*. Cuando no hay otra opción, las tablas, consultas, formularios e informes deben cerrarse primero. Es mejor cerrar el archivo de la base de datos y crear una copia del archivo.
- El generador de informes es un complemento. Aunque ya no es visible como una extensión separada, funciona en gran medida independientemente del archivo de la base de datos. Cambiar el nombre del archivo elimina el generador de informes de su esencia.
- Guardar una tabla, consulta, formulario o informe, no implica que se haya guardado todo el archivo de la base de datos. Este guardado debe hacerse por separado. Cuando se guarda un objeto (tabla, consulta, formulario, informe), la información se escribe en el archivo de la base de datos en la memoria; cuando se guarda el archivo de la base de datos, todo lo contenido en el archivo de la base de datos en la memoria se escribe en el archivo **.odb*.
- Este comportamiento de la memoria es especialmente apropiado para trabajar con el Generador de informes. La preparación de un informe sigue siendo el componente más inestable dentro del archivo Base. Por lo tanto, se deben guardar después de cada paso el informe y el archivo *.odb. Una vez creado el informe, funciona por sí solo sin ningún problema en particular.
- Cuando se finaliza un archivo *.odb, los datos agregados a la base de datos solo se escriben en el archivo de la base de datos en la memoria, pero no en el archivo *.odb. Solo cuando cierra el archivo, *.odb se guardan los datos en él.
- El contenido de la base de datos HSQLDB se volverá a escribir en el archivo. Un bloqueo en este punto puede provocar la pérdida de datos. Por lo tanto, se debe desarrollar una estrategia para que las copias de seguridad se realicen a tiempo.
- El *Capítulo 9, Macros*, incluye una macro para hacer una copia de seguridad cuando abre un archivo de base de datos.

  Del mismo modo, se muestra otra manera de hacerlo mientras se abre el archivo Base que es tan buena como segura.

  Finalmente, se puede alcanzar un nivel de seguridad significativamente mayor utilizando bases de datos de servidores externos como MySQL, MariaDB o PostgreSQL.

  Para esto, Base puede servir como interfaz con consultas, formularios e informes para la base de datos.

Una base de datos simple: ejemplo de prueba en detalle

La creación de una base de datos se trata en el *Capítulo 2, Creación de una base de datos*.

El siguiente ejemplo se basa en la base de datos predeterminada HSQLDB que se instala con LibreOffice como una base de datos interna. Por lo tanto, es una base de datos incrustada creada por primera vez sin ningún registro en LibreOffice. Se pueden conectar varios otros sistemas de bases de datos además del HSQLDB interno.

El primer paso a seguir después de encontrar una ubicación para el archivo de base de datos y guardarlo allí es completar el asistente.

La base de datos está destinada a organizar una competición deportiva en diferentes disciplinas. Por lo tanto, *Example_Sport* fue elegido como el nombre del archivo.[2]

Crear tablas

Tan pronto como se haya guardado la base de datos, aparece la ventana principal de la base de datos. De manera predeterminada, Tablas se selecciona en la sección Base de datos en el lado

2 La base de datos *Example_Sport.odb* se incluye en las bases de datos de muestra para esta guía.

izquierdo de la ventana (Figura 10). Las tablas son el almacenamiento central de datos; sin tablas, no hay base de datos.

Haga clic en **Crear tabla en Vista Diseño** para abrir la ventana que se muestra en la Figura 11.

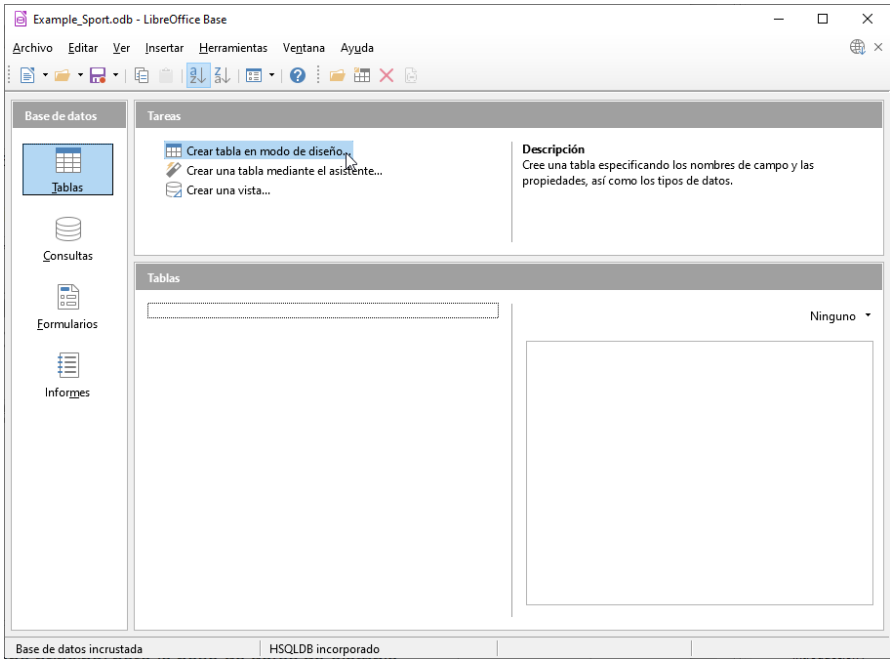


Figura 10: Ventana principal para la base de datos de ejemplo Example_Sport.odb

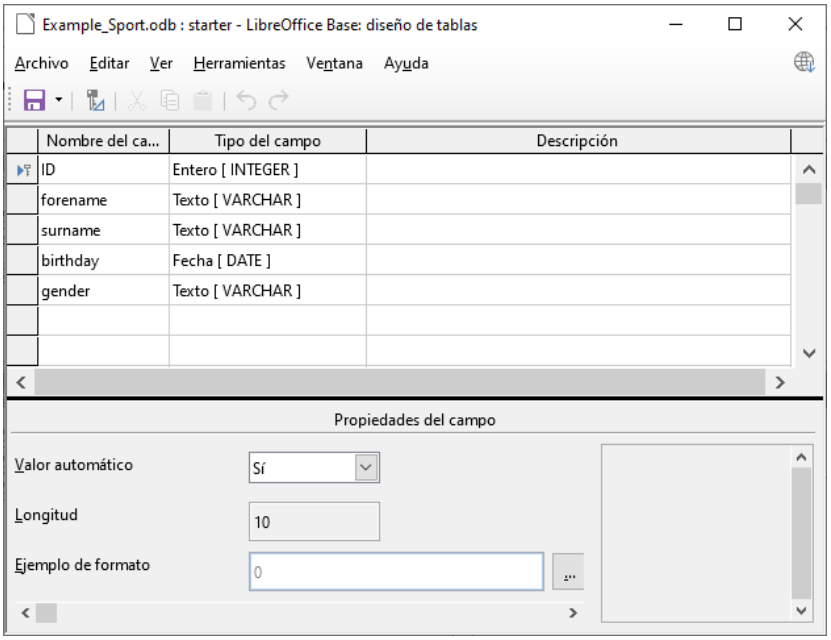


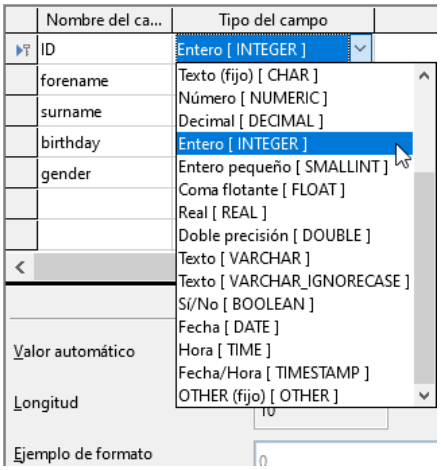
Figura 11: Vista diseño para una tabla

Los nombres de campo para la primera tabla se ingresarán aquí. La tabla debe incluir los entrantes de hombres y mujeres. Aquí los nombres de campo se reducen primero a los componentes clave.

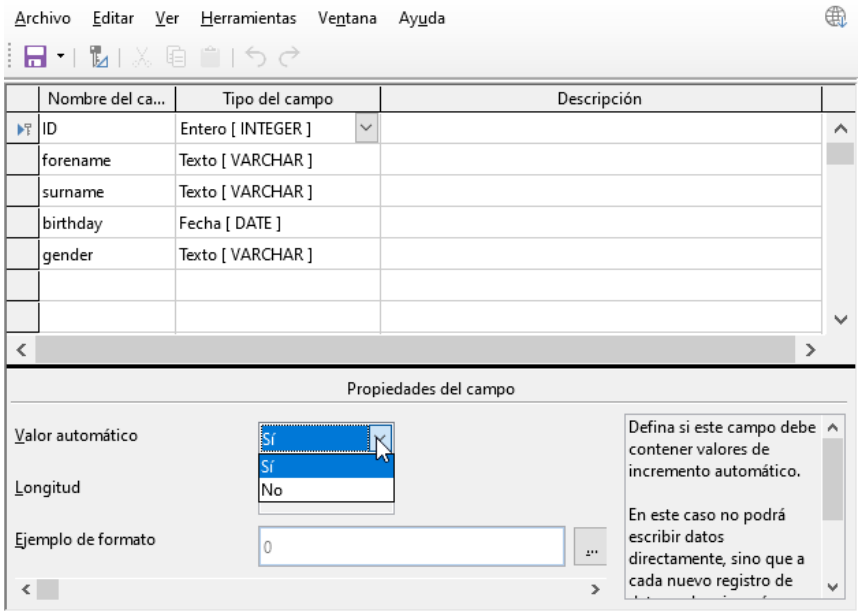
Es deseable que los nombres de campo nombre (*forename*), apellido (*surname*) y fecha de nacimiento sean claros. Además, se agregó un campo llamado *ID*. Este campo luego tomará un valor que es único para cada registro. Un campo clave único es necesario para la base de datos

integrada. De lo contrario, no se pueden introducir registros en la tabla. Este campo clave se denomina clave principal o clave primaria en las bases de datos.

Se podría usar otro campo para esta propiedad. Sin embargo, si, por ejemplo, se usa el apellido solo para esto, dos personas con el mismo apellido no podrían guardarse. En este caso, podría ser útil declarar dos campos juntos en una clave primaria compartida. No hay garantía de que funcione a largo plazo y no lo hará. Así que es preferible la versión simple.



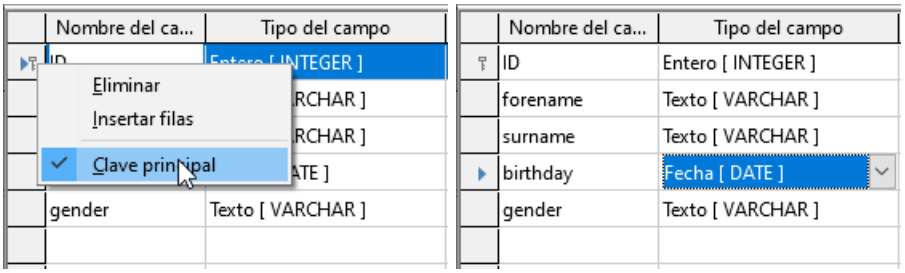
En el segundo paso, seleccione los tipos de campo para los campos ya nombrados de las listas. Establezca el campo *ID* con el tipo de campo *Integer*. Este tipo de campo tiene la ventaja de que HSQLDB incorporado puede proporcionarlo automáticamente con el siguiente entero más alto.



Edite la propiedad de campo del campo ID. Para este campo, active la configuración automática de valores numéricos ascendentes: **Propiedades del campo > Valor automático > Sí**.

Después de seleccionar *Valor automático*, debe aparecer un icono de llave en el encabezado de la fila al salir de la selección del tipo de campo. Esto indica que este campo es la clave principal de la tabla.

Si no se ha seleccionado *Valor automático*, también se puede seleccionar la clave principal en el menú contextual (haga clic con el botón derecho y seleccione clave principal).



Seleccione el *tipo fecha* para el campo Cumpleaños. Esto garantiza que solo se agreguen entradas de fecha válidas. También se usa para ordenar fechas o, por ejemplo, para calcular la edad.

La tabla ahora se puede guardar con el nombre *Starters*. Posteriormente, se pueden introducir datos. La entrada en el campo ID no es necesaria. Se realiza automáticamente cuando se guarda el registro.

Nota

El archivo de base de datos es una carpeta comprimida de archivos individuales. Por lo tanto, almacenar un solo objeto, como una tabla, no se escribe directamente en el archivo de la base de datos. Es por eso que se debe hacer clic en el botón *Guardar* para el archivo de la base de datos después de la creación de tablas, consultas, formularios e informes.

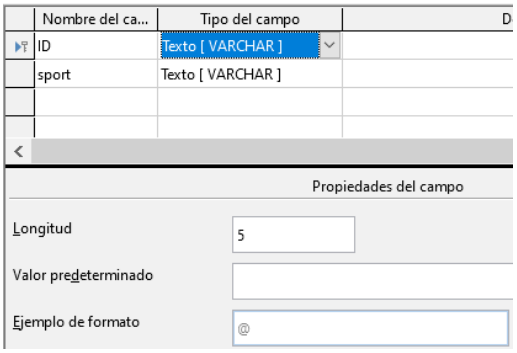
Los datos introducidos se guardan automáticamente solo con salir de la fila de datos,.

Ahora se pueden introducir principiantes (atletas).

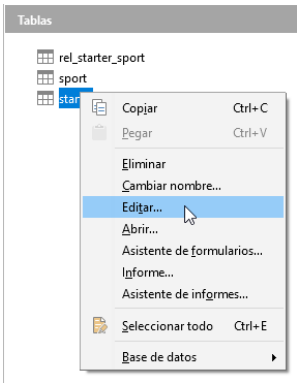
Sin embargo, a primera vista, falta la siguiente información:

- Una lista de deportes en los que los principiantes quieren competir.
- Para las competiciones, la distinción entre principiantes masculinos y femeninos.

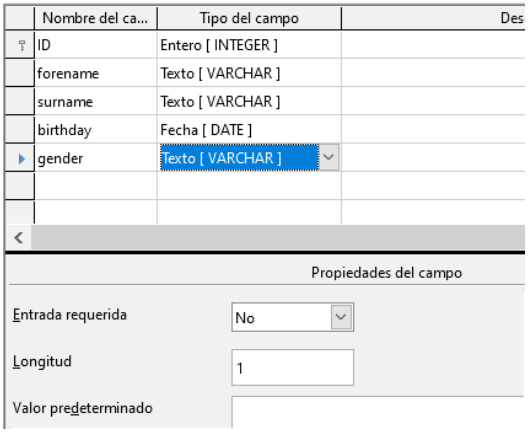
Crearemos una tabla deportiva. Como no hay muchos deportes diferentes, no se selecciona *Valor automático* para la clave principal. En cambio, el tipo de campo se deja como *Texto*, pero está limitado a 5 caracteres. Son suficientes 5 caracteres para encontrar una abreviatura adecuada para los deportes.



Abra la tabla *Principiantes* nuevamente para editar, no para introducir datos, utilizando el menú contextual de la tabla.



Agregue el campo "género" a la tabla. Se puede agregar un nuevo campo diálogo *Diseño de tabla* solo al final de la tabla. También es posible usar SQL para agregar nuevos campos en ciertas posiciones.

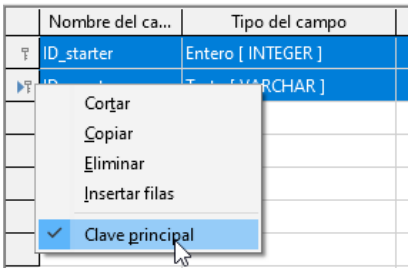


La longitud del texto en este campo está limitada a un carácter, suficiente para "m" y "f" como entrada.

De alguna manera, las dos tablas deben estar vinculadas para que cada titular pueda registrarse en varios deportes y se puedan registrar más titulares para cualquier deporte.

Esto se realiza a través de una tabla en la que se guardan los valores de las dos claves principales de las tablas *Principiantes* y *Sport*.

Dado que solo la combinación de estos campos se guardará en conjunto, estos campos son la clave principal para esta tabla.



Para asignar la clave principal a ambos campos, haga clic en el encabezado de la fila del primer campo, luego presione *Mayús* + clic en el encabezado de la fila del segundo campo; Esto selecciona ambos campos. Haga clic con el botón derecho en cualquier encabezado de fila, luego haga clic en **Clave primaria** en el menú contextual para especificar la clave primaria.

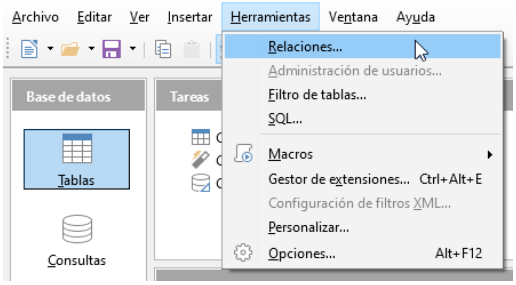
Los valores apropiados se pueden tomar de *Principiantes* y *Sport*, ya que los campos deben coincidir exactamente con los tipos de campo que desea guardar. *ID_starter* debe tener el tipo de campo Integer. *ID_sport* tiene el tipo de campo Texto y está limitado a 5 caracteres también, como el campo *ID* de la tabla *Sport*.

Guarde la tabla como *rel_starter_sport*.

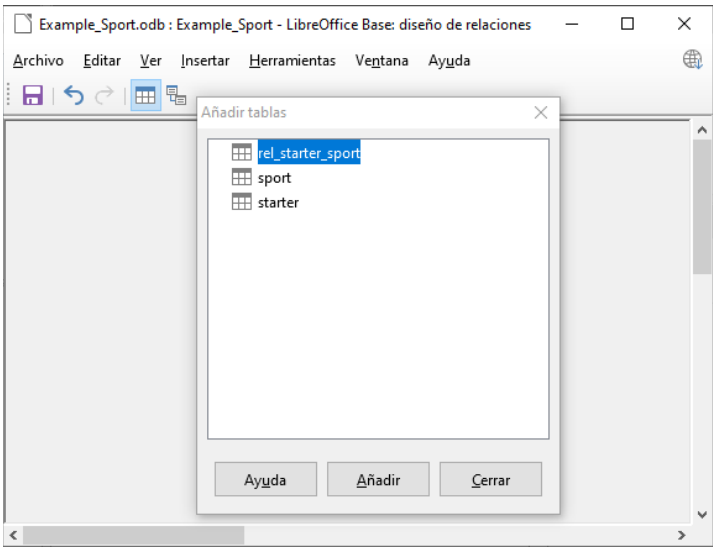
Consejo

Los resultados de una competición también podrían incluirse en esta tabla. Sin embargo, si se llevan a cabo varias competiciones, se debe adjuntar una fecha de carrera a la clave primaria común.

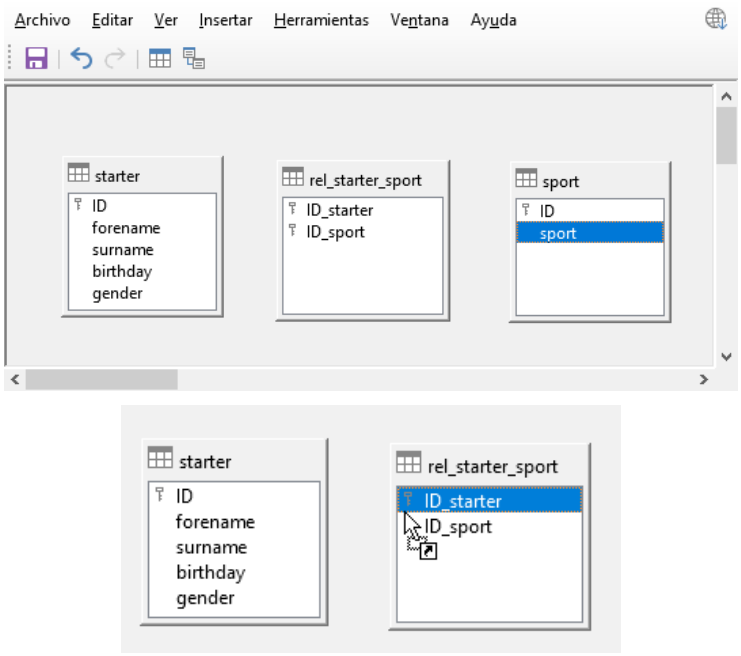
A medida que se completan las tablas, se debe definir una relación entre las tablas. Esto puede evitar*, por ejem*plo, que aparezca un número para un principiante en la tabla *rel_starter_sport* que no figura en la tabla de principiantes. Para abrir la ventana de definición de relaciones haga clic en **Herramientas > Relaciones**.



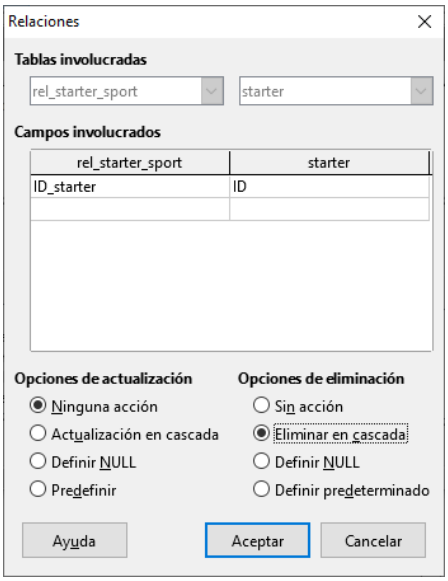
Todas las tablas creadas hasta ahora son necesarias para la definición de la relación. Haga clic en cada tabla individual y haga clic en el botón **Añadir** para agregarlas al diseño de la relación. Luego cierre el diálogo *Añadir tablas*.



Se muestran todos los campos de cada una de las tablas agregadas. Los campos clave principal están marcados con un símbolo de llave. Los rectángulos para las tablas se pueden mover y cambiar de tamaño.



Haga clic en ID en la tabal starter. Mantenga presionado el botón del ratón y mueva el puntero a *ID_starter* en la tabla *rel_starter_sport*. El cursor indica un enlace. Soltar el botón del ratón. Aparecerá el siguiente diálogo para definir la relación:

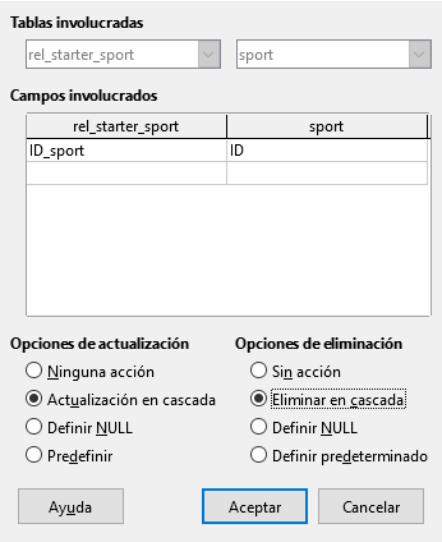


El campo "rel_starter_sport". "ID_sport" no debe cambiarse cuando se cambia "starter"."ID". Este es el valor predeterminado.

"ID" no cambia porque es un campo autoincrementado automáticamente y no se necesita entrada.

El registro en la tabla r*el_starter_sport* debe eliminarse cuando "ID_sport" es igual a "starter"."ID" y "starter"."ID" se elimina. Por lo tanto, si se elimina un principiante de la tabla de *starter*, se eliminarán todos los registros relevantes de la tabla *rel_starter_sport*. Este procedimiento se llama *Eliminar en cascada*

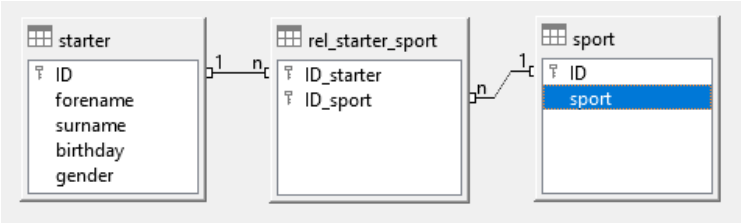
En el siguiente paso, "rel_starter_sport"."ID_sport" y "sport"."ID" se conectan arrastrando el ratón mientras se mantiene presionado el botón izquierdo del ratón.



Aquí también se eliminará un registro cuando se elimine el deporte correspondiente.

Sin embargo, para un cambio de datos en "deporte"."ID", se ha seleccionado otra variante. Si se cambia "deporte"."ID", "rel_starter_sport". "ID_sport" también se cambiará.

Por lo tanto, la abreviatura de un deporte de hasta 5 caracteres se puede ajustar fácilmente, aunque haya muchos registros en la tabla rel_starter_sport. Este procedimiento se llama *Actualización en cascada*.



Los campos ahora están completamente conectados. En los extremos de las conexiones aparece 1:n. Un principiante (1) puede aparecer repetidamente en la tabla *rel_starter_sport* (n). Un deporte (1) también puede aparecer repetidamente en la tabla *rel_starter_sport* (n). Una combinación dada de Starter y Sport puede aparecer en la tabla solo 1 vez.

A partir de dos relaciones 1:n, ahora existe una relación n:m a través de la tabla intermedia rel_starter_sport. Tal diseño de tabla puede ser malo cuando se llena escribiendo contenido en tablas. Requiere que se abran las tres tablas cuando se asigna un titular a un deporte.

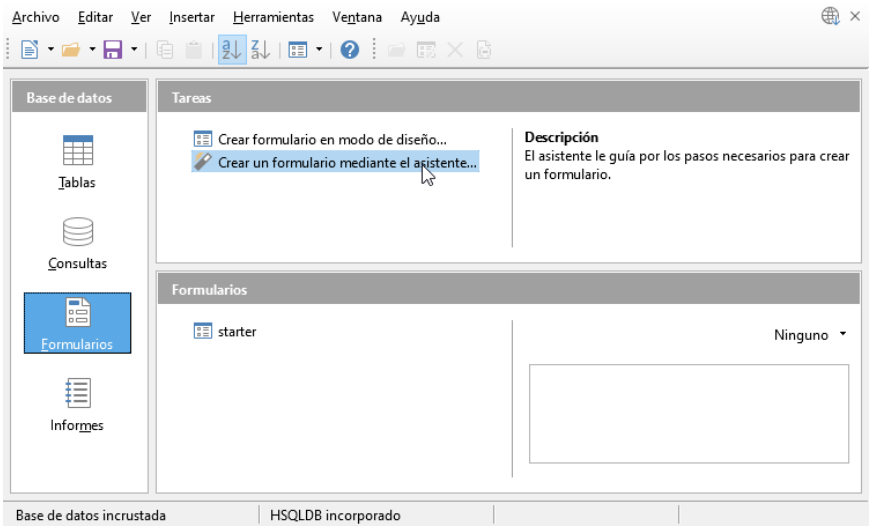
"starter"."ID" debe buscarse en la tabla Starter y transmitirse a *"rel_starter_sport"."ID_starter"* y *"sport"."ID"* debe buscarse en la tabla Sport y transmitirse a *"rel_starter_sport"."ID_sport".*

Esto es muy complicado. Un formulario resuelve esto con más elegancia.

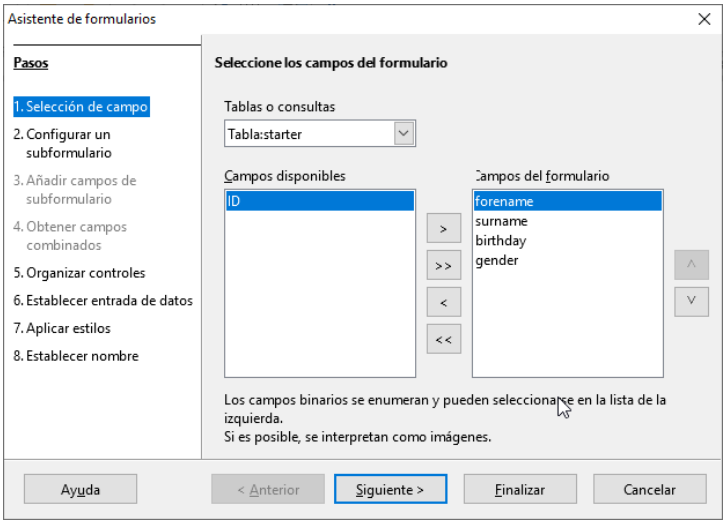
Crear un formulario de entrada de datos

Los formularios se pueden crear directamente en la vista de diseño o mediante el asistente. Incluso las personas experimentadas saben que pueden usar rápidamente el asistente y luego personalizar lo que sea necesario para producir lo que se desea.

Esta es a menudo la mejor forma de ahorrar tiempo.



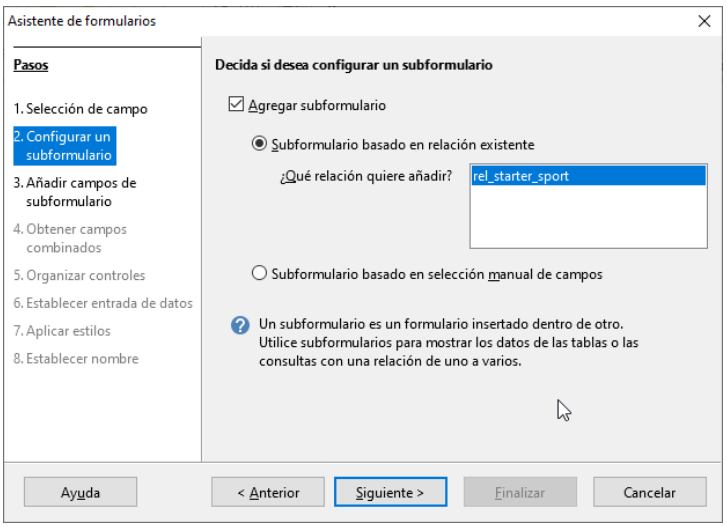
Primero seleccione Formularios en la sección Base de datos. Luego, en la sección *Tareas*, seleccione **Crear un formulario mediante el asistente**.



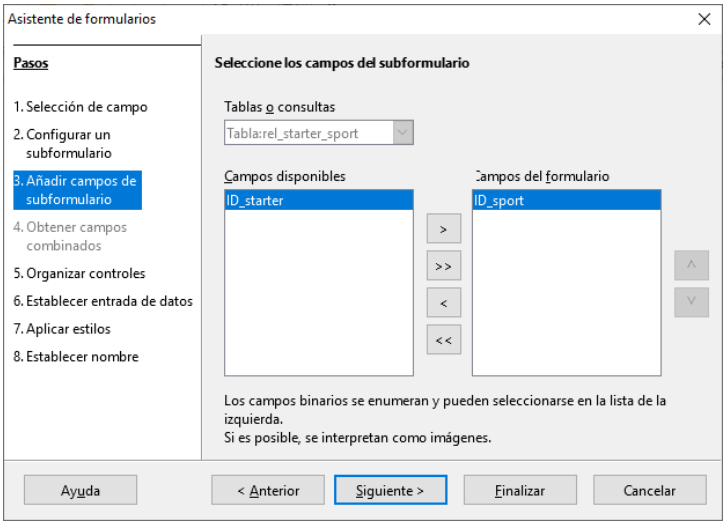
Los datos de la tabla *starter* deben escribirse en el formulario principal. Los datos de la tabla *sport* se cargan directamente con los pocos deportes necesarios y rara vez se actualizan.

Son necesarios todos los campos de la tabla de inicio, excepto el campo de clave principal *ID*. El campo de la clave primaria se llena automáticamente con un valor único.

Seleccione los campos en la lista *Campos disponibles* y use los botones de flecha para moverlos a los *Campos del formulario*. Haga clic en **Siguiente**.

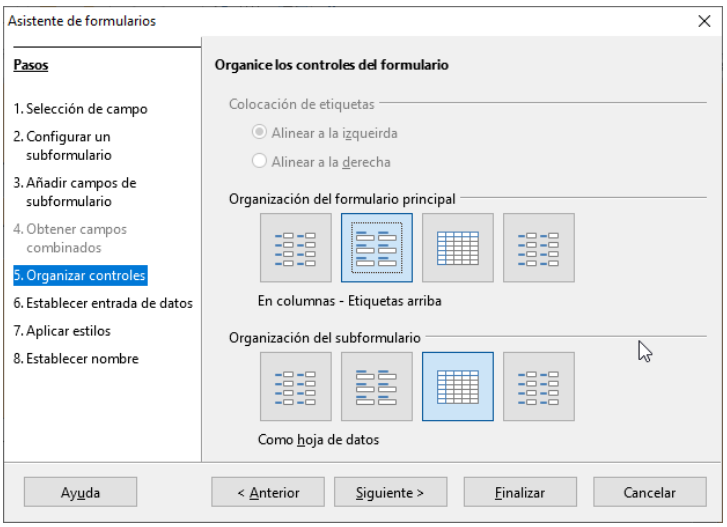


Se debe configurar un subformulario donde se pueda asignar un deporte a un principiante. En el paso 2, seleccione *Agregar subformulario* y *Subformulario basado en relación existente*. Para confirmar la relación definida previamente, seleccione *rel_starter_sport*. Haga clic en **Siguiente**.



Solo se requiere el campo *ID_sport* de la tabla *rel_starter_sport*. La clave principal en la tabla de *starter* proporciona el valor para el campo *ID_sport* para el registro actual mediante la conexión del formulario principal al subformulario.

También se puede ver que este acceso directo ya está regulado en el Paso 4 del asistente: los campos para combinar están inactivos.

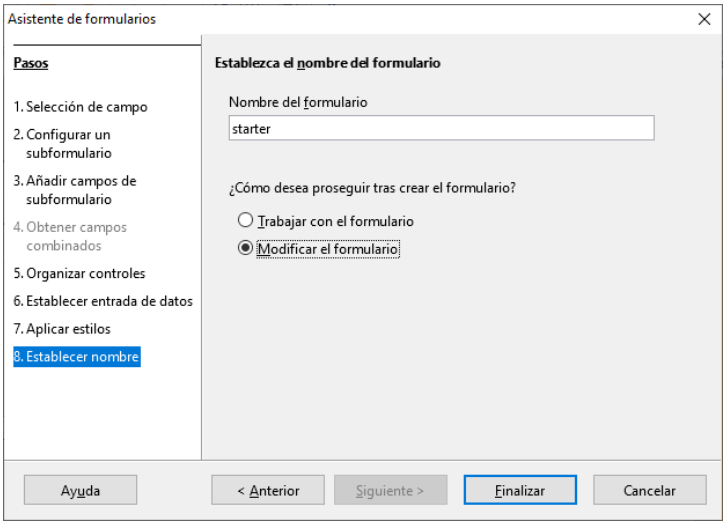


La forma en que se organizan los elementos en el formulario principal y el subformulario en última instancia no importa.

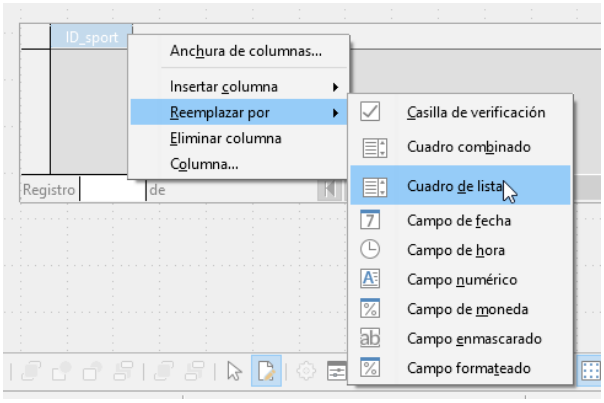
Sin embargo, la asignación de datos también debe ser simple para los inexpertos. En nuestro ejemplo, no debe seleccionarse *Como hoja de datos* para el formulario principal; en su lugar, elegir **En columnas - Etiquetas arriba**. Los campos en el subformulario luego mostrarán todos los deportes de los principiantes, por lo que es mejor dejar la disposición del subformulario como está: **Como hojas de datos**.

El paso 6 (*Establecer entrada de datos*) debe permanecer como está: **El formulario es para mostrar todos los datos**. De modo que es posible una nueva entrada, así como una modificación de los datos existentes.

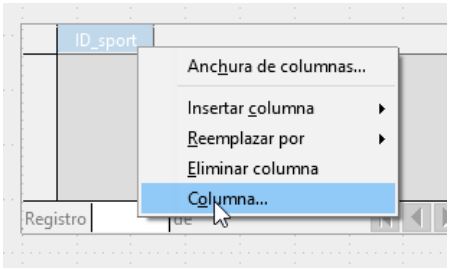
El paso 7 (*Aplicar estilos*) es una cuestión de gustos. Solo hay que tener cuidado: algunos estilos involucran imágenes inesperadas de bajo contraste, especialmente en los campos de control de tabla, por lo que el color de fuente de los campos de la hoja de datos debe reajustarse si es necesario.



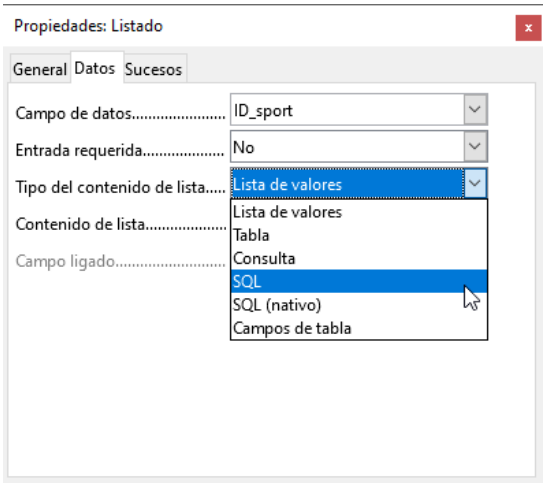
El formulario no funcionará porque el subformulario aún requiere que se ingrese la abreviatura de los deportes. Lo que se necesita es poder seleccionar los deportes después de introducir el nombre completo deseado. Para ello, en el Paso 8 (*Establecer nombre*), asigne al formulario el nombre *starter* y seleccione *Modificar el formulario*. Haga clic en **Finalizar**.



En la hoja de datos, Haga clic derecho en el encabezado de la tabla, *ID_sport*. En el menú contextual, seleccione **Reemplazar con > Cuadro de lista**.

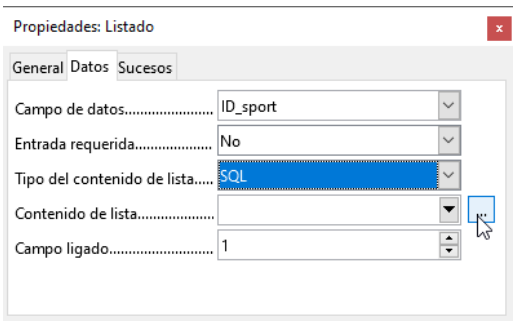


Luego, la lista debe procesarse para que también pueda mostrar los datos correspondientes. Haga clic derecho nuevamente en ID_sport y seleccione **Column**.



Esto abre las propiedades del cuadro de lista seleccionado. Seleccione **Datos > Tipo de contenido de la lista> SQL**. Con la ayuda de SQL (Structured Query Language - lenguaje de consulta estructurado para bases de datos), el campo debe obtener su contenido de la tabla Sport.

Cuando se selecciona SQL, el contenido de la lista tiene un botón de elipse a la derecha (...). Haga clic en este botón para abrir el editor para crear consultas. La consulta que se está creando se unirá y finalmente se guardará en el cuadro de lista.



En el diálogo *Añadir tabla o consulta*, elija la tabla *sport*, luego haga clic en **Añadir** y **Cerrar**.

En la primera columna en la parte inferior del editor de consultas, haga clic en el cuadro junto a Campo y seleccione sport en la lista desplegable.

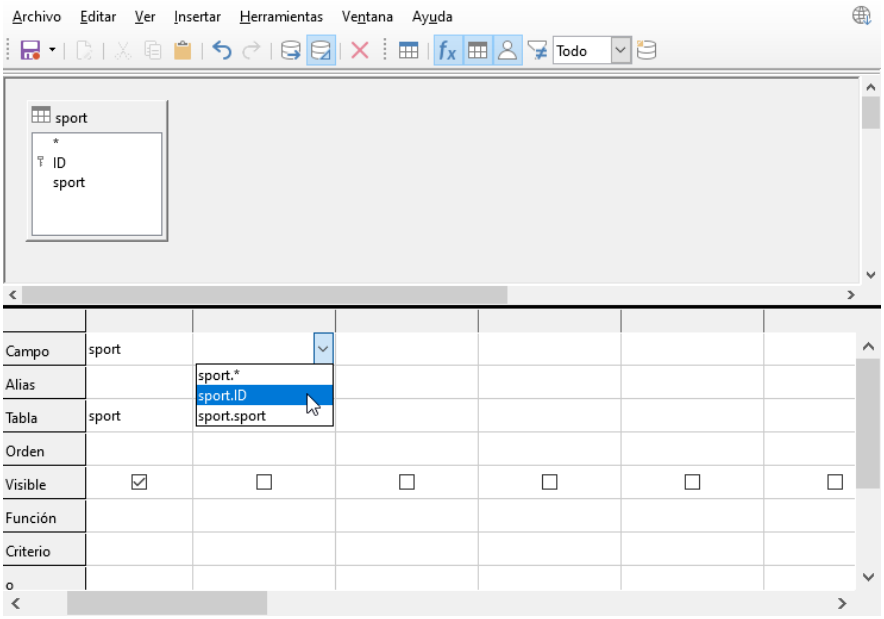
En la segunda columna, seleccione el campo *sport.ID*. Este campo luego pasa su valor a la tabla, que es la fuente de datos del subformulario. Por lo tanto, se muestran las palabras prescritas y se almacenan los atajos apropiados.

Guarde la consulta, que luego se transmite a las propiedades del cuadro de lista. Cierre el editor de consultas.

Ahora el campo Contenido de la lista en el diálogo Propiedades muestra el código SQL que se ha creado en el editor de consultas:

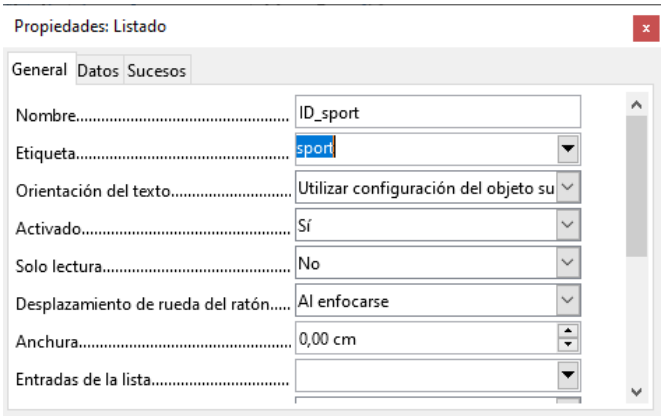
```
SELECT "sport"."ID" FROM "sport"
```

Este código dice: en la tabla "*deporte*", seleccione el campo "*deporte*" y el valor clave asociado *"deporte"."ID"*.

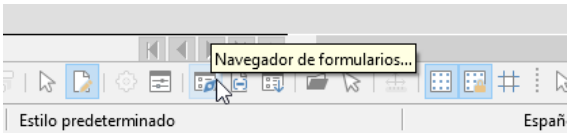


Esta consulta ilustra el mínimo que debe seleccionarse. Por supuesto, se podría ordenar. Guardar abreviaturas de una manea hábil, proporciona una lista útil de deportes almacenados en "ID". Si los registros no se ordenan de una manera específica, la clasificación siempre se realiza por el campo de clave primaria. Para ver los deportes en el cuadro de lista, este contenido debe introducirse primero en la tabla *Sports*.

Ahora necesitamos cambiar la etiqueta para el campo *ID_sport*. Todavía se almacena como ID_sport, pero queremos que sea visible como *sport*. Para ello en la pestaña *General,* seleccione *Etiqueta* y escriba *sport*. Cierre el diálogo *Propiedades*.



Si desea cambiar los nombres de otros campos, se hace mejor a través del *Navegador de formularios*. Si hace clic en los campos, no solo se seleccionan los campos, sino también las etiquetas. A través del asistente, se agruparon. Esto requiere una acción adicional del menú contextual de la selección.



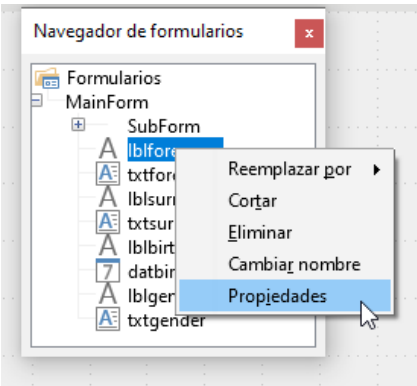
El navegador de formularios se inicia desde la barra de herramientas de diseño de formularios en la parte inferior de la ventana.

Consejo

A veces al procesar los formularios, la barra de herramientas no se abre correctamente al crear formularios. En este caso no se puede ver el navegador de formularios.

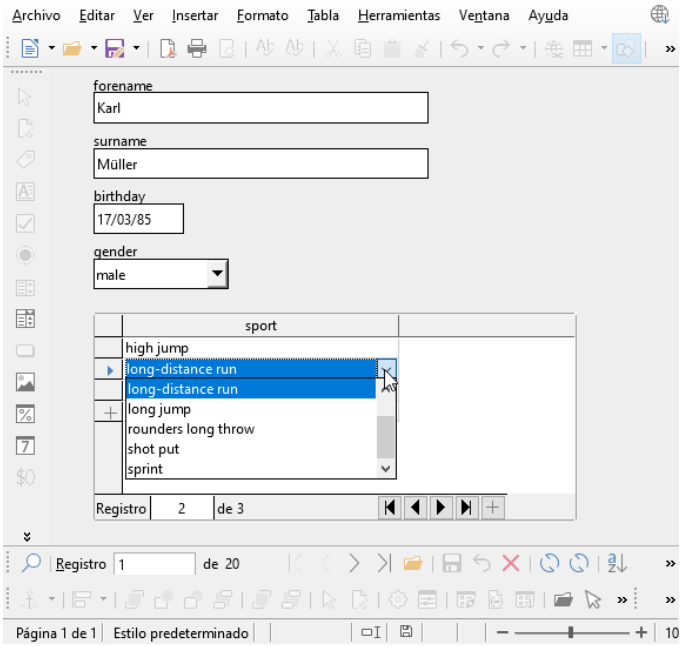
Para mostrar las barras de herramientas, elija **Ver > Barras de herramientas > Diseño de formulario** y **Controles de formulario**. Ahora deberían visualizarse durante la entrada de datos y la vista debería modificarse de acuerdo con ellos.

Con el navegador de formularios, cada campo se puede examinar individualmente. Las propiedades del campo son accesibles en el menú contextual. Cada propiedad se guarda automáticamente al moverse a otra propiedad. Es posible saltar de un campo a otro incluso con el diálogo de propiedades abierto. También se almacena el nivel intermedio respectivo.



Si el diseño ya se ha completado, guardar y cerrar el formulario. Luego guardar de nuevo el archivo de Base.

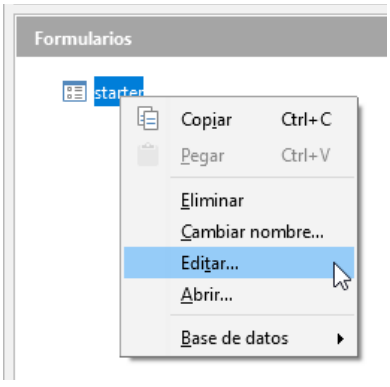
Ahora, si desea ingresar datos en el formulario, podría verse más o menos así. En el siguiente formulario, se ha ingresado un registro para la prueba. Después de ingresar el género, haga clic en el botón Guardar. Por supuesto, se podría incorporar un orden.



Al usar el formulario, notará algunos inconvenientes:

- El subformulario con deportes no es directamente accesible. Si se navega con la tecla tabulador, después de ingresar el género, la pestaña salta directamente al siguiente registro inicial.
- Si está en el subformulario, la barra de navegación muestra el número de registro del subformulario. Sería mejor navegar por el formulario principal solamente.
- El género se ingresa dependiendo de la preferencia del usuario encontrada en la memoria. En la tabla, la longitud del campo se ha limitado a solo 1 carácter. De este modo se garantiza aquí una entrada más segura.

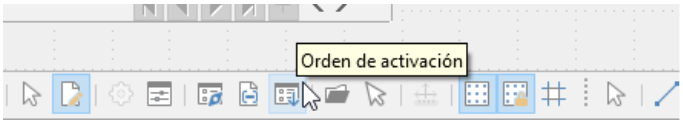
Para evitar estos problemas, abra el formulario para editarlo, no para ingresar datos.



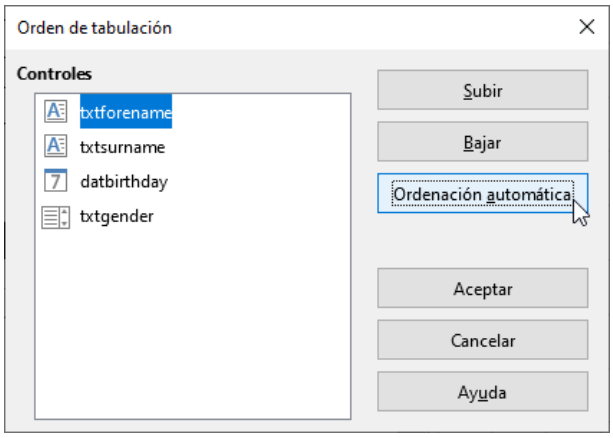
Tabular el subformulario

Para no ir directamente al siguiente registro de inicio usando la tecla *Tab* después de ingresar el género, es necesario modificar la secuencia de activación.

Haga clic en el botón **Orden de activación** en la barra de herramientas *Diseño de formulario*.

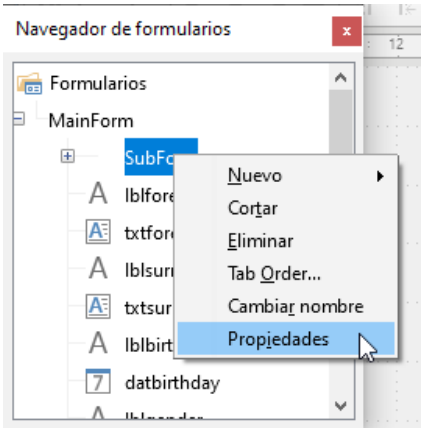


En el diálogo *Orden de Tabulación*, seleccione **Clasificación automática**, lo que afecta no solo a la clasificación de los controles mostrados, sino también al redireccionamiento automático en el subformulario. Aunque no se ve desde el diálogo, se regula de esta manera en segundo plano.

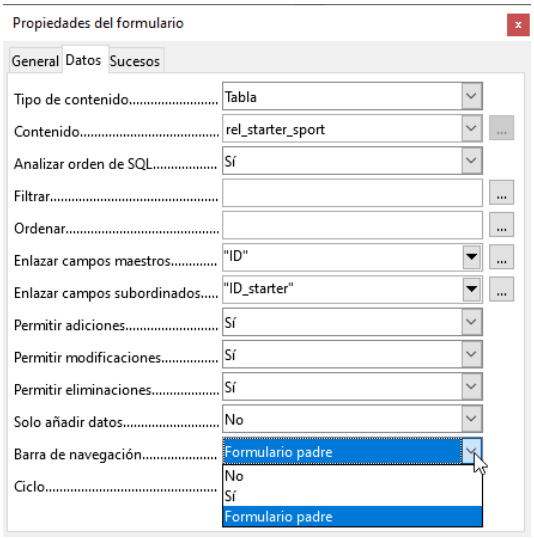


Activar la barra de navegación del formulario principal en el subformulario

Abrir el *Navegador de formularios* para ver las propiedades del subformulario.



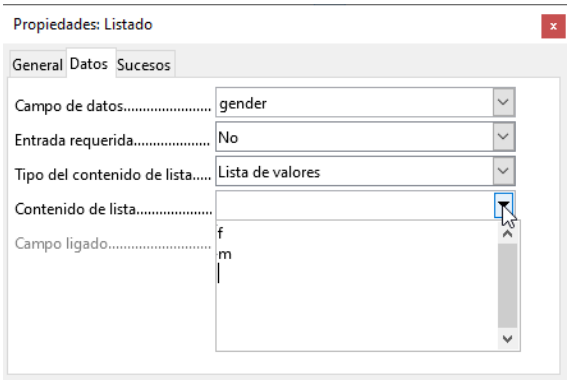
En **Datos > Barra de navegación**, cambie el valor de *Sí* por el de *Formulario padre*. La barra de herramientas de navegación siempre mostrará ahora el número del registro de la tabla de inicio.



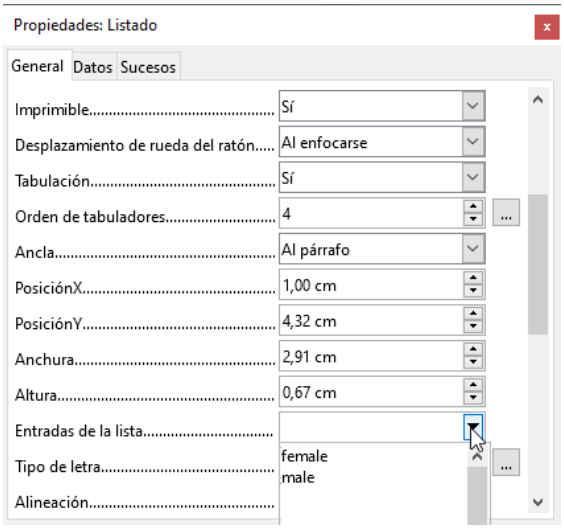
Restringir la entrada a un control

El control para limitar la entrada de valores concretos, no puede ser un simple cuadro de texto. En el formulario principal, la solución se puede encontrar ingresando el *género* en un *Cuadro de grupo*. Otra solución es presentar las opciones en un *Cuadro de lista*.

Para ello, resaltar el campo de género *gender* en el *Navegador*. Haga clic derecho para abrir el menú contextual. Seleccione **Reemplazar > Cuadro de lista**. Las propiedades se seleccionarán usando el menú contextual.



En *Datos* aparece predeterminado **Tipo de contenido de la lista > Lista de valores**. En **Contenido de la lista**, ingresar los caracteres **f** y **m** uno debajo del otro (usando *Mayús* + *Intro*). Estas abreviaturas son los datos que se darán a la tabla de inicio.



En **General > Entradas de la lista**, especificar lo que se mostrará en el cuadro de lista. Puede ser **f** y **m**, o también puede ser *female* y *male* como se muestra en la ilustración. En cualquier caso, debe tener el mismo orden que los elementos en la lista de *Datos*. Seleccione **Sí** para la propiedad *Desplegable* que se encuentra más abajo en la pestaña *General*.

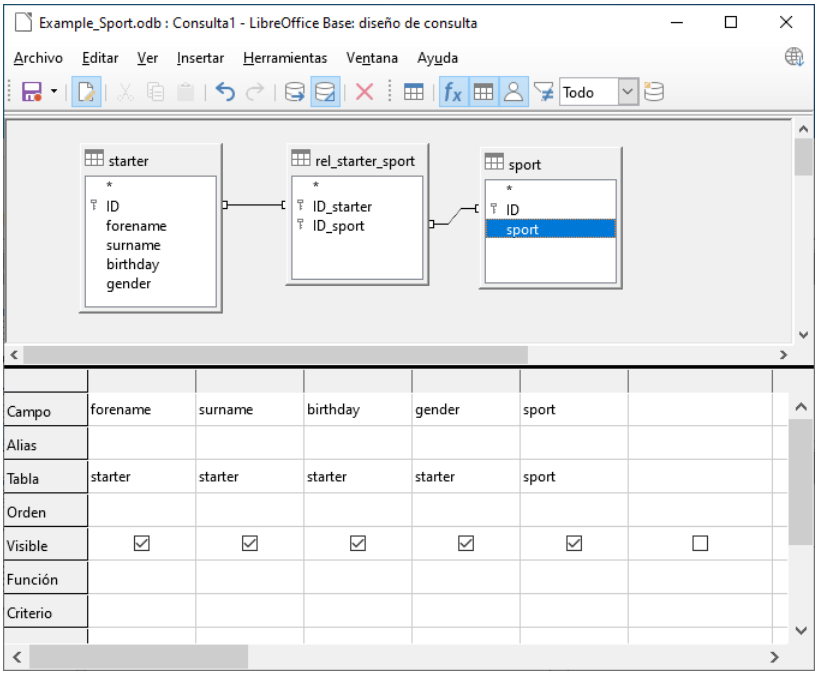
Los inconvenientes más acusados hasta ahora han sido eliminados. Ahora, se debe guardar nuevamente el formulario y el archivo de la base de datos. La entrada para *starters* masculinos y femeninos puede comenzar, así como su asignación a los deportes.

Esto es útil para el siguiente paso: los registros deben ingresarse solo una vez. Asegúrese de que los *starters* también puedan competir entre sí por edad y por deporte. De lo contrario, las consultas e informes posteriores no tienen sentido.

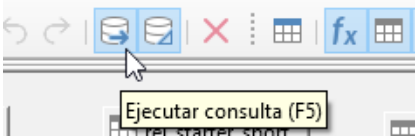
Crear una consulta

En una consulta se puede agrupar el contenido de varias tablas. Cada uno de los *starters* debe mostrarse con los deportes en los que han ingresado.

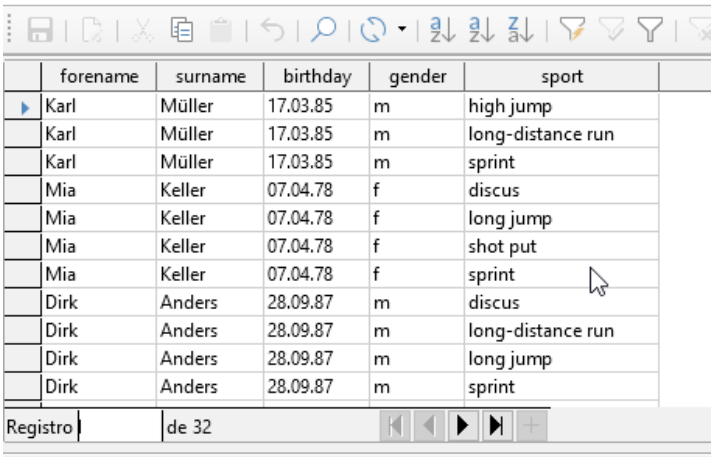
En *Consultas,* haga clic en **Crear consulta en vista de diseño**. Aparecerá un diálogo que enumera las tablas de las que seleccionaremos todas las tablas para la consulta. Quedará claro, cuando *se seleccione* la tabla *rel_starter_sport* como la segunda tabla, se reconocerán también las conexiones.



Los campos que se deben mostrar en la consulta se pueden agregar haciendo doble clic en los nombres de campo en la tabla o seleccionando la fila *Campo*. Aparecerá una lista desplegable, con los nombres de los campos y de sus tablas correspondientes ordenados por el nombre de la tabla. Para hacer coincidir los campos de las tablas correctamente con sus tablas, en las consultas se etiquetan como "nombre de la tabla"."nombre del campo". Si se usa "*" en lugar del nombre del campo, se mostrarán todos los campos de la tabla correspondiente.



En principio, se debe ejecutar una consulta antes de guardar para ver si realmente produce el resultado deseado. Para ello, haga clic en el botón **Ejecutar consulta**, que se muestra arriba, o presione *F5* o elija **Editar > Ejecutar consulta**.



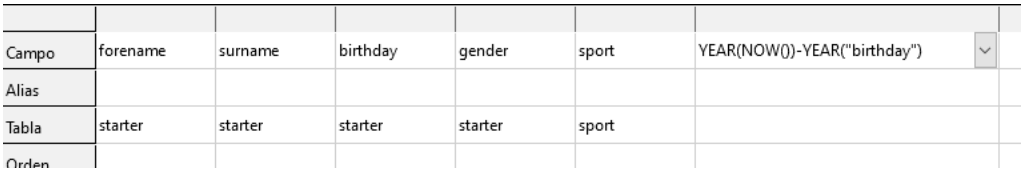
forename	surname	birthday	gender	sport
Karl	Müller	17.03.85	m	high jump
Karl	Müller	17.03.85	m	long-distance run
Karl	Müller	17.03.85	m	sprint
Mia	Keller	07.04.78	f	discus
Mia	Keller	07.04.78	f	long jump
Mia	Keller	07.04.78	f	shot put
Mia	Keller	07.04.78	f	sprint
Dirk	Anders	28.09.87	m	discus
Dirk	Anders	28.09.87	m	long-distance run
Dirk	Anders	28.09.87	m	long jump
Dirk	Anders	28.09.87	m	sprint

Registro de 32

La consulta mostrará todas las combinaciones de *starters* y *sports*. Los *starters* tendrán tantos registros como deportes tengan. Los *starters* sin deportes no aparecerán.

Para reunir diferentes grupos de edad en una competición el año de nacimiento es decisivo. Depende de la edad que tenga una persona en ese año.

El año de nacimiento se puede usar con diferentes funciones. Por ejemplo, considere los siguientes datos:

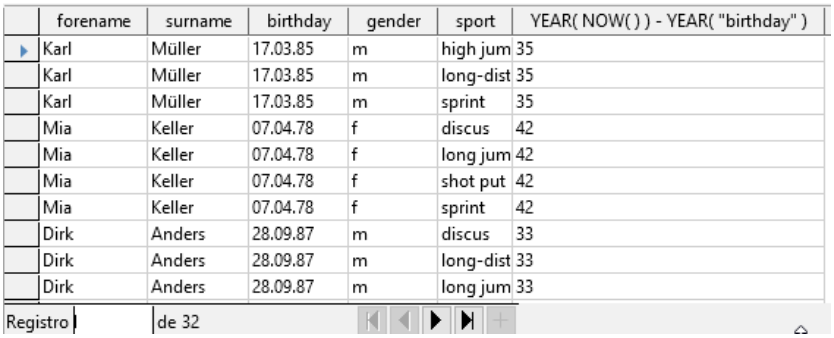


Campo	forename	surname	birthday	gender	sport	YEAR(NOW())-YEAR("birthday")
Alias						
Tabla	starter	starter	starter	starter	sport	
Orden						

```
YEAR(NOW())-YEAR("birthday")
```

`NOW()` significa "ahora", por ejemplo, la fecha u hora actual. El año actual es **`YEAR(NOW())`** y **`YEAR("birthday")`** selecciona el año de nacimiento. La diferencia entre ambos indica la edad que la persona tiene o tendrá en el año en curso.

Estas y otras funciones que funcionan con el HSQLDB incorporado se describen en el apéndice de este libro.

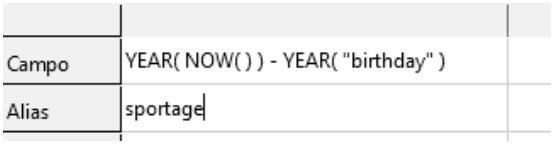


forename	surname	birthday	gender	sport	YEAR(NOW( )) - YEAR("birthday")
Karl	Müller	17.03.85	m	high jum	35
Karl	Müller	17.03.85	m	long-dist	35
Karl	Müller	17.03.85	m	sprint	35
Mia	Keller	07.04.78	f	discus	42
Mia	Keller	07.04.78	f	long jum	42
Mia	Keller	07.04.78	f	shot put	42
Mia	Keller	07.04.78	f	sprint	42
Dirk	Anders	28.09.87	m	discus	33
Dirk	Anders	28.09.87	m	long-dist	33
Dirk	Anders	28.09.87	m	long jum	33

Registro | de 32

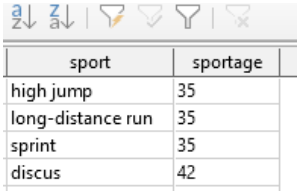
Cuando la consulta se ejecute durante un año determinado (por ejemplo, 2020), aparecerán los cálculos correspondientes para ese mismo año.

Todo el código que se haya ingresado en el campo también aparecerá en el encabezado de la columna. Esto se soluciona ingresando un alias para el código bajo el cual aparecerá el resultado.



Campo	YEAR(NOW( )) - YEAR("birthday")
Alias	sportage

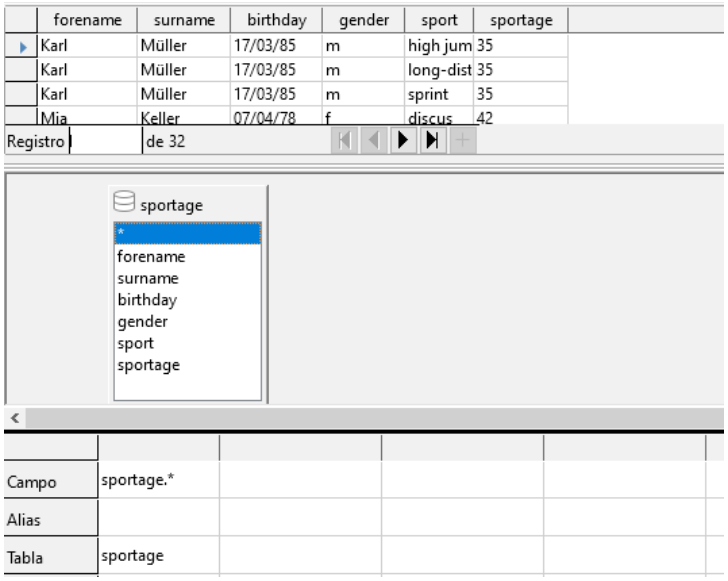
En la fila *Alias*, se ingresa el término "*sportage*". No se debe usar esta edad para nadie que aún no haya cumplido años en este año.



sport	sportage
high jump	35
long-distance run	35
sprint	35
discus	42

Si la consulta se ejecuta nuevamente, el encabezado de la columna ya no contendrá el código sino el término *sportage*.

Por lo tanto, la consulta debe guardarse con el nombre *sportage* y no confundirse por usar el código como nombre. Esta consulta se utilizará como base para la siguiente consulta, en la que *sportage* asignará un *grupo_de_edad*.

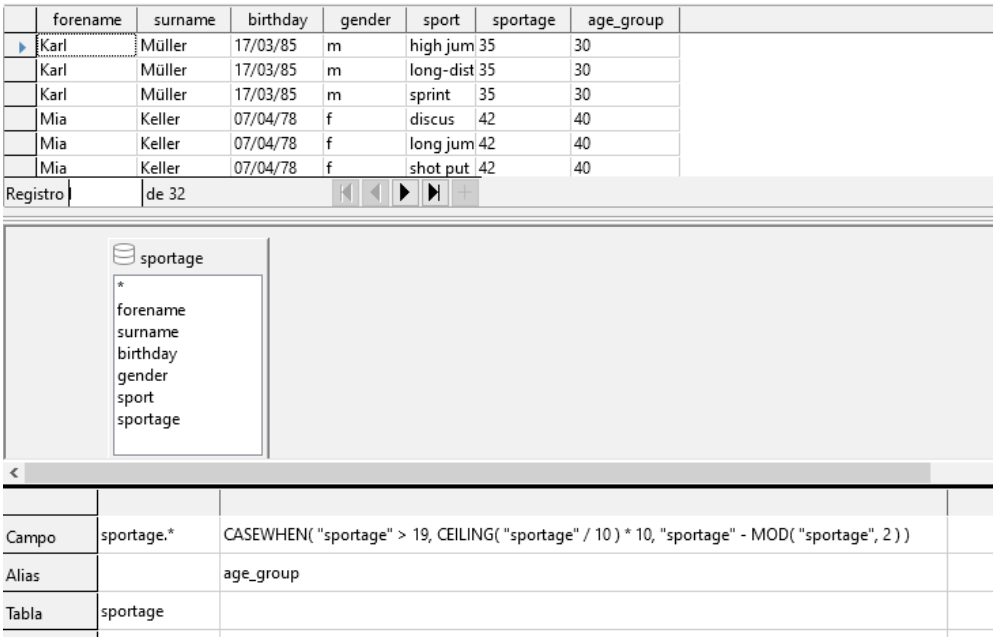


La consulta *sportage* fue elegida como base para la segunda consulta. Antes del término *sportage* en el contenedor de la tabla hay un icono diferente al que se ve para las tablas de la primera consulta. Este símbolo indica que el cometido de esta consulta es una consulta y no una tabla.

Haga doble clic en * o seleccione *sportage.** para seleccionar todos los campos, así la siguiente consulta devolverá el mismo resultado, pero la fórmula ya no será visible.

Acceda al campo *sportage* que determinará en qué grupo de edad participará la persona respectiva.

Para que el cálculo no sea demasiado complejo, se deben modificar las siguientes clasificaciones: para los menores de 20 años, los *starters* se dividen en grupos de edad que contengan dos edades por grupo a partir de 0. A partir de 20 años se dividirán en grupos que contengan tramos de diez años de edad cada uno, por ejemplo, 20–29 años.



Tales fórmulas ya no pertenecen realmente al grupo de habilidades de nivel principiante. Es posible una asignación de edades más simple con la ayuda del siguiente informe. Para asignaciones más sofisticadas, consulte el Apéndice de este libro.

```
CASEWHEN( "sportage" > 19, CEILING( "sportage" / 10 ) * 10, "sportage" -
MOD( "sportage", 2 ) )
```

CASEWHEN (*condición para probar, valor si es verdadero, valor si es falso*).

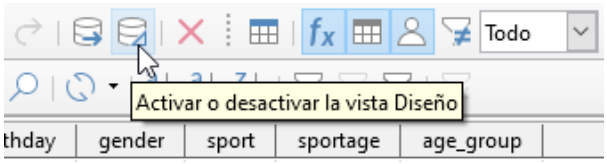
En inglés, esto significa:

- Si *sportage* tiene más de 19 años, se utilizará para calcular la siguiente edad más baja que termina en cero para obtener el *age_group*.
- Si *sportage* es menor de 19 años; se calcula MOD ("*sportage*", 2) dividiendo *la edad del atleta* por 2 y restando el resto (que será 0 o 1) del valor de *sportage* para obtener el *age_group*.

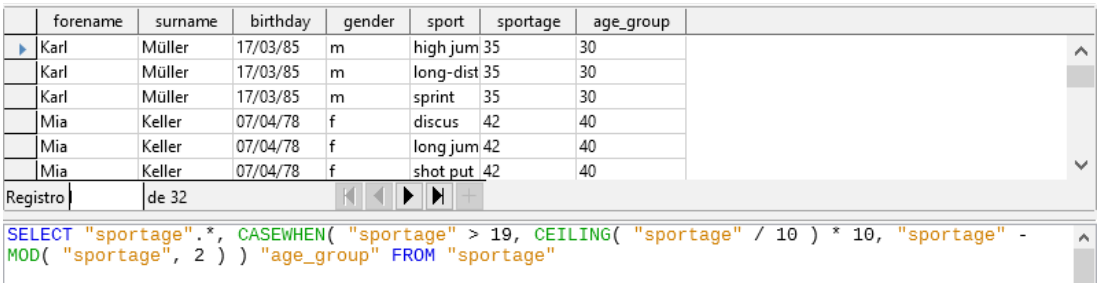
Todos los *starters* mayores de 19 años se asignan a un grupo de edad por cada diez años. Todos los *starters* hasta los 19 años se asignan a un grupo de edad para cada dos años.

A esta fórmula se le asignó nuevamente un alias, en este caso el alias es *age_group*.

Guarde la consulta con el nombre *registration*.



Desactivar la vista de diseño no es realmente necesario, ya que todas las entradas son posibles en la vista de diseño sin mayores problemas. Sin embargo, profundizar en el código en una consulta nunca puede perjudicar. A veces, hay expresiones SQL que encajan mal en la vista de diseño o que no son posibles allí, en esos casos se usa la entrada directa del código SQL.



forename	surname	birthday	gender	sport	sportage	age_group
Karl	Müller	17/03/85	m	high jum	35	30
Karl	Müller	17/03/85	m	long-dist	35	30
Karl	Müller	17/03/85	m	sprint	35	30
Mia	Keller	07/04/78	f	discus	42	40
Mia	Keller	07/04/78	f	long jum	42	40
Mia	Keller	07/04/78	f	shot put	42	40

Registro de 32

```
SELECT "sportage".*, CASEWHEN( "sportage" > 19, CEILING( "sportage" / 10 ) * 10, "sportage" -
MOD( "sportage", 2 ) ) "age_group" FROM "sportage"
```

Consejo

Los campos de código y tablas se encierran entre comillas dobles y se muestran en color ocre. Los términos del código SQL están en azul, las funciones de la base de datos en verde.

```
SELECT "sportage".*, CASEWHEN( "sportage" > 19, CEILING( "sportage" /
10 ) * 10,"sportage" - MOD( "sportage", 2 ) ) AS "age_group" FROM
"sportage"
```

Las partes de la fórmula ya se mencionaron. Aquí ahora está la estructura:

```
SELECT "sportage".*,... AS "age_group" FROM "sportage"
```

La consulta selecciona todos los *registros* de la tabla *sportage* y además lo que determine la fórmula. Lo que está determinado por la fórmula, se conoce como "*age_group*".

El código no distingue entre tablas y consultas como origen de los datos. Por eso, solo funciona en la interfaz gráfica de usuario de Base. Una consulta no puede tener el mismo nombre que una tabla; una tabla no puede tener el mismo nombre que una consulta.

Crear un informe

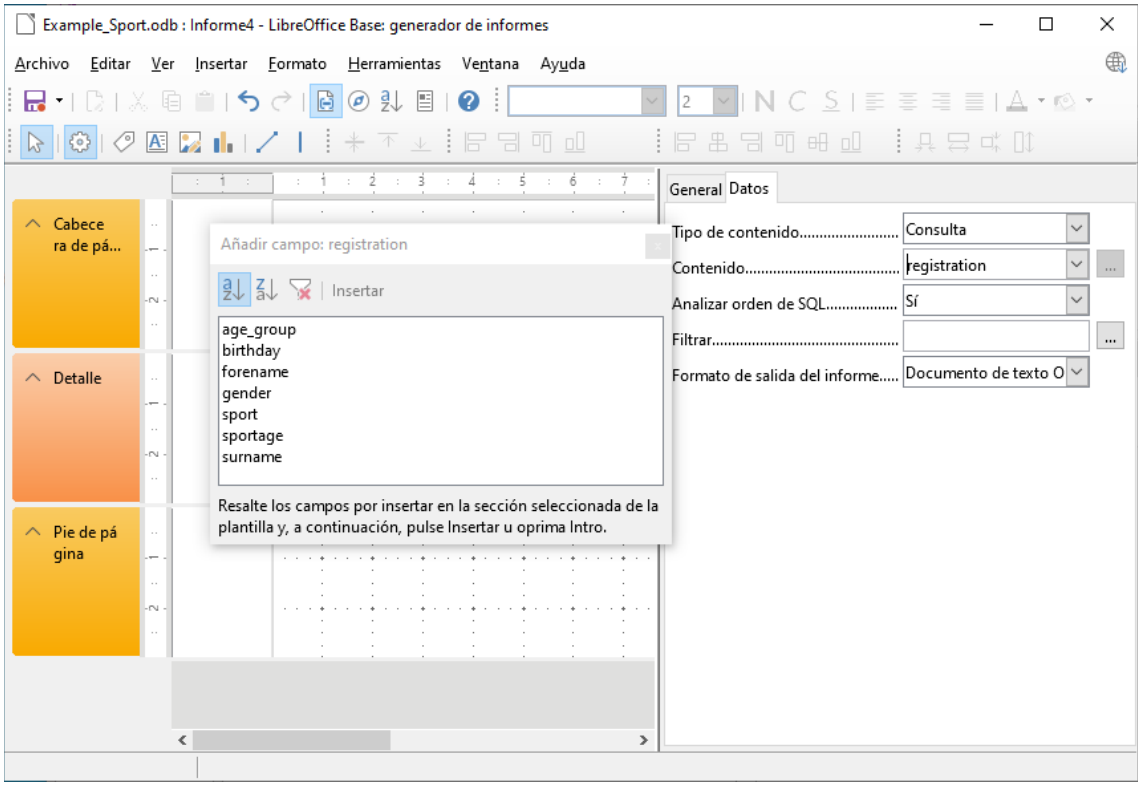
Use **Informes > Crear informe en modo de diseño** para crear un informe con una lista de *starters*, ordenados por deporte, género y grupo de edad.

Al iniciar el editor, el diálogo *Agregar campo* aparece primero en primer plano. Use este diálogo para tomar campos de la tabla ordenada alfabéticamente como base. La consulta debe elegirse como fuente de datos.

En la ventana del informe, el lado derecho muestra una descripción general de las propiedades del objeto actualmente activo.

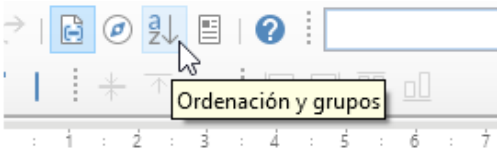
Si no está visible, seleccione **Ver > Propiedades**.

En Propiedades, seleccione **Datos > Tipo de contenido > consulta** . Para la propiedad Contenido, seleccione la consulta *Registro*, que fue la consulta de resumen final.



El generador de informes muestra todos los campos de la consulta seleccionada.

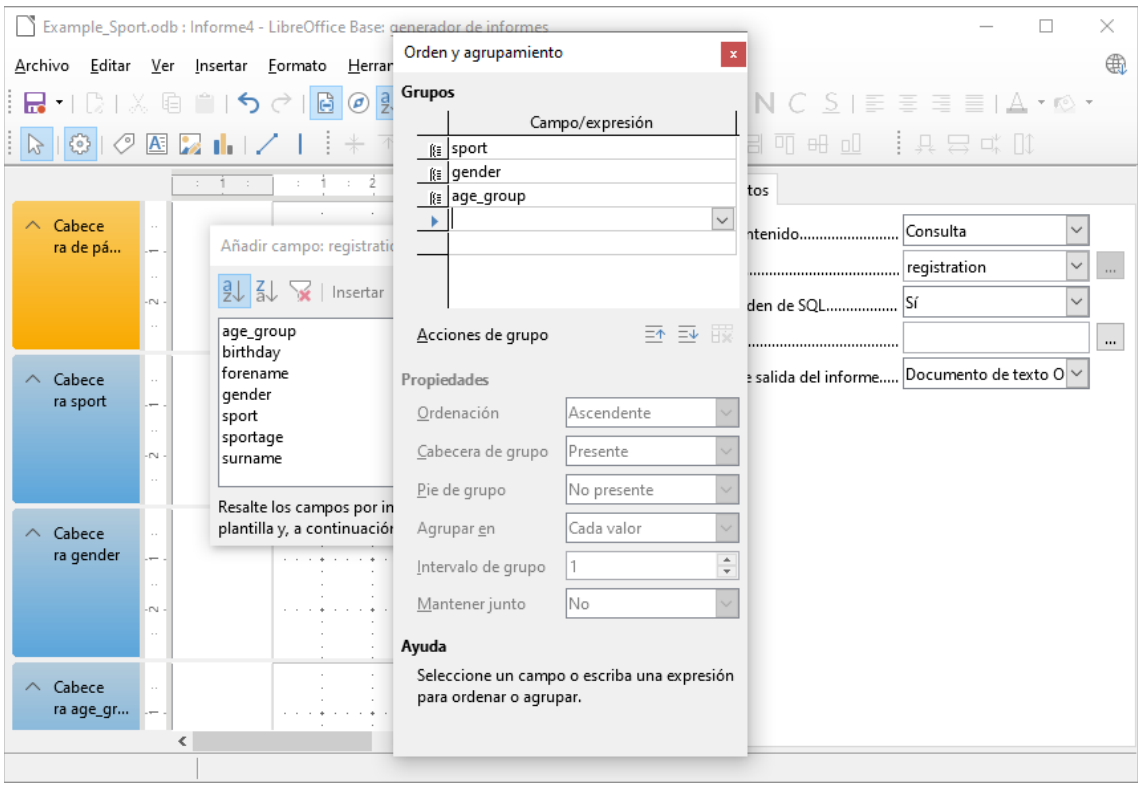
En el siguiente paso, abra el diálogo *Ordenar y agrupar*, utilizando el botón de la barra de herramientas o seleccionando **Ver > Ordenar y agrupar**.



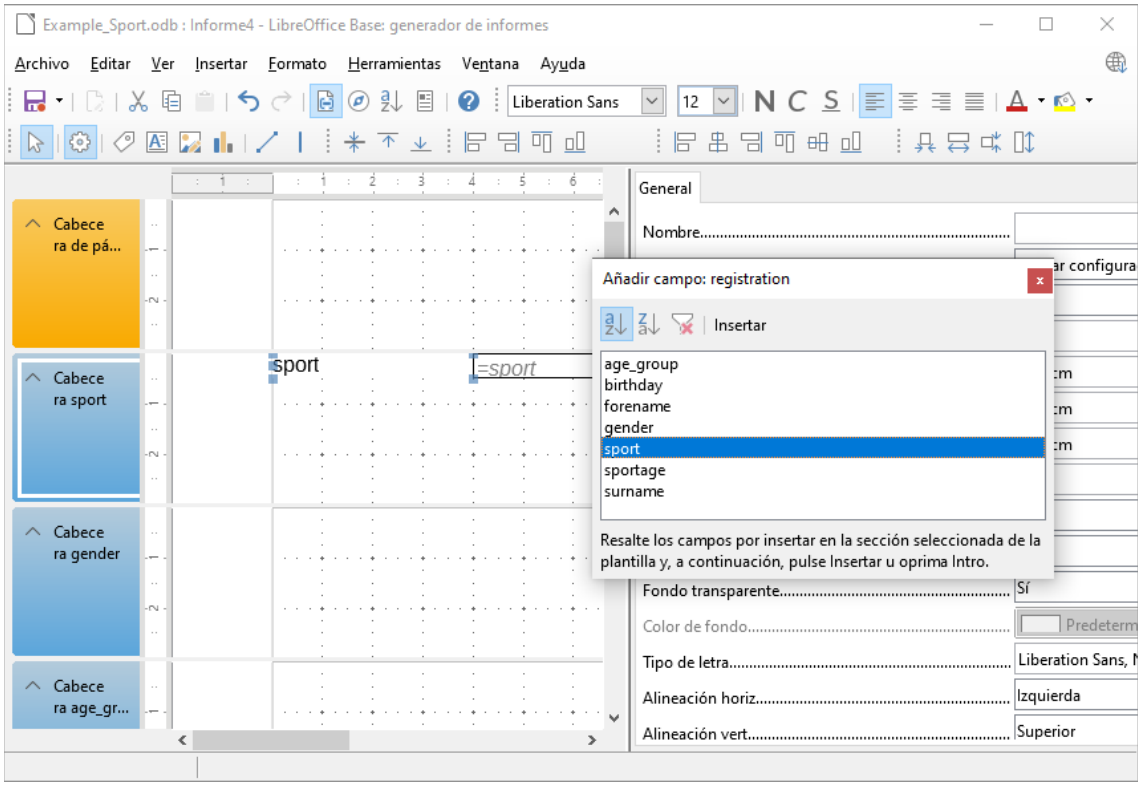
En el diálogo *Ordenación y grupos*, seleccione los campos deporte, género y age_group.

Aparecerá un encabezado de grupo en el lado izquierdo del informe para ordenar cada selección de ordenación. Use la configuración predeterminada para las propiedades de las selecciones de clasificación y agrupación.

Cierre el diálogo *Ordenación y grupos*.



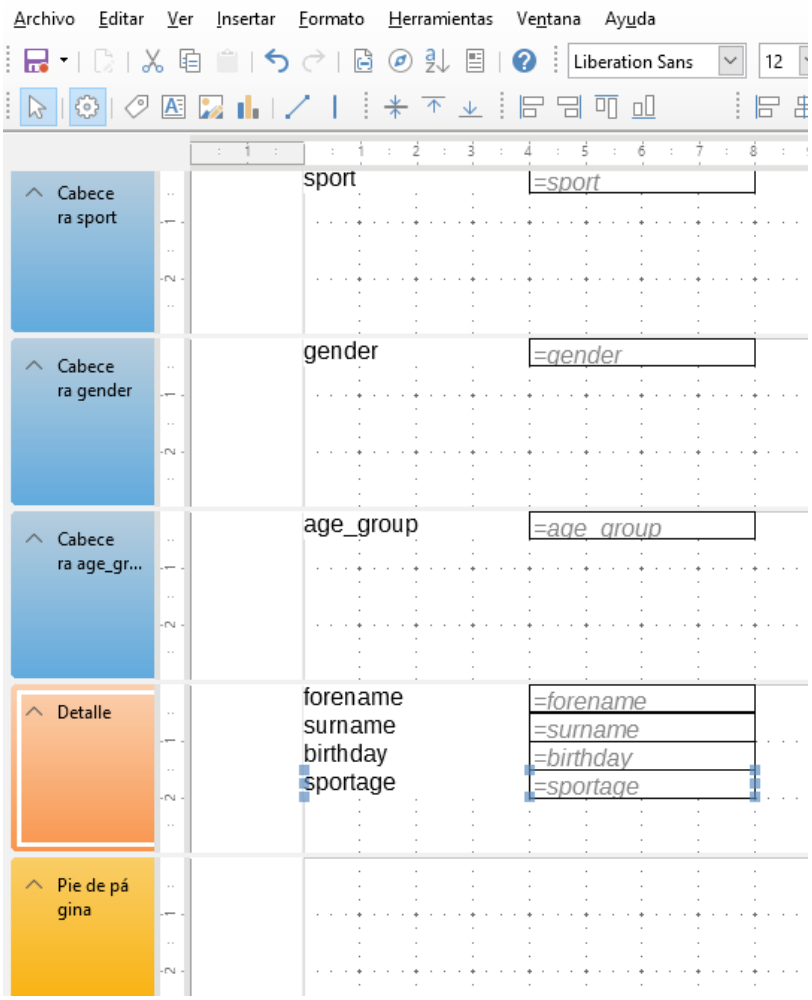
Seleccione el encabezado del grupo, *Cabecera sport*. La selección se indicará con un borde blanco en el encabezado. Seleccione el botón **Añadir un campo** para mostrar el diálogo *Añadir campo.* Haga clic en el campo *sport* para insertar un campo de etiqueta y un campo de texto que mostrará el contenido del campo de *starters*.



Por el mismo método, asignar los campos *gender* y *age_group* a los encabezados de grupo apropiados.

Inserte todos los campos restantes en el área Detalle.

El borrador del informe ahora debería verse así:



Guarde el informe. El nombre podría ser algo así como *Lista de principiantes*.

Luego guarde el propio archivo de la base de datos, de lo contrario, el informe se almacenará solo temporalmente.

Nota

El Generador de informes, a menudo falla cuando se diseña un informe debido a la inestabilidad del programa. Es importante guardar tanto el informe como el archivo de la base de datos.

Afortunadamente, la ejecución posterior de un informe no se ve afectada por estas inestabilidades.

Si este informe se ha ejecutado con los datos apropiados, el resultado aparecerá como sigue:

sport	discus
gender	f
age_group	30.0
forename	Klothilde
surname	zu Guttenstein
birthday	07/03/81
sportage	34
forename	Dorle
surname	van Hoge
birthday	12/23/84
sportage	31

El comienzo del informe muestra a dos mujeres principiantes que desean ingresar al deporte de disco y pertenecen al grupo de edad 30 años.

Al ejecutar el informe, aparecen algunos defectos de diseño:

- Las distancias entre los encabezados de grupo y el contenido en la sección Detalle son demasiado grandes.
- El género está indicado por *m* y *f*. Sería mejor nombres como *Muje*res y *Hombres*.
- Probablemente debido a la división en la consulta, el campo *age_group* contiene números con un decimal.
- Los campos de etiqueta de la sección Detalle (*forname*, *surname*, ...) sería mejor colocarlos horizontalmente como encabezados de tabla debajo de la etiqueta *age_group* y el campo en el encabezado age_group.

Establecer las distancias entre los campos del informe

Para reducir las distancias verticales entre los campos, usar el ratón para arrastrar el borde inferior del encabezado *age_group* al final de los encabezados de la tabla. También puede resaltar una sección y regular la altura del encabezado del grupo en Propiedades. No se debe etiquetar ningún campo, sino solo el encabezado del grupo.

No es posible que una sección sea más pequeña que las etiquetas y los campos que contiene.

El área de encabezado *age_group* debe hacerse lo suficientemente grande como para contener los campos de etiqueta de la sección Detalle.

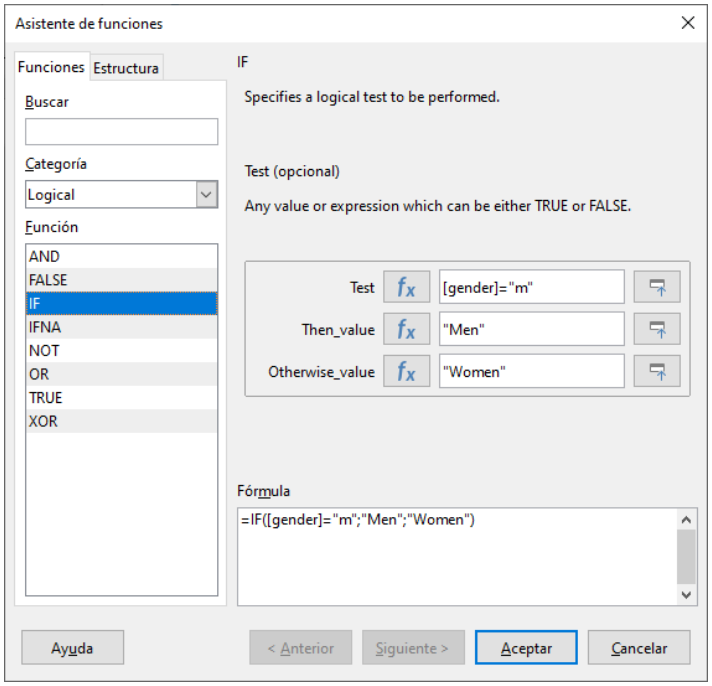
Tenga en cuenta al seleccionar las distancias que el siguiente grupo no aparezca demasiado cerca del grupo anterior. La etiqueta y los cuadros de texto también pueden necesitar una distancia a la parte superior de la sección que los contenga. Si no se desea esta distancia hasta la parte superior, se pueden mostrar pies de página de grupo en lugar de encabezados para proporcionar esta distancia.

Dicha preferencia para cada grupo se puede obtener en **Ver > Ordenación y grupos**.

Modificar el contenido de un campo de texto mediante una fórmula

La designación del género en la tabla no es suficiente para las demandas de una lista de *starters*. Se puede cambiar el nombre del campo en la consulta, pero como la consulta ya se ha creado, se pueden usar funciones en el informe.

Resaltar el campo de texto "= género". En las *Propiedades* en el lado derecho de la pestaña Generador de informes, seleccione **Datos > Campo de datos**. Haga clic en el botón con la elipse (...). Aparecerá el *Asistente de funciones*.



En **Categoría**, seleccione **Logical** y luego haga doble clic en **IF**. La predicción en esta función, que se basa en otras funciones, tiene que desactivarse.

Ingresar el valor de prueba. Este es el campo de la consulta desde la que se leen los datos y se colocan entre corchetes. Los textos deben estar entre comillas dobles. Cuando género tenga el valor **m**, se mostrará **Men** en el campo del informe. Si no hay **m**, aparecerá **Women**.

Haga clic en **OK** para confirmar la entrada.

El contenido del campo que se muestra será modificado en consecuencia.

Cambiar el formato de un campo de texto

El campo que recibe el contenido de la base de datos se identifica en el informe como un cuadro de texto, pero puede formatearse como los campos en las tablas de Writer o Calc.

Resaltar el campo "= age_group".

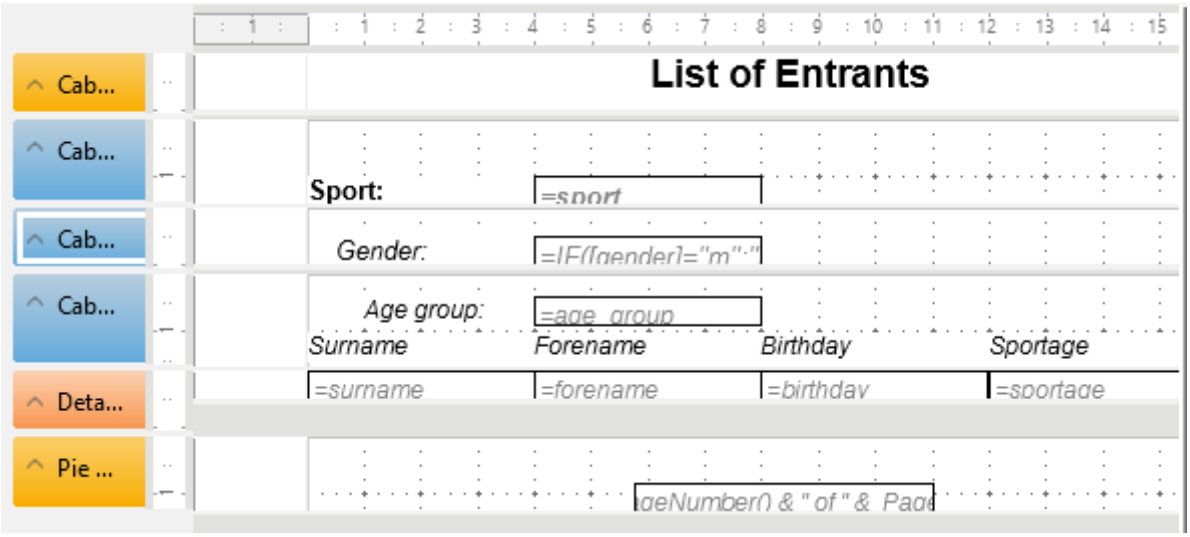
En Propiedades a la derecha, seleccione **General > Formato**. La propiedad de *Formato* estará establecida como *Texto*. Haga clic en el botón con la elipse (...). Se abrirá el diálogo de formato, que también se usa en Calc, Writer o al crear formularios. Seleccione **Categoría > Número** y confirme con **Aceptar**. El formato en que se muestra el campo queda modificado de *Texto* a *Número*.

Mover cuadros en el Generador de informes

Los campos también se pueden mover más allá de los límites de una sección a otra sección en el *Generador de informes*. Pero debe haber suficiente espacio para dicho campo en el destino. Una parte de un campo no puede estar en la misma ubicación que una parte de otro campo.

La ubicación de los campos con el ratón no es precisa, por lo que se debe proporcionar un amplio espacio en cada sección para los campos. En una sección como el encabezado del grupo *age_group*, los campos se pueden colocar con precisión utilizando las teclas de flecha.

Para posicionar campos usando el teclado, fíjese en las propiedades **General > Posicion X** y **General > Posicion Y**.



Aquí se agregó un campo de etiqueta para el encabezado. La entrada del texto para el campo de etiqueta aparece en las Propiedades de los campos.

El pie de página se establece en las propiedades: **Visible > No**. El margen inferior del documento contiene demasiado espacio. Recuerde que la cantidad disponible ya se reduce adicionalmente por el tamaño de los márgenes de la página. De forma predeterminada, todos los márgenes de página están configurados para el formato de página A4 a 2 cm.

Para obtener más opciones de formato para informes, consulte el *Capítulo 6, Informes*.

Ampliación de la base de datos de muestra

El ejemplo presentado aquí es solo el primer paso para una base de datos en el sector deportivo. Ahora se puede agregar un campo para cada posible elemento de información necesaria. Un buen lugar para agregar dicho campo es en la tabla rel_starter_sport.

Sin embargo, si las entradas se aplican a varias competiciones para el mismo deporte, se debe incluir un campo de fecha u otro campo que pueda asignarse a la competición respectiva en la tabla *rel_starter_sport*. El campo también se convertirá en parte de la clave principal de la tabla.

Quizás también podría agregarse la membresía del club. Agregar un campo en la tabla *starters* sería suficiente. Cuando muchos clubes tengan el mismo nombre, se sugiere agregar una tabla Club separada y también la clave foránea correspondiente en la tabla *starters*.

Luego, por supuesto, se tiene que determinar, como con todas las competiciones, quién debe ubicarse en qué grupo de edad y deporte. Aquí se requiere ordenar, lo que podría terminar nuevamente como un informe con una lista de resultados.

Sería bueno que cada principiante (de nuevo a través de un informe) obtuviera un certificado bellamente diseñado con el rendimiento personal y la ubicación. Dichas ampliaciones son posibles fácilmente, como se describe en otros capítulos de esta guía.

LibreOffice

Guía de Base

Capítulo 2, Creación de una base de datos

Derechos de autor

Colaboradores

Este libro está adaptado de versiones anteriores del mismo.

De esta edición

Pulkit Krishna
Juan Peramos
Celia Palacios
Dan Lewis
Juan Carlos Sanz Cabrero
Jean Hollis Weber
B. Antonio Fernández

De ediciones previas

Jean Hollis Weber
Peter Schofield

Comentarios y sugerencias

Puede dirigir cualquier clase de comentario o sugerencia acerca de este documento a: documentation@es.libreoffice.org.

Nota

Todo lo que envíe a la lista de correo, incluyendo su dirección de correo y cualquier otra información personal que escriba en el mensaje se archiva públicamente y no puede ser borrada

Fecha de publicación y versión del programa

Versión en español publicada el 14 de junio de 2021. Basada en la versión 6.2 de LibreOffice.

Uso de LibreOffice en macOS

Algunas pulsaciones de teclado y opciones de menú son diferentes en macOS de las usadas en Windows y Linux. La siguiente tabla muestra algunas sustituciones comunes para las instrucciones dadas en este capítulo. Para una lista detallada vea la ayuda de la aplicación.

Windows o Linux	*Equivalente en Mac*	*Efecto*
Herramientas > Opciones opción de menú	**LibreOffice > Preferencias**	Acceso a las opciones de configuración
Clic con el botón derecho	*Control+clic* o *clic derecho* depende de la configuración del equipo	Abre menú contextual
Ctrl (Control)	⌘ *(Comando)*	Utilizado con otras teclas
F5	*Mayúscula+⌘+F5*	Abre el navegador
F11	*⌘+T*	Abre la ventana de estilos y formato

Contents

Introducción

Los conceptos básicos para crear una base de datos en LibreOffice se describen en el “Capítulo 8, Introducción a Base” de la *Guía de primeros pasos*.

El componente de base de datos de LibreOffice, llamado **Base**, proporciona una interfaz gráfica para trabajar con bases de datos. Además, LibreOffice contiene una versión del motor de base de datos HSQL. Esta base de datos HSQLDB puede ser utilizada por un solo usuario. Todo el conjunto de datos se almacena en un archivo ODB que tiene un mecanismo de bloqueo de archivos en la carpeta o directorio de configuración cuando lo abre un usuario.

Desde la versión LibreOffice 4.2 ha estado disponible una base de datos Firebird, además de la interna y predeterminada HSQLDB. La base de datos Firebird se incluyó como “característica experimental” hasta la versión 6.0. Los ejemplos de bases de datos en este libro continúan haciendo referencia a HSQLDB, pero están personalizados para que la mayoría de las funciones sean directamente transferibles a Firebird. Se muestran alternativas en Firebird, si es necesario.

Crear una nueva base de datos utilizando el motor interno HSQL

Si no se planifica crear una base de datos con múltiples usuarios o si desea adquirir experiencia inicial con una base de datos, el motor de base de datos interno (incorporado en LibreOffice) será suficiente. En una etapa posterior es posible transferir la base de datos a un entorno HSQLDB externo, donde varios usuarios pueden tener acceso concurrente a la base de datos en el servidor HSQLDB. Este proceso se describe en el Apéndice de este libro.

Para crear una base de datos interna desde la pantalla de inicio de LibreOffice, haga clic en el botón *Base de datos de Base*. O desde cualquier lugar de LibreOffice, use el menú **Archivo > Nuevo > Base de Datos**. Se abrirá el *Asistente de bases de datos* (figura 12)

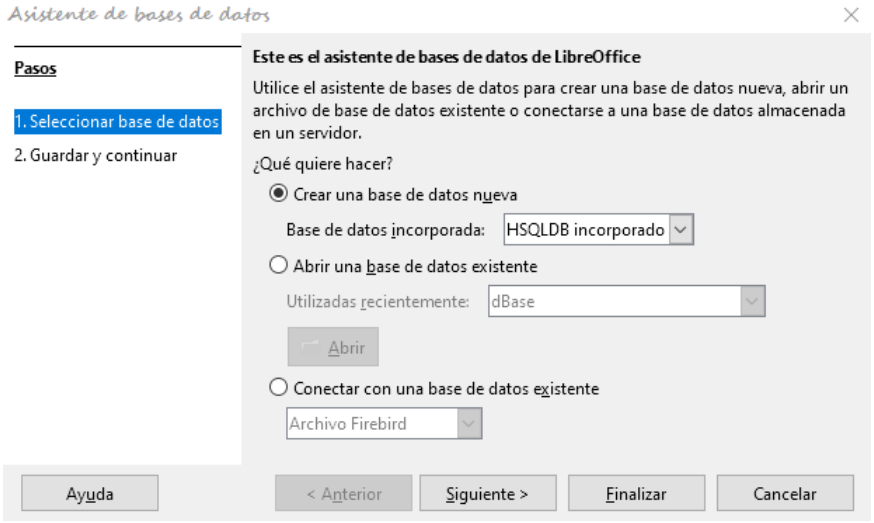


Figura 12: Paso 1 del Asistente de base de datos

Seleccione *Crear una base de datos nueva*. De manera predeterminada, será una base de datos con HSQLDB incorporado. También está disponible la opción de crear una base de datos con Firebird incorporado. Consulte la nota de precaución más adelante.

Las otras opciones del *Paso 1* sirven para abrir un archivo ODB existente o para crear una conexión a una base de datos externa, como una libreta de direcciones o una base de datos MySQL.

Elija *Siguiente* > para continuar con el *Paso 2* del *Asistente de base de datos* (figura 13).

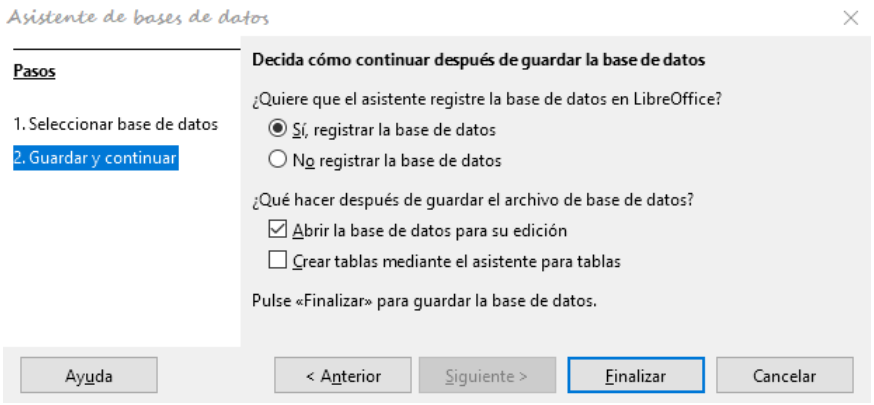


Figura 13: Paso 2 del Asistente de base de datos

Una base de datos registrada puede ser utilizada por otros componentes de LibreOffice, por ejemplo, para la combinación de correspondencia de cartas en Writer. Se recomienda que todas las bases de datos se registren cuando se creen, pero la elección final corresponde al usuario.

Seleccione *Abrir la base de datos para su edición* y anule la selección de *Crear tablas mediante el asistente para tablas*. Esta última opción fue descrita en el capítulo 1. En el resto de este libro no se utilizan los asistentes para crear tablas, consultas, etc.

Haga clic en *Finalizar* para guardar la base de datos. Se abre el diálogo *Guardar como*, que solicita un nombre y una ubicación para el archivo `*.odb`, que está preparado para la entrada de registros de la base de datos interna y el almacenamiento de consultas, formularios e informes. A diferencia de los otros módulos de LibreOffice, es necesario guardar el archivo antes de que haya realizado ninguna entrada visible.

Precaución

Si las características experimentales están activas (en el menú **Herramientas > Opciones > LibreOffice > Avanzado > Funcionalidades opcionales**) y abre una base de datos existente con HSQLDB incorporado, al usar las **Herramientas > SQL** se mostrará un mensaje en inglés que pregunta si desea migrar la base de datos a Firebird en este momento, pues el documento contiene datos con HSQL incrustado, del cual se desaconseja su uso.

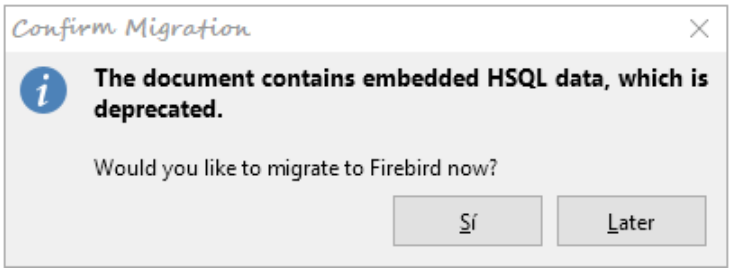


Figura 14: Diálogo que sugiere iniciar la migración de la base de datos a Firebird

Se recomienda tener precaución y no iniciar la migración inmediatamente. Tenga mucho cuidado en este punto. Pulse el botón *Later* (Después) y haga una copia de seguridad del archivo de la base de datos. Cuando esté listo para migrar la base de datos, siga este procedimiento:

8) Asegúrese de tener una copia de seguridad del archivo de base de datos HSQLDB.
9) Adapte las funciones tanto como pueda de acuerdo con la lista en esta página wiki: https://wiki.documentfoundation.org/Documentation/FirebirdMigration.
10) Copie las vistas que no se pueden convertir directamente del código SQL y guárdelas como consultas. Vea el “Capítulo 5, Consultas” donde viene más información.

11) Los nombres de tabla y los nombres de columna en Firebird pueden tener un máximo de 31 caracteres. Si es necesario, introduzca un nombre más corto.

12) Las tablas de texto puro (tabla `*.csv` integrada, etc.) no se permiten en Firebird, por lo que deben colocarse en otro lugar.

Acceder a bases de datos externas

Una base de datos externa tiene que existir antes de poder acceder a ella. Si se desea acceder a una base de datos, esta debe configurarse previamente para permitir conexiones de red con un nombre de usuario y contraseña específicos, antes de que los programas externos se puedan conectar a ella. Cuando dicha base de datos está configurada correctamente, según el software de conexión disponible (el controlador de la base de datos), ya puede crear tablas, datos de entrada y datos de consulta.

Haga clic en **Archivo > Nuevo > Base de datos** para abrir el *Asistente de base de datos* (figura 12) y elija *Conectar con una base de datos existente*. La lista de tipos de bases de datos disponibles varía según el sistema operativo y la interfaz de usuario, pero siempre deben estar disponibles los siguientes:

- JDBC
- Oracle JDBC
- ADO
- Hoja de cálculo
- dBASE
- Texto
- MySQL
- ODBC
- PostgreSQL
- Documento de Writer
- …así como varios tipos de libretas de direcciones.

Las opciones de conexión del siguiente paso varían según el tipo de base de datos seleccionada. Puede cambiar las opciones después de crear el archivo `*.odb`.

Algunos tipos de bases de datos (por ejemplo, una conexión a hojas de cálculo) no permiten ingresar datos nuevos. Estos tipos se utilizan solo para buscar o informar sobre datos existentes.

Las descripciones en los siguientes capítulos tratan exclusivamente de LibreOffice Base utilizando la base de datos HSQLDB incorporada. La mayor parte del trabajo de diseño se puede extender a bases de datos que usan MySQL, PostgreSQL, etc.

A continuación hay un par de breves ejemplos de cómo puede conectarse Base con una base de datos externa.

Bases de datos MySQL y MariaDB

Base puede conectarse a bases de datos MySQL y MariaDB mediante tres métodos. La forma más simple y rápida es la conexión directa con el conector MySQL. Las otras dos son la conexión con el conector ODBC y con el JDBC.

Nota

En las bases de datos MySQL y MariaDB es posible ingresar y cambiar datos en las tablas sin un campo de clave principal. La interfaz de Base mostrará estas tablas, pero no ofrece opciones de entrada o modificación.

Si desea usar tablas sin una clave primaria puede usar **Herramientas > SQL…** en su lugar o, dentro de los formularios, mediante macros que proporcionen datos a las tablas.

Creación de un usuario y una base de datos

Después de instalar MySQL o MariaDB, realice los siguientes pasos en la secuencia que sigue:

1) La cuenta de administrador en MySQL se llama `root`. Los usuarios de Linux deben tener en cuenta que este no es el usuario `root` del sistema operativo Linux. Al usuario `root` de MySQL se le tiene que asignar una contraseña directamente después de la instalación, si no se hizo ya antes.

```
mysql -u root -p
```

Al empezar, no hay una contraseña establecida, así que solo presione *Entrar*. Aparecerá el *prompt* (cursor) de entrada de la consola MySQL:

```
mysql>
```

Todas las siguientes entradas se realizan en la consola MySQL. Las contraseñas pueden ser diferentes, dependiendo de si la solicitud proviene de la computadora local (localhost) o de una computadora diferente que actúa como un servidor MySQL (host):

```
SET PASSWORD FOR root@localhost=PASSWORD('Password');
SET PASSWORD FOR root@host=PASSWORD('Password');
```

Para los usuarios de Windows, la segunda línea dice:

```
SET PASSWORD FOR root@'%'=PASSWORD('Password');
```

2) Como medida de seguridad, se eliminan todos los usuarios anónimos actuales.

```
DELETE FROM mysql.user WHERE User='';
DELETE FROM mysql.db WHERE User='';
FLUSH PRIVILEGES;
```

3) Se crea una base de datos llamada *libretest*.

```
CREATE DATABASE libretest;
```

4) Todos los derechos de la base de datos *libretest* se otorgan al usuario `lotest`, que iniciará sesión con la contraseña libre:

```
GRANT ALL ON libretest.* TO lotest IDENTIFIED BY 'libre';
```

Ahora la base de datos MySQL está disponible y se puede conectar de la siguiente manera.

Conexión directa a MySQL usando una extensión

Comenzando con esta versión de LibreOffice 6.2, la conexión directa de Base a MySQL o MariaDB se ha integrado en LibreOffice. Ya no es necesario instalar una extensión.

Conexión MySQL a través de JDBC

Para poder usar JDBC, es necesario instalar `mysql-connector-java.jar`

Este archivo Java es mejor copiarlo en la misma carpeta o directorio donde se encuentra la versión actual de Java utilizada en LibreOffice. Para una instalación de Linux es probable que sea una subcarpeta como:

```
...javapath.../lib/ext
```

De manera alternativa, la carpeta apropiada para contener el archivo Java se puede configurar a través de **Herramientas > Opciones > LibreOffice > Avanzado > Opciones de Java > Ruta de clase…** En el diálogo *Ruta de clase*, pulse el botón *Añadir archivador…* y seleccione la ruta correcta en su computadora donde se encuentra el archivo Java con el conector JDBC.

Conexión MySQL a través de ODBC

Para conectarse a través de ODBC, es necesario primero tener instalado el software ODBC. Los detalles de cómo hacerlo no se proporcionan aquí.

Después de la instalación del software, puede suceder que LibreOffice rechace el servicio porque no puede encontrar la biblioteca `libodc.so`. En la mayoría de los sistemas estará presente, por ejemplo, como `libodbc.so.2`. Necesita hacer un enlace simbólico a este archivo en la misma carpeta o directorio con el nombre `libodbc.so`.

En los archivos `odbcinst.ini` y `odbc.ini`, que son necesarios para el sistema, debe realizar entradas similares a las siguientes:

`odbinst.ini:`

```
[MySQL]
Description = ODBC Driver for MySQL
Driver = /usr/lib/libmyodbc5.so
```

`odbc.ini:`

```
[MySQL-test]
Description = MySQL database test
Driver = MySQL
Server = localhost
Database = libretest
Port = 3306
Socket =
Option = 3
Charset = UTF8
```

En los sistemas Linux, estos dos archivos se encuentran en la carpeta o directorio `/etc/UnixODBC`.

Los detalles para los parámetros de conexión se pueden encontrar en el *Manual de referencia de MySQL* de la versión que esté usando de la página web https://dev.mysql.com/doc.

Nota

Si no ingresa el juego de caracteres (*charset*) que va a utilizar, puede haber problemas con las diéresis, incluso si la configuración fuera la misma tanto en MySQL o MariaDB como en Base.

Conexión a una base de datos MySQL con el Asistente de base de datos

Para acceder a una base de datos MySQL existente usando una conexión directa, siga estos pasos:

1) En el *Paso 1* del *Asistente de base de datos*, seleccione *Conectar con una base de datos existente*. De la lista de formatos de base de datos en el menú desplegable (figura 15), seleccione *MySQL.* Haga clic en el botón *Siguiente >.*

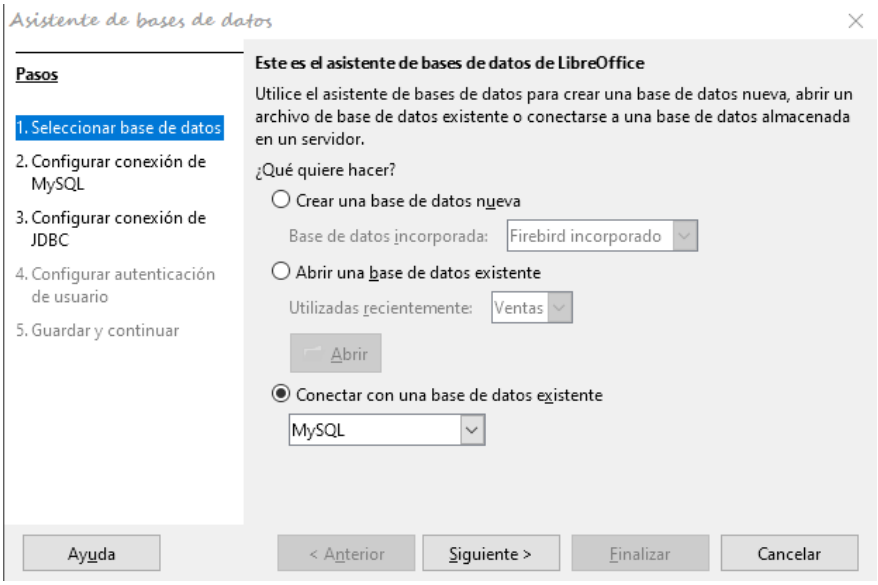


Figura 15: Conectarse a una base de datos MySQL existente

2) En el *Paso 2* del *Asistente de base de datos* (figura 16) puede elegir conectarse mediante ODBC, mediante JDBC o directamente. Le mostraremos cómo realizar la conexión con MySQL de las tres maneras y los restantes pasos 3 y 4 del asistente variarán de acuerdo a cada una.

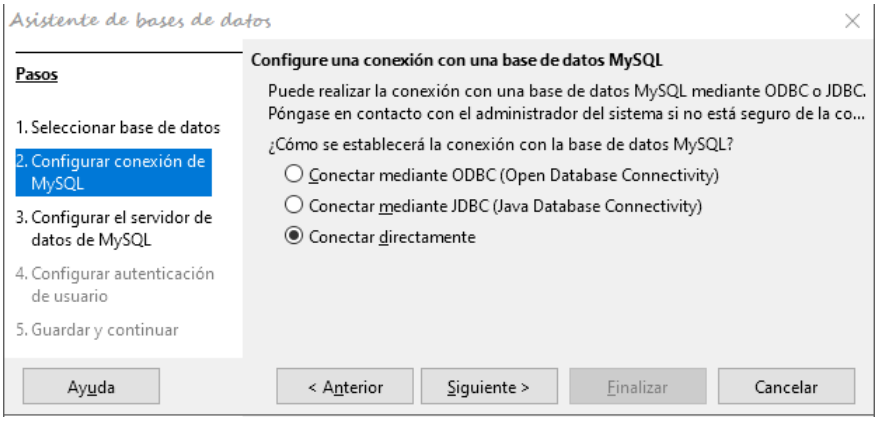


Figura 16: Paso 2 del Asistente de base de datos.

Conexión directa

La conexión directa es la mejor, tanto para la velocidad como para la funcionalidad. En el *Paso 3* (figura 17), complete la información necesaria.

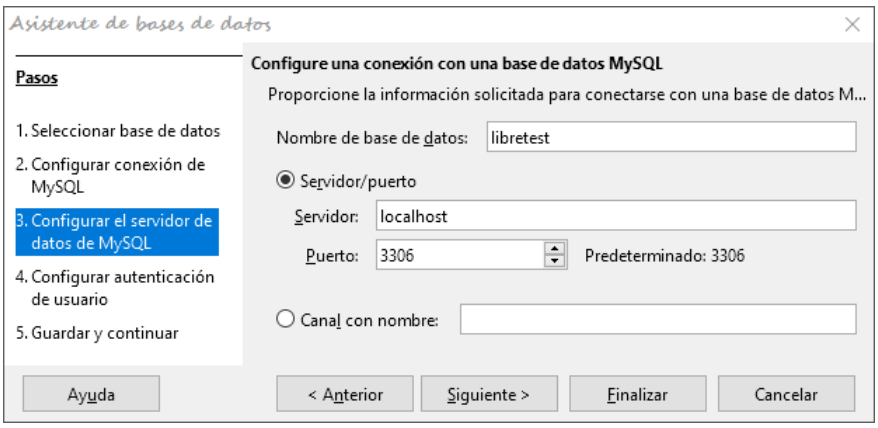


Figura 17: Paso 3 del Asistente de base de datos

Debe conocer previamente el nombre de la base de datos. Si el servidor está en la misma computadora que la interfaz de usuario en la que se creará la base de datos, puede seleccionar localhost como servidor. De lo contrario, puede usar una dirección IP o, de acuerdo con la estructura de la red, el nombre de la computadora o incluso una dirección de Internet. Por lo tanto, es posible que Base acceda a una base de datos que se encuentra en la página de inicio de alguien.

Cuando trabaje con Base a través de Internet, debe saber cómo se configura la conexión. ¿Es segura la conexión? ¿Cómo se transmite la contraseña?

Cualquier base de datos accesible a través de Internet debe estar protegida por un nombre de usuario específico con una contraseña. Esto proporciona una forma directa de probar si la conexión debe continuar. El usuario correspondiente debe configurarse en MySQL o MariaDB para el servidor nombrado.

En el *Paso 4* (figura 18) proporcione un nombre de usuario y marque *Contraseña obligatoria*. Haga clic en el botón *Probar conexión* para iniciar la autenticación con el nombre de usuario dado. Después de ingresar la contraseña, se le informará si la conexión tuvo éxito.

Si, por ejemplo, MySQL no se está ejecutando actualmente, recibirá un mensaje de error.

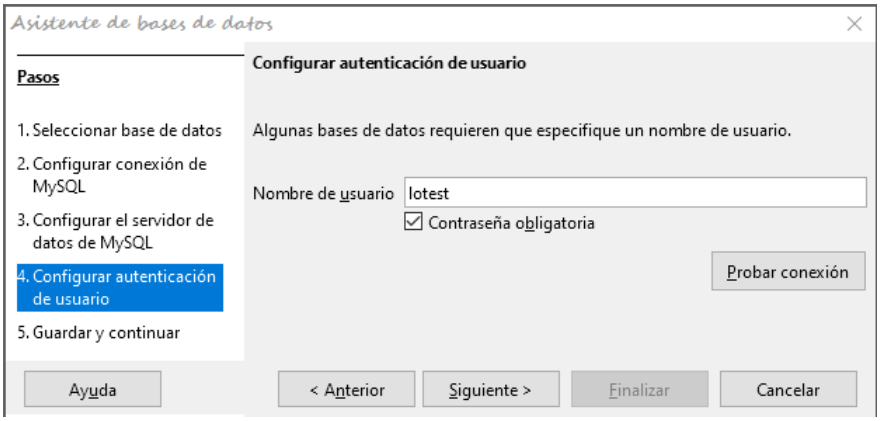
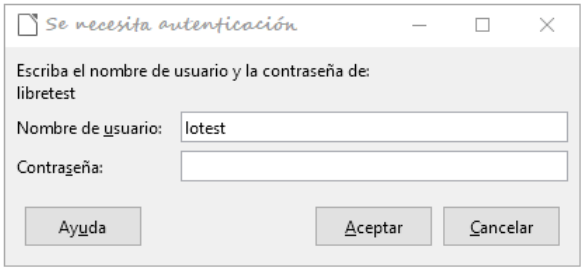


Figura 18: Paso 4 del Asistente de base de datos: configurar la autenticación de usuario

Nota

Cada vez que acceda al archivo de la base de datos MySQL, aparecerá el mismo diálogo que cuando ingresó por primera vez a ella.



Haga clic en el botón *Siguiente>* del *Asistente de base de datos* para mostrar el *Paso 5* (figura 19). Seleccione *No registrar la base de datos* y marque *Abrir la base de datos para su edición*. Haga clic en *Finalizar*.

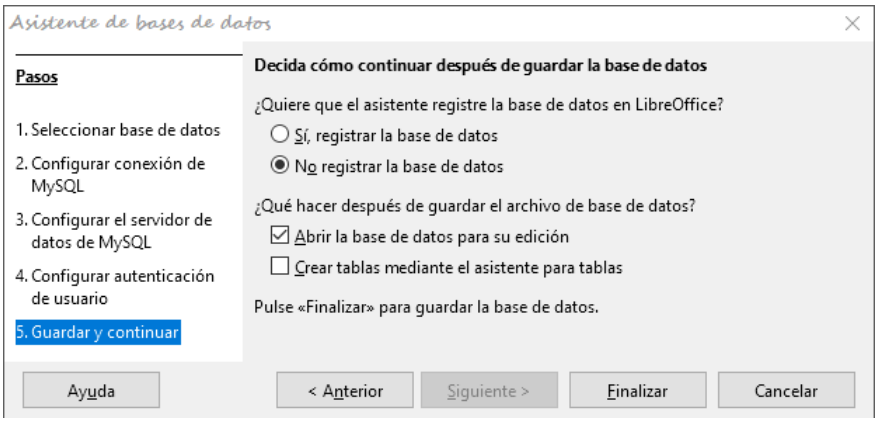


Figura 19: Paso 5 del Asistente de base de datos: decida cómo proceder después de guardar la base de datos

En este ejemplo, la base de datos no se registrará, ya que solo se está creando para pruebas. El registro solo es necesario si otros programas como Writer van a acceder a los registros, por ejemplo, para una combinación de correspondencia.

El asistente finaliza la configuración de la conexión guardando la conexión de la base de datos. Se crea el archivo Base y se abre una vista de las tablas de la base de datos MySQL (figura 20). Las tablas de la base de datos se muestran bajo el nombre de la base de datos.

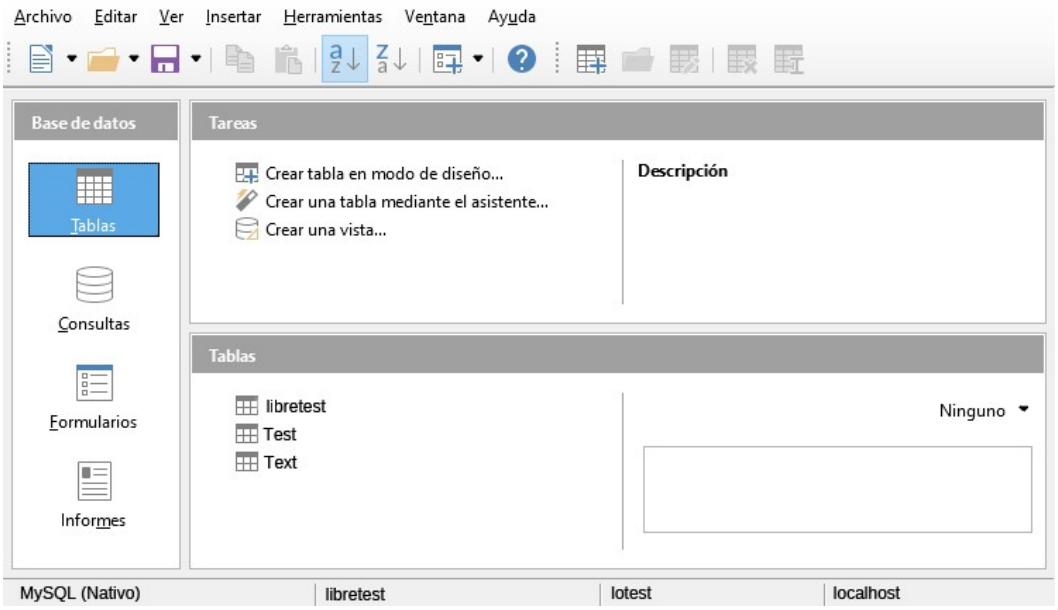


Figura 20: Vista del archivo de la base de datos abierto, con una descripción general de la tabla, y en el pie de página,las características: Controlador = MySQL (Native). Nombre de la base de datos = libretest . Usuario = lotest y Servidor de la base de datos = localhost.

En esta etapa, el archivo `*.odb` contiene solo la información de conexión que se leerá cada vez que se inicie la base de datos para poder acceder a las tablas de la base de datos MySQL.

Algunos controladores mostrarán solo la base de datos *libretest* para la cual se ordenó la conexión. Otros controladores podrán mostrar también otras bases de datos MySQL o MariaDB en el mismo servidor.

Incluso con los controladores que muestran solo una base de datos, el acceso a otras tablas para consultas es posible si el usuario de la base de datos (`lotest`, en el ejemplo anterior) puede acceder a los registros con su contraseña.

A diferencia de los controladores nativos de LibreOffice anteriores, este no proporciona acceso de escritura a otras bases de datos MySQL en el mismo servidor.

A diferencia de la base de datos interna de Base, las consultas en MySQL requieren el nombre de la base de datos para definir las tablas. Por ejemplo:

```
... FROM "test"."Class" AS "Class", ...
```

En el pasado era necesario dar a la combinación de nombre de base de datos y nombre de tabla un nombre alternativo (alias) usando la instrucción `AS`. Desde hace varias versiones, el uso de `AS` se eliminó; es decir, ya no es necesario que al alias de la tabla le preceda esta instrucción; es decir, es suficiente:

```
... FROM "test"."Class" "Class", ...
```

Las tablas se pueden crear y eliminar en la base de datos. Los valores de incremento automático (valores automáticos) funcionan y se pueden seleccionar en la etapa de diseño de la tabla.

En MySQL, los valores comienzan en 1.

Conexión mediante ODBC

Los primeros pasos para hacer una conexión ODBC son los mismos que para una conexión directa. Si se selecciona una conexión mediante ODBC a MySQL en el segundo paso, entonces aparecerá, en el tercer paso del *Asistente de base de datos,* lo que ilustra la figura 21.

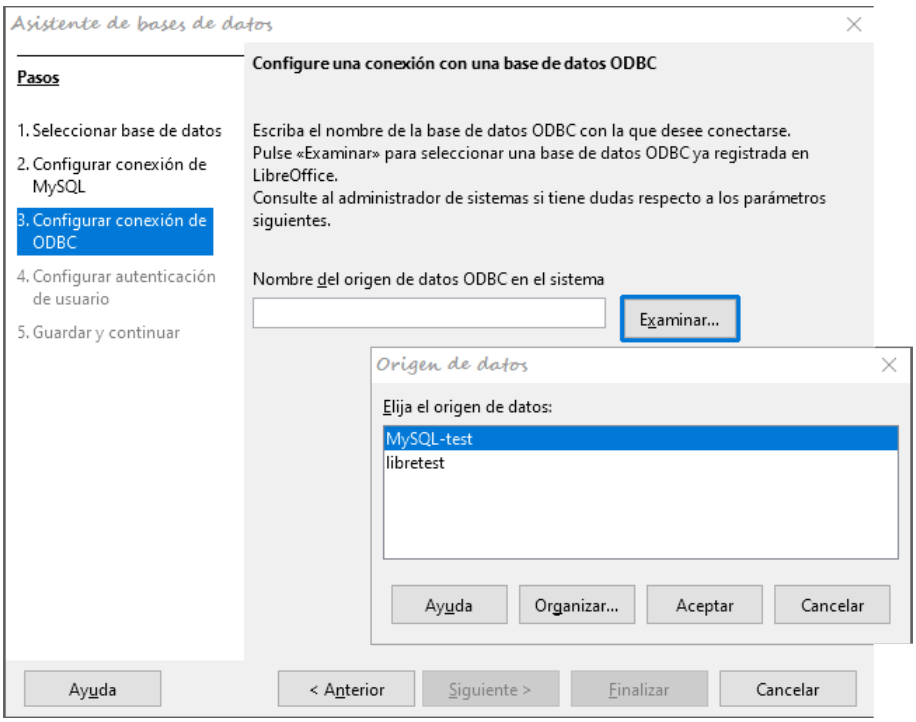


Figura 21: Paso 3 del Asistente de base de datos: configurar una conexión a una base de datos MySQL con conector ODBC

La fuente de datos ODBC no necesita tener el mismo nombre que la base de datos en MySQL. Aquí se tiene que ingresar el nombre que figura en el archivo `odbc.ini`. La forma más sencilla de hacerlo es leer el nombre directamente de `odbc.ini` mediante el botón *Examinar*.

En el diálogo emergente aparecerá el nombre de la base de datos (sin extensión) que viene en el archivo `odbc.ini`. Aquí también, cuando se conecta a una base de datos, se pueden leer fácilmente otras tablas en el servidor MySQL.

Los pasos 4 y 5 del asistente son idénticos a los de una conexión directa.

Conexión mediante JDBC

Para una conexión con un conector JDBC, los primeros pasos son los mismos que en los casos anteriores. La diferencia aparece solo en el *Paso 3* (figura 22).

En este caso, el asistente solicita la misma información que para una conexión directa. El nombre de la base de datos es el mismo que usa MySQL.

Use el botón *Probar clase* para verificar si Java tiene acceso al archivo `mysql-connector-java.jar`. Este archivo deberá estar en la ruta de la versión Java elegida o ser parte de los paquetes preinstalados con LibreOffice.

Todos los pasos adicionales son idénticos a las conexiones anteriores. Las conexiones a otras bases de datos en el mismo servidor MySQL vuelven a ser de solo lectura.

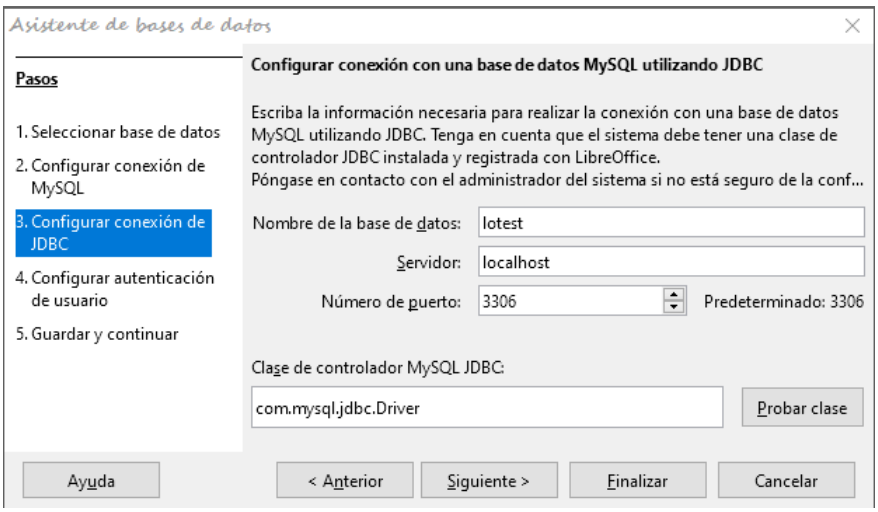


Figura 22: Paso 3 del Asistente de base de datos: configurar una conexión con conector JDBC

PostgreSQL

LibreOffice tiene un controlador directo para las bases de datos PostgreSQL, el cual viene preinstalado. Para garantizar una conexión segura, siga estas breves instrucciones para los primeros pasos, después de instalar PostgreSQL.

Crear un usuario y una base de datos

Los siguientes pasos son necesarios después de la instalación, si se utiliza el administrador de paquetes en OpenSUSE. Puede asumir pasos similares en otros sistemas operativos.

1) Al usuario `postgres` se le tiene que asignar una contraseña. Se puede hacer usando la utilidad del sistema operativo.
2) El servidor PostgreSQL debe ser iniciado por el administrador:

```
service postgresql start
```

o, de manera alternativa:

```
rcpostgresql start
```

3) El usuario `postgres` inicia sesión en la consola con:

```
su postgres
```

4) Un usuario de base de datos sin privilegios, aquí llamado `lotest`, se crea con una contraseña:

```
createuser -P lotest
```

5) Para permitir que el usuario de la base de datos se conecte a la base de datos que se va a crear, se debe cambiar una entrada en el archivo `/var/lib/pqsql/data/pg_hba.conf`. Este archivo incluye los métodos utilizados para identificar usuarios en varios niveles. El método que LibreOffice utiliza para comunicarse es el de `"password"` (contraseña) y no el método de `"ident"` (identificación), como viene establecido inicialmente en el archivo.
6) El usuario de sistema PostgreSQL inicia sesión con la instrucción `psql`:

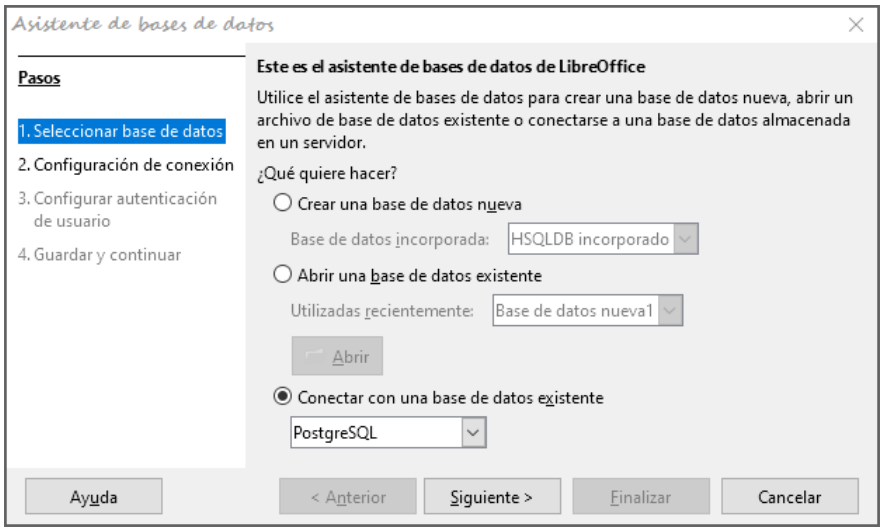
```
psql -d template1 -U postgres
```

7) El usuario del sistema crea la base de datos *libretest*:

```
CREATE DATABASE libretest;
```

Conexión directa a Base

Elija la entrada *PostgreSQL* en la lista *Conectar con una base de datos existente* en el *Paso 1* del asistente.



Para realizar la conexión, proporcione el nombre de la base de datos (en este ejemplo, *dbname*) y el del host. En algunas circunstancias, es necesario proporcionar el nombre completo del host, incluido el nombre de dominio.

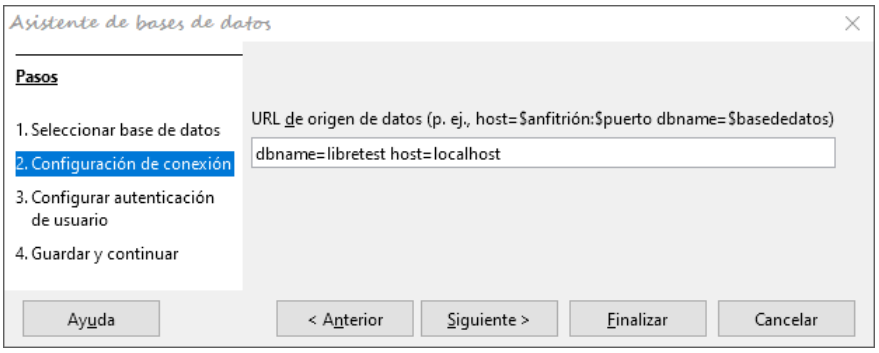


Figura 23: Paso 2 del Asistente de base de datos: configurar una conexión directa

La autenticación del usuario en el *Paso 3* es exactamente la misma que para MySQL. El diálogo *Guardar como* (figura 28) muestra los diversos esquemas en PostgreSQL. El único en el que realmente se puede guardar en LibreOffice es el esquema *public*, a menos que se hayan otorgado derechos extendidos a este usuario.

Nota

Si las tablas de la base de datos interna de HSQLDB se copian a PostgreSQL, el *Asistente de importación* usará los nombres de tabla simples de HSQLDB, por ejemplo, *Table1*. Sin embargo, importar con este nombre generará errores. En lugar de ello, el nombre del esquema debe anteponerse, de modo que *Table1* se convierta en *public.Table1*.

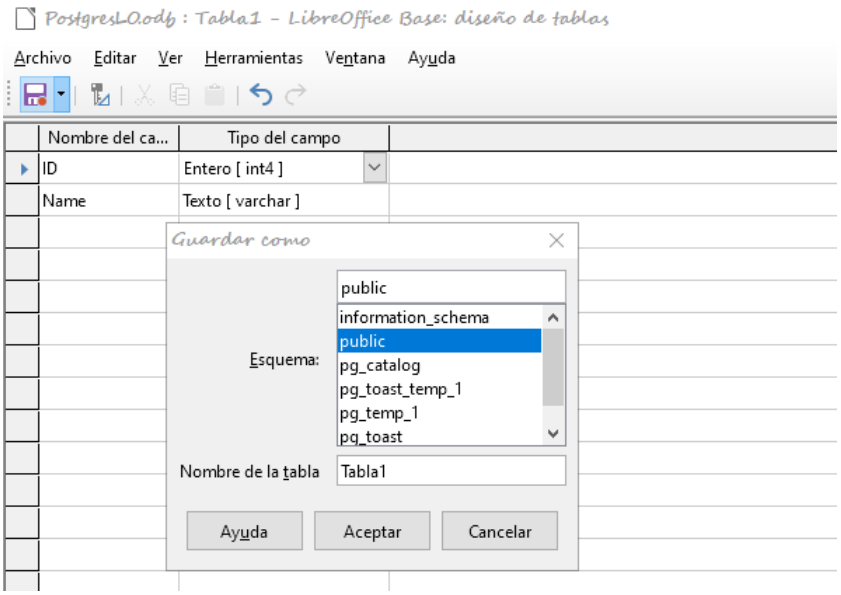


Figura 24: Guardar nombres de tabla con esquema public

Cuando se crean tablas, Base puede sugerir tipos de datos que la instalación actual de PostgreSQL no puede manejar. Por ejemplo, los campos de texto predeterminados reciben el tipo de campo *Texto [carácter_datos]*. PostgreSQL no puede procesar este tipo de campo. Cambiar el tipo a *Texto [varchar]* resuelve el problema.

Los diversos esquemas aparecen en la vista de tabla de PostgreSQL. En el esquema *public* puede ver una tabla llamada *Nombre* (figura 25).

Si la base de datos se abre por primera vez, verá muchas tablas en la carpeta *information_schema* en el panel *Tablas*.

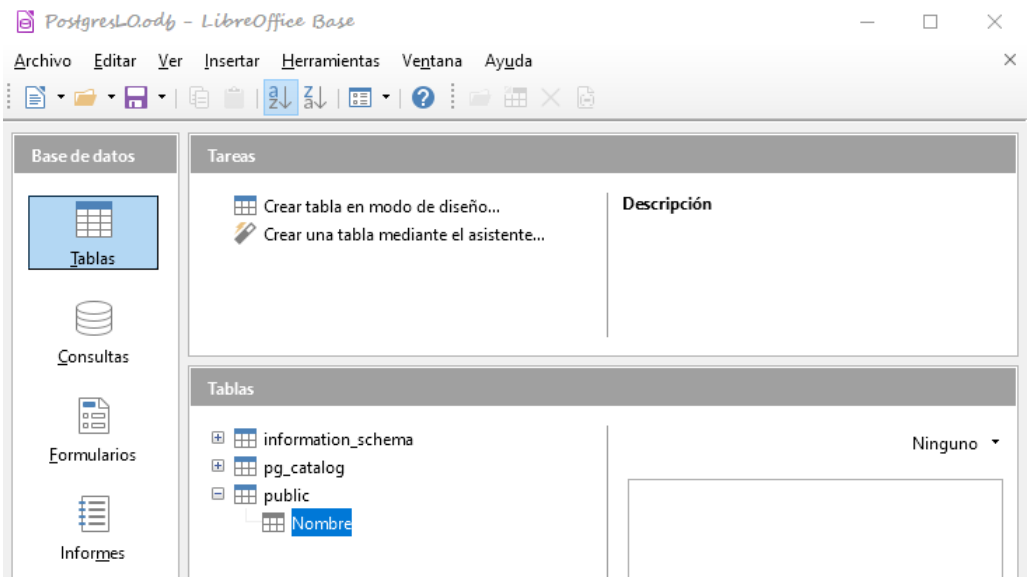


Figura 25: Vista de tabla de PostgreSQL en LibreOffice Base

Estas tablas, como las del área o carpeta *pg_catalog* no pueden ser leídas o escritas por el usuario ordinario. Estas áreas diferentes se llaman *schema* en PostgreSQL. Los usuarios crean nuevas tablas en el esquema *public*.

Bases de datos dBase

Las bases de datos dBase tienen un formato donde todos los datos están contenidos en tablas separadas, previamente inicializadas. Los enlaces entre las tablas deben hacerse en código de programa. Las relaciones de estas bases de datos no son compatibles con Base.

El formato dBase es especialmente adecuado para el intercambio y la edición extensiva de datos. Además, los cálculos de hoja de cálculo pueden acceder directamente a las tablas de dBase. Sin embargo, dBase no tiene una forma de evitar la eliminación de, por ejemplo, los artículos en una base de datos de una biblioteca a los que se sigue haciendo referencia en la tabla de préstamos.

Nota

En la actualidad, las únicas bases de datos dBase que se reconocen son aquellas contenidas en archivos con la terminación `*.dbf` en minúsculas. No se reconocen otras terminaciones, tales como `*.DBF` (bug 46180).

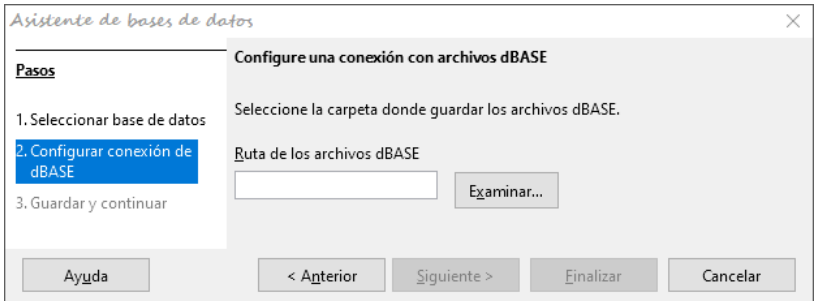


Figura 26: Configurar una conexión a una base de datos dBase

La conexión se realiza con una carpeta o directorio específico. Todos los archivos `*.dbf` en esta carpeta se incluirán y se mostrarán en la base de datos de Base. Se pueden vincular mediante consultas.

Precaución

Cuando usa el *Asistente de bases de datos*, Base le preguntará mediante un mensaje si desea crear una nueva carpeta o directorio con el mismo nombre de su archivo `.dbf` (figura 28). Si responde pulsando el botón *Sí* aparecerá un nuevo mensaje que advierte que no es posible crear la carpeta y no podrá pasar de este punto (figura 27). Lo que debe responder en el primer diálogo *Confirmación* es *No*. De esta manera se guardará la ruta correcta y entonces podrá pulsar en el botón *Siguiente* del asistente.

Nota

Es probable que al abrir una base de dBase, Base le pedirá que especifique el conjunto de caracteres que usa el archivo (figura 29). La lista es larga. Es recomendable que conozca la información técnica del archivo antes de realizar la importación de la base de datos para elegir correctamente.

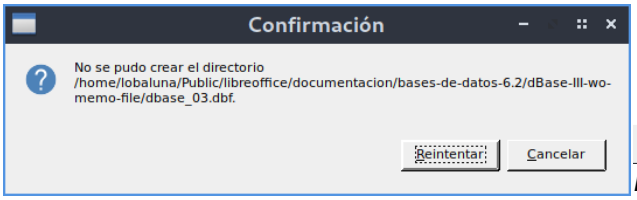


Figura 27: Diálogo Confirmación con un mensaje de error

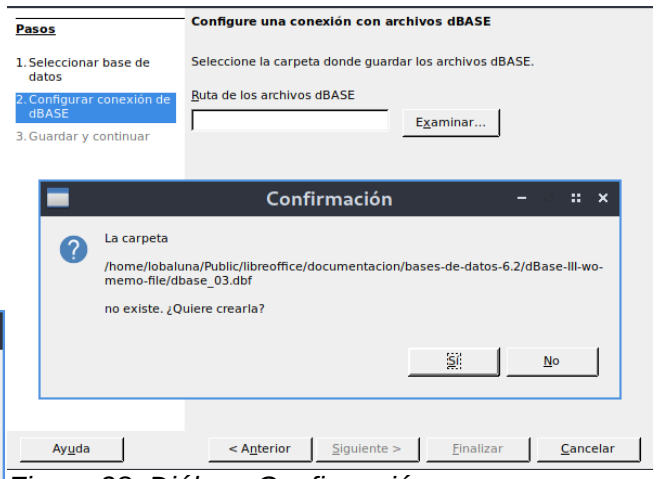


Figura 28: Diálogo Confirmación para crear una carpeta donde expandir las tablas del archivo DBF de la base de datos dBase

Las tablas en dBase no tienen clave primaria. Como regla general, pueden describirse como si fueran hojas de cálculo de Calc.

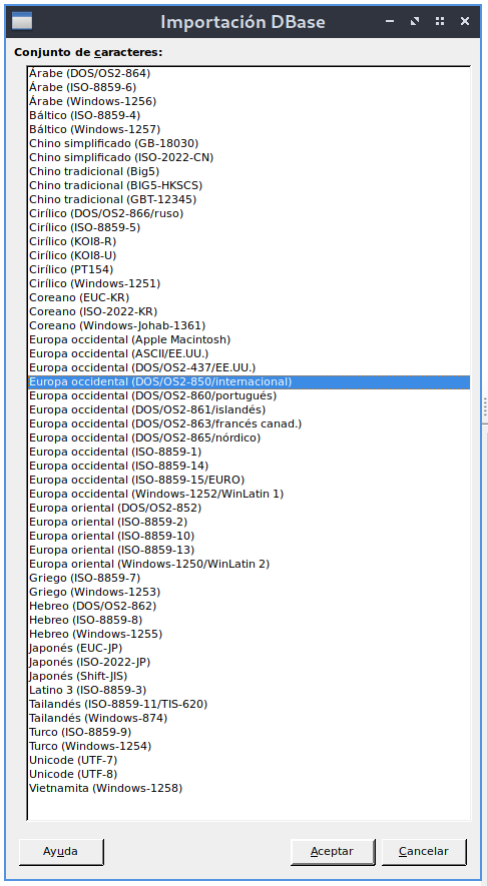


Figura 29: Listado del conjunto de caracteres para importar una base de datos dBase

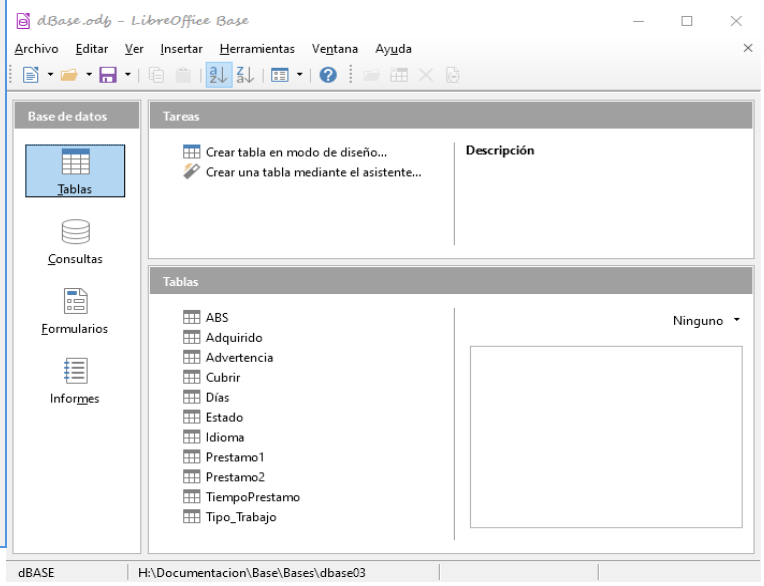


Figura 30: Tablas en un archivo dBase

Las tablas se crean y luego se copian como archivos nuevos en la carpeta o directorio previamente seleccionado.

El número de tipos de campo diferentes para una nueva tabla de dBase es claramente menor que cuando se usa el formato interno HSQLDB.

En la figura 31 se pueden ver todavía algunos tipos de campo con el mismo nombre de tipo.

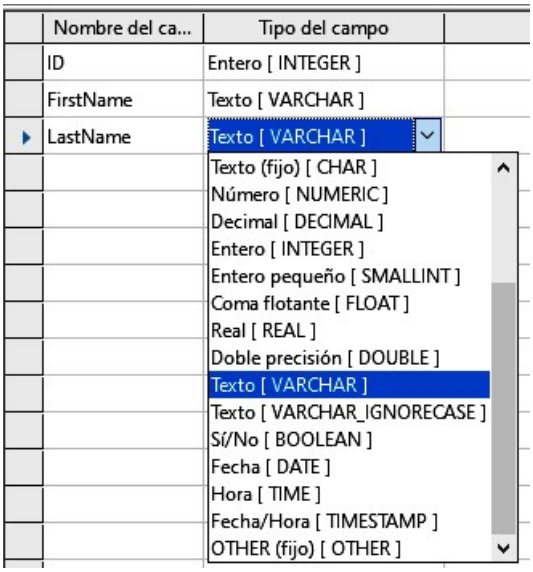


Figura 31: Tipos de campo para una nueva tabla de dBase

Base se hace cargo de la codificación de caracteres del sistema operativo. Aún así, los archivos dBase antiguos pueden presentar fácilmente errores cuando se importan caracteres especiales. El juego de caracteres se puede corregir posteriormente usando **Editar > Base de datos > Propiedades…** y en el diálogo *Propiedades de la base de datos* que se abre, eligiendo el conjunto apropiado en la pestaña *Configuración adicional*, en la lista desplegable *Conjunto de caracteres* de la sección *Conversión de datos* (figura 32).

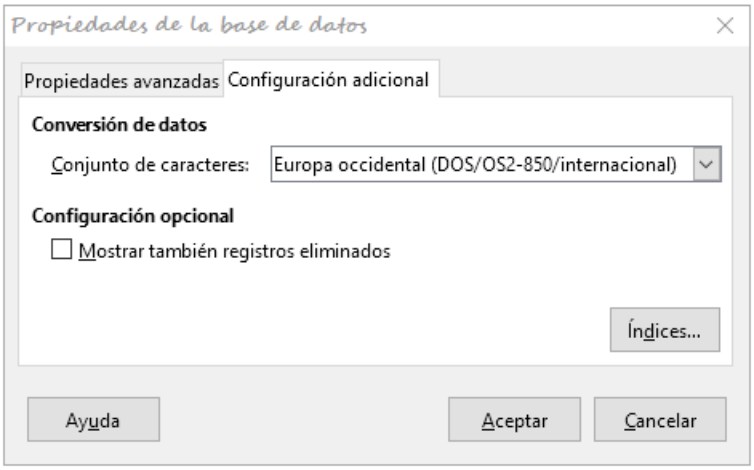


Figura 32: Modificación de la codificación del juego de caracteres

Nota

El *Asistente de importación* para dBase tiene problemas con el reconocimiento automático de los tipos de campo numérico y los campos booleanos (Sí / No) (bug 53027). Esto puede obligarle a que realice correcciones posteriores.

Hojas de cálculo

Las hojas de cálculo Calc o Excel también se pueden usar como fuente de tabla para bases de datos. Sin embargo, si se utiliza una hoja de cálculo no es posible editar los datos de la tabla. Si la hoja de cálculo aún está abierta, estará protegida contra escritura.

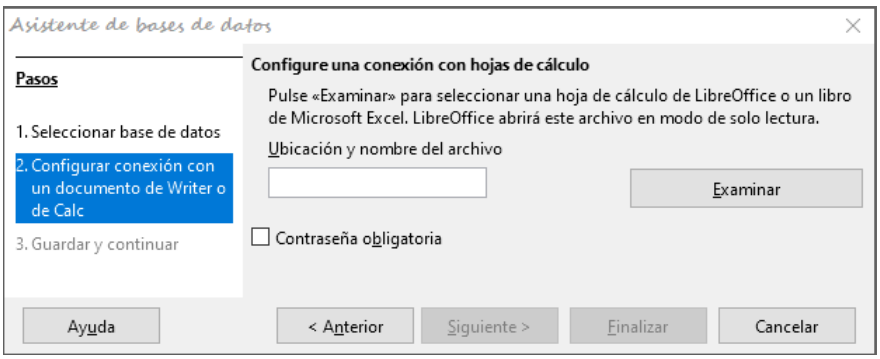


Figura 33: Configurar una conexión a una hoja de cálculo

Las únicas preguntas que deben responderse son la ubicación del archivo de hoja de cálculo y si está protegido con contraseña o no. Luego, Base abre la hoja de cálculo e incluye todas las hojas de trabajo en el documento. La primera fila se usa para los nombres de campo y los nombres de la hoja de trabajo se convierten en los nombres de la tabla.

Las relaciones entre hojas de cálculo no se pueden configurar en Base, ya que ni las hojas de cálculo de Calc ni las de Excel no son adecuadas para su uso como base de datos relacional.

Libreta de direcciones de Thunderbird

El asistente buscará automáticamente una conexión a una libreta de direcciones, por ejemplo, como se usa en Thunderbird. El asistente le solicitará la ubicación del archivo ODB que se generará.

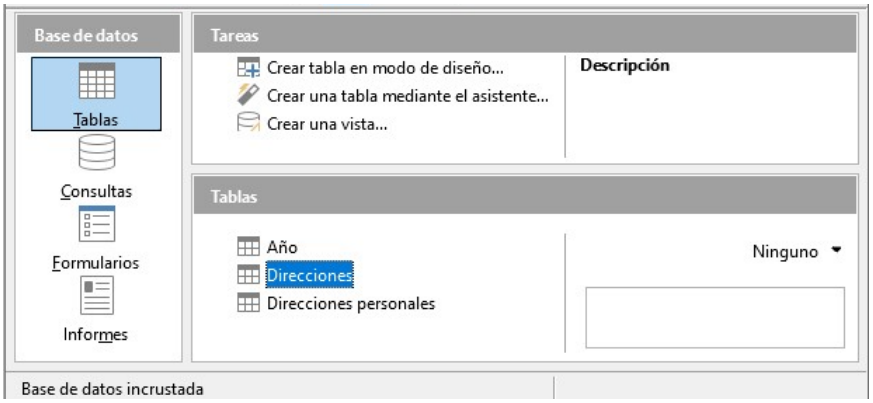


Figura 34: Tablas en una libreta de direcciones de Thunderbird

En el panel *Tablas* aparecerán todas las tablas. Al igual que con las hojas de cálculo, las tablas no se pueden editar. Base utiliza los datos de la tabla solo para consultas y aplicaciones de combinación de correspondencia.

Nota

Solo el archivo de direcciones personales se lee como una libreta de direcciones en Linux y macOS. Los grupos recopilados actualmente solo son visibles como parte de las direcciones personales. Los grupos de direcciones recopiladas no se muestran.

Tablas de texto

En Base puede crear una base de datos completa accediendo a tablas de texto. También se puede acceder a las tablas de texto desde una base de datos interna.

El formato CSV (*comma separated values*, o valores separados por comas) es un formato de intercambio común entre bases de datos. Los registros se almacenan en una forma que puede ser

leída y modificada por un simple editor de texto. Los campos individuales están separados por comas. Si un campo contiene texto que incluye una coma, los campos de texto están encerrados entre comillas dobles. Cada nuevo registro comienza con una nueva línea.

Por ejemplo, el contenido de una libreta de direcciones que está en un formato no admitido por ningún otro controlador de Base puede importarse a través de un archivo `.csv` (usando Calc como intermediario si es necesario) o importando directamente el archivo a la base de datos como tabla de texto Para poder editar los datos posteriormente, el archivo `.csv` debe incluir un campo con valores únicos que puedan servir como clave principal.

Tablas de texto dentro de una base de datos interna HSQLDB

Para crear una tabla de texto, cree una base de datos nueva en un directorio reservado para esa base de datos. Asegúrese que la nueva base de datos ODB tenga el motor integrado HSQLDB. Las tablas de texto son una funcionalidad exclusiva del motor HSQLDB, la cual no está presente en Firebird. En ese directorio también guarde el archivo `.cvs` que se va a importar.

No se puede crear una tabla de texto usando la interfaz gráfica de usuario[3]. En su lugar, debe usar **Herramientas > SQL…** para crear una tabla de texto mediante órdenes directas de SQL (vea la figura 35). Los campos en la instrucción de SQL para la tabla de texto deben corresponder en tipo y orden a los que la tabla de texto CSV pone a disposición. Por ejemplo, el campo *ID* debe contener enteros positivos y el campo *Nacimiento* debe contener valores de fecha en el formato Año-Mes-Día.

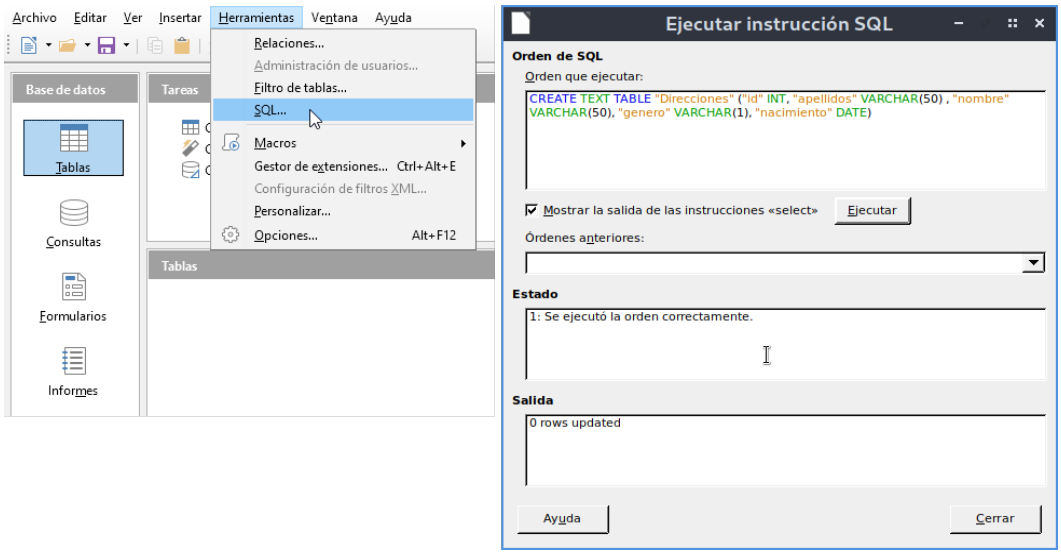


Figura 35: Crear una tabla de texto usando Herramientas>SQL…

Una vez ejecutada la instrucción SQL, la tabla no será visible en la interfaz de usuario. Para verla en el panel *Tablas*, solo es necesario refrescar el panel con **Ver > Actualizar tablas** para que la tabla recién creada esté visible y disponible. El icono de la tabla indica que esta no es una tabla «normal» de base de datos (figura 36).

3 Consulte la base de datos complementaria de esta guía, `Ejemplo_importacion_CSV.odb`.

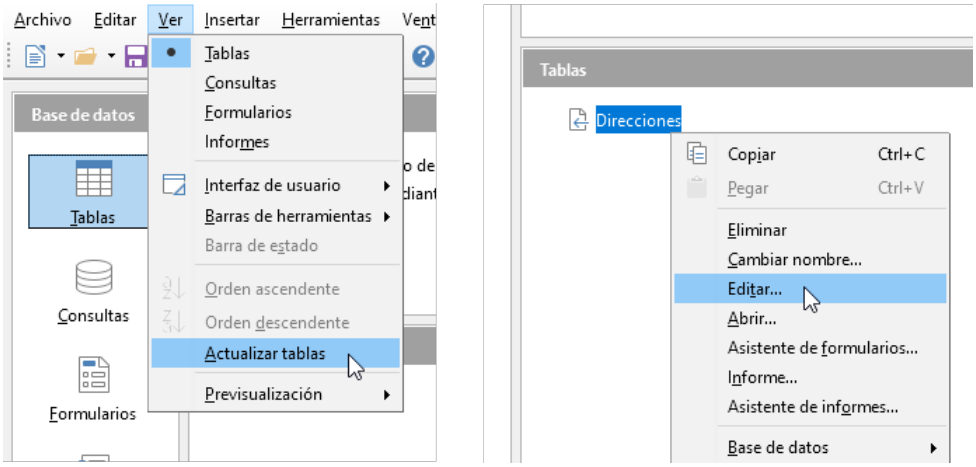


Figura 36: Tabla de conexión para un archivo CSV externo. Note el icono diferente al de una tabla normal

Abra la tabla para editar y asegúrese que la propiedad *Valor automático* del campo "ID" se incrementará automáticamente (figura 37).

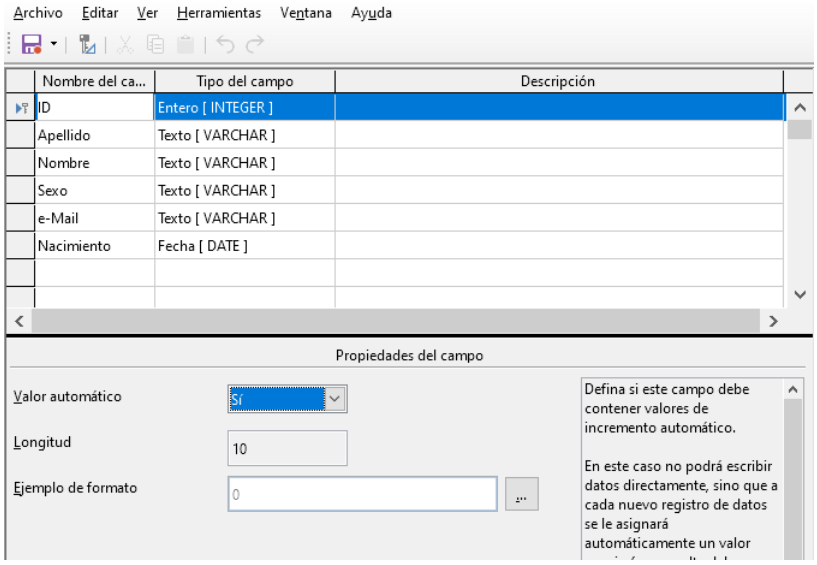


Figura 37: Editar una tabla de texto

Ahora debe hacer una conexión de la tabla recién creada a la tabla de texto CSV externa. Para ello, use el mismo menú **Herramientas > SQL…**. La tabla de texto externa debe estar en la misma carpeta o directorio en la que está la base de datos donde está realizando las instrucciones SQL.

```
SET TABLE "Direcciones" SOURCE "Direcciones.csv;encoding=UTF-8"; [4]
```

4 La codificación de caracteres (*encoding*) UTF-8 funciona en muchos sistemas operativos. En algunos casos deberá usar otra codificación en lugar de UTF-8.

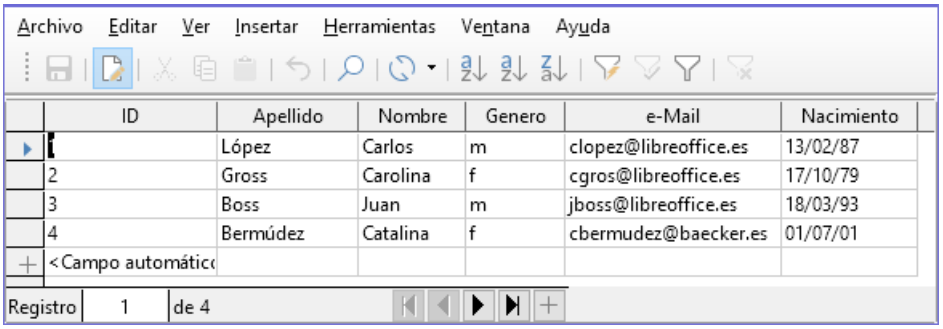


Figura 38: Tabla de texto conectada a un archivo CSV externo

Después de esto, la tabla de texto estará disponible para la entrada de la manera normal (figura 38). Pero deben tenerse en cuenta los siguientes puntos:

- Las tablas de texto se pueden abrir y editar simultáneamente mediante programas de texto externos. La pérdida de datos no puede excluirse en estas circunstancias.
- Los cambios en los registros ya escritos originan que se borre la línea correspondiente en el archivo original y se agregue la nueva versión al final de la tabla. La vista de tabla que se muestra arriba presenta cuatro líneas escritas con números de identificación correctamente ordenados. En el archivo original, el segundo registro ha sido alterado, lo que lleva a la siguiente secuencia de registros por ID: 1, línea en blanco, 3, 4, 2.

Cuando se conecta a un archivo de texto, los siguientes parámetros están disponibles (toda la instrucción debe ir en un solo renglón).

```
SET TABLE "Direcciones" SOURCE "Direcciones.csv;ignore_first=false;all_quoted=true;
encoding=UTF-8";
```

- ignore_first = true significaría que la primera línea no se lee. Esto tiene sentido si la primera línea contiene solo encabezados de campo. El valor predeterminado interno para HSQLDB es false.
- De manera predeterminada, los campos de texto HSQLDB solo se colocan entre comillas dobles si contienen una coma interna, ya que la coma es el separador de campo predeterminado. Si se va a citar cada campo, establezca all_quoted = true.

Para saber sobre más estos parámetros, vea la sección correspondiente en la página de HSQLDB: http://www.hsqldb.org/doc/1.8/guide/guide.html#set_table_source-section

SET TABLE "Direcciones" READONLY TRUE evita que algo se escriba en la tabla. La tabla estará entonces disponible como solo lectura, igual que una libreta de direcciones de un programa de correo. Establecer la protección contra escritura por separado solo es necesario cuando se haya establecido una clave primaria para la tabla.

Tablas de texto como base para una base de datos independiente

Como en el ejemplo anterior, los archivos `*.csv` se utilizan como fuente de datos. Si se usa Base, una carpeta o directorio que contenga los archivos `*.csv` se agregará como una carpeta de datos.

Comenzaremos conectándonos a una base de datos existente. Aquí se ha seleccionado el formato *Texto*.

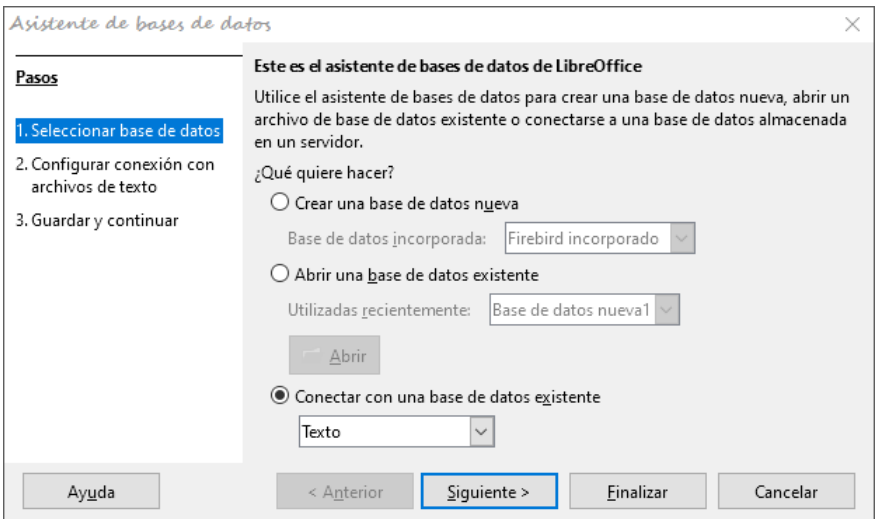


Figura 39: Paso 1 — Conexión con una base de datos conformada por archivos de tipo Texto

Al pulsar *Siguiente*, en el *Paso 2* debe establecer la ruta a la carpeta o directorio donde están los archivos de texto (aunque en realidad debe elegir un archivo para seleccionar la ruta). En la carpeta elegida, todos los archivos del tipo especificado (CSV, TXT o alguno personalizado) se enumerarán más adelante en el panel *Tablas*, al crear la nueva base de datos ODB. Para archivos `*.csv`, elija la opción correspondiente en el diálogo que aparecerá a continuación.

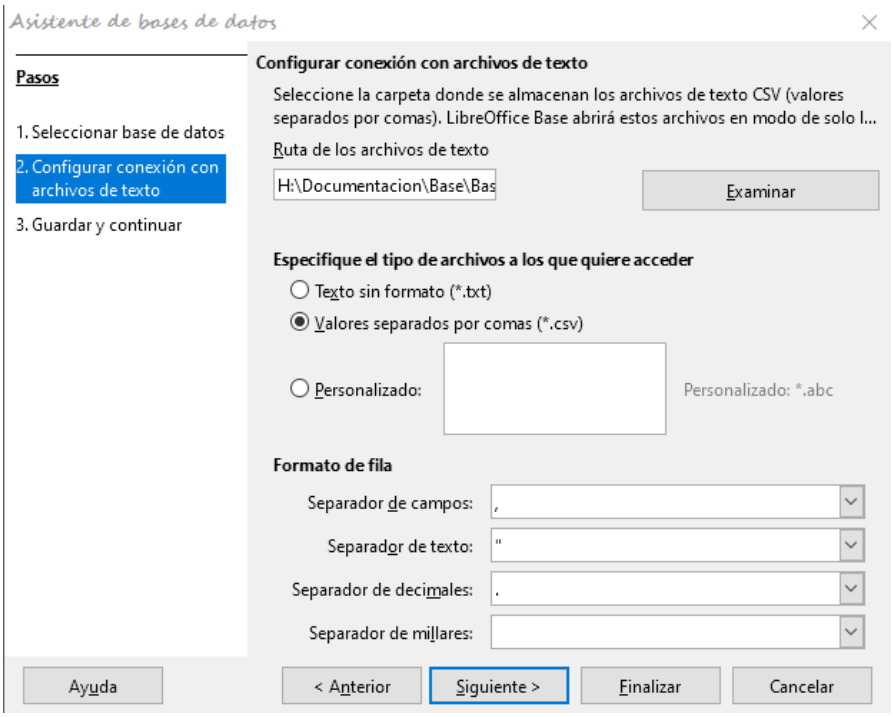


Figura 40: Paso 2 — Elección de ruta, tipo de archivo y formato de fila para base de datos conformada por archivos de tipo Texto

En esta etapa, ya puede ver una advertencia. Los archivos se abrirán en modo de solo lectura, sin acceso de escritura.

Configure los parámetros del *Formato de fila* de acuerdo al archivo seleccionado.

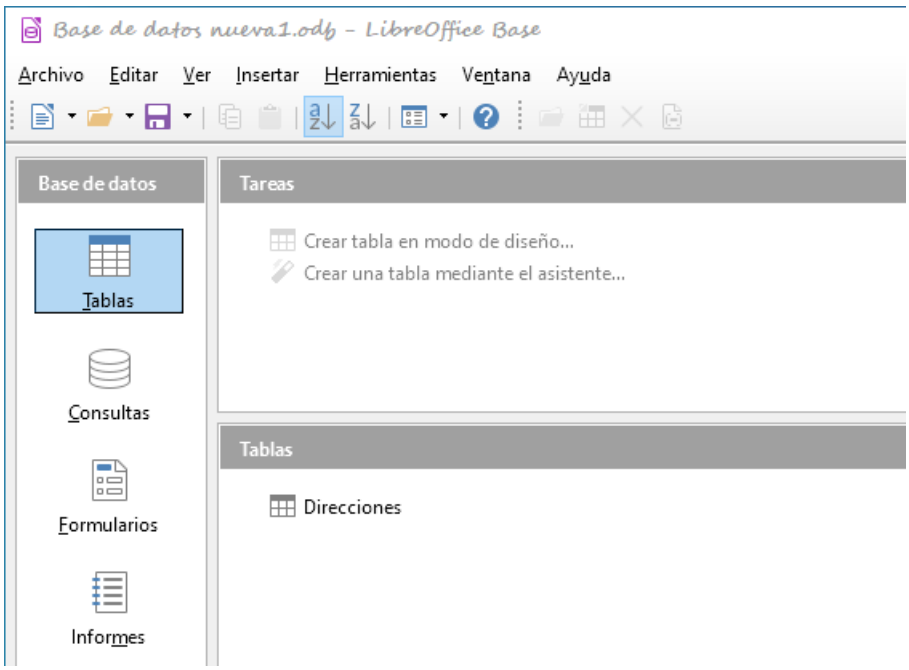


Figura 41: Panel Tablas mostrando los archivo de tipo Texto, presentes en la ruta elegida, como tablas de la base de datos

En el panel *Tablas*, todos los archivos de la carpeta o directorio especificado que son tablas de texto se muestran por sus nombres de archivo, pero sin el sufijo del tipo de archivo. Las tareas para crear tablas no están activas. Las tablas en sí pueden leerse pero no escribirse.

El acceso a las tablas mediante consultas también se limita a una tabla y sin el uso de funciones.

Cuando esta base de datos se utiliza para buscar registros en un archivo `*.csv` o para importar un archivo `*.csv` a otra base de datos mediante la función de copia, cumple su bien propósito. El archivo `*.csv` correspondiente solo se mueve a la carpeta especificada y se puede buscar o copiar directamente. Dichas bases de datos de texto no son adecuadas para un uso más general.

Firebird

En algún momento, la antigua versión de HSQLDB utilizada internamente para bases de datos será reemplazada por una base de datos interna de Firebird. Si desea ver lo que Firebird puede ofrecer, a continuación verá el procedimiento para conectarse a una base de datos externa de Firebird.

La documentación de Firebird no es tan completa como en el caso de MySQL o PostgreSQL. Los que siguen son los pasos más importantes en la instalación.

Crear un usuario y una base de datos

Linux proporciona paquetes de Firebird a través de sus administradores de paquetes. Después de la instalación, el servidor debe estar configurado. Los siguientes pasos en OpenSUSE 12.3 enlazan a una base de datos de Firebird en funcionamiento:

1) `sysdba` es el nombre del usuario administrador (superusuario). La contraseña predeterminada es `masterkey`, que debe ser cambiada en un entorno de producción.
2) Para cambiar la contraseña en una terminal o consola, introduzca:

```
gsec -user sysdba -pass masterkey -mo sysdba -pw newpassword
```

3) Para acceder a una base de datos que funcione, necesita derechos de administrador en la computadora. El usuario de sistema `firebird` debe tener una contraseña asignada previamente.

4) Inicie sesión con el usuario `firebird` en la terminal o consola de la siguiente manera:

```
su firebird
```

5) Se crea un nuevo usuario, que se muestra aquí con la contraseña predeterminada original para `sdba`:

```
gsec -user sysdba -pass masterkey -add lotest -pw libre
```

Esto crea un nuevo usuario `lotest`, cuya contraseña es `libre`.

6) A continuación, aún como superusuario, cree una base de datos para el usuario. Para esto utilizamos el programa auxiliar `isql-fb`. Teclee en la terminal o consola:

```
isql-fb
```

Verá el siguiente mensaje:

```
Use CONNECT or CREATE DATABASE to specify a database
followed directly by the SQL> prompt.
```

Lo que debe teclear en la consola, en el *prompt* de Firebird, es:

```
SQL> CREATE DATABASE 'libretest.fdb'
CON> user 'lotest' password 'libre';
```

Si estas tareas se llevan a cabo como administrador del sistema (`root`), la base de datos no se asignará al usuario correcto cuando esté en red. Se debe ingresar a la base de datos con el usuario `firebird` que pertenece al grupo `firebird`. De lo contrario, no funcionará la conexión posteriormente.

Conexión a Firebird a través de JDBC

Primero deberá incrustar el archivo `jar` en LibreOffice. Sin embargo, no hay un archivo llamado `firebird-*jar.` El controlador JDBC se puede encontrar en http://www.firebirdsql.org/en/jdbc-driver/. El nombre del controlador comienza con `jaybird`…

Descomprima el archivo `jaybird-full-2.2.8.zip` (o el que corresponda a la versión actualizada) y coloque el archivo `jaybird-full-2.2.8.jar` en la ruta de la instalación de Java o impórtelo directamente en LibreOffice como archivo. Vea la sección correspondiente en MySQL.

Al instalar JDBC, los siguientes parámetros son importantes:

JDBC URL	`jdbc:firsql://host[:port]/<path_or_alias>`
Nombre del controlador:	`org.firg.firebirdsql.jdbc.FBDriver`

En el ejemplo anterior, esto se convierte en la URL:

```
jdbc:firebirdsql://localhost/libretest.fdb?charSet=UTF-8
```

Si no especifica la codificación o juego de caracteres, obtendrá el error mostrado en la figura 42:

Figura 42: Mensaje de error resultante de no especificar el juego de caracteres apropiado al conectar una base de datos

Al crear tablas, tenga cuidado de que el formato de los campos correspondientes (propiedades de campo) coincida desde el principio. De lo contrario, LibreOffice establecerá el formato predeterminado para todos los valores numéricos que, curiosamente, es de tipo *Moneda*.

No es posible la alteración posterior de las propiedades de campo en las tablas, pero puede ampliar la tabla o eliminar campos.

Conexión Firebird usando ODBC

Primero debe descargar el controlador ODBC apropiado de: http://www.firebirdsql.org/en/odbc-driver/. Este controlador suele ser un archivo normal llamado `libOdbcFb.so`.

Este archivo generalmente se coloca en una ruta accesible en el sistema y debe tener permisos de ejecución.

En los archivos `odbcinst.ini` y `odbc.ini`, que son los archivos de configuración necesarios para el sistema, se necesitan las siguientes entradas:

`odbinst.ini:`

```
[Firebird]
Description = Firebird ODBC driver
Driver64 = /usr/lib64/libOdbcFb.so
```

`odbc.ini:`

```
[Firebird-libretest]
Description = Firebird database libreoffice test
Driver = Firebird
Dbname = localhost:/srv/firebird/libretest.fdb
SensitiveIdentifier = Yes
```

En un sistema Linux, estos dos archivos deben estar en la carpeta o directorio `/etc /unixODBC`.

La variable `SensitiveIdentifier` siempre debe establecerse en `"Yes"` para que la entrada en tablas funcione cuando los nombres y las definiciones de campo no están en mayúsculas.

Edición posterior de las propiedades de conexión

Puede que una conexión no funcione como se desea, especialmente con conexiones a bases de datos externas. Es posible que el conjunto de caracteres no sea el correcto o que los subformularios funcionen con errores, o que algo en los parámetros subyacentes tenga que cambiarse.

Las siguientes capturas de pantalla ilustran cómo puede cambiar los parámetros de conexión para una base de datos PostgreSQL externa.

En **Editar > Base de datos**, encontrará las opciones *Propiedades…*, *Tipo de conexión…* y *Configuración avanzada…* Elija *Propiedades…* (figura 43).

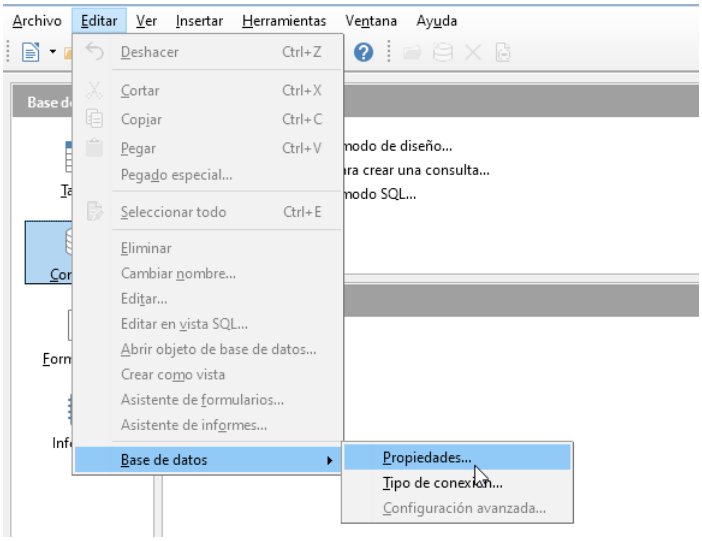


Figura 43: Menú Base de datos para modificar las propiedades de la base de datos

Si el nombre de la fuente de datos ha cambiado, se puede modificar aquí (figura 44). Para una conexión ODBC, el nombre por el que se llama a la base de datos se establece en el archivo `odbc.ini`. El nombre generalmente no es el mismo que el nombre real de la base de datos en PostgreSQL.

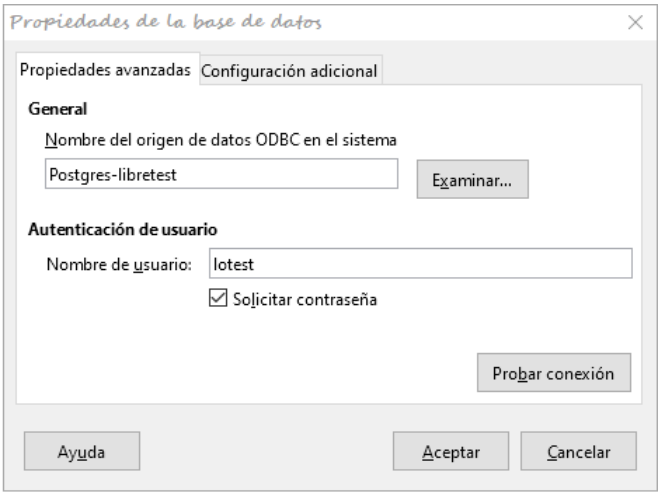


Figura 44: Pestaña Propiedades avanzadas del diálogo Propiedades de la base de datos

¿Hay algún problema con el juego de caracteres? Estos problemas se pueden resolver utilizando la pestaña *Configuración adicional* del diálogo *Propiedades de la base de datos* (figuras 45 y 46).

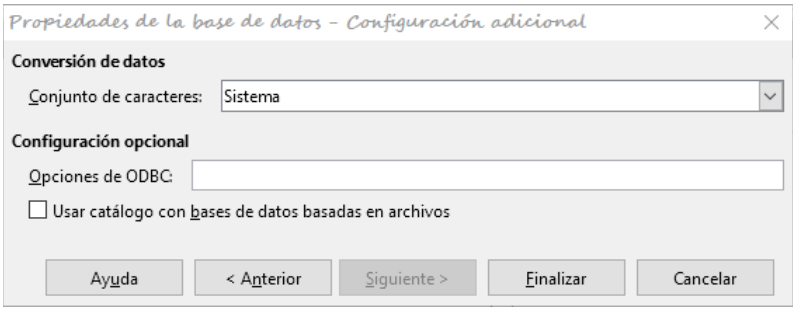


Figura 45: Pestaña Configuración adicional del diálogo Propiedades de la base de datos

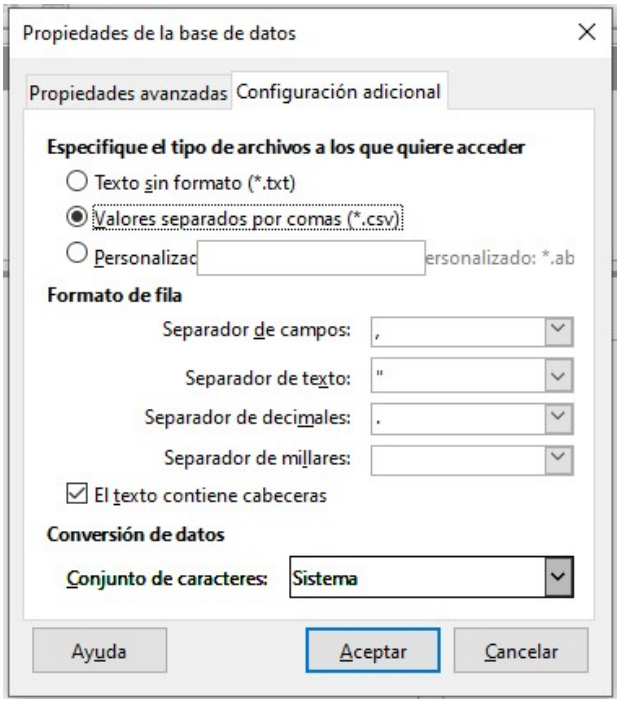


Figura 46: Pestaña Configuración adicional del diálogo Propiedades de la base de datos

Es posible crear una configuración especial adicional del controlador, si fuera necesario implementar un parámetro que no está actualmente en el archivo `odbc.ini` (figura 47).

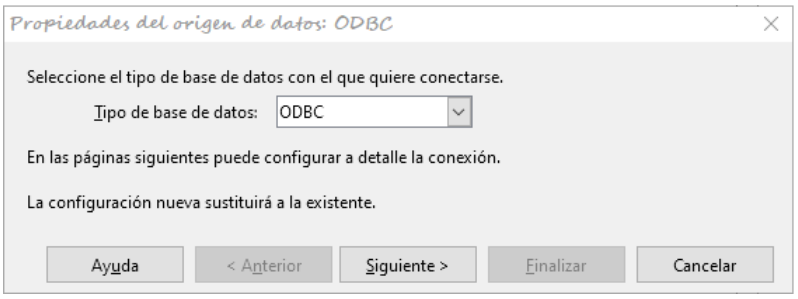


Figura 47: Diálogo Propiedades del origen de datos

Si se selecciona y cambia el tipo de conexión, se puede alterar todo el contacto con la fuente de datos.

Los siguientes pasos son similares a los del *Asistente para bases de datos*, a partir del *Paso 2* en adelante. Así, por ejemplo, puede cambiar de ODBC a JDBC o una conexión directa con el controlador interno de LibreOffice. Esto es útil si está haciendo pruebas preliminares para determinar qué método de conexión es el más adecuado para un proyecto.

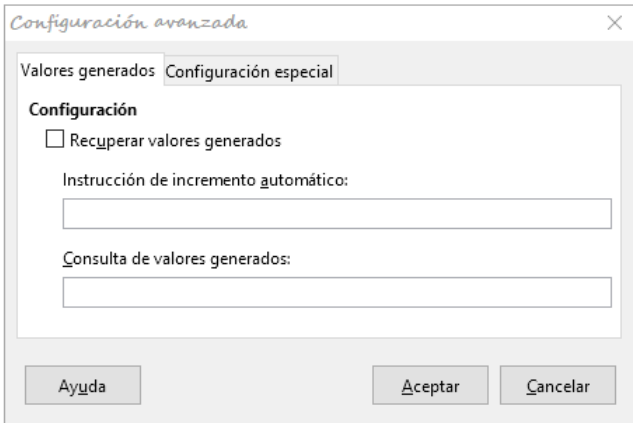


Figure 48: Diálogo Configuración avanzada

En función del sistema usado en cada de base de datos, existen diferentes instrucciones para crear valores de incremento automático.

Si necesita hacer algo similar y no es posible con este controlador, deberá hacerlo manualmente. Esto requerirá una instrucción para crear un campo de incremento automático y otra para consultar el valor más reciente.

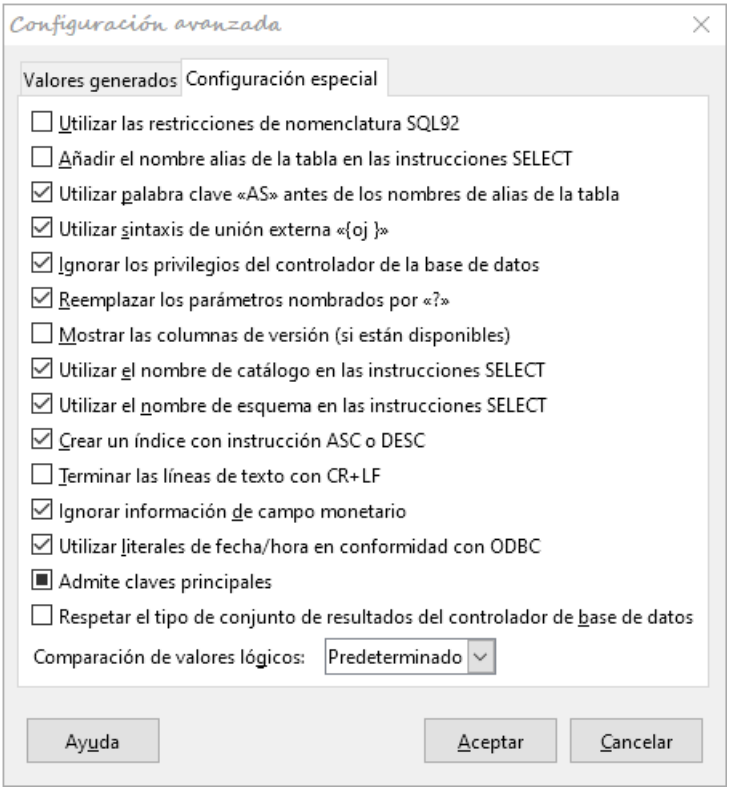


Figure 49: Pestaña Configuración especial del diálogo Configuración avanzada

La pestaña *Configuración especial*, accesible a través de **Herramientas > Base de datos> Configuración avanzada…**, afecta la interacción de bases de datos externas con Base de varias maneras.

Algunas opciones aparecen atenuadas, ya que no se pueden cambiar en la base de datos subyacente. En el ejemplo anterior, *Reemplazar los parámetros nombrados por «?»* se ha marcado. Está demostrado que, de lo contrario, la transmisión de valores de un formulario principal a un subformulario en PostgreSQL no funciona. Solo con esta configuración, la construcción de formularios en el “Capítulo 4, Formularios” ya funcionará correctamente.

LibreOffice

Guía de Base

Capítulo 3
Tablas

Información general sobre tablas

Las bases de datos almacenan datos en tablas. La principal diferencia con las tablas en una hoja de cálculo normal es que los campos en los que se escriben los datos deben estar claramente definidos de antemano. Por ejemplo, una base de datos no permite que un campo de texto contenga números para usar en los cálculos. Dichos números se muestran, pero solo como cadenas, cuyo valor numérico real es cero. Del mismo modo, las imágenes no se pueden incluir en todos los tipos de campos.

Los detalles sobre qué tipos de datos están disponibles, se pueden obtener en Base desde la ventana Diseño de tabla. Se muestran en el «Apéndice» de esta guía.

Las bases de datos simples se basan en una sola tabla. Todos los elementos de datos se ingresan de forma independiente, lo que puede conducir a la entrada múltiple de los mismos datos. De este modo, se puede crear una libreta de direcciones simple para uso privado. Sin embargo, la libreta de direcciones de una escuela o una asociación deportiva podría contener tantas repeticiónes de códigos postales y calles que estos datos se ubican mejor en una o incluso dos tablas separadas.

Almacenar datos en tablas separadas ayuda a:

- Reducir la entrada repetida del mismo contenido.
- Evitar errores de ortografía debido a la entrada repetida.
- Mejora el filtrado de datos en las tablas mostradas.

Al crear una tabla, siempre debe considerar si se van a producir múltiples repeticiones, especialmente de texto o imágenes (que consumen mucho almacenamiento) en la tabla. Si es así, debe exportarlos a otra tabla. Cómo hacerlo, en principio, se describe en el «Capítulo 1» de *Introducción a Base*, en la sección «Una base de datos simple: ejemplo de prueba en detalle».

Nota

Una base de datos relacional es un grupo de tablas que están vinculadas entre sí a través de atributos comunes. El propósito de una base de datos relacional es en lo máximo la entrada duplicada de elementos de datos. Se deben evitar redundancias.

Esto se puede lograr al:

• Separar el contenido en tantos campos únicos como sea práctico (por ejemplo, en lugar de usar un campo para una dirección completa, use campos separados para el número de casa, calle, ciudad y código postal).

• Evitar datos duplicados para un campo en varios registros (por ejemplo, importando el código postal y la ciudad desde otra tabla).

Estos procedimientos se conocen como *Normalización de la base de datos*.

Relaciones entre tablas

Este capítulo explica muchos de estos pasos en detalle, utilizando una base de datos de ejemplo para una biblioteca: **`media_without_macros`**. La construcción de las tablas para esta base de datos es un trabajo extenso, ya que cubre no solo la adición de elementos a una biblioteca, sino también el préstamo posterior de los mismos.

Relaciones entre tablas en Bases de Datos

Las tablas en una base de datos interna HSQLDB siempre tienen un campo único y distintivo: la *clave primaria*. Este campo se tiene que definir antes de que se puedan escribir datos en la tabla. Usando este campo, se pueden encontrar registros concretos en una tabla.

En ciertos casos, se forma una clave primaria a partir de varios campos juntos. Estos campos tienen que ser únicos. Esto se conoce como una *clave primaria compuesta*.

La *Tabla 2* puede tener un campo que refiere a un registro en la *Tabla 1*. La clave primaria de la *Tabla 1* se escribe como un valor en el campo de la *Tabla 2*.

La *Tabla 2* ahora tiene un campo que apunta al campo clave de otra tabla, conocido como *clave externa*. Esta clave externa existe en la *Tabla 2* además de su clave primaria.

Cuantas más relaciones haya entre tablas, más compleja será la tarea de diseño.

La Figura 50 muestra la estructura general de la base de datos de ejemplo. Para visualizar mejor su contenido puede ampliar la página.

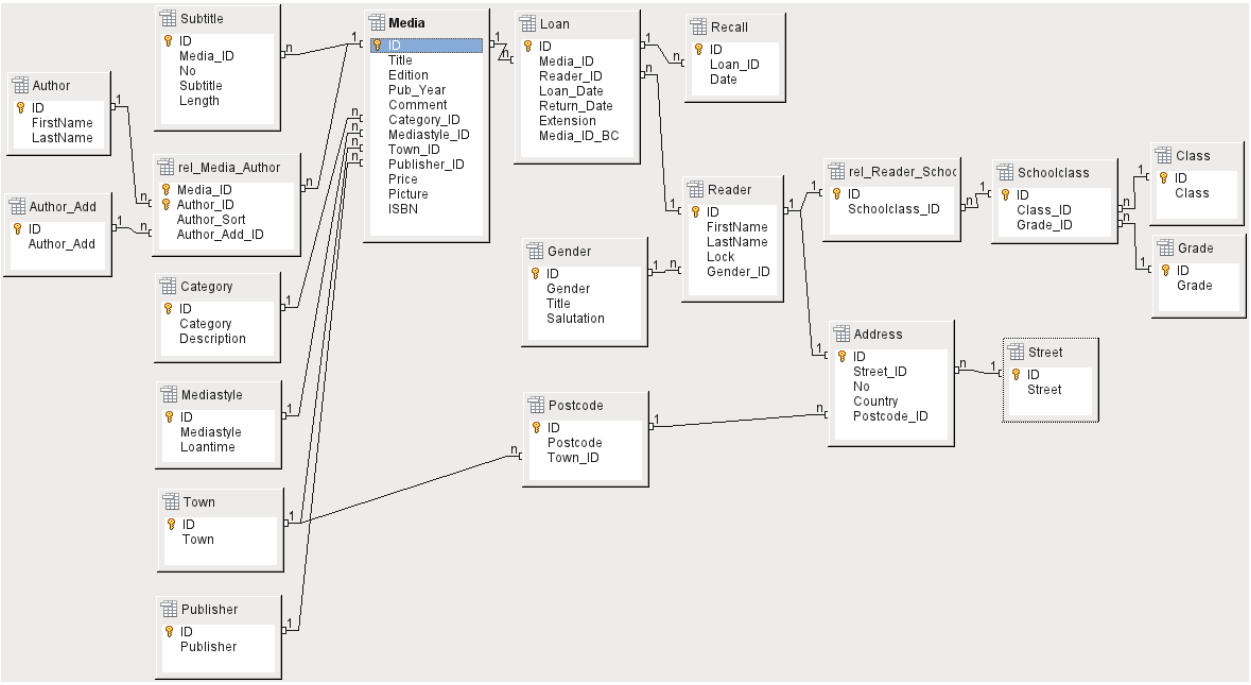


Figura 50: Diagrama de Relaciones en la base de datos de ejemplo: `media_without_macros`

Relaciones uno a muchos

La base de datos **`media_without_macros`** de ejemplo, enumera los títulos de los artículos, en una tabla. Debido a que los títulos pueden tener múltiples subtítulos o, a veces, ninguno, los subtítulos se almacenan en una tabla separada.

Esta relación se conoce como *uno a muchos* (`1:n`). Se pueden asignar muchos subtítulos a un artículo, por ejemplo, los títulos de muchas pistas para un CD de música. La clave principal para la tabla de artículos (*Media*) se almacena como una clave externa en la tabla de subtítulos (*Subtitle*). La mayoría de las relaciones entre tablas en una base de datos son relaciones de uno a muchos.

Relaciones de muchos a muchos

Una base de datos para una biblioteca puede contener una tabla para los nombres de los autores y una tabla para sus obras, denominada *Media*. La conexión entre un autor y los libros que ha escrito, es obvia. La biblioteca puede contener más de un libro de un autor. Pero también puede contener libros con el mismo título pero de diferentes autores. Esta relación se conoce como *muchos a muchos* (`n:m`). Dichas relaciones necesitan una tabla que actúe como intermediaria entre las dos tablas en cuestión. En la Figura 51, la tabla *rel_Media_Author* es una tabla intermedia.

Por lo tanto, en la práctica, la relación `n:m` se resuelve al tratarla como dos relaciones `1:n`. En la tabla intermedia, *Media_ID* puede aparecer más de una vez, al igual que *Author_ID*. Pero al

usarlos como un par, no hay duplicación: no hay dos pares idénticos. Por lo tanto, este par cumple los requisitos de una clave primaria para la tabla intermedia.

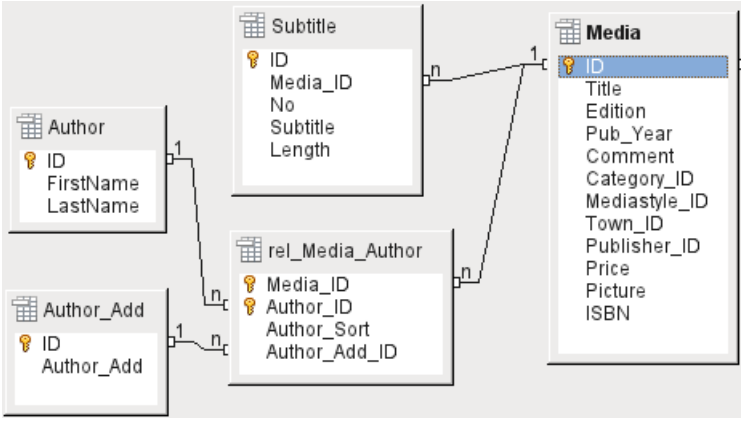


Figura 51: Ejemplo relación 1: n; y relación n: m

Nota

Un autor es único al tener un nombre y apellido (aunque puede darse alguna rara coincidencia), así mismo un libro también es único al tener un número ISBN. Para un par determinado de estos valores, estas características hace que el par sea único.

Relaciones uno a uno

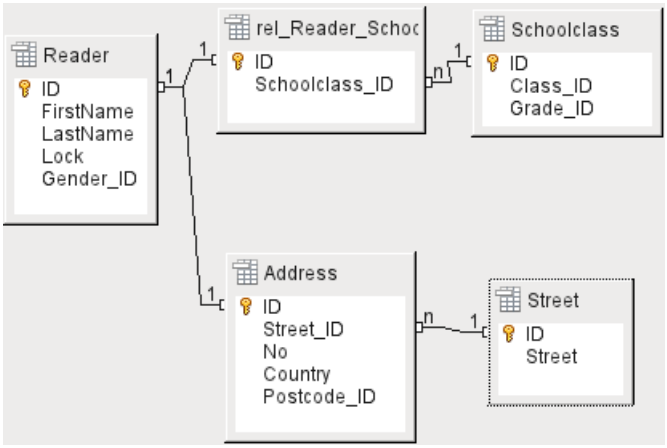


Figura 52: Ejemplo de relación 1: 1

La base de datos de la biblioteca, descrita anteriormente, requiere una tabla para lectores. En esta tabla solo se planificaron por adelantado los campos que son directamente necesarios.

Pero para una base de datos de una escuela, también se puede necesitar la clase escolar a la que pertenece el lector. Con los registros de la tabla, *Schoolclass*, se pueden encontrar las direcciones de los prestatarios cuando sea necesario. Por lo tanto, no es necesario incluir estas direcciones en la base de datos.

La relación entre la tabla de las clases escolares de los alumnos *Schoolclass* está separada de la tabla del lector, *Reader*, porque la asignación de las clases escolares no es siempre apropiada. De esto surge una relación `1:1` entre el lector y la clase escolar individual asignada.

En una base de datos para una biblioteca pública, se requieren las direcciones de los lectores. Para cada lector hay una sola dirección. Si hay varios lectores en la misma dirección, esta estructura requeriría que la dirección se ingrese nuevamente, ya que la clave primaria de la tabla *Reader* se ingresa directamente como clave principal en la tabla *Address*. La clave primaria y la clave externa son una y la misma en la tabla de direcciones. Por lo tanto esta es una relación `1:1`.

Una relación `1:1` no significa que para cada registro en una tabla haya un registro correspondiente en otra tabla. Al menos habrá un único un registro correspondiente. Por lo tanto una relación `1:1` lleva a la exportación de campos que se completarán con contenido solo para algunos de los registros.

Tablas y relaciones en la base de datos de ejemplo

La base de datos de ejemplo `media_without_macros.odb` debe atender a tres necesidades: adición y eliminación de artículos, gestión de préstamos y administración de usuarios.

Tabla de adición de artículos (Media)

Primero deben agregarse los artículos a la base de datos para que una biblioteca pueda trabajar con ellos. Sin embargo, para un resumen simple de una colección de artículos en casa, puede crear bases de datos más fáciles con el asistente; eso podría ser suficiente para uso doméstico.

La tabla central para la adición de artículos es la tabla *Media* (ver Figura 53).

En esta tabla, se supone que todos los campos que se ingresan directamente no se usan también para otros artículos con el mismo contenido. Por lo tanto, se debe evitar la duplicación.

Por esta razón, los campos planificados en la tabla incluyen el título, el ISBN, una imagen de la portada y el año de publicación. La lista de campos se puede ampliar si es necesario. En este caso los administradores de la biblioteca pueden querer incluir, por ejemplo, campos para el tamaño (número de páginas), el título de la serie, etc.

La tabla *Subtitle* contiene el contenido detallado de los CD. Como un CD puede contener varias piezas de música, un registro de las piezas individuales en la tabla principal requeriría muchos campos adicionales (*Subtítle_1*, *Subtítle_2*, etc.) o el mismo elemento tendría que ingresarse muchas veces. La tabla *Subtitle*, por lo tanto, se encuentra en una relación `n:1` con la tabla *Media*.

Los campos de la tabla *Subtitle* son (además del subtítulo en sí) el número de secuencia del subtítulo y la duración de la pista. El campo *Length* debe definirse primero como un campo de tiempo. De esta manera, la duración total del CD se puede calcular y mostrar en un resumen si fuese necesario.

Los autores tienen una relación `n:m` con los artículos. Un artículo puede tener varios autores, y un autor puede haber creado varios artículos. Esta relación está controlada por la tabla *rel_Media_Author*. La clave principal de esta tabla de vinculación es la clave externa, formada a partir de las tablas *Author* y *Media*. La tabla *rel_Media_Author* incluye una ordenación adicional (`Author_Sort`) de autores, por ejemplo, por la secuencia en la que se nombran en el artículo. Además, se agrega una etiqueta complementaria como Productor, Fotógrafo, etc. al autor cuando sea necesario.

Category, *Mediastyle*, *Town*, y *Publisher* tienen una relación `1:n`.

Para *Category*, una pequeña biblioteca puede usar algo como Arte o Biología. Para bibliotecas más grandes, están disponibles sistemas generales para bibliotecas. Estos sistemas proporcionan abreviaturas y descripciones completas. Por lo tanto, ambos campos aparecen en la tabla *Category*.

El *Mediastyle* está vinculado al período de préstamo en tiempo real. Por ejemplo, los DVD de video pueden, en principio, tener un período de préstamo de 7 días, pero los libros pueden prestarse por 21 días. Si el período del préstamo se quiere vincular a algún otro criterio, habrá que hacer los cambios oportunos en su metodología.

La tabla *Town* no solo sirve para almacenar datos de ubicación de los artículos, sino también para almacenar las ciudades usadas en las direcciones de los usuarios.

Dado que *Publishers* también se repiten con frecuencia, se les proporciona una tabla separada.

La tabla *Media* tiene en total cuatro claves externas y una clave primaria, que se utiliza como clave externa en dos tablas, como se muestra en la Figura 53.

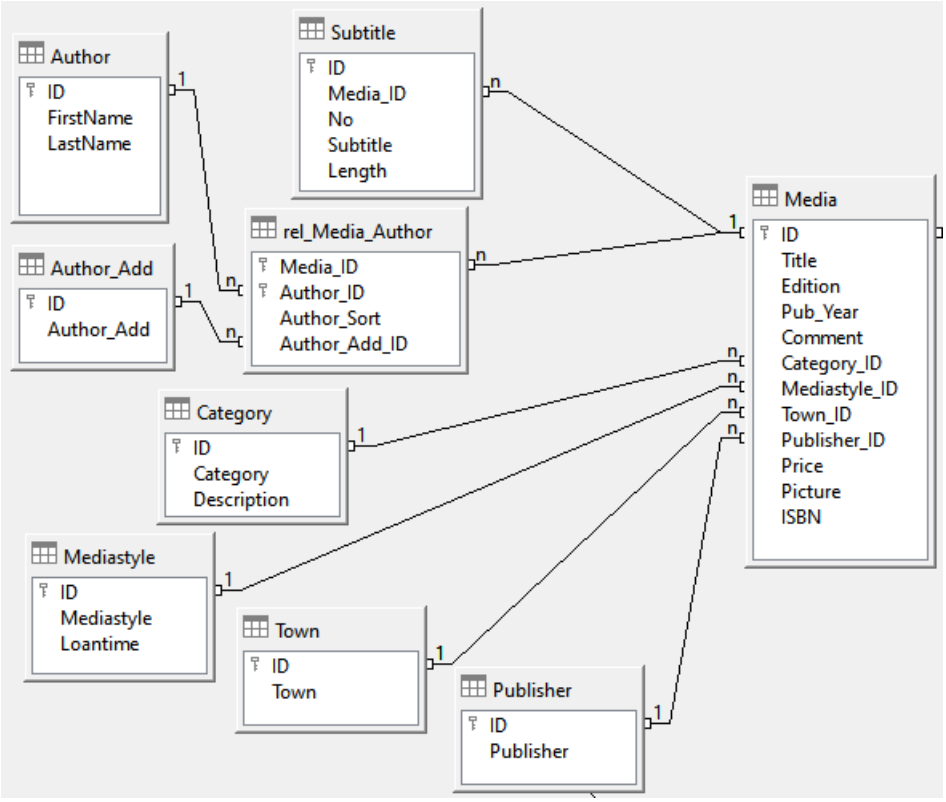


Figura 53: Adición de artículos (Media)

Tabla de préstamos (Loan)

La tabla central es *Loan* (ver Figura 54). Es el enlace entre las tablas *Media* y *Reader*.

Cuando se devuelve un artículo, muchos de sus datos se pueden eliminar ya que ya no son necesarios. Pero dos de esos campos no deberían ser eliminados: *ID* y *Loan_Date*. La primera es la clave principal. El segundo indica cuándo se prestó el artículo. Tiene dos propósitos. Primero, es útil para determinar los artículos más populares. En segundo lugar, si se observa que el artículo está dañado cuando se entrega de nuevo, este campo mostrará quién fue la última persona que lo tomó prestado. Además, el *Return_Date* se registra cuando se devuelve el artículo.

Del mismo modo, los avisos de vencimiento de préstamo se integran en el procedimiento de préstamo. Cada aviso se ingresa por separado en la tabla *Recall* para que se pueda determinar el número total de avisos.

Además de un período de extensión en semanas, hay un campo adicional en el registro del préstamo que permite que los artículos se presten usando un escáner de código de barras (*Media_ID_BC*). Los códigos de barras contienen, además del *Media_ID* individual, un dígito de verificación que el escáner puede usar para determinar si el valor escaneado es correcto. Este campo de código de barras se incluye aquí solo para fines de prueba. Sería mejor si la clave principal de la tabla de artículos (*Media*) se pudiera ingresar directamente en forma de código de barras, o si se usara una macro para eliminar el dígito de verificación del número de código de barras ingresado antes del almacenamiento.

Finalmente, necesitamos conectar *Reader* con *Loan*. En la tabla del lector, solo se incluyen en el plan el nombre, un bloqueo opcional, y una clave externa que vincula a la tabla *Gender*.

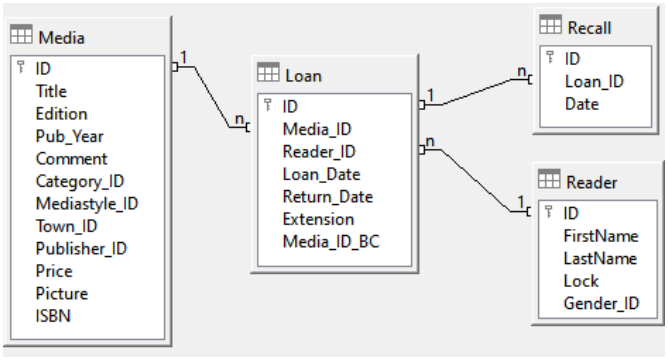


Figura 54: Préstamos (Loan)

Tabla de administración de usuarios (Reader)

Para este diseño de tabla, se prevén dos escenarios. La cadena de tablas que se muestra en la figura 55 está diseñada para bibliotecas escolares. Aquí no hay necesidad de direcciones, ya que los alumnos pueden ser contactados a través de la escuela. Los avisos no necesitan enviarse por correo postal, sino que pueden distribuirse internamente. La cadena de direcciones es necesaria en el caso de las bibliotecas públicas. Aquí debe ingresar los datos que serán necesarios para la creación de cartas de comunicación con el lector.

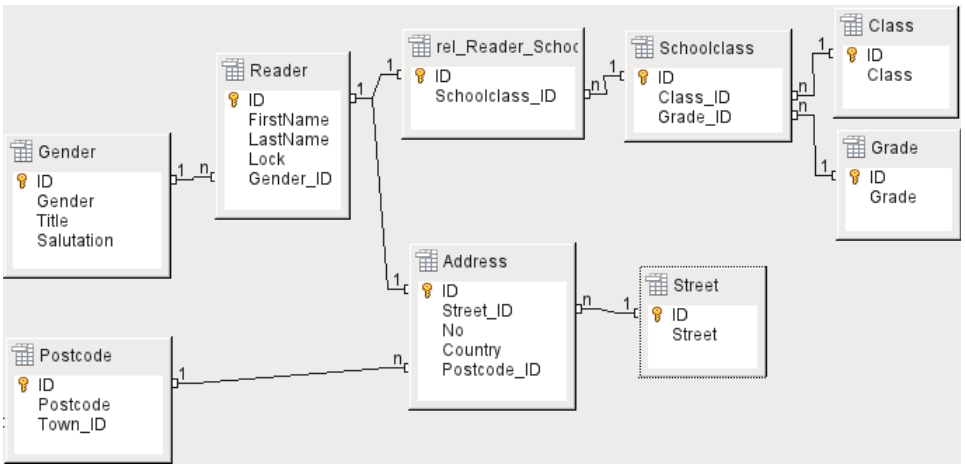


Figura 55: Lectores: una cadena de clase escolar y otra de direcciones

La tabla *Gender* garantiza que se use el saludo correcto en los avisos. La redacción de avisos se puede automatizar en la medida de lo posible. Además, algunos nombres de pila pueden ser igualmente masculinos o femeninos. Por lo tanto, se requiere una lista separada de género incluso cuando los avisos se escriben a mano.

La tabla *rel_Reader_Schoolclass*, como la tabla *Address*, tiene una relación `1:1` con la tabla *Reader*. Esto se eligió porque podría ser necesaria la clase de la escuela o la dirección. De lo contrario, el *Schoolclass_ID* podría colocarse directamente en la tabla del alumno, Lo mismo ocurriría con el contenido completo de la tabla de direcciones en un sistema de biblioteca pública.

Una *School class* generalmente contiene una designación de año y un sufijo de nivel. En una escuela de 4 niveles, este sufijo puede ir de la **a** a la **d**. El sufijo se ingresa en la tabla *Class*. El año está en una tabla *Grade* separada. De esa manera, si los lectores ascienden de clase al final de cada año escolar, simplemente puede cambiar la entrada del año para todos.

La tabla *Address* también está dividida. La calle se almacena por separado porque los nombres de las calles dentro de un área a menudo se repiten. El código postal y la ciudad están separados porque a menudo hay varios códigos postales para una sola área y, por lo tanto, hay más códigos postales que ciudades. Por eso en comparación con la tabla *Address*, la tabla *Postcode* contiene significativamente menos registros y la tabla *Town* aún menos.

La forma en que se utiliza esta estructura de tabla se explica más adelante en el Capítulo 4, «Formularios», de esta guía.

Crear tablas

La mayoría de los usuarios de LibreOffice generalmente usarán la interfaz gráfica de usuario (GUI) exclusivamente para crear tablas. La entrada directa de órdenes SQL se hace necesaria cuando, por ejemplo, un campo debe insertarse posteriormente en una posición particular, o un valor estándar debe establecerse después de que se haya guardado la tabla.

Terminología de la tabla: la siguiente imagen muestra la división estándar de las tablas en columnas y filas.

	Tabla (TABLE)							
	Columna (COLUMN)				Columna (COLUMN)			
	Nombre campo (FIELD)	Tipo campo (TYPE)	Vacío (NULL)	Predeterminado. (DEFAULT)	Nombre campo (FIELD)	Tipo campo (TYPE)	Vacío (NULL)	Predeterminado. (DEFAULT)
Fila (ROW)	Registro							

Los registros de datos se almacenan en una sola fila de la tabla. Las columnas individuales se definen en gran medida por el campo, el tipo y las reglas que determinan si el campo puede estar vacío. Según el tipo, también se puede determinar el tamaño del campo en caracteres. Además, se puede especificar un valor predeterminado para usar cuando no se ingresó nada en el campo.

En la interfaz de Base, los términos para una columna se describen de una manera diferente, como se muestra a continuación.

Notaciones en la interfaz de Base			
Columna (COLUMN)			
		Propiedades de campo	
Nombre campo (FIELD)	Tipo campo (TYPE)	Entrada requerida (NULL / NOT NULL)	Valor predeterminado (DEFAULT)

Field se convierte en Field Name, Type se convierte en Field Type. Field Name y Field Type se ingresan en el área superior de la ventana Table Design. En el área inferior, tiene la oportunidad de establecer, bajo las propiedades de Field, las otras propiedades de columna, en la medida en que se puedan establecer utilizando la interfaz. Las limitaciones incluyen establecer el valor predeterminado de un campo de fecha en la fecha real de entrada. Esto solo es posible mediante el uso de la orden SQL apropiada (consulte “Entrada directa de órdenes SQL” en la página 103).

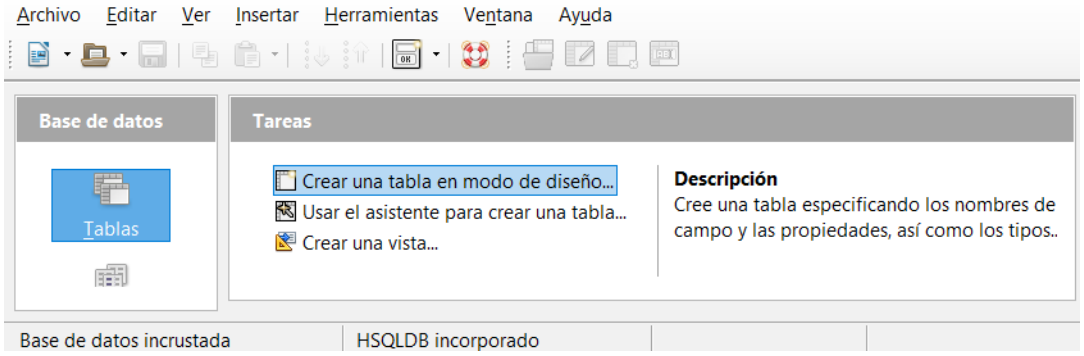
Nota

Valores predeterminados: Este término en la interfaz no significa lo que el usuario de la base de datos generalmente entiende como un valor predeterminado.La interfaz muestra un cierto valor visualmente que se guarda con los datos.

El valor predeterminado en una base de datos se almacena en la definición de la tabla. Luego se escribe en un campo de un nuevo registro de datos siempre que esté vacío. Los valores predeterminados de SQL no aparecen al editar las propiedades de la tabla.

Crear una base de datos utilizando la interfaz gráfica de usuario.

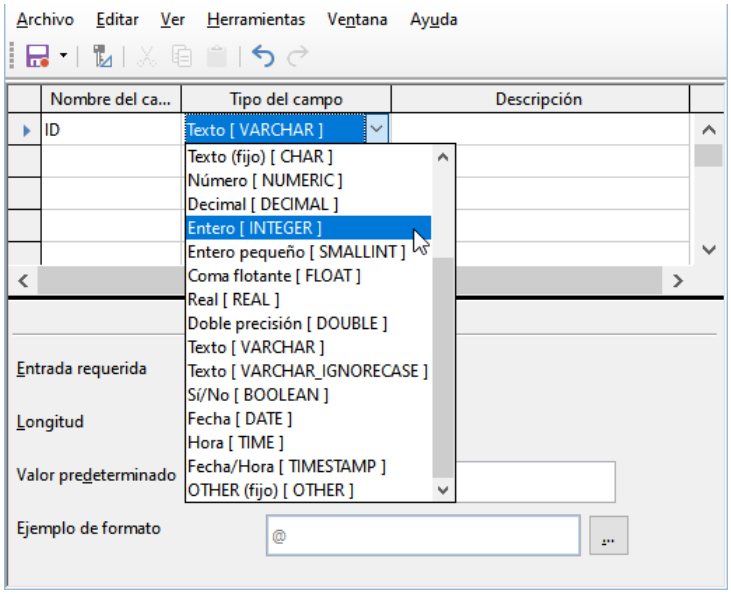
La creación de la base de datos utilizando la interfaz de usuario gráfica (GUI) se explica paso a paso, utilizando la tabla *Media* como ejemplo.



Inicie el editor de tablas haciendo clic en *Crear una tabla en modo de diseño*.

7) Campo ID:

   a) En la primera columna, ingrese la ID del nombre del campo. Luego presione la tecla Tab para moverse a la columna Tipo del campo. A continuación haga clic con el ratón en la siguiente columna para seleccionarla o presione la tecla Intro.



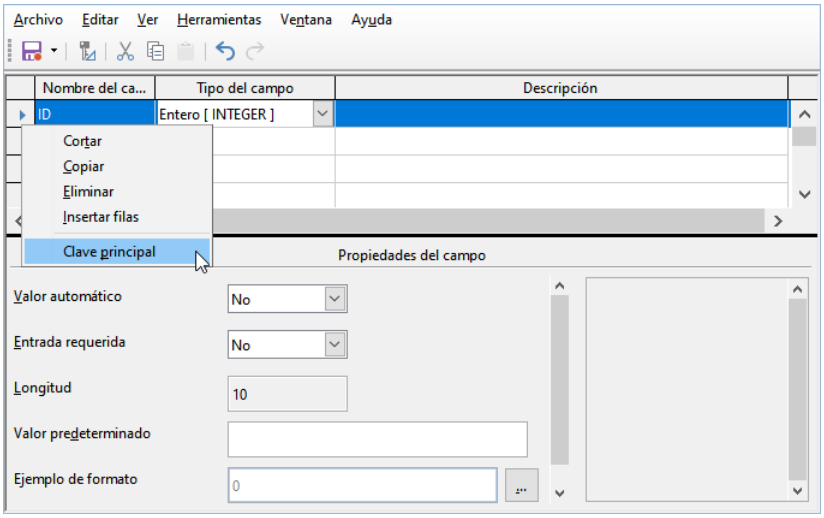
   b) El valor predeterminado es para *Típo del campo* es `Texto [VARCHAR]`, Cámbielo a `Entero [INTEGER]` mediante la lista desplegable.

      Los enteros pueden almacenar un número de hasta 10 dígitos. Además, Integer es el único tipo disponible en la interfaz que puede recibir un valor de incremento automático.

Consejo

Para realizar una selección rápida de la lista Tipo del campo con el teclado, presione la tecla correspondiente a la primera letra que elija. Presionar repetidamente esta tecla puede usarse para cambiar la selección. Por ejemplo, presionar D puede cambiar su selección de Fecha a Fecha / Hora o a Decimal.

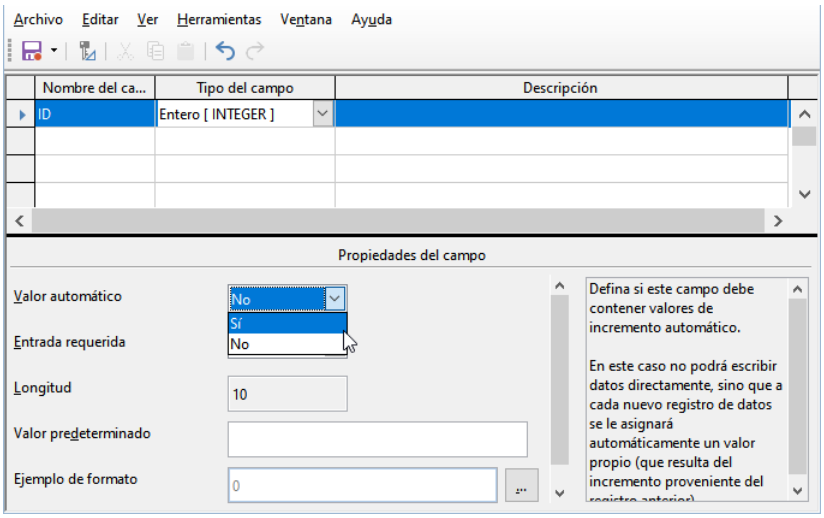
   c) Establezca el campo *ID* como la clave principal haciendo clic con el botón derecho del ratón en el triángulo verde a la izquierda de su fila y seleccionando *Clave principal* en el menú contextual. Aparecerá el símbolo de una llave antes de la identificación.



Nota

La clave principal tiene un solo propósito: identificar de forma exclusiva el registro. Por lo tanto, puede usar un nombre arbitrario para este campo. En el ejemplo, hemos usado el nombre más usado: ID (abreviatura de identificación).

d) En *Propiedades del campo* para el campo *ID*, cambie el *Valor automático* a *Sí*. Esto hace que la clave primaria se incremente automáticamente. En la base de datos interna, la cuenta comienza en 0.

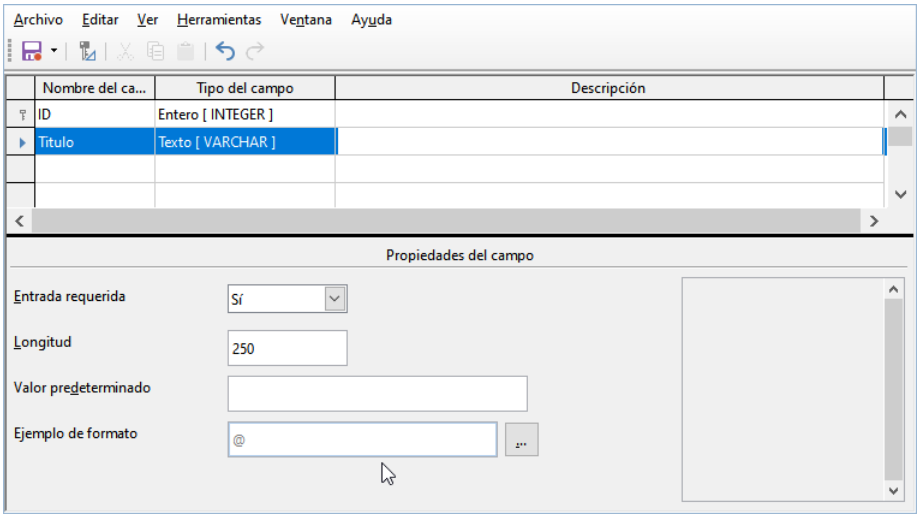


El valor automático solo se puede configurar para un campo en una tabla. Al elegir Sí, este campo se configura automáticamente como la clave primaria si aún no se había configurado una.

8) El campo siguiente es *Title*.

a) El nombre del campo *Title* se ingresa en la columna Nombre del campo.

b) El tipo de campo no necesita cambiarse, ya está establecico como `Texto [VARCHAR]`.



c) En Propiedades del campo, cambie *Entrada requerida* de No a Sí. Un artículo sin título no tendría sentido.

d) Ajuste también la longitud del campo. La longitud predeterminada es de 100 en las versiones recientes de LibreOffice, pero debe aumentarse a 250 para los títulos de artículos.

e)

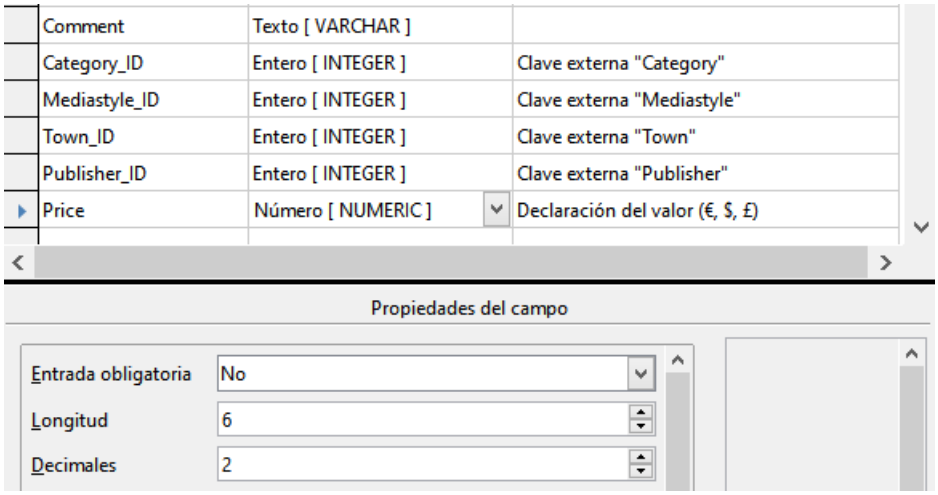
Nombre del campo	Tipo del campo	Descripción
ID	Entero [INTEGER]	
Title	Texto [VARCHAR]	
Edition	Texto [VARCHAR]	No. de edición. nueva edición, etc.
Pub_Year	Entero pequeño [SMALLINT]	Año de publicación

9) La columna *Descripción* para todos los campos podrá ser cualquier cosa. Esta columna también se puede dejar vacía. Solo sirve para explicar el contenido del campo para quienes deseen ver la definición de la tabla.

10) Para el campo *Pub_Year*, se ha elegido el tipo `Entero pequeño [SMALLINT]`. Puede contener un número entero con un tamaño máximo de 5 dígitos. La fecha de publicación no se utiliza en los cálculos, pero definirla como entero asegura que no contendrá caracteres alfabéticos.

Nombre del campo	Tipo del campo	Descripción
ID	Entero [INTEGER]	
Title	Texto [VARCHAR]	
Edition	Texto [VARCHAR]	No. de edición. nueva edición, etc.
Pub_Year	Entero pequeño [SMALLINT]	Año de publicación
Comment	Texto [VARCHAR]	
Category_ID	Entero [INTEGER]	Clave externa "Category"

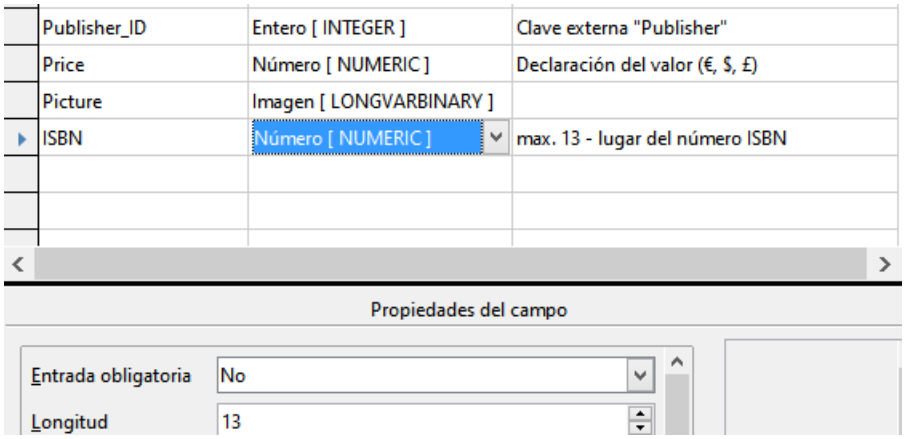
11) Para el campo *Category_ID*, se elije el tipo `Entero[INTEGER]`.Este campo almacenará el mismo valor que la clave primaria de la tabla *Category*, (*ID*) las claves primarias tienen que ser de este tipo, por lo que lo que se ingrese aquí como clave externa tiene que ser del mismo tipo. Esto también se aplica a los siguientes campos de claves externas *Mediastyle_ID*, *Town_ID* y *Publisher_ID*.



12) Para el campo *Price*, use el tipo `Número [NUMERIC]` o `[DECIMAL]`. Ambos tipos de campo pueden contener valores con un punto decimal. En Propiedades del campo, establezca una longitud de 6 caracteres. Esto debería ser suficiente para los precios de nuestros artículos.

   a) El número de lugares decimales se establece en 2. Esto proporciona un precio máximo de 9999.99 ya que el punto decimal en sí no está incluido en el dato del campo.

   b) No es necesario poner el carácter $ en el formato, ya que una fórmula lo manejará.



13) Para *ISBN*, use el tipo `Número[NUMERIC]`. Se puede establecer con exactitud la longitud de campo correcta para este campo. Los ISBN tienen 10 o 13 caracteres de longitud. Se almacenarán como números sin un separador. La longitud se establece en un máximo de 13 caracteres. El número de decimales se establece en cero.

14) Guarde la tabla con el nombre *Media*.

La tabla principal para la base de datos de ejemplo ya está pues creada.

Todas las otras tablas se pueden crear de manera similar. Tenga cuidado de que los tipos de campo y las propiedades de campo coincidan con lo que se va a almacenar en esos campos. Esta es la diferencia con una hoja de cálculo, en la que una columna puede contener una mezcla de propiedades.

Nota

El orden de los campos en la tabla solo se podrá cambiar cuando la tabla se guarde por primera vez en la interfaz. Cuando los datos se ingresan posteriormente directamente en la tabla, el orden de los campos es fijo. Sin embargo, el orden todavía se puede cambiar libremente en consultas, formularios e informes.

Claves primarias

Al guardar una tabla creada en modo diseño, si no se establece una clave principal, aparecerá un cuadro de diálogo preguntándole si se debe crear una clave primaria. Esto indica que falta un campo significativo en la tabla. Sin una clave primaria, la base de datos HSQLDB no puede acceder a la tabla. Este campo generalmente se denomina *ID* y se le da el tipo `INTEGER` con *Valor automático* para incrementar automáticamente su valor cuando se añade un registro. Al hacer clic en *Sí* en el cuadro de diálogo, se crea automáticamente un campo de clave principal. Al hacer clic en *No* o *Cancelar* en el cuadro de diálogo **no** se creará un nuevo campo de clave primaria, pero, puede designar un campo existente, haciendo clic con el botón derecho en la flecha verde a la izquierda del campo correspondiente y entonces la tabla estará operativa.

También puede usar una combinación de campos como su clave principal. Los campos deben declararse como clave principal juntos (mantenga presionada la tecla *Control* o *Shift*). Luego, al hacer clic con el botón derecho, la combinación de los campos resaltados será la clave principal.

Si la información se importa a esta tabla desde otras (por ejemplo, una base de datos de direcciones con códigos postales y ciudades almacenadas externamente), se debe incluir un campo con el mismo tipo de datos que la clave primaria de la otra tabla. Supongamos que la tabla *Postcode* tiene como clave primaria un campo llamado *ID* con el tipo Entero minúsculo. La tabla de direcciones debe tener un campo *Postcode_ID* con el mismo tipo. Lo que se ingresa en la tabla de direcciones es siempre el número que sirve como clave primaria para la ubicación dada en la tabla *Postcode*. Esto significa que la tabla *Address* ahora tiene una clave externa además de su propia clave primaria.

Una regla fundamental para nombrar campos en una tabla es que no hay dos campos que puedan tener el mismo nombre. Por lo tanto, no puede tener un segundo campo llamado *ID* como una clave externa en la tabla *Address*.

El tipo de campo solo puede modificarse de forma limitada. Siempre se permite aumentar una propiedad (por ejemplo la longitud de un campo), ya que todos los valores ya ingresados coincidirán con las nuevas condiciones. Al disminuir una propiedad pueden surgir problemas y puede conducir a una pérdida de datos.

Los campos de hora en el diseño de tablas no se pueden crear para contener fracciones de segundo. Para eso, necesita hacerlo en dos pasos: Primero crear un campo de Marca de Tiempo. en el diseño y posteriormente modificar este campo usando **Herramientas > SQL**.

```
ALTER TABLE "Table_name" ALTER COLUMN "Field_name" TIMESTAMP (6)
```

El parámetro "6" hace que el campo Timestamp sea capaz de almacenar fracciones de segundo.

Formatear campos

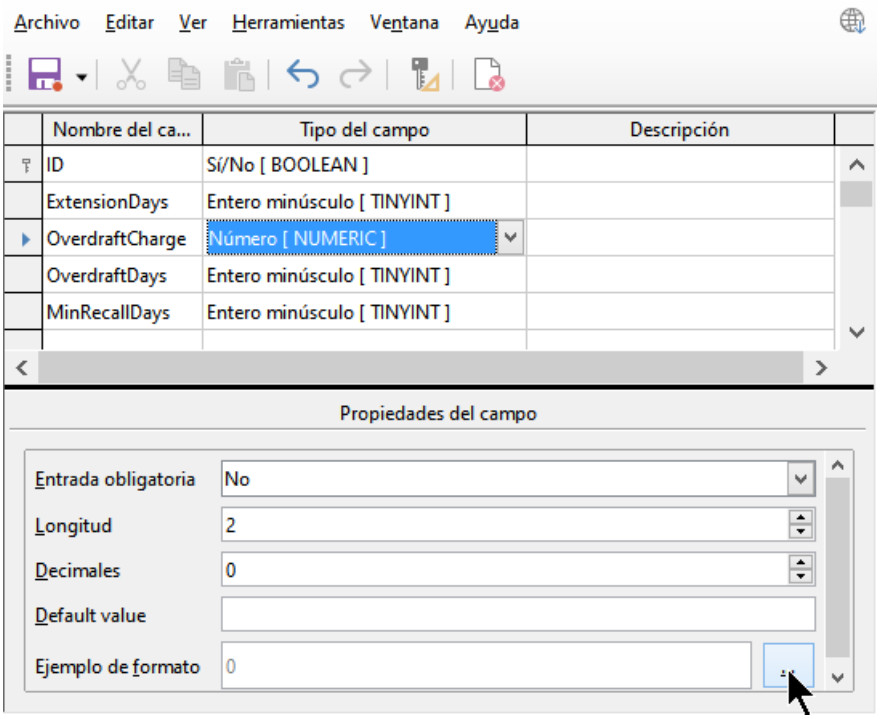
El formateo presenta los valores de la base de datos al usuario y permite la entrada de valores dependiendo de las convenciones de entrada normales en ese país. Sin un formato establecido, las cifras decimales se representarán con un punto, cuando la mayoría de los países europeos usan una coma (4.21 en lugar de 4,21). Los valores de fecha se presentan en la forma 2014-12-22. Al configurar el formato, debe tener en cuenta las normas locales del país.

El formateo solo proporciona una representación de los contenidos. Una fecha representada por un número de año de dos caracteres se almacenará como un año de cuatro caracteres. Si se crea un campo para un número con dos decimales, como el cargo vencido (*OverdarftCharge* en el siguiente ejemplo), el número se almacena con dos decimales, incluso si no se ha configurado el formato para mostrarlos. Se puede ingresar un número con dos decimales en un campo formateado sin decimales. La parte decimal parece desaparecer en la entrada pero se vuelve visible si se omite el formato.

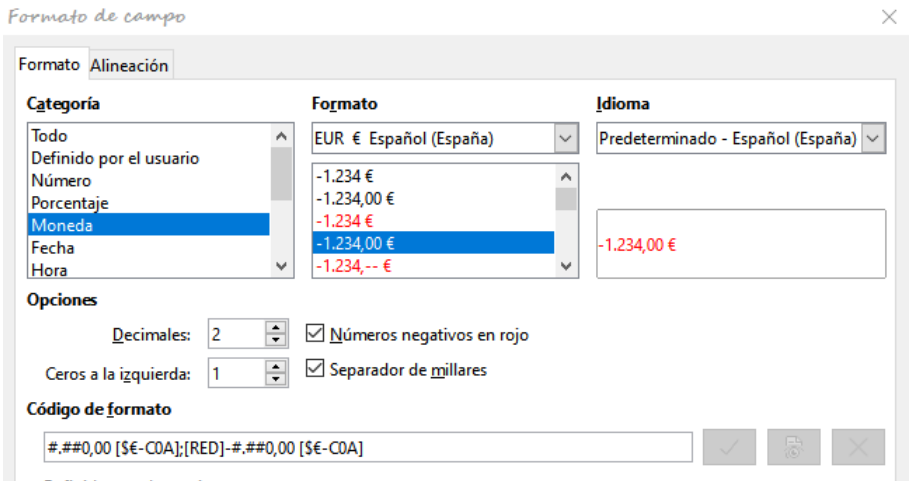
Los formularios se pueden configurar para mostrar solo la información necesaria, para mostrar solo la hora, de un campo `Fecha/Hora[TIMESTAMP]`. En el caso de almacenar tiempo desde un cronómetro, por ejemplo, los minutos, segundos y fracciones de segundo en un campo de marca

de tiempo se pueden mostrar usando `MM:SS.00` en el formato de visualización. Un formato sin fecha se puede establecer más adelante en formularios utilizando el campo formateado, pero no directamente en el campo.

El formato de los campos cuando se crea la tabla o posteriormente, a través de las propiedades del campo, utiliza un diálogo separado:



En *Propiedades del campo*, el botón junto a *Ejemplo de formato* abre el cuadro de diálogo para cambiar el formato.



Al crear campos de moneda, tenga cuidado de que el campo numérico tenga dos lugares decimales establecidos. El formateo se puede realizar al crear la tabla en la interfaz para usar la moneda adecuada durante la entrada. Esto solo afecta la entrada en la tabla y en las consultas que usan el valor de entrada sin cálculos. En los formularios, la designación de moneda debe formatearse en las propiedades del control.

Nota

Base guarda el formato de las tablas cuando se crean los campos o durante la entrada de datos si los formatos de columna se modifican haciendo clic con el botón derecho en los encabezados de las columnas. Los anchos de columna en la pantalla de entrada también se guardan cuando se modifican durante la entrada de datos.

En consultas, formularios o informes, el formato de visualización se puede modificar según sea necesario.

En el caso de los campos que deben contener un porcentaje, tenga en cuenta que el resultado de 1% debe almacenarse como 0.01. Por lo tanto, escribir porcentajes requiere al menos dos lugares decimales. Si es necesario almacenar porcentajes fraccionarios como 3,45, el valor numérico almacenado requiere cuatro decimales (0.0345).

Crear un índice

A veces es útil indexar otros campos o una combinación de otros campos además de la clave primaria. Un índice acelera la búsqueda y también se puede utilizar para evitar entradas duplicadas.

Cada índice tiene un orden de clasificación definido. Si se muestra una tabla sin ordenar, el orden de clasificación estará de acuerdo con el contenido de los campos especificados en el índice.

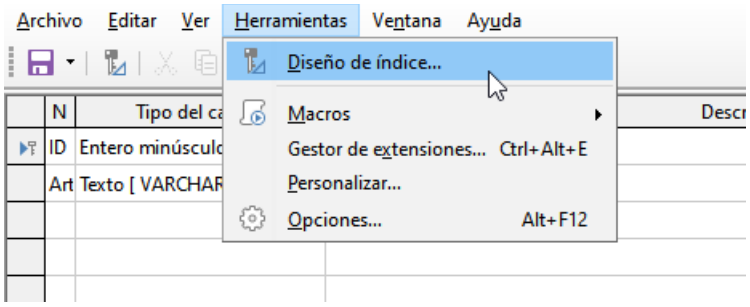


Figure 56: Acceso al Diseño de índice

Abra la tabla para editar haciendo clic derecho y usando el menú contextual. Luego puede acceder a la creación de índice con **Herramientas > Diseño de índice**.

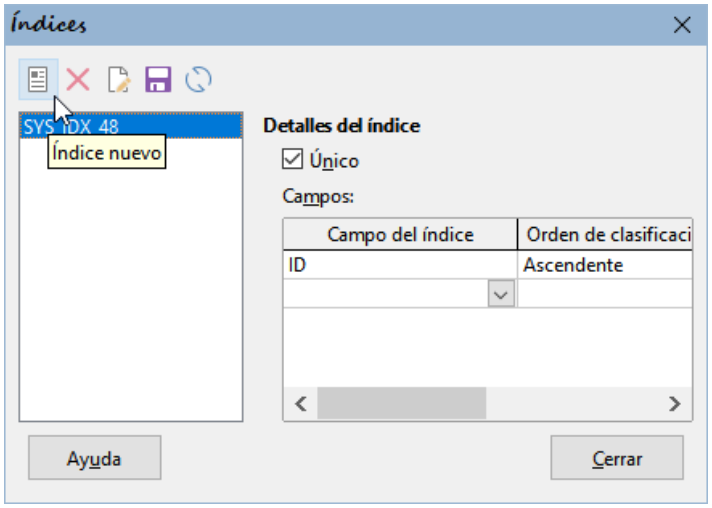


Figura 57: Crear un nuevo índice

En el cuadro de diálogo Índices (Figura 57) haga clic en el botón *Índice nuevo* para crear un índice, además de la clave principal.

El nuevo índice recibe automáticamente el nombre *index1*. El Índice especifica qué campo o campos se utilizarán para este índice. Al mismo tiempo, puede elegir el orden de clasificación.

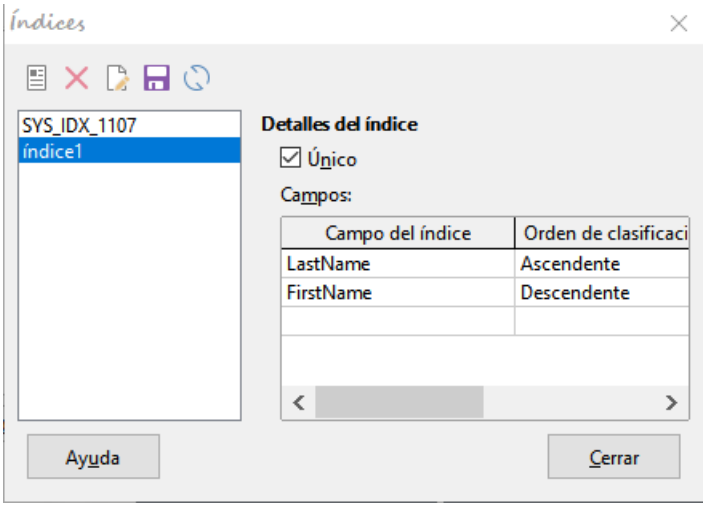


Figura 58: El índice se define como único

En principio, también se puede crear un índice a partir de campos de tabla que no contienen valores únicos. Sin embargo, en la Figura 58, en Detalles de índice se ha marcado Único, de modo que el campo *LastName* junto con el campo *FirstName* solo puede tener entradas que aún no se encuentren en esa combinación. Por ejemplo, Robert Miller y Robert Maier son posibles, y también Robert Miller y Eva Miller. Pero nunca dos Robert Miller.

Si se crea un índice solo para un campo, la unicidad se aplica a ese campo. Tal índice suele ser la clave principal. En este campo, cada valor puede aparecer solo una vez. Además, en el caso de claves primarias, el campo nunca puede ser `NULL`.

Una circunstancia excepcional para un índice único es cuando no hay entrada en un campo (el campo es `NULL`). Como `NULL` puede tener cualquier valor arbitrario, un índice que usa dos campos siempre puede tener la misma entrada repetidamente en uno de los campos siempre que no haya entrada en el otro.

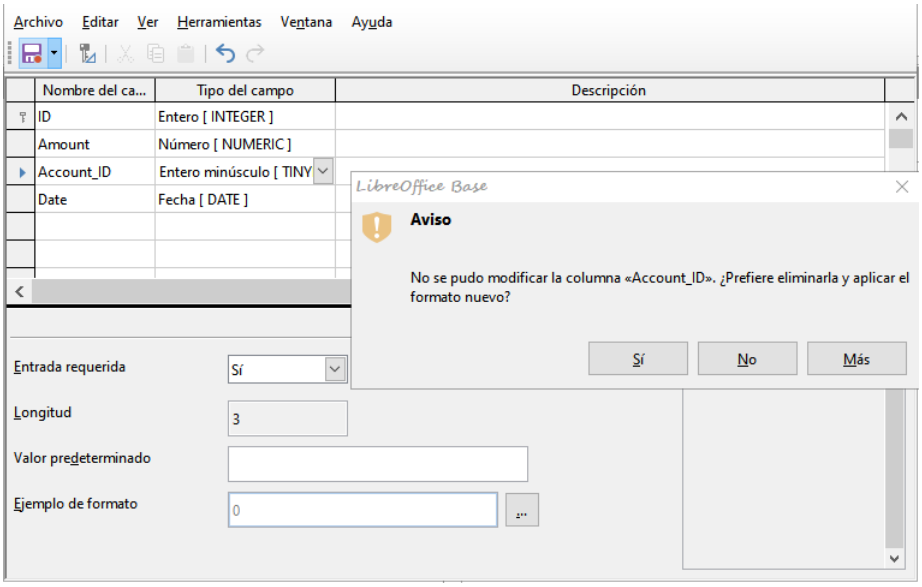
Nota

NULL se usa en bases de datos para designar una celda vacía, que no contiene nada. No es podible ningún cálculo usando un campo NULL. Esto contrasta con las hojas de cálculo, en las que a las celdas vacías se les asigna automáticamente el valor 0 (cero).

Ejemplo: en una base de datos, el número de artículos y la fecha del préstamo se ingresan cuando se presta el artículo. Cuando se devuelve, se ingresa una fecha de devolución. En teoría, un índice que utiliza los campos *Media_ID* y *ReturnDate* podría evitar fácilmente que se preste el mismo elemento repetidamente sin que se indique la fecha de devolución. Lamentablemente, esto no funcionará porque la fecha de devolución inicialmente no tiene ningún valor. El índice evitará que un artículo se marque como devuelto dos veces con la misma fecha, pero no hará nada más.

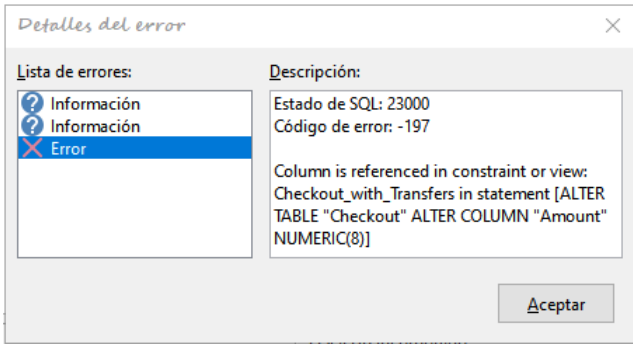
Problemas al modificar tablas

Es mejor crear tablas completas con todas las configuraciones necesarias, antes de establecer relaciones o crear consultas, de modo que los cambios en la configuración de la tabla no sean necesarios más adelante. Cuando las propiedades de los campos (nombre de campo, entrada obligatoria, etc.) se cambian más tarde, puede generar mensajes de error que no se deben a la interfaz, sino a un intento de modificar la base de datos subyacente de una manera no deseada.

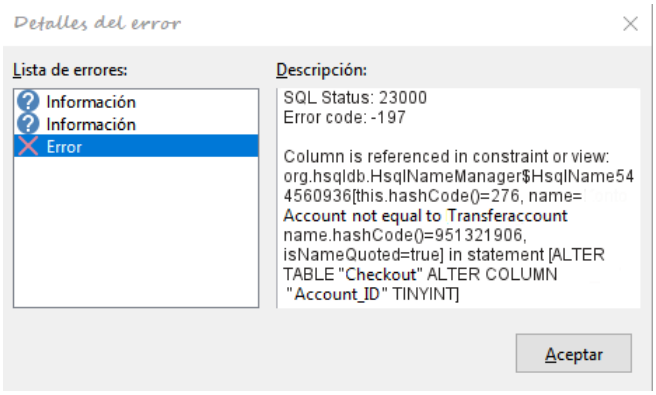


En este caso, el campo *Amount* se intenta cambiár a una *Entrada requerida* = Sí. El símbolo de advertencia nos notifica que este cambio puede conducir a la pérdida de datos. No es posible un cambio simple porque puede haber registros que ya tengan una entrada en este campo.

Al hacer clic en *Sí* aparece otro aviso de error, ya que la estructura de la base de datos no permite que se elimine este campo. Al hacer clic en *No*, se cancela toda la operación. El botón *Más* se proporciona siempre que sea posible para brindarle información adicional sobre la resolución del problema.



La columna con el nombre de campo *Amount* se menciona en otra parte de la base de datos. Esto podría ser una definición restrictiva o una vista de tabla que fue creada por algún usuario después de que se creó la tabla misma. La ilustración anterior muestra que el nombre de la restricción o vista es *View_Checkout_with_Transfers*. Esto deja claro dónde deben realizarse cambios en la base de datos. Por ejemplo, el código SQL para la vista podría guardarse primero como una consulta, y luego la vista podría destruirse y hacer un nuevo intento para modificar el campo.



En este caso, el nombre de la restricción *Account not equal to Transferaccount* nos lleva hasta la definición de esa restricción. La condición es que el valor en el campo *Account_ID* no puede ser el mismo que el valor en el campo *TransferAccount_ID*. La columna solo puede modificarse si se elimina esta condición.

Ahora, si se produce un error adicional, lo más probable es que sea causado por el campo correspondiente que está vinculado a un campo en otra tabla por una relación definida. En este caso, el enlace debe romperse utilizando *Herramientas > Relaciones* antes de que se pueda realizar el cambio.

Limitaciones del diseño de tablas gráficas.

La secuencia de campos en una tabla no se puede cambiar una vez que se ha guardado la base de datos. Para mostrar una secuencia diferente se necesita una consulta.

Solo mediante órdenes SQL directas puede insertar un campo en una posición específica en la tabla. Sin embargo, los campos ya creados no se pueden mover con este método.

Las propiedades de las tablas deben establecerse al principio: por ejemplo, qué campos no deben ser estar vacíos (`NULL`) y cuáles deben contener un valor predeterminado. Estas propiedades no se pueden cambiar posteriormente con la interfaz.

Los valores predeterminados que puede establecer en la interfaz no son tan potentes como los posibles valores predeterminados dentro de la base de datos. Por ejemplo, no puede definir el valor predeterminado para un campo de fecha como la fecha de entrada. Eso solo es posible con órdenes SQL directas.

Entrada directa de órdenes SQL

Para ingresar órdenes SQL directas, vaya a **Herramientas > SQL**.

Las órdenes se ingresan en el área superior de la ventana (Figura 59). En el área de *Estado*, se muestra el éxito o la razón del fracaso. Los resultados de las instrucciones `SELECT` se pueden mostrar en el área *Salida* si la casilla de verificación está seleccionada.

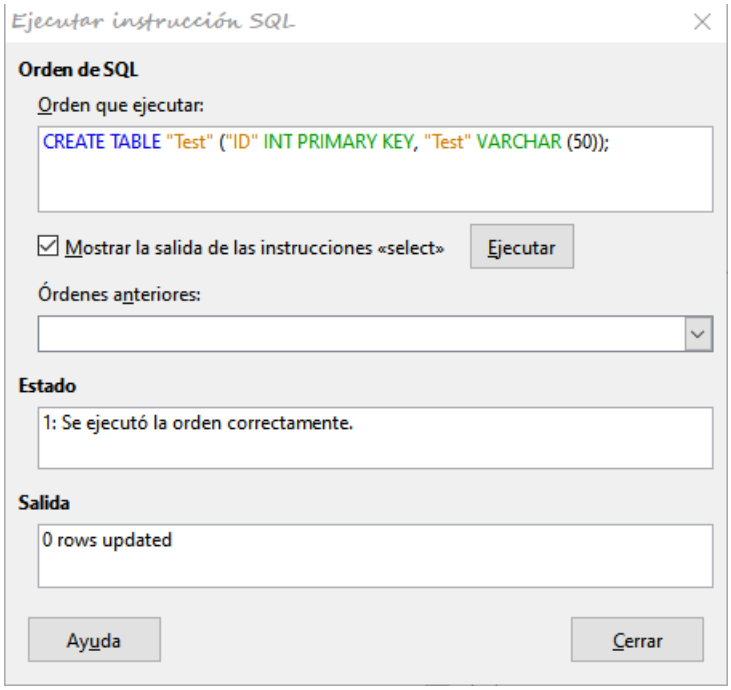


Figura 59: Entrada directa de órdenes SQL

Puede encontrar un resumen de lás posibles órdenes para el motor HSQLDB incorporado en http://www.hsqldb.org/doc/1.8/guide/ch09.html. Los contenidos se describen en las siguientes

secciones. Algunas órdenes solo tienen sentido cuando se trata de una base de datos HSQLDB externa (Especificar usuario, etc.). Cuando las necesite, puede consultar la sección «Conectar una base de datos a un HSQLDB externo» en el *Apéndice A* de esta guía.

Nota

LibreOffice se basa en la versión 1.8.0 de HSQLDB. La versión del servidor actualmente disponible es 2.5. Las funciones de la nueva versión son más amplias. Se pueden encontrar en `http://hsqldb.org/web/hsqlDocsFrame.html`. La Versión 1.8 se describe en `http://www.hsqldb.org/doc/1.8/guide/`. Una descripción adicional en los paquetes de instalación para HSQLDB, se pueden descargar desde `http://sourceforge.net/projects/hsqldb/files/hsqldb/`.

Crear tabla

Una orden simple para crear una tabla utilizable es:

```
CREATE TABLE "Test" ("ID" INT PRIMARY KEY, "Texto" VARCHAR(50));
```

Desglose de esta órden:

`CREATE TABLE "Test"`: Crear una tabla de nombre "Test".

Los nombres de campo, tipos de campo y opciones especificados se insertan entre paréntesis `( )`: `("ID" INT PRIMARY KEY, "Text" VARCHAR(50))`: campo de nombre ID de tipo numérico entero como clave principal; campo de nombre Texto de tipo texto variable y la longitud limitada a 50 caracteres.

Parámetros de la orden CREATE:

```
CREATE [MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT] TABLE
"Nombre_Tabla" ( <Definiciones_Campos> [, ...] [, <Restricciones_Campos>...] ) [ON
COMMIT {DELETE | PRESERVE} ROWS];
```

[MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT]:
Especifica la ubicación de la tabla recién creada. La configuración predeterminada es `MEMORY`. HSQLDB crea todas las tablas en la memoria central. Esta configuración también se aplica a las tablas que LibreOffice Base escribe en la base de datos incrustada. Otra posibilidad sería escribir las tablas en el disco duro y usar la memoria solo para almacenar el acceso al disco duro (`CACHED`).

Nota

`CREATE TEXT TABLE "Test" ("ID" INT PRIMARY KEY, "Text" VARCHAR(50));`

Crear una tabla test en HSQLDB. Tiene que estar vinculada a un archivo de texto externo (por ejemplo, un archivo *.csv): `SET TABLE "Text" SOURCE "Text.csv";`

Naturalmente, el archivo Text.csv debe tener los campos correspondientes en el orden correcto. Al crear el enlace, se pueden seleccionar varias opciones adicionales. Para más detalles, consulte http://www.hsqldb.org/doc/1.8/guide/guide.html#set_table_source-section

Las tablas de texto no están protegidas de escritura, contra otros programas. Por lo tanto, puede suceder que otro programa o usuario altere la tabla justo cuando Base accede a ella. Las tablas de texto se utilizan principalmente para el intercambio de datos entre diferentes programas.

Las tablas en formato texto (TEXT) (como CSV) no se pueden escribir en bases de datos internas que se configuran únicamente en MEMORIA, mientras que Base no puede acceder a las tablas TEMPORALES o TEMP. Las órdenes SQL se ejecutan en este caso, pero las tablas no se muestran (y, por lo tanto, no se pueden eliminar) utilizando la interfaz, y los datos ingresados a través de SQL tampoco son visibles para el módulo de consulta de la interfaz, a menos que se elimine automáticamente el contenido después de que se evita la confirmación final (con ON COMMIT PRESERVE ROWS). Cualquier solicitud en este caso mostrará una tabla sin ningún contenido.

Las tablas creadas directamente con SQL no se muestran inmediatamente. Debe usar **Ver > Actualizar tablas** o simplemente cerrar la base de datos y luego volver a abrirla.

<Definiciones_Campos>:

> "Nombre_Campo" Tipo de Datos[(Número de caracteres[,Lugares decimales])] [{DEFAULT "Valor_Predeterminado" | GENERATED BY DEFAULT AS IDENTITY (START WITH <n>[, INCREMENT BY <m>])}] | [[NOT] NULL] [IDENTITY] [PRIMARY KEY]

Permite que los valores predeterminados se incluyan en la definición del campo.

Para los campos de texto, puede ingresar texto entre comillas simples o NULL. La única función SQL permitida es CURRENT_USER. Esto solo tiene sentido si HSQLDB se está utilizando como una base de datos de servidor externa con varios usuarios.

Para los campos de fecha y hora, se puede ingresar una fecha, una hora o una combinación de las dos entre comillas simples o NULL. Debe asegurarse de que la fecha sigue las convenciones americanas (aaaa-mm-dd), que la hora tiene el formato hh:mm:ss, y que un valor combinado de fecha/hora[TIMESTAMP] tiene el formato aaaa-mm-dd hh:mm:ss .

Funciones SQL permitidas:

para la fecha actual CURRENT_DATE, TODAY, CURDATE ()

para la hora actual CURRENT_TIME, NOW, CURTIME ()

para la marca de tiempo de datos actual CURRENT_TIMESTAMP, NOW.

Para los campos booleanos (sí / no) se pueden ingresar las expresiones FALSE, TRUE, NULL. Tienen que ingresarse sin comillas simples.

Para los campos numéricos, es posible cualquier número válido en el rango o NULL. Aquí también, si ingresa NULL, tampoco use comillas. Al ingresar decimales, asegúrese de que el punto decimal sea un punto y no una coma. (Algunas personas de habla inglesa usan una coma como separador decimal).

Para campos binarios (imágenes, etc.) es posible cualquier cadena hexadecimal válida entre comillas simples o NULL. Una cadena de ejemplo hexadecimal es: 0004ff, que representa 3 bytes: primero 0, luego 4 y finalmente 255 (0xff). Como los campos binarios en la práctica solo necesitan ingresarse para las imágenes, debe conocer el código binario de la imagen que se utilizará como predeterminado.

Nota

> Sistema hexadecimal: los números tienen 16 como base de numeración. Un sistema mixto que consiste en los números del 0 al 9 y las letras de la **a** a la **f** proporciona 16 dígitos posibles para cada columna. Con dos columnas, puede tener 16*16=256 valores posibles. Esto corresponde a 1 Byte (2^8).

NOT NULL: El valor del campo no puede ser NULL. Esta condición solo puede darse en la definición de campo.

Ejemplo:

```
CREATE TABLE "Test" ("ID" INT GENERATED BY DEFAULT AS IDENTITY (START WITH 10),
"Name" VARCHAR(50) NOT NULL, "Date" DATE DEFAULT TODAY);
```

Se crea una tabla llamada *Test*. El campo *ID*, clave primaria se define como *Valor automático*, cuyo valor comienza en 10. El campo *Name* es un campo de texto con longitud máxima de 50 caracteres. No debe estar vacío. Finalmente tenemos el campo de fecha *Date* que almacena la fecha actual, si no se ingresa ninguna otra fecha. Este valor predeterminado solo se hace efectivo cuando se crea un nuevo registro. Eliminar una fecha en un registro existente deja el campo vacío.

<Restricciones_Campos>:

```
[CONSTRAINT "Nombre_Restricción"]
UNIQUE ( "Nombre_Campo1" [,"Nombre_Campo2"...] ) |
PRIMARY KEY ( "Nombre_Campo1" [,"Nombre_Campo2"...] ) |
FOREIGN KEY ( "Nombre_Campo1" [,"NombreCampo2"...] )
REFERENCES "otro_Nombre_Tabla" ( "Nombre_Campo1" [,"Nombre_Campo2"...])
[ON {DELETE | UPDATE}
{CASCADE | SET DEFAULT | SET NULL}] |
CHECK(<Condición_Búsqueda>)
```

CONSTRAINT define las condiciones que deben cumplirse cuando se ingresan los datos. Las restricciones pueden recibir un nombre.UNIQUE ("Nombre_Campo"): el valor del campo debe ser único dentro de ese campo

PRIMARY KEY ("Nombre_Campo"): el valor del campo debe ser único y no puede ser NULL (clave primaria)

FOREIGN KEY ("Nombre_Campo") REFERENCES <"otro_Nombre_Tabla"> "Nombre_Campo"): Los campos especificados en esta tabla están vinculados a los campos de otra tabla. El valor del campo debe ser probado para integridad referencial como claves foráneas; es decir, debe haber una clave primaria correspondiente en la otra tabla, si se ingresa un valor aquí.

[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}]: En el caso de una clave externa, esto especifica qué sucederá si, por ejemplo, se elimina el registro externo. No tiene sentido, en una tabla de préstamos para una biblioteca, tener un número de usuario para el cual el usuario ya no existe. El registro correspondiente debe modificarse para que la relación entre las tablas siga siendo válida. Por lo general, el registro simplemente se elimina. Esto sucede si selecciona ON DELETE CASCADE.

CHECK(<Condición_Búsqueda>): Formulado como una condición WHERE, pero solo para el registro actual.

```
CREATE TABLE "Time_measurement" ("ID" INT PRIMARY KEY, "Start_time" TIME,
"End_time" TIME, CHECK ("Start_time" <= "End_time"));
```

La condición CHECK excluye la entrada de un valor de *End_time* a la hora *Start_time*. Un intento de hacer esto produce un mensaje de error similar a:

```
Check constraint violation SYS_CT_357 table: Time_measurement ...
```

A la restricción de búsqueda se le asigna un nombre que no es muy informativo. En lugar de eso, el nombre podría definirse en la definición de la tabla:

```
CREATE TABLE "Time_measurement" ("ID" INT PRIMARY KEY, "Start_time" TIME,
"End_time" TIME, CONSTRAINT "Start_time<=End_time" CHECK ("Start_time" <=
"End_time"));
```

Esto da un mensaje de error algo más claro en el que aparece el nombre de la restricción involucrada.

Se deben respetar las restricciones al establecer relaciones entre tablas o la indexación para campos particulares. Las restricciones se establecen usando la condición «CHECK», en la interfaz

usando **Herramientas > Relaciones**, y también en con los índices creados al editar la Tabla ya diseñada con **Herramientas > Diseño de índices**.

[ON COMMIT {DELETE | PRESERVE} ROWS]:

> El contenido de las tablas del tipo TEMPORARY o TEMP se borra de manera predeterminada cuando ha terminado de trabajar con un registro en particular (ON COMMIT DELETE ROWS). Esto le permite crear registros temporales, que contienen información para otras acciones que se llevarán a cabo al mismo tiempo.

Si desea que una tabla de este tipo contenga datos disponibles para una sesión completa (desde abrir una base de datos hasta cerrarla), elija ON COMMIT PRESERVE ROWS.

Modificar la tabla

A veces es posible que desee insertar un campo adicional en una posición particular de la tabla. Supongamos que tiene una tabla llamada *Direcciones* con campos *ID*, *Apellido*, *Calle*, etc. Se observa que se nos ha olvidado incluir el campo *Nombre*.

```
ALTER TABLE "Direcciones" ADD "Nombre" VARCHAR(25) BEFORE "Apellido";
```

ALTER TABLE "Direcciones": Cambiar la tabla *Direcciones*.

ADD "Nombre" VARCHAR(25): Insertar el campo *Nombre* con una longitud de 25 caracteres.

BEFORE “Apellido”: Antes del campo *Apellido*.

La posibilidad de especificar la posición de los campos adicionales después de la creación de la tabla no está disponible en la interfaz.

```
ALTER TABLE "Nombre_Tabla" ADD [COLUMN] <Definición_Campo> [BEFORE
"Nombre_Campo_ya_existente"];
```

La designación adicional COLUMN no es necesaria en casos donde no hay opciones alternativas disponibles.

```
ALTER TABLE "Nombre_Tabla" DROP [COLUMN] "Nombre_Campo";
```

El campo *Nombre_Campo* se borra de la tabla *Nombre_Tabla*. Sin embargo, esto no ocurre si el campo está involucrado en una vista o como una clave externa en otra tabla.

```
ALTER TABLE "Nombre_Tabla" ALTER COLUMN "Nombre_Campo" RENAME TO
"Nuevo_Nombre_Campo";
```

Cambia el nombre de un campo.

```
ALTER TABLE "Nombre_Tabla" ALTER COLUMN "Nombre_Campo" SET DEFAULT
<Valor_Predeterminado>};
```

Establece un valor predeterminado específico para el campo. NULL elimina un valor predeterminado existente.

```
ALTER TABLE "Nombre_Tabla" ALTER COLUMN "Nombre_Campo" SET [NOT] NULL;
```

Establece o elimina una condición NOT NULL para un campo.

```
ALTER TABLE "Nombre_Tabla" ALTER COLUMN <Definición_Campo>;
```

La definición del campo corresponde a Crear tabla con las siguientes restricciones:

- El campo tiene que ser ya un campo de clave principal para aceptar la propiedad IDENTITY. IDENTITY significa que el campo tiene la propiedad Valor automático. Esto solo es posible para los campos INTEGER o BIGINT. Para estas descripciones de tipo de campo, consulte el Apéndice de este manual.
- Si el campo ya tiene la propiedad IDENTITY pero no se repite en la definición del campo, se elimina la propiedad IDENTITY existente.

- El valor predeterminado será el especificado en la nueva definición de campo. Si la definición del valor predeterminado se deja en blanco, se elimina cualquier valor predeterminado ya definido.
- La propiedad NOT NULL continúa en la nueva definición, si no se define de otra manera. Esto está en contraste con el valor predeterminado.
- En algunos casos, según el tipo de modificación, la tabla debe estar vacía para que se produzca el cambio. En todos los casos, el cambio tendrá efecto solo si es posible en principio (por ejemplo, un cambio de NOT NULL a NULL) y los valores existentes se pueden traducir (por ejemplo, un cambio de TINYINT a INTEGER).

```
ALTER TABLE "Nombre_Tabla" ALTER COLUMN "Nombre_Campo" RESTART WITH
<Nuevo_Valor_Campo>;
```

Esta orden se usa exclusivamente para un campo IDENTITY. Determina el siguiente valor para un campo con el conjunto de funciones Valor automático. Se puede usar, por ejemplo, cuando una base de datos se usa inicialmente con datos de prueba y luego se proporciona con datos reales. Esto requiere que se elimine el contenido de las tablas y que se establezca un nuevo valor como "1" para el campo.

```
ALTER TABLE "Nombre_Tabla" ADD [CONSTRAINT "Nombre_Condición"] CHECK
(<Condición_Búsqueda>);
```

Esto agrega una condición de búsqueda introducida por la palabra CHECK. Tal condición no se aplicará retrospectivamente a los registros existentes, pero se aplicará a todos los cambios posteriores y los registros ingresados recientemente. Si no se define un nombre de restricción, se asignará uno automáticamente. Ejemplo:

```
ALTER TABLE "Loan" ADD CHECK (IFNULL("Return_Date","Loan_Date")>="Loan_Date");
```

La tabla Loan necesita protegerse de los errores de entrada. Por ejemplo, debe evitar que se proporcione una fecha de devolución anterior a la fecha del préstamo. Si este error ocurre durante el proceso de devolución, recibirá un mensaje de error Check constraint violation…

```
ALTER TABLE "Nombre_Tabla"
ADD [CONSTRAINT "Nombre_Restricción"] UNIQUE ("Nombre_Campo1",
"Nombre_Campo2"…);
```

Aquí se agrega una condición que obliga a los campos nombrados a tener valores diferentes en cada registro. Si se nombran varios campos, esta condición se aplica a la combinación en lugar de los campos individuales. NULL no cuenta aquí. Por lo tanto, un campo puede tener el mismo valor repetidamente sin causar ningún problema, si el otro campo en cada uno de los registros es NULL. Esta orden no funcionará si ya existe una condición ÚNICA para la misma combinación de campo.

```
ALTER TABLE "Nombre_Tabla"
ADD [CONSTRAINT "Nombre_Restricción"] PRIMARY KEY ("Nombre_Campo1",
"Nombre_Campo2"…);
```

Agrega una clave primaria, opcionalmente con una restricción, a una tabla. La sintaxis de la restricción es la misma que cuando se crea una tabla.

```
ALTER TABLE "Nombre_Tabla" ADD [CONSTRAINT "Nombre_Restricción"] FOREIGN KEY
("Nombre_Campo1", "Nombre_Campo2"…)
REFERENCES "Nombre_de_otra_Tabla" ("Nombre_Campo1_otra_Tabla",
"Nombre_Campo2_otra_Tabla"…)
[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}];
```

Agrega una clave externa (FOREIGN KEY) a la tabla. La sintaxis es la misma que cuando se crea una tabla. La operación finalizará con un mensaje de error si algún valor en la tabla no tiene un valor correspondiente en la tabla que contiene esa clave primaria.

Ejemplo: las tablas *Nombre* y *Direcciones* deben vincularse. La tabla *Nombre* contiene un campo con el nombre *Direcciones_ID*. El valor de este debe estar vinculado al campo *ID* en la tabla *Direcciones*. Si el valor 1 se encuentra en *Direcciones_ID* pero no en el campo *ID* de la tabla *Direcciones*, el enlace no funcionará. Tampoco funcionará si los dos campos son de diferentes tipos.

`ALTER TABLE "Nombre_Tabla" DROP CONSTRAINT "Nombre_Restricción";` Esta orden elimina la restricción nombrada (UNIQUE, CHECK, FOREIGN KEY) de una tabla.

`ALTER TABLE "Nombre_Tabla" RENAME TO "Nuevo_Nombre_Tabla";` Finalmente, esta orden cambia solo el nombre de una tabla.

Nota

Cuando cambia una tabla usando SQL, el cambio afecta la base de datos pero no es necesariamente aparente o efectivo en la interfaz. Cuando la base de datos se cierra y se vuelve a abrir, los cambios también aparecen en la interfaz.

Los cambios también se muestran si elige **Ver > Actualizar tablas** en el contenedor de la tabla.

Eliminar tablas

```
DROP TABLE "Nombre_Tabla" [IF EXISTS] [RESTRICT | CASCADE];
```

Elimina la tabla *Nombre_Tabla*.

`IF EXISTS` evita que ocurra un error si esta tabla no existe. `RESTRICT` es la disposición predeterminada y no necesita ser elegida explícitamente; significa que la eliminación no se produce si la tabla está vinculada a otra tabla mediante el uso de una clave externa o si hay una vista activa de esta tabla. Las consultas no se ven afectadas, ya que no se almacenan en HSQLDB.

Si, en cambio, elige `CASCADE`, se eliminan todos los enlaces a la tabla *Nombre_Tabla*. En las tablas vinculadas, todas las claves externas se establecen en `NULL`. Todas las vistas que se refieren a la tabla nombrada también se eliminan por completo.

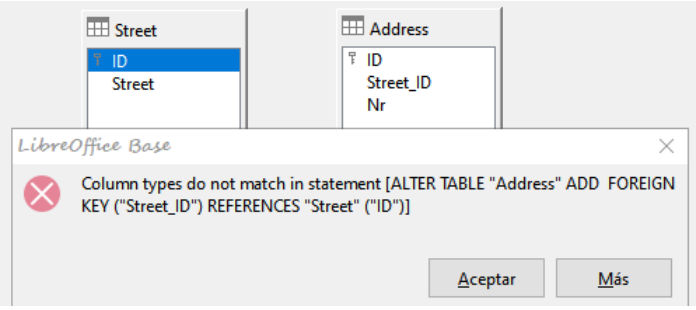
Vincular tablas

En principio, puede tener una base de datos sin enlaces entre tablas. El usuario tiene que asegurarse durante la entrada de datos, que las relaciones entre las tablas permanecen correctas. Esto generalmente se hace mediante el uso de formularios de entrada adecuados que lo manejan.

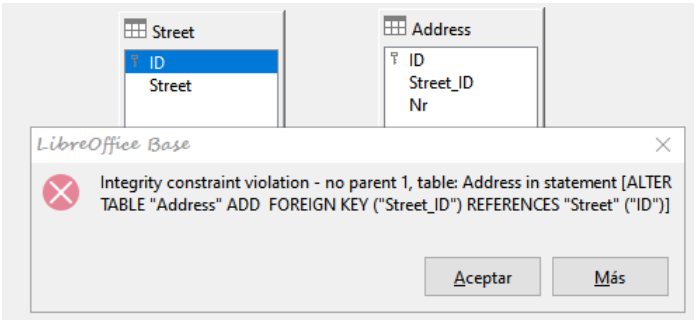
Eliminar registros en tablas vinculadas no es una cuestión simple. Suponga que desea eliminar una calle en particular de la tabla *Street* en la Figura 55, donde el campo *ID* está vinculado con la tabla *Address* como clave externa en esa tabla. Las referencias en la tabla *Address* desaparecerían. La base de datos no permite esto, una vez que se ha creado la relación. Para eliminar la calle, se debe cumplir la condición previa, que ya no se haga referencia en la tabla de direcciones.

Los enlaces básicos se hacen usando **Herramientas > Relaciones**. Esto crea una línea de conexión desde la clave primaria en una tabla a la clave externa definida en la otra.

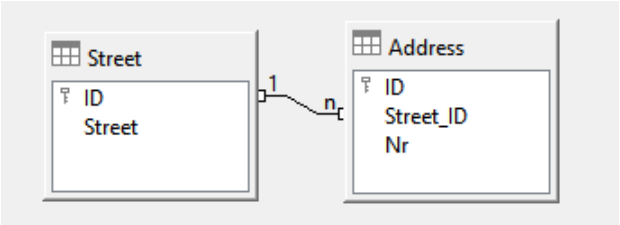
Puede recibir el siguiente mensaje de error al crear dicho enlace:



Este mensaje muestra el error que ocurrió y la orden interna de SQL que causó el error:*Column types do not match in statement*— Como también se muestra la orden SQL, la referencia es claramente a las columnas *Address.street_ID* y *Street.ID*. Para fines de prueba, uno de estos campos se definió como un Entero, el otro como Entero minúsculo. Por lo tanto, no se pudo crear ningún enlace, ya que un campo no puede ser de distinto tipo que el otro.



En este caso, los tipos de columna coinciden. La instrucción SQL es la misma que en el primer ejemplo. Pero de nuevo hay un error: Integrity constraint violation – no parent 1, table: Address — No se pudo establecer la integridad de la relación. En el campo *Street_ID* de la tabla *Address*, hay un número **1**, que no está presente en el campo *ID* de la tabla *Street*. La tabla principal aquí es *Street*, ya que su clave principal es la que debe existir. Este error es muy común, cuando se van a vincular dos tablas y algunos campos de la tabla con la clave externa prospectiva ya contienen datos. Si el campo de clave externa contiene una entrada que no está presente en la tabla principal (la tabla que contiene la clave primaria), esta es una entrada no válida.



Si la vinculación se lleva a cabo con éxito y posteriormente se intenta introducir un registro igualmente inválido en la tabla, aparece el siguiente mensaje de error:

De nuevo esto es una violación de integridad. Base se niega a aceptar el valor 1 para el campo *street_ID* después de que se haya realizado el enlace porque la tabla *Street* no contiene dicho valor en el campo *ID*.

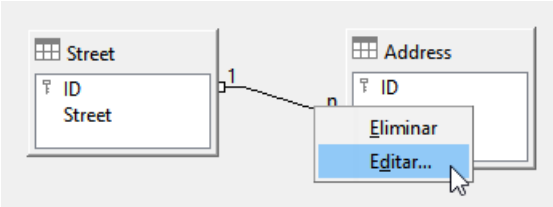


Figura 60: Los enlaces se editan con clic derecho

Las propiedades de un enlace se pueden editar para que la eliminación de un registro de la tabla *Street*, establezca simultáneamente en `NULL` las entradas correspondientes en la tabla *Address*.

Las propiedades que se muestran en la Figura 60 siempre se relacionan con una acción vinculada al cambio en un registro de la tabla que contiene la clave primaria correspondiente. En nuestro caso esta es la tabla *Street*.

Si se modifica la clave principal de un registro en esta tabla (*Opciones de actualización*), podrían ocurrir las siguientes acciones:

Sin acción
Cambiar la clave primaria *Street.ID* no está permitido en este caso, ya que rompería la relación entre las tablas.

Actualizar en cascada
Si se cambia la clave principal *Street.ID*, la clave externa se cambia automáticamente a su nuevo valor. Esto asegura que el enlace no esté dañado. Por ejemplo, si un valor se cambia de 3 a 4, todos los registros de la tabla *Address* que contienen la clave externa *Address.Street_ID* con el valor 3, se cambian a 4.

Definir NULL
Todos los registros que contienen esta clave primaria en particular ahora no tendrán entrada en el campo de clave externa *Address.Street_ID*; El campo será `NULL`.

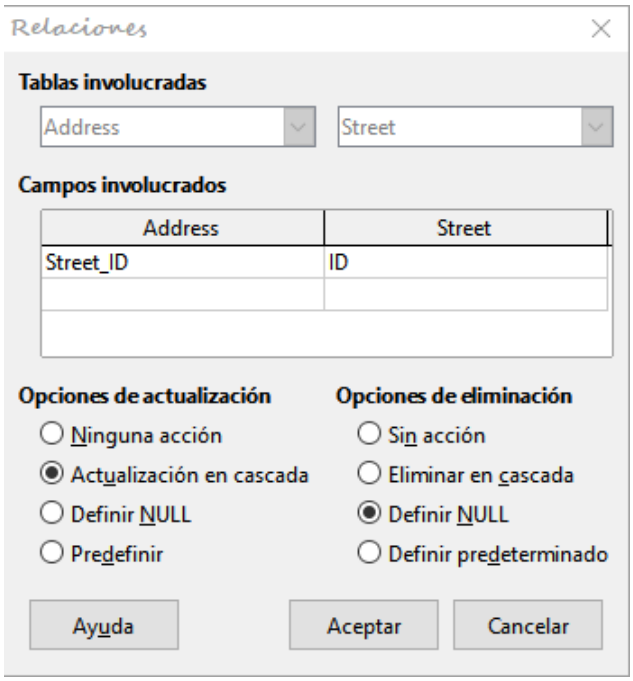


Figura 61: Editar las propiedades de la relación

Definir predeterminado

Si se cambia la clave principal *Street_ID*, el valor de *Address.Street_ID* originalmente vinculado a ella se establece en el valor predeterminado previamente definido. Para este propósito, necesitamos una definición inequívoca de un valor predeterminado. Si el valor predeterminado se establece utilizando la instrucción SQL:

```
ALTER TABLE "Address" ALTER COLUMN "Street_ID" SET DEFAULT 1;
```

la definición del enlace asegura que el campo volverá a este valor en el caso de una Actualización. Por lo tanto, si se cambia la clave principal en la tabla Street, la clave externa correspondiente en la tabla *Address* se establecerá en `1`. Esto es útil cuando se requiere un registro para tener un campo *street*, en otras palabras, este campo no puede ser `NULL`. Pero tenga cuidado: si `1` no está en uso, habrá creado un enlace a un valor inexistente. Por lo tanto, es posible destruir la integridad de la relación.

Precaución

Si el valor predeterminado en un campo de clave externa no está vinculado a un valor de clave primaria de la tabla externa, se creará un enlace a un valor que no es posible. La integridad referencial de la base de datos sería destruida.

Sería mejor **no utilizar** la posibilidad de establecer el valor predeterminado.

Si se elimina un registro de la tabla *Street* (*Opciones de eliminación*), las siguientes opciones están disponibles:

Sin acción

No se lleva a cabo ninguna acción. Si la eliminación solicitada afecta un registro en la tabla *Address*, la solicitud será rechazada.

Eliminar en cascada

Si se elimina un registro de la tabla *Street* y esto afecta a un registro en la tabla *Address*, ese registro también se eliminará.

Puede parecer extraño en este contexto, pero hay otras estructuras de tabla en las que tiene sentido. Supongamos que tiene una tabla *CD* y una tabla que almacena los títulos en estos CD. Ahora, si se elimina un registro en la tabla *CD*, muchos títulos en la otra tabla no tienen sentido ya que ya no están disponibles para usted. En tales casos, una eliminación en cascada tiene sentido. Significa que no necesita eliminar todos los títulos antes de eliminar el CD de la base de datos.

Definir NULL

Esto es lo mismo que para la opción de actualización.

Definir predeterminado

Esto es lo mismo que para la opción de actualización y requiere las mismas precauciones.

Consejo

La opción *Sin acción* debe evitarse en la mayoría de los casos para que no se muestren mensajes de error de la base de datos al usuario, ya que estos no son facilmente comprensibles.

Las relaciones a claves foráneas que se refieren a un solo campo de otra tabla se establecen arrastrando con el ratón un campo seleccionado hacia el otro campo de la otra tabla.

Para vincular a una tabla que tiene una clave principal compuesta, vaya a **Herramientas > Relaciones**, luego **Insertar > Nueva relación**, o use el botón correspondiente. Aparece un cuadro

de diálogo para editar las propiedades de una relación con la posibilidad de elegir las tablas disponibles.

Introducir datos en tablas

Las bases de datos que consisten en una sola tabla generalmente no requieren un formulario de entrada a menos que contengan un campo para imágenes. Sin embargo, cuando una tabla contiene claves externas de otras tablas, los usuarios deben recordar qué números de clave ingresar o mirar las otras tablas simultáneamente. En tales casos, un formulario es más útil.

Entrada utilizando la interfaz de base

Las tablas en el *contenedor de tablas* se abren haciendo doble clic en su nombre. Si la clave primaria es un campo que se incrementa automáticamente, uno de los campos visibles contendrá el texto *Valor automático*. No es posible introducir ningún valor en el ese campo. Su valor asignado puede modificarse si es necesario, pero solo después de que se haya confirmado el registro.

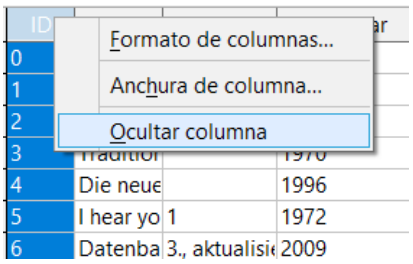


Figura 62: Ocultar columnas

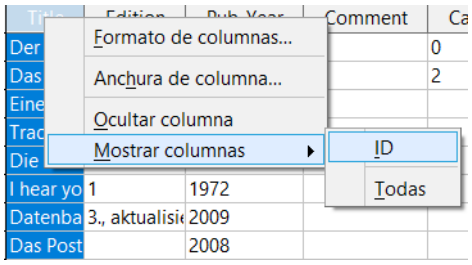


Figura 63: Mostrar columnas

Las columnas individuales en la Vista de datos de la tabla se pueden ocultar. Por ejemplo, si el campo de la clave primaria no necesita ser visible, esto se puede especificar en la tabla en la vista de ingreso de datos haciendo clic derecho en el encabezado de la columna. Esta configuración se almacena en los ajustes de la interfaz. La columna sigue existiendo en la tabla y siempre se puede volver a hacer visible.

La entrada de registros en la tabla generalmente se realiza de izquierda a derecha usando el teclado (teclas *Tab* o *Intro*) o con el ratón.

Cuando se alcanza el último campo de un registro, el cursor salta automáticamente al siguiente registro. La entrada anterior se almacena. El almacenamiento adicional usando **Archivo > Guardar** no es necesario y, de hecho, no es posible. Los datos ya están en la base de datos.

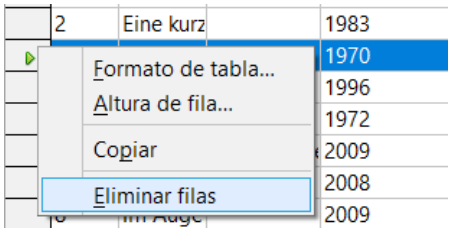
Precaución

Para HSQLDB, los datos están en la memoria de trabajo. Solo se transferirán al disco duro cuando Base esté cerrada (desafortunadamente es una debilidad desde el punto de vista de la seguridad de los datos). Si Base por alguna razón no se cierra de manera ordenada, puede conducir a la pérdida de datos.

Si no se ingresan datos en un campo que se haya definido previamente durante el diseño de la tabla como obligatorio (NOT NULL), se muestra el correspondiente mensaje de error:Attempt to insert null into a non-nullable column.

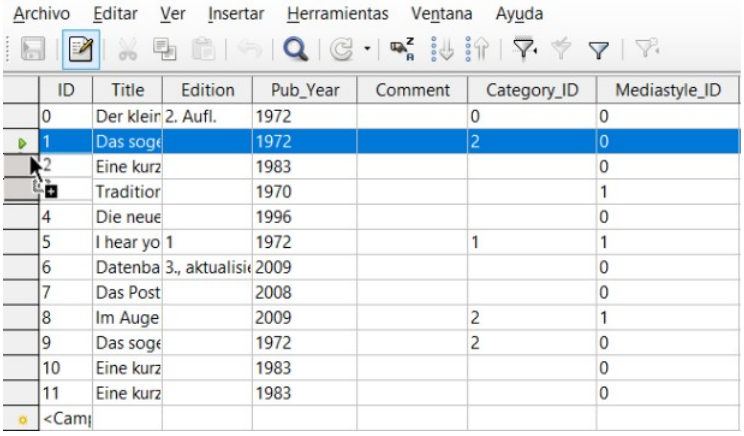
También se muestran la columna correspondiente, la tabla y la orden SQL (según lo traducido por la interfaz).

Cambiar un registro es fácil: busque el campo, ingrese un valor diferente y abandone el registro.



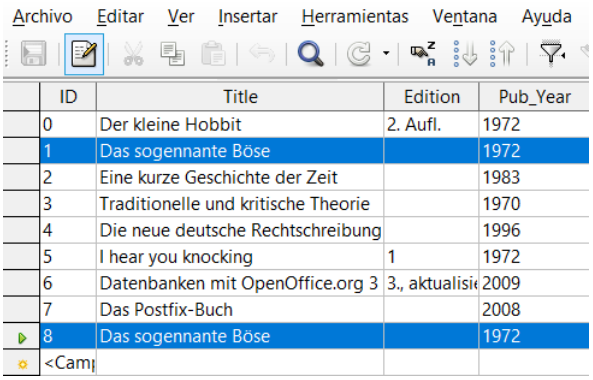
Para eliminar un registro, seleccione la fila haciendo clic en su encabezado (el área gris de la izquierda), haga clic con el botón derecho y elija Eliminar filas.

Hay un método, no visible, para copiar filas completas. Para que esto funcione, la clave primaria de la tabla debe estar definida como *Valor automático*.



ID	Title	Edition	Pub_Year	Comment	Category_ID	Mediastyle_ID
0	Der klein	2. Aufl.	1972		0	0
1	Das soge		1972		2	0
2	Eine kurz		1983			0
	Traditior		1970			1
4	Die neue		1996			0
5	I hear yo	1	1972		1	1
6	Datenba	3., aktualisi	2009			0
7	Das Post		2008			0
8	Im Auge		2009		2	1
9	Das soge		1972		2	0
10	Eine kurz		1983			0
11	Eine kurz		1983			0
<Cam						

Primero se resalta el encabezado de la fila con el botón izquierdo del ratón. Mantenga presionado el botón y arrastre el ratón. El cursor cambiará a un símbolo con un signo +. Esto significa que el registro debe ser copiado. Tan pronto aparezca el símbolo se puede soltar el botón del ratón.



ID	Title	Edition	Pub_Year
0	Der kleine Hobbit	2. Aufl.	1972
1	Das sogennante Böse		1972
2	Eine kurze Geschichte der Zeit		1983
3	Traditionelle und kritische Theorie		1970
4	Die neue deutsche Rechtschreibung		1996
5	I hear you knocking	1	1972
6	Datenbanken mit OpenOffice.org 3	3., aktualisi	2009
7	Das Postfix-Buch		2008
8	Das sogennante Böse		1972
<Cam			

En el ejemplo, el registro con la clave primaria **1** se inserta como un nuevo registro con la nueva clave primaria **9**. Si utiliza la tecla control o shift para resaltar un grupo de registros, estos se copiarán como un grupo.

Consejo

Los encabezados de columna se pueden arrastrar para obtener un ancho adecuado para la entrada. Si esto se hace en una tabla, Base guarda automáticamente el nuevo ancho para la columna en la tabla.

Los anchos de columna en las tablas afectan a los de las consultas. Si las columnas de una consulta son demasiado estrechas, ampliarlas en la consulta solo tendrá un efecto temporal. El nuevo ancho no se guardará. Se debe ampliar la columna en la tabla para que aparezca correctamente en las consultas.

Las funciones *Ordenar*, *Buscar* y *Filtrar* son muy útiles para recuperar registros concretos.

Ordenar Tablas

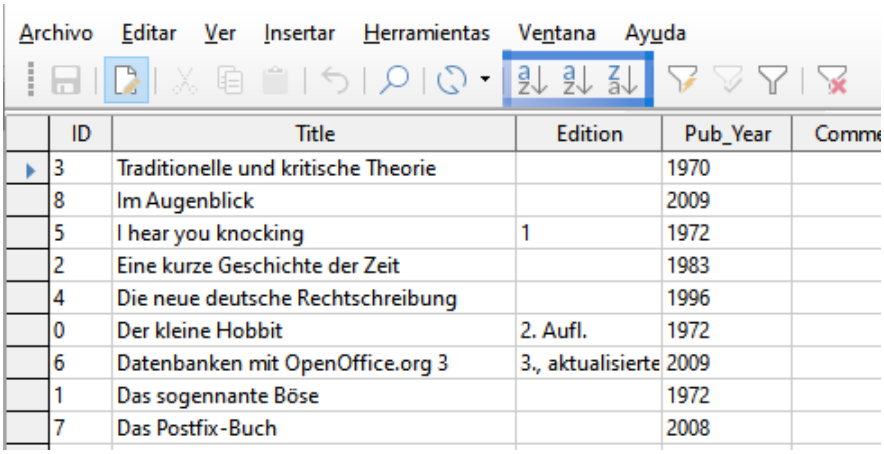


Figura 64: Ordenación rápida

Los botones resaltados en la figura 64 son los botones de ordenación. El segundo y tercer botón permiten una ordenación rápida. Primero, seleccione un campo. Luego, haga clic en el botón correspondiente al orden ascendente o descendente, y los datos se ordenan por esa columna. En este caso se muestra un orden descendente por el campo *Title*.

La ordenación rápida solo es efectiva para una columna. Para ordenar por varias columnas simultáneamente, se proporciona una función de clasificación más avanzada con el primer botón de ordenación:

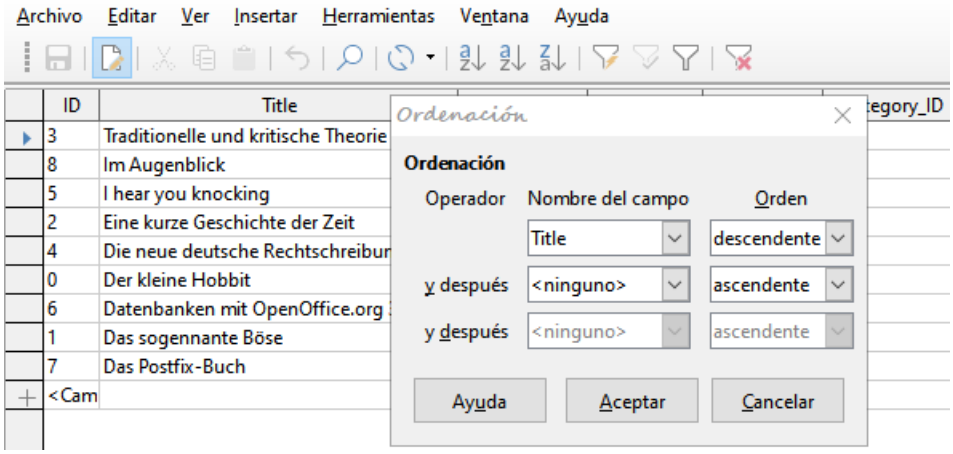
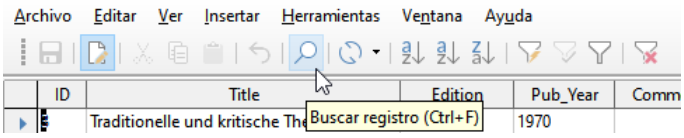


Figura 65: Ordenar por más de una columna

Se seleccionan el nombre del campo de la columna y el orden de clasificación. Si se ha llevado a cabo una ordenación rápida previa, la primera fila ya contendrá los correspondientes nombre del campo y orden de clasificación.

Buscar en Tablas



El botón Buscar registro es un método simple para localizar registros en una tabla grande. Sin embargo, la función de búsqueda es muy lenta para bases de datos grandes, ya que la búsqueda no utiliza una orden SQL dentro de la base de datos. Para una búsqueda más rápida, en lugar de

usar Buscar registro, use una consulta. Para evitar la modificación frecuente de la consulta, puede diseñarse para ejecutarse con parámetros. Consulte la sección «Uso de parámetros en consultas» en el «Capítulo 5, Consultas».

Consejo

Antes iniciar la búsqueda, asegúrese de que las columnas en las que buscará sean lo suficientemente anchas como para mostrar correctamente los registros que se encuentren. La ventana de búsqueda permanece en primer plano y no podrá corregir la configuración del ancho de columna en la tabla subyacente. Para acceder a la tabla, debe interrumpir la búsqueda.

El botón Buscar registros rellena automáticamente el término de búsqueda con el contenido del campo seleccionado cuando se invoca.

Para que la búsqueda sea efectiva, el área de búsqueda debe ser lo más limitada posible. No tendría sentido buscar el texto de un campo *Título* en un campo *Author*. El nombre de campo *Título* ya se sugiere como el nombre del campo único.

A través de combinaciones específicas pueden obtenerse resultados más precisos. Puede usar los marcadores de posición SQL normales ("_" para un carácter variable, "%" para un número arbitrario de caracteres variables o "\" como carácter de escape para permitir que se busquen caracteres especiales).

Las expresiones regulares se describen en detalle en la *Ayuda de LibreOffice*. Aparte de estas expresiones, la Ayuda disponible para este módulo es bastante escasa.

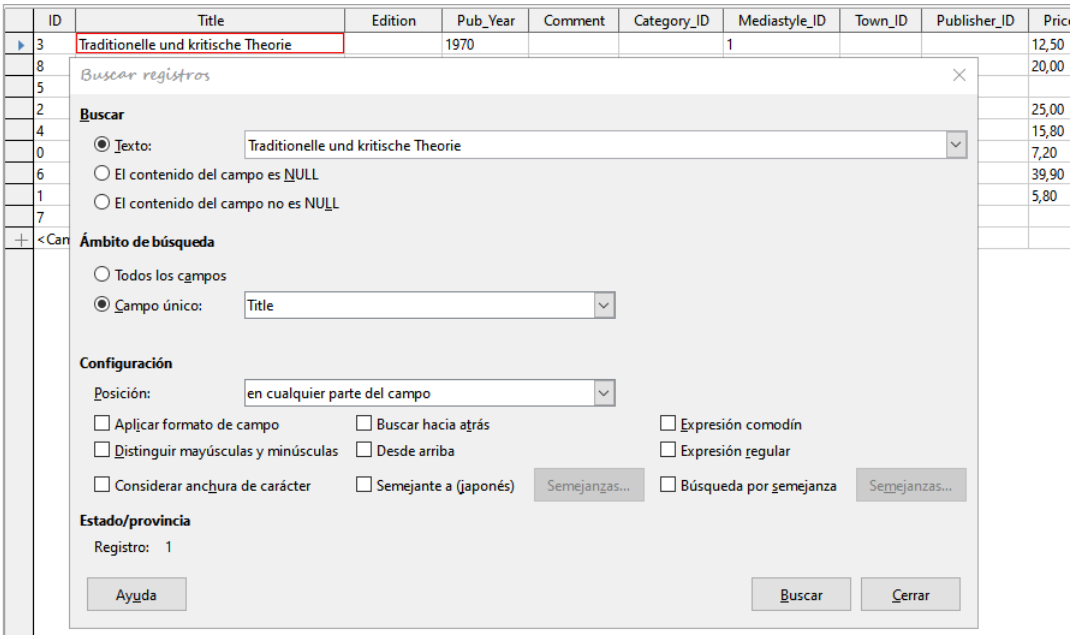


Figura 66: Máscara de entrada para buscar registros

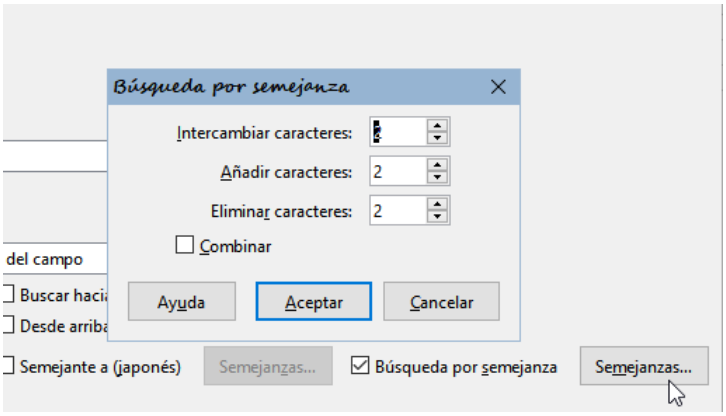


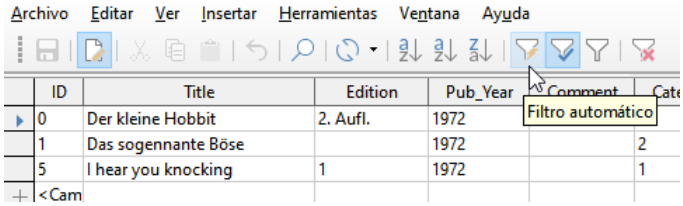
Figura 67: Limitar la búsqueda de similitudes

La función de búsqueda por semejanza es útil cuando necesita excluir errores ortográficos. Cuanto más altos sean los valores que establezca, más registros se mostrarán en la lista final.

Este módulo de búsqueda es el más adecuado para las personas que saben, por el uso regular, exactamente cómo lograr un resultado dado. La mayoría de los usuarios tienen más probabilidades de encontrar registros utilizando un filtro.

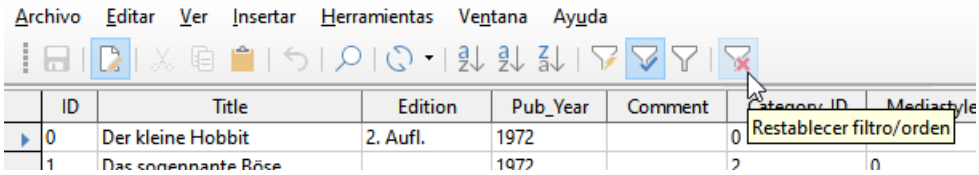
El «Capítulo 4» de esta guía describe el uso de formularios para la búsqueda y cómo el uso de SQL y macros puede lograr una búsqueda con palabras clave.

Filtrar tablas



Puede filtrar una tabla rápidamente utilizando el *Filtro automático*. Coloque el cursor en un campo, y un clic en el icono hace que el filtro se haga cargo del contenido de este campo. Solo se muestran aquellos registros para los que el campo elegido tiene el mismo contenido. La siguiente figura muestra el filtrado de acuerdo con una entrada en la columna *Pub_Year*.

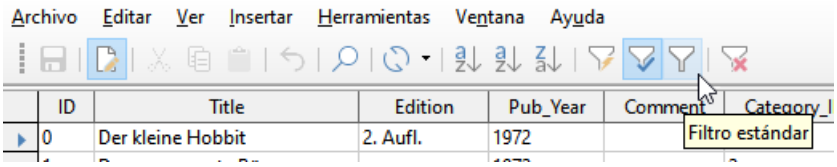
El filtro está activo, como lo muestra el icono del filtro con una marca de verificación. El símbolo de filtro se muestra presionado. Este botón es una palanca, por lo que si se hace clic nuevamente, el filtro continúa existiendo, pero ahora se muestran todos los registros. Por lo tanto, si lo desea, siempre puede volver al estado filtrado.



Al hacer clic en el icono *Restablecer filtro/orden* en el extremo derecho, se eliminan todos los filtros y tipos existentes. Los filtros se vuelven inactivos y ya no se pueden recuperar con sus valores anteriores.

Consejo

Todavía se pueden ingresar registros normalmente en una tabla filtrada o en una que haya sido restringida por una búsqueda. Permanecen visibles en la vista de tabla hasta que la tabla se actualiza presionando el botón *Actualizar.*



El icono *Filtro estándar* abre un cuadro de diálogo en el que puede filtrar utilizando varios criterios simultáneos, de forma similar a la ordenación. Si el *Filtro automático* está en uso, la primera línea del Filtro estándar ya mostrará este criterio de filtro existente.

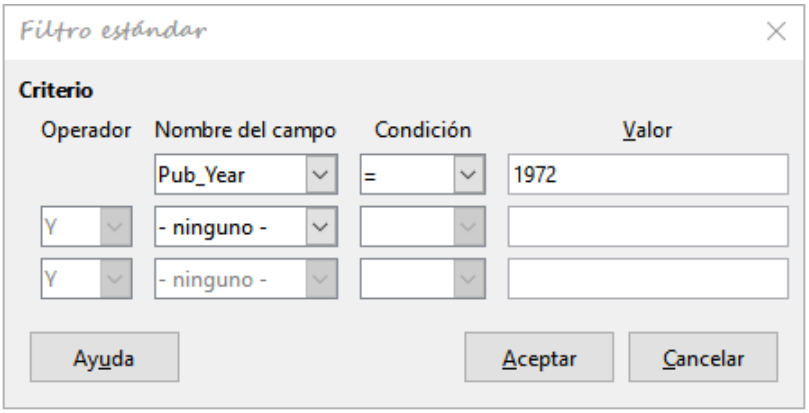


Figura 68: Filtrado de datos múltiples con el filtro estándar

El filtro estándar proporciona muchas de las funciones de filtrado de datos SQL. Las siguientes órdenes SQL en el desplegable *Condición* están disponibles:

Condición	***Descripción***
=	Igualdad exacta corresponde a como, pero sin ningún marcador de posición adicional
<>	No igual
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
como	Para texto, escrito entre comillas (''); "_" para un carácter variable. "%" para un número arbitrario de caracteres variables
no como	Opuesto a como. En SQL NOT LIKE
nulo	Sin entrada, ni siquiera un carácter de espacio. En SQL, NULL
No nulo	Opuesto a nulo. En SQL NOT NULL

Antes de que un criterio de filtro se pueda combinar con otro, la siguiente fila debe tener al menos un nombre de campo seleccionado. En la Figura 68, la palabra - ninguno - se muestra en lugar de un nombre de campo, por lo que la combinación no está activa. Los operadores de combinación disponibles son Y(AND) y O (OR).

El nombre del campo puede ser un nuevo nombre de campo o uno previamente seleccionado.

Incluso para grandes masas de datos, el número de registros recuperados puede reducirse a un conjunto más manejable con un filtrado hábil utilizando estas tres posibles condiciones.

En el caso de los formularios de filtrado también, hay algunas posibilidades adicionales (descritas en el «Capítulo 4») que no están disponibles en la interfaz.

Entrada directa usando SQL

La entrada directa de datos usando SQL es útil para ingresar, cambiar o eliminar múltiples registros con una sola órden.

Introducir nuevos registros

```
INSERT INTO "Nombre_Tabla" [( "Nombre_Campo" [,...] )] { VALUES("Valor_campo" [,...]) |
<Fórmula_Selección>};
```

Si no se especifica *Nombre_Campo*, todos los campos se deben completar y en el orden correcto (como se establece en la tabla). Eso incluye el campo de clave primaria incrementado automáticamente, donde está presente. Los valores ingresados también pueden ser el resultado de una consulta (<Fórmula_Selección>). A continuación se proporciona información más exacta.

```
INSERT INTO "Nombre_Tabla" ("Nombre_Campo") VALUES ('Test');
CALL IDENTITY();
```

En la tabla, en la columna *Nombre_Campo*, se inserta el valor *Test*. No se toca la *ID* del campo de la clave primaria incrementada automáticamente. El valor correspondiente para *ID* debe crearse por separado utilizando CALL IDENTITY (). Esto es importante cuando se utilizan macros, para que el valor de este campo clave se pueda utilizar más adelante.

```
INSERT INTO "Nombre_Tabla" ("Nombre_Campo") SELECT "Otro_nombre_Campo" FROM
"Nombre_de_otra_Tabla";
```

En la primera tabla, se insertan tantos registros nuevos en *Nombre_Campo*, como están presentes en la columna *Otro_nombre_Campo* de la segunda tabla. Naturalmente, para limitar el número de entradas, se puede utilizar aquí una fórmula de selección.

Editar registros existentes

```
UPDATE "Nombre_Tabla" SET "Nombre_Campo" = <Expresión> [, ...] [WHERE
<Expresión>];
```

Cuando está modificando muchos registros a la vez, es muy importante verificar cuidadosamente la orden SQL que está ingresando. Suponga que todos los estudiantes en una clase deben ascender un año:

```
UPDATE "Nombre_Tabla" SET "Año" = "Año"+1;
```

Nada podría ser más rápido: todos los registros de datos se modifican con una sola orden. Pero, imagine que ahora debe determinar qué estudiantes no deberían haberse visto afectados por este cambio. Hubiera sido más sencillo usar un campo *Repetidor* tipo `Sí/No` para los repetidores y luego subir solo a aquellos estudiantes para los que **no** se marcó este campo:

```
UPDATE "Nombre_Tabla" SET "Año" = "Año"+1 WHERE "Repeticion" = FALSE;
```

Estas condiciones funcionan cuando el campo en cuestión solo puede tomar dos valores: `TRUE` o `FALSE`, pero en este caso puede tomar un valor vacío `NULL`. Sería más seguro si la condición se formulara como `WHERE "Repeticion" <> TRUE.`

Si posteriormente desea que se ingrese un valor predeterminado en un campo en particular donde esté vacío, puede hacerlo con la órden:

```
UPDATE "Tabla" SET "Campo" = 1 WHERE "Campo" IS NULL;
```

Puede modificar varios campos a la vez asignándoles directamente valores. Suponga que una tabla para libros incluye los nombres de sus autores. Se descubre que *Erich Kästner* ha sido frecuentemente ingresado como *Eric Käschtner*.

```
UPDATE "Libros" SET "Nombre_Autor" = 'Erich', "Apellido_Autor" = 'Kästner' WHERE "Nombre_Autor" = 'Eric' AND "Apellido_Autor" = 'Käschtner';
```

También es posible hacer cálculos con UPDATE. Si, por ejemplo, las mercancías que cuestan más de 150,00 € se incluyen en una oferta especial y el precio se reduce en un 10%, se puede llevar a cabo de la siguiente manera:

```
UPDATE "Nombre_Tabla" SET "Precio" = "Precio"*0.9 WHERE "Precio" >= 150.00
```

Cuando elige el tipo de datos CHAR, el campo tiene un ancho fijo. Cuando haga falta, el texto se rellenará con caracteres nulos. Si se convierte a VARCHAR, estos caracteres nulos permanecen. Para eliminarlos, use la función RTRIM para eliminar caracteres de vacíos o de control de la derecha:

```
UPDATE "Nombre_Tabla" SET "Nombre_Campo" = RTRIM("Nombre_Campo");
```

Eliminar registros existentes

```
DELETE FROM "Nombre_Tabla" [WHERE <Expresión>];
```

Sin la expresión condicional, la órden…

```
DELETE FROM "Nombre_Tabla";
```

elimina todo el contenido de la tabla.

Por esta razón, es preferible que la orden sea más específica. Por ejemplo, si se proporciona el valor de la clave primaria, solo se eliminará el registro con esa clave.

```
DELETE FROM "Nombre_Tabla" WHERE "ID" = 5;
```

Si, en el caso de un préstamo, el registro de un artículo se debe eliminar cuando se devuelva el artículo, se puede hacer usando:

```
DELETE FROM "Nombre_Tabla" WHERE NOT "Return_date" IS NULL;
```

o de manera alternativa con:

```
DELETE FROM "Nombre_Tabla" WHERE "Return_date" IS NOT NULL;
```

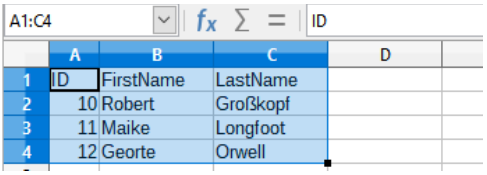
Importar datos de otras fuentes

A veces necesitamos importar a Base conjuntos de datos de otro programa a través del portapapeles. Esto implica crear una nueva tabla o agregar registros a una existente.

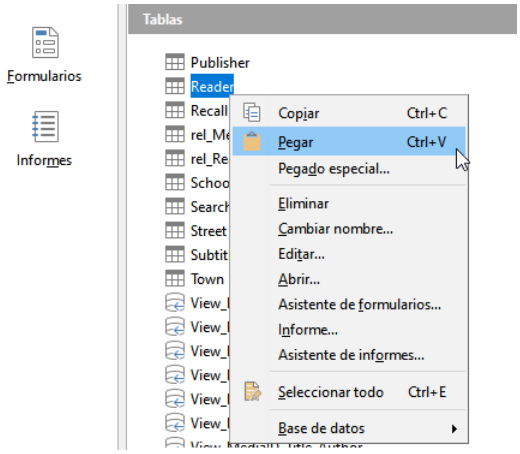
Nota

Para importar datos usando el portapapeles, Base debe poder leer el formato de datos. Este siempre será el caso para los archivos de datos abiertos en LibreOffice.

Por ejemplo, si las tablas de una base de datos externa deben leerse en un archivo *.odb, esa base de datos primero debe abrirse en LibreOffice o registrarse con LibreOffice como fuente de datos. Consulte «Acceso a bases de datos externas» en el Capítulo 2, «Creación de una base de datos».

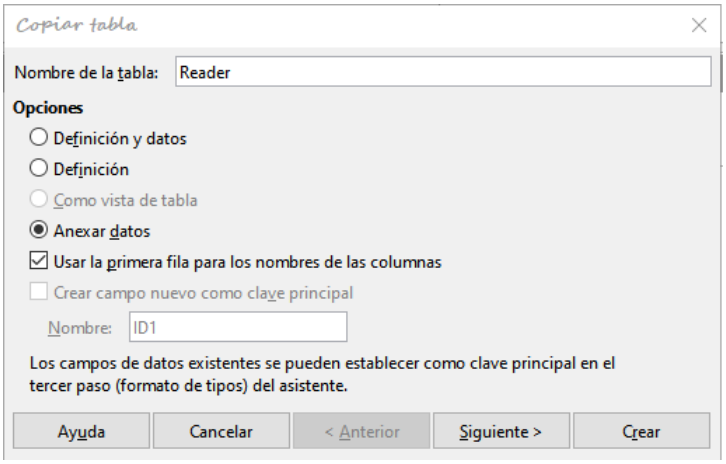


Aquí se ha copiado una pequeña tabla de ejemplo de una hoja de cálculo Calc en el portapapeles. Luego se pega en el contenedor de la tabla de Base. Por supuesto, esto también podría haberse hecho seleccionándolo con el botón izquierdo del ratón y luego arrastrándolo.

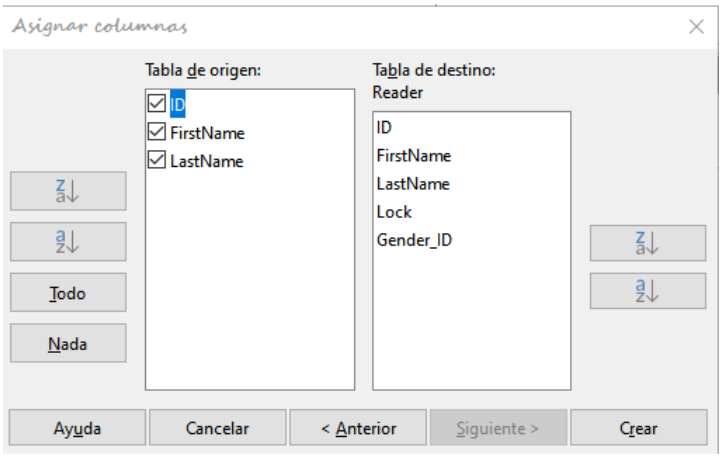


En el contenedor de tablas, haga clic con el botón derecho para abrir el menú contextual de la tabla a la que se agregarán los registros.

Agregar registros importados a una tabla existente



El nombre de la tabla aparece en el asistente de importación. Seleccionar Al mismo tiempo *Anexar datos* y *Usar la primera fila para los nombres de las columnas*, puede ser necesario o no, dependiendo de su versión de LibreOffice. Si se van a agregar los registros, no se requiere definición de datos. Una clave principal también tiene que estar disponible para su uso.



Las columnas de la tabla de origen Calc y la tabla de destino en Base no tienen que coincidir en su secuencia, nombres o número general. Solo se transfieren los elementos seleccionados del lado izquierdo. La correspondencia entre las tablas de origen y destino debe ajustarse utilizando los botones de flecha a cada lado.

Esto completa la importación.

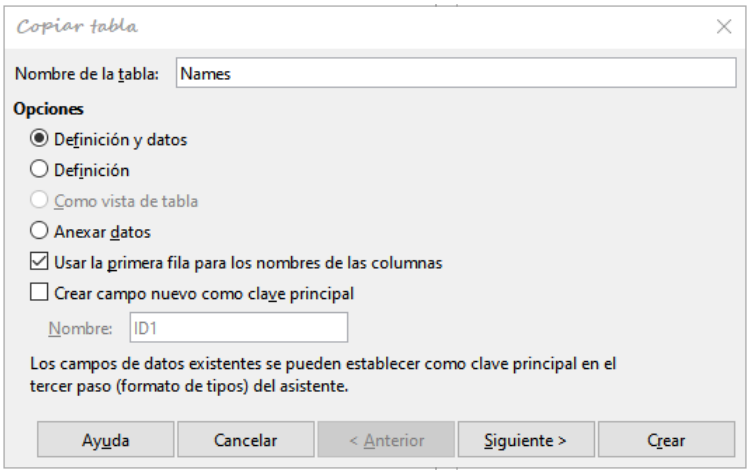
La importación puede generar problemas si:

- Los campos en la tabla de destino requieren una entrada obligatoria, pero la tabla de origen no proporciona datos para ellos.
- Las definiciones de campo en la tabla de destino no están en concordancia con las de la tabla de origen (por ejemplo, se debe ingresar un nombre en un campo numérico, o el campo de destino tiene muy pocos caracteres para los datos).
- La tabla de origen proporciona datos incongruentes con los de la tabla de destino, por ejemplo, valores no únicos para claves primarias u otros campos definidos como únicos.

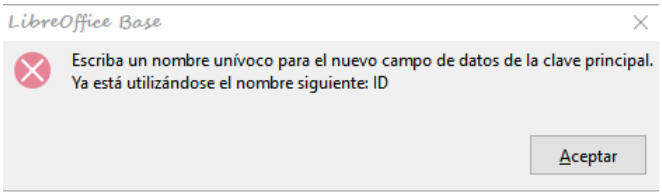
Crear una nueva tabla para datos importados

Se puede crear una nueva tabla seleccionando por ejemplo un rango de una hoja de cálculo Calc, copiandolo al portapapeles y al hacer clic derecho en el contenedor de tablas seleccionando *Pegar* del menú emergente.

Cuando se inicia el asistente de copia, el nombre de la tabla que estaba seleccionada anteriormente en el contenedor de tablas aparece automáticamente. Debe cambiar el nombre si está creando una nueva tabla, ya que no es posible tener dos tablas con el mismo nombre en la base de datos . El nombre de la nueva tabla en el ejemplo es *Names*. En la importación, se transfieren la definición de la tabla y los datos, y se usa la primera fila para los nombres de las columnas.



En este punto, puede crear un nuevo campo adicional para una clave primaria. El nombre de este campo no debe existir como encabezado de columna en la tabla Calc. De lo contrario, recibirá el mensaje de error:



Lamentablemente, este mensaje no explica la situación correctamente.

Si desea utilizar un campo existente del rango de datos como su clave principal, no marque *Crear campo nuevo como clave principal*. Establecerá su campo de clave principal en la tercera página del Asistente (figura 69).

Todas las columnas disponibles se transfieren usando el segundo botón y pasarán del área izquierda a la derecha (>I).

El formato de los tipos de tabla a menudo requiere un ajuste. De manera predeterminada, los campos se definen como campos de texto con un tamaño muy grande.

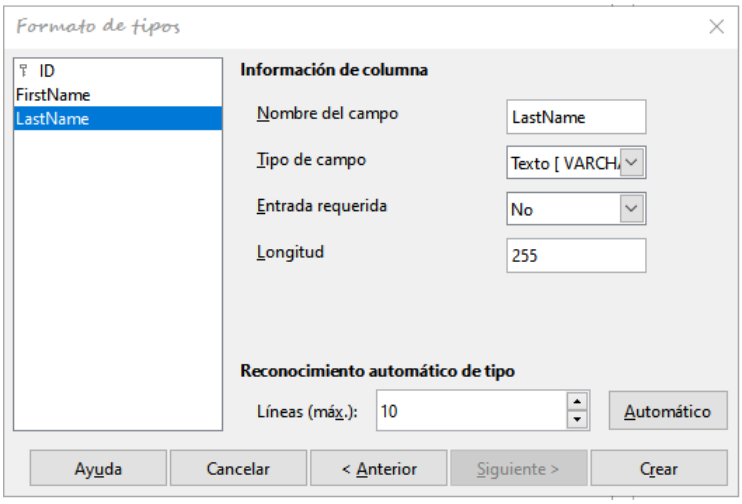
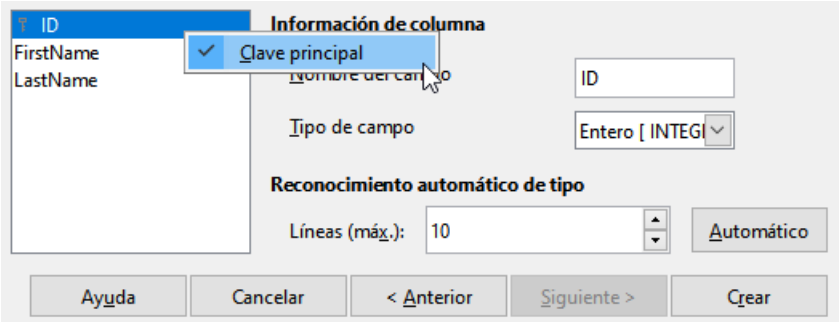


Figura 69: Tercera página del Asistente

Por lo tanto, deberá ajustar su tamaño si así lo desea. Los campos numéricos o de fecha se deben establecer del tipo adecuado con el desplegable *Tipo de campo*. En el caso de los números decimales, deberá también verificar el número de lugares decimales.



La opción para elegir una clave principal está presente, pero algo oculta, en el menú contextual del campo que queramos designar. En este ejemplo, el campo *ID* se ha formateado de manera que permita su uso como clave principal. Después debe establecerse la clave primaria usando el menú contextual del nombre del campo, si no se eligió *crear campo nuevo como clave principal* en el primer paso del asistente (aparecerá un icono de una llave a la izquierda del campo).

Cuando hace clic en el botón *Crear*, la tabla se crea y se llena con los datos copiados. La nueva clave primaria no es una clave de Valor automático. Debe editar la tabla para cambiarla si desea un valor automático y también si desea realizar más operaciones de formateo en el resto de campos.

División de datos al importar

A veces, los datos de origen no están disponibles en la forma deseada. Las direcciones, por ejemplo, a menudo se introducen en hojas de cálculo como un solo registro (calle, número, ciudad y código postal). Al importarlos, es posible que desee colocar la ciudad y el código postal en una tabla separada, que luego se puede vincular a la tabla principal *Addresses*.

La siguiente es una forma posible de crear esta relación directamente:

1) La tabla completa con toda la información de la dirección se importa a Base como una tabla llamada *Addresses*. (Vea las secciones anteriores para más detalles).
2) Los datos del campo Código postal y Ciudad se leen con una consulta,

```
SELECT DISTINCT "Postcode", "Town" FROM "Addresses;
```

3) Se copian y almacenan como una tabla separada *Postcode_Town*. Para ello, se agrega un campo *ID* y se especifica como una clave principal con Valor automático.Aquí está la consulta:
4) Se agrega un nuevo campo llamado *Postcode_ID* a la tabla *Addresses* del mismo tipo que el campo *ID* de la tabla *Postcode_Town*
5) Usando **Herramientas > SQL**, se realiza una actualización de la tabla *Addresses*:

```
UPDATE "Addresses" AS "a" SET "a"."Postcode_ID" = (SELECT "ID" FROM "Postcode_Town"
WHERE "Postcode"||"Town" = "a"."Postcode"||"a"."Town");
```

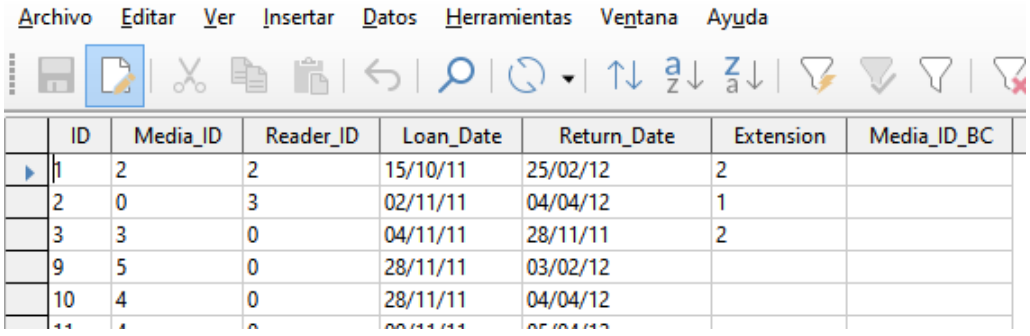
6) La tabla *Addresses* se abre para editar y los campos Código postal y Ciudad se eliminan. Este cambio se guarda y la tabla se cierra nuevamente.

Una vez separadas las tablas se crea una relación `1:n` entre la tabla *Postcode_Town* y la tabla *Addresses*. Esta relación se define usando **Herramientas > Relaciones**.

Para obtener detalles sobre el código SQL, consulte también el Capítulo 5, «Consultas».

Problemas con estos métodos de entrada de datos

La entrada utilizando una tabla sola no tiene en cuenta los enlaces a otras tablas. Esto queda claro en un ejemplo de un préstamo de artículos.



ID	Media_ID	Reader_ID	Loan_Date	Return_Date	Extension	Media_ID_BC
1	2	2	15/10/11	25/02/12	2	
2	0	3	02/11/11	04/04/12	1	
3	3	0	04/11/11	28/11/11	2	
9	5	0	28/11/11	03/02/12		
10	4	0	28/11/11	04/04/12		

La tabla *Loan* contiene claves externas para el artículo prestado (*Media_ID*) y el lector correspondiente (*Reader_ID*), así como una fecha de préstamo (*Loan_Date*). En la tabla, por lo tanto, debemos ingresar en el momento del préstamo dos valores numéricos (número de artículo y número de lector) y una fecha. La clave principal se ingresa automáticamente en el campo *ID*. Si el lector realmente corresponde al número no es manifiesto a menos que una segunda tabla para los lectores esté abierta al mismo tiempo. Si el artículo se prestó con el número correcto tampoco es evidente. Aquí el préstamo debe basarse en la etiqueta del artículo o en otra tabla abierta.

Todo esto es mucho más fácil de lograr usando formularios. Los lectores y los artículos se pueden buscar utilizando controles de cuadro de lista. En los formularios, los nombres de usuario y artículo son visibles y sus identificadores numéricos están ocultos. Además, un formulario puede diseñarse de modo que se pueda seleccionar primero un usuario, luego una fecha de préstamo, y cada conjunto de artículos se asigna a esta fecha por número. En otros lugares, estos números pueden hacerse visibles con descripciones exactas que correspondan a los artículos.

La entrada directa en tablas es útil solo para bases de datos con tablas simples. Cuando existan relaciones entre tablas, es mejor utilizar un formulario especialmente diseñado. En los formularios, estas relaciones se pueden manejar utilizando subformularios o campos de lista.

LibreOffice

Guía de Base

Capítulo 4

Formularios

Los formularios facilitan la entrada de datos

Los formularios se usan cuando no es conveniente la entrada directa en una tabla, para detectar errores en la entrada de datos rápidamente o cuando demasiadas tablas hacen imposible la administración directa de datos.

Nota

Un formulario en Base es una estructura invisible para el usuario. Sirve dentro de la aplicación para permitir el contacto con la base de datos. Lo que es visible para el usuario es el conjunto de controles, que sirven para la entrada o visualización de texto, números, etc. Estos controles están divididos por la interfaz de usuario en varios tipos.

Figura 70

El término **formulario** tiene dos significados. Puede representar todo el contenido de la ventana de entrada que se utiliza para administrar los datos de una o más tablas. Esta ventana puede contener uno o más formularios principales y cada uno de ellos puede contener subformularios.

El término formulario también se usa para estas áreas parciales. A partir de este contexto, debe quedar claro el significado para evitar malentendidos.

Crear formularios

La manera más sencilla de crear un formulario es usando el Asistente para formularios. Su uso para crear un formulario se describe en el «Capítulo 8, Primeros pasos con Base», en la *Guía de primeros pasos*. En ese capítulo también se explica cómo puede modificar el formulario después de usar el asistente.

Esta guía describe la creación de un formulario sin usar el asistente y también describe las propiedades de los distintos tipos de controles de un formulario.

Un formulario sencillo

Comenzaremos usando la tarea *Crear formulario en modo de diseño* en el área *Formularios* de la ventana principal de Base.

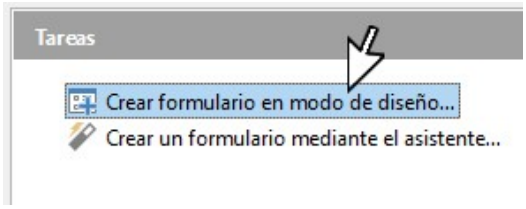


Figura 71

Al hacer clic en *Crear formulario en modo de diseño* (Figura 71) se invoca al *Editor de formularios* y el formulario se mostrará en la ventana *Vista de diseño* (Figura 72).

La barra de herramientas *Controles de formulario* está ubicada en el lado izquierdo. La barra de herramientas *Diseño de formulario* está ubicada en la parte inferior (puede ver sus botones en la Figura 75). Si estas barras de herramientas no apareciesen automáticamente, utilice **Ver > Barras de herramientas** del menú principal para mostrarlas.

Sin estas barras de herramientas, no es posible crear un formulario.

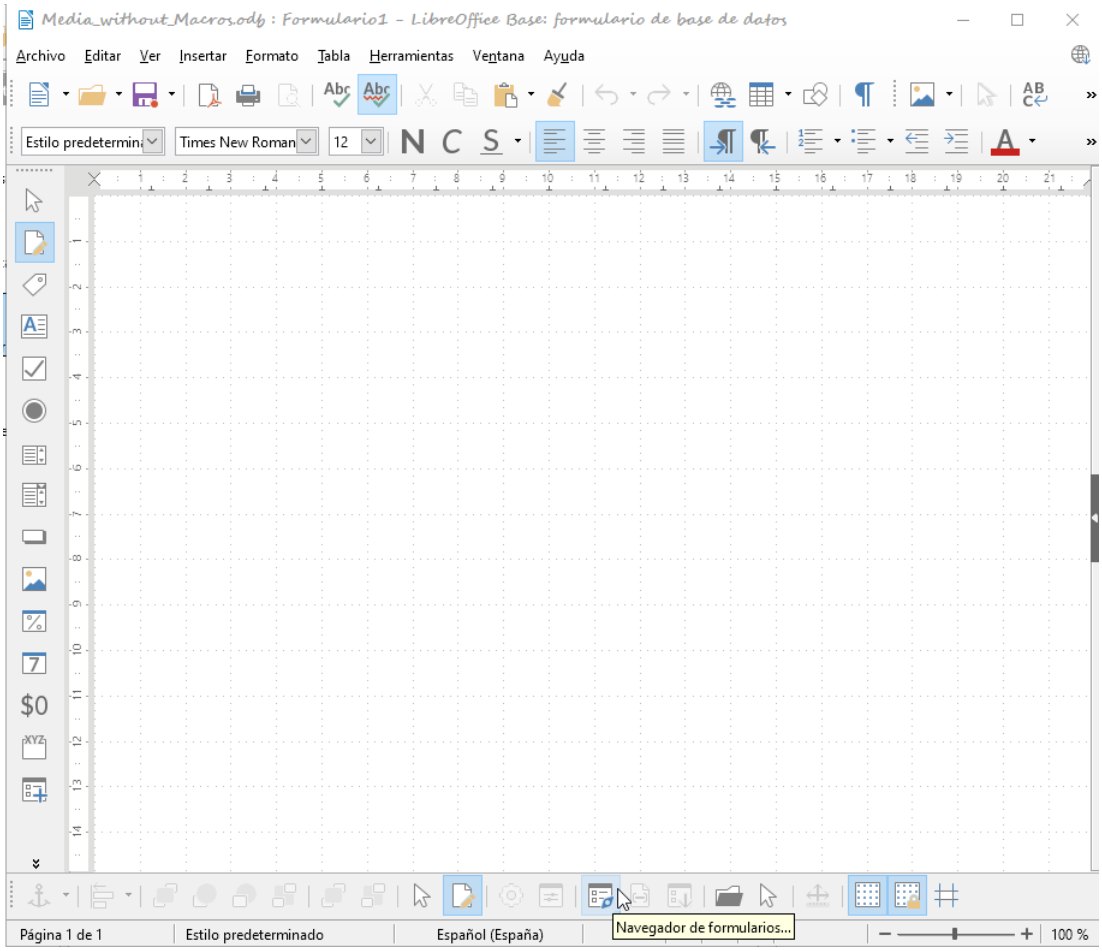


Figura 72: Vista de diseño (señalado el Navegador de formularios)

El área en blanco muestra una cuadrícula de puntos. Esta cuadrícula le ayuda a colocar los controles con precisión. Los símbolos en el extremo derecho de la barra de herramientas de *Diseño de formulario* muestran que la cuadrícula está visible y activa.

Barras de herramientas para el diseño de formularios

Un formulario se crea en la página vacía. Esto se puede hacer de dos maneras:

- Usando el navegador de formularios para diseñar un formulario
- Diseñando los controles del formulario y configurándolos mediante el menú contextual.

Diseñar un formulario con el navegador de formularios

Para poder acceder al *Navegador de formularios*, es preciso que esté dentro de un formulario, en vista diseño. Haga clic en el botón correspondiente (que se señala en la Figura 72).

Aparecerá una ventana, (Figura 73) mostrando solo una carpeta, etiquetada como *Formularios*. Este es el nivel más alto del área que se está editando. Aquí se pueden ir acomodando varios formularios.

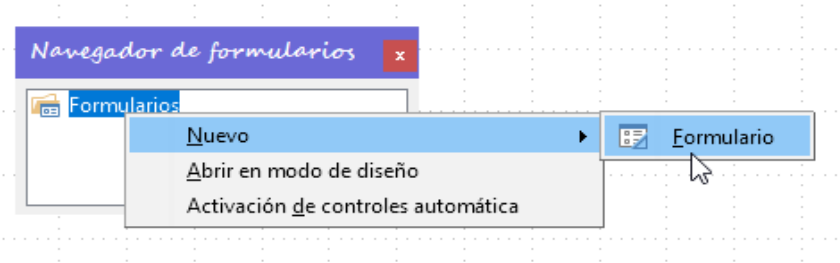


Figura 73: Crear un nuevo formulario mediante el navegador de formularios.

Haga clic con el botón derecho del ratón sobre *Formularios* en el *Navegador de formularios* (Figura 73) para abrir un menú contextual y seleccione **Nuevo > Formulario**. Las otras dos opciones que figuran en el menú contextual, también aparecen en la barra de botones *Diseño de formulario* (Figura 75) y se discutirán más tarde.

Nota

Para que un formulario se inicie con el cursor en el primer campo, use la opción *Activación de controles automática*. Lo que cuenta como primer elemento está determinado por la secuencia de activación del formulario y el orden de activación de los controles se irá asignando progresivamente a los controles según su orden de inclusión en el formulario.

Desafortunadamente, por el momento hay un error (error 87290) en esta función. Si un formulario contiene un control de tabla, el foco se establece automáticamente en el primer campo de este control. Curiosamente este error desaparece si, después de elegir el enfoque de control automático, cambia el idioma de la interfaz de usuario.

El formulario lleva como nombre predeterminado *Formulario*. Puede cambiarlo de inmediato o más tarde. No tiene importancia a menos que necesite acceder a alguna parte del formulario utilizando macros. Lo único que debe asegurarse es que dos elementos con el mismo nombre **no** se encuentren en el mismo nivel en el árbol de carpetas. El menú contextual del formulario (Figura 74) proporciona la manera de cambiar las propiedades de formulario.

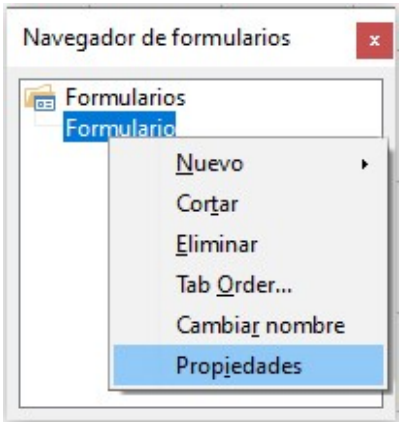


Figura 74

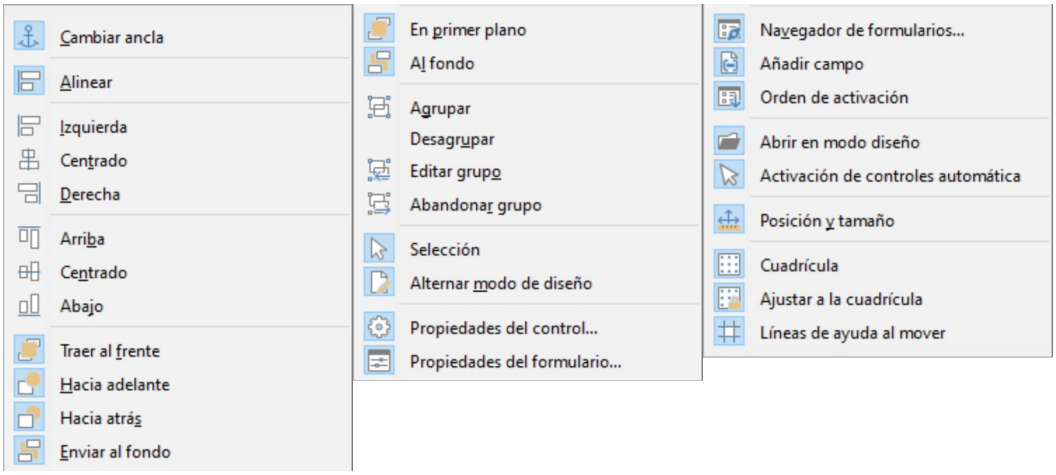


Figura 75: Botones de la barra de herramientas Diseño de formulario

Crear un formulario usando un campo de formulario

La barra de herramientas *Controles de formulario* (Figura 76) contiene varios tipos de controles. Los primeros cuatro elementos también incluidos en de la barra de herramientas *Diseño de formulario* (Figura 75); son para selección y acceso a las propiedades del formulario y controles. Después encontrará los tipos de control de formulario más comunes para el diseño de un formulario.

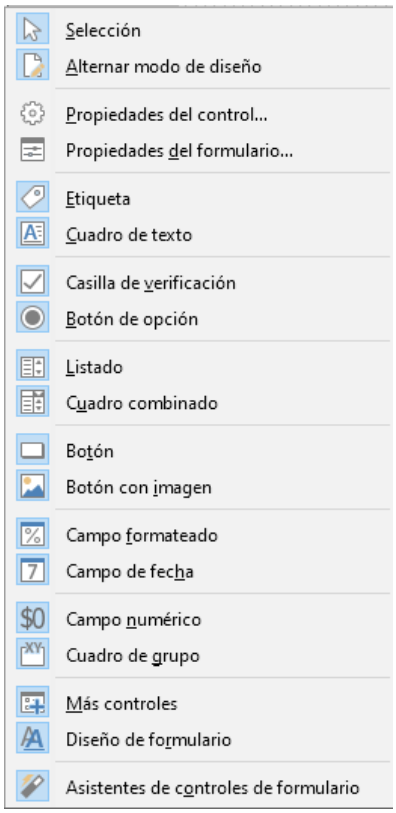


Figura 76: Botones de la barra de herramientas Controles de formulario

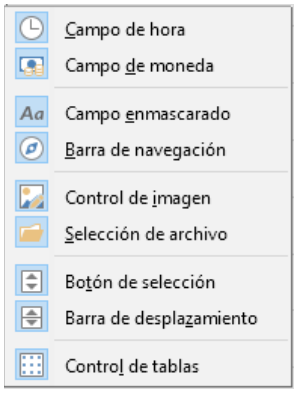


Figura 77: Botones de la barra de herramientas Más campos

Cuando incluye un control de formulario, en una página vacía, está creando automáticamente un formulario. Por ejemplo, suponga que elige un *Cuadro de texto*: el cursor cambia de forma y se puede dibujar un rectángulo en la superficie blanca del formulario. En la superficie punteada del formulario, aparecerá un campo de texto (Figura 78). Se habrá creado un formulario que contiene ese control y el programa le habrá asignado el nombre predeterminado (*Formulario)*.

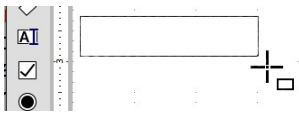


Figura 78

Puede acceder a las propiedades del formulario haciendo clic derecho en el control pulsando en el icono *Formulario* del menú contextual que aparecerá (figura 79).

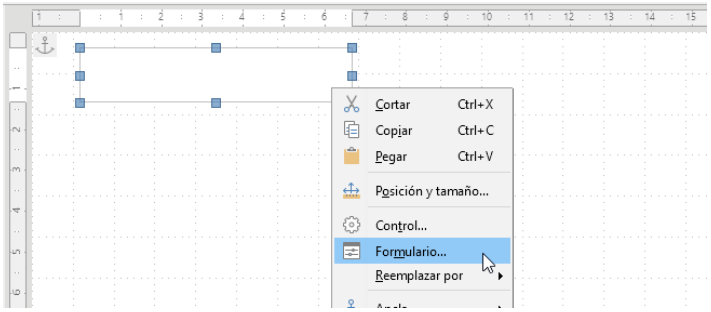


Figura 79: Menú contextual de un control

Con esta acción podrá establecer las propiedades del formulario que se acaba de crear.

Formularios externos

Además de los formularios que se crean dentro de Base, también existe la posibilidad de crear formularios en Writer o Calc. Estos se describen en el «Capítulo 7, Vincular bases de datos».

Propiedades de formulario

Cuando se invocan las propiedades del formulario utilizando el menú contextual en el *Navegador de formularios* o en el de un control de formulario, aparece una ventana *Propiedades del formulario*. Tiene tres pestañas: *General*, *Datos* y *Sucesos.*

Pestaña General

Aquí se puede cambiar el nombre del formulario. Además, encontrará otras posibilidades de diseño para formularios (y aunque se describen no tienen mucha relevancia dentro de Base pues son más indicadas para el diseño de formularios web).

Figura 80

URL: Dirección web destino de los datos.

Marco: Sección del sitio web de destino que se especificará cuando sea necesario.

Codificación al enviar: Además de la codificación habitual de caracteres para la transmisión a la URL, puede especificar aquí la codificación de texto o la codificación multiparte para la transferencia de datos.

Tipo de envío: GET (visible a través de la URL adjunta al nombre del archivo; puede ver esto a menudo en la web si usa un motor de búsqueda) o POST (no visible; adecuado para grandes volúmenes de datos).

Pestaña Datos

Esta es la pestaña más importante a la hora de crear formularios internos en Base. Aquí se establecen las siguientes propiedades del formulario:

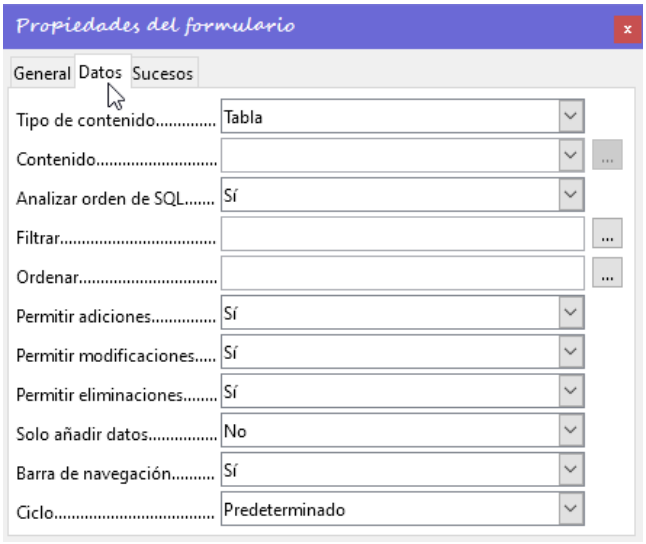


Figura 81

Tipo de contenido: Puede elegir entre *Tabla*, *Consulta* y *Orden de SQL*. Si bien la tabla es lo más utilizado (entrada de datos en un formulario), a veces esto no es lo que se necesita. Se puede usar una consulta a una base de datos (vea el «Capítulo 5, Consultas») o utilizar una entrada directa de un comando SQL. Con lo que se puede hacer una consulta que no es visible en el contenedor de consultas de Base, pero que tiene, en principio, la misma estructura.

Contenido: Según haya elegido *Tabla* o *Consulta* anteriormente, se enumeran las tablas y consultas disponibles. Si se va a crear una orden SQL, se puede invocar el *Editor de consultas* utilizando los puntos suspensivos (…) a la derecha del campo *Contenido*.

Analizar orden de SQL: Si no se debe permitir el análisis de los mandatos SQL (por ejemplo, está utilizando un código que la interfaz no puede mostrar correctamente), aquí debe elegir «No». Sin embargo, esto evitará que el formulario acceda a los datos subyacentes utilizando un filtro o una clasificación. Lo más normal es que se establezca como «Sí».

Filtrar: Aquí se puede configurar un filtro. Para obtener ayuda, haga clic en el botón (…) a la derecha del campo. Vea también el «Capítulo 3, Tablas».

Ordenar: Puede configurar el orden de sus datos. Para obtener ayuda, haga clic en el botón (…) a la derecha del campo. Vea también el «Capítulo 3, Tablas».

Enlazar campos maestros y *Enlazar campos subordinados*. Solo en los **subformularios** aparecen estos dos campos a continuación de *Ordenar*: Estos campos se utilizan para unir determinados campos del formulario principal con otros del subformulario dependiente con objeto de transferir datos entre los mismos.

Permitir adiciones: Para permitir la entrada de nuevos datos.

Permitir modificaciones: Para permitir la edición de los datos.

Permitir eliminaciones: Para permitir el borrado de datos.

— El valor predeterminado es «Sí» en estos tres casos.

Solo añadir datos: con esta opción activada siempre se mostrará un formulario vacío. No habrá acceso a los datos existente: no se podrán editar ni ver.

Barra de navegación: Se puede activar o desactivar la visualización de la barra de navegación en la parte inferior de la pantalla. También existe la posibilidad de mostrar siempre la barra de navegación para el formulario principal, de modo que esta solo afecte al formulario principal (aunque se esté en un subformulario).

Esta configuración para la barra de navegación no es relevante para la barra de herramientas de navegación interna que se puede agregar como control de formulario si es necesario.

Ciclo: El modo en que se comportará al saltar desde el último campo de un registro. La opción *Predeterminado* es que después de introducir datos en el último campo de un registro, la tecla *Tab* le lleve al primer campo del siguiente registro, es decir, se cree un nuevo registro. Para las bases de datos, esto tiene el mismo efecto que *Todos los registros*. Por el contrario, si elige *Registro activo*, el cursor se moverá solo dentro del registro; cuando llegue al último campo, volverá al primer campo en ese registro. *Página actual* se refiere particularmente a los formularios HTML. El cursor salta desde el final de un formulario al siguiente formulario de esa página.

Pestaña Sucesos

Todas las opciones de *Sucesos* pueden hacer una llamada a una macro. Un clic en el botón de la derecha (…) de cada suceso permite vincular una macro a ese suceso.

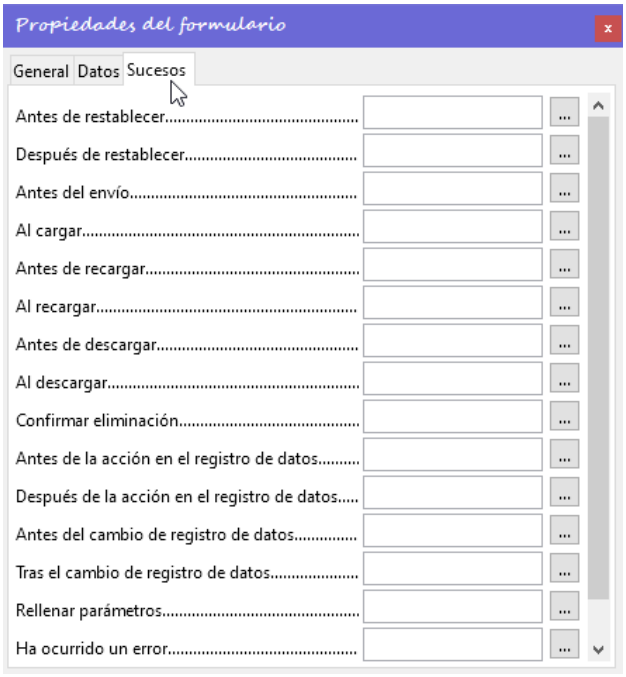


Figura 82

Antes de restablecer: El formulario se vacía de todas las entradas nuevas que aún no se han guardado.

Antes del envío: Antes de enviar los datos del formulario. Solo útil en los formularios web.

Al cargar: solo al abrir el formulario, **no** cuando se carga un nuevo registro en el formulario.

Al recargar: Cuando se actualiza el contenido del formulario, por ejemplo, mediante el uso de un botón en la barra de navegación.

Al descargar: esta opción parece no funcionar. Aunque debería obedecer al cierre del formulario.

Acciones con un registro: esto incluye, por ejemplo, el almacenamiento mediante un botón. En las pruebas, esta acción se duplica regularmente; Las macros se ejecutan dos veces seguidas. Esto se debe a que aquí se llevan a cabo diferentes funciones (implementaciones). Estas funciones son: `org.openoffice.comp.svx.FormController` y `com.sun.star.comp.forms.OdatabaseForm`. Si dentro de la macro que usa `oForm.ImplementationName`, se consulta el nombre correspondiente y la macro se puede limitar a una sola ejecución.

Antes de la acción en el registro de datos / *Después de la acción en el registro de datos:* La apertura de un formulario cuenta como un cambio de registro. Cada vez que un registro cambia a otro dentro del formulario. Esta acción también se sucede en dos ocasiones, con lo que las macros se ejecutan dos veces seguidas. Aquí también debemos distinguir entre las causas de este resultado.

Rellenar parámetros: Esta macro se ejecutará si se tiene que hacer una llamada a una consulta de parámetros desde un subformulario, pero por alguna razón el parámetro no se transmite correctamente desde el formulario principal. Si este suceso no se detecta, una consulta de parámetros seguirá la carga del formulario.

Ha ocurrido un error: Este suceso no se puede utilizar.

Propiedades de los controles

Una vez creado un formulario, se debe completar con controles visibles. Algunos controles permiten mostrar el contenido de la base de datos o ingresar datos en la base de datos. Otros controles se utilizan exclusivamente para la navegación, la búsqueda y la ejecución de comandos (interacción). Determinados controles sirven para hacer algún retoque gráfico al formulario.

Entrada de datos y visualización de datos	
Control	***Uso***
Cuadro de texto	Entrada de texto.
Campo numérico	Entrada de números.
Campo de fecha	Entrada de fechas.
Campo de hora	Entrada de horas.
Campo moneda	Entrada de números con un formato determinado tipo moneda.
Campo formateado	Entrada y/o visualización en formato especial, ej. unidades de medida.
Listado	Elección entre varias posibilidades se transfiere a la base de datos el valor seleccionado.
Cuadro combinado	Similar al anterior, Se transfiere el valor seleccionado o un valor introducido manualmente.
Casilla verificación	Campo Sí/No.
Botón de opción	(Botón radial), permite elegir entre un número reducido de opciones.
Control de imagen	Visualiza imágenes de una base de datos o incluye imágenes en la base de datos mediante la selección de una ruta.
Campo enmascarado	Entrada con una máscara preestablecida; limita las posibilidades de entrada a formatos con caracteres específicos.
Control de tablas	Módulo de entrada universal. Puede mostrar una tabla completa. En este control se integran muchos de los controles anteriores.

Diseño	
Control	***Uso***
Etiqueta	Descripción de otros controles o texto para secciones del formulario.
Cuadro de grupo	Marco alrededor de los controles. Ej.: grupo de botones de opción.

Interacción	
Control	***Uso***
Botón	Botón con un texto.
Botón con imagen	Igual que el botón, pero mostrando una imagen.
Barra de navegación	Barra de herramientas muy similar a la del borde inferior de la pantalla.
Selección de archivo	Seleccionar archivos, por ejemplo para cargar archivos en un formulario HTML; no se describe con más detalle.
Botón de selección	Solo se puede usar con una macro; no se describe con más detalle.
Barra de desplazamiento	Solo se puede usar con una macro; no se describe con más detalle.
Control oculto	Se puede almacenar un valor usando una macro y luego recuperarlo.

Propiedades comunes para los controles

Al igual que con los formularios, las propiedades de control se agrupan en tres categorías: *General*, *Datos* y *Sucesos.*

- **General** Comprende todo lo que es visible para el usuario.
- **Datos** Categoría que especifica el enlace con una base de datos.
- **Sucesos** Categoría que controla las acciones que pueden vincularse a una macro. En una base de datos sin macros, esta categoría no tiene transcendencia.

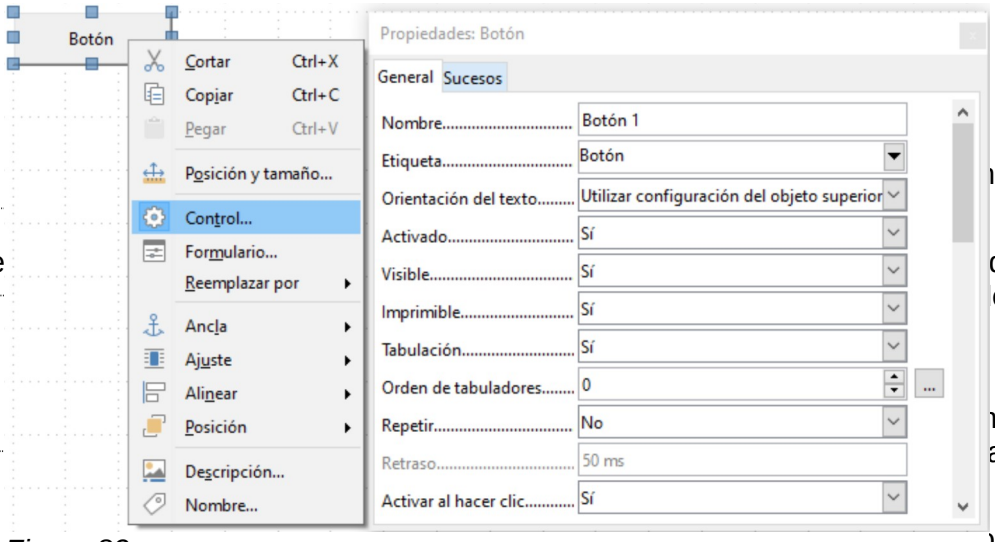


Figura 83

Pestaña General

Nombre

ulario, se

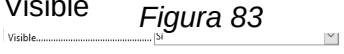
Campo de

queta (se e ese

Activado

troles alor, el

Visible

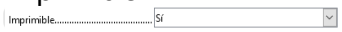


os invisibles se pueden usar como almacenamiento intermedio, por ejemplo al crear controles combinados con macros. Vea el «Capítulo 9, Macros».

Solo Lectura

Permite o evita cualquier modificación del valor. Es útil, por ejemplo: para una clave primaria generada automáticamente.

Imprimible



Si el control debe aparecer en la impresión de un formulario

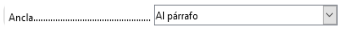
Tabulación

Permite o evita que el control se active con la tecla *Tab*.
Un control de solo lectura no lo necesita. Se puede omitir.

Orden de tabuladores

Si el control tiene una tabulación se especifica la secuencia de activación dentro del formulario.

Ancla



Anclaje de gráficos dentro de un cuadro de texto.

Posición X

Posición desde el borde izquierdo en relación con el formulario.

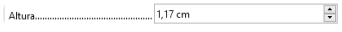
Posición Y

Posición desde el borde superior en relación con el formulario.

Anchura

Anchura del control.

Altura

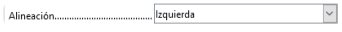


Altura del control.

Tipo de letra

Configuración del tipo de fuente, tamaño y efectos gráficos.

Alineación

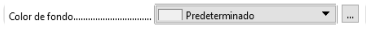


Alineación del texto en el control.

Alineación vertical

Alineación vertical del texto en el control.

Color de fondo



Color de fondo del control.

Marco

Tipo de marco que se aplica al control (Sin marco | 3D | Plano).

Color del borde

Color del borde del marco (solo posible con el marco *Plano*)

Ocultar selección

Especifica si una selección de texto en un control permanece seleccionada cuando el control deja de estar activo.

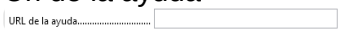
Información adicional

Especifica una información adicional para su uso con macros. Vea «Capítulo 9, Macros».

Texto de ayuda

Muestra una información emergente cuando el ratón se desplaza sobre el control.

Url de la ayuda



Apunta a una página de ayuda alojada en la red. Se puede invocar usando F1 cuando el foco está en el control.

Además, los controles numéricos, de fecha, etc. tienen las siguientes propiedades:

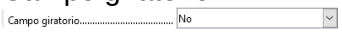
Control de formato

Exige que se ingresen números y puntos decimales cuando está en «sí».

Desplazam. rueda ratón

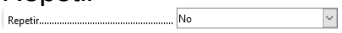
Posibilita o evita que la rueda del ratón cambie los valores establecidos cuando el puntero del ratón está sobre el control.

Campo giratorio



Incorpora las flechas arriba y abajo en la parte derecha del control para que se pueda cambiar el valor con un clic del ratón o con el teclado cuando el control tiene el foco.

Repetir



Si una flecha se pulsa y se mantiene pulsada determina si el valor del control debe cambiar progresivamente.

Retraso

Determina el retardo mínimo que activa el cambio del valor (permitido en la opción anterior).

Pestaña Datos

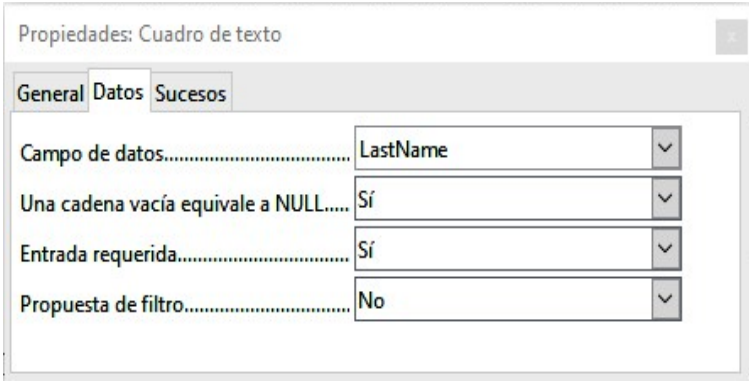


Figura 84

Campo de datos: Crea el enlace con la tabla en la que se basa el formulario.

Una cadena vacía equivale a NULL: Si una cadena vacía debe tratarse como «NULL» o si el contenido simplemente se elimina.

Entrada requerida: Esta condición debe coincidir con la de la tabla. La Interfaz exigirá que se rellene el dato.

Propuesta de filtro: Cuando los datos se van a filtrar, el contenido de este campo se almacena temporalmente como una sugerencia. Precaución: si la base está concebida para almacenar muchos datos, esta opción puede consumir mucha memoria.

Pestaña Sucesos

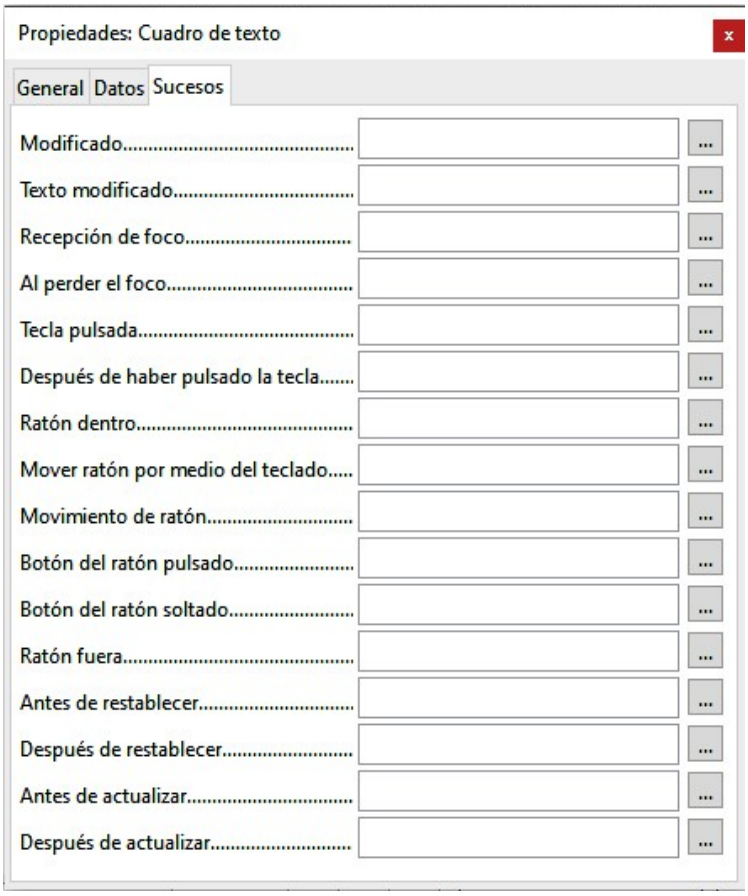


Figura 85

Modificado: Este suceso tiene lugar cuando se modifica el valor de un control y luego pierde el foco. El suceso se pierde si se cambia directamente a otro registro. En estas circunstancias, un cambio se guarda sin ser detectado.
[`com.sun.star.lang.EventObject`]

Texto modificado: Se refiere al contenido, que puede ser texto, numérico etc. Se produce después de que se ingresa un carácter adicional. [`com.sun.star.awt.TextEvent`]

Recepción de foco: Cuando el control se activa al situarse sobre él.

Precaución

Bajo ninguna circunstancia la macro tiene que crear un cuadro de diálogo de tipo mensaje en la pantalla; Al hacer clic en el mensaje, el campo de formulario pierde el foco y luego lo recupera, activando la macro nuevamente. Se crea un bucle que solo se puede romper usando el teclado.

Al perder el foco: Cuando el cursor sale del control. Esto puede producir la misma repetición no deseada indicada en la nota de advertencia anterior.

Tecla pulsada: Se refiere al teclado. Por ejemplo, se presiona una tecla cuando se recorre el formulario con la tecla *Tab*. Esto hace que un campo reciba el foco.

Después de haber pulsado la tecla: tras liberar la tecla pulsada.

— Estos sucesos se gestionan usando el *KeyCode* (código de tecla) o *KeyChar* (carácter) de la tecla pulsada o liberada (letra, número, clave especial).
[`com.sun.star.awt.KeyEvent`]

Ratón dentro: Este suceso solo tienen lugar si el ratón está o ya estaba dentro del control.

Raton fuera: lo mismo que en el caso anterior,

Los sucesos del ratón se gestionan con .[`com.sun.star.awt.MouseEvent`]

Restablecer: El formulario se vacía de todos los datos (al crear un nuevo registro) o se restablece a su estado original (al editar un registro existente). Para un control de formulario, este evento se desencadena solo cuando se utiliza el botón de *Deshacer* en la barra de navegación.[`com.sun.star.lang.EventObject`]. Cuando se carga un formulario por primera vez, los dos eventos *Antes de restablecer* y *Después de restablecer* se desencadenan en sucesión, antes de que el formulario esté disponible para introducir datos.

Actualizar: Si el suceso está vinculado a un control de formulario, la actualización se lleva a cabo cuando se pierde el foco y salta a otro control de formulario, después de alterar el contenido del campo.

Los cambios en el formulario son aceptados y mostrados. Cuando se cierra un formulario, los dos eventos *Antes de actualizar* y *Después de actualizar* se suceden. [`com.sun.star.lang.EventObject`]

Campo de texto

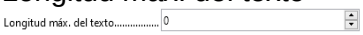
Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Orientación del texto

Permite cambiar la orientación de izquierda a derecha o viceversa (usada en algunos idiomas orientales).

Longitud máx. del texto

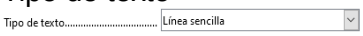


Máximo número de caracteres admitido. Cuando el valor es 0, no se permite la entrada. Normalmente, aquí se usa la longitud del campo en la base de datos sobre el que actúa el control.

Texto predeterminado

Texto predeterminado que debería introducirse en el control. Este texto será eliminado cuando se introduzca un texto distinto.

Tipo de texto

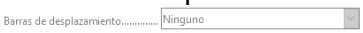


Linea sencilla | *Multilínea* | *Multilínea con formato* (los dos últimos funcionan de manera diferente al tabular y, además, un control de este tipo no puede vincularse a una base de datos). La alineación vertical no está activa para los controles multilínea.

Las líneas acaban con…

Esto afecta principalmente las terminaciones de línea. Windows usa dos caracteres de control (*CR y LF*) en cambio Unix solo *LF*.

Barras de desplazamiento



Solo para controles multilínea: *Ninguno* |*Horizontal* | *Vertical* | *Ambos.*

Carácter de contraseña

Cambia los caracteres por el definido para ocultar una contraseña. Activo solo para controles de *Línea sencilla*.

Pestaña Datos

Nada que destacar.

Pestaña Sucesos

Nada que destacar.

Campo numérico

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Valor min.

Valor mínimo para el control. Debe estar de acuerdo con el valor mínimo definido en la tabla.

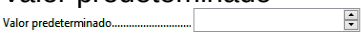
Valor máx

Valor máximo.

Valor de inc./decremento

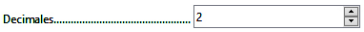
Incremento de desplazamiento al usar la rueda del ratón o con las flechas arriba y abajo.

Valor predeteminado



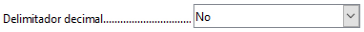
Valor predeterminado que debería introducirse en el control. Este valor será eliminado al introducir un dato distinto.

Decimales



Número de decimales, se establece en 0 para enteros.

Delimitador decimal



Activa o desactiva el separador de millares.

Pestaña Datos

No se verifica si un campo puede ser nulo. (Si no hay entrada, el campo será «NULL» **no** «0»). No se ofrece ninguna propuesta de filtro.

Pestaña Sucesos

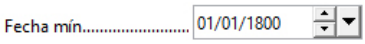
El suceso *Modificado* no está disponible. Los cambios deben manejarse con el suceso *Texto modificado* (la palabra texto no se debe tomar literalmente).

Campo de fecha

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

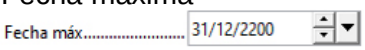
Pestaña General

Fecha mínima



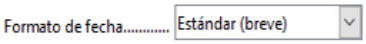
Valor mínimo para el control, también se selecciona mediante un calendario desplegable.

Fecha máxima



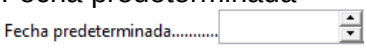
Valor máximo.

Formato de fecha



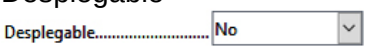
Formato para la fecha.(hágalo coincidir con su formato en la base de datos)

Fecha predeterminada



Puede indicar una fecha predeterminada, pero desafortunadamente no acepta la fórmula =(*Hoy*), fecha del día en que se abre el formulario.

Desplegable



Activa o desactiva un calendario mensual para seleccionar fechas.

Pestaña Datos

No se verifica si un campo puede ser nulo. (Si no hay entrada, el campo será «NULL» no «0»). No se ofrece ninguna propuesta de filtro.

Pestaña Sucesos

El suceso *Modificado* no está disponible. Los cambios deben manejarse con el suceso *Texto modificado* (la palabra texto no se debe tomar literalmente).

Campo de hora

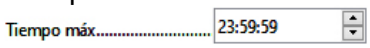
Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Tiempo mínimo

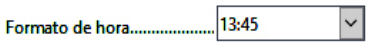
Valor mínimo para el campo, establecido de forma predeterminada en 0. (sin limitación)

Tiempo máximo



Valor máximo, establecido de manera predeterminada en 1 segundo antes de las 24:00. (sin limitación)

Formato de hora



Formato para la hora (debe coincidir con los datos de la base)

Hora predeterminada

Puede indicar una hora predeterminada, pero, como en el caso de la fecha, no acepta la hora real al escribir en el registro.

Pestaña Datos

No se verifica si un campo puede ser nulo. (Si no hay entrada, el campo será «NULL» no «0»). No se ofrece ninguna propuesta de filtro.

Pestaña Sucesos

El suceso *Modificado* no está disponible. Los cambios deben manejarse con el suceso *Texto modificado* (la palabra texto no se debe tomar literalmente).

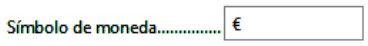
Campo de moneda

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

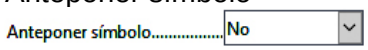
El valor *Mínimo*, *máximo*, *incremento*, *valor predeterminado*, *lugares decimales* y *separador de miles* se corresponden con las propiedades generales enumeradas en la página 139. Además de estas, están disponibles:

Símbolo de moneda



El símbolo se muestra, pero no se almacena en los datos.

Anteponer símbolo



Si el símbolo de moneda debe colocarse antes de la cantidad.

Pestaña Datos

No se verifica si un campo puede ser nulo. (Si no hay entrada, el campo será «NULL», no «0»). No se ofrece ninguna propuesta de filtro.

Pestaña Sucesos

El suceso *Modificado* no está disponible. Los cambios deben manejarse con el suceso *Texto modificado* (la palabra texto no se debe tomar literalmente).

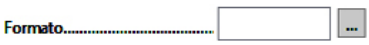
Campo formateado

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Los valores *mínimo*, *máximo y valor predeterminado*, dependen del formato. El control de formato es un campo flexible que hace innecesario el uso de la mayoría de los controles de moneda y numéricos. A diferencia de un campo de moneda simple, un campo enmascarado puede mostrar valores negativos en rojo.

Formato



Formato aplicado. El botón a la derecha (...) ofrece una gran variedad de formatos numéricos.

Se pueden elegir varios formatos numéricos (Fecha, Hora, Moneda o formato numérico normal), También existe la posibilidad de usar campos formateados con una unidad de

medida como kg (vea la figura 86). Consulte también la Ayuda general sobre «códigos de formato numérico».

Un campo formateado permite crear y escribir datos de «marca de tiempo» en tablas usando un único campo. El Asistente de formularios utiliza una combinación de campos de fecha y hora para este propósito.

Si desea ingresar datos en la forma de [`minutos:segundos:centésimas`] en un campo de «marca de tiempo», deberá usar macros.

Pestaña Datos

Nada especial que señalar.

Pestaña Sucesos

El suceso *Modificado* no está disponible. Los cambios deben manejarse con el suceso *Texto modificado* (la palabra texto no se debe tomar literalmente).

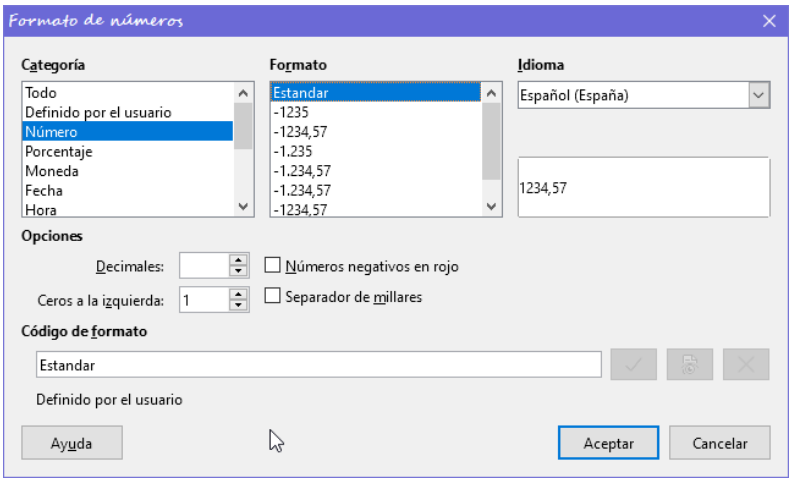


Figura 86: Formatos para Campos formateados

Listado (cuadro de lista)

Cuando se crea un cuadro de lista, el *Asistente de cuadros de lista* aparece de manera predeterminada. Este asistente se puede desactivar si así se desea, utilizando el botón de *Asistentes de controles de formulario*.

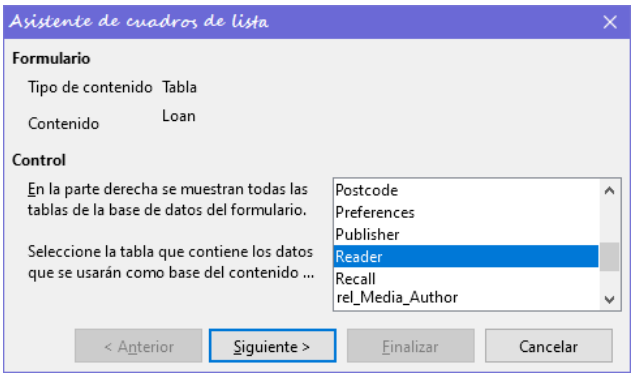


Figura 87

Asistente de cuadros de lista.

El formulario ya estaba definido. Está vinculado a una tabla llamada *Loan*. Un cuadro de lista muestra al usuario diferentes datos de los que realmente se transmiten a la tabla. Estos datos generalmente provienen de otra tabla en la base de datos y no necesariamente de la tabla a la que está vinculado el formulario.

Se supone que la tabla de préstamos (*Loan*) muestra el artículo prestado y el lector que ha tomado prestado ese artículo. Sin embargo, esta tabla no almacena el nombre del lector,

sino la *clave principal* correspondiente a ese lector, que toma de la tabla *Reader*. Por lo tanto, es la tabla *Reader* la que constituye la base del cuadro de lista.

El campo *LastName* de la tabla *Reader* debe estar visible en el cuadro de lista. (Campo que se visualizará)

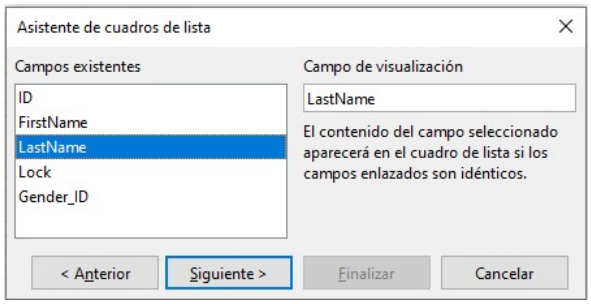


Figura 88

El formulario toma el campo *Reader_ID* de la tabla *Loan* como campo de visualización. Esta tabla se describe como la tabla de valores. La clave *ID* (clave principal) de la tabla *Reader* tiene que estar vinculada al campo de la tabla de listas.

La tabla *Reader* será la que se utilice para obtener el *Listado* y se describe como la tabla de listas.

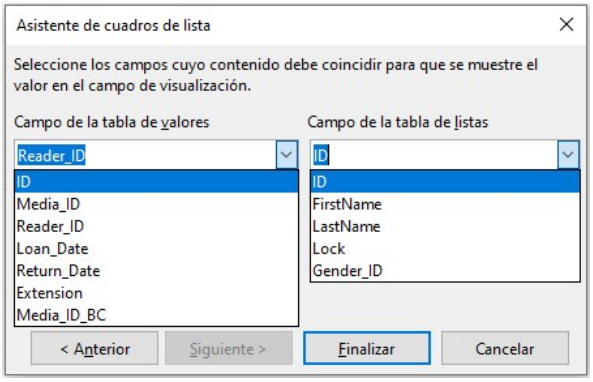


Figura 89

Con esto el cuadro de lista ya se habrá creado con acceso a datos y la configuración predeterminada y es completamente funcional.

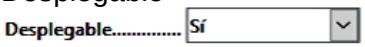
Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Entradas de la lista

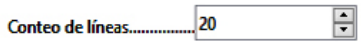
Las entradas de la lista ya se han establecido con el asistente. Aquí puede agregar más entradas aunque no estén en ninguna tabla en la base de datos. Las entradas de la lista son las entradas visibles, no necesariamente las que el formulario vaya a transmitir a la tabla.

Desplegable



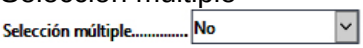
Si el control cuando se configura como desplegable se convierte automáticamente en un campo de varias líneas, en el que se resalta el valor seleccionado. En caso contrario aparecerán flechas de desplazamiento en el lado derecho del cuadro de lista y solo se verá una línea.

Conteo de líneas



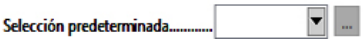
Si el control es desplegable, esta propiedad proporciona el número máximo visible de líneas. Si el contenido se extiende más de este límite, aparecerá una barra de desplazamiento cuando la lista se despliegue.

Selección múltiple



Configura si se puede seleccionar más de un valor. En el ejemplo anterior, esto no es posible, ya que se está usando una clave externa. Por lo general, esta función no se usa para bases de datos, ya que cada campo solo debe contener un valor. Si es necesario, las macros pueden ayudar en la interpretación de múltiples entradas en el campo de lista.

Selección predeterminada



Una selección predeterminada tiene poco sentido en el contexto de un enlace con una tabla de base de datos. Al crear el control con el *Asistente para campos de lista,* esta opción se desactiva. Se podría dar el caso de que el registro correspondiente a la selección predeterminada en la tabla de ejemplo *Readers* ya no esté presente.

Pestaña Datos

Además de las propiedades de datos habituales, *Campo de datos* y *Entrada requerida*, hay otras propiedades que afectan significativamente al enlace entre los datos mostrados y los datos que se ingresarán en la tabla a la que se refiere el control.

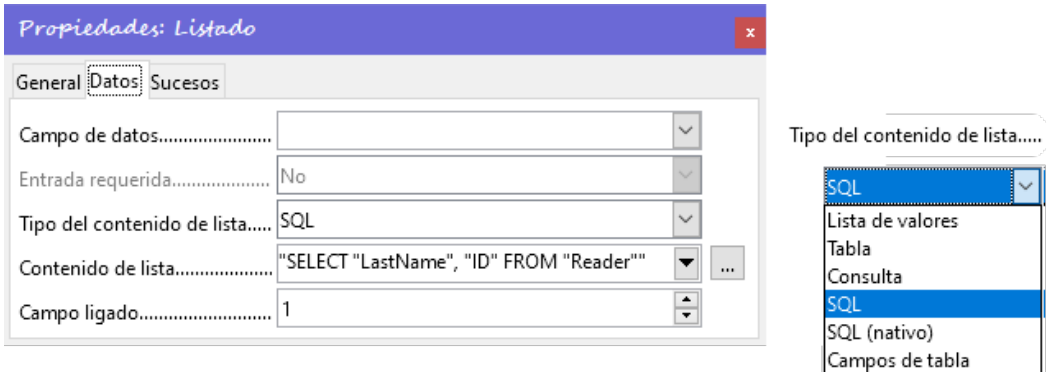


Figura 90

Tipo del contenido de lista: Lista de valores |Tabla | Consulta | SQL | SQL [nativo] | Campos de tabla

Lista de valores: Si las entradas de la lista se han creado en *General*, se muestran aquí los valores correspondientes que se emplearán. El *contenido de la lista* se carga con elementos individuales separados por *Mayús+Intro*. El campo *Contenido de la lista* los muestra como "Valor1"; "Valor2"; "Valor3"… La propiedad del *Campo ligado* está inactiva.

Tabla: Se puede seleccionar una de las tablas de la base de datos, sin embargo, esto rara vez es posible. Requiere que el contenido de la tabla esté estructurado de manera que el primer campo de la tabla contenga los valores que se mostrarán en el *Contenido de lista* y uno de los siguientes campos contenga la clave principal que utiliza la tabla consultada por el formulario como una clave externa.

La posición de este campo dentro de la tabla se especifica en *Campo ligado*, donde la numeración comienza por 0 para el primer campo de la tabla de la base de datos. En el contenido mostrado, *0* hace referencia a *LastName*, mientras que el *1* se refiere al campo *ID*.

Consulta: Se crea primero una consulta y se almacena. La creación de estas consultas se describe en el «Capítulo 5, Consultas». Con la consulta, es posible mover el campo *ID*

desde la primera posición de la tabla subyacente a la segunda posición, aquí representada en la Figura 91 por el campo ligado 1.

SQL: Si se usa el Asistente para cuadros de lista, este campo se rellenará automáticamente. La consulta construida por el asistente se ve así:

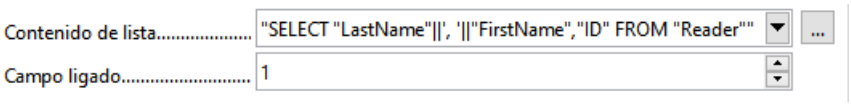


Figura 91

La consulta aplicada es la más simplificada. El campo *LastName* aparece en la posición 0, el campo *ID* en la posición 1. Ambos se leen de la tabla *Reader*. Como el campo enlazado es el Campo 1, esta fórmula SQL funciona.

Se debe agregar `ORDER BY "LastName" ASC` para evitar desplazarse demasiado por la lista para encontrar a alguien. Un problema adicional podría ser que *LastName* (apellido) fuera el mismo para más de un lector. Por lo tanto, *FirstName* (Nombre) debe agregarse en la vista de *Contenido de lista*. Cuando haya lectores con el mismo apellido y el mismo nombre. También se debe mostrar la *ID* de la clave principal. Consulte el «Capítulo 5, Consultas», para información sobre este proceso.

SQL [Nativo]: El código SQL se ingresa directamente, sin utilizar el asistente. Base no evalúa la consulta. Esto es adecuado cuando la consulta contiene funciones que quizás la interfaz de Base no entienda. En este caso, la consulta no se verifica en busca de errores. Puede encontrar más información sobre el *Modo SQL directo* en el «Capítulo 5, Consultas».

Campos de tabla: Se enumeran los nombres de campo de una tabla, no su contenido. Para la tabla *Reader*, el contenido de la Lista sería *ID*, *Firstname*, *Lastname*, *Lock*, *Gender_ID*.

Nota

Si desea un campo de hora que pueda manejar el tiempo en milisegundos, necesitará un campo de «marca de tiempo», como se describe en la sección «Campo de hora». La representación de milisegundos es incompatible con la forma en que los caracteres se ensamblan en los cuadros de lista. Para evitar esto, debe convertir la marca de tiempo en texto:

```
SELECT REPLACE( LEFT ( RIGHT ( CONVERT( "Required_service(??)"."Time",
VARCHAR),15),8)'.',',') AS "Listcontent", "ID" FROM "Required_service"
```

Lo que se mostrará como: [`minutos:segundos:centésimas`].

Campo ligado: Los campos de lista muestran un contenido que no es necesariamente idéntico al que se almacenará en el formulario. Por lo general, se muestra un nombre o algo similar y la clave principal correspondiente se convierte en el valor almacenado de este campo. Atendiendo a una consulta SQL:

```
SELECT "Name", "ID" FROM "Table" ORDER BY "Name"
```

El campo *ID* se almacena en la tabla subyacente como una clave externa. El recuento de campos en las bases de datos comienza en cero, por lo que el campo vinculado en el formulario tiene el número 1. Si en su lugar selecciona 0, el contenido del campo *Name* se utiliza en el formulario. En ese caso, puede eliminar el campo *ID* de la consulta.

Incluso es posible elegir la posición –1. Entonces **no es el contenido de la consulta** sino la posición de la entrada la que se almacena en la lista. (El primer registro tiene la posición 1).

La consulta anterior produce el siguiente resultado:

Tabla 1: Ejemplo de campos ligados

Name	***ID***
Anneliese	2
Dorothea	3
Sieglinde	1

En los campos de lista, solo se puede seleccionar el nombre. En caso de que el campo de lista esté configurado para el nombre «Dorothea». Se guardarán los siguientes datos en función del campo ligado:

Campo ligado = **1** se guarda «**3**», (el contenido del campo *ID*). Campo ligado = **0** se guarda «**Dorothea**», (el contenido del campo *Name*). Campo ligado = **–1** se guarda «**2**», (porque "Dorothea" ocupa el segundo lugar en la lista).

Nota

La incorporación del los valores «**0**» o «**–1**» en el campo ligado se introdujo con LibreOffice versión 4.1. Anteriormente, solo era posible la selección de valores >=1.

Pestaña Sucesos

Además de los sucesos estándar, están disponibles los siguientes sucesos:

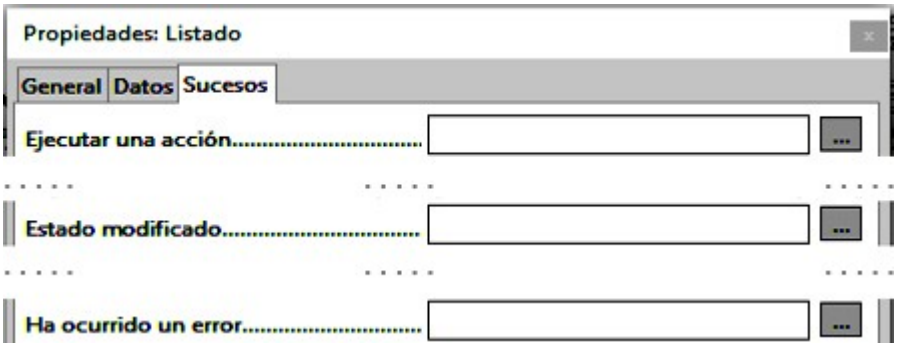


Figura 92

Ejecutar una acción: Si mediante el teclado o el ratón se elige un valor, del cuadro de lista.

Estado modificado: Tras el cambio de contenido mostrado en el cuadro de lista mediante el uso del botón desplegable o por un clic con el ratón dentro del control.

Ha ocurrido un error: Este suceso no se puede utilizar para cuadros de lista.

Cuadro combinado

Cuando se pulsa crear un *Cuadro combinado* aparece el asistente de formulario correspondiente, de la misma manera que con un *cuadro de lista*. Este comportamiento automático se puede desactivar si es necesario utilizando el botón de *Asistentes de controles de formulario* ().

Los cuadros combinados escriben el texto seleccionado directamente en la tabla subyacente al formulario. Así el siguiente ejemplo muestra que la tabla vinculada al formulario y la seleccionada para el control que es la tabla *Reader*.

Asistente

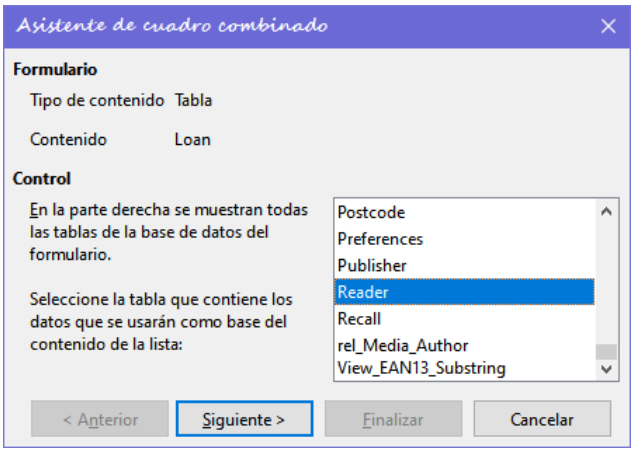


Figura 93

Nuevamente, el formulario está predefinido, esta vez con la tabla de origen de datos *Reader*. Como los datos que se deben mostrar en el cuadro combinado también se almacenarán en esta tabla, la fuente seleccionada para los *datos de la lista* también es la tabla *Reader*.

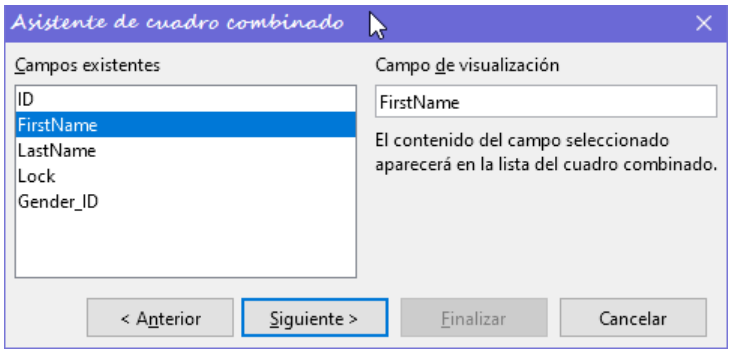


Figura 94

En la tabla *Reader* aparece el campo *FirstName*. (el que se mostrará en el cuadro combinado).

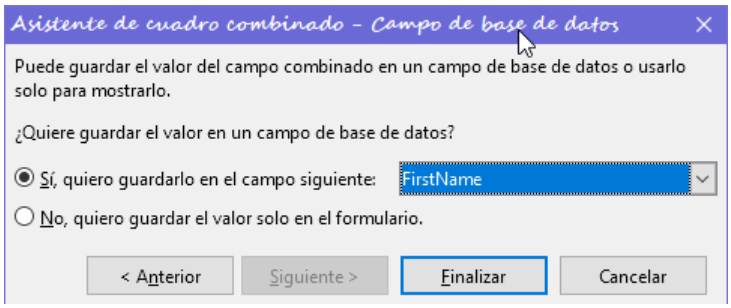


Figura 95

En una base de datos, parece tener poco sentido no almacenar el valor de un cuadro combinado dentro de un campo.

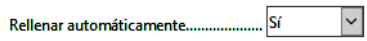
Queremos leer los nombres de la tabla *Reader* y también poder agregar nuevos lectores. Ya que no es necesario crear otro registro para un lector ya existente en la base de datos, el cuadro combinado debe mostrar el nombre del lector, sin necesidad de introducir el nombre de un lector que ya está registrado.

Si queremos agregar un nuevo lector, esto se puede hacer fácilmente pues en un cuadro combinado, se pueden introducir nuevos datos y el cuadro muestra exactamente lo que se va a escribir en la tabla subyacente del formulario.

Además de las propiedades comunes, enumeradas en la página 135 y las descritas para los cuadros de lista, están disponibles las siguientes:

Pestaña General

Rellenar automáticamente



Durante la entrada de nuevos valores, se muestra una lista de valores coincidentes (si los hay) para su posible selección.

Pestaña Datos

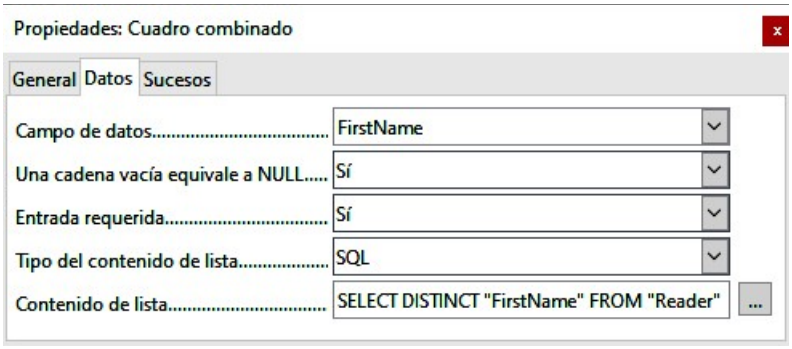


Figura 96

Los campos de datos se ajustan a la configuración predeterminada existente y a la configuración de un cuadro de lista. Sin embargo, el comando SQL muestra una característica especial:

```
SELECT DISTINCT "FirstName" FROM "Reader"
```

Agregar la palabra clave `DISTINCT` asegura que los nombres duplicados se muestren solo una vez. Sin embargo, la creación con el asistente una vez más hace que sea imposible ordenar el contenido.

Pestaña Sucesos

Los eventos son los mismos que los de un cuadro de lista.

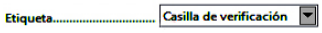
Casilla de verificación

La *Casilla de verificación* aparece como una combinación de una casilla de verificación y una etiqueta para esta casilla.

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

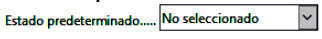
Pestaña General

Etiqueta



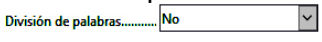
La etiqueta de este control aparece de manera predeterminada a la derecha del cuadro. Además, puede vincularse a un campo de etiqueta separado.

Estado predeterminado



Dependiendo de la elección en el campo *Estado triple*, están disponibles hasta tres posibilidades: No seleccionado | Seleccionado | No definido. No definido corresponde a una entrada `NULL` en la tabla subyacente al formulario.

División de palabras

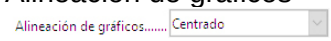


De manera predeterminada el texto de la etiqueta **no** admite división de palabras. La etiqueta se trunca si el campo no es lo suficientemente grande.

Imágen

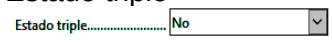
Se puede especificar una imagen en lugar de o además de la etiqueta. Debe tener en cuenta que estas imágenes no se incluyen en el documento *.odb de manera predeterminada. Para imágenes pequeñas, es útil incrustar el gráfico y quitar el vínculo para que se incluyan en el documento.

Alineación de gráficos



Si se ha elegido una imagen, Permite elegir su alineación con la etiqueta (0 = izquierda | 1 = centrado | 2 = derecha).

Estado triple



De manera predeterminada, las casillas de verificación tienen solo dos estados: Seleccionado (Valor: 1) y No seleccionado (Valor: 0). Con *Estado triple*, se agrega una tercera definición de campo vacío (`NULL`).

Pestaña Datos

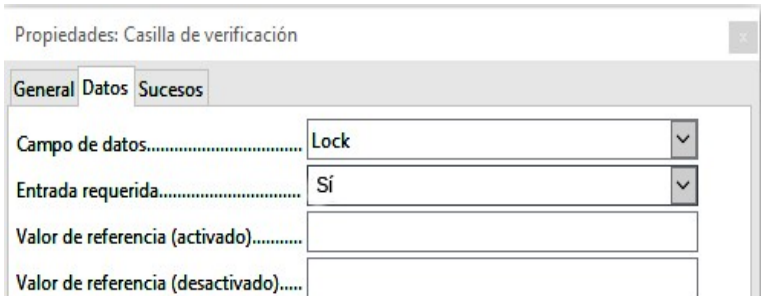


Figura 97

La casilla de verificación puede recibir un *Valor de referencia*. Sin embargo, solo los valores de **1** (para activado) o **0** (para desactivado) se pueden transferir al campo de datos subyacente. Las casillas de verificación actúan como campos booleanos.

Pestaña Sucesos

Los sucesos: *Modificado*, *Texto modificado*, *Antes de actualizar y Después de actualizar* no están disponibles.

Los campos adicionales para una casilla de verificación son: *Ejecutar una acción* (vea «Listado (cuadro de lista)») y *Estado modificado* (que se corresponde con *Modificado*).

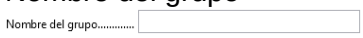
Botón de opciones

El botón de opción es similar a la *Casilla de verificación* descrita anteriormente, excepto por sus propiedades generales y su forma externa (redonda).

Cuando varios botones de opción en el formulario están vinculados al mismo campo de tabla, solo se puede seleccionar una de las opciones.

Pestaña General

Nombre del grupo



El botón de opción está diseñado para usarse en grupos. Solo se puede seleccionar una opción del grupo. Aquí se debe indicar un nombre de grupo que agrupa las opciones.

Pestaña Datos

Igual a la de la *Casilla de verificación*. Aquí, sin embargo, los valores de referencia que se ingresan se transfieren realmente al campo de datos.

Pestaña Sucesos

Igual a la de la *Casilla de verificación*.

Control de imagen

Un control gráfico (imagen) gestiona la entrada y visualización de imágenes para la base de datos. El campo de datos subyacente debe ser un campo binario para almacenar la imagen directamente. También puede ser un campo de texto que almacena la ruta relativa a la imagen. En ese caso, debe tener cuidado de que la ruta a las imágenes siga siendo válida si se copia la base de datos.

Precaución

En cualquier caso, las imágenes deben reducirse de tamaño cuando se almacenan en la base de datos. Con fotografías de 3 MB. en una base de datos interna de HSQLDB, recibirá inmediatamente errores de Java `[NullpointerException]` que hace imposible almacenar los registros. Esto puede conducir a la pérdida de todos los registros que se ingresaron en la sesión en curso.

La entrada en un control de imagen se realiza haciendo doble clic con el ratón para abrir un cuadro de diálogo de selección de archivos o haciendo clic con el botón derecho para elegir si se va a eliminar o reemplazar una imagen. De manera predeterminada, un control gráfico no tiene tabulación.

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Imagen

El gráfico (imagen) seleccionado aquí solo se muestra dentro del control, mientras se edita el formulario. No tiene importancia para una entrada posterior.

Escala

No: la imagen no se ajustará al control.
Si es demasiado grande, el control solo mostrará una parte de la imagen. La imagen no se distorsiona.
Mantener proporciones: la imagen se ajusta al control, pero no se distorsiona (se conserva la relación de aspecto).
Tamaño Automático: la imagen se ajusta al control aunque se puede mostrar de manera distorsionada.

Pestaña Datos

Nada especial que señalar.

Pestaña Sucesos

Los sucesos *Modificado*, *Texto modificado*, *Antes de actualizar* y *Después de actualizar* no están disponibles.

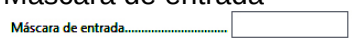
Campo enmascarado

Utiliza una máscara de entrada para controlar la entrada al campo. Los caracteres se especifican previamente para posiciones particulares, determinando las propiedades de los caracteres ingresados. Los caracteres preestablecidos se almacenan junto con los ingresados.

Además de las propiedades comunes, enumeradas en la página 135 están disponibles las siguientes:

Pestaña General

Máscara de entrada



Determina los caracteres que se pueden ingresar.

Máscar de caracteres

La presentación en el formulario (lo que verá el usuario).

La longitud de la *Máscara de entrada* determina cuántos caracteres se pueden ingresar. Si la entrada del usuario no coincide con la máscara, la entrada se rechaza al abandonar el control. Los siguientes caracteres están disponibles para definir la máscara de edición:

Carácter	*Significado*
L	Un texto constante. Esta posición no se puede editar. El carácter real se muestra en la posición correspondiente en la máscara literal.
a	Representa cualquiera de las letras a – z / A – Z. Las letras mayúsculas no se convierten en minúsculas.
A	Representa cualquiera de las letras A – Z. Si se ingresan letras minúsculas, se convertirán automáticamente a mayúsculas.
c	Representa cualquiera de los caracteres a – z / A – Z más los dígitos 0 – 9. Las letras mayúsculas no se convierten en minúsculas.
C	Representa cualquiera de las letras A – Z más los dígitos 0–9. Si se ingresan letras minúsculas, se convertirán automáticamente a mayúsculas.
N	Solo se pueden ingresar los dígitos 0 – 9.
x	Todos los caracteres imprimibles están permitidos.
X	Todos los caracteres imprimibles están permitidos. Si se ingresan letras minúsculas, se convertirán automáticamente a mayúsculas.

Así, por ejemplo, puede definir la *máscara de caracteres* como «__ / __ / 2012» y la *máscara de entrada* como "NNLNNLLLLL", para permitir al usuario ingresar solo cuatro caracteres para una fecha (día y mes).

Pestaña Datos

Nada que destacar.

Pestaña Sucesos

El suceso *Modificado* no está disponible.

Control de tablas

Este es el control más completo. Proporciona una tabla, que puede estar provista con controles individuales para las columnas. Esto no solo permite ver los datos reales durante la entrada, sino también los datos ingresados anteriormente, sin la necesidad de usar la barra de navegación para desplazarse por los registros.

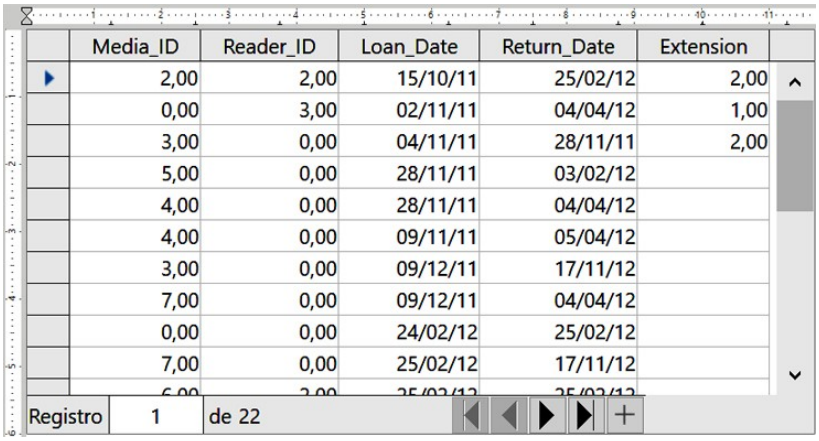
Figura 98

No todos los controles posibles en un formulario pueden seleccionarse para un control de tablas. Los *botones*, los *botones de imagen* y los *botones de opción* no se pueden usar.

El *Asistente de elementos de tabla* reúne en una ventana los campos que aparecerán en la tabla.

En el control, la tabla *Loan* está disponible para editar. Además del campo *ID* (clave principal) y el campo *Media_ID_BC* (entrada de datos mediante un escáner de código de barras), todos los campos deben usarse en el control.

El control de la tabla creado anteriormente ahora debe desarrollarse más para permitir la entrada en la tabla *Loan*. Para campos como *Reader_ID* o *Media_ID*, sería más útil poder elegir el lector o los datos directamente, en lugar de un número que represente al lector o los datos. Para este propósito, se pueden colocar dentro del control de la tabla controles como los *Cuadros de lista*. Esto se declara más tarde. El formato del campo *Extensión* con dos decimales no fue especialmente intencionado.



Media_ID	Reader_ID	Loan_Date	Return_Date	Extension
2,00	2,00	15/10/11	25/02/12	2,00
0,00	3,00	02/11/11	04/04/12	1,00
3,00	0,00	04/11/11	28/11/11	2,00
5,00	0,00	28/11/11	03/02/12	
4,00	0,00	28/11/11	04/04/12	
4,00	0,00	09/11/11	05/04/12	
3,00	0,00	09/12/11	17/11/12	
7,00	0,00	09/12/11	04/04/12	
0,00	0,00	24/02/12	25/02/12	
7,00	0,00	25/02/12	17/11/12	

Registro 1 de 22

Figura 99: Salida del Asistente de elementos de tabla

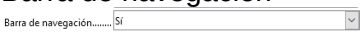
Además de las propiedades comunes, enumeradas en la página 135 están disponibles las siguientes:

Pestaña General

Altura de fila

Con líneas individuales. La altura se ajusta automáticamente al tamaño de fuente. Los campos de texto multilínea se muestran como líneas individuales con controles para desplazarse.

Barra de navegación



Al igual que con las tablas, el borde inferior del control muestra el número de registro y las ayudas de navegación.

Marcador de registros

De manera predeterminada, hay un marcador de registro en el borde izquierdo del control. Indica el registro en uso. Puede utilizar el marcador para eliminar todo el registro.

Pestaña Datos

Dado que este es un campo que no contiene datos en sí, pero administra otros campos, no hay propiedades de datos.

Pestaña Sucesos

Los sucesos *Modificado* y *Texto modificado no están disponibles.* Se agrega el suceso *Se produjo un error*.

Etiqueta

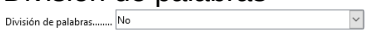
Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Etiqueta

La *Etiqueta* actúa como una descripción de otro control. Si la etiqueta está vinculada a un control, se puede usar como un acceso directo a él. Un símbolo «~» indica una letra en concreto que se convierte en el atajo «No~mbre» define la **m** como atajo de teclado. Cuando el cursor está en cualquier parte del formulario, *Alt* + *m* va directamente al campo.

División de palabras



Se aplica solo en modo multilínea. De forma predeterminada, el texto una etiqueta no está ajustado. Si es demasiado larga se trunca. **Precaución**: la *División de palabras* no reconoce espacios, por lo que si el campo es demasiado pequeño, el texto no aparecerá completo.

Pestaña Datos

No hay propiedades de datos.

Pestaña Sucesos

El campo de etiqueta solo reacciona a sucesos que están conectados con el ratón, una tecla o el foco.

Grupo de opciones

Un grupo de opciones es un marco que agrupa gráficamente varios controles y les asigna una etiqueta colectiva.

Si se crea un grupo de opciones mediante el *Asistente de elementos de grupo*, el asistente asume que habrá varios botones de opción y estarán juntos dentro de ese marco.

Figura 100

En este caso, el formulario se basa en la tabla *Reader*. Se pretende usar opciones para la elección de género del lector. Los elementos, que se verán como etiquetas al lado del botón, se deben definir manualmente. Más adelante, en el mismo asistente, se vincularán con el campo adecuado de la tabla.

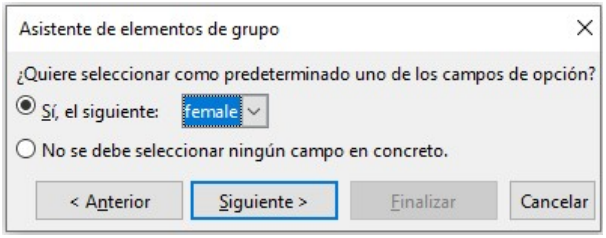


Figura 101

Aquí la opción predeterminada es *female*. Si no debe haber valor predeterminado en la tabla subyacente, la entrada predeterminada es `NULL`.

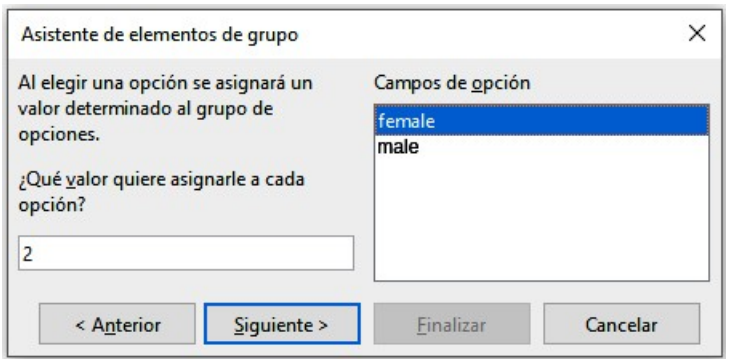


Figura 102

De manera predeterminada, el asistente asigna a los botones de opción valores distintos, en este caso *1* para *female* y *2* para *male*. Estos valores corresponden a los ejemplos de campos clave principal en la tabla *Gender*.

Al hacer clic en un botón de opción el valor seleccionado se transfiere al campo *Gender_ID* de la tabla *Reader* subyacente del formulario. De esta manera la tabla *Reader* se enlaza con la clave externa correspondiente de la tabla *Gender* utilizando el botón de opción.

Figura 103

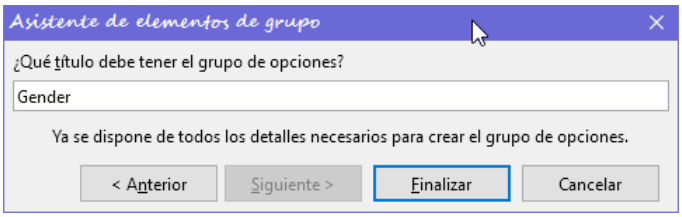


Figura 104

El grupo de botones de opción recibe un marco (cuadro de grupo) con la etiqueta *Gender*.

Figura 105

Si se selecciona *female* en el formulario activo, se deselecciona *male*. Esta es una característica de los botones de opción que están vinculados al mismo campo en la tabla subyacente. En el

ejemplo que se muestra arriba. Los botones de opción son un remplazo de un cuadro de lista de dos elementos.

Además de las propiedades descritas en la página 135, están disponibles las siguientes:

Pestaña General

Se puede cambiar el valor predeterminado de la etiqueta. Por el momento, las propiedades del marco (*grosor de línea*, *color de línea*) no se pueden cambiar, pero puede cambiar el formato de la fuente.

Pestaña Datos

Ninguna, ya que este control solo sirve para la agrupación visual de campos.

Pestaña Sucesos

El cuadro de grupo solo reacciona a sucesos que están conectados con el ratón, una tecla o el foco.

Botón

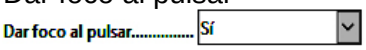
Además de las propiedades comunes, enumeradas en la página 135 están disponibles las siguientes:

Pestaña General

Etiqueta

Al insertar el carácter «~», una letra específica de la *Etiqueta* se puede convertir en un acceso directo. «Push~Button» define la «b» como el acceso directo. Cuando el cursor está en cualquier parte del formulario, *Alt+b* lo llevará al botón.

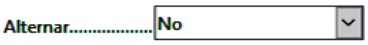
Dar foco al pulsar



El botón recibe el foco cuando se hace clic en él

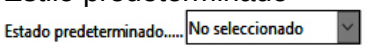
Esto no siempre es lo deseado. Por ejemplo, es posible que desee hacer clic en el botón para colocar contenido en un control de formulario diferente, en cuyo caso el cursor debe permanecer en ese control cuando hace clic en el botón.

Alternar



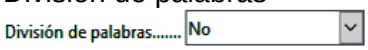
Presentación del botón, en caso afirmativo, el botón se puede mostrar presionado. El estado del botón se muestra como un interruptor. Cuando lo presiona por segunda vez, muestra un botón no presionado.

Estilo predeterminado



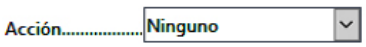
Solo activo, cuando *Alternar* está configurado en Sí. Seleccionado corresponde al botón presionado.

División de palabras



Ajusta el texto introducido si el botón es demasiado estrecho.

Acción



Están disponibles una variedad de acciones similares a las de la barra de navegación.

Si la acción es abrir un documento o sitio web, se debe especificar la URL en el siguiente campo.

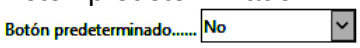
URL

HTML: Archivo a llamar con este botón. Aquí debe elegir el recurso que se abrirá mediante *Abrir documento / sitio web* en *Acción*.
Además de HTML, se puede usar *Acción* para abrir otros módulos de LibreOffice. Si ingresa aquí «`.uno: RecSearch`», se asignará la función de búsqueda del navegador al botón.
También puede abrir archivos de Writer para que al presionar el botón se realice una *Combinación de correspondencia* utilizando registros de la base de datos.
Las órdenes posibles para la macro se pueden conseguir utilizando la grabadora de macros.

Marco

Solo para formularios HTML: el marco de destino (disposición de marcos para diferentes páginas HTML) en el que se debe abrir el archivo.

Botón predeterminado

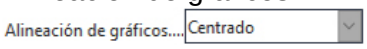


El botón predeterminado se enmarca cuando se establece en Sí. Cuando hay varios botones alternativos en un formulario, el más útil debe tener esta característica. Se activa presionando la tecla *Intro*, cuando ninguna otra acción necesita depender de esta tecla. Solo un botón en el formulario puede ser el botón predeterminado.

Imagen

Determina si debería aparecer una imagen en el botón.

Alineación de gráficos



Especifica la alineación del gráfico con el texto. Solo está activo cuando se ha seleccionado una imagen.

Pestaña Datos

Ninguna. Un botón solo realiza acciones.

Pestaña Sucesos

Aprobar acción, *Ejecutar acción* y *Estado del elemento cambiado*.

Botón de imagen

Además de las propiedades ya descritas en la página 135, están disponibles las siguientes:

Pestaña General

Similar a un botón normal. Sin embargo, este botón no tiene texto y no parece un botón, solo ve un marco alrededor de la imagen.

De manera predeterminada, un botón de imagen no tiene tabulación.

Precaución: En el momento de escribir esta guía, casi ningún suceso funciona con este botón. Prácticamente solo se puede usar con macros.

Pestaña Datos

Ninguna; este control solo realiza acciones.

Pestaña Sucesos

Aprobar acción y todos los sucesos que involucren al ratón, una tecla o el foco.

Barra de navegación

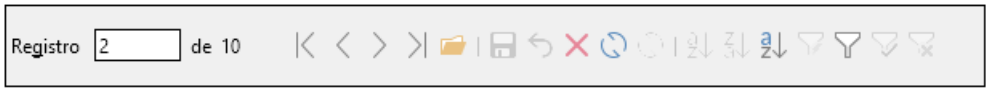


Figura 106: Controles de barra de navegación

La *Barra de navegación de formularios estándar* se muestra en el borde inferior de la pantalla en los formularios. Esta barra navegación puede causar un breve desplazamiento hacia la derecha del formulario, a medida que se va dibujando en la pantalla. Esto puede causar cierta distracción cuando la barra de navegación se desactiva para algunas partes del formulario visible (cuando hay subformularios o varios formularios en el formulario visible).

Por el contrario, un control de *Barra de navegación* que forma parte del formulario, separado de los elementos correspondientes, indica claramente a través de qué elementos está navegando con la barra de herramientas. El formulario de préstamos (*Loan*), debe buscar primero entre los lectores (tabla *Readers)* y luego mostrar los artículos prestados al lector.

El control de la barra de navegación se puede colocar cerca del campo correspondiente al lector, por lo que el usuario distingue mejor que la barra de navegación se utiliza para los lectores y no para los artículos prestados al lector.

La barra de navegación de formularios estándar pone a disposición los botones que se muestran en la figura 107. El control de la barra de navegación muestra los mismos botones, excepto los de *Buscar registro*, *Filtros* basados en formularios y *Fuente de datos como Tabla.*

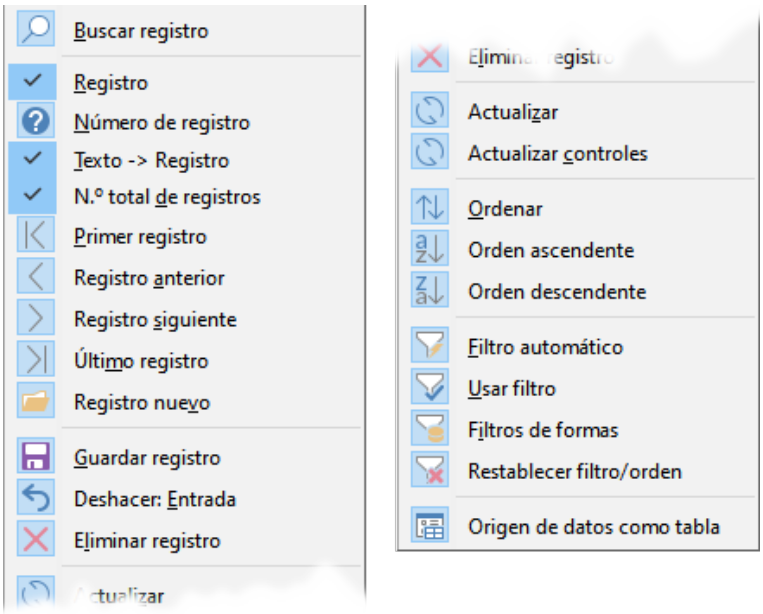
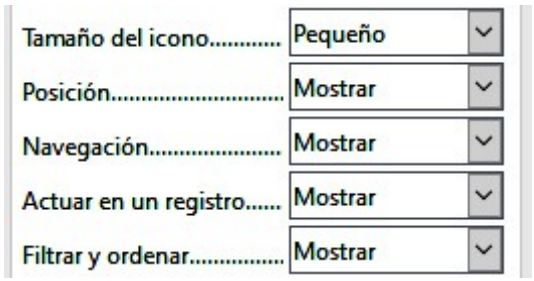


Figura 107: Botones de Barra de navegación

Además de las propiedades comunes, enumeradas en la página 135, Están disponibles las siguientes:.

Pestaña General



El tamaño del icono es ajustable. Además, puede elegir qué grupos se muestran. Estos se muestran en la figura 106 de izquierda a derecha: *Posición*, *Navegación*, *Actuar en un registro* y *Filtrar y ordenar*.

Pestaña Datos

Ninguno, ya que este control solo realiza acciones.

Pestaña Sucesos

Todos los sucesos que involucren al ratón, una tecla o el foco.

Independientemente de este control de formulario, la barra de navegación de las propiedades del formulario, naturalmente, sigue existiendo con los mismos elementos que la figura anterior.

Esta barra de navegación proporciona además la búsqueda de registros generales, el filtro basado en formularios y la visualización de la fuente de datos subyacente del formulario en la vista de tabla sobre el formulario.

Si no está trabajando con un único formulario, sino con subformularios y formularios auxiliares, debe evitar que esta barra de navegación desaparezca al cambiar de formulario. Eso crea un efecto perturbador en la pantalla (utilice la opción «acoplar barra de herramientas»).

Botones de selección y barras de desplazamiento

Ninguno de estos campos tiene una conexión directa a los datos en la base de datos. Solo se pueden utilizar por medio de macros, donde el botón de selección está integrado, por ejemplo, en un campo numérico.

Una barra de desplazamiento se diferencia de un botón de selección en que, además de los botones para aumentar y disminuir el valor dentro de un rango fijo, tiene un control deslizante

Cuando se abre el formulario, se debe indicar a la barra de desplazamiento el valor del campo al que está vinculada.

Esta macro está vinculada a **Propiedades del formulario > Sucesos > Después del cambio de registro.**

```
SUB scrollbar_form(oEvent AS OBJECT)
  DIM oForm AS OBJECT
  DIM oField AS OBJECT
  oForm = oEvent.Source
  oField = oForm.getByName(»scrollbar»)
  oField.ScrollValue = oForm.getInt(oForm.findColumn(»numeric_value»))
END SUB
```

Primero se declaran las variables. El suceso que manejará se especifica en el formulario al que se vinculará la macro. La barra de desplazamiento se encontrará por su nombre en el formulario. Su valor actual se establece por el número almacenado en *numeric_field* de la tabla subyacente.

Si el valor se modifica utilizando la barra de desplazamiento, este cambio debe transmitirse al campo subyacente.

Esta macro está vinculada a **Propiedades: Barra de desplazamiento > Sucesos > Mientras se ajusta.**

```
SUB Scrollbar_change(oEvent AS OBJECT)
  DIM oForm AS OBJECT
  DIM oField AS OBJECT
  DIM inValue AS INTEGER

  oField = oEvent.Source.Model
  inValue = oEvent.Value
  oForm = oField.Parent
  oForm.updateInt(oForm.findColumn("numeric_value"),inValue)
END SUB
```

Primero se declaran las variables. El valor seleccionado se lee como un entero del campo. El campo correspondiente en la tabla subyacente *numeric_value* se actualiza a este valor.

En el «Capítulo 9, Macros», se proporciona una explicación detallada de este tipo de código.

Control oculto

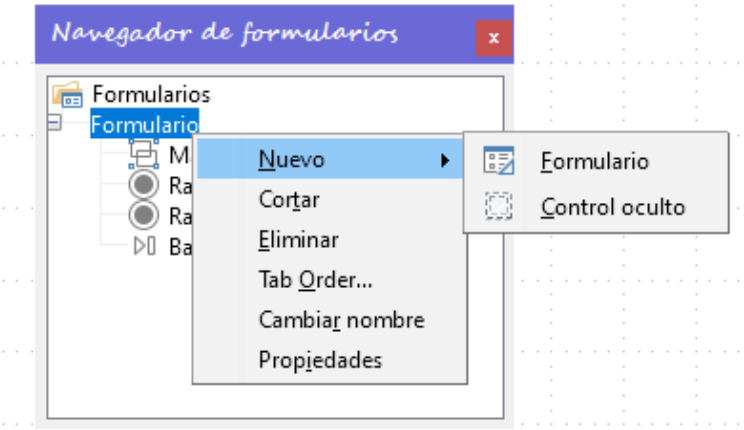


Figura 108

El *Control oculto* es un tipo de control que no se puede insertar desde la barra de herramientas de controles. Debe crearse dentro del *Navegador de formularios*, utilizando el menú contextual.

Como cualquier otro control, un control oculto es parte del formulario. Sin embargo, no es visible en la interfaz de usuario. Al igual que para su creación, para acceder a sus propiedades se debe hacer desde el navegador de formularios.

Un control oculto tiene solo unas pocas propiedades. El *Nombre* y la *información adicional* que tienen el mismo significado que para los otros controles. Además, se le puede asignar un *Valor*.

Un control oculto no tiene sentido si no se están utilizando macros. Sin embargo, con las macros, puede ser útil para almacenar valores intermedios en algún lugar del formulario y acceder a estos posteriormente. Un ejemplo de esta manera de usar macros se describe en el «Capítulo 9, Macros», en la sección «Cuadros de lista jerárquicos».

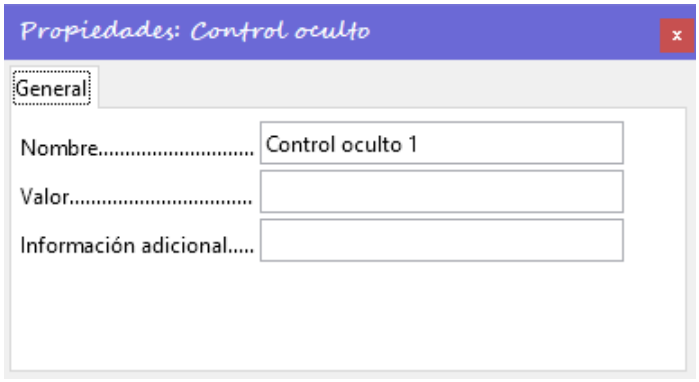


Figura 109

Selección múltiple

Si usa el botón *Seleccionar* para seleccionar una región amplia o varios elementos de un formulario, se pueden modificar conjuntamente las propiedades comunes a dichos controles.

Advertencia

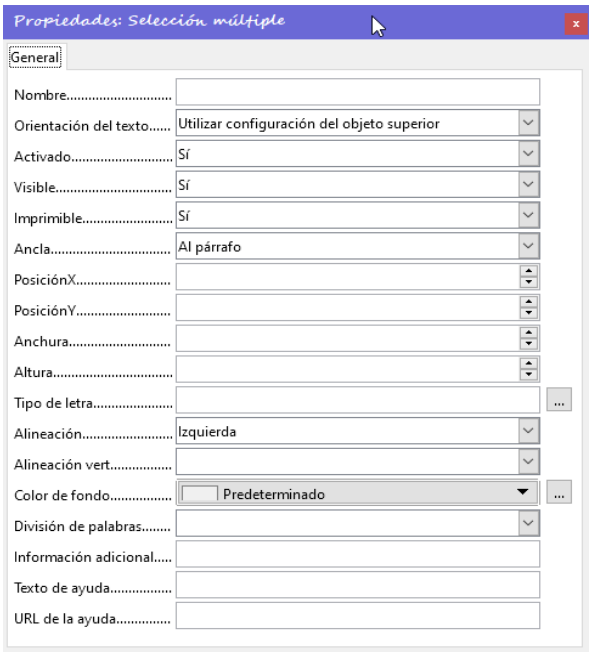
No debe escribir en la propiedad *«nombre»*. Eso haría que todos los elementos seleccionados adquirieran el mismo nombre y haría muy difícil encontrar elementos individuales con el *Navegador de formularios*. La gestión del formulario con macros mediante el uso del nombre de los controles se hace totalmente imposible.

La *Selección múltiple* es útil para cambiar la fuente, la altura o el color de fondo de los controles conjuntamente para hacerlos más homogéneos. (Tenga en cuenta que cambiar el color de fondo también afecta a las etiquetas).

Si desea modificar solo un tipo de controles determinado, mantenga presionada la tecla *Mayús* mientras hace clic en cada control directamente (o bien selecciónelos en el Navegador). Haga clic derecho en la selección para acceder a las propiedades. El número de las propiedades que se pueden cambiar será mayor, ya que se trata de controles del mismo tipo. Podrá cambiar en bloque cualquier elemento que esté disponible para ese tipo de control.

Las posibilidades de selección múltiple dependen, por lo tanto, de la selección de los controles.

Figura 110:



Propiedades generales de los campos de formulario en una Selección múltiple.

Un formulario sencillo completo

Un formulario simple tiene controles de formulario para escribir o leer registros de una sola tabla o consulta. Su construcción se muestra en el siguiente ejemplo:

Figura 111

Hasta ahora hemos estado viendo las propiedades del los controles y cómo agregarlos a un formulario. Ahora vamos a mostrar la creación de un formulario simple para préstamos de biblioteca, usando algunas variantes. Una manera rápida puede ser utilizando el asistente para formularios que se describe en el «Capítulo 8, Primeros pasos con Base», en la *Guía de primeros pasos.*

En este caso, describimos la creación en *modo diseño*.

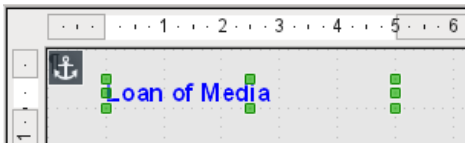


Figura 112

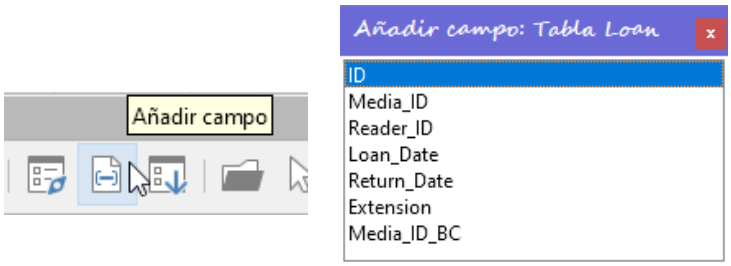
El encabezado del formulario se creó utilizando un campo de etiqueta y se cambió la fuente.

El campo de etiqueta está anclado a un párrafo en la esquina superior izquierda del documento. Usando el menú contextual del campo de etiqueta, se creó un formulario vinculado a la tabla de préstamos (*Loan*). Consulte «Propiedades de formulario» en la página 131.

La página también ha recibido un fondo de color uniforme seleccionándolo desde el *menú principal*, eligiendo **Formato > Página** y pulsando en la pestaña *Área.*

Agregar grupos de campos

Una variante rápida para la entrada directa de controles con etiquetas es usar el icono *Añadir campo* de la barra de herramientas *Diseño de formulario*.



Esta función permite seleccionar los campos de la tabla subyacente.

Haga doble clic en cada campo de la tabla *Loan* para insertar los controles correspondientes en el formulario. Cada campo de la tabla se insertará en el formulario como un grupo (etiqueta y control). Puesto que estos grupos se colocan superpuestos en el formulario, debe separarlos para que el formulario se vea como en la Figura 113.

Se han seleccionado todos los campos excepto *Media_ID_BC*, diseñado para usar un escáner de código de barras.

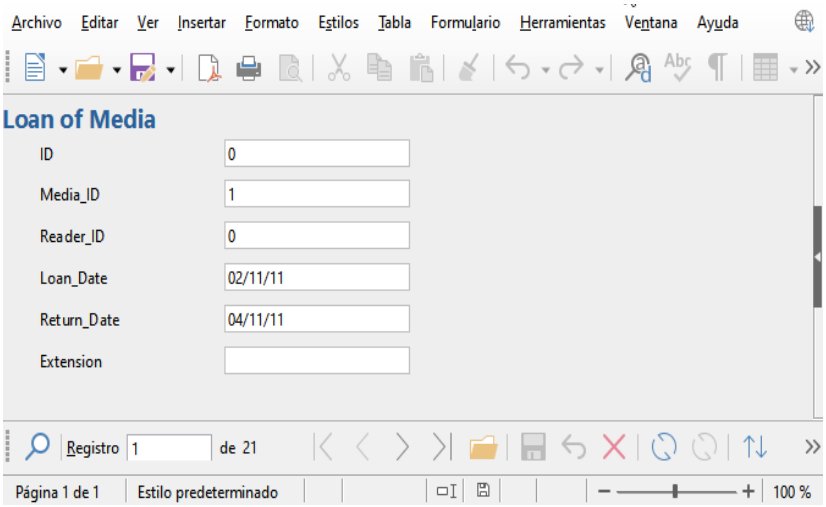


Figura 113: Formulario sencillo

Para cada campo de tabla, se ha seleccionado automáticamente un control de formulario apropiado:

- Los números están en campos numéricos y se declaran como enteros sin decimales.

- Los campos de fecha se representan correctamente como control de campo de fecha.
- Todos los campos tienen el mismo ancho. Si se hubiera incluido un control gráfico, aparecería como un campo cuadrado.

Ajuste de las proporciones de los controles

Ahora podemos hacer algunas cosas creativas, como ajustar la longitud de los controles y convertir las fechas en controles desplegables.

Lo más importante es que los campos *Media_ID* y *Reader_ID* sean fáciles de usar, a menos que cada usuario de la biblioteca tenga una tarjeta de identificación y cada artículo prestado reciba una *ID* al ingresar (esta alternativa no se contempla en el ejemplo).

Para ajustar controles individuales, (recordemos que se han creado como grupos) debemos editar el grupo. Esto se puede hacer con un clic derecho en el grupo y luego siguiendo el menú contextual (Figura 114). Pero para futuros trabajos, será más claro si usamos el Navegador de formularios, puesto que se pueden seleccionar los controles de un grupo de manera individual.

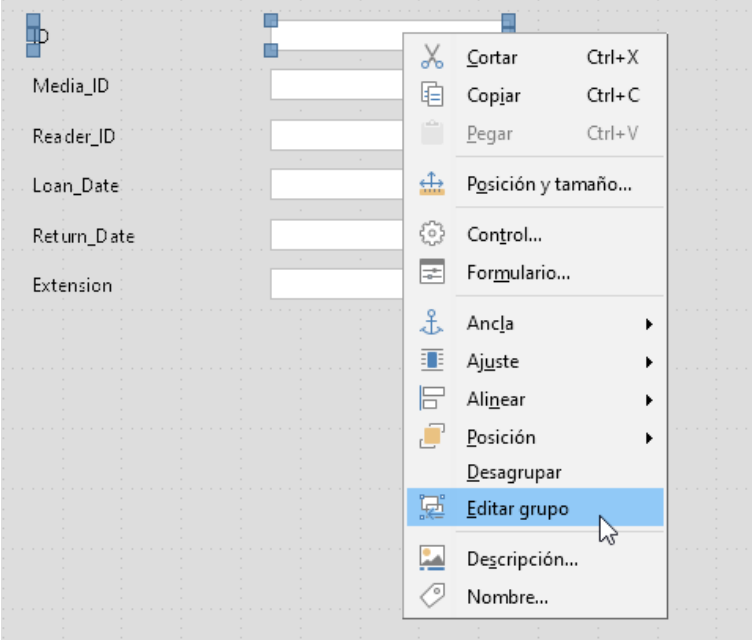


Figura 114: Controles de formulario: edición del grupo

El navegador de formularios muestra todos los elementos del formulario con sus etiquetas. Para los controles, los nombres se toman directamente de los nombres de los campos de la tabla. Las etiquetas tienen nombres con el sufijo «Etiqueta» al estar agrupadas con los controles.

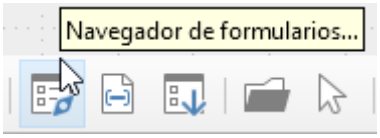


Figura 115: Selección de controles usando el navegador de formularios

Un clic en *Media_ID* selecciona este control (Figura 115). Para reemplazar el tipo de control seleccionado con un tipo distinto debe hacer clic con el botón derecho y seleccionar en el menú contextual *Reemplazar por*…(Figura 116)

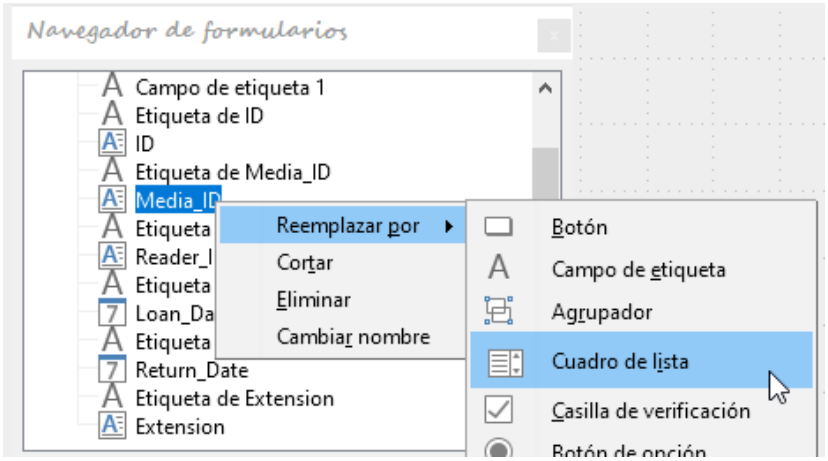


Figura 116: Reemplazar un tipo de control por otro utilizando el Navegador de formularios

Se remplazarán los controles *Media_ID* y *Reader_ID,* antes campos de texto, por cuadros de lista. El cambio se hace visible en el *Navegador de formularios* por el cambio del icono asociado.

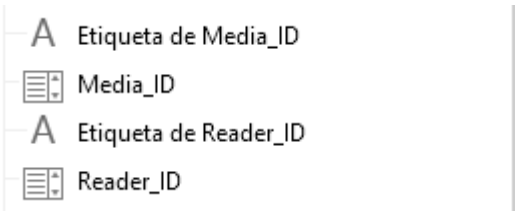


Figura 117

Figura 118

Ahora se puede crear la consulta SQL para el campo de lista mediante la interfaz gráfica de usuario haciendo clic en el botón de la derecha. Esto se lleva a cabo automáticamente cuando se crea un cuadro de lista directamente, pero no cuando se forma mediante la conversión de otro tipo de control.

Los controles cuadros de lista permiten mostrar algo diferente a los campos *ID* (números) sin usar macros, mediante el *tipo de contenido de la lista.* En este caso haremos consultas SQL para que nos muestren el título del artículo y el nombre del lector las consultas aquí serían:

Para el control *Media_ID:* `SELECT "Title","ID" FROM "Media"`

Para el control *Reader_ID:* `SELECT "LastName"||', '||"FirstName","ID" FROM "Reader"`

Para más detalles sobre los comandos SQL, consulte el «Capítulo 5, Consultas».

Además de corregir los *Cuadros de lista* para hacerlos desplegables, se pueden corregir los siguientes defectos:

- Las etiquetas para estos cuadros de lista deberían ser *Media* en lugar de *Media_ID* y *Reader* en lugar de *Reader_ID.*
- El control de *ID* debe declararse como de solo lectura.
- Cualquier campo que no sea absolutamente necesario para emitir préstamos para un nuevo artículo no necesita un tabulador. Así se puede recorrer el formulario más rápidamente. Si es necesario, el orden de tabulación también se puede ajustar mediante la secuencia de activación (consulte la página 135). Solo los campos *Media_ID*, *Reader_ID* y *Loan_date* deben estar accesibles en todos los casos con la tecla *Tab*
- Si el formulario está destinado a realizar préstamos, es innecesario y también confuso que se muestren los artículos devueltos. Los artículos con una fecha de devolución deben filtrarse. Además, el orden de visualización podría ordenarse por *Reader*, de modo que los artículos prestados a la misma persona se muestren sucesivamente. Consulte la nota sobre «Propiedades de formulario» en la página 131. Sin embargo, aquí existe el problema de que los lectores solo pueden ordenarse por su número identificativo (*ID*), no alfabéticamente, porque la tabla subyacente al formulario solo contiene este número.

Agregar campos individuales

La adición de controles individuales es un poco más complicada.

Debe seleccionar el control adecuado de la barra de herramientas, dibujar el tamaño en el formulario y especificar el campo apropiado de la tabla subyacente. Además, el tipo de campo debe elegirse correctamente; por ejemplo, los campos numéricos tienen dos decimales de manera predeterminada.

Solo cuando se crean cuadros de lista entra en juego el asistente, haciendo más fácil a un usuario novel llevar a cabo los pasos para crear los controles (hasta cierto punto) correctos.

El asistente no llega a cumplir todos los requisitos porque:

- Las entradas no se ordenan automáticamente.
- No es posible combinar varios campos en el contenido de un cuadro de lista.

Una vez más, debemos realizar mejoras retrospectivas, para que el código SQL necesario se pueda crear con bastante rapidez utilizando el editor de consultas incorporado.

Al agregar controles individuales, el control y su etiqueta deben estar explícitamente asociados (consulte «Propiedades comunes para los controles» en la página 135).

En la práctica, puede ser mejor si no asocia controles con etiquetas, porque no tendrá que editar el grupo antes de cambiar las propiedades de un control.

Formulario sencillo con controles de tabla

En este caso crearemos otro formulario, pero en lugar de usar los controles sencillos usaremos controles de tabla. Crearemos dos controles de tabla mediante el uso del asistente (ya descrito en la página 151), los dos con base en la tabla *Loan*.

El uso del *Asistente de tabla* para crear un control de tabla, sin embargo, tiene algunos defectos que deben mejorarse:

- Los campos del control de tabla *Media_ID* y *Reader_ID* deben convertirse en cuadros de lista.
- Los campos numéricos no deben contener decimales, (el Asistente siempre especifica dos lugares decimales para los números).

No es posible cambiar los campos dentro del *control de la tabla* utilizando el mismo método descrito para otros controles. En el *navegador de formularios* no aparecen los controles asociados al control de tabla. El navegador desconoce los controles que se encuentran dentro del control de la tabla, que se refirieren a los campos de la tabla subyacente. Esto también es una limitación para las macros, cuando se intenta acceder a los campos de un control de tabla mediante macros no se puede acceder por su nombre.

El concepto de menú contextual para los controles dentro de un *control de tabla* cambia, en su lugar está el concepto de columna, al seleccionar un control de tabla y hacer clic derecho en una columna aparece el menú contextual. Para acceder a las propiedades del control debe seleccionar *Columna*.

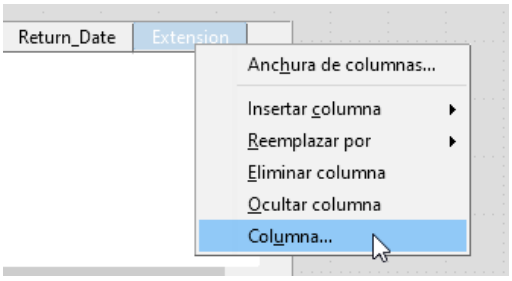


Figura 119

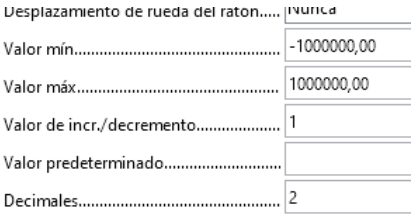


Figura 120

Aquí, por ejemplo, el campo numérico *Extensión* se puede cambiar para que no se muestren lugares decimales. Además, el valor mínimo predeterminado de –1,000,000.00 apenas tiene sentido para una extensión de préstamo. El número siempre debe ser pequeño y positivo.

Cuando se activen las propiedades de una columna, puede seleccionar otra columna sin cerrar el cuadro de diálogo de propiedades. De esta manera, puede modificar las propiedades de los campos, sin tener que guardar las modificaciones entre uno y otro.

Nota

El valor se guarda en el cuadro de diálogo cuando se realiza el movimiento entre las propiedades. También ocurre lo mismo si se pasa de una columna a la siguiente.

Usando el menú contextual, también es posible reemplazar un tipo de campo por otro. Sin embargo, no todos los tipos están disponibles. No hay *botones*, *botones de opción* ni *controles de imagen*.

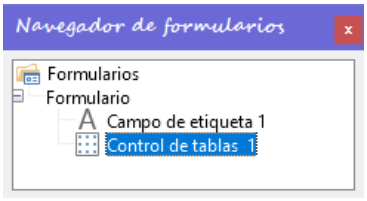


Figura 121

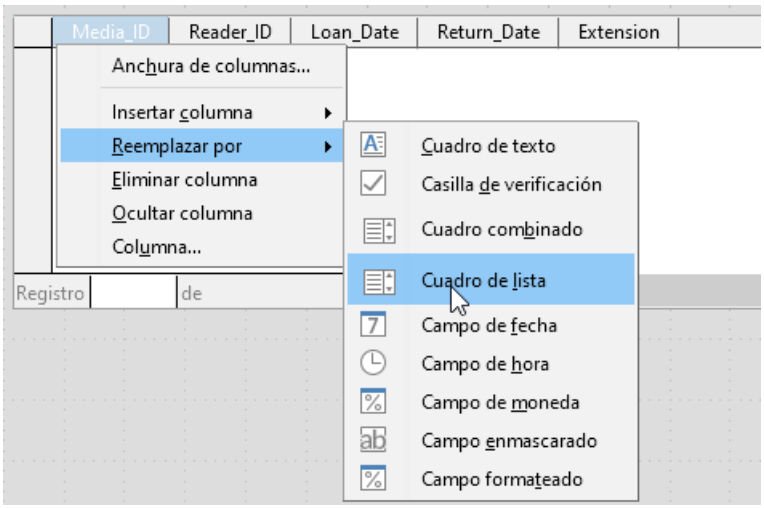


Figura 122

Las propiedades de los campos integrados en un control de tabla no son tan completas como las de los que están fuera. La fuente, por ejemplo, solo se puede configurar para todo el control de la tabla. Además, no tiene la opción de omitir el salto del tabulador en columnas individuales.

Consejo

Puede moverse por un formulario con el ratón o la tecla *Tab*. Si tabula en un control de tabla, el cursor se moverá un campo a la derecha para cada registro, al final de la línea, volverá al primer campo del siguiente registro en el control de la tabla. Para salir del control de la tabla, use *Ctrl+Tab.*

Si solo deben estar visibles ciertos campos, puede usar varios controles de tabla en el formulario, ya que la tecla *Tab* es utilizada de manera predeterminada para cada control de tabla.

El formulario que se muestra en la figura 123 es para el préstamo de artículos. En el control de tabla superior solo se muestran los campos estrictamente necesarios. En el inferior se muestran todos los campos, de modo que es evidente para qué lector y artículo se especifica la devolución.

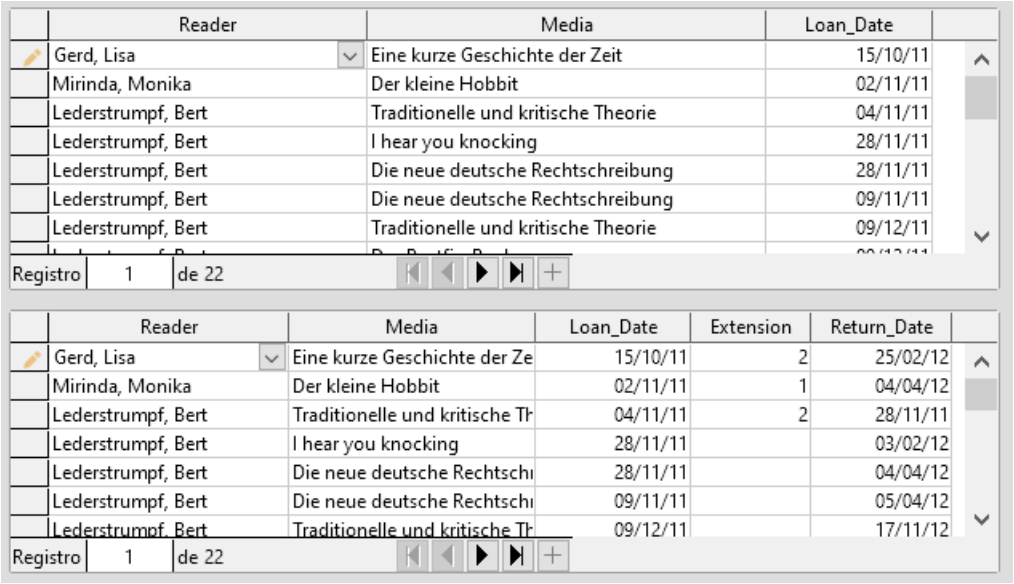


Figura 123: Un formulario con múltiples controles de tabla

Esta figura muestra una fallo estético que requiere atención urgente. En el control de la tabla superior, el mismo elemento se repite varias veces. Esto sucede porque la tabla también muestra los artículos prestados que ya se han devuelto. Por lo tanto, los datos deben filtrarse para mostrar solo los préstamos. Los registros con una fecha de devolución no deberían aparecer.

Este filtrado es posible mediante una consulta o directamente usando las propiedades del formulario. Si se hace usando las propiedades del formulario, el filtro se puede desactivar temporalmente durante la entrada de datos. El filtrado mediante consultas se describe en el «Capítulo 5, Consultas». Aquí se describe el modo de hacerlo usando *Propiedades del formulario*.

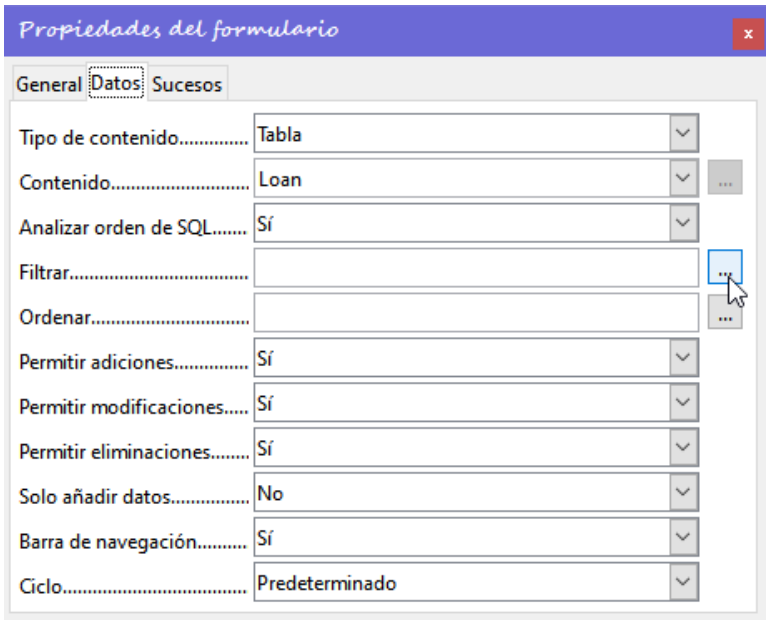


Figura 124

El filtrado se lleva a cabo utilizando el botón con los tres puntos, que abre el cuadro de diálogo que se muestra a continuación. También puede ingresar el filtro directamente en el campo *Filtrar* si conoce la sintaxis SQL.

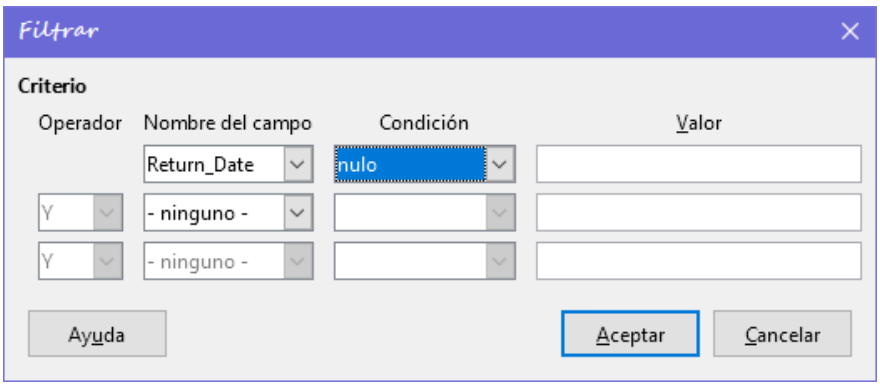


Figura 125

Usando la interfaz, ahora puede seleccionar el campo llamado *Return_Date*. Solo mostrará los registros para los que el campo esté vacío; «vacío» aquí se representa como nulo y en la consulta SQL se corresponde con NULL.

El formulario corregido que se muestra en la figura 126, ahora parece bastante más limpio. Por supuesto, todavía hay margen de mejora, pero en comparación con el formulario anterior, esta versión tiene una clara ventaja: todos los artículos prestados son visibles de un vistazo.

El procesamiento de datos usando controles de tabla es similar a usar una tabla real. Un clic derecho en el encabezado de un registro existente hace que se elimine y una entrada puede cancelarse o guardarse en el caso de nuevos registros.

Cuando se abandona una línea, el registro se guarda automáticamente.

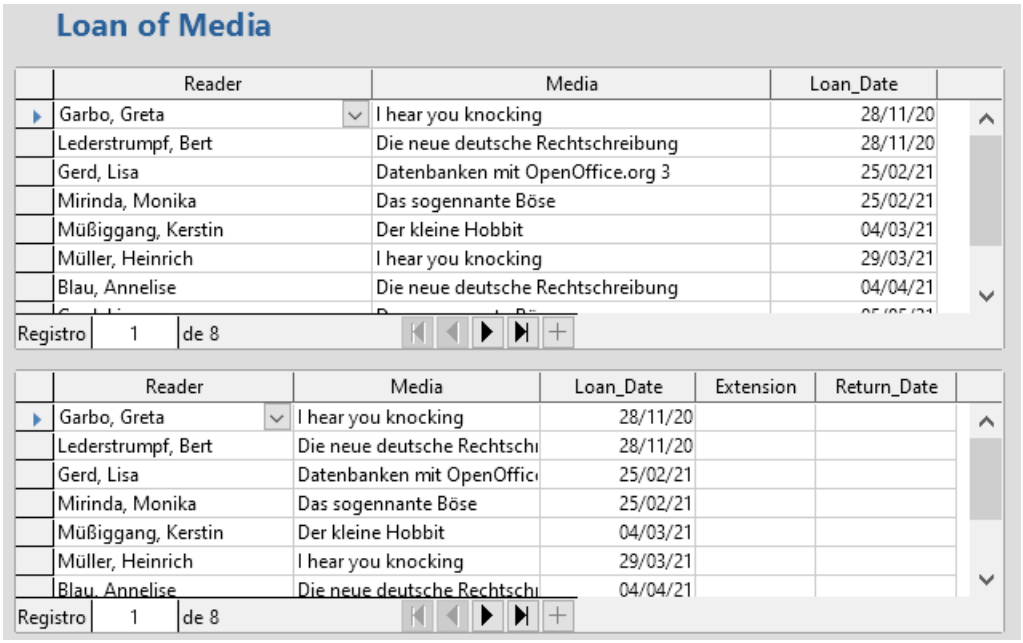


Figura 126: Formulario corregido

Aún se puede mejorar el formulario de préstamo de artículos con varios retoques:

- Sería interesante que si al seleccionar un lector en una parte del formulario los artículos prestados a este lector se mostraran en otra.
- En la tabla que se muestra arriba, puede ver muchos registros que no son necesarios porque estos artículos ya están prestados. La tabla se creó para permitir que se otorguen préstamos, por lo que sería mejor que solo apareciera una página vacía, que luego se pudiera llenar con un nuevo préstamo.

Dichas soluciones están disponibles utilizando formularios adicionales que se organicen jerárquicamente y hagan posibles vistas separadas de los datos.

Consejo

Los controles de tabla se pueden combinar con otros campos en el formulario. Por ejemplo, el campo de control de tabla muestra solo los títulos de aquellos artículos para los que se han realizado cambios en los campos de formulario subyacentes.

Cuando los controles de tabla se combinan con otros campos de formulario, hay un pequeño error, que puede solucionarse fácilmente. Si ambos tipos de campo están presentes juntos en un formulario, el cursor se mueve automáticamente desde los otros campos al control de la tabla, aunque de manera predeterminada este tipo de campo tiene la configuración *Tabstop = No.*

La solución consiste en cambiar la propiedad *Tabstop* a «Sí» y luego otra vez a «No». Lo que hará que se registre correctamente el valor «No».

Formularios principales y subformularios

Un subformulario se encuentra dentro de un formulario como un control de formulario. Al igual que un control de formulario, está vinculado a los datos del formulario principal. Sin embargo, su fuente de datos puede ser otra tabla o una consulta (o un comando SQL). Lo importante para un subformulario es que su fuente de datos está de alguna manera vinculada a la fuente de datos del formulario principal.

Figura 127

Las estructuras de tabla típicas que se prestan para su uso como subformularios son tablas con una relación de uno a muchos consulte el «Capítulo 3, Tablas». El formulario principal muestra una tabla con registros a los que se pueden vincular y mostrar muchos registros dependientes en el subformulario.

Primero usaremos la relación de la tabla *Reader* con la tabla *Loan* (vea «Capítulo 3, Tablas»). La tabla *Reader* formará la base del formulario principal y la tabla *Loan* se usará en el subformulario.

El **formulario principal** está vinculado a la tabla *Reader*. Para acelerar la búsqueda de lectores, la tabla se ordena alfabéticamente (haciendo clic en el botón «…» se puede establecer el criterio de ordenación).

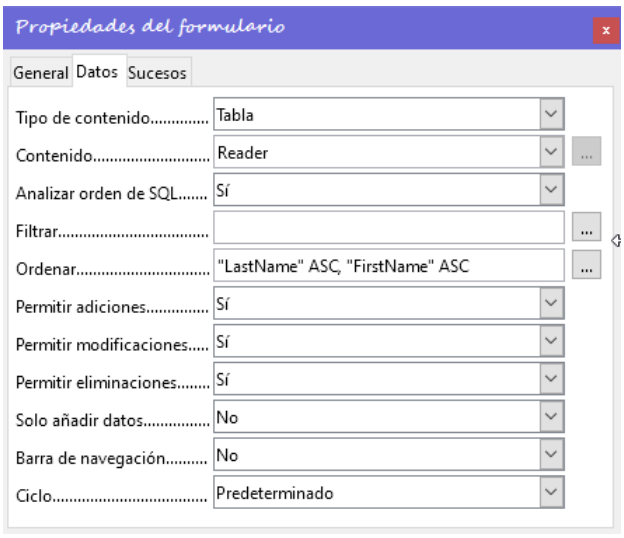


Figura 128

Lo haremos sin una barra de navegación, ya que el contenido del subformulario vendría entre el formulario principal y la barra de navegación. En su lugar, utilizaremos el control de barra de navegación (figura 106).

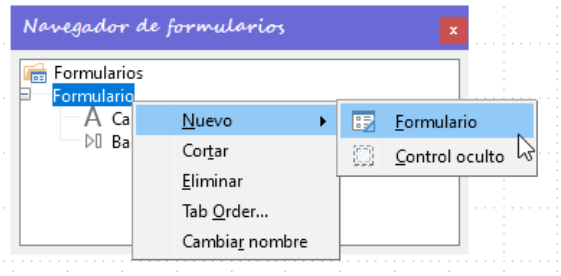


Figura 129

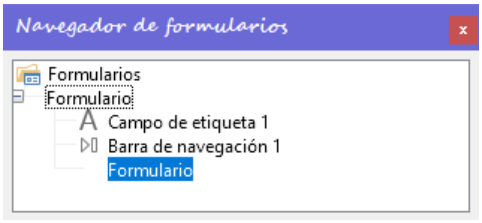


Figura 130

Haga clic con el botón derecho en el formulario principal en el *Navegador de formularios* para usar el menú contextual y crear un nuevo formulario. Una vez más, este formulario tiene el nombre predeterminado de *Formulario*, pero ahora es un elemento en la subcarpeta del formulario principal. Las propiedades del subformulario deben configurarse para darle el origen de datos adecuado, a fin de reproducir los datos correctos para el lector.

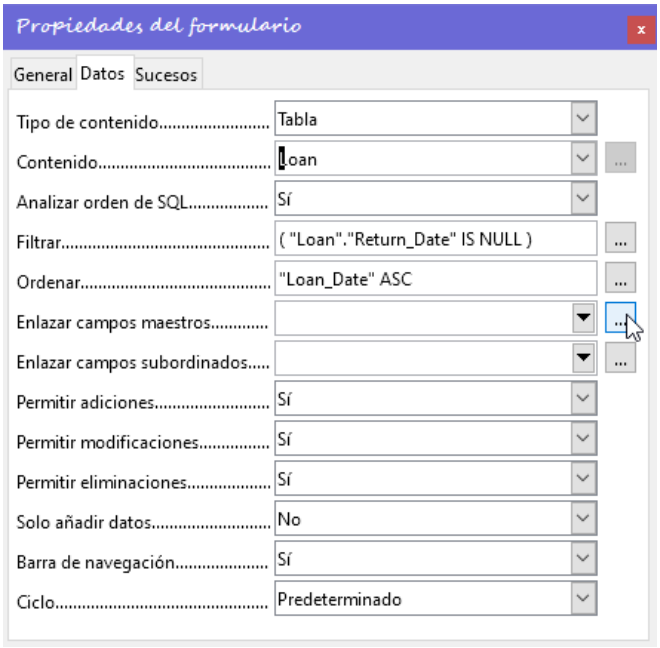
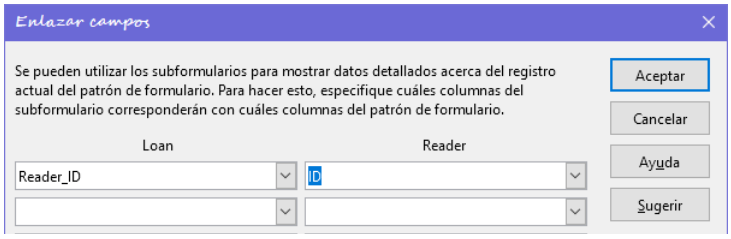


Figura 131

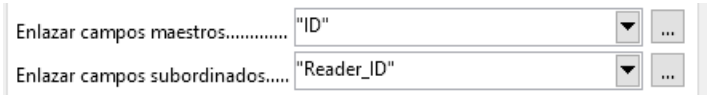
Para el **subformulario** elegimos la tabla *Loan*. Para el filtro, especificamos que el campo *Return_Date* debe estar vacío (`"Return_Date" IS NULL`). Esto evita que aparezcan los artículos que ya han sido devueltos. Los registros deben ordenarse por fecha de préstamo. El orden ascendente muestra el artículo con fecha más antigua en la parte superior.

Los campos de enlace, maestro y esclavo, se utilizan para crear un enlace al formulario principal, en el que se encuentra el subformulario. El botón con tres puntos muestra una vez más que hay disponible un diálogo para crearlos.



En *Loan*, se muestran los campos en la tabla *Loan*, en *Reader* los de la tabla *Reader*. El campo *Reader_ID* de la tabla *Loan* debe establecerse como enlazado al campo *ID* de la tabla *Reader*.

Aunque este enlace ya se ha creado en la base de datos utilizando **Herramientas > Relaciones** (consulte el «Capítulo 3, Tablas»), la función utilizada por el botón *Sugerir* en este cuadro de diálogo no lo resuelve bien (sugeriría que la primera clave externa en la tabla Loan, *Media_ID*, estuviera vinculada con la *ID* de la tabla *Reader)*. El Asistente de creación de formularios resuelve esto mejor leyendo la relación de la base de datos, pero al hacerlo manualmente, debemos elegir:



El enlace elegido entre la tabla para el subformulario y la tabla para el formulario principal ahora se especifica en términos de campos de las tablas.

Para crear un control de tabla para el formulario principal, ahora debemos seleccionar el formulario principal en el *Navegador de formularios* y arrastrar el control de tablas de la barra de herramientas de controles al formulario. Si el *Asistente de control de tabla* está habilitado, mostrará los campos disponibles en el formulario principal.

Tratamos el subformulario de manera similar para insertar otro control de tabla en el subformulario. Una vez que se han agregado los controles de la tabla, debemos llevar a cabo las modificaciones ya discutidas en la sección correspondiente del formulario sencillo:

En el control de tabla del formulario principal

- Establecer el control de campo numérico (columna) *ID* a solo lectura (evitará modificar el número asignado al lector) además, eliminar los decimales (no son necesarios), limitar su valor mínimo a 1 (no necesitamos números negativos) y reducir su valor máximo (se trata de una biblioteca pequeña que no va a tener muchos usuarios).
- Reemplazar el control de campo numérico (columna *Gender_ID*) del control de tablas con un cuadro de lista y en la pestaña *datos* establecer una consulta SQL a la tabla *Gender,* dicha consulta será `SELECT"Gender","ID"FROM"Gender"`, de esta manera, en lugar de aparecer el número relacionado con el género del lector, nos mostrará realmente el género (*male* o *female*)

En el control de tabla del subformulario

- Poner una etiqueta que defina la tabla ej. «Loaned Media of the chosen Reader»
- Eliminar columnas innecesarias *Reader ID* y *Media_ID_BC*
- Establecer los controles de campo numérico de la columna *Extension* con las mismas propiedades que las de la columna *ID* del formulario principal
- Reemplazar el control de campo numérico (columna *Media_ID*) con un cuadro de lista y al igual que con el formulario principal, establecer una consulta SQL a la tabla *Media*: `SELECT"Title","ID"FROM"Media"` (para que nos muestre el título del artículo, no su número identificativo)

Para los dos controles de tabla

- Cambiar las etiquetas de algunas de las columnas, para que sean más comprensibles o estéticas al usuario. Ejemplo: *LastName* por *Last Name* y *Media_ID* por *Media* etc.

nota

Los nombres de los campos en las bases de datos están limitados a caracteres ASCII. Al diseñar un formulario, estos nombres se reflejan en las etiquetas de los controles, para hacerlos más amigables o descriptivos se pueden modificar las etiquetas de los controles que hacen referencia a esos campos. El control seguirá teniendo el nombre en ASCII, pero cambiamos la manera de presentárselos al usuario.

Ya agregado el control de barra de navegación en el formulario principal lo limitaremos a las funciones de clasificación y filtro. Dentro de las propiedades de esta barra de navegación se ocultarán los controles de *Posición*, *Navegación* y *Actuar en registro,* Ya que no son necesarios, porque están disponibles desde el mismo control de la tabla (visualización de registros, navegación de registros) o bien se realizan mediante el movimiento a través del control de la tabla (almacenamiento de datos). El formulario final podría verse como en la Figura 132.

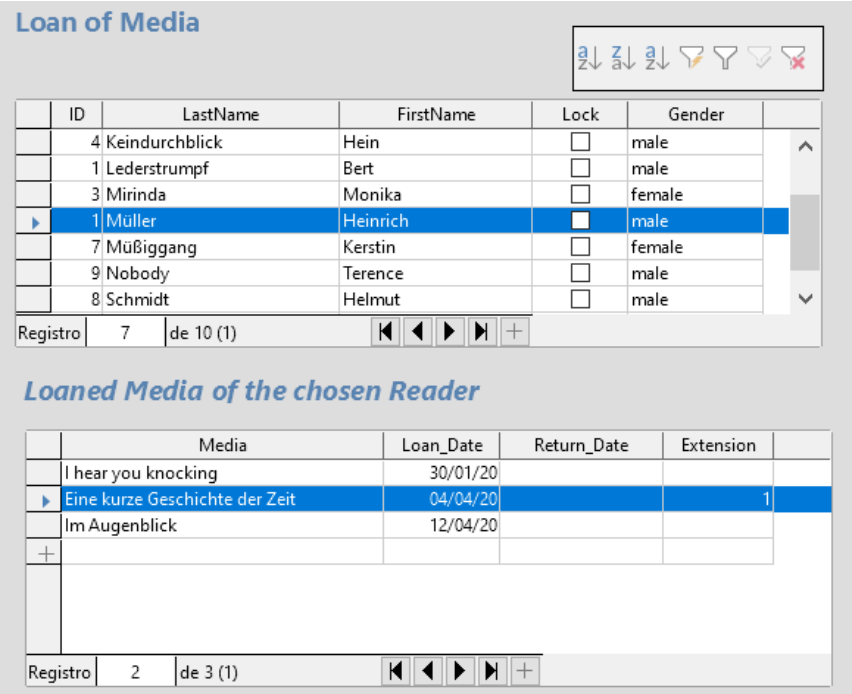


Figura 132: Formulario con un formulario principal (arriba) y un subformulario (abajo).

Si ahora se selecciona un lector en el formulario principal, el subformulario mostrará los artículos prestados a ese lector. Cuando devuelve un artículo, continúa apareciendo en el formulario hasta que se actualiza el formulario. Esto ocurre automáticamente cuando se carga otro registro en el formulario principal. Si se vuelve a seleccionar el lector original, los artículos devueltos ya no se muestran.

Esta actualización retrasada es la deseada en este caso, ya que permite inspeccionar los artículos que actualmente se encuentran en el mostrador de la biblioteca y ver de inmediato si se han registrado.

Esta estructura de formulario es significativamente más fácil de usar que la anterior con un solo formulario. Sin embargo, todavía hay detalles que se pueden mejorar:

- Los artículos prestados y las fechas de los préstamos se pueden cambiar cuando los artículos se van a prestar por más tiempo. Cambiar la fecha de los artículos prestados puede hacer que sea imposible rastrear qué artículo aún está disponible en la biblioteca y cuál está prestado. Cambiar la fecha del préstamo podría provocar errores. Los avisos de recuperación no se pueden verificar.
- Si no se selecciona un registro de lector haciendo clic en el encabezado de la fila del registro (a la izquierda), solo la pequeña flecha verde en el encabezado muestra qué registro está actualmente activo. Es posible que el registro activo se desplace fuera de la ventana de control de la tabla.
- Sería mejor que se mostrara el nombre del lector en lugar del texto «Loaned Media of the chosen Reader».
- Es posible prestar el mismo artículo dos veces sin que haya sido devuelto.
- Es posible eliminar el registro de un artículo prestado con bastante facilidad.
- Los datos se pueden cambiar o eliminar en el formulario principal. Esto puede ser útil para bibliotecas pequeñas con poco tráfico de público. Sin embargo, cuando las cosas se complican en el mostrador de préstamos, la edición de los datos del usuario no debe realizarse al mismo tiempo que la emisión de préstamos. Sería mejor si se pudieran registrar nuevos usuarios, pero los datos de los usuarios existentes no se modificaran.

Para las bibliotecas, esto se aplica igualmente a eliminaciones o cambios completos de nombre.

Para ello vamos a incluir las mejoras y modificar el diseño.

Primero, debemos mejorar la selección de lectores. Lo que debería proteger de cambios en los préstamos. Una solución simple sería no permitir ninguna modificación, excepto la entrada de nuevos registros. Esto necesita una función de búsqueda cuando un lector desea pedir prestado un artículo.

En el formulario principal sería mejor utilizar un cuadro de lista (en lugar del control de tabla) para encontrar el lector. Así eliminaríamos el problema y las operaciones de devolución se verían en un control de tabla separado (en un subformulario). Eliminaremos el control de tabla del lector, así como el control de filtro.

Para el formulario principal, se necesita una nueva tabla, en la cual el cuadro de lista pueda escribir un valor vinculado a esta tabla. La tabla necesita tener un campo entero y una clave primaria. Siempre contendrá un solo registro, por lo que la *ID* del campo de la clave principal se puede declarar de modo seguro como *Tiny Integer*. Por lo tanto, deberíamos crear una tabla *Filter*. (en la base de datos de ejemplo ya está creada con lo que usaremos esa misma).

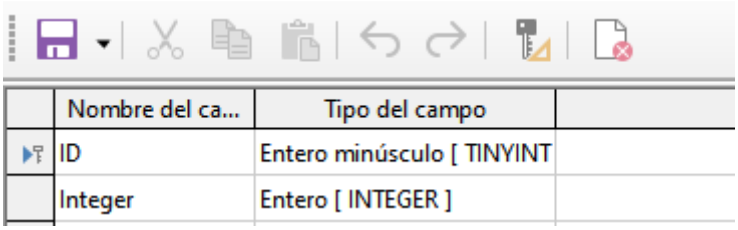


Figura 133

La tabla contiene una clave primaria «ID» con el valor «0» que corresponde al único registro el cual será leído y reescrito repetidamente por el formulario principal.

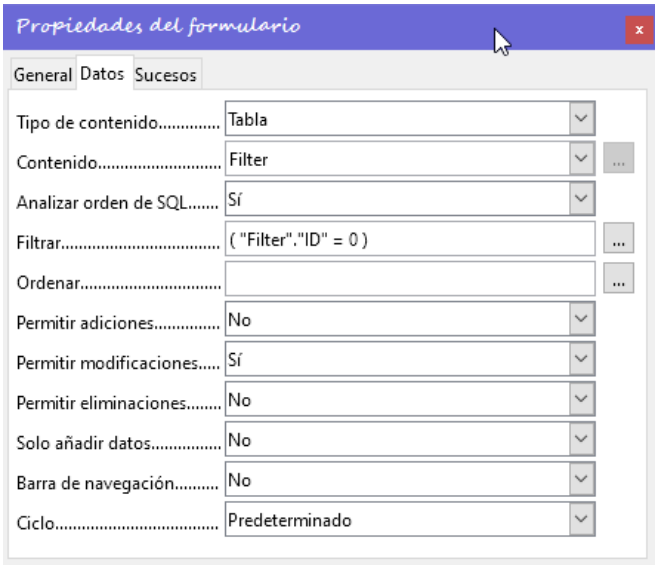


Figura 134: Formulario principal

El formulario principal por tanto, basado en la tabla *Filter,* simplemente leerá el valor de la tabla. No se agregarán datos; en el registro en curso se sobreescribe repetidamente con la *ID* del lector seleccionado en su campo *integer*. Como solo se permiten las ediciones de un solo registro desactivamos la barra de navegación del formulario porque sería superflua.

En el subformulario, mantenemos el control de tabla Las propiedades del subformulario no han cambiado desde la versión anterior. Sigue obteniendo los datos de la tabla *Loan*, con un filtro de

registros para que no aparezcan los que tienen fecha de devolución y ordenación ascendente de la fecha de préstamo.

Pero ahora cambiamos los enlaces de los campos en el subformulario, de tal manera que el valor del campo *Reader_ID* en la Tabla de *Loan* esté enlazado con *Integer* en la tabla *Filter*.

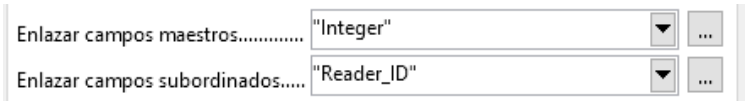


Figura 135

Antes de crear un cuadro de lista en el formulario principal, debemos desactivar los asistentes. El Asistente para cuadros de lista solo le permite crear un cuadro de lista que muestre el contenido de un solo campo;

Para tener apellido, nombre de pila y un número de lector en el área de visualización del cuadro de lista Usaremos una consulta SQL. (como en el formulario sencillo) :

```
SELECT "LastName"||', '||"FirstName"||'- No.'||"ID", "ID" FROM "Reader"
ORDER BY "LastName"
```

Independientemente de lo que nos muestre, el cuadro de lista transmitirá únicamente la ID del lector al control de tabla del subformulario para que nos muestre sus artículos en préstamo.

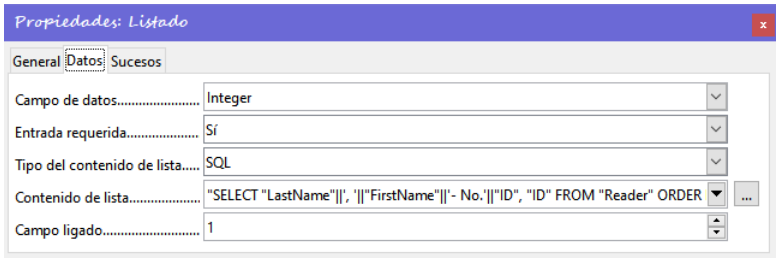


Figura 136

Al lado del cuadro de lista, se crea un botón. Este botón se debe crear en el subformulario y asume dos funciones: guardar el registro seleccionado por el control de lista en la tabla *FIlter* y actualizar la tabla del subformulario.

El botón simplemente puede etiquetarse como «OK». La acción que se le asigna en las propiedades generales del botón es *Actualizar formulario*.

El formulario ahora funciona así:

- Seleccionamos un lector del cuadro de lista y pulsamos el botón *OK* con lo que actualizamos el lector y lo pasamos al subformulario, sus préstamos aparecerán en el control de tabla.
- Podemos hacer la devolución de un artículo prestado escribiendo en la columna *Return_Date* y pulsando nuevamente el botón, se escribirá el registro en la tabla de préstamos.
- También se puede crear otro registro en dicho control de tabla eligiendo el nuevo artículo e insertando la fecha correspondiente. Debemos pulsar de nuevo el botón para registrar este nuevo préstamo.

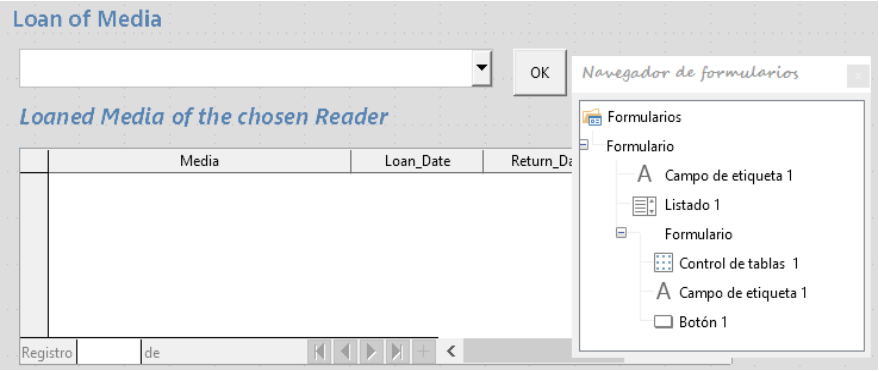


Figura 137: Formulario principal como filtro para un subformulario

El formulario principal consta solo del encabezado y el cuadro de lista; el subformulario contiene otro encabezado, el control de la tabla de la versión anterior y el botón. El formulario ahora funciona mejor porque:

- Ahora, ningún lector puede ser editado, alterado o eliminado
- Los lectores se puede encontrar más rápido seleccionándolo o escribiendo en el control de lista que utilizando un filtro.

Nuevas mejoras en el diseño

Hemos visto que el formulario funciona mejor, pero todavía podemos mejorarlo más para hacerlo más funcional. Reutilizaremos algunos controles del diseño anterior, pero usaremos más subformularios:

Vamos a crear otros dos subformularios y ambos obtendrán los datos de la tabla *Loan.* Para garantizar la funcionalidad del cuadro de lista de la figura 137, ambos subformularios deben colocarse un nivel más abajo, como subformularios del subformulario. Los datos se actualizarán jerárquicamente desde el formulario principal hacia abajo a través de los subformularios mediante los enlaces de campos maestros.

En el *Formulario Principal* dejaremos el campo de texto para el título del formulario y el cuadro de lista para encontrar al lector.

En el *Subformulario* dejaremos solo el botón, para trasladar la información del lector seleccionado al resto de formularios, actualizar el contenido de los formularios y al mismo tiempo grabar los datos introducidos, pero reutilizaremos el control de tabla que trasladaremos al *Formulario_A*.

Un nivel más abajo, colgando del *Subformulario* pondremos los nuevos subformularios a los que daremos los nombres *SubFormulario_A* y *SubFormulario_B* para que no surja confusión en ningún nivel del árbol. En estos subformularios pondremos el control de tabla anterior que trasladaremos.

Nota

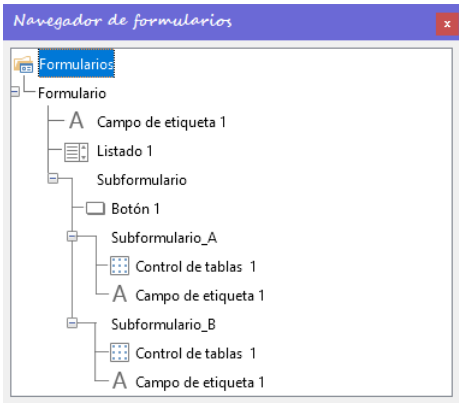
Básicamente, los nombres de los formularios y controles no tienen importancia. Pero, tienen que estar diferenciados para que las macros puedan acceder a ellos.

Cuando se crean estructuras complejas de formularios se deben aplicar nombres significativos a los formularios y sus controles para diferenciarlos. De lo contrario, encontrar el control correcto puede convertirse en un serio problema.

Para trasladar el control de tabla, en el navegador de formularios se selecciona ese control de tabla y se arrastra del Subformulario al *Subformulario_A*. Haremos una copia de este control de tabla y de la misma manera lo trasladaremos al *Subformulario_B*:

Duplicamos el control de tabla: seleccionar, copiar, quitar selección y pegar. Al duplicarlo, la copia queda en la misma posición que el original y, por lo tanto, debe arrastrarse para separarlos.

Seleccionamos esta copia del control de tablas y la arrastramos al *Subformulario_B*.



Una vez creada la nueva estructura tenemos que hacer funcionales los formularios para lo cual tenemos que modificar tanto sus parámetros como los de los controles:

El *Formulario principal* no se modificará, seguirá usando la tabla *Filter* como contenido de datos.

El *Subformulario* (figura 138) ahora va a usar la misma tabla: *Filter;* como su función es la de actualizar los datos lo vinculamos con el formulario principal enlazando los campos de la tabla *Filter* (*Integer* como C*ampo maestro* y de nuevo *Integer* como *Campo subordinado.*), el valor del campo *Filter* contiene el número identificativo del lector, cuyo valor se transmitirá al resto de subformularios.

Como en este formulario no se ingresan datos, todos los campos para datos de este formulario se establecen como *No*.

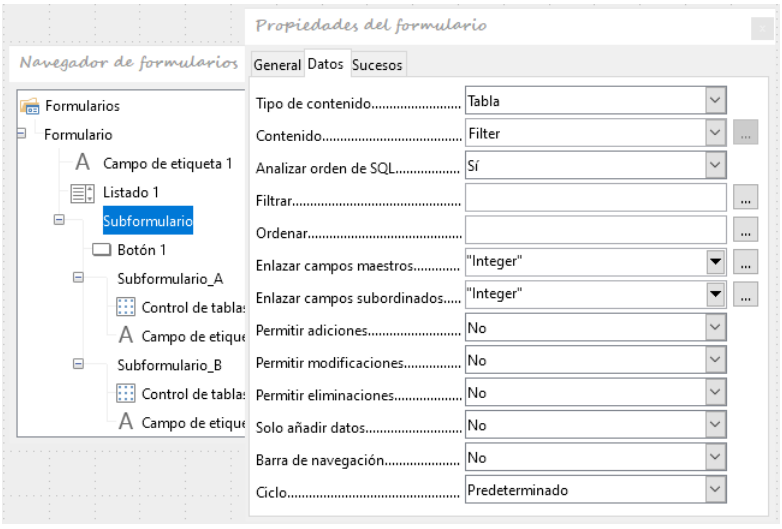


Figura 138: Propiedades del Subformulario

En el *Subformulario_A* (figura 139) debemos, enlazar los campos *integer* con *Reader_ID* para obtener los datos del lector seleccionado

Para evitar que en el control de tabla muestre ningún artículo usaremos el filtro `"Loan_Date" IS NULL"`. (Este control se mostrará vacío hasta que se despliegue con la flecha de su derecha). Según su funcionamiento solo se mostrarán los registros correspondientes al lector seleccionado y al usar el campo de fecha de préstamo vacío no se mostrará ninguno.

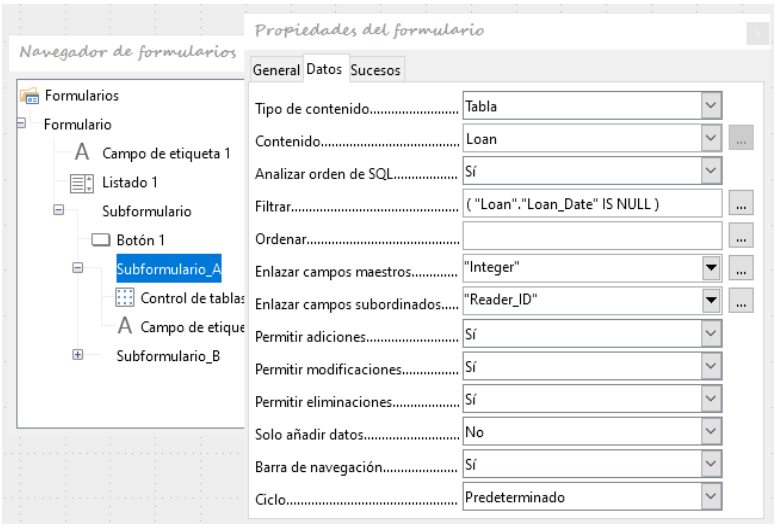


Figura 139: Propiedades del Subformulario_A

Añadiremos una etiqueta para distinguir el control de tablas de préstamos del de artículos prestados, Esta etiqueta debe mostrar «Current Loan».

En el control de tabla eliminamos todos los campos que no tienen nada que ver con el préstamo. Solo dejamos dos columnas: el artículo *Media* como campo de selección y la fecha de préstamo *Loan_Date*.

Al seleccionar un lector con el control de lista del formulario principal, tenemos que pulsar el botón *OK* para trasladar la información del lector seleccionado al resto de formularios.

En el control de tabla del Subformulario_A no se muestra ningún dato debido a las propiedades del subformulario, pero debemos programar la columna «Media» de manera que al pulsar en la flecha del listado nos muestre los artículos que podemos prestar.

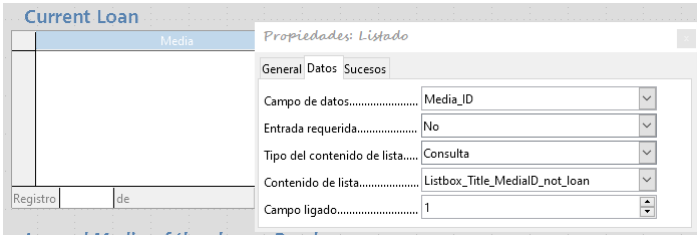
En el control de tabla del Subformulario_B al haber actualizado los formularios (pulsando el botón *OK)* ahora aparecerán los artículos que tiene prestado el lector elegido.

En la columna *Media* del control de tabla del *Subformulario_A* es necesario utilizar una consulta.

Como guardamos todos los préstamos, la tabla de préstamos almacena la fecha en que se ha prestado un libro y su devolución, así como el lector, para que la columna *Media* nos muestre los artículos que realmente tenemos en la biblioteca (que no están prestados),.

Esta consulta tiene que ser capaz de mostrar los artículos de la biblioteca que no están prestados, para lo cual tiene que filtrar las fechas de devolución, además nos tiene que facilitar el nombre del autor con nombre y apellidos, puesto que puede haber varios artículos con el mismo título pero distinto autor y a su vez varios autores con el mismo apellido. Puede encontrar más información en el «Capítulo 5, Consultas».

Como esta consulta ya está creada en la base de datos de ejemplo la vamos a asignar al control esta columna. (El nombre de la consulta es *Listbox_Title_MediaID_not_loan*).



El *Subformulario_B* (figura 140) requiere más configuraciones, también está relacionado con la tabla *Loan*. También necesitamos enlazar los campos *Integer* y *Reader_ID.* Filtraremos los datos

para una fecha de devolución vacía. (`"Loan", "Return_Date" IS NULL`). Es decir los artículos que están en posesión del lector.

Los datos se ordenan de modo que los artículos que llevan prestados más tiempo son visibles de inmediato (`"Loan_Date" ASC`).

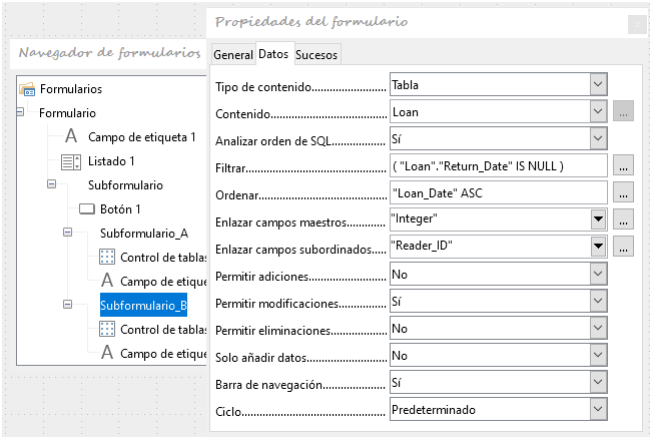


Figura 140: Propiedades del Subformulario_B

Los siguientes puntos también son importantes. Los registros antiguos se deben poder cambiar, (para hacer las devoluciones) pero no se deben agregar registros nuevos y la eliminación tampoco se debe permitir, puesto que se pretende mantener un registro de préstamos.

A pesar de estas limitaciones, aún sería posible cambiar el artículo y la fecha del préstamo desde el control de tabla, con lo que las propiedades de las columnas necesitan un ajuste: la fecha del préstamo *(Loan_Date)* y el artículo *(Media)* deben mostrarse pero protegidos de modificaciones. (solo lectura).

Mientras que para la columna de fecha (*Loan_Date)* basta con activar su propiedad de *Solo lectura*, esto no es suficiente para los cuadros de lista (columna *Media*). Esta configuración no impide que el cuadro de lista se use para realizar cambios.

Si un cuadro de lista se establece como no activado, no se puede hacer ninguna selección. Un cuadro de lista contenido en el control de la tabla (columna desplegable) se comporta como un campo de texto de solo lectura. Y esto es lo que buscamos.

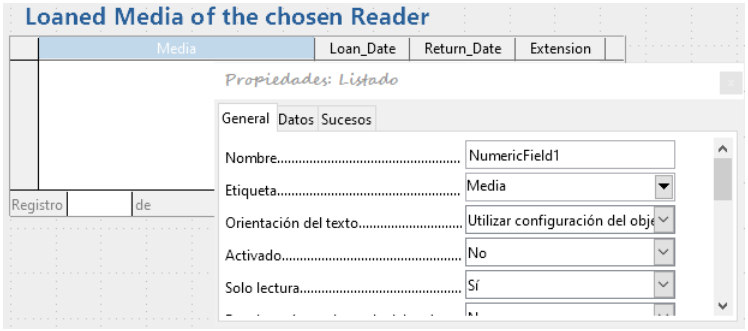


Figura 141

El formulario de préstamo de artículos ya es significativamente más útil. Cuando un lector llega al mostrador de préstamos, buscamos su nombre. Los artículos a prestar pueden seleccionarse utilizando el cuadro de lista, pero se tiene que rellenar la fecha de préstamo. La tecla *Tab* te lleva al siguiente registro.

Como el control de tabla no se actualiza continuamente, los datos de los artículos recién prestados en el *subformulario_A* permanecerán dentro de él hasta que se pulse el botón de actualización *OK*, para transferir los datos al *Subformulario_B*.

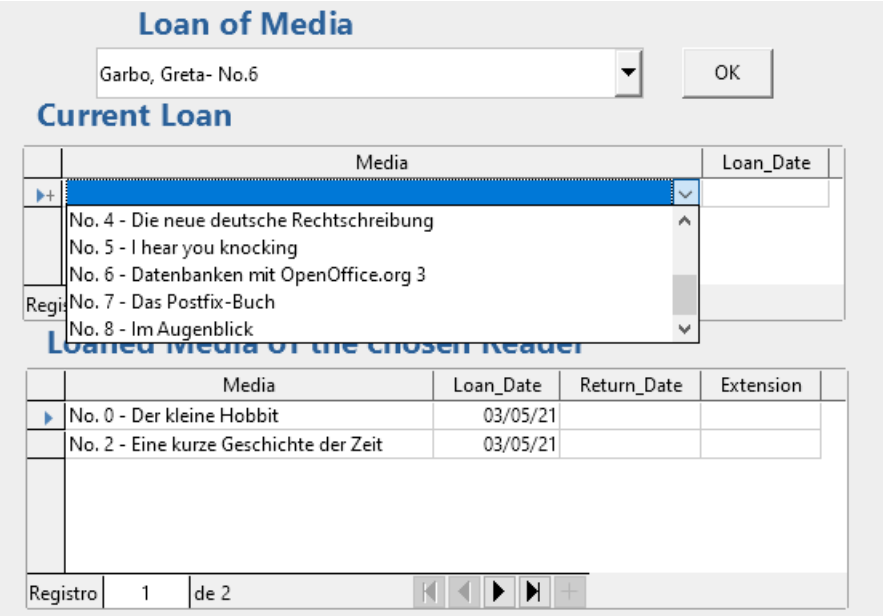


Figura 142: El campo de selección en el subformulario superior muestra solo los artículos que no están prestados.

También sería conveniente aplicar una mejora más: por el momento, se debe introducir la fecha del préstamo cada vez que se hace un préstamo. Imagine un día en la biblioteca con quizás 200 transacciones de préstamos, quizás a solo una persona se le prestan alrededor de 10 artículos de golpe. Eso requiere la misma entrada para el campo de fecha una y otra vez. Debe haber una manera de simplificar esto.

Nuestro formulario principal está vinculado a la tabla *Filter*. El formulario principal solo funciona con el único registro que tiene como clave principal el *"ID" 0*. Pero se pueden construir campos adicionales en la tabla *Filter*. En esta tabla podemos crear fácilmente un campo nuevo que almacene la fecha. Crearemos el campo *Date* y el tipo de campo *Fecha*. En la tabla *Filter* ahora hemos almacenado no solo el identificador del lector (campo Integer) sino también la fecha de préstamo (campo Date).

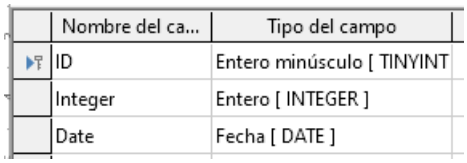


Figura 143

En el formulario principal, crearemos un control de fecha adicional, junto con una etiqueta que haga referencia a su contenido.

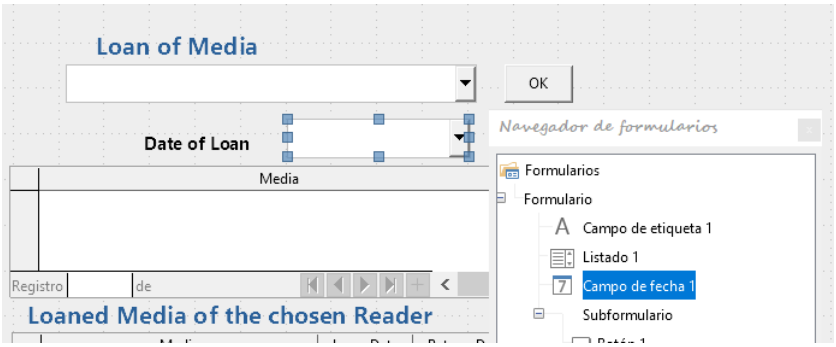


Figura 144

El valor del campo de fecha se almacena en la tabla *Filter* y se transfiere mediante los enlaces entre formularios. Por lo que debemos modificar los enlaces en todos formularios, agregando los campos maestros y subordinados correspondientes.

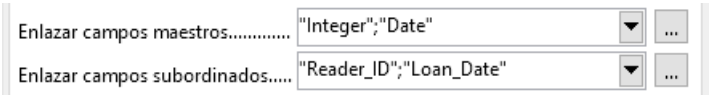


Figura 145

Los enlaces entre los formularios ahora se refieren dos campos maestros y subordinados. El campo entero está vinculado al campo *Reader_ID*. El campo *Date* está vinculado al campo *Loan_Date*. Esto garantiza que *Loan_Date* también se transfiera automáticamente de la tabla *Filter* a la tabla *Loan* cuando se realice el préstamo.

El campo de fecha ahora se elimina del control de tabla del *Subformulario_A*, con lo que este ya solo contiene un campo de búsqueda de artículos. Este sería el requisito ideal para acelerar aún más el trabajo de la biblioteca.

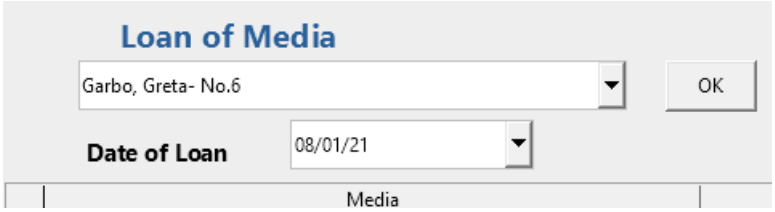


Figura 146: La fecha del préstamo se ingresa solo una vez. Cuando el lector cambia, se debe volver a introducir.

Ya que cada artículo tendrá un número de identificación impreso, entonces, ¿por qué hay que buscarlo? Simplemente puede ingresar el número directamente. O, mejor aún, podría escanearse con un lector de código de barras. Luego, los artículos se pueden prestar tan rápido como el lector pueda guardarlos en su bolso.

Esto se ilustra en la base de datos de ejemplo. El ejemplo anterior debería bastar para comprender el diseño inicial del formulario.

En la base de datos de ejemplo, *Media_without_Macros.odb*, el formulario está aún más desarrollado, las mejoras incluidas se describen brevemente a continuación.

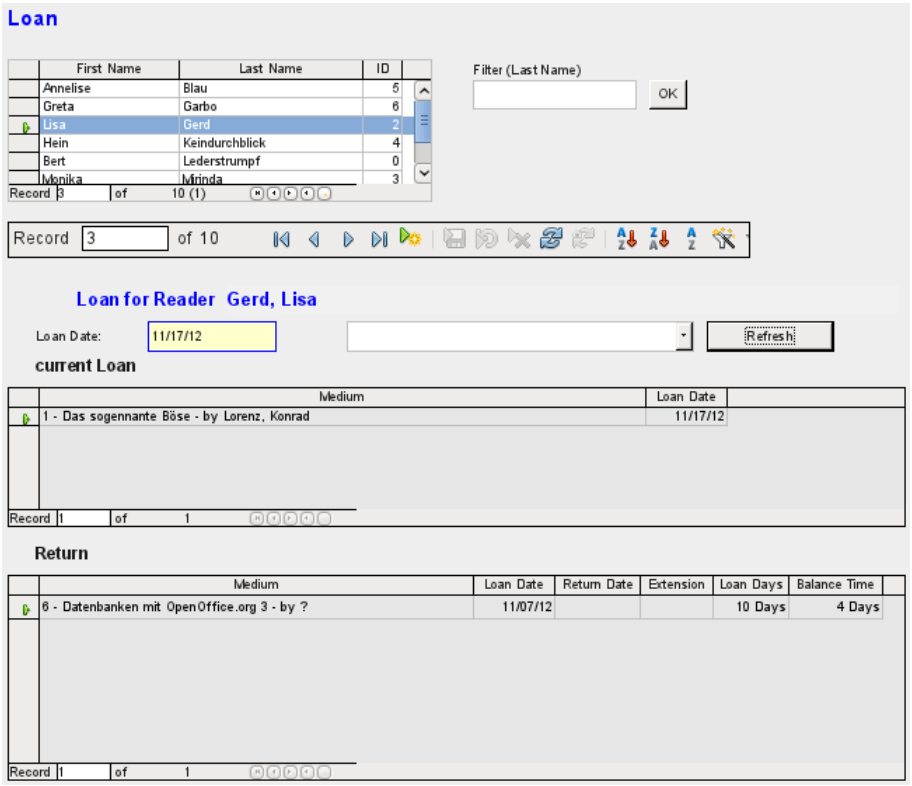


Figura 147

El formulario de Préstamo Loan tiene las siguientes propiedades:

- Los lectores se muestran en una *tabla de control* donde también puede ingresar nuevos lectores.
- Usando un filtro, vinculado a la tabla *Filter*, los nombres se pueden buscar usando su letra inicial. Por lo tanto, si ingresa A, solo se mostrarán las personas cuyo apellido comience con A. Este filtrado es independiente de mayúsculas y minúsculas.
- El subtítulo (etiqueta común para las tablas inferiores) muestra nuevamente el nombre del lector a quien se va a otorgar el préstamo y si está bloqueado.
- La fecha del préstamo se establece en la fecha actual. Esto se hace en la tabla de filtro usando SQL de tal manera que, cuando no se ingresa una fecha, el valor predeterminado que se almacenará (para los nuevos préstamos) es la fecha actual.
- Los artículos que se pueden prestar se seleccionan mediante un cuadro de lista. Cuando se presiona el botón *Actualizar*, el préstamo se transfiere al control de tabla siguiente.
- El control de la tabla del medio, solo sirve para mostrar el préstamo y la fecha real del préstamo que se va a efectuar. Aquí también es posible corregir un posible error eliminando la línea.
- En el control de la tabla inferior, como en el ejemplo anterior, no es posible modificar artículos ni fechas de préstamo. Tampoco es posible eliminar registros.
- Además de la entrada de la fecha de devolución o, si corresponde, una extensión del préstamo, esta tabla también muestra la cantidad de días para los cuales se puede prestar el artículo y cuántos días quedan del período actual del préstamo.
- Si este tiempo restante se vuelve negativo, el artículo tiene que ser devuelto inmediatamente. Se marca al prestatario como bloqueado y queda bloqueado para nuevos préstamos. El préstamo vuelve a ser posible solo cuando se devuelve el artículo. Después de la devolución, solo es necesario pulsar una vez el botón *Refresh*.

Este formulario, realizado mediante el uso de consultas, es significativamente más complejo en su estructura que la versión mostrada anteriormente. Puede obtener más información sobre lo esencial de esto en el «Capítulo 5, Consultas».

Una vista - muchos formularios

Mientras que el ejemplo para el formulario de Préstamos solo involucra entradas en una tabla (la tabla *Loan*) y además permite un ingreso simplificado en el formulario para nuevos lectores, el procedimiento de ingreso para los artículos es significativamente más extenso.

En su formulario final, para entrada de artículos (Figura 150), la tabla de artículos está rodeada por un total de ocho tablas adicionales (consulte el «Capítulo 3, Tablas»).

Las tablas *Subtitle* y *rel_Media_author* están vinculadas a subformularios del formulario *Media* a través de una relación **n:1**. Por el contrario, las tablas con una relación **1:n** con la tabla de artículos deben estar representadas por formularios que se encuentran sobre la tabla de artículos(*Media*). Como hay varias tablas de este tipo, sus valores se ingresan en el formulario principal mediante cuadros de lista.

La tabla de un formulario principal se encuentra en la tabla de un subformulario en una relación **1:n**, o en algunos casos excepcionales **1:1**. Por lo tanto, después de un uso prolongado de la base de datos, el formulario principal generalmente administra una tabla que tiene significativamente menos registros que la tabla que pertenece al subformulario.

Un subformulario no puede estar presente en varios formularios principales. Por lo tanto, no es posible crear una disposición de formularios para muchas relaciones **1:n** simultáneas en las que el subformulario tenga el mismo contenido. Cuando hay una relación **1:n** para la tabla que pertenece a un formulario, se puede usar un cuadro de lista. Aquí hay unos pocos términos disponibles de otra tabla, aquellos cuyas claves externas se pueden ingresar en la tabla del formulario principal de esta manera.

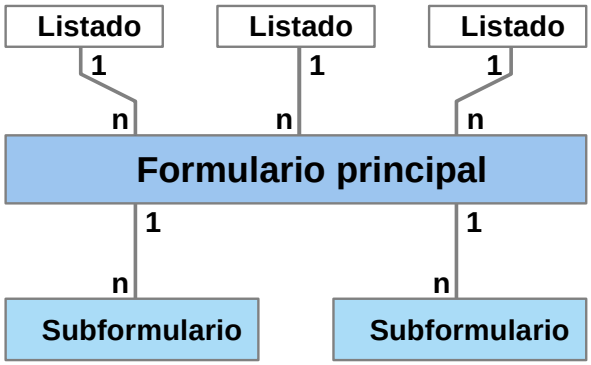


Figura 148

Usando cuadros de lista, al formulario principal basado en la tabla *Media* se le pueden asignar valores de las tablas *Category*, *Town* o *Publisher*. Los subformularios se utilizan para vincular las tablas *rel_Media_Author* y *Subtitle* con el formulario principal y, a través de él, con la tabla *Media.*

El subformulario para la tabla *rel_Media_Author* consta de dos cuadros de lista para que las claves externas de *Authors* y *Author_Add_ID* (las adiciones pueden ser, por ejemplo, editor, fotógrafo, etc.) no tengan que ingresarse directamente como números.

Para el formulario de entrada de artículos, los cuadros de lista generalmente deben llenarse gradualmente durante el proceso de entrada. Para este propósito, se incorporan formularios adicionales junto con el formulario principal. Estos son independientes del formulario principal.

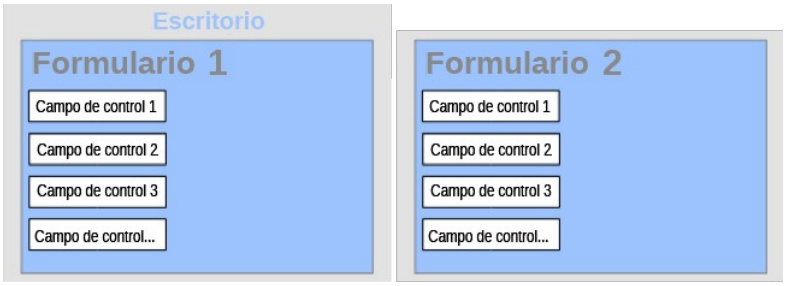


Figura 149

A continuación se muestra una vista general del formulario para entrada de artículos

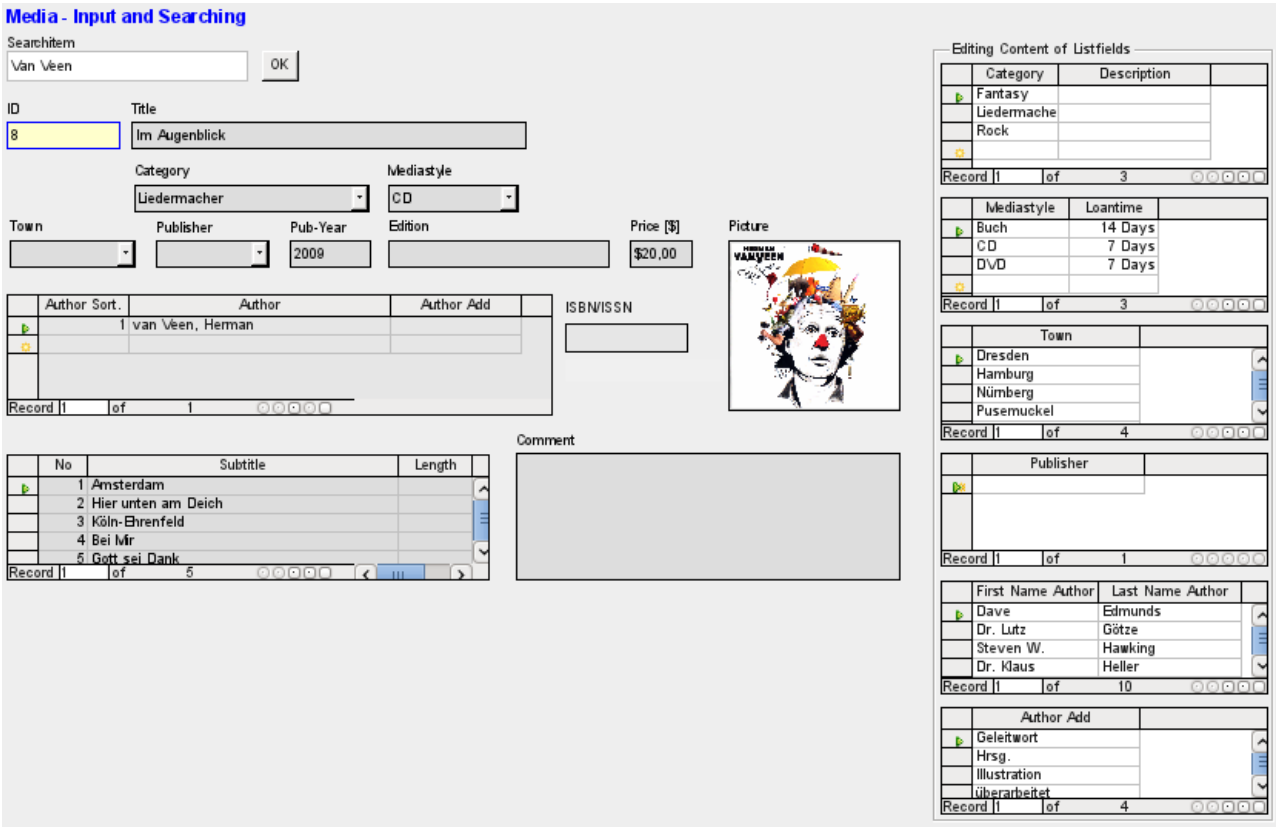


Figura 150: Formulario de Entrada de artículos

En el lado izquierdo está el formulario principal con vistas a la búsqueda y entrada de nuevos artículos. En el lado derecho hay un cuadro de grupo con la etiqueta *Editing Contents of List box*, que proporciona un área separada destinada a llenar los cuadros de lista en el formulario principal.

Estos datos estarán disponibles para los cuadros de lista del formulario principal. Con una base de datos reciente, será necesario realizar entradas en estos campos más a menudo. Sin embargo, cuantos más registro se incorporen, será necesario acceder a esta tabla con menos frecuencia.

Los siguientes controles de tabla están subordinados como formularios secundarios individuales al formulario principal, el formulario de entrada.

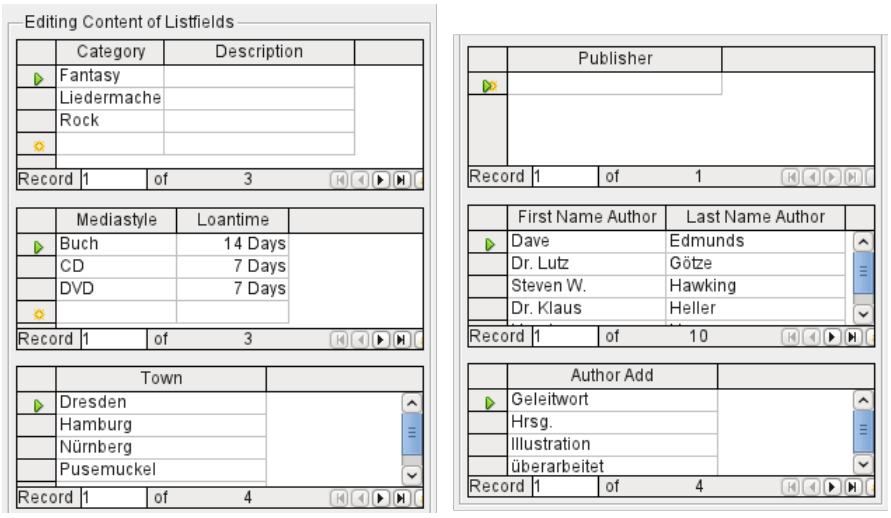


Figura 151

En cada caso se ingresan los datos completos de una tabla. En las primeras etapas, a menudo es necesario recurrir a estos formularios secundarios, ya que todavía no hay muchos autores almacenados en la tabla correspondiente.

Cuando se almacena un nuevo registro en uno de los controles de la tabla, es necesario encontrar el cuadro de lista correspondiente en el formulario principal y usar el control Actualizar (ver Barra de navegación) para leer los nuevos valores.

El navegador de formularios muestra una lista consecuentemente bastante extensa de formularios.

Figura 152

Los formularios han sido nombrados individualmente para ayudar a su reconocimiento. Solo el formulario principal todavía tiene el nombre de *MainForm* que le dio el asistente. En total hay ocho formularios paralelos. El formulario *Filter* aloja una función de búsqueda, mientras que el formulario *MainForm* es la interfaz de entrada principal. Todos los otros formularios se relacionan con uno u otro de los controles de la tabla que se muestran arriba.

Sin los controles de tablas, el formulario principal parece algo más simple:

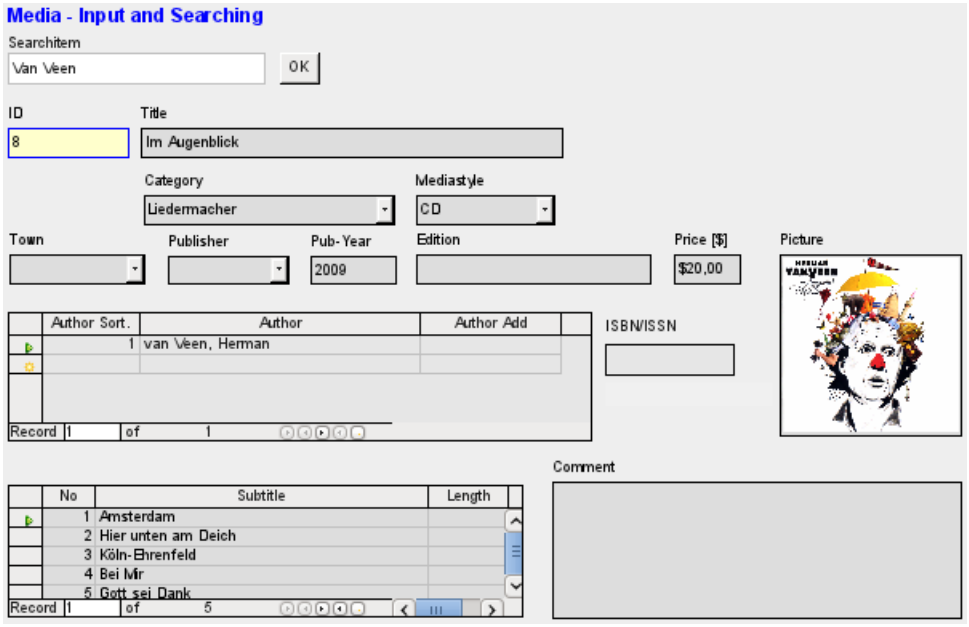


Figura 153

El campo para el término de búsqueda se encuentra en el formulario *Filter*, los dos controles de tabla (*Author* y *Subtitle*) se encuentren en el subformulario del formulario principal *Media Entry main*.

Este formulario parece mucho más complejo si se observa en el *Navegador de formularios*, pues todos los controles y etiquetas aparecen visibles. En el formulario anterior, la mayoría de los campos eran en realidad columnas dentro de los controles de la tabla y, por lo tanto, eran invisibles para el Navegador de formularios.

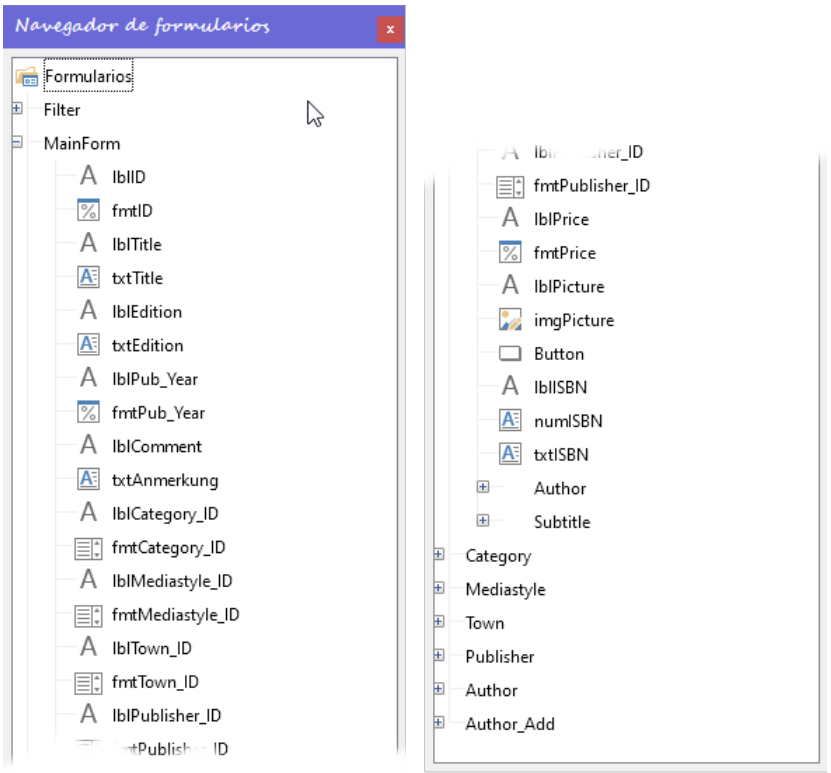


Figura 154

Lamentablemente, la secuencia dentro del *Navegador de formularios* no se puede cambiar fácilmente. Parecería más sensato colocar los subformularios Subtítulo y Autor como ramas de

MainForm al principio. Pero dentro del *Navegador de formularios*, los controles individuales y los subformularios simplemente se muestran en el orden en que fueron creados.

El Asistente de formularios proporciona los controles con abreviaturas particulares que aparecen junto a los iconos e indican qué tipo de control es este. Las etiquetas comienzan con *lbl*, los campos de texto con *txt* y así sucesivamente. En general, las etiquetas proporcionan solo información secundaria para la entrada de datos. También se pueden colocar encima de los marcos de texto para que no aparezcan en el Navegador de formularios.

La secuencia de los elementos en el navegador no tiene nada que ver con la secuencia de la tabulación. Eso está determinado por la Orden de Activación.

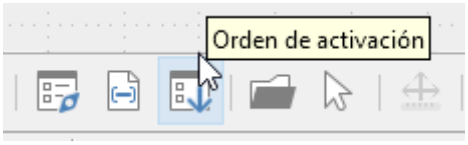


Figura 155

El diálogo *Orden de activación* para un formulario aparece cuando se selecciona un elemento del formulario y luego se hace clic en el botón *Orden de activación*. Para un formulario simple, no es necesario seguir este orden, pero donde hay muchos formularios paralelos, la función necesita saber de qué formulario se trata. De lo contrario, de manera predeterminada, aparece la secuencia del primer formulario que se muestra en el *Navegador de formularios*.

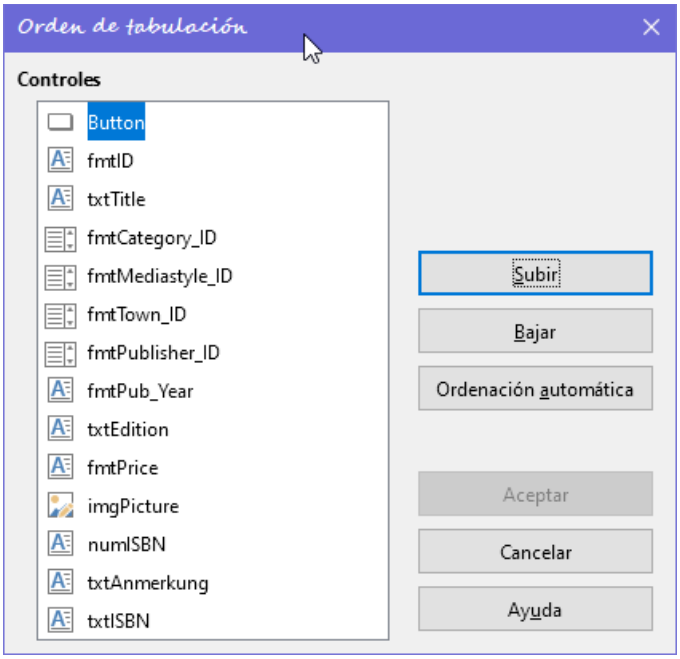


Figura 156

El orden de activación permite poner en orden todos los elementos que transmiten datos a la tabla subyacente del formulario o que pueden realizar acciones. Esto se lleva a cabo al configurar el orden de activación en las propiedades de control enumeradas en la página 135.

Tenga en cuenta que en el orden de activación, aparecen algunos controles para los que la tabulación está realmente desactivada. Aparecen en la lista, pero en realidad no se accede a ellos a través del teclado cuando se trabaja con el formulario.

En el orden de activación los controles se pueden ordenar automáticamente (orden en que están dispuestos en el formulario). Cuanto más arriba se encuentra un campo, antes aparece en la lista. Para campos a la misma altura, el campo más a la izquierda será el primero. Esta ordenación funciona sin errores solo cuando los elementos se posicionaron exactamente utilizando la cuadrícula al crear el formulario. De lo contrario, deberá ajustarlos. Simplemente seleccione un control y use *Subir* o *Bajar* para colocarlos adecuadamente en la secuencia.

Si hay un subformulario, la *Clasificación automática* salta directamente al subformulario después de completar el formulario principal. En el caso de un control de tabla, esto hace que el cursor durante la entrada del teclado quede atrapado dentro de este subformulario; solo se puede liberar usando el ratón o pulsando *Ctrl+Tab*.

La Clasificación automática funciona solo una vez para los controles de tabla. No se activará para el control de tabla de un subformulario posterior. Tampoco se tienen en cuenta los de los formularios paralelos. No se puede realizar una *Clasificación automática* retrospectiva para un subformulario con un control de tabla. El subformulario tiene que eliminarse por completo (trasladarse temporalmente a otro formulario).

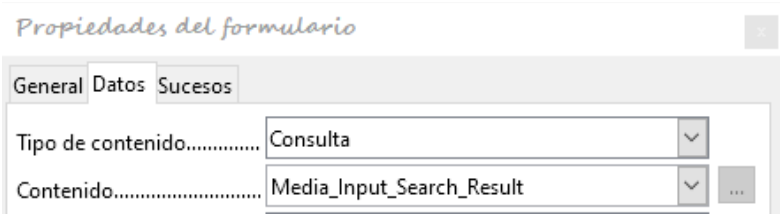


Figura 157

El sustrato de datos para el formulario de entrada de artículos no es en este caso una tabla sino una consulta. Esto es necesario, ya que el formulario debe usarse no solo para ingresar registros, sino también para buscarlos. El formulario también contiene un campo de texto que muestra, después de guardar, si el número ISBN ingresado es correcto. Esto solo es posible mediante una consulta exhaustiva. Para comprender el trasfondo de este comportamiento, es necesario comprender los fundamentos de las consultas, cubiertos en el «Capítulo 5» de esta guía.

Mensajes de error durante la entrada a formularios

A continuación se explican brevemente algunos mensajes de error que suelen ocurrir cuando se diseña un formulario por primera vez:

- Está intentando insertar nulo en una columna no anulable: columna: ID de tabla: [nombre de tabla] en la declaración …

Hay campos que no pueden estar vacíos. Si están definidos en la tabla *Entrada requerida* = Sí (NOT NULL), recibirá este mensaje de error. También si se ha declarado *Entrada requerida* en el formulario, verá un mensaje en idioma nativo con el nombre exacto del campo que debe completarse.

El mensaje anterior aparece muy a menudo cuando la clave principal, en este caso *ID*, no se ha importado al formulario. Fue diseñado como un campo que se incrementa automáticamente, pero desafortunadamente el diseñador olvidó definirlo como *campo automático*. Por lo tanto, se tiene que corregir o el campo debe aparecer en el formulario, de modo que pueda ingresar un valor.

Definir un campo retrospectivamente como campo automático a veces es problemático, si la tabla tiene relaciones con otras tablas o cuando se accede a la tabla usando una vista. Todos los enlaces deben eliminarse para que el campo de clave principal de la tabla se pueda editar. El contenido del comando SQL que se utilizó para crear la vista se puede almacenar temporalmente en una consulta.

- Infracción de restricción de integridad: Sin tabla padre SYS_FK_95: [nombre de tabla] en la declaración …

Aquí tenemos un enlace entre la tabla subyacente al formulario y otra tabla. Se supone que esta tabla debe proporcionar su clave principal para su uso en un campo de clave externa. Normalmente esto se hace usando un cuadro de lista que realiza la consulta correcta para recuperar la clave. Si no está utilizando un cuadro de lista o si el cuadro de lista se construyó incorrectamente, el campo de clave externa puede adquirir un valor incorrecto que en realidad no está presente en la tabla de origen como clave principal. Eso es lo que se entiende por «violación

de restricción de integridad». «Sin padre SYS_FK_95» significa que en la segunda tabla, la fuente «Padre», el valor de índice correspondiente con el nombre «SYS_FK_95» no está presente.

- Errores al ingresar nuevos registros
- Errores al ejecutar funciones

Supongamos que un formulario proporciona datos a un subformulario. LibreOffice Base no ve esto como un cambio en un valor de campo. La interfaz gráfica de usuario ofrece guardar el valor, pero el registro aparece vacío. La solución es incluir un campo simple, por ejemplo, un campo sí / no, en la tabla subyacente. Ahora, antes de guardar, elimine este campo y el registro se puede guardar sin ningún problema.

Los valores transmitidos a través de otros formularios aparentemente solo se ingresan en los campos correspondientes cuando hay al menos una acción adicional en el formulario.

Buscar y filtrar formularios usando la barra de navegación

La barra de navegación del formulario ofrece varias posibilidades para buscar y filtrar. Los elementos individuales en la barra de navegación ya se han enumerado anteriormente.

Búsqueda de registros usando parámetros

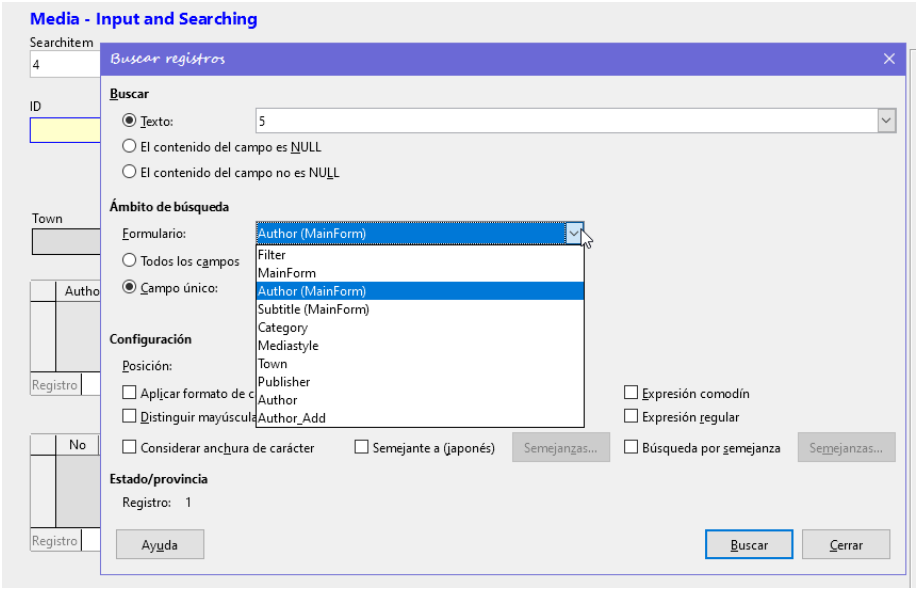


Figura 158

Una búsqueda simple de registros con parámetros se describe en el «Capítulo 3, Tablas». Se lanza haciendo clic en el icono de búsqueda del navegador de formularios y ofrece una cantidad considerable de configuraciones. Una característica de este tipo de búsqueda es que básicamente todos los registros se seleccionan en lugar de limitar los registros que se muestran a los que contienen los parámetros de búsqueda.

Las búsquedas de registros en formularios le permiten buscar tanto el formulario como cualquier subformulario, pero no es posible buscar todos los formularios a la vez.

La ilustración muestra el acceso al subformulario *Auth*or en el formulario principal *MainForm*. Este es un campo de lista que contiene los nombres de los autores. Sin embargo, lo que se busca no es el texto en el campo de lista, sino los valores que se ingresan como claves externas cuando se usa el campo.

Lamentablemente, este tipo de búsqueda tiene las siguientes deficiencias:

La búsqueda de parámetros es demasiado complicada para el uso normal.

- La función de búsqueda en sí es muy lenta, ya que no utiliza las funciones de consulta de la base de datos.
- La búsqueda solo funciona para un formulario a la vez, aunque para todos los campos. Los subformularios deben buscarse por separado.
- Estamos buscando un subformulario que pertenece al formulario principal actual. Las entradas en otros subformularios no serán reveladas. Entonces, por ejemplo, no es posible encontrar un subtítulo de ningún artículo que no sea el que se muestra actualmente.
- La búsqueda se aplica a las funciones de campo de lista en función de los campos clave (claves externas) almacenados en la tabla. Los datos mostrados no se pueden usar.

Filtrar con el Filtro automático

El *Filtro automático* se puede seleccionar directamente en la barra de navegación. Haga clic en un campo en el formulario principal y se filtrará en el contenido de ese campo. Este filtro también funciona con campos de lista y tiene una ventaja obvia sobre la entrada de claves externas a mano.

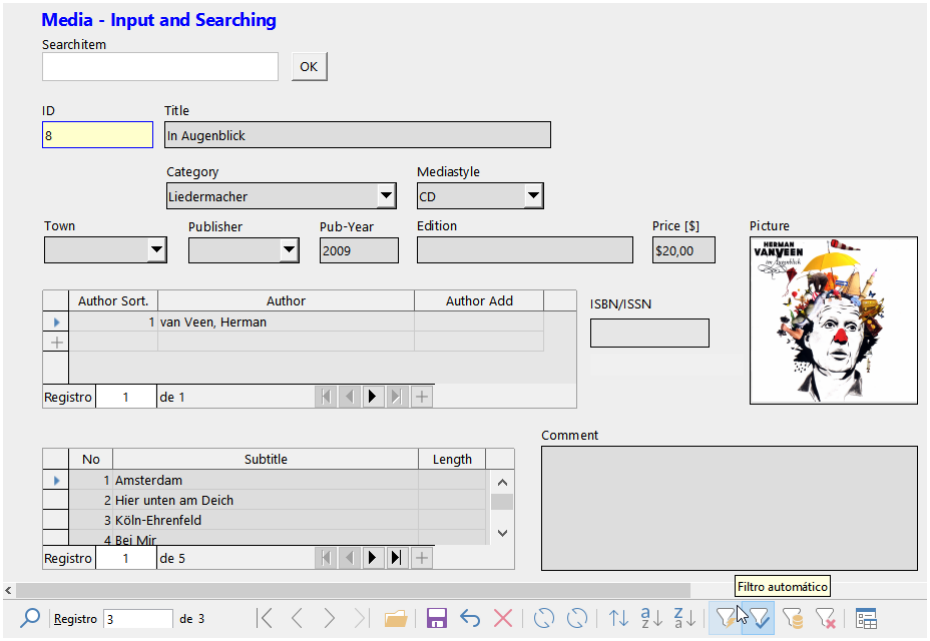


Figura 159

En el ejemplo que se muestra arriba, se recupera un registro que tiene «CD» en el campo *Mediastyle*. Luego se activa el botón del *Filtro automático*. De los 9 registros originales en la tabla de artículos, ahora solo se muestran 3 registros en el contador de registros.

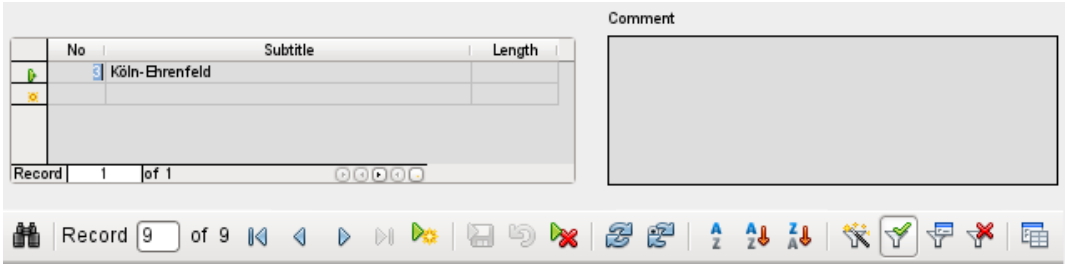


Figura 160

Aunque solo dos registros tienen un «3» en el campo *No.*, los formularios principales para todos los demás registros continúan mostrándose. En el subformulario, solo se muestran los registros que coinciden con «3».

Sin embargo, usar el filtro automático no puede resolver los siguientes problemas:

- Si busca en un subformulario, solo se mostrará el valor especificado, pero el formulario principal se muestra para todos los registros, incluso si no contienen ese valor. Entonces, si, por ejemplo, está buscando un artículo con un segundo autor, seguirá viendo todos los artículos que tienen un solo autor. En estos registros, el campo de autor estará vacío.
- El valor utilizado para el filtro debe ser específico. No hay posibilidad de una búsqueda de similitud para encontrar materiales relacionados.

Filtrar con el filtro de formas

El filtro basado en formularios, (*filtro de formas*) en lugar de solo un botón de filtro, ofrece un formulario para insertar valores reales que se utilizarán para el filtrado. Aquí puede filtrar varios valores al mismo tiempo. Estos valores pueden combinarse como una condición conjunta (`AND`) o alternativa (`OR`).

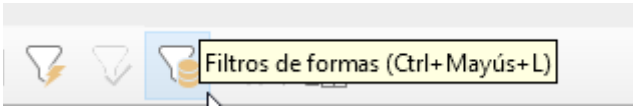


Figura 161

Los valores de filtro ingresados no necesitan ser únicos. Aquí, por lo tanto, puede formular condiciones como las descritas en el filtrado de tablas.

Por lo tanto, la expresión «`LIKE '% Auge%'»` en el campo de título de artículos nos devolverá todos los registros para los cuales el título contiene la palabra «auge»

El filtrado de los campos del cuadro de lista se lleva a cabo mediante la selección directa del contenido del campo de la lista que se muestra. No necesita ingresar las claves externas subyacentes.

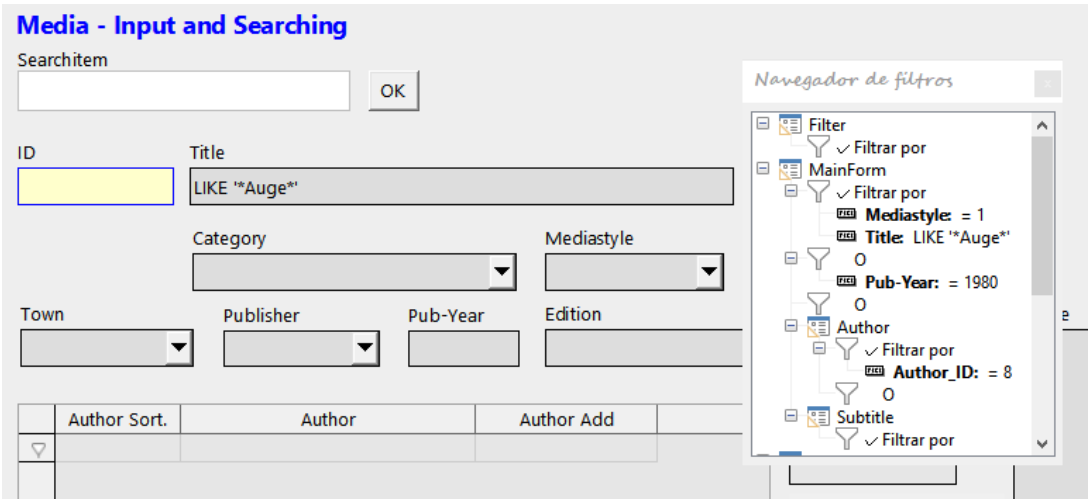


Figura 162

Aquí buscamos todos los registros con el estilo de artículos «CD» (correspondiente a un valor de clave externa de «1») y también tenemos en algún lugar del título la secuencia de caracteres. Atención**,** cuando ingresa `LIKE '*Auge*'`, la interfaz reemplaza los símbolos comodín SQL habituales «%» con un asterisco, un símbolo más familiar para la mayoría de los usuarios.

También agregamos la condición de que se muestren todos los registros para los cuales el año de publicación es «1980».

El autor «Edmunds, Dave» se selecciona del cuadro de lista en el subformulario. El valor de la clave externa correspondiente es «8».

Durante la entrada, el filtro basado en formularios muestra solo los campos que muestran una marca antes de `OR`. De este modo, se pueden establecer varias condiciones para el campo.

Desafortunadamente, hace que el filtro suprima la visualización del contenido del campo de lista de *Mediastyle* y si embargo este funciona en el control de tabla del subformulario.

Durante la creación del filtro, no se ve la barra de navegación; en su lugar, se muestran las opciones para el filtro basado en formularios. Ajuste El filtro de acuerdo sus necesidades, inserte los valores para el filtro y luego simplemente cierre el diálogo.

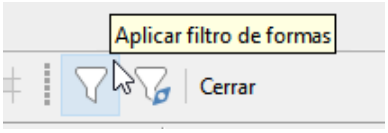


Figura 163

El filtro se aplica y el formulario se filtra de acuerdo con sus especificaciones.

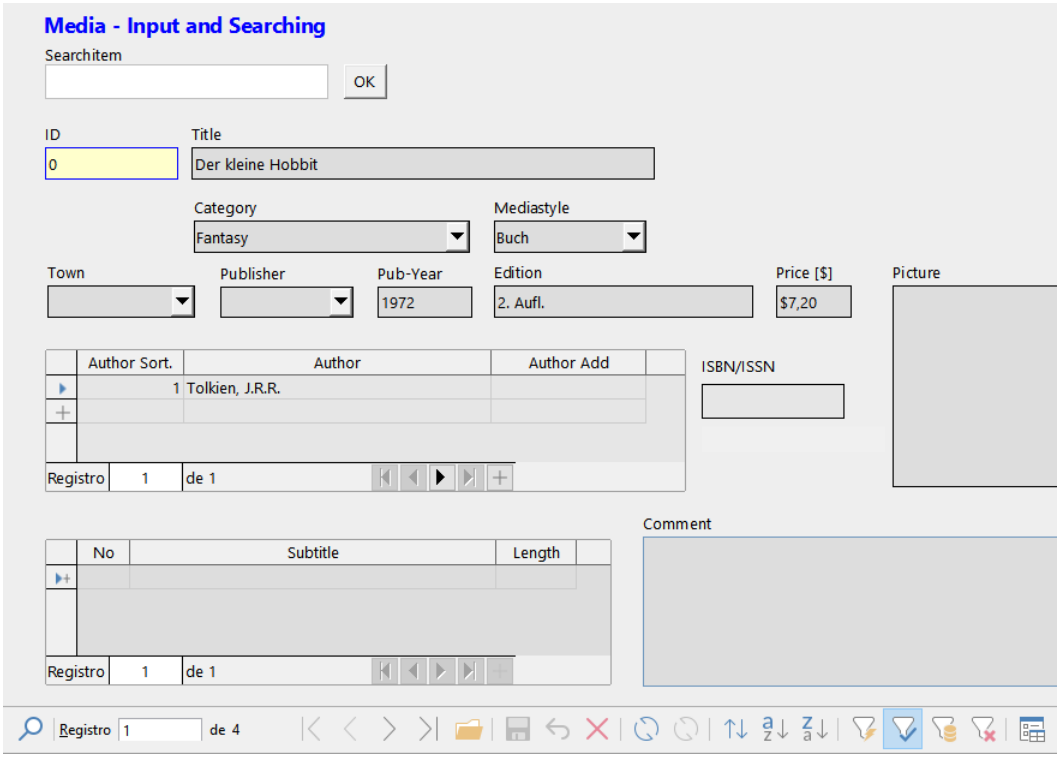


Figura 164

Cuando el filtrado está activo, el número de registros se limita a los que coinciden. En este caso hay cinco coincidencias. En la ilustración, ni el título ni el tipo de artículo coinciden; la condición de que el año de publicación sea inferior a 1980 proporciona la coincidencia.

El subformulario muestra algo interesante. Como este también se debe filtrar, no se muestra el autor del libro «Der Kleine Hobbit». El valor del filtro está vinculado al autor Dave Edmunds. Esto muestra nuevamente la lógica del filtrado: utiliza las propiedades de filtro del formulario actual. Los subformularios se filtran por separado y un filtro aplicado al subformulario no afecta al formulario principal y viceversa.

Este filtro, bastante efectivo, tiene no obstante las siguientes desventajas:

- El filtrado nuevamente funciona solo en el formulario principal. En el subformulario, como en el caso del *Filtro automático*, no se muestran los valores que no coinciden. Esto no tiene ningún efecto en la visualización del formulario principal. Por tanto, en el formulario principal, se muestra el contenido que no puede generar ningún contenido filtrado para el subformulario.
- Necesita un conocimiento detallado de cómo establecer condiciones si está buscando algo más complicado que campos completos. La mayoría de los usuarios, incluso si están acostumbrados a usar motores de búsqueda, carecen de las habilidades necesarias.

Filtrar con el filtro estándar

Se puede acceder al *filtro estándar* si se usa el botón para la visualización de *origen de datos como tabla*. Los procedimientos son los mismos que cuando se usa el filtro estándar en una tabla.

Figura 165:Activar el filtro predeterminado con origen de datos como tabla

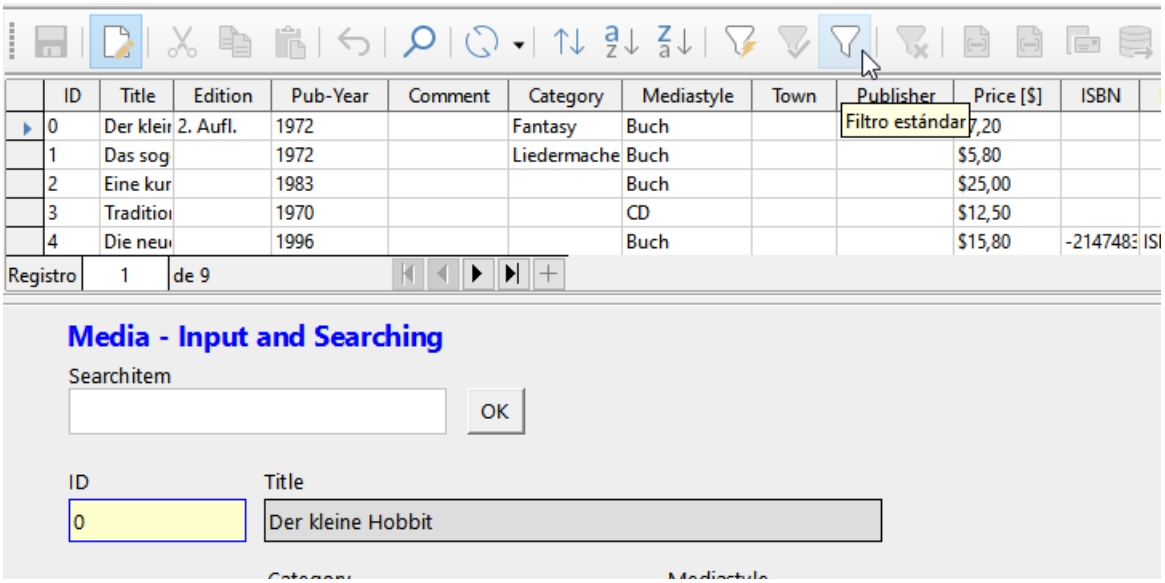


Figura 166

A la derecha del navegador de formularios, puede ver un el botón *origen de datos como tabla*. En la vista de tabla, está disponible el filtro estándar. A diferencia de la vista de la tabla real de artículos, se muestran las claves externas con el contenido que le corresponde (no con sus valores reales). Un clic en el campo *Mediastyle* muestra que esta es la fuente del campo de lista que aparece en el formulario.

Nota

Lamentablemente, la vista anterior muestra un error que se remonta al comienzo de Base. El número *ISBN* de 13 dígitos no está formateado correctamente. El campo correspondiente se toma en lugar del número mínimo posible de dígitos. Si un número tiene más de 9 dígitos, tenga cuidado: la edición puede conducir a la pérdida de datos (error 82411). Este error también aparece en los controles de la tabla. Se muestran correctamente si usa un campo formateado en lugar de un campo numérico.

La vista de tabla de un formulario tiene las siguientes desventajas para este método de búsqueda:

- La búsqueda solo funciona dentro de una tabla o consulta que subyace al formulario y no en un subformulario que le pertenece.
- En la vista de datos, se muestran los campos de la lista, pero no puede usarlos para filtrar. Una vez más, necesita conocer los valores reales de clave externa para esos campos. Así, por ejemplo, si el tipo de artículo es «Book» y este tiene el valor de clave principal «1», entonces la tabla de artículos tendrá un «1» almacenado en ese campo. Debe buscar este valor, aunque la vista en pantalla muestre el valor «Book».

Resumen

Las funciones de búsqueda y filtro son de uso limitado en formularios. Una búsqueda es demasiado lenta y no limita el número total de registros a los que realmente coinciden.

Los filtros solo funcionan dentro de un formulario a la vez. Si desea emplear un filtro en un subformulario, solo el subformulario se ve afectado. Simplemente no puede obtener el número correcto de registros coincidentes.

Aunque en los formularios de la base de datos de ejemplo, se ha integrado una función de búsqueda fácil dentro del formulario, tal procedimiento debe elaborarse a mano y requiere cierto conocimiento de SQL.

Registro de entrada y navegación

Si un formulario ha sido diseñado de manera tal que al abrirlo se activa el foco de control, el cursor aparecerá en el primer campo de entrada. En lugar de usar el ratón para moverse de un campo a otro, se puede usar la tecla de tabulación para pasar al siguiente campo de la secuencia.

A veces, un formulario le permite saltar a otro campo, ya sea con el ratón o con un atajo de teclas.

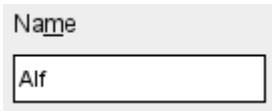


Figura 167

En un control de cuadro de texto, al usar la propiedad*:* **Cuadro de texto > Campo de etiqueta,** se le asigna un control de etiqueta. En la etiqueta de este campo de etiqueta cambie de «Name» a «Na~me». En el formulario, la letra "m" de la etiqueta ahora se muestra subrayada.

Ahora puede saltar a este campo de texto instantáneamente utilizando el atajo *Alt+m*. Esto solo funciona cuando el cursor está en un campo del formulario.

En principio, cualquier letra se puede usar como un atajo, ya que el salto se realiza completamente dentro del formulario. Sin embargo, si el cursor no está inicialmente en un control de formulario, los accesos directos activan la interfaz de usuario de LibreOffice. Entonces, por ejemplo, un acceso directo «a» no saltaría a un control con ese atajo, sino que abriría el menú *Tabla*.

Desafortunadamente, los saltos por este método solo funcionan dentro de un formulario sencillo, no hay estructuras que involucren a subformularios. No es posible saltar de un subformulario al formulario principal.

Cuando el cursor está en un control de tabla, debería saltar al utilizar *Ctrl+Tab*. No siempre esta combinación de teclas tiene el efecto deseado. Así que aquí hay una alternativa:[5]

Se crea un botón en el subformulario. Su título es «~ Nuevo». El carácter tilde se usa, como se describió anteriormente, para que *Alt+N* se pueda usar para activar el botón.

En la información adicional para el botón, se nombra el control del formulario principal al que se debe transferir el cursor para su activación.

Al botón se le asigna la siguiente macro usando **Propiedades > Eventos > Ejecutar acción**:

```
SUB JumpToMainform(oEvent AS OBJECT)
  DIM oField AS OBJECT
  DIM oForm AS OBJECT
   oDoc AS OBJECT
  DIM oController AS OBJECT
```

5 Vea también la base de datos de ejemplo: `Example_cursorjump_subform_mainform.odb`

```
  DIM oView AS OBJECT
  DIM stShortcut AS STRING
  oField = oEvent.Source.Model
  stShortcut = Mid(oField.Label, InStr(oField.Label, "~") + 1, 1)
  oForm = oField.Parent.Parent

  SELECT CASE stShortcut
    CASE "n"
      oForm.MoveToInsertRow()
    CASE "a"
      IF oForm.isLast() THEN
        oForm.MoveToInsertRow()
      ELSE
        oForm.Next()
      END IF
  END SELECT
  oDoc = thisComponent
  oController = oDoc.getCurrentController()
  oView = Controller.getControl(oForm.getByname(oField.Tag))
  oView.setFocus
END SUB
```

La macro se puede usar para volver al formulario principal. También se puede usar con la configuración anterior para crear un nuevo registro de inmediato o, usando otro botón de la barra de navegación para *Siguiente registro*, (para saltar al siguiente registro). Si el cursor en el formulario principal está en el último registro, el salto conducirá a un nuevo registro. Para información adicional sobre macros, vea el «Capítulo 9».

Saltar a un botón requiere que el botón esté visible. Sin embargo, se puede hacer lo bastante pequeño para que pase desapercibido o incluso 0 cm. Si el ancho y la altura no están definidos, puede que el botón aparezca como una línea que se extiende por toda la pantalla.

Imprimir desde formularios

Los formularios se pueden construir de manera que sea posible una impresión directa del formulario. Para obtener resultados adecuados, debe tener cuidado de no colocar elementos fuera del área imprimible. Elija **Ver > Normal** para configurar las propiedades de la página. No se recomienda el uso de un fondo de color. Cualquier elemento individual se puede excluir de la impresión cambiando esta propiedad en sus propiedades: **Propiedades > General > Imprimible**

Si la base de datos está registrada en LibreOffice (usando **Herramientas > Opciones > LibreOffice Base > Base de datos** o directamente como parte del proceso de creación), el formulario se puede usar para combinar correspondencia. El formulario se abre para su edición. Las fuentes de datos se hacen accesibles usando **Ver > Fuentes de datos** o presionando *F4*. Los campos de la base de datos ahora se pueden arrastrar al formulario por su encabezado de tabla. Luego se guarda el formulario.

Si se abre el mismo formulario para la entrada de datos, LibreOffice reconoce que estos campos son para una combinación de correspondencia y le preguntará si desea imprimir las cartas.

Los detalles para hacer una combinación de correspondencia se encuentran en el «Capítulo 7, Vinculación a bases de datos» de esta guía.

Figura 168

LibreOffice

Guía de Base

Capítulo 5
Consultas

Información general sobre consultas

Las consultas a una base de datos son la herramienta más poderosa que tenemos para usar bases de datos de manera práctica. Pueden reunir datos de diferentes tablas, calcular resultados cuando sea necesario y filtrar rápidamente un registro específico de una masa de datos. Las grandes bases de datos de Internet que las personas usan todos los días existen principalmente para ofrecer un resultado rápido y práctico para el usuario a partir de una gran cantidad de información mediante una selección cuidadosa de palabras clave, incluidos los anuncios relacionados con la búsqueda que alientan a las personas a realizar compras.

Crear consultas

Las consultas se pueden crear tanto en la Interfaz como directamente como código SQL. En ambos casos, se abre una ventana, donde puede crear una consulta y también corregirla si es necesario.

Crear consultas utilizando el diálogo Diseño de consulta

La creación de consultas con el asistente se describe brevemente en el «Capítulo 8, Primeros pasos con Base» de la *Guía de primeros pasos*. Aquí explicaremos la creación directa de consultas en la Vista de diseño.

En la ventana principal de la base de datos, haga clic en el icono *Consultas* en la sección *Bases de datos*, luego en la sección *Tareas*, haga clic en *Crear consulta en modo de diseño*. Aparecerá la ventana de diseño con dos áreas para la creación en vista de diseño de la consulta y sobre ella un diálogo para agregar tablas o consultas.

La ventana permite combinar varias tablas (y también vistas y consultas). Para agregar una tabla seleccione la opción *Tablas* en el diálogo superpuesto, seleccione una tabla, y luego haga clic en el botón *Añadir*. O haga doble clic en el nombre de la tabla. Cualquiera de los métodos agrega la tabla al área gráfica del diálogo *Diseño de consulta* (figura 169).

Cuando se hayan seleccionado todas las tablas o consultas necesarias, haga clic en el botón *Cerrar*. Se pueden agregar tablas y consultas adicionales más adelante si es necesario. Sin embargo, no se puede crear una consulta sin al menos una tabla, por lo que se tiene que hacer una selección al principio.

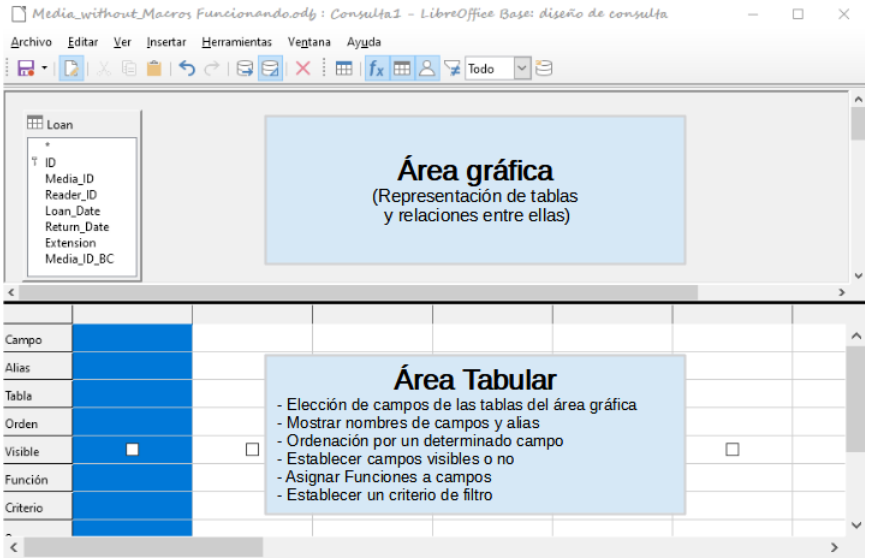
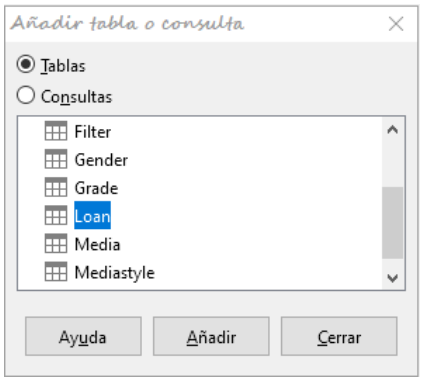


Figura 169: Áreas de Diseño de consulta

La figura 169 muestra las divisiones básicas del diálogo *Diseño de consulta*: El *área gráfica* muestra las tablas que se vincularán a la consulta. También se pueden mostrar las relaciones

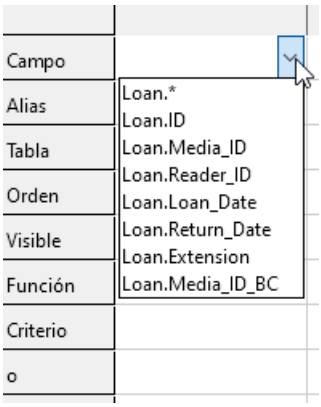
entre sí en relación con la consulta. El *área tabular* es para la selección de campos para mostrar o para establecer condiciones relacionadas con estos campos.

Como nuestro formulario sencillo se refiere a la tabla *Loan*, primero explicaremos la creación de una consulta usando esta tabla.



En el cuadro de diálogo seleccione Tablas y, de las disponibles, seleccione la tabla *Loan* y haga clic en el botón *Añadir*, luego en el botón *Cerrar.*

En el *área tabular* haga clic en *Campo*, en la primera columna, para visualizar una flecha hacia abajo. Haga clic en esta flecha para abrir la lista desplegable de campos disponibles. El formato es `[Nombre_de_tabla.Nombre_de_campo]`, razón por la cual todos los nombres de campo comienzan aquí con la palabra *Loan* y un punto.

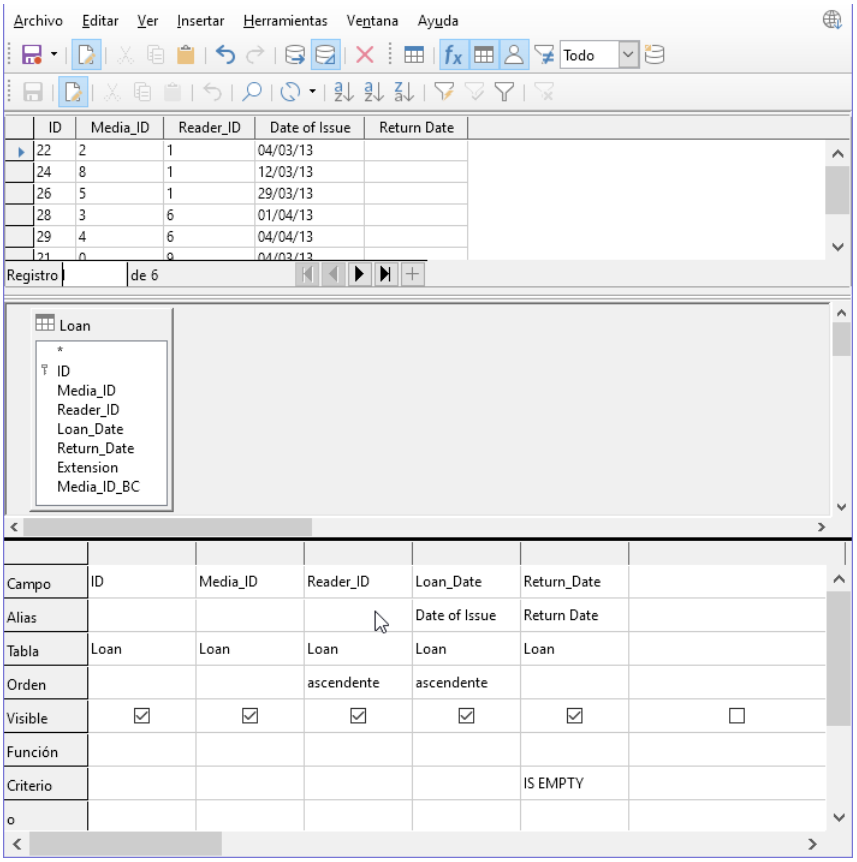


La selección del campo, designada como *Loan.**, tiene un significado especial. Un clic en esta opción agregará todos los campos de la tabla subyacente a la consulta. Cuando utiliza esta designación de campo con el comodín «*», la apariencia de la consulta se vuelve indistinguible de la tabla que se consulta.

Consejo

Para una transferencia rápida de los campos de una tabla al área tabular sitúese en la vista de tabla en el área gráfica y haga doble clic en cada campo, esta operación inserta ese campo en el área tabular en la siguiente posición libre.

Para la consulta que estamos creando, seleccione los primeros cinco campos de la tabla Loan, uno a continuación del otro. Vaya a la columna *Loan_Date* y en la fila criterio escriba `IS EMPTY`. De esta manera habrá creado su primera consulta consistente en que se muestren las filas de la tabla *Loan* exceptuando las que tengan algo escrito en su correspondiente columna.



Las consultas en modo de diseño siempre se pueden probar. Para comprobar el resultado de una consulta debe hacer clic en el botón *Ejecutar consulta* de la barra de herramientas. Esto hace que aparezca una nueva área con el resultado de la consulta en vista tabular de los datos resultantes sobre el área gráfica.

Una ejecución de prueba de una consulta siempre es útil antes de guardarla, para aclarar si la consulta realmente logra su objetivo. A menudo, un error lógico impide que una consulta obtenga datos. En otros casos, puede suceder que se muestren precisamente esos registros que desea excluir.

En principio, una consulta que produce un mensaje de error en la base de datos subyacente no se puede guardar hasta que se corrija el error.

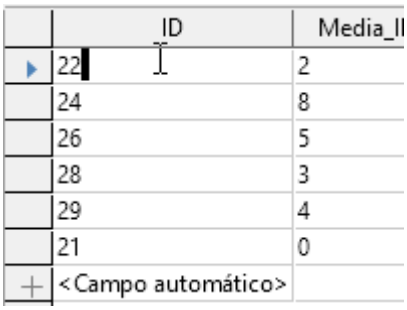


Figura 170: Consulta editable

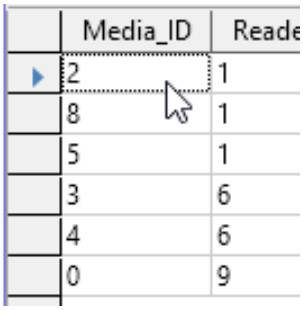


Figura 171: Consulta no editable

En el resultado de una prueba, preste especial atención a la primera columna de la tabla resultante de la consulta. El marcador de registro activo (flecha) siempre aparece en el lado izquierdo de la tabla, en este caso apuntando al primer registro como el registro activo.

Mientras que el texto del primer campo del primer registro en la figura 170 está resaltado, el campo correspondiente en la figura 171 muestra solo un borde discontinuo. El resaltado indica que este campo se puede modificar, es decir, que los registros se pueden editar. El borde discontinuo indica que este campo no se puede modificar.

La figura 170 también contiene una línea adicional para la entrada de un nuevo registro, con el campo *ID* ya marcado como <*Campo automático*>. Esto también muestra que se pueden agregar nuevas entradas.

Consejo

Una regla básica es que si la clave principal (de la tabla consultada) no está incluida en la consulta no son posibles nuevas entradas.

Campo	ID	Media_ID	Reader_ID	Loan_Date	Return_Date
Alias				Date of Issue	Return Date

Los campos *Loan_Date* y *Return_Date* tienen alias. Esto no hace que se les cambie el nombre a los campos de la tabla, sino que aparezcan estos otros nombres en la consulta del usuario.

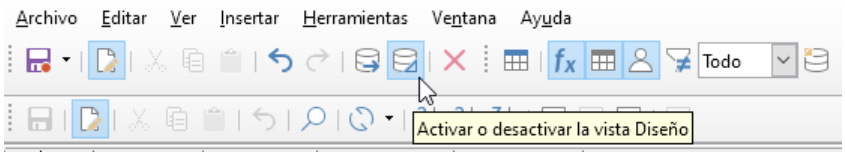
	ID	Media_ID	Reader_ID	Date of Issue	Return Date
▶	22	2	1	04/03/13	

La vista de tabla anterior muestra cómo los alias reemplazan los nombres de campo reales.

Return_Date
Return Date
Loan
☑
IS EMPTY

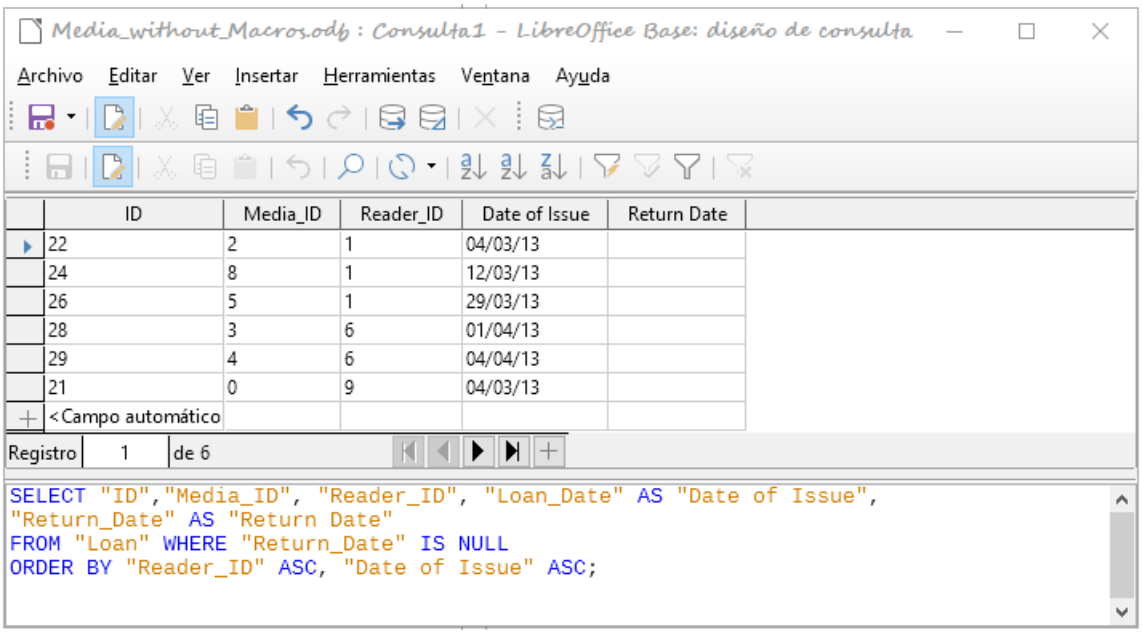
El campo *Return_Date* no solo tiene un alias, sino también un criterio de búsqueda, lo que hará que solo se muestren aquellos registros para los que este campo esté vacío (Al poner `IS EMPTY` en la fila *Criterio* del campo *Return_Date*).

Este criterio de exclusión hará que solo se muestren aquellos registros relacionados con artículos del préstamo que aún no han sido devueltos.



Para aprender mejor el lenguaje SQL, es aconsejable cambiar de vez en cuando al *modo SQL*. Esto se hace mediante el botón *Activar o desactivar la vista Diseño.* Lo cual muestra la ventana del editor de texto de SQL y oculta el área gráfica y el área tabular.

Para volver al modo de vista *Diseño* vuelva a pulsar este mismo botón.



Aquí se revela la fórmula SQL creada por nuestras elecciones anteriores. Para facilitar la lectura, se han incluido algunos saltos de línea. Desafortunadamente, el editor no almacena estos saltos de línea, por lo que cuando se vuelve a llamar la consulta, aparecerá como un salto de línea continuo en el borde de la ventana.

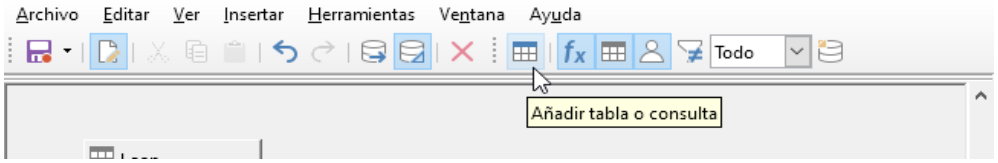
- `SELECT` es el principio de los criterios de selección (los campos que se mostrarán).
- `AS` especifica los alias de campo que se utilizarán.(en algunos casos este instrucción no aparece, simplemente se muestra el nombre del campo y su alias a continuación).
- `FROM` muestra la tabla que se utilizará como fuente de la consulta.
- `WHERE` proporciona las condiciones para la consulta, en este caso, que el campo *Return_Date* debe estar vacío (*IS NULL*).
- `ORDER BY` define los criterios de clasificación también en este caso orden ascendente (*ASC*) para los dos campos *Reader_ID* y *Loan_Date*. Esta especificación de clasificación ilustra cómo se puede usar el alias para el campo *Loan_Date* dentro de la consulta misma.

Consejo

Cuando trabaje en el modo Vista de diseño, use `IS EMPTY` para un campo vacío. Cuando trabaje en modo SQL, use `IS NULL`, que es lo que reconoce SQL.

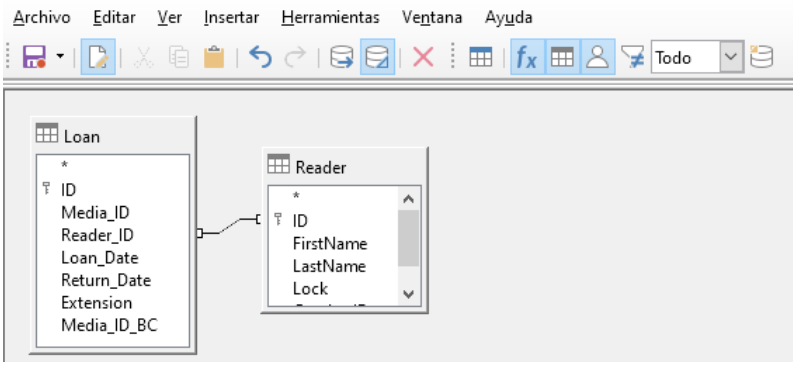
Cuando desee usar el orden descendente en modo SQL, use `DESC` en lugar de `ASC`.

Hasta ahora, los campos *Media_ID* y *Reader_ID* solo son visibles como campos numéricos. Los nombres de los lectores no están claros. Para mostrarlos en una consulta, se debe incluir la tabla *Reader*. Para este propósito volvemos al modo de diseño. y agregamos una nueva tabla a la vista *Diseño*.



Mediante el botón *Añadir tabla o consulta* se pueden agregar más tablas o consultas y hacerlas visibles en la interfaz gráfica de usuario. Si los enlaces entre las tablas ya se declararon en el

momento de su creación (consulte el «Capítulo 3, Tablas»), estas tablas se muestran con los enlaces directos correspondientes.



Si no hay un enlace, se puede crear en este momento arrastrando el ratón desde un campo de una tabla hasta otro de otra tabla. Crearemos un enlace entre el campo *Reader_ID* de la tabla *Loan* y el campo *ID* de la tabla *Reader*.

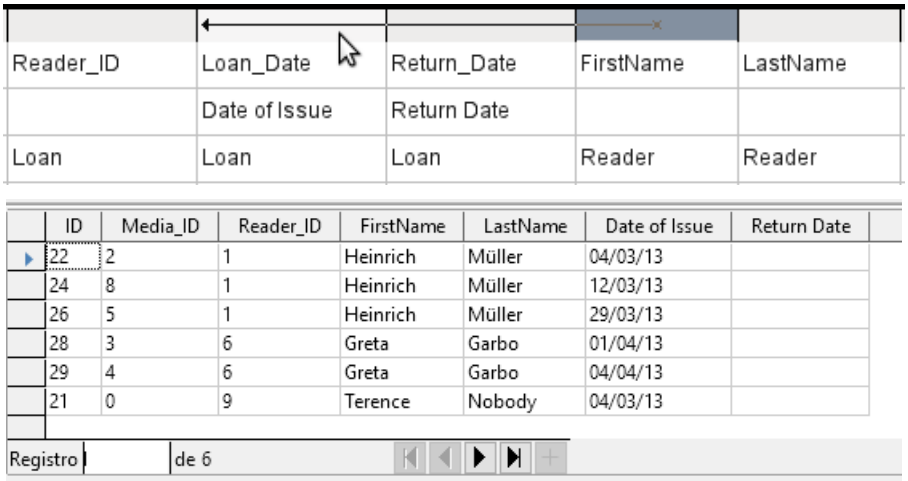
Nota

La vinculación de tablas de momento solo funciona en la base de datos interna o en bases de datos relacionales externas. Por ejemplo, las tablas de una hoja de cálculo no se pueden vincular entre sí. Primero tienen que importarse a una base de datos interna.

Para crear un enlace entre las tablas, basta con una simple importación, sin necesidad de crear una clave primaria.

Ahora los campos de la tabla *Reader* se pueden incorporar en el área tabular. Los campos se agregan al final de la consulta.

La posición de los campos se puede corregir en el área tabular del editor con el ratón. Así, por ejemplo, el campo *FirstName* se ha arrastrado a su posición directamente antes del campo *Loan_Date*.



Reader_ID	Loan_Date	Return_Date	FirstName	LastName
	Date of Issue	Return Date		
Loan	Loan	Loan	Reader	Reader

ID	Media_ID	Reader_ID	FirstName	LastName	Date of Issue	Return Date
22	2	1	Heinrich	Müller	04/03/13	
24	8	1	Heinrich	Müller	12/03/13	
26	5	1	Heinrich	Müller	29/03/13	
28	3	6	Greta	Garbo	01/04/13	
29	4	6	Greta	Garbo	04/04/13	
21	0	9	Terence	Nobody	04/03/13	

Registro de 6

Ahora los nombres son visibles. El *Reader_ID* se ha vuelto superfluo, con lo que se eliminará. Tampoco tiene sentido ordenar por Reader_ID. Se ordenará por Apellido y luego Nombre.

Esta consulta ya no es adecuada para su uso como una consulta que permita nuevas entradas en la tabla resultante, ya que carece de la clave principal para la tabla *Reader* agregada.

Solo si esta clave principal está integrada, la consulta se puede editar de nuevo. De hecho, se puede editar totalmente, los nombres de los lectores también se pueden modificar. Por esta razón,

hacer que los resultados de la consulta se puedan editar es una facilidad que debe usarse con extrema precaución, si es necesario bajo el control de un formulario.

Precaución

Una consulta editable puede crear problemas. La edición de datos en la consulta también edita los datos de la tabla subyacente y los registros de la tabla. Al editar los datos se modifica su significado. Por ejemplo, al cambiar el ID de un lector los artículos que el lector había tomado prestados y devuelto no quedan a su nombre.

Si tiene que facilitar la edición de datos, hágalo de forma que pueda ver los efectos de la edición de los datos editados.

Incluso aunque una consulta puede editarse, no es tan fácil de usar como un formulario con controles de listado, (estos controles pueden mostrar los nombres de los lectores, pero realmente contienen el *Reader_ID* de la tabla). Los controles de listado no se pueden agregar a una consulta; solo se pueden usar en formularios.

```
SELECT "Loan"."ID", "Loan"."Media_ID", "Loan"."Reader_ID",
"Reader"."FirstName", "Reader"."LastName",
"Loan"."Loan_Date" AS "Date or Issue", "Loan"."Return_Date" AS "Return Date"
FROM "Loan", "Reader"
WHERE "Loan"."Reader_ID" = "Reader"."ID" AND "Loan"."Return_Date" IS NULL
ORDER BY "Loan"."Reader_ID" ASC, "Date or Issue" ASC
```

Si ahora volvemos a la Vista SQL, vemos que todos los campos ahora se muestran entre comillas dobles y los nombres de los campos van precedidos del nombre de la tabla y un punto: ["Nombre_de_tabla"."Nombre_de_campo"]. Esto es necesario para que la base de datos sepa de qué tabla provienen los campos previamente seleccionados. Después de todo, los campos en diferentes tablas pueden tener fácilmente los mismos nombres de campo. En la estructura de la tabla anterior, esto se nota claramente en el campo ID.

Nota

La siguiente consulta funciona sin poner nombres de tabla delante de los nombres de campo:

```
SELECT "ID", "Number", "Price" FROM "Stock", "Dispatch" WHERE "Dispatch"."stockID" =
"Stock"."ID"
```

ID se toma de la tabla que aparece primero en la definición FROM. La definición de la tabla en la Fórmula WHERE también es superflua, porque *stockID* solo aparece una vez (en la tabla *Dispatch*) y la *ID* se tomó claramente de la tabla *Stock* (de la posición de la tabla en la consulta).

Si un campo en la consulta tiene un alias, puede referirse a él, por ejemplo, en la ordenación, sin este nombre de tabla.

La ordenación se lleva a cabo en la interfaz gráfica de acuerdo con la secuencia de campos en la vista tabular. Si, desea ordenar primero por *Loan_Date* y luego por *Reader_ID* (de la tabla *Loan*), puede hacerlo usando una de las siguientes maneras:

- Cambiando la secuencia de campos en el área de la tabla de la interfaz gráfica de usuario (arrastre y suelte *Loan_Date* a la izquierda de *Reader_ID)*
- Agregando un campo adicional, solo para la ordenación, configurado para que sea invisible (el editor lo registrará solo temporalmente si no se definió ningún alias).

  Ej.: agregue otro campo *Loan_Date* justo antes de *Reader_ID* o bien agregue otro campo *Reader_ID* justo después de *Loan_Date*

- O modificando el texto para la orden ORDER BY en el editor de SQL de manera adecuada: [ORDER BY "Loan_Date", "Loan"."Reader_ID"].

Consejo

Una consulta puede requerir un campo que no sea parte del resultado de la consulta. En el gráfico de la siguiente sección, *Return_Date* es un ejemplo. Esta consulta está buscando registros que no contengan una fecha de retorno. Este campo proporciona un criterio para la consulta, pero no proporciona datos visibles útiles.

Usar funciones en una consulta

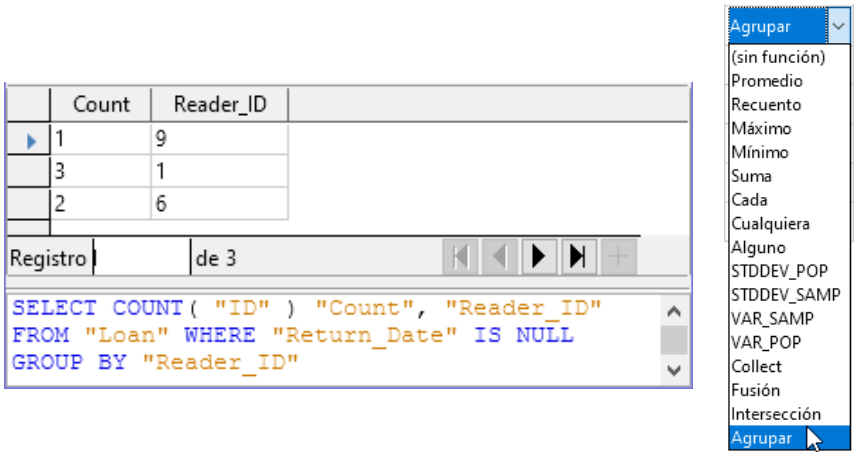
El uso de funciones permite que una consulta proporcione más que una vista filtrada de los datos en una o más tablas. La siguiente consulta calcula cuántos artículos se han prestado, según el *Reader_ID*.

Campo	ID	Reader_ID	Return_Date
Alias	Count		
Tabla	Loan	Loan	Loan
Orden			
Visible	☑	☑	☐
Función	Recuento	Agrupar	
Criterio			IS EMPTY
o			

Figura 172: Funciones en consulta

Para la *ID* de la tabla *Loan*, se selecciona la función *Recuento*. En principio, no importa qué campo de una tabla se elija para ello. La única condición es que el campo no debe estar vacío en ninguno de los registros. Por esta razón, el campo de clave principal, que nunca está vacío, es la opción más adecuada. (Se cuentan todos los campos con un contenido que no sea NULL).

Para el *Reader_ID*, que da acceso a la información del lector, se elige la función *Agrupar*. De esta forma, los registros con el mismo *Reader_ID* se agrupan. El resultado muestra el número de registros para cada *Reader_ID*. Como criterio de búsqueda, *Return_Date* se establece como IS EMPTY, igual que en el ejemplo anterior.



El resultado de la consulta muestra que *Reader_ID* 1 tiene un total de 3 artículos prestados.

Si la función *Recuento* (COUNT) se hubiera asignado a *Return_Date* en lugar de *ID*, cada *Reader_ID* tendría un artículo 0 en préstamo, ya que *Return_date* se ha definido como NULL.

La fórmula correspondiente en el código SQL se muestra arriba. La interfaz gráfica de usuario proporciona las funciones que se muestran a la derecha, que corresponden a funciones HSQLDB.

Consulte «Mejora de consultas usando el modo SQL» en la página 212, para obtener una explicación de las funciones.

Si un campo en una consulta está asociado con una función, el resto de los campos mencionados en la consulta también deben estar asociados con funciones. Si esto no es así, recibirá el siguiente mensaje de error:

En la figura 172 el campo *Return_Date* no tiene función asociada y se produce dicho error. Una traducción algo libre sería: La siguiente expresión no contiene ninguna función agregada o agrupación.

Consejo

Cuando se utiliza el *Modo de vista de diseño*, un campo solo es visible si la fila *Visible* contiene una marca de verificación para el campo. Cuando se utiliza el modo SQL, un campo solo es visible cuando sigue a la palabra clave `SELECT`.

Nota

Cuando un campo no está asociado con una función, el número de filas en la salida de la consulta está determinado por las condiciones de búsqueda.

Cuando un campo está asociado con una función, el número de filas en la salida de la consulta se determina si hay alguna agrupación o no. Si no hay agrupación, solo se verá una fila en la salida de la consulta. Si hay agrupación, el número de filas coincide con el número de valores distintos que tiene el campo de agrupación. Por lo tanto, todos los campos visibles deben estar asociados con una función o no estar asociados con ninguna para evitar el conflicto en la salida de la consulta.

Después de esto, el código de la consulta aparece en el mensaje de error, pero sin referencia específica al campo ofensivo. En este caso, el campo *Return_Date* se ha agregado como un campo visible. Este campo no tiene ninguna función asociada y tampoco está incluido en la declaración de agrupación.

La información proporcionada al pulsar el botón *Más* del mensaje de error no es muy esclarecedora para el usuario final de la base de datos. Simplemente muestra el código de error de SQL.

Para corregir el error, elimine la marca de verificación en la fila «Visible» del campo *Return_Date*. Aunque su condición de búsqueda (Criterio) se aplica cuando se ejecuta la consulta, no es visible en el resultado.

Usando la Interfaz se pueden aplicar cálculos básicos y funciones adicionales.

Suponga que una biblioteca no emite avisos cuando un artículo debe devolverse, sino que emite avisos de retraso cuando el período del préstamo haya expirado y el artículo no haya sido devuelto. Esta es una práctica común en las bibliotecas escolares y públicas que hacen préstamos

solo por períodos cortos y fijos. En este caso, la emisión de un aviso de retraso significa que se debe pagar una multa. ¿Cómo calculamos estas multas?

Campo	ID	Media_ID	Reader_ID	Date	Recuento("Recall"."Date") * 2	Return_Date
Alias				RecallCount	RecallAmmount	
Tabla	Loan	Loan	Loan	Recall		Loan
Orden						
Visible	☑	☑	☑	☑	☑	☐
Función	Agrupar	Agrupar	Agrupar	Recuento		
Criterio						IS EMPTY

En la consulta que se muestra, las tablas *Loan* y *Recall* se consultan conjuntamente. A partir del recuento de las entradas de datos en la tabla Recall, se determina el número total de avisos de retraso. La multa por los artículos vencidos se establece en la misma consulta en 2,00 € y se aplica creando un nuevo campo. En lugar de un nombre de campo, se utiliza la fórmula correspondiente: «Recuento("Recall"."Date")*2».

La interfaz gráfica de usuario agrega las comillas y convierte el término «Recuento» en la orden SQL apropiada.

Precaución

Solo para usuarios que utilizan la coma para su separador decimal:

Si desea ingresar números con lugares decimales utilizando la Interfaz, debe asegurarse de que usa un punto decimal en lugar de una coma como separador decimal dentro de la instrucción SQL. Las comas se utilizan como separadores de campo, por lo que si utiliza comas, se crearán nuevos campos de consulta para la parte decimal.

Una entrada con una coma en la vista SQL siempre conduce a un campo adicional que contiene el valor numérico de la parte decimal.

ID	Media_ID	Reader_ID	RecallCount	RecallAmount
24	8	1	1	2
22	2	1	1	2

Registro de 2

```
SELECT "Loan"."ID", "Loan"."Media_ID", "Loan"."Reader_ID",
COUNT( "Recall"."Date" ) "RecallCount",
COUNT( "Recall"."Date" ) * 2 "RecallAmount"
FROM "Recall", "Loan"
WHERE "Recall"."Loan_ID" = "Loan"."ID"
AND "Loan"."Return_Date" IS NULL
GROUP BY "Loan"."ID", "Loan"."Media_ID", "Loan"."Reader_ID", "Loan"."Return_Date"
```

La consulta ahora arroja para cada artículo aún en préstamo las multas que se han acumulado, según los avisos de retraso acumulados y el campo de multiplicación adicional. La siguiente estructura de consulta también será útil para calcular las multas adeudadas por usuarios individuales.

Campo	Reader_ID	Date	Recuento("Recall"."Date") * 2	Return_Date
Alias		RecallCount	RecallAmmount	
Tabla	Loan	Recall		Loan
Orden				
Visible	☑	☑	☑	☐
Función	Agrupar	Recuento		
Criterio				IS EMPTY

Se han eliminado los campos *Loan.ID* y *Loan.ID_Media*. Se usaron en la consulta anterior para crear un registro separado para cada artículo mediante una agrupación. Ahora se agrupará solo por el lector. El resultado de la consulta se ve de la siguiente manera:

Reader_ID	RecallCount	RecallAmount
1	2	4

Registro | de 1

En lugar de enumerar los artículos para *Reader_ID* = 1 por separado, se contaron todos los campos *"Recall". "Date"* y se calculó un total de 4.00 € como la multa debida.

Definir relaciones en la consulta

Cuando se buscan datos en tablas o formularios, la búsqueda generalmente se limita a una tabla o un formulario. La función de búsqueda incorporada no puede recorrer los enlaces de un formulario principal con un subformulario. Para tal fin, los datos que se quieren buscar se recopilan mejor mediante una consulta.

Title
Der kleine Hobbit
Das sogennante Böse
Eine kurze Geschichte der Zeit
Traditionelle und kritische Theorie
Die neue deutsche Rechtschreibung
I hear you knocking
Datenbanken mit OpenOffice.org 3
Das Postfix-Buch
Im Augenblick

Registro | 1 | de 9

Campo	Title
Alias	
Tabla	Media
Orden	
Visible	☑ ☐
Función	
Criterio	

Title	Subtitle
I hear you knocking	Youn can't catch me
I hear you knocking	The stumble
I hear you knocking	Sabre dance (Single version)
Im Augenblick	Amsterdam
Im Augenblick	Hier unten am Deich
Im Augenblick	Köln-Ehrenfeld
Im Augenblick	Bei Mir
Im Augenblick	Gott sei Dank

Registro | de 8

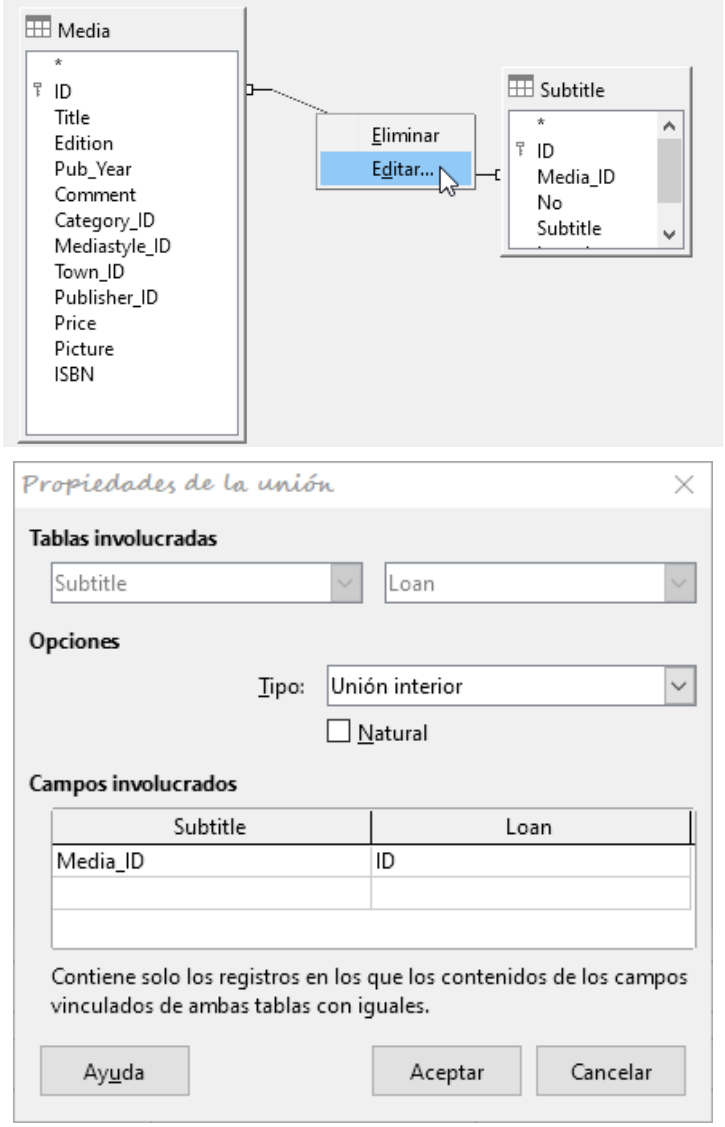
Campo	Title	Subtitle
Alias		
Tabla	Media	Subtitle
Orden		
Visible	☑	☑
Función		
Criterio		

Una consulta sencilla para el campo *Título* de la tabla *Media* arroja un resultado de: 9 registros en total (los que tienen un título). Pero si se añade la tabla *Subtitle* y el campo *Subtitle* de esta tabla

en la consulta, el resultado de la consulta se reduce a solo 2 Títulos. Solamente estos dos artículos tienen también subtítulos. El resto de los registros, aunque tienen título, no tienen subtítulo.

Este resultado se debe a la relación de unión entre tablas. En la consulta, la unión está establecida para que solo se muestren aquellos registros en los que el campo *Media_ID* de la tabla *Subtitle* sea igual al campo *ID* de la tabla *Media*. Los demás registros quedan excluidos.

Las condiciones de unión se deben editar para poder mostrar los registros deseados. Nos referimos aquí no a uniones en el diseño de relación entre tablas, sino a uniones dentro de consultas.



De manera predeterminada, las relaciones se establecen como uniones interiores. El diálogo proporciona información sobre cómo funciona este tipo de combinación.

Las dos tablas previamente seleccionadas se enumeran como tablas involucradas que no se pueden seleccionar aquí.

Los campos relevantes de las dos tablas se leen de las definiciones de tabla. Si no hay una relación especifica en las definiciones de las tablas, se puede crear una en este momento para la consulta. Sin embargo, si ha planificado su base de datos de manera ordenada utilizando HSQLDB, no debería ser necesario modificar estos campos.

La configuración más importante es el tipo de unión. Las relaciones se pueden establecer de modo que se seleccionen todos los registros de la tabla *Subtitle*, pero solo aquellos registros de la tabla *Media* que tengan un subtítulo.

O por el contrario, que se muestren todos los registros de la tabla *Media*, independientemente de si tienen o no un subtítulo.

La opción de la casilla de verificación *Natural* especifica que los campos vinculados en las tablas se tratan como iguales. También puede evitar el uso de esta configuración definiendo sus relaciones correctamente al comienzo de la planificación de su base de datos.

Para el tipo *Unión a la derecha*, la descripción apunta que se mostrarán todos los registros de la tabla *Media* [Subtitle RIGHT JOIN Media]. Como no hay subtítulos que carezcan de un título en los artículos, pero sí hay títulos en los artículos que carecen de subtítulos, esta es la opción correcta.

Después de confirmar la unión correcta, los resultados de la consulta se ven como queríamos. Título y subtítulo se muestran juntos en la consulta. Naturalmente, los títulos aparecen más de una vez como en la relación anterior. Sin embargo, siempre que no se cuenten los resultados, esta consulta se puede utilizar como base para funciones de búsqueda. Consulte los fragmentos de código que se utilizan en este capítulo, en el «Capítulo 8, Tareas de base de datos» y en el «Capítulo 9, Macros».

Title	Subtitle
Der kleine Hobbit	
Das sogennante Böse	
Eine kurze Geschichte der Zeit	
Traditionelle und kritische Theorie	
Die neue deutsche Rechtschreibung	
I hear you knocking	Youn can't catch me
I hear you knocking	The stumble
I hear you knocking	Sabre dance (Single version)
Datenbanken mit OpenOffice.org 3	
Das Postfix-Buch	
Im Augenblick	Amsterdam
Im Augenblick	Hier unten am Deich
Im Augenblick	Köln-Ehrenfeld
Im Augenblick	Bei Mir
Im Augenblick	Gott sei Dank

Registro 1 de 15

Definir propiedades de consulta

A partir de la versión 4.1 de LibreOffice, es posible definir propiedades adicionales en el editor de consultas.

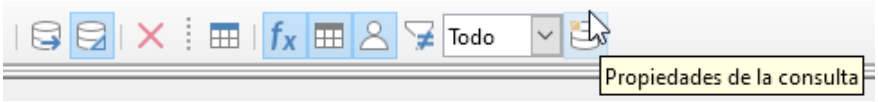
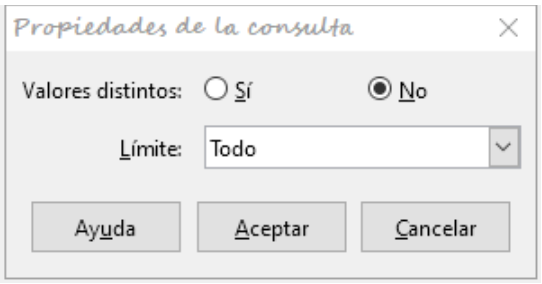


Figura 173: Abrir propiedades de la consulta en el editor de consultas

Al la izquierda del botón para abrir las *Propiedades de la consulta* hay un Control de lista: *Límite* para regular la cantidad de registros que se muestran, así como un botón para *Valores distintos*. Estas opciones también están presentes en el diálogo *Propiedades de la consulta*:



La configuración de *Valores distintos* determina si la consulta debe suprimir registros duplicados.

	FirstName	LastName	Return_Date
▶	Greta	Garbo	
	Greta	Garbo	
	Lisa	Gerd	
	Lisa	Gerd	
	Lisa	Gerd	
	Lisa	Gerd	
	Heinrich	Müller	
	Heinrich	Müller	
	Heinrich	Müller	
	Terence	Nobody	

Supongamos que se realiza una consulta para determinar qué lectores aún tienen artículos prestados. Sus nombres se muestran si el campo de fecha de retorno está vacío. Los nombres se muestran más de una vez para los lectores que tienen varios artículos prestados.

Si elige *Valores distintos,* los registros con el mismo contenido desaparecerán. La consulta se verá así:

	FirstName	LastName	Return_Date
▶	Greta	Garbo	
	Lisa	Gerd	
	Heinrich	Müller	
	Terence	Nobody	

```
SELECT DISTINCT
"Reader"."FirstName", "Reader"."LastName", "Loan"."Return_Date"
FROM "Loan", "Reader"
WHERE "Loan"."Reader_ID" = "Reader"."ID" AND "Loan"."Return_Date" IS NULL
ORDER BY "Reader"."LastName" ASC
```

La consulta original:

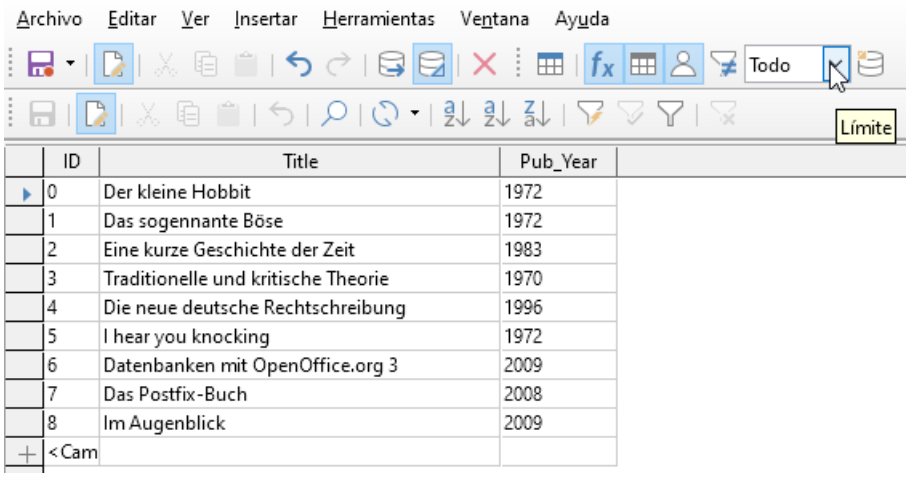
```
SELECT "Reader"."FirstName", "Reader"."LastName" ...
```

Agregar `DISTINCT` a la consulta suprime los registros duplicados.

```
SELECT DISTINCT "Reader"."FirstName", "Reader"."LastName" ...
```

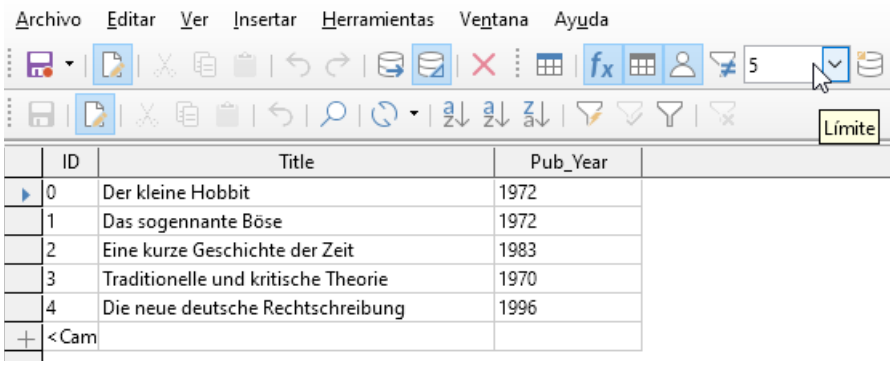
La capacidad de especificar registros únicos también estuvo presente en versiones anteriores. Sin embargo, era necesario cambiar de la vista de diseño a la vista SQL para insertar el calificador DISTINCT. Esta propiedad es compatible con versiones anteriores de LibreOffice.

La configuración del *Límite* determina cuántos registros se mostrarán en la consulta.



ID	Title	Pub_Year
0	Der kleine Hobbit	1972
1	Das sogennante Böse	1972
2	Eine kurze Geschichte der Zeit	1983
3	Traditionelle und kritische Theorie	1970
4	Die neue deutsche Rechtschreibung	1996
5	I hear you knocking	1972
6	Datenbanken mit OpenOffice.org 3	2009
7	Das Postfix-Buch	2008
8	Im Augenblick	2009
<Cam		

Figura 174: Se muestran todos los registros en la tabla Media. La consulta se puede editar, ya que incluye la clave primaria.



ID	Title	Pub_Year
0	Der kleine Hobbit	1972
1	Das sogennante Böse	1972
2	Eine kurze Geschichte der Zeit	1983
3	Traditionelle und kritische Theorie	1970
4	Die neue deutsche Rechtschreibung	1996
<Cam		

Solo se muestran los primeros cinco registros (*ID* 0-4). No se solicitó una ordenación, por lo que se utiliza el orden predeterminado (por clave principal). A pesar de la limitación en la salida, la consulta se puede editar. Esto distingue la entrada en la interfaz gráfica de lo que, en versiones anteriores, solo era accesible usando SQL.

```
SELECT "ID", "Title", "Pub_Year" FROM "Loan" LIMIT 5
```

Simplemente se ha agregado LIMIT 5 a la consulta original. El tamaño del límite puede ser el que se necesite.

Precaución

Establecer límites en la interfaz gráfica no es compatible con versiones anteriores. En las versiones de LibreOffice anteriores a la 4.1, un límite solo se podía establecer en modo SQL directo. El límite entonces requería una clasificación (ORDER BY…) o una condición (WHERE…).

Mejora de consultas usando el modo SQL

Si durante la entrada gráfica usa **Ver > Activar o desactivar la vista Diseño** para desactivar la vista Diseño, verá el editor de SQL en lugar de la vista Diseño. Esta es la mejor manera para que

los principiantes aprendan el lenguaje de consulta estándar para bases de datos. A veces, también es la única forma de realizar una consulta en la base de datos cuando la interfaz no puede traducir sus requisitos a las órdenes SQL necesarias.

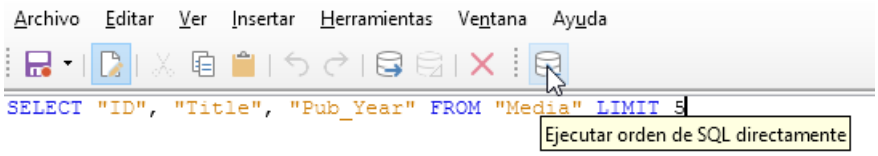
```
SELECT * FROM "Nombre_Tabla"
```

Esto mostrará todo lo que está en la tabla indicada. El asterisco "*" presenta todos los campos de la tabla.

```
SELECT * FROM "Nombre_Tabla" WHERE "Nombre_Campo" = 'Karl'
```

Aquí hay una restricción significativa. Solo se muestran aquellos registros para los que el campo "Nombre_Campo" contiene el término exacto «Karl», no mostrará, por ejemplo, «Karl Egon».

A veces, las consultas en Base no se pueden llevar a cabo utilizando la interfaz, ya que puede que no sean reconocidas algunas órdenes SQL concretas. En estos casos, es necesario desactivar la vista de *Diseño* y usar **Editar > Ejecutar orden de SQL directamente** para acceder a la base de datos. Este método tiene la desventaja de que solo se puede trabajar con la consulta en modo *editor de SQL*.



El uso directo de las órdenes SQL también es accesible mediante la interfaz gráfica de usuario, como muestra la figura anterior. Haga clic en el icono resaltado *Ejecutar orden de SQL directamente* una vez desactivada la vista *Diseño*. Ahora, cuando hace clic en el icono *Ejecutar*, la consulta ejecuta las órdenes SQL directamente.

A continuación se muestran las amplias posibilidades disponibles para plantear consultas a la base de datos y especificar el tipo de resultado requerido:

```
SELECT [{LIMIT <offset> <límite> | TOP <límite>}][ALL | DISTINCT]
{ <Formulación_para_select> | "Nombre_Tabla".* | * } [, ...]
[INTO [CACHED | TEMP | TEXT] "nueva_Tabla"]
FROM "Lista_de_Tablas"
[WHERE Expresión_SQL]
[GROUP BY Expresión_SQL [, ...]]
[HAVING Expresión_SQL]
[{ UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT] } |
INTERSECT [DISTINCT] } Declaración_consulta]
[ORDER BY Criterio_de_Ordenación [, ...]]
[LIMIT <límite> [OFFSET <offset>]];
```

[{LIMIT <offset> <límite> | TOP <límite>}]:

Esto limita el número de registros que se mostrarán:

`LIMIT 10 20` comienza en el undécimo registro y muestra los siguientes 20 registros.

`TOP 10` siempre muestra los primeros 10 registros. Es lo mismo que `LIMIT 0 10`.

`LIMIT 10 0` omite los primeros 10 registros y muestra todos los registros a partir del 11.

Puede hacer lo mismo usando la condición `SELECT` En la fórmula `[LIMIT <límite> [OFFSET <offset>]]`.`LIMIT 10` muestra solo 10 registros. Agregar `OFFSET 20` hace que comience en el registro n.º 21. Esta forma final de limitación de visualización requiere una instrucción de clasificación (`ORDER BY…`) o una condición (`WHERE…`).

Todos los valores empleados en el límite deben ser de tipo *integer*. No es posible reemplazar una entrada por una subconsulta para que, por ejemplo, se muestren los cinco últimos registros de una serie.

[ALL | DISTINCT]

SELECT ALL es el valor predeterminado. Se muestran todos los registros que cumplen las condiciones de búsqueda. Ejemplo: SELECT ALL "Nombre" FROM "Tabla_nombres" produce todos los nombres; si "Peter" aparece tres veces y "Egon" cuatro veces en la tabla, estos nombres se muestran tres y cuatro veces respectivamente. SELECT DISTINCT "Nombre" FROM "Tabla_nombres" suprime los resultados de la consulta que tienen el mismo contenido. En este caso, "Peter" y "Egon" aparecen solo una vez. DISTINCT se refiere a la totalidad del campo al que accede la consulta. Si, por ejemplo, también se solicita el apellido, los registros de "Peter Müller" y "Peter Maier" contarán como distintos. Incluso si especifica la condición DISTINCT, se mostrarán ambas.

<Formulación_para_select>

```
{ Expresión | COUNT(*) |
{ COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP | STDDEV_POP |
STDDEV_SAMP }
([ALL | DISTINCT]] Expresión) } [[AS] "alias_a_mostrar"]
```

Los nombres de campo, los cálculos, los totales de registros son todas las entradas posibles.

Nota

Los cálculos dentro de una base de datos a veces conducen a resultados inesperados. Suponga que una base de datos contiene las calificaciones otorgadas por algún trabajo de clase y desea calcular la calificación promedio.

Primero deberá sumar todas las calificaciones. Supongamos que la suma es 80. Ahora debe dividirse por el número de alumnos (digamos 30). La consulta produce 2.

Esto ocurre porque está trabajando con campos de tipo INTEGER. Por lo tanto, el resultado del cálculo produce un valor entero. Debe tener al menos un valor del tipo DECIMAL en su cálculo. Puede lograr esto utilizando una función de conversión HSQLDB o (simplemente) puede dividir entre 30.0 en lugar de 30. Esto dará un resultado con un decimal. Dividir por 30.00 da dos lugares decimales.

Tenga el cuidado de usar puntos para la separación decimal (estilo inglés). El uso de comas en las consultas está reservado para los separadores de campo.

Además, hay diferentes funciones disponibles para el campo que se muestra. Excepto COUNT(*) (que cuenta todos los registros) ninguna de las siguientes funciones accede a los campos NULL.

```
COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP | STDDEV_POP |
STDDEV_SAMP
```

COUNT("Nombre") cuenta todas las entradas para el campo Nombre.

MIN("Nombre") muestra el primer nombre alfabéticamente. El resultado de esta función siempre se formatea tal como esté en el campo. El texto se muestra como texto, los enteros como enteros, los decimales como decimales, etc.

MAX ("Nombre") muestra el último nombre alfabéticamente.

SUM ("Número") puede sumar solo valores en campos numéricos. La función falla en campos de fecha u hora.

Consejo

Aunque la función SUM falla en los campos de hora, se puede hacer efectiva empleando la siguiente fórmula:

```
SELECT (SUM( HOUR("Time") )*3600 + SUM( MINUTE("Time") ))*60 +
SUM( SECOND("Time") ) AS "seconds" FROM "Table"
```

La suma mostrada se compone de horas, minutos y segundos separados. Las horas y los minutos se convierten a segundos. Con lo que se obtiene el total en segundos.

Añadiendo una división por 3600 obtendrá un resultado horario en horas con los minutos y segundos como lugares decimales.

```
SELECT ((SUM( HOUR("Time") )*3600 + SUM( MINUTE("Time") ))*60 +
SUM( SECOND("Time") )) / 3600.0000 AS "hours" FROM "Table"
```

Con un formato adecuado, se puede convertir a un valor horario correcto en una consulta o formulario.

AVG ("Número") muestra el promedio del contenido de una columna. Esta función también está limitada a campos numéricos.

SOME("Nombre_Campo"), EVERY ("Nombre_Campo"): Los campos utilizados con estas funciones deben tener el tipo de campo *Sí / No [BOOLEAN]* (contiene solo 0 o 1). Además, producen un resumen del contenido del campo al que se aplican.

SOME devuelve TRUE (o `1`) si al menos una entrada para el campo es `1`, y devuelve FALSE (o `0`) solo si todas las entradas son `0`.

EVERY devuelve TRUE solo si todos los valores en el campo son `1`, y devuelve FALSE si al menos un valor es `0`.

Consejo

El tipo de campo booleano es *Sí / No [BOOLEAN]*. Este campo solo almacena 0 o 1. (equivalentes a FALSE y TRUE). No se almacena SI o NO.

Si utiliza la búsqueda en consultas, utilice `TRUE` o `1` (para SI), o bien `FALSE` o `0` (para NO). Si intenta utilizar «Sí» o «No» no encontrará nada.

Ejemplo:

```
SELECT "Clase", EVERY("Nadador")
FROM "Tabla1"
GROUP BY "Clase"
```

En una tabla *Tabla1* tenemos los alumnos de la escuela, esta tabla tiene varios campos:

Clase que contiene las clases a que pertenecen los alumnos, *Alumno* contiene los nombres de los estudiantes, *Nadador* es un campo booleano que describe si un estudiante sabe nadar o no (`1` o `0`). y una clave principal. *ID*

Solo *Clase* y *Nadador* están incluidos en esta consulta.

Debido a que la consulta está agrupada por las entradas del campo *Clase*, EVERY devolverá el valor del campo, *Nadador*, para cada clase. Si todos los alumnos de una clase saben nadar, EVERY devolverá TRUE. Si al menos un alumno de la clase no sabe nadar, EVERY devuelve FALSE.

En el resultado de la consulta, el campo *Nadador* es una casilla de verificación, una marca en esta casilla indica TRUE y ninguna marca FALSE.

```
VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP
```

Son funciones estadísticas y solo afectan a los campos enteros y decimales.

Todas estas funciones devuelven 0, si los valores dentro del grupo son todos iguales. Las funciones estadísticas son incompatibles con DISTINCT. Básicamente, calculan sobre todos los valores de los registros en la consulta, y DISTINCT excluye registros con los mismos valores con lo que el resultado sería erróneo.

[AS] "alias_a mostrar": Devuelve el alias del campo indicado dentro de la consulta.

"Nombre_Tabla".* | * [, ...]

Cada campo a mostrar se proporciona con sus nombres de campo, separados por comas. Si se utilizan campos de varias tablas en la consulta, es necesaria una combinación del nombre de la tabla con el nombre del campo: `"Nombre_Tabla"."Nombre_campo"`.

En lugar de una lista detallada de todos los campos de una tabla, Puede mostrar su contenido total utilizando el símbolo asterisco «*», de este modo no es necesario usar el nombre de la tabla si los resultados solo se aplican a una tabla. Sin embargo, si la consulta incluye todos los campos de una tabla y al menos un campo de una segunda tabla, use:

```
"Nombre_Tabla1".*, "Nombre_Tabla2"."Nombre_Campo"
```

[INTO [CACHED | TEMP | TEXT] "nueva_tabla"]

El resultado de esta consulta se debe escribir directamente en una tabla nueva con el nombre indicado. Las propiedades de campo para la nueva tabla se definen a partir de las definiciones de campo contenidas en la consulta. Escribir en una tabla nueva no funciona desde el modo SQL, ya que solo maneja los resultados que se pueden mostrar. En su lugar, debe usar **Herramientas > SQL**. La tabla resultante inicialmente no es editable, ya que carece de una clave primaria.

FROM <Lista_de_Tablas>

```
"Nombre_Tabla1" [{CROSS | INNER | LEFT OUTER | RIGHT OUTER} JOIN "Nombre_Tabla2"
ON Expresión] [, ...]
```

Las tablas que se deben buscar conjuntamente generalmente están en una lista separada por comas. La relación de las tablas entre sí se define con la palabra clave WHERE.

Si las tablas están vinculadas a través de JOIN en lugar de una coma, su relación se define mediante el término que precede inmediatamente a ON, después de la segunda tabla.

Un simple JOIN tiene el efecto de mostrar solo aquellos registros para los que se aplican las condiciones de ambas tablas.

Ejemplo:

```
SELECT "Tabla1"."Nombre", "Tabla2"."Clase"
FROM "Tabla1", "Tabla2"
WHERE "Tabla1"."ClassID" = "Tabla2"."ID"
```

Equivale a:

```
SELECT "Tabla1"."Nombre", "Tabla2"."Clase" FROM "Tabla1"
   JOIN "Tabla2"
```

```
ON "Tabla1"."ClassID" = "Tabla2"."ID"
```

Aquí se muestran los nombres y las clases correspondientes. Si un nombre no tiene una clase en la lista, ese nombre no se incluye en el resultado. Si una clase no tiene nombres, tampoco se muestra. La incorporación de INNER no altera esto.

```
SELECT "Tabla1"."Nombre", "Tabla2"."Clase"
FROM "Tabla1"
   LEFT JOIN "Tabla2"
ON "Tabla1"."ClassID" = "Tabla2"."ID"
```

Si se agrega LEFT, todos los nombres de Tabla1 se muestran incluso si no tienen clase. Si, por el contrario, se agrega RIGHT, todas las clases se muestran incluso si no tienen nombres en ellas. La adición de OUTER no necesita usarse aquí. (RIGHT OUTER JOIN es lo mismo que RIGHT JOIN, lo mismo ocurre con LEFT OUTER JOIN).

```
SELECT "Tabla1"."Jugador1", "Tabla2"."Jugador2"
FROM "Tabla1" AS "Tabla1"
   CROSS JOIN "Tabla2" AS "Tabla1"
WHERE "Tabla1"."Jugador1" <> "Tabla2"."Jugador2"
```

Un CROSS JOIN requiere que la tabla se suministre con un alias, pero no siempre es necesario agregar el término ON. Todos los registros de la primera tabla se emparejan con todos los registros de la segunda tabla. La consulta anterior produce todos los emparejamientos posibles de registros de la primera tabla con los de la segunda tabla, excepto los emparejamientos entre registros para el mismo jugador. En el caso de CROSS JOIN, la condición no debe incluir un enlace entre las tablas especificadas en el término ON. En cambio, las condiciones WHERE se pueden usar. Si las condiciones se formulan exactamente como en el caso de un simple JOIN, obtendrá el mismo resultado:

```
SELECT "Tabla1"."Nombre", "Tabla2"."Clase"
FROM "Tabla1"
   JOIN "Tabla2"
ON "Tabla1"."ClaseID" = "Tabla2"."ID"
```

da el mismo resultado que:

```
SELECT "Tabla1"."Nombre", "Tabla2"."Class"
FROM "Tabla1" AS "Tabla1"
   CROSS JOIN "Tabla2" AS "Tabla2"
WHERE "Tabla1"."ClaseID" = "Tabla2"."ID"
```

[WHERE Expresión SQL]

La introducción estándar de las condiciones para solicitar un filtrado más preciso de los datos. Aquí generalmente se definen también las relaciones entre las tablas si no están ya enlazadas con JOIN.

[GROUP BY Expresión SQL [, …]]

Se usa cuando se desea dividir los datos de la consulta en grupos antes de aplicar las funciones para cada uno de los grupos por separado. La división se basa en los valores del campo o campos contenidos en el término GROUP BY.

Ejemplo:

```
SELECT "Nombre", SUM("Entrada"-"Salida") AS "Saldo"
FROM "Tabla1"
GROUP BY "Nombre";
```

Se suman los registros que contienen el mismo nombre. El resultado de la consulta, proporciona la suma de Entrada - Salida para cada persona. Este campo se llamará *Saldo*.

Cada fila del resultado de la consulta contiene el valor del campo *Nombre* y el saldo calculado para ese valor específico.

Consejo

Cuando los campos se procesan usando una función particular (por ejemplo, COUNT, SUM …), todos los campos que no se procesan con una función pero que se deben mostrar se agrupan usando GROUP BY.

[HAVING Expresión SQL]

La fórmula HAVING se parece mucho a la fórmula WHERE. La diferencia es que la fórmula WHERE se aplica a los valores de los campos seleccionados en la consulta, mientras que la fórmula HAVING se aplica a los valores calculados seleccionados. Específicamente, la fórmula WHERE no puede usar una función agregada como parte de una condición de búsqueda; la fórmula HAVING en cambio sí puede.

La fórmula HAVING tiene dos propósitos, como se muestra en los ejemplos que siguen. En el primer ejemplo, la condición de búsqueda requiere que el tiempo de ejecución mínimo sea inferior a 40 minutos. En el segundo ejemplo, la condición de búsqueda requiere que el saldo de un individuo sea positivo.

Los resultados de la consulta para el primer ejemplo enumeran los nombres de las personas cuyo tiempo de ejecución ha sido inferior a 40 minutos al menos una vez y el tiempo de ejecución mínimo. Las personas que han tenido todos los tiempos de ejecución mayores a 40 minutos no figuran en la lista.

Los resultados de la consulta para el segundo ejemplo enumeran los nombres de las personas que tienen un saldo mayor de entrada que de salida, los nombres cuyo saldo sea 0 o el de entrada inferior al de salida no se mostrarán.

Ejemplo 1:

```
SELECT "Nombre", "Runtime"
FROM "Tabla1"
GROUP BY "Nombre", "Runtime"
HAVING MIN("Runtime") < '00:40:00';
```

Ejemplo 2:

```
SELECT "Nombre", SUM("Entrada"-"Salida") AS "Saldo"
FROM "Tabla1"
GROUP BY "Nombre"
HAVING SUM("Entrada"-"Salida") > 0;
```

[Expresión SQL]

Las expresiones SQL se combinan de acuerdo con el siguiente esquema:

```
[NOT] condición [{ OR | AND } condición]
```

Ejemplo:

```
SELECT *
FROM "Nombre_Tabla"
WHERE NOT "Return_date" IS NULL AND "ReaderID" = 2;
```

Los registros leídos de la tabla son aquellos que contienen un *"Return_date"* y el *ReaderID* es 2. En la práctica, esto significa que todos los artículos prestados a un lector específico y devueltos se pueden utilizar. Las condiciones están vinculadas con AND. La expresión NOT se refiere solo a la primera condición.

```
SELECT *
```

```
FROM "Nombre_Tabla"
WHERE NOT ("Return_date" IS NULL AND "ReaderID" = 2);
```

Los paréntesis alrededor de la condición, con `NOT` fuera de ellos, muestran solo aquellos registros que no cumplen por completo la condición entre paréntesis. Esto cubriría todos los registros, excepto los del *ReaderID* número 2, que aún no hayan sido devueltos.

[Expresión SQL]: condiciones

```
{ valor [|| valor]
```

Un valor puede ser un valor sencillo o varios valores unidos por dos líneas verticales ||. Naturalmente, esto también se aplica al contenido del campo.

```
SELECT "Apellido" || ', ' || "Nombre" AS "Nombre Completo"
FROM "Tabla_nombres"
```

El contenido de los campos *Apellido* y *Nombre* se muestran juntos en un campo llamado *Nombre Completo*. Además se inserta una coma y un espacio entre Apellido y Nombre.

```
| valor { = | < | <= | > | >= | <> | != } valor
```

Estos signos corresponden a los operadores matemáticos conocidos:

{ Igual que | Menor que | Menor o igual que | Mayor que | Mayor o igual que | Distinto de | No igual a}

```
| valor IS [NOT] NULL
```

El campo correspondiente no tiene contenido, porque no se ha escrito nada en él. Esto no se puede determinar de manera inequívoca en la interfaz, ya que un campo de texto visualmente vacío no significa que el campo esté completamente sin contenido. De manera predeterminada en Base los campos vacíos están configurados como `NULL`.

```
| EXISTS(Resultado_Consulta)
```

Ejemplo:

```
SELECT "Nombre"
FROM "Tabla1"
WHERE EXISTS
   (SELECT "Nombre"
   FROM "Tabla2"
   WHERE "Tabla2"."Nombre" = "Tabla1"."Nombre")
```

Se muestran los nombres de la *Tabla1* que coinciden con los nombres de la *Tabla2*.

```
| valor BETWEEN valor AND valor
```

`BETWEEN valor1 AND valor2` devuelve todos los valores desde el valor1 hasta el valor2 (incluido). Si los valores son letras, se utiliza un orden alfabético en el que las letras minúsculas tienen el mismo valor que las mayúsculas correspondientes.

```
SELECT "Nombre"
FROM "Tabla_nombres"
WHERE "Nombre" BETWEEN 'A' AND 'E';
```

Esta consulta devuelve todos los nombres que comienzan con A, B, C o D (incluyendo también los que tengan las letras minúsculas). Como E se establece como el límite superior, los nombres que comienzan con E no se incluyen. La letra E aparece justo antes de los nombres que comienzan con E y solo si el nombre es exactamente «E» se incluye.

```
| valor [NOT] IN ( {valor [, ...] | Resultado_Consulsta } )
```

Esto requiere una lista de valores o una consulta. La condición se cumple si el valor se incluye en la lista de valores o en el resultado de la consulta.

```
| valor [NOT] LIKE valor [ESCAPE] valor }
```

El operador LIKE es uno que se necesita en muchas funciones de búsqueda simples. El valor se ingresa utilizando el siguiente patrón:

'%' representa cualquier número de caracteres (incluido 0),

'_' reemplaza exactamente un carácter.

Para buscar un carácter especial como '%' o '_', los caracteres deben seguir inmediatamente a otro carácter definido como ESCAPE.

```
SELECT "Nombre"
FROM "Tabla_nombres"
WHERE "Nombre" LIKE '\_%' ESCAPE '\'
```

Esta consulta muestra todos los nombres que comienzan con un guión bajo. '\' se utiliza como el carácter ESCAPE. (para indicar que este carácter es parte de la consulta y no un carácter específico de sustitución).

[Expresión SQL]: valores

```
[+ | -] { Expresión [{ + | - | * | / | || } Expresión]
```

Los valores pueden tener uno de los signos anteriores. Se permiten la suma, resta, multiplicación, división y concatenación de expresiones. Un ejemplo de concatenación:

```
SELECT "Apellido"||', '||"Nombre"
FROM "Tabla_nombres"
```

De esta forma, la consulta muestra los registros como un campo que contiene el campo *"Apellido"* seguido de coma y el campo *"Nombre"*. El operador de concatenación puede calificarse mediante las siguientes expresiones.

```
| ( Condición )
```

Se aplica igual que en la sección anterior.

```
| Función ( [Parámetro] [,...] )
```

Vea la sección sobre funciones en el apéndice.

Las siguientes consultas también se denominan sub-consultas (sub-selecciones).

```
| Resultado de Consulta que devuelve exactamente una respuesta
```

Como un registro solo puede tener un valor en cada campo, solo una consulta que produce exactamente un valor puede mostrarse en su totalidad.

```
| {ANY|ALL} (Consulta que devuelve exactamente una respuesta de una columna completa)
```

A menudo hay una condición que compara una expresión con un grupo completo de valores. Combinado con ANY, esto significa que la expresión debe aparecer al menos una vez en el grupo. Esto también se puede especificar utilizando la condición IN. = ANY produce el mismo resultado que IN.

Combinado con ALL significa que todos los valores del grupo deben corresponder a una expresión única.

[Expresión SQL]: Expresión

```
{ 'Texto' | Entero | Numero de coma flotante | ["Tabla".]"Campo" | TRUE | FALSE | NULL }
```

Básicamente, los valores sirven como argumentos para diversas expresiones, dependiendo del formato de origen. Para buscar el contenido de los campos de texto, coloque el contenido entre comillas. Los enteros (Integer) se escriben sin comillas, al igual que los números de coma flotante.

Los campos representan los valores que hay en esos campos en la tabla. Por lo general, los campos se comparan entre sí o con valores específicos. En SQL, los nombres de campo deben colocarse entre comillas dobles, ya que de lo contrario no se reconocerán correctamente. Por lo general, SQL supone que el texto sin comillas dobles no tiene caracteres especiales, es decir, una sola palabra sin espacios y en mayúsculas.

Si hay varias tablas en la consulta, un campo se debe identificar con el nombre de la tabla seguido de un punto y el nombre del campo.

`TRUE` y `FALSE` generalmente derivan de los campos Sí/No.

`NULL` significa que no hay contenido. No es lo mismo que 0, sino que está «vacío».

UNION [ALL | DISTINCT] Resultado_Consulta

Esto vincula las consultas para que el contenido de la segunda consulta se escriba debajo de la primera. Para que esto funcione, todos los campos en ambas consultas deben coincidir en tipo. Este enlace de varias consultas funciona solo en el modo de órdenes SQL directas.

```
SELECT "Nombre"
FROM "Tabla1"
UNION DISTINCT
   SELECT "Nombre"
   FROM "Tabla2";
```

Esta consulta devuelve todos los nombres de *Tabla1* y *Tabla2*; el término adicional `DISTINCT` significa que no se mostrarán nombres duplicados. `DISTINCT` es el valor predeterminado en este contexto. Por defecto, los nombres se ordenan alfabéticamente en orden ascendente. `ALL` hace que se muestren todos los campos *Nombre* de *Tabla1*, seguidos de los campos *Nombre* de *Tabla2*. En este caso, se ordena por la clave primaria de manera predeterminada.

El uso de esta técnica de consulta permite enumerar consecutivamente valores de un registro tras otro en una columna. Supongamos que tiene una tabla llamada *Stock* en la que hay campos para *Precio_venta*, *Precio_Rebaja_1* y *Precio_Rebaja_2.* A partir de esto, se desea calcular un campo de combinación que enumerará estos precios directamente uno debajo del otro:

```
SELECT
   "Precio_venta"
FROM "Stock" WHERE "Stock_ID" = 1
   UNION
   SELECT
      "Precio_Rebaja_1"
   FROM "Stock" WHERE "Stock_ID" = 1
   UNION
   SELECT
      "Precio_Rebaja_2"
   FROM "Stock" WHERE "Stock_ID" = 1;
```

La clave principal para la tabla *Stock* naturalmente tiene que establecerse mediante una macro, ya que el campo de combinación tendrá una entrada coincidente.

MINUS [DISTINCT] | EXCEPT [DISTINCT] Resultado_consulta

```
SELECT "Nombre"
```

```
FROM "Tabla1"
EXCEPT
   SELECT "Nombre"
   FROM "Tabla2";
```

Muestra todos los nombres de *Tabla1*, excepto los nombres que figuran en *Tabla2*. MINUS y EXCEPT conducen al mismo resultado. La clasificación es alfabética.

INTERSECT [DISTINCT] Resultado_consulta

```
SELECT "Nombre"
FROM "Tabla1"
INTERSECT
   SELECT "Nombre"
   FROM "Tabla2";
```

Esto mostrará los primeros nombres que aparecen en ambas tablas. La ordenación es nuevamente alfabética. Actualmente esto solo funciona en modo de orden SQL directa.

[ORDER BY Criterio_de_Ordenación [, ...]]

Expresión puede ser un nombre de campo, un número de columna (empezando a contar por 1 desde la izquierda), un alias (formulado con AS, por ejemplo) o una expresión de valor compuesto (consulte *[Expresión SQL]: valores*). El orden de clasificación es ascendente (ASC). Si desea una ordenación descendente, debe especificar DESC.

```
SELECT "Nombre", "Apellido" AS "Nombre Completo"
FROM "Tabla1"
ORDER BY "Apellido";
```

devuelve el mismo resultado que:

```
SELECT "Nombre", "Apellido" AS "Nombre Completo"
FROM "Tabla1"
ORDER BY 2;
```

al igual que:

```
SELECT "Nombre", "Apellido" AS "Nombre Completo"
FROM "Tabla1"
ORDER BY "Nombre Completo";
```

Usar un alias en una consulta

Las consultas pueden devolver un resultado con nombres de campos cambiados.

```
SELECT "First_name", "Surname" AS "Name"
FROM "Table1"
```

El campo *Surname* pasa a llamarde *Name* en el resultado.

Si una consulta involucra dos tablas, cada nombre de campo debe estar precedido por el nombre de la tabla:

```
SELECT "Table1"."First_name", "Table1"."Surname" AS "Name", "Table2"."Class"
FROM "Table1", "Table2"
WHERE "Table1"."Class_ID" = "Table2"."ID"
```

El nombre de la tabla también puede tener un alias, pero no se reproducirá en la vista de tabla. Si se establece dicho alias, todos los nombres de tabla en la consulta deben modificarse en consecuencia:

```
SELECT "a"."Nombre", "a"."Apellido" AS "Nombre Completo", "b"."Clase"
```

```
FROM "Tabla1" AS "a", "Tabla2" AS "b"
WHERE "a"."Class_ID" = "b"."ID"
```

La asignación de un alias para una tabla se puede llevar a cabo más brevemente sin usar el término AS:

```
SELECT "a"."Nombre", "a"."Apellido" "Nombre Completo", "b"."Clase"
FROM "Tabla1" "a", "Tabla2" "b"
WHERE "a"."Class_ID" = "b"."ID"
```

Sin embargo, esto hace que el código sea menos legible. Por lo que se recomienda usar la forma abreviada solo en circunstancias excepcionales.

Nota

Desafortunadamente, el código para el editor de consultas en la interfaz gráfica de usuario se ha cambiado recientemente para que las designaciones de alias se creen sin usar el prefijo AS. Este cambio se debe a que al utilizar bases de datos externas, cuyo método de especificar alias no es predecible, la inclusión de AS ha dado lugar a mensajes de error.

Si se edita el código de la consulta utilizando la interfaz (no directamente en SQL) los alias existentes pierden su prefijo AS. Si se quiere evitar esto, las consultas siempre se tienen que editar en la *vista SQL*.

Un nombre de alias también permite usar una tabla con el filtrado correspondiente más de una vez dentro de una consulta:

```
SELECT "CuentasCaja"."Saldo", "Cuenta"."Fecha",
   "a"."Saldo" AS "Actual",
   "b"."Saldo" AS "Deseado"
FROM "Cuenta"
   LEFT JOIN "Cuenta" AS "a"
   ON "Cuenta"."ID" = "a"."ID" AND "a"."Saldo" >= 0
   LEFT JOIN "Cuenta" AS "b"
   ON "Cuenta"."ID" = "b"."ID" AND "b"."Saldo" < 0
```

Consultas para la creación de campos en Listado

Los controles de *Listado*,generalmente, muestran un valor que no se corresponde con el contenido de la tabla subyacente. Se utilizan para mostrar el valor asignado a una clave externa en lugar de la clave en sí. El valor que finalmente se guarda en el formulario no debe aparecer en la primera posición de los campos del *Listado*.

```
SELECT "Nombre", "ID"
FROM "Tabla1";
```

Esta consulta mostrará todos los valores de *Nombre* y los valores de *ID* de la clave principal que proporciona la tabla subyacente del formulario. Por supuesto, aún no es óptimo. Los nombres aparecen sin clasificar y, en el caso de nombres idénticos, es imposible determinar a qué persona hace referencia.

```
SELECT "Nombre"||' '||"Apellido", "ID"
FROM "Tabla1"
ORDER BY "Nombre"||' '||"Apellido";
```

Ahora aparecen *Nombre* y *Apellido*, separados por un espacio. Los nombres se vuelven distinguibles y también se ordenan. Pero el orden sigue la lógica habitual de comenzar con la

primera letra de la cadena, por lo que se ordena por *Nombre*, y después por *Apellido*. Un orden diferente al de los campos mostrados sería confuso.

```
SELECT "Apellido"||', '||"Nombre", "ID"
FROM "Tabla1"
ORDER BY "Apellido"||', '||"Nombre";
```

Esto, ahora, muestra a una ordenación que se corresponde mejor con la manera habitual de ordenar. Los miembros de una familia aparecen juntos, uno debajo del otro; sin embargo, estarían intercalados miembros de otras familias con el mismo apellido. Para distinguirlos, necesitaríamos agruparlos de manera diferente dentro de la consulta.

Hay un problema final: si dos personas tienen el mismo apellido y nombre, no podrán distinguirse. Una solución podría ser usar un sufijo en el nombre. ¡Pero, imagine cómo se vería un saludo dirigido al Sr. «Müller II»! Una opción más acertada sería usar su *ID*.

```
SELECT "Apellido"||', '||"Nombre"||' - Id:'||"ID", "ID"
FROM "Tabla1"
ORDER BY "Apellido"||', '||"Nombre"||"ID";
```

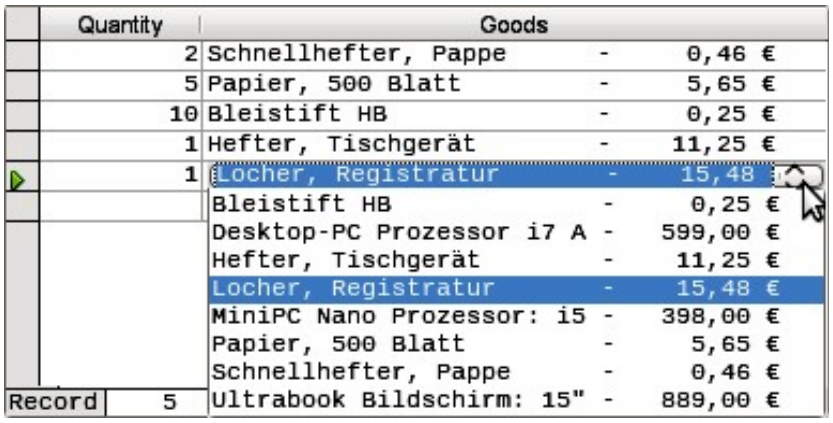
Aquí todos los registros son distinguibles. Lo que realmente se muestra es: «Muller, Henry - Id: 2» (poniendo como ejemplo al Sr. Henry Muller con un ID de valor 2).

En el formulario *Loan* de la base de ejemplo *Media*, hay un control de listado que debe mostrar solo los artículos que aún no se han prestado. Se crea usando la siguiente fórmula SQL:

```
SELECT "Title" || ' - Nr. ' || "ID", "ID"
FROM "Media"
WHERE "ID" NOT IN
   (SELECT "Media_ID"
   FROM  "Loan"
   WHERE "Return_Date" IS  NULL)
ORDER BY  "Title" || ' - Nr. ' || "ID"  ASC
```

Es importante que este control de listado se actualice siempre que un artículo dentro de él sale en préstamo.

En el siguiente ejemplo el control de tabla utiliza controles de listado que muestran el contenido de varios campos para una selección de artículos y su precio.



Para hacer que su representación funcione adecuadamente y estén alineados los campos que aparecen en el listado primero se debe elegir una fuente de ancho fijo adecuada. Se puede utilizar «Courier» o cualquier fuente «monotype» como «Liberation Mono».

Se crea el control de tabla y en sus datos se usa el código SQL:

```
SELECT LEFT("Stock"||SPACE(25),25) || ' - ' || RIGHT(SPACE(8)||"Price",8) || ' €',"ID"
FROM "Stock"
```

```
ORDER BY ("Stock" || ' - ' || "Price" || ' €') ASC
```

El contenido del campo Stock se ha rellenado con espacios para que toda la cadena tenga una longitud mínima de 25 caracteres para asegurar que no se trunque una cantidad alta de stock. Luego se colocan las primeras 25 letras de la descripción del artículo y se corta el excedente.

Cuando el contenido del campo de lista contiene caracteres que no se imprimen (como líneas nuevas) se vuelve más complejo. Se debe adaptar el código para que el texto se ajuste al control.

```
SELECT LEFT(REPLACE("Stock",CHAR(10),' ')||SPACE(25), 25) || ' - ' || …
```

Esto reemplaza una línea (`CHAR10` en Linux) con un espacio. Para Windows, se debe usar `CHAR(13)` para eliminar el retorno de carro.

La cantidad de espacios necesarios también se determina en función de la consulta.

```
SELECT LEFT("Stock"||SPACE((SELECT MAX(LENGTH("Stock"))
FROM "Stock")),(SELECT MAX(LENGTH("Stock")) FROM "Stock"))|| ' - ' || RIGHT('
'||"Price",8) || ' €', "ID"
FROM "Stock"
ORDER BY ("Stock" || ' - ' || "Price" || ' €') ASC
```

Como el precio se debe mostrar justificado a la derecha, para separarlo del artículo se rellena con espacios a la izquierda del campo y además se coloca un máximo de ocho caracteres desde la derecha. La representación seleccionada funcionará para cualquier precio hasta 99999,99€.

Si desea reemplazar el punto decimal con una coma en SQL, necesitará un código adicional:

```
REPLACE(RIGHT('    '||"Price",8),'.',',')
```

Consultas como base para información adicional en formularios

Si desea que un formulario muestre una información adicional que de otro modo no sería visible, existen varias posibilidades de consulta. Lo más sencillo es recuperar esta información con consultas independientes e insertar los resultados en el formulario.

La desventaja de este método es que los cambios en los registros pueden afectar al resultado de la consulta, y desafortunadamente estos cambios no se muestran de manera automática.

Aquí hay un ejemplo del control de existencias para un pago.

La tabla de la factura contiene las claves foráneas del total de artículos y un número de identificación de la factura. El comprador tiene muy poca información si no se imprime la factura una información más detallada (resultado de una consulta).

Los artículos se identifican solo mediante la lectura de un código de barras. Sin una consulta aplicada, el formulario solo muestra la información de la siguiente tabla:

Total	***C_Barras***
3	17
2	24

Lo que está oculto detrás de estos números no se puede hacer visible usando un *Listado*, ya que la clave externa se ingresa directamente usando el código de barras. Tampoco es posible usar un *Listado* junto al artículo para mostrar al menos el precio unitario.

Una consulta puede ayudar:

```
SELECT "Checkout"."Receipt_ID", "Checkout"."Total", "Stock"."Item", "Stock"."Unit_Price",
"Checkout"."Total"*"Stock"."Unit_price" AS "Total_Price"
FROM "Checkout", "Item"
```

```
WHERE "Stock"."ID" = "Checkout"."Item_ID";
```

Ahora, al menos después de haber ingresado la información, sabemos cuánto se debe pagar por los 3 artículos de la referencia 17. Solo la información relevante para el número de factura (*Receipt_ID)* debe filtrarse a través del formulario. Lo que ahora falta es el total del precio de la factura.

```
SELECT "Checkout"."Receipt_ID", SUM("Checkout"."Total"*"Stock"."Unit_price") AS "Sum"
FROM "Checkout", "Stock"
WHERE "Stock"."ID" = "Checkout"."Item_ID"
GROUP BY "Checkout"."Receipt_ID";
```

Diseñe el formulario para mostrar un registro de la consulta cada vez. Dado que la consulta se agrupa por *Receipt_ID*, el formulario muestra información sobre un cliente cada vez.

Consejo

Si un formulario debe mostrar valores de fecha que dependan de otra fecha (por ejemplo, en una biblioteca, un período de préstamo podría ser de 21 días, ¿cuál es la fecha de devolución?), No puede utilizar las funciones integradas de HSQLDB porque no hay una función «DATEADD».

La consulta: `SELECT "Date", DATEDIFF('dd','1899-12-30',"Date")+21 AS "ReturnDate" FROM "Table"`

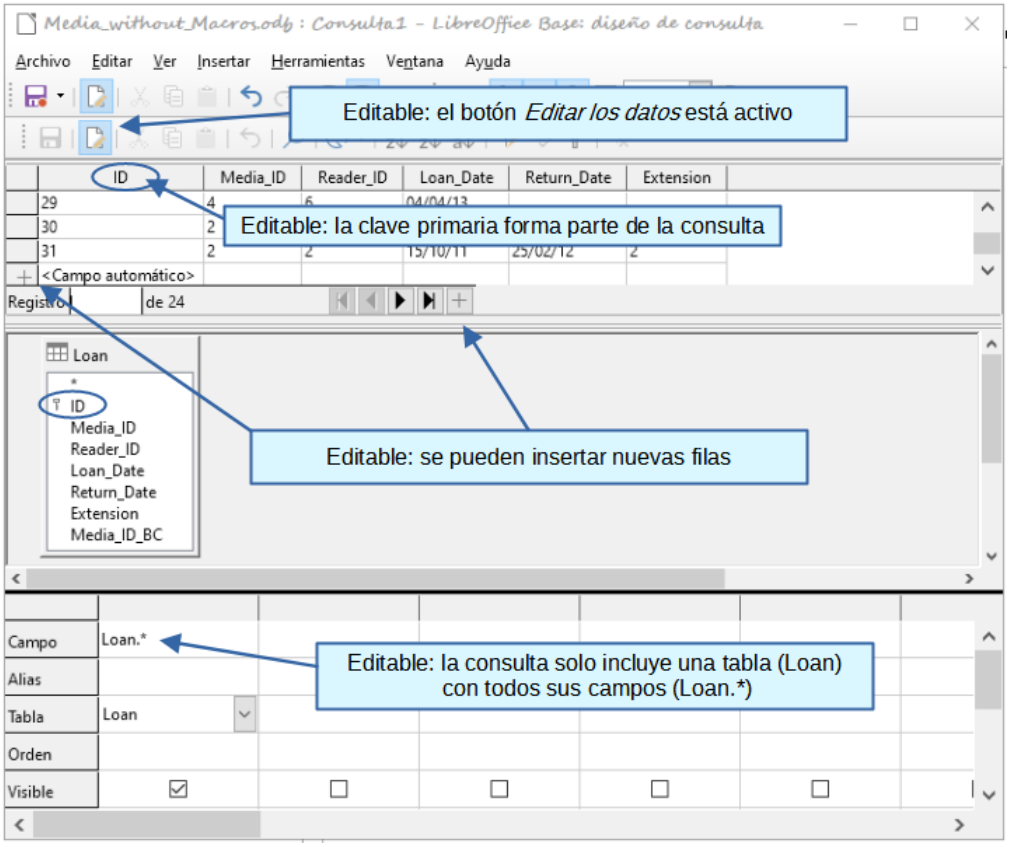
Devolverá la fecha correcta para la devolución en un formulario. Esta consulta cuenta los días del 30.12.1899. (Fecha predeterminada, que Calc también usa como valor 0)

El valor devuelto solo es un número y no una fecha que se pueda utilizar en otra consulta.

El número devuelto no es adecuado para su uso en consultas porque el formato de las consultas no se guarda. Necesita crear una vista en su lugar.

Posibilidades de introducción de datos en consultas

Para introducir datos en una consulta, la clave principal de la tabla utilizada en la consulta debe estar presente. Esto también se aplica a una consulta que relaciona varias tablas: todas las claves primarias de las tablas implicadas deben estar presentes.



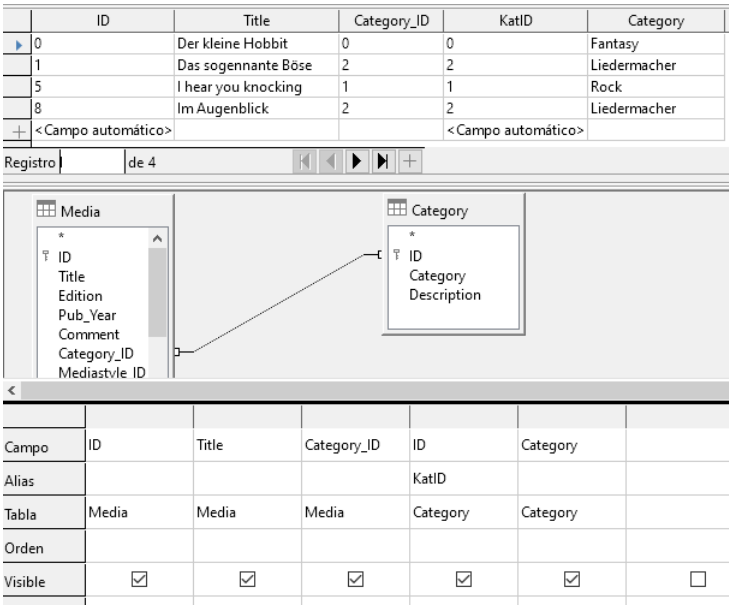
En el préstamo de artículos, no tiene sentido mostrar a un lector elementos que ya han sido devueltos hace algún tiempo.

```
SELECT "ID", "Reader_ID", "Media_ID", "Loan_Date"
FROM "Loan"
WHERE "Return_Date" IS NULL;
```

De esta manera, un formulario puede mostrar dentro de un campo de control de tabla todos los artículos que un lector en particular ha tomado prestados durante su pertenencia a la biblioteca. La consulta también debe filtrarse utilizando una estructura de formulario adecuada (lector en el formulario principal, consulta en el subformulario), de modo que solo se muestren los artículos que realmente tiene prestados.

La consulta es adecuada para introducir datos, ya que la clave principal se incluye en la consulta.

Una consulta no se puede editar si consta de más de una tabla y se accede a las tablas mediante un alias. En ese caso, no influye el que la clave principal esté o no presente en la consulta.

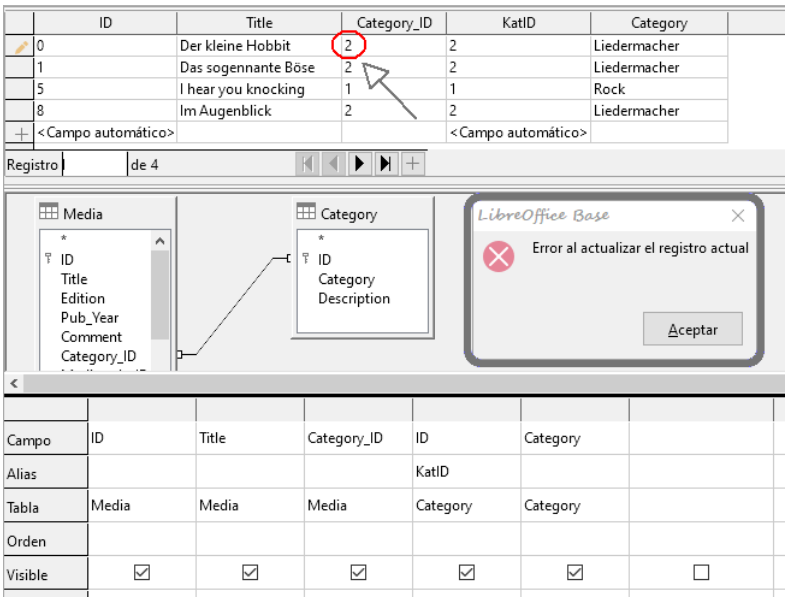



```
SELECT "Media"."ID", "Media"."Title", "Media"."Category_ID", "Category"."ID" AS "katID",
"Category"."Category"
FROM "Media", "Category"
WHERE "Media"."Category_ID" = "Category"."ID";
```

Esta consulta se puede editar, ya que ambas claves principales están incluidas y se puede acceder a ellas en las tablas (que no tienen un alias de tabla). Una consulta que vincula varias tablas juntas también requiere que todas las claves primarias estén presentes.

En una consulta que involucra varias tablas, no es posible alterar un campo de clave externa de una tabla que se refiere a un registro de otra tabla.

En el registro que se muestra a continuación, se intentó cambiar la categoría del título «Der kleine Hobbit». El campo *Category_ID* se cambió de 0 a 2. El cambio pareció realizarse y la nueva categoría apareció en el registro. Sin embargo, resultó imposible guardarlo.

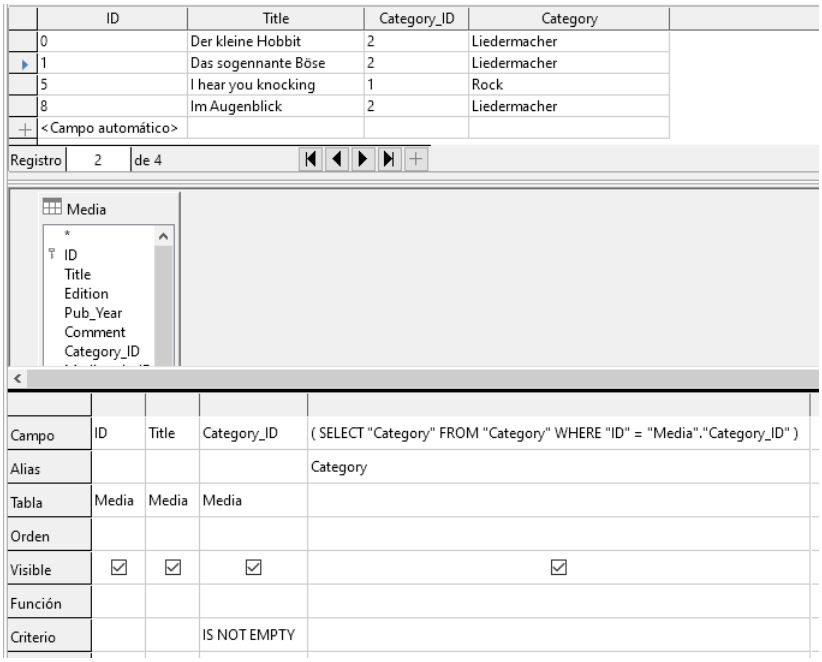


En cambio, sí que es posible editar el contenido de un registro perteneciente a *Category* . Por ejemplo, para reemplazar «Fantasy» por «Liedermacher». Se debe tomar en cuenta que el nombre de la categoría se modificará para todos los registros vinculados con esta categoría.

La siguiente consulta accede a la tabla de artículos usando un alias. La consulta no se puede editar.

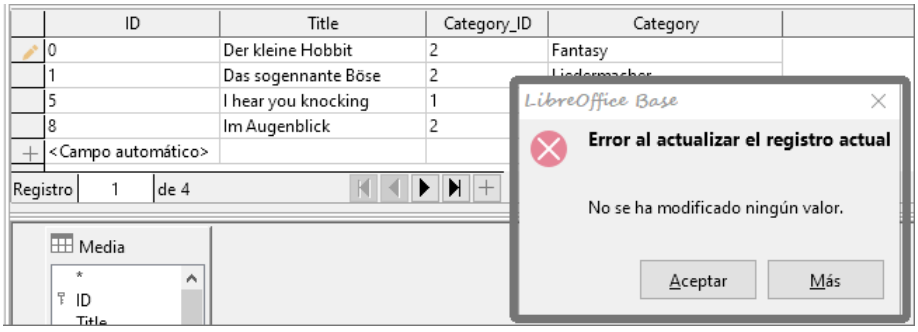
```
SELECT "m"."ID", "m"."Title", "Category"."Category", "Category"."ID" AS "katID"
FROM "Media" AS "m", "Category"
WHERE "m"."Category_ID" = "Category"."ID";
```

En el ejemplo anterior, este problema se puede evitar, si realmente necesita usar un alias de tabla, utilizando una subconsulta relacionada (consulte la página 232). En ese caso, una consulta solo se puede editar si unicamente contiene una tabla en la consulta principal.



En la vista de diseño solo aparece una tabla. La tabla *Media*. En la consulta se ha empleado un alias «m» para la tabla y acceder al contenido del campo *Category_ID* utilizando una subconsulta relacionada.

En una consulta como esta, ahora es posible cambiar el campo de un registro (de clave externa *Category_ID*) a otra categoría. En el ejemplo anterior, el campo *Category_ID* del artículo «Der Kleine Hobbit» se cambia de 0 a 2. El artículo pertenecerá ahora a la categoría «Liedermacher».



Sin embargo, ahora ya no es posible cambiar un valor en el campo que ha recibido su contenido a través de una subconsulta relacionada. Se muestra un intento de volver el registro con título «Der Kleine Hobbit» a su categoría anterior, este cambio no se registra y tampoco se puede guardar. En la tabla que muestra la vista de diseño, el campo *Category* no está presente porque pertenece a la subconsulta.

Uso de parámetros en consultas

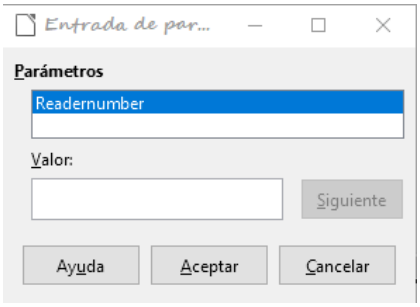
Si a menudo se usa la misma consulta básica pero en determinados momentos su utilizan valores diferentes, se pueden crear consultas con parámetros. En principio, las consultas con parámetros funcionan igual que las consultas para un subformulario:

```
SELECT "ID", "Reader_ID", "Media_ID", "Loan_Date"
FROM "Loan"
WHERE "Return_Date" IS NULL AND "Reader_ID"=2;
```

Esta consulta muestra solo los artículos prestados al lector con *ID 2*.

```
SELECT "ID", "Reader_ID", "Media_ID", "Loan_Date"
FROM "Loan"
WHERE "Return_Date" IS NULL AND "Reader_ID" = :Readernumber;
```

Con esta modificación, cuando ejecute la consulta, aparecerá un diálogo que le pedirá un valor (número de lector). Cuando se introduzca un valor, se mostrarán los artículos actualmente prestados a ese lector.




```
SELECT
    "Loan"."ID",
    "Reader"."LastName"||', '||"Reader"."FirstName",
    "Loan"."Media_ID",
    "Loan"."Loan_date"
FROM "Loan", "Reader"
WHERE "Loan"."Return_Date" IS NULL
    AND "Reader"."ID" = "Loan"."Reader_ID"
    AND "Reader"."LastName" LIKE '%' || :Readername || '%'
ORDER BY "Reader"."LastName"||', '||"Reader"."FirstName" ASC;
```

Esta otra opción es claramente más fácil de usar que la anterior. Ya no es necesario saber el número del lector. Solo necesita ingresar parte del apellido en el control de listado y se muestran todos los artículos prestados a los lectores coincidentes.

Con otra pequeña modificación. si se reemplaza

```
"Reader"."LastName" LIKE '%' || :Readername || '%'
```

por

```
LOWER("Reader"."LastName") LIKE '%' || LOWER( :Readername ) || '%'
```

Ya no importa si el nombre se ingresa en mayúscula o minúscula.

Si el parámetro en la consulta anterior se deja vacío, todas las versiones de LibreOffice hasta 4.4 mostrarán a todos los lectores, ya que a partir de la Versión 4.4 un campo de parámetro vacío se lee como NULL en lugar de una cadena vacía. Si no desea este comportamiento, debe usar un truco para evitarlo:

```
LOWER ("Reader"."LastName")
```

```
LIKE '%' || IFNULL( NULLIF ( LOWER (:Readername), '' ), '§§' ) || '%'
```

El campo de parámetro vacío devuelve una cadena vacía y no un NULL a la consulta. Por lo tanto, al campo de parámetro vacío se le debe asignar la propiedad NULL usando NULLIF. Luego, dado que la entrada del parámetro ahora produce NULL, se puede restablecer a un valor que normalmente no aparece en ningún registro. En el ejemplo anterior, esto es '$$'. Este valor, por supuesto, no se encontrará en la búsqueda.

A partir de la versión 4.4, son necesarias adaptaciones a esta técnica de consulta:

```
LOWER ("Reader"."LastName") LIKE '%' || LOWER (:Readername) || '%'
```

al no introducir ningún valor, la combinación:

```
 '%' || LOWER (:Readername) || '%'
```

devuelve inevitablemente el valor NULL.

Para evitar esto, agregue una condición adicional, que para un campo vacío, todos los valores se muestren realmente:

```
(LOWER ("Reader"."LastName") LIKE '%' || LOWER (:Readername) || '%' OR :Readername
IS NULL)
```

Todo esto debe ponerse entre paréntesis. Primero se busca un nombre, si el campo está vacío (NULL de LibreOffice 4.4), se aplicará la segunda condición.

Cuando se usan formularios, el parámetro se puede pasar del formulario principal a un subformulario. Sin embargo, a veces puede que no se actualicen las consultas que usan parámetros en subformularios si los datos se cambian o se ingresan nuevamente.

Sería aconsejable alterar el contenido de los controles de listado utilizando cierta configuración del formulario principal. Por ejemplo, podríamos evitar que los artículos de la biblioteca se presten a determinados lectores que están bloqueados. Lamentablemente, no es posible controlar esta configuración del control de listado mediante el uso de parámetros.

Subconsultas

Las subconsultas integradas en campos solo pueden devolver un único registro. El campo también devuelve solo un valor.

```
SELECT "ID", "Income", "Expenditure",
   ( SELECT SUM( "Income" ) - SUM( "Expenditure" )
      FROM "Checkout") AS "Balance"
FROM "Checkout";
```

Esta consulta permite la entrada de datos (clave principal incluida). La subconsulta arroja precisamente un único valor: el saldo total. Esto permite leer el saldo en la caja registradora después de cada entrada.

Esto todavía no es comparable con el formulario de pago del supermercado descrito en "Consultas como base para información adicional en formularios" en la página 225. Naturalmente, carece de los cálculos individuales de Total*Unit_price, pero también de la presencia del número de factura (Receipt_ID) Solo se da la suma total. Se puede incluir al menos el número de factura mediante el uso de un parámetro de consulta:

```
SELECT "ID", "Income", "Expenditure",
   ( SELECT SUM( "Income" ) - SUM( "Expenditure" )
      FROM "Checkout"
      WHERE "Receipt_ID" = :Receipt_Number) AS "Balance"
FROM "Checkout" WHERE "Receipt_ID" = :Receipt_Number;
```

En una consulta con parámetros, el parámetro debe ser el mismo en ambas declaraciones de consulta si se va a reconocer como un parámetro.

Se pueden incluir dichos parámetros en subformulario. El subformulario recibe, el nombre del parámetro correspondiente en lugar de un nombre de campo. Este enlace solo se puede especificar en las propiedades del subformulario, no se puede especificar cuando se usa el asistente.

Nota

Los subformularios basados en consultas no se actualizan automáticamente en función de sus parámetros. Es más apropiado pasar el parámetro directamente desde el formulario principal.

Subconsultas relacionadas

Usando una consulta aún más refinada, una consulta editable le permite incluso llevar el saldo corriente para la caja registradora:

```
SELECT "ID", "Income", "Expenditure",
( SELECT SUM( "Income" ) - SUM( "Expenditure" )
   FROM "Checkout"
   WHERE "ID" <= "a"."ID" ) AS "Balance"
FROM "Checkout" AS "a"
ORDER BY "ID" ASC
```

La tabla de la factura *Checkout* tiene un alias (AS "a"). “a” solo devuelve la relación con los valores actuales en este registro. De esta manera, el valor actual de *ID* de la consulta externa se puede evaluar dentro de la subconsulta. Por lo tanto, dependiendo de la *ID*, se determina el saldo anterior en el momento correspondiente, si se cuenta con el hecho de que la *ID*, que es un valor automático, se incrementa por sí misma.

Nota

Si la subconsulta se va a filtrar por contenido utilizando la función de filtro del editor de consultas, solo funciona si usan paréntesis dobles en lugar de sencillos al principio y al final de la subconsulta: ((SELECT ….)) AS "Balance"

Consultas como tablas fuente para consultas

Se necesita una consulta para establecer un bloqueo a los lectores que hayan recibido un tercer aviso de retraso para un artículo.

```
SELECT "Loan"."Reader_ID",'3rd Overdue–the reader is blacklisted' AS "Lock" FROM (SELECT
COUNT("Date") AS "Total_Count", "Loan_ID"
      FROM "Recall" GROUP BY "Loan_ID") AS "a", "Loan"
WHERE "a"."Loan_ID" = "Loan"."ID" AND "a"."Total_Count" > 2
```

Primero examinemos la consulta interna, con la que se relaciona la consulta externa:

```
(SELECT COUNT("Date") AS "Total_Count", "Loan_ID"   FROM "Recall" GROUP BY "Loan_ID") AS
"a"
```

En esta consulta, se determina el número de entradas de *Date* agrupadas por la clave externa *Loan_ID*. Esto no debe hacerse dependiente de *Reader_ID*, ya que a la hora del recuento devolvería no solo tres avisos de vencimiento para un solo artículo, sino también tres artículos con

un aviso de vencimiento cada uno. La consulta interna recibe un alias (*a*) para que se pueda vincular con el *Reader_ID* desde la consulta externa.

La consulta externa se relaciona en este caso solo con la fórmula condicional de la consulta interna. Muestra solo un *Reader_ID* y el texto para el campo *Lock* cuando "Loan"."ID" y "a"."Loan_ID" son iguales y *"a"* es mayor que 2 ("Total_Count" > 2).

En principio, todos los campos de la consulta interna están disponibles para la externa. Entonces, la suma "a"."Total_Count" se puede combinar en la consulta externa para dar el total de multas reales.

Este tipo de construcciones son válidas en *vista SQL* y se deben formular en el editor de consultas SQL, pero es posible que al intentar volver a la *vista Diseño* de consulta no se pueda mostrar, recibirá el mensaje de la figura 175, pero se puede guardar y es totalmente operativo. El modo de vista de diseño no puede encontrar el campo contenido en la consulta interna *Loan_ID*, que gobierna la relación entre las consultas internas y externas.

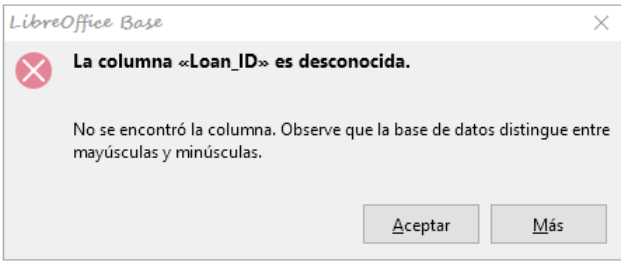
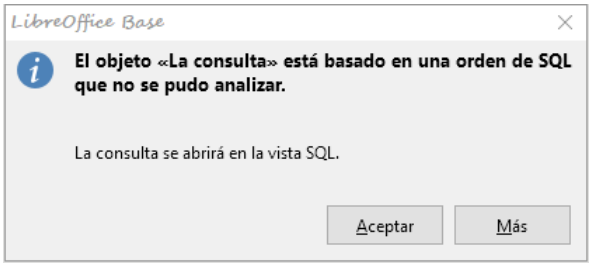


Figura 175: Mensaje de error

Al cerrar la consulta e intentar abrirla posteriormente recibirá el siguiente mensaje:



Si luego abre la consulta para editarla en la *vista SQL* e intenta cambiar de allí a la *Vista de diseño*, recibirá el mismo mensaje de error de la figura 175.

Cuando la consulta se ejecuta en modo SQL, el contenido correspondiente de la subconsulta se reproduce sin error. Por lo tanto, no necesita usar el modo SQL directo en este caso.

La consulta externa utiliza los resultados de la consulta interna para producir los resultados finales. Esta es una lista de los valores de *Loan_ID* que deben ser bloqueados y por qué. Si desea limitar aún más los resultados finales, utilice las funciones de ordenación y filtro de la interfaz gráfica de usuario.

Las siguientes capturas de pantalla muestran otra forma de actuación de una consulta con subconsultas.

En este caso, una consulta a una base de datos de *Stock* está tratando de determinar lo que el cliente debe pagar en la caja registradora. Los precios individuales se multiplican por el número de artículos comprados dando un subtotal. La suma de estos subtotales necesita ser determinada. Todo esto debe poderse editar para que la consulta se pueda utilizar como base de un formulario.

	ID	quantity	articles_ID	articleID	price	subtotal
	44	1	6	6	$398.00	398
	45	10	2	2	$0.46	4,6
	46	1	5	5	$889.00	889
	47	1	6	6	$398.00	398
	48	1	7	7	$599.00	599
+	<Campo automá			<Campo automático>		

Registro 1 de 49

```
SELECT
    "sale"."ID",
    "sale"."quantity",
    "sale"."articles_ID",
    "articles"."ID" AS "articleID",
    "articles"."price",
    "quantity" * "price" AS "subtotal"
FROM "sale",
    "articles"
WHERE "sale"."articles_ID" = "articles"."ID"
```

Figura 176: Consulta usando dos tablas. Para hacerla editable, se deben incluir ambas claves principales.

Nota

Debido a un error (61871), Base no actualiza el resultado parcial automáticamente.

	ID	quantity	articles_ID	articleID	price	subtotal
	44	1	6	6	$398.00	398
	45	10	2	2	$0.46	4,6
	46	1	5	5	$889.00	889
	47	1	6	6	$398.00	398
▶	48	1	7	7	$599.00	599
+	<Campo automá			<Campo automático>		

Registro 49 de 49

```
SELECT
    "sale"."ID",  "sale"."quantity",
    "sale"."articles_ID","articles"."price",
    "quantity" * "price" AS "subtotal"
FROM "sale",
        ( SELECT
            "ID",
            "price"
        FROM "articles")
    AS "articles"
WHERE "sale"."articles_ID" = "articles"."ID"
```

Figura 177: La tabla articles se mueve a una subconsulta, que se crea en el área de la tabla (después de FROM) y se le asigna un alias. La clave principal de la tabla articles no es estrictamente necesaria para que la consulta sea editable.

Figura 178: La suma calculada debe aparecer en la consulta. La consulta sencilla para la calcular la suma ya no es editable, por que esta agrupada y realiza una suma.

Figura 179: Con la segunda subconsulta, lo aparentemente imposible se vuelve posible. La consulta anterior se inserta como una subconsulta en la definición de tabla de la consulta principal (después de "FROM"). Como resultado, toda la consulta permanece editable. En este caso, las entradas solo son posibles en las columnas "quantity" y "articles_ID". Esto se aclara posteriormente en el formulario de consulta.

Resumiendo datos con consultas

Cuando se buscan datos en una base de datos completa, el uso de las funciones de formulario sencillo a menudo genera problemas. Después de todo, un formulario se refiere a una sola tabla, y la función de búsqueda se mueve solo a través de los registros subyacentes para este formulario.

Obtener todos los datos es más fácil mediante el uso de consultas que puedan proporcionar una imagen de todos los registros. La sección «Definir relaciones en la consulta» sugiere una construcción de consulta de este tipo. Esto se construye para la base de datos de ejemplo de la siguiente manera:

```
SELECT "Media"."Title", "Subtitle"."Subtitle", "Author"."Author"
FROM "Media"
   LEFT JOIN "Subtitle"
      ON "Media"."ID" = "Subtitle"."Media_ID"
   LEFT JOIN "rel_Media_Author"
      ON "Media"."ID" = "rel_Media_Author"."Media_ID"
   LEFT JOIN "Author"
      ON "rel_Media_Author"."Author_ID" = "Author"."ID"
```

Aquí todos los títulos, subtítulos y autores se muestran juntos.

La tabla *Media* contiene un total de 9 títulos. Para dos de estos títulos, hay un total de 8 subtítulos. Sin una instrucción LEFT JOIN, ambas tablas que se muestran juntas generarían solo 8 registros. Para cada subtítulo, se busca el título correspondiente y ese es el final de la consulta. Los títulos sin subtítulos no se mostrarían.

Para mostrar todos los artículos, incluidos aquellos sin subtítulos: los artículos están en el lado izquierdo de la tarea, los subtítulos en el lado derecho. Un LEFT JOIN mostrará todos los títulos de los artículos, pero solo un subtítulo para aquellos que tengan un título. Los artículos se convierten en la tabla decisiva para determinar qué registros se mostrarán. Esto ya estaba planeado cuando se construyó la tabla (vea «Capítulo 3, Tablas»). Como existen subtítulos para dos de los nueve títulos, la consulta ahora mostrará 9 + 8 - 2 = 15 registros.

Nota

La vinculación normal de las tablas, después de contar todas las tablas, sigue a la palabra clave WHERE. Si hay un LEFT JOIN o un RIGHT JOIN, la asignación se define directamente después de los dos nombres de tabla usando ON. La secuencia tiene que ser:

```
Table1 LEFT JOIN Table2 ON Table1.Field1 = Table2.Field1 LEFT JOIN Table3 ON
Table2.Field1 = Table3.Field1 ...
```

Dos títulos de la tabla de artículos aún no tienen una entrada de autor o un subtítulo. Al mismo tiempo, un título tiene un total de tres autores. Si la tabla del autor está vinculada sin un LEFT JOIN, no se mostrarán los dos artículos sin un autor. Pero como un artículo tiene tres autores en lugar de uno, el número total de registros mostrados seguirá siendo 15.

Solo al usar LEFT JOIN se le indicará a la consulta que use la tabla *Media* para determinar qué registros mostrar. Ahora los registros sin subtítulo o autor aparecen nuevamente, dando un total de 17 registros.

El uso de combinaciones apropiadas generalmente aumenta la cantidad de datos que se muestran. Este conjunto de datos ampliado puede explorar fácilmente, ya que se muestran los autores y los subtítulos además de los títulos. En la base de datos de ejemplo, se puede acceder a todas las tablas dependientes de los artículos.

Acceso más rápido a consultas mediante vistas de tabla

Las vistas en SQL son más rápidas que las consultas, especialmente para bases de datos externas, ya que están ancladas directamente en la base de datos y el servidor devuelve los resultados directamente. En el caso de consultas, estas se envían primero al servidor, el servidor las tiene que procesar y luego devolver los datos.

Si una consulta nueva se relaciona con otra consulta, la vista SQL en Base hace que la otra consulta se vea como una tabla. Si crea una Vista a partir de ella, puede ver que realmente está trabajando con una subconsulta (Selección utilizada dentro de otra Selección). Debido a esto, una *Consulta 2* que se relaciona con otra *Consulta 1* no se puede ejecutar mediante la orden **Editar > Ejecutar SQL** directamente, ya que solo la interfaz gráfica de usuario y no la base de datos en sí misma conoce la *Consulta 1*.

La base de datos no le da acceso directo a las consultas. Esto también se aplica al acceso mediante macros. Por otro lado, se puede acceder a las vistas desde macros y tablas. Sin embargo, no se pueden editar registros en una vista. (Deben editarse en una tabla o formulario).

Consejo

Una consulta creada con *Crear consulta en modo SQL* tiene la desventaja de que no se puede ordenar o filtrar con la interfaz. Por lo tanto, hay límites para su uso.

Una vista, por otro lado, se puede administrar en Base igual que una tabla normal, con la excepción de que no es posible cambiar los datos. De esta manera, todas las posibilidades para ordenar y filtrar están disponibles, incluso en órdenes SQL directas.

Además, el formato de las columnas en una vista se conserva cuando se cierra la base de datos, a diferencia de las columnas en una consulta.

Las vistas son una solución para muchas consultas si desea obtener algún resultado. Si, por ejemplo, se va a utilizar una Subselección en los resultados de una consulta, cree una Vista que le brinde estos resultados. Luego use la subselección en la Vista. Los ejemplos correspondientes se encuentran en el «Capítulo 8, Tareas de base de datos».

Crear una vista desde una consulta es bastante fácil y directo: (Debe tener creada una consulta que responda a sus necesidades para pasarla a modo vista).

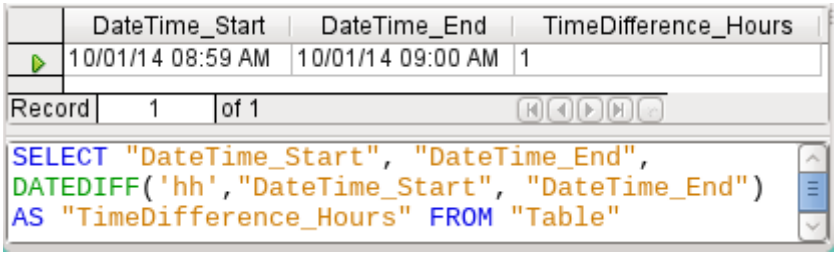
7) Haga clic en el objeto *Tablas* en la sección Base de datos.
8) Haga clic en *Crear vista*.
9) Cierre el diálogo Agregar tabla.
10) Haga clic en el icono de *Activar o desactivar la vista Diseño*. (Este es el modo SQL para una vista).
11) Obtenga el SQL de la consulta para crear la Vista:
    c) Edite la consulta creada en Vista SQL.
    d) Use *Control+A* para seleccionar todo el código el SQL de la consulta.
    e) Use *Control+C* para copiar ese código.
12) En el modo SQL de la vista que va a crear use *Control+V* para pegar el código SQL obtenido de la consulta.
13) Cierre la vista, (se le preguntará si quiere guardarla) asigne un nombre a la vista y guárdela.

Errores de cálculo en consultas

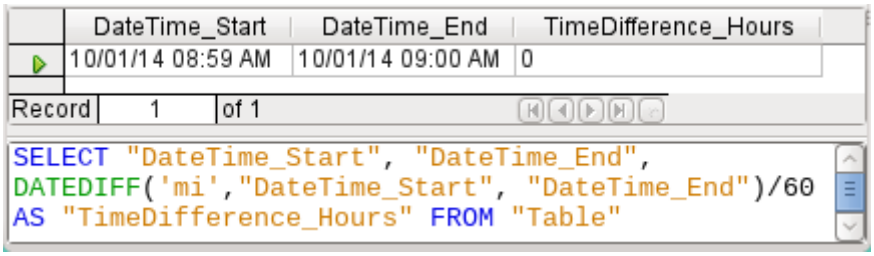
Las consultas también se utilizan para calcular valores. A veces, la base de datos interna HSQLDB produce errores aparentes, que en un examen más detallado resultan ser interpretaciones correctas (lógicamente) de los datos. También hay problemas de redondeo, que pueden causar confusión fácilmente.

Los datos horarios dentro de HSQLDB se formatean correctamente solo hasta una diferencia de 23:59:59 horas. Si se van a agregar varias veces, por ejemplo, para calcular las horas trabajadas, se debe encontrar otra forma. Aquí hay varios enfoques para resolver estas complicaciones:

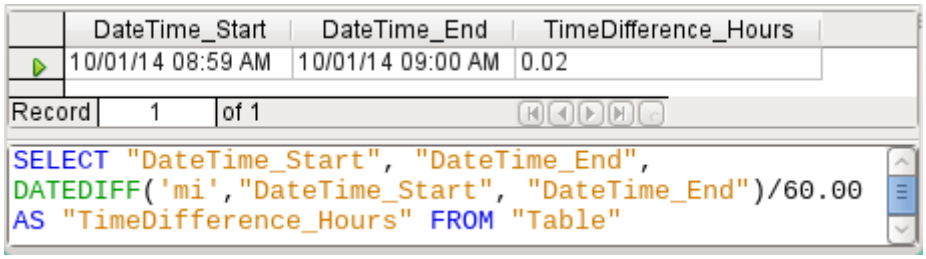
- Los datos horarios se expresan directamente solo como un total de minutos o incluso segundos. **Ventaja**: los valores permiten un cálculo posterior sin problemas.
- Los datos se dividen en horas, minutos y segundos, y se concatenan posteriormente como texto usando «:» como separador. **Ventaja**: el texto aparece en las consultas como hora con el formato correcto desde el principio.
- La datos se crean como un número decimal. Un día es 1, una hora es 1/24 y así sucesivamente. **Ventaja**: los valores pueden formatearse posteriormente como dato horario en la consulta y presentarse como un campo formateado de formulario.



DATEDIFF permite determinar los intervalos de tiempo. Pide explícitamente la diferencia que debe determinarse. En el ejemplo anterior, no se han solicitado minutos, pero se consideran todos los elementos que son mayores que un minuto. Eso da una hora (!) Como la diferencia entre 8:59 y 9:00. Por el contrario, una diferencia de fecha se calcula como una diferencia de tiempo en horas. Si, por ejemplo, el campo *Date_time_end* se establece en 2.10.14 09:00, eso se calcula como 25 horas.



Si, en cambio, el intervalo de tiempo se calcula en minutos y luego se divide por 60, la diferencia de tiempo en horas se convierte en cero. Esto se parece más al valor correcto, excepto: ¿a dónde fue ese minuto?



El intervalo de tiempo se ha dado como un entero. Un entero ha sido dividido por un entero. El resultado de la consulta en tales casos también debe ser un número entero, no un número

decimal. Esto se puede corregir fácilmente. La diferencia de tiempo en minutos se divide por un número decimal con dos lugares decimales (60.00). Esto da un resultado que también tiene dos decimales. 0.02 horas todavía no es exactamente un minuto, pero está mucho más cerca que antes. El número de lugares decimales podría aumentarse usando más ceros. Un período de 0.016 es una aproximación más cercana aún, pero los errores de cálculo posteriores no siempre pueden excluirse.

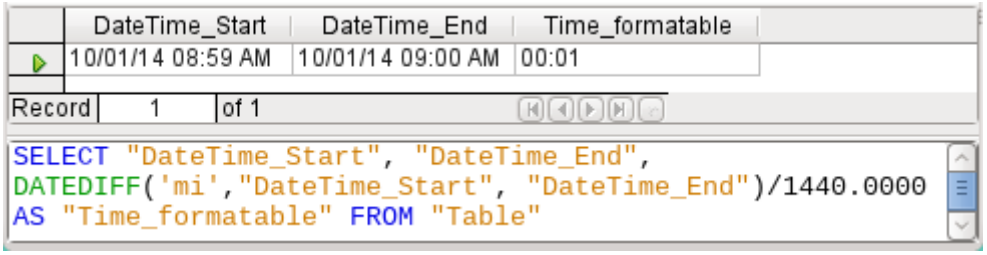
En lugar de tener que trabajar con muchos ceros agregados, se puede influir en el tipo de datos de DATEDIFF directamente. Usando:

```
(CONVERT(DATEDIFF('mi',"Date_time_start","Date_time_end"), DECIMAL(50,49)))/60
```

puede lograr una precisión de 49 decimales.

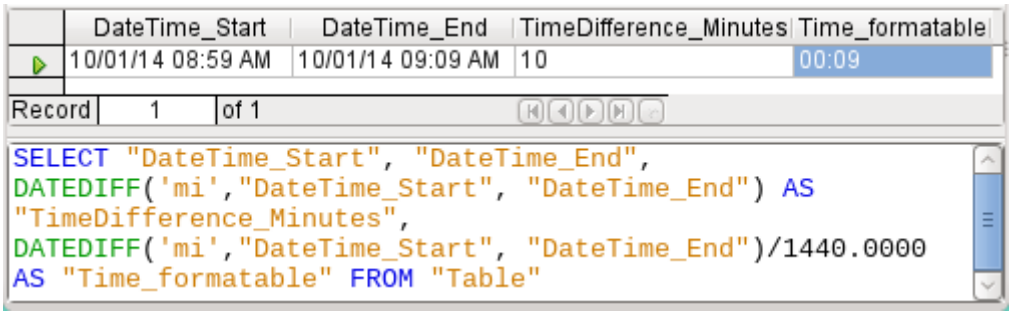
Cuando utilice funciones de cálculo, siempre debe comprender que los tipos de datos en HSQLDB tienen una precisión limitada. Independientemente de la cantidad de decimales que utilice, el hecho es que los resultados intermedios que intervienen en un cálculo de tiempo solo se pueden usar de forma limitada para cálculos posteriores.

Si posteriormente se va a utilizar un valor de hora en un formulario o informe como hora formateada, debe asegurarse de que el día sea válido como una base para el formato de hora.



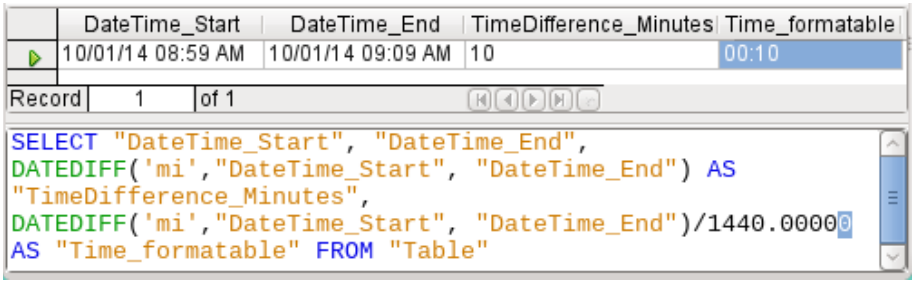
Aquí la diferencia se calcula en minutos. El resultado se da como una fracción de un día. Un día tiene 60*24 minutos. Si simplemente se divide por 1440, el resultado será cero, por lo que una vez más se deben expresar los lugares decimales explícitamente. Aquí aparece como un tiempo formateado de 0 horas y 1 minuto.

El código de formato para un tiempo superior a un día es [HH]:MM. Si se usa el formato incorrecto, una diferencia horaria de 1 día y 1 minuto podría mostrarse como solo 1 minuto.



¡El demonio del error ataca de nuevo! Una diferencia de tiempo de 10 minutos no debe aparecer como 9 minutos cuando se formatea correctamente. Para descubrir dónde radica el problema, debemos considerar exactamente cómo se realiza el cálculo:

1) 10/1440 = 0.00694. El resultado se redondea a 0.0069 porque solo se especificaron cuatro decimales.
2) 0.0069 * 1440 = 9.936 minutos, que son 9 minutos 56.16 segundos. ¡Y los segundos no se muestran en el formato elegido!



	DateTime_Start	DateTime_End	TimeDifference_Minutes	Time_formatable
▶	10/01/14 08:59 AM	10/01/14 09:09 AM	10	00:10

Record 1 of 1

```
SELECT "DateTime_Start", "DateTime_End",
DATEDIFF('mi',"DateTime_Start", "DateTime_End") AS
"TimeDifference_Minutes",
DATEDIFF('mi',"DateTime_Start", "DateTime_End")/1440.00000
AS "Time_formatable" FROM "Table"
```

Este error se corrige alargando el divisor con tan solo un decimal (1440.00000 en lugar de 1440.0000). Ahora el redondeo es con cinco cifras 0.00694 * 1440 da 9.9936 que son 59.616 segundos. El número de segundos se redondea a 60 segundos, por lo que los 9 minutos tienen 1 minuto agregado, lo que hace 10 minutos en total.

Aún puede haber más problemas. ¿Podría haber más decimales que, cuando se formateen, no produzcan 1 minuto?

Para resolver esto, un cálculo corto, el uso de Calc con redondeos similares puede ayudar. En la columna A pondremos una secuencia de números enteros a partir del 1 (serán los minutos). Y en la columna B la fórmula `= ROUND (A1 / 1440; 4)`, formateamos la columna B para que muestre horas y minutos. Si vamos bajando, vemos en la columna A `10` minutos y en la columna B `00:09`. Similar desfase se da en el minuto `28`, etc. Si redondea a 5 lugares, estos errores desaparecen.

Por muy bueno que sea tener una presentación con un formato adecuado en un formulario, debe tener en cuenta que se trata de valores con redondeo que no son adecuados para su uso posterior en cálculos mecánicos a ciegas. Si se necesita usar un valor para un cálculo posterior, es preferible usar solo una representación de la diferencia de tiempo en las unidades más pequeñas disponibles, en este caso, minutos.

LibreOffice

Guía de Base

Capítulo 6
Informes

Crear informes utilizando el Generador de informes

Los informes se utilizan para presentar los datos de una manera que las personas sin conocimiento de bases de datos entiendan fácilmente. Los informes pueden:

- Presentar datos en tablas fáciles de leer.
- Crear gráficos para mostrar datos.
- Permitir el uso de datos para imprimir etiquetas.
- Producir cartas modelo como facturas, avisos de recordatorio o notificaciones a personas que se unen o abandonan una asociación.

Para crear un informe se requiere un trabajo preparatorio cuidadoso en la base de datos subyacente. A diferencia de un formulario, un informe no puede incluir subinformes y, por lo tanto, incorporar fuentes de datos adicionales. Un informe tampoco puede presentar elementos de datos diferentes a los que están disponibles en la fuente de datos subyacente, como puede ser un formulario utilizando *Listados*.

Los informes se preparan mejor mediante consultas. De esta forma se pueden determinar todas las variables. En particular, si se requiere ordenar dentro del informe, utilice siempre una consulta que prevea la ordenación. Esto significa que las consultas en modo SQL directo deben evitarse en estas condiciones. Si debe utilizar una consulta de este tipo en su base de datos, puede llevar a cabo la ordenación creando primero una vista desde la consulta. Esta vista siempre se puede ordenar y filtrar utilizando la interfaz gráfica de usuario (GUI) de Base.

Precaución

Al usar el Generador de informes, debe guardar frecuentemente su trabajo durante la edición. Además de guardar dentro del Generador de informes después de cada paso significativo, también debe guardar toda la base de datos.

Dependiendo de la versión de LibreOffice que esté utilizando, el Generador de informes a veces puede bloquearse durante la edición.

La funcionalidad de los informes completados no se ve afectada incluso si se crearon con otra versión, en la que no se produce el problema.

Nota

Desde LibreOffice 4.1, el Generador de informes se ha integrado completamente y ya no aparece en Extensiones. Es esencial que no instale una extensión del Generador de informes que no coincida con su versión junto con el Generador de informes integrado.

La interfaz de usuario del Generador de informes

Para iniciar el Generador de informes desde Base, use **Informes > Crear informe en modo de Diseño.**

La ventana inicial del *Generador de informes* (Figura 180) muestra tres partes. A la izquierda está la división actual del informe en *Encabezado de página*, *Detalle* y *Pie de página*; en el medio están las áreas correspondientes donde se ingresará el contenido; y, a la derecha, se muestran las propiedades de estas regiones.

Al mismo tiempo, se muestra el diálogo *Agregar campos*. Este diálogo corresponde al que está en la creación del formulario. Esto crea campos con sus correspondientes etiquetas de campo.

Sin contenido desde la base de datos, un informe no tiene una función adecuada. Por este motivo, el diálogo se abre en la pestaña *Datos*. Aquí puede configurar el contenido del informe; en el ejemplo es la tabla View_Report_Recall. Siempre que el comando *Analizar orden de SQL* esté establecido en Sí, el informe puede estar sujeto a clasificación, agrupación y filtrado. Se ha elegido una vista para la base de este informe, por lo que no se aplicará ningún filtro; ya se ha incluido en la consulta subyacente a la vista.

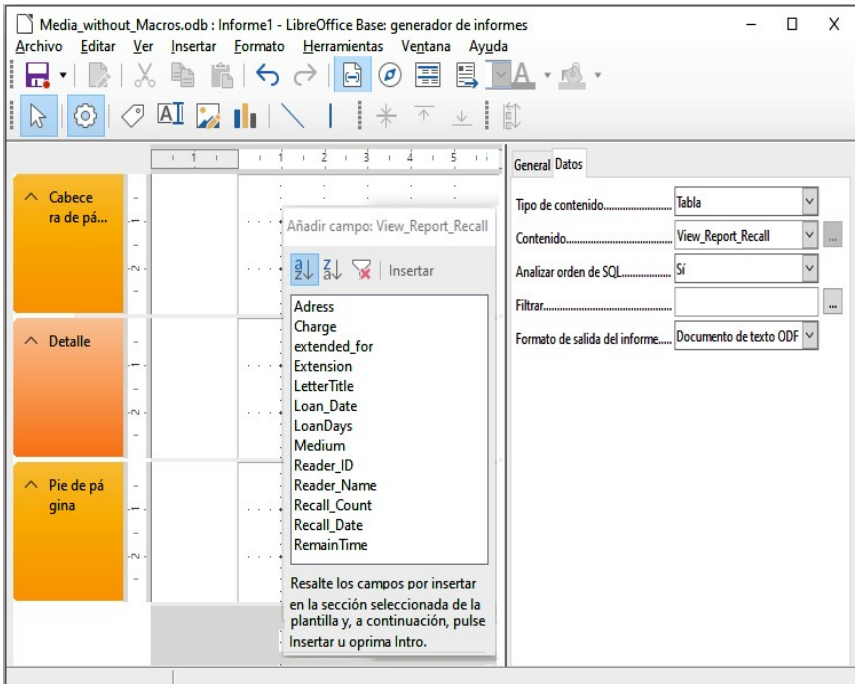


Figura 180: Ventana inicial del Generador de informes

Nota

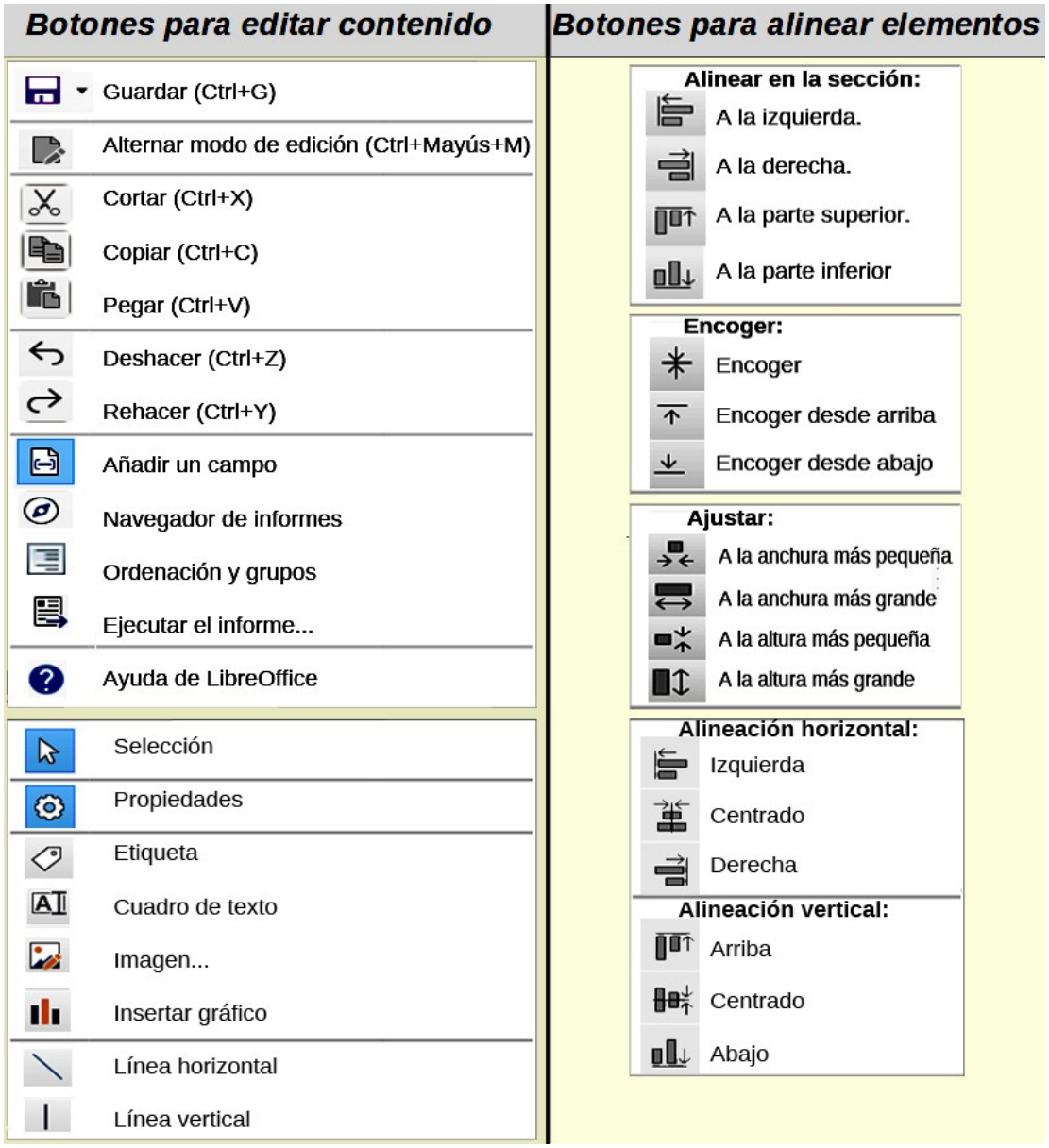
El tipo de contenido proporcionado puede ser una tabla, una vista, una consulta o una codificación directa de SQL. El *Generador de informes* funciona mejor cuando se le proporcionan datos que se han preparado con la mayor anticipación posible. Así, por ejemplo, los cálculos en las consultas se pueden realizar por adelantado y el alcance de los registros que deben aparecer en el informe se limita cuando sea necesario.

En versiones anteriores de LibreOffice a veces había problemas cuando las consultas que involucraban más de una tabla debían agruparse más tarde. Tales problemas podrían evitarse si usa una vista en lugar de una consulta. Una vista se parece a una tabla de base de datos para programar elementos como el *Generador de informes*. Los nombres de los campos están predefinidos y fijos y es posible un acceso posterior utilizando los comandos de clasificación y agrupación sin causar errores.

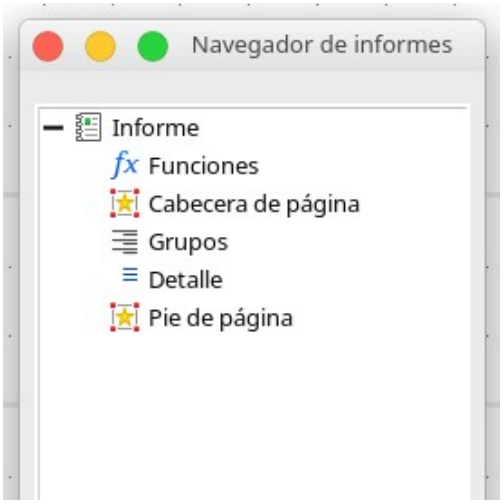
Se pueden seleccionar dos formatos de salida para informes: documento de texto ODF (un documento de Writer) u hoja de cálculo ODF (un documento de Calc). Si solo desea una vista tabular de sus datos, definitivamente debe elegir el documento Calc para su informe. Es significativamente más rápido de crear y también es más fácil de formatear posteriormente, ya que hay menos opciones a considerar y las columnas se pueden arrastrar fácilmente al ancho requerido después.

De manera predeterminada, el *Generador de informes* busca su fuente de datos en la primera tabla de la base de datos. Esto asegura que al menos una prueba de las funciones sea posible. Se debe elegir una fuente de datos antes de que el informe pueda proporcionarse con campos.

El Generador de informes proporciona muchos botones adicionales, por lo que la figura siguiente muestra los botones con sus descripciones. Los botones para alinear elementos no se describen con más detalle en este capítulo. Son útiles para el ajuste rápido de los campos en un área única del Generador de informes, pero en principio todo se puede hacer mediante la edición directa de las propiedades del campo.



Al igual que con los formularios, es útil usar el navegador adecuadamente. Así, por ejemplo, un clic descuidado al comienzo del Generador de informes puede dificultar la búsqueda de las propiedades de los datos para el informe. Es posible acceder a dichos datos haciendo clic izquierdo en el *Navegador de informes*. El Informe y las propiedades del informe vuelven a estar de nuevo accesibles.



Inicialmente, el navegador muestra, además de las secciones visibles del documento (Cabecera de página, Grupos, Detalle y Pie de página), la posibilidad de incluir Funciones. Los grupos se pueden usar, por ejemplo, para asignar todos los medios que se retiran a la persona que los tomó prestados, para evitar múltiples avisos de retiro. Las áreas de detalle muestran los registros que pertenecen a un grupo. Las funciones se utilizan para cálculos como sumas.

Para obtener resultados útiles en el ejemplo anterior, el contenido de la vista debe reproducirse con una agrupación adecuada. Cada lector debe estar vinculado a los avisos de retirada de todos sus medios prestados y vencidos.

Ver > Ordenación y grupos, o el botón correspondiente inicia la función de agrupación.

Figura 181: Ordenar y agrupar

Aquí la agrupación y la ordenación son por el campo Reader_Name. Los campos adicionales también podrían incluirse en la tabla anterior. Por ejemplo, si también desea agrupar y ordenar por el campo Loan_Date, elija esto como la segunda línea.

Directamente debajo de la tabla, hay varias acciones de agrupación disponibles para su selección. Puede mover un grupo hacia arriba o hacia abajo en la lista o eliminarlo por completo. Como solo se necesita un grupo para el informe planificado, en la Figura 181 el extremo derecho de las *Acciones de grupo* solo muestra como seleccionable el símbolo de *Eliminar*.

La propiedad de clasificación se explica por sí misma.

Cuando se creó la entrada, el lado izquierdo del Generador de informes mostró inmediatamente una nueva división. Junto a la descripción del campo Reader_Name ahora puede ver el encabezado. Esta sección es para el encabezado del grupo en el informe. El encabezado puede contener el nombre de la persona que recibirá el aviso de retirada. En este caso no hay pie de página grupal. Tal pie de página podría contener la multa debida, o el lugar y la fecha actual y un espacio para la firma de la persona que envía el aviso.

Por defecto, hay un nuevo grupo para cada valor. Entonces, si el Reader_Name cambia, se inicia un nuevo grupo. Alternativamente, puede agrupar por letra inicial. Sin embargo, en el caso de un aviso de retirada, esto pondría a todos los lectores con la misma inicial en un solo grupo. Schmidt, Schulze y Schulte recibirían un aviso de retirada común, que sería bastante inútil en este ejemplo.

Al agrupar por letra inicial, también puede especificar cuántas letras más tarde debe comenzar el siguiente grupo. Uno puede imaginar, por ejemplo, una agrupación para una pequeña guía telefónica. Según el tamaño de la lista de contactos, uno podría imaginar una agrupación en cada segunda letra inicial. Entonces A y B formarían el primer grupo, luego C y D, y así sucesivamente.

Se puede configurar un grupo para que se mantenga junto con la primera sección de detalles o, en la medida de lo posible, como un grupo completo. De manera predeterminada, esta opción está configurada en No. Para los avisos de recuperación, es probable que desee organizar el grupo de manera que se imprima una página separada para cada persona que recibirá una carta de recuperación. En otro menú, puede elegir que cada grupo (en este caso, el nombre de cada lector), sea seguido por un salto de página antes de tratar con el siguiente valor.

Si ha elegido tener un encabezado de grupo y quizás un pie de página de grupo, estos elementos aparecerán como secciones en el navegador de informes bajo el nombre de campo correspondiente Reader_Name. Aquí también tiene la posibilidad de usar funciones, que luego se limitarán a este grupo.

Para agregar campos, use la función *Agregar campo*, como con los formularios. Sin embargo, en este caso, la etiqueta y el contenido del campo no están unidos. Ambos se pueden mover independientemente, cambiar de tamaño y arrastrar a diferentes secciones.

La Figura 182 muestra el diseño del informe para el aviso de retirada. En el encabezado de la página está el encabezado *Libre Office Library*, insertado como un campo de etiqueta. Aquí también podría tener un membrete con un logotipo, ya que se pueden incluir gráficos. Este nivel se llama Encabezado de página, pero eso no implica que no haya espacio encima. Eso depende de la configuración de la página; Si se ha establecido un margen superior, se encuentra sobre el encabezado de la página.

Reader_Name Header es el encabezado de los datos agrupados y ordenados. En los campos que deben contener datos, los nombres de los campos de datos correspondientes se muestran en gris claro. Entonces, por ejemplo, la vista subyacente al informe tiene un campo llamado Dirección, que contiene la dirección completa del destinatario con la calle y la ciudad. Para poner esto en un solo campo requiere saltos de línea en la consulta. Puede usar `CHAR (13) || CHAR (10)` para crearlos.

Ejemplo:

```
SELECT "Salutation"||CHAR(13)||CHAR(10)||"FirstName"||' '||"LastName"||CHAR(13)||
```

```
CHAR(10)||"Street"||' '||"No"||CHAR13||CHAR(10)||"Postcode"||' '||"Town" AS "Adress" FROM "Reader"
```

El campo = TODAY () representa una función incorporada, que inserta la fecha actual en esta posición.

En *Reader_Name Header*, además del saludo, vemos los encabezados de columna para la siguiente vista de tabla. Estos elementos deberían aparecer solo una vez, incluso si se enumeran varios medios.

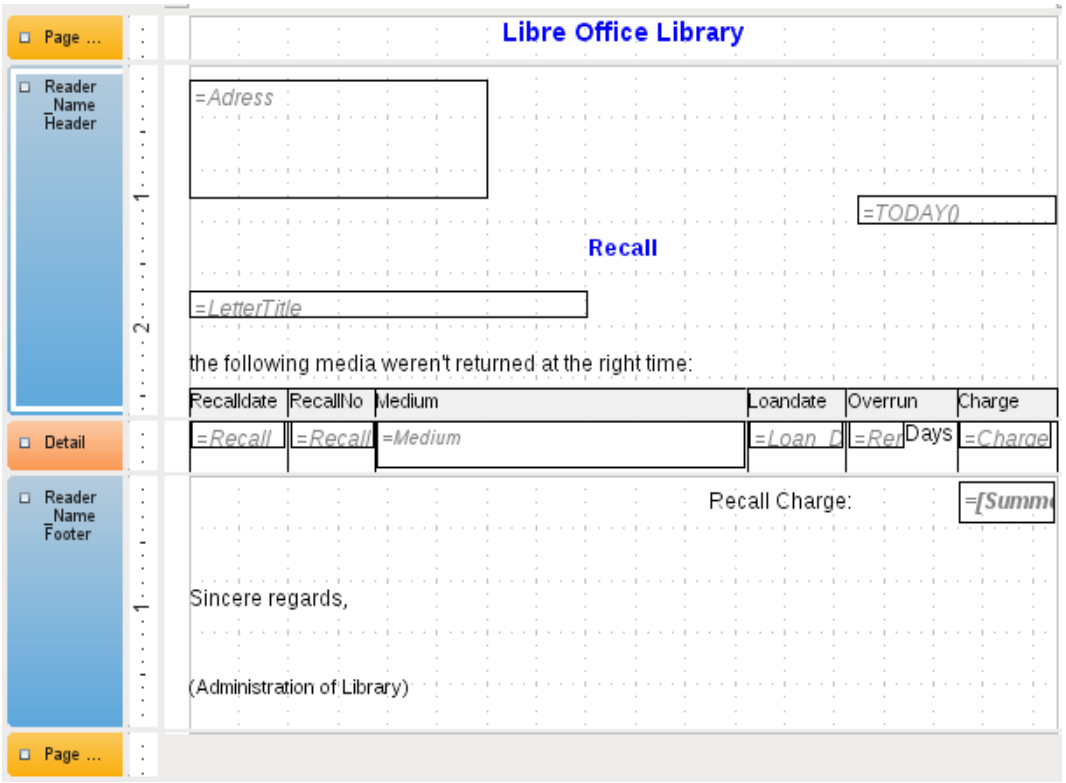


Figura 182: Diseño del informe de aviso de retirada (por ejemplo).

Como fondo de estos encabezados de columna hay un rectángulo gris, que también sirve como marco para los datos.

El área *Detail* se repite con la frecuencia en que haya registros separados con los mismos datos de Reader_Name. Aquí se enumeran todos los medios que no se han devuelto a tiempo. Hay otro rectángulo en el fondo para enmarcar el contenido. Este rectángulo está lleno de blanco en lugar de gris.

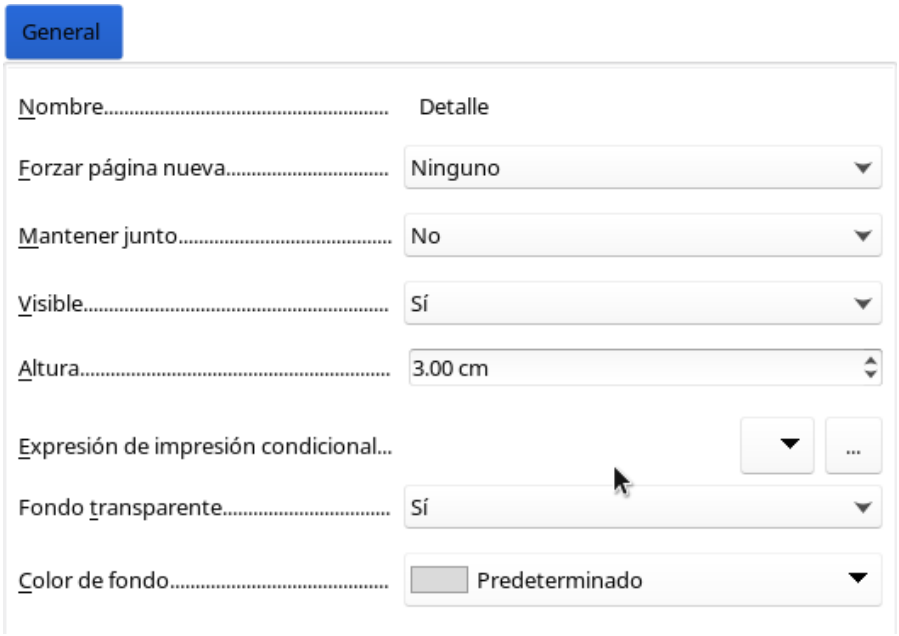
Nota

En principio, LibreOffice ofrece la posibilidad de agregar líneas horizontales y verticales. Estas líneas tienen la desventaja de que se interpretan solo como rayas. Se pueden reproducir mejor si se utilizan rectángulos. Establezca el fondo del rectángulo en negro y el tamaño en, por ejemplo, 17 cm de ancho y 0,03 cm de alto. Esto creará una línea horizontal con un grosor de 0.03cm y una longitud de 17cm. Desafortunadamente, esta variante también tiene una desventaja: los elementos gráficos no se pueden colocar correctamente si el área se extiende en más de una página.

El pie de página Reader_Name cierra la carta con una fórmula de saludo y un área para la firma. El pie de página está tan definido que se producirá un salto de página adicional después de este área. Además, a diferencia de la configuración predeterminada, se especifica que este área debe

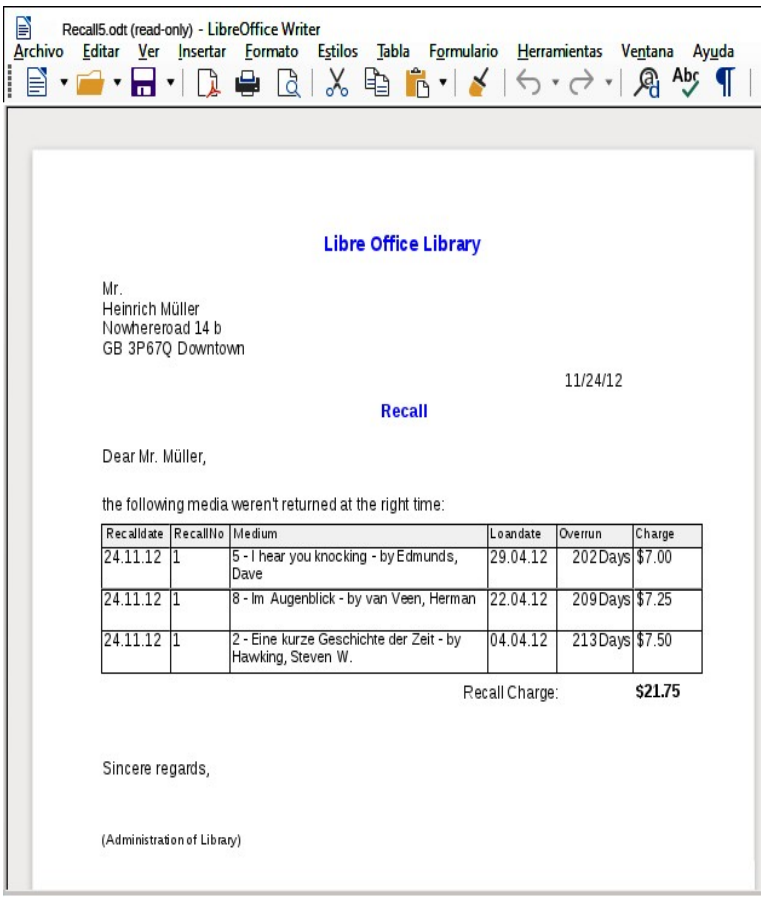
mantenerse unida en todos los casos. Después de todo, sería bastante extraño si muchos avisos de retiro tuvieran la firma en una página separada.

Mantener junto se refiere aquí al salto de página. Si desea que el contenido de un registro se mantenga junto independientemente del salto, esto solo es posible en la actualidad si el registro no se lee como *Detalle*, sino que se utiliza como base para una agrupación. Puede elegir *Mantener junto = Sí*, pero no funciona; El área *Detalle* se separa. Tienes que poner el contenido de *Detalle* en un grupo separado para mantenerlo unido.



Se utiliza una función incorporada para calcular las multas totales.

A continuación se muestra cómo se vería un aviso de retiro real. El área de detalles contiene 5 medios que el lector ha prestado. El pie de página del grupo contiene la multa total adeudada.



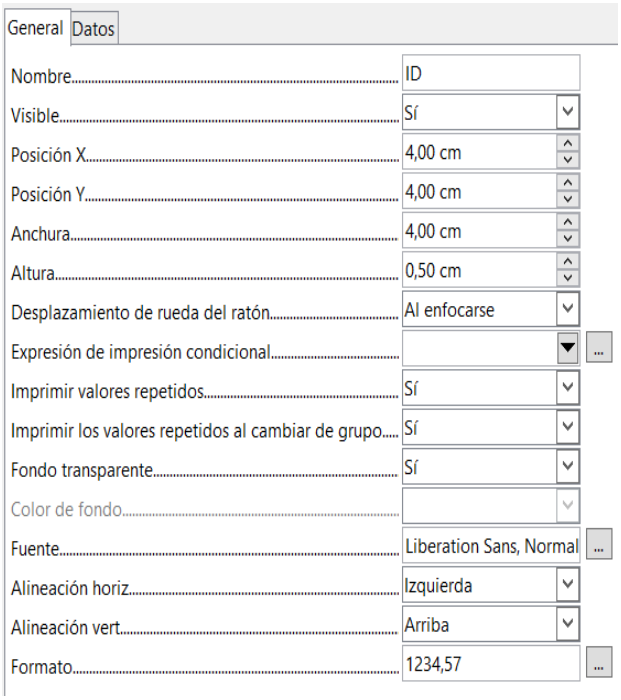
Nota

Los informes para registros individuales también pueden extenderse a más de una página. El tamaño del informe está bastante alejado del tamaño de la página. Sin embargo, estirar el área de detalles en más de una página puede provocar saltos defectuosos. Aquí el Generador de informes todavía tiene problemas para calcular el espaciado correctamente. Si se incluyen tanto las áreas de agrupación como los elementos gráficos, esto puede dar como resultado tamaños impredecibles para ciertas áreas.

Hasta ahora, los elementos individuales se pueden mover a posiciones fuera del tamaño de una sola página solo con el ratón y las teclas del cursor. Las propiedades de los elementos siempre proporcionan la misma distancia máxima desde la esquina superior de cualquier área que se encuentre en la primera página.

Propiedades generales de los campos.

Solo hay tres tipos de campo para la presentación de datos. Además de los campos de texto (que, al contrario de su nombre, también pueden contener números y formatos), también hay un tipo de campo que puede contener imágenes de la base de datos. El campo del gráfico muestra un resumen de datos.



Al igual que con los formularios, los campos reciben nombres. Por defecto, el nombre es el del campo de base de datos subyacente.

Un campo se puede configurar para que sea invisible. Esto puede parecer un poco inútil en el caso de los campos, pero es útil para los encabezados y pies de página del grupo, que pueden ser necesarios para llevar a cabo otras funciones de la agrupación sin contener nada que deba mostrarse.

Si Imprimir valores repetidos está desactivado, la visualización del campo se inhibe cuando se carga un campo con el mismo contenido directamente antes. Esto funciona correctamente solo para campos de datos que contienen texto. Los campos numéricos o los campos de fecha ignoran la instrucción de desactivación. Los campos de etiqueta se desvanecen por completo cuando se desactivan, incluso si ocurren solo una vez.

En el Generador de informes, la visualización de cierto contenido puede inhibirse mediante el uso de la expresión de impresión condicional o el valor del campo puede usarse como base para formatear texto y fondo. Se brinda más información sobre las expresiones condicionales en “Impresión condicional ” en la página 265.

La configuración de la rueda del ratón no tiene ningún efecto porque los campos de informe no son editables. Parece ser un remanente del diálogo de definición de formulario.

La función Imprimir cuando cambio de grupo tampoco se pudo reproducir en los informes.

Si el fondo no se define como transparente, se puede definir un color de fondo para cada campo.

Las otras entradas tratan del contenido interno del campo en cuestión. Esto cubre la fuente (para el color de la fuente, el peso de la fuente, etc., consulte la Figura 183), la alineación del texto dentro del campo y las características en su correspondiente cuadro específico (consulte la Figura 184).

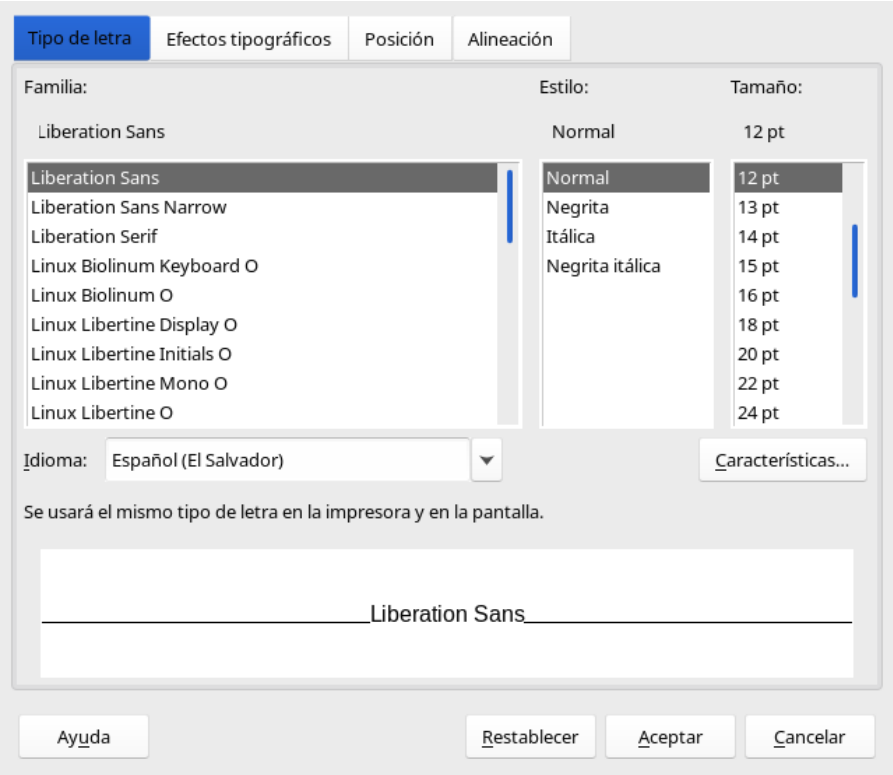


Figura 183: Fuentes: Configuración de caracteres

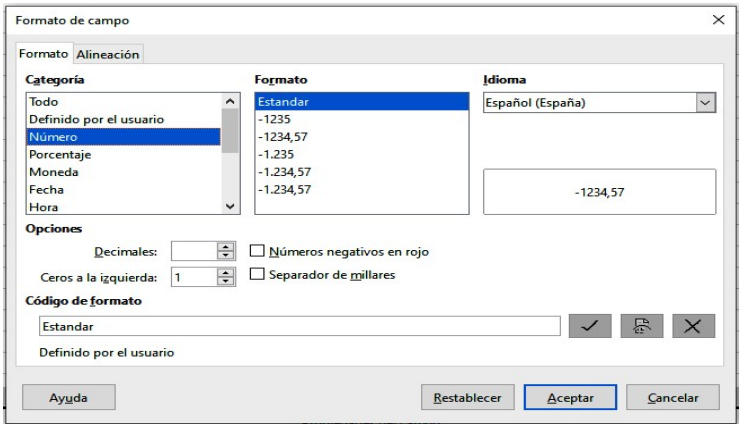
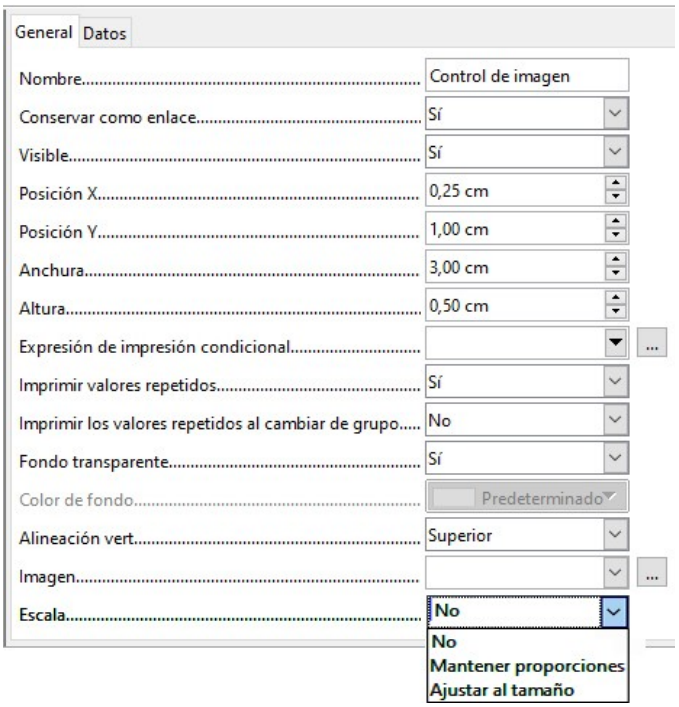


Figura 184: Formato de números

Propiedades especiales de controles gráficos.



Un control gráfico puede contener gráficos tanto dentro como fuera de la base de datos. Desafortunadamente, actualmente no es posible almacenar un gráfico como un logotipo de forma permanente en Base. Por lo tanto, es esencial que el gráfico esté disponible en la ruta de búsqueda, incluso cuando se le presenta la opción de incrustar en lugar de vincular imágenes y el primer campo *Conservar como enlace* se puede configurar (literalmente cerrado) a una funcionalidad planificada correspondiente. Esta es una de varias funciones planificadas para Base y están en la GUI pero aún no se han implementado, por lo que los botones y las casillas de verificación no tienen ningún efecto.

Alternativamente, por supuesto, un gráfico se puede almacenar en la base de datos y, por lo tanto, está disponible internamente. Pero en ese caso, debe ser accesible a través de uno de los campos en la consulta subyacente al informe.

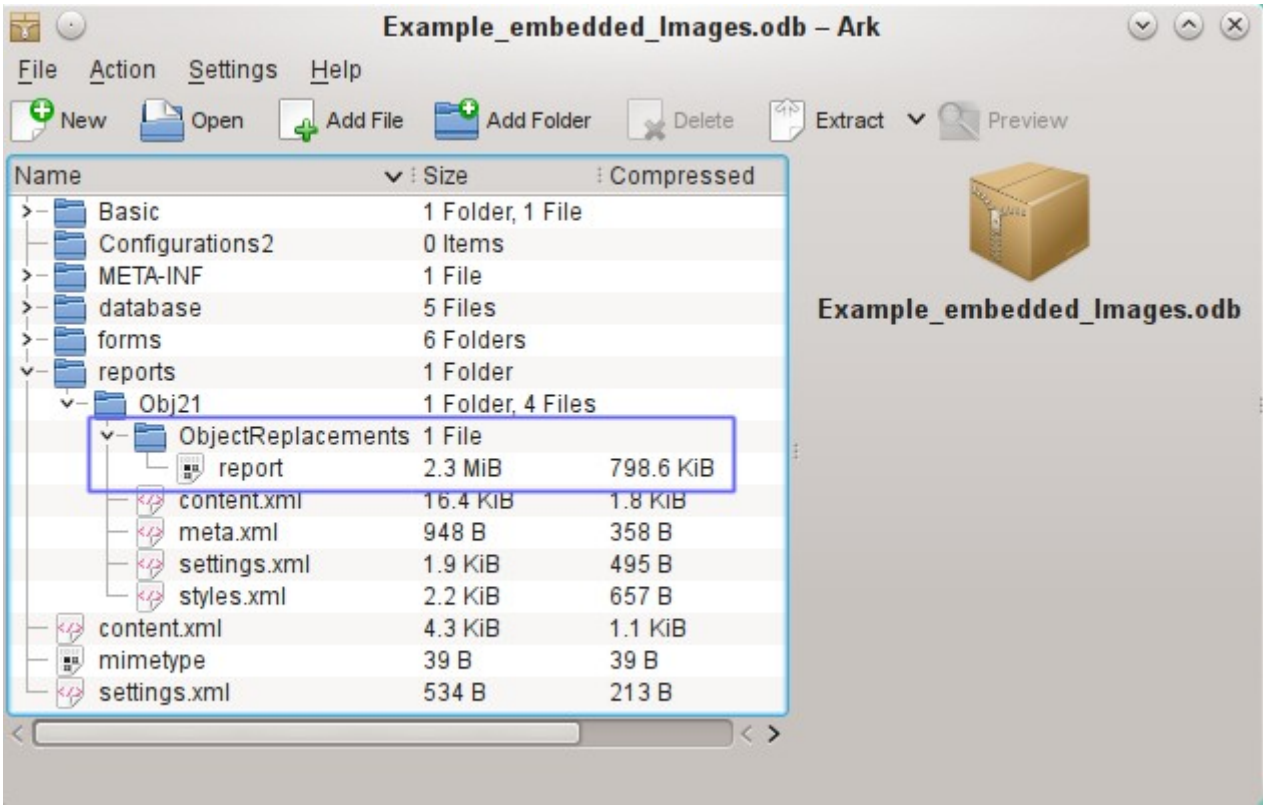
Para buscar un gráfico externo, use el botón de selección (…) al lado del campo *Imagen*. Para cargar un campo de base de datos gráfica, especifique el campo en la pestaña *Datos*.

La configuración de alineación vertical no parece tener ningún efecto durante la etapa de diseño. Sin embargo, cuando llama al informe, el gráfico aparece en la posición correcta.

Al escalar, puede seleccionar *No, Mantener proporciones* o *Ajustar al tamaño*. Esto corresponde a la configuración de un formulario:

- *No*: La imagen no está ajustada al control. Si es demasiado grande, se muestra una versión recortada. La imagen original no se ve afectada por esto.
- *Mantener proporciones*: La imagen se ajusta al control pero no se distorsiona.
- *Ajustar al tamaño*: La imagen se ajusta al control y, en algunos casos, puede distorsionarse.

Cuando se editan informes que contienen imágenes, puede suceder que la base de datos se vuelva significativamente más grande. Dentro del archivo *.odb, Base coloca en el directorio de informes una carpeta *ObjectReplacements*, por razones que no se entienden completamente. Esta carpeta contiene un archivo de "*report*" responsable de la ampliación.



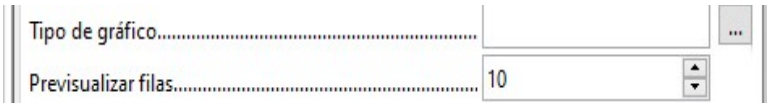
Si la base de datos se abre en un programa de archivo, esta carpeta con su contenido es visible en el subdirectorio de informes. Puede utilizar de forma segura el programa de archivo para buscar y eliminar la carpeta.

Nota

Si los informes no se van a editar repetidamente, es suficiente eliminar una vez la carpeta ObjectReplacements. El tamaño de esta carpeta puede crecer muy rápidamente. Esto depende del número y tamaño de los archivos incluidos. ¡Una prueba mostró que un solo archivo jpg de 2.8MB agrandó el archivo *.odb en 11MB! Ver error 80320.

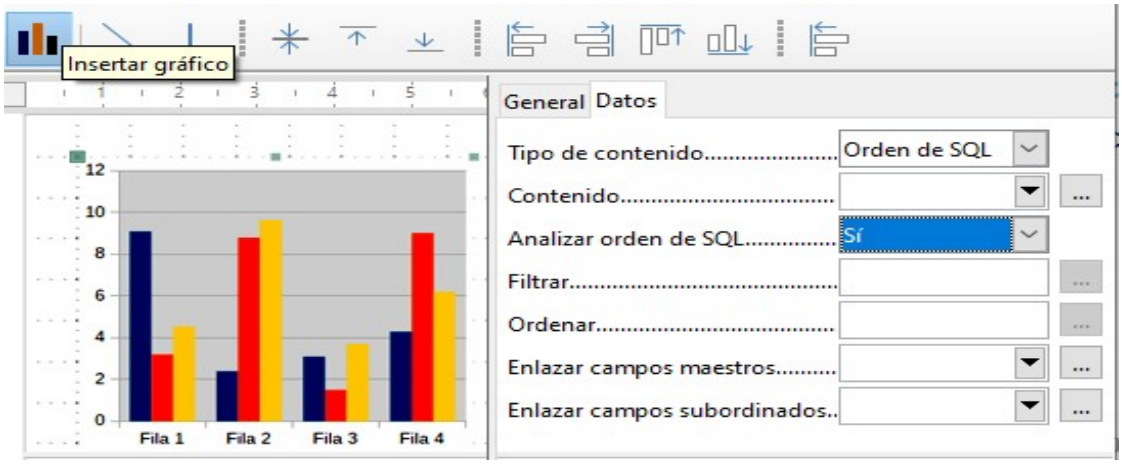
Incorporar gráficos al informe.

Puede insertar gráficos en un informe utilizando el control *Insertar gráfico,* o en la barra con **Insertar> Controles de informe> Gráfico**. Un gráfico es la única forma de reproducir datos que no se encuentran en la fuente de datos especificada para el informe. Por lo tanto, un gráfico puede verse como una especie de subinforme, pero también como un componente independiente del informe.



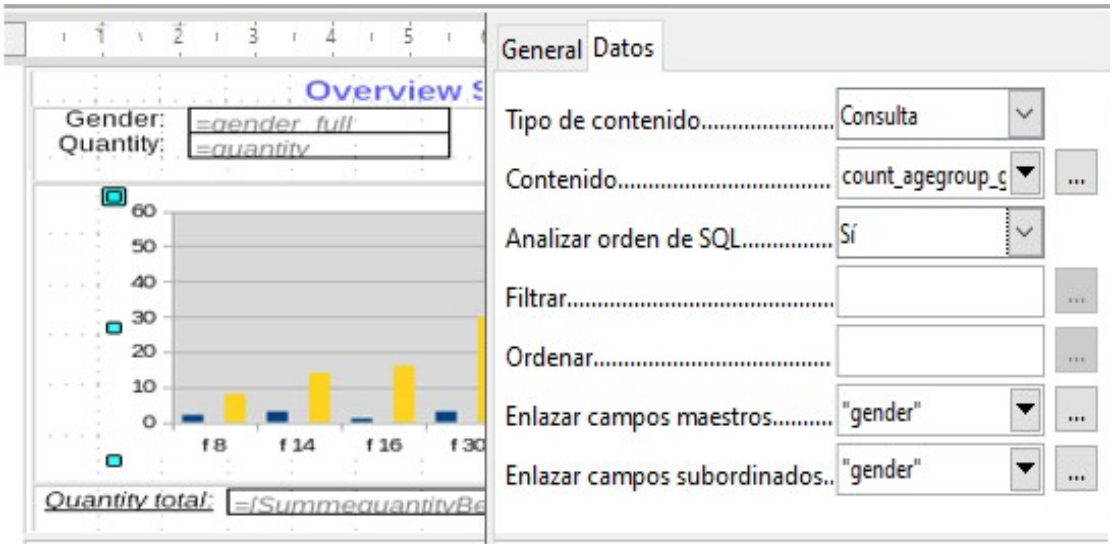
Tiene que dibujar el lugar del gráfico con el ratón. En las propiedades generales, además de los campos familiares, puede elegir un *Tipo de Gráfico* (consulte los tipos correspondientes en Calc). Además, puede establecer un número máximo de registros para la *Vista previa*, lo que facilitará una muestra de cómo se verá finalmente el gráfico.

Los gráficos se pueden formatear de la misma manera que en Calc (haga doble clic en el gráfico). Para obtener más información, consulte la descripción en la *LibreOffice Calc Guide*.



El gráfico está vinculado en la sección Datos con los campos de datos necesarios. Aquí, en un ejemplo de lista de los 10 principales medios, el gráfico muestra la frecuencia con la que se prestan medios particulares. El Editor de consultas se utiliza para crear una orden SQL adecuada, como lo haría para un *Listado* en un formulario. La primera columna de la consulta se usará para proporcionar las etiquetas para las barras verticales en el gráfico, mientras que la segunda columna arroja el número total de transacciones de préstamos, que se muestran en la altura de las barras.

En el ejemplo anterior, el gráfico muestra muy poco al principio, ya que solo se realizaron préstamos de prueba limitados antes de que se emitiera la orden SQL.



En las propiedades de datos del gráfico, se ha especificado una *Consulta*. Este gráfico de la base de datos Example_sport.odb[6] muestra algo especial además de los conceptos básicos de creación de gráficos en informes: la vista previa del gráfico muestra más columnas de las anticipadas. Esto resulta del contenido de la consulta, que crea columnas adicionales que no aparecerán todas en el gráfico en sí.

No es necesario filtrar y ordenar con las herramientas internas del Generador de informes, ya que esto ya se ha hecho en la medida de lo posible dentro de la consulta.

6 Esta base de datos se incluye en el paquete de bases de datos de ejemplo para este manual.

Consejo

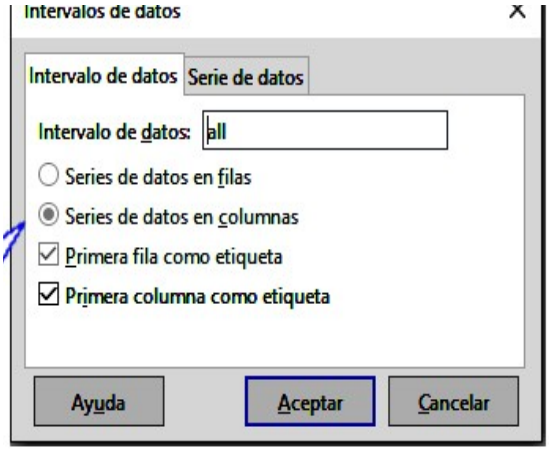
Básicamente, desea eliminar tantas tareas como sea posible de la creación de sus informes. Lo que se pueda gestionar al principio del proceso mediante el uso de consultas no necesita hacerse nuevamente durante el proceso relativamente lento de creación del informe en sí.

Al igual que con los formularios principales y subformularios, algunos campos ahora están vinculados entre sí. En el informe real, los grupos de edad para los participantes del campamento deportivo masculino y femenino se enumeran en una tabla. Están agrupados por género. En cada grupo, ahora hay un gráfico separado. Para garantizar que el gráfico solo use datos para el género correcto, los dos campos llamados "Gender", en el informe y en el gráfico, están vinculados entre sí.

El eje X del gráfico se vincula automáticamente a la primera columna de la tabla de origen de datos. Si hay más de dos columnas en la tabla, las columnas adicionales se colocan automáticamente en el gráfico.

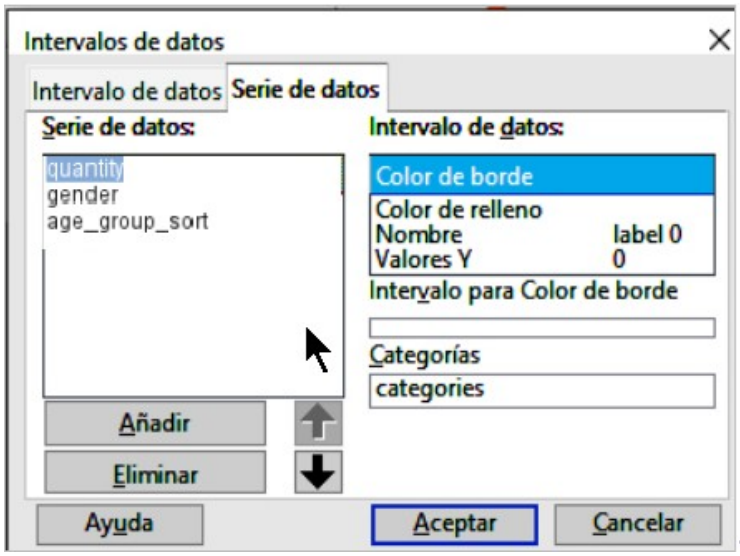
Se puede acceder a otras configuraciones para el gráfico seleccionando todo el gráfico con un doble clic. Al hacer clic con el botón derecho del ratón, se abre un menú contextual sobre el diagrama, cuyo contenido depende del elemento que se haya seleccionado.

Contiene configuraciones posibles para los *Intervalos de datos:*



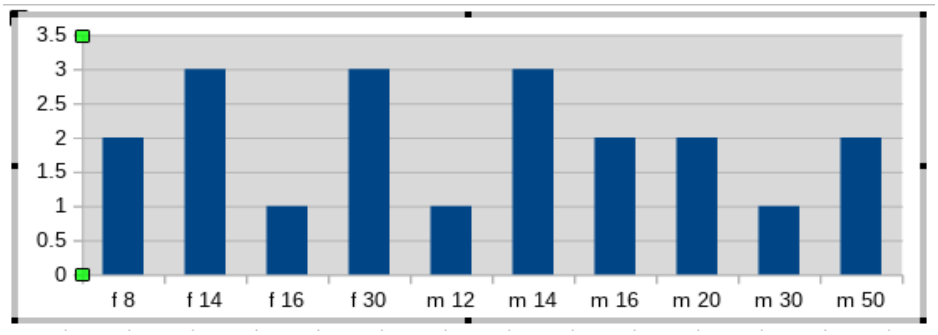
Las series de datos en columnas están atenuadas y, por lo tanto, no se pueden cambiar. Tampoco puede cambiar la casilla *Primera fila como etiqueta*. La configuración restante en la pestaña *Intervalos de datos* no debe modificarse, ya que aquí hay más posibilidades de las que realmente puede manejar el Generador de informes.

La pestaña *Serie de datos*, por otro lado, oculta un par de configuraciones que pueden cambiar significativamente la apariencia predeterminada de su gráfico. Muestra todas las series de datos disponibles para la primera columna de la consulta. Todo lo que no desee mostrar se puede eliminar en este momento.



En el ejemplo que se muestra arriba, había demasiadas columnas visibles en el gráfico. ¡Esto requiere mejoras! Ni la serie *gender* ni la serie *age_group_sort*, cuyos nombres provienen de la consulta subyacente, son de ninguna utilidad aquí. La serie de género (*gender*) se utiliza para vincular el gráfico con la fuente de datos del informe y, en ningún caso, puede representarse numéricamente. Y la serie de clasificación por grupo de edad (*age_group_sort*) solo sirve para garantizar un tipo correcto de los valores en la consulta, ya que de lo contrario un código como "m8" vendría inmediatamente antes de "m80" en lugar de al principio (la clasificación en campos de texto a menudo tiene resultados indeseables similares).

Cuando todas las filas se han eliminado antes de la fila Total, la vista previa del gráfico se ve así:

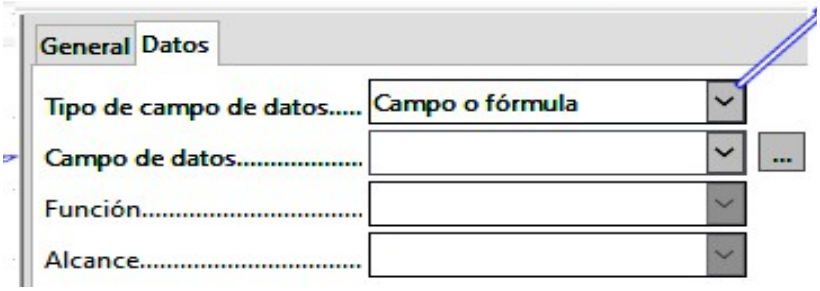


Esta vista previa muestra 10 columnas, las primeras diez columnas de la consulta. En la ejecución real, solo se mostrarán las columnas que pertenecen al género correcto: las columnas "m" para hombres y las "f" para mujeres.

El eje Y todavía muestra una característica inapropiada. Después de todo, ¡no existe la mitad de una persona! De nuevo, esto podría mejorarse. Sin embargo, en una ejecución automática con esta configuración, estos números se cambiarán a enteros si el rango de valores, como en el ejemplo anterior, no se detiene en '3'. Si este procesamiento automático se desactivara, entonces sería necesaria alguna mejora manual.

Todas las configuraciones adicionales son similares a las que Calc usa para la creación de gráficos.

Propiedades de datos de los campos.



En el diálogo de *Propiedades* (por ejemplo en Imagen), la pestaña *Datos* muestra de forma predeterminada el *Campo de datos* de la base desde donde se leerán los datos para este campo de informe.

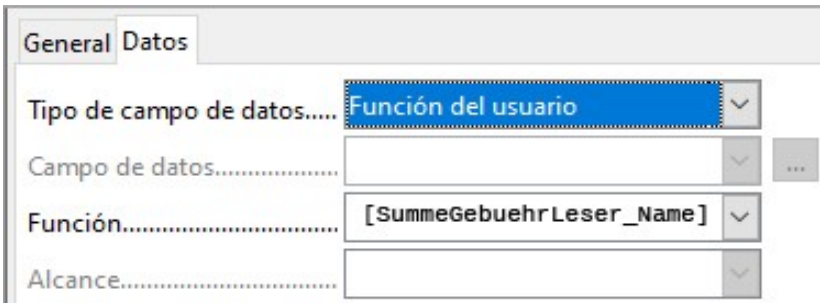
Sin embargo, en *Tipo de campo de datos*, además de *Campo o Fórmula*, están disponibles los tipos *Función*, *Contador* y *Función del usuario*.

Puede seleccionar de antemano las funciones Suma, Mínimo y Máximo. Se aplicarán al grupo actual o al informe completo. Estas funciones pueden generar problemas si un campo está vacío (`NULL`). En dichos campos, si han sido formateados como números, aparece `NaN`; es decir, no hay valor numérico presente. Para los campos vacíos, no se realiza ningún cálculo y el resultado es siempre 0.

Dichos campos pueden reformatearse para mostrar un valor de 0 utilizando la siguiente fórmula en el área de datos de la vista.

```
IF([numericfield];[numericfield];0)
```

Esta función calcula con el valor real de un campo que no tiene valor. Parece más simple formular la consulta subyacente para el informe de modo que se dé 0 en lugar de `NULL` para los campos numéricos.
El contador solo cuenta los registros que ocurrirán en el grupo o en el informe como un todo. Si el contador se inserta en el área *Detalles*, se proporcionará a cada registro un número consecutivo. La numeración se aplicará solo a los registros en el grupo o en todo el informe.



Finalmente, la función detallada definida por el usuario está disponible. Puede suceder que el Generador de informes elija esta variante si se ha solicitado un cálculo, pero por alguna razón no puede interpretar correctamente la fuente de datos.

Funciones en el generador de informes

El generador de informes proporciona una variedad de funciones, tanto para mostrar datos como para establecer condiciones. Si esto no es suficiente, se pueden crear funciones definidas por el usuario utilizando simples pasos de cálculo, que son particularmente útiles en los pies de página y resúmenes grupales.

Introducir fórmulas

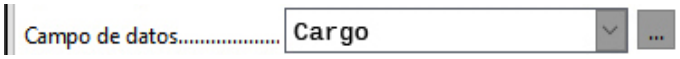
El generador de informes se basa en el generador de informes de Pentaho.
Una pequeña parte de su documentación está en
http://wiki.pentaho.com/display/Reporting/9.+Report+Designer+Formula+Expressions.

Una fuente adicional son las Especificaciones para el Estándar OpenFormula:
http://www.oasis-open.org/committees/download.php/16826/openformula-spec-20060221.html

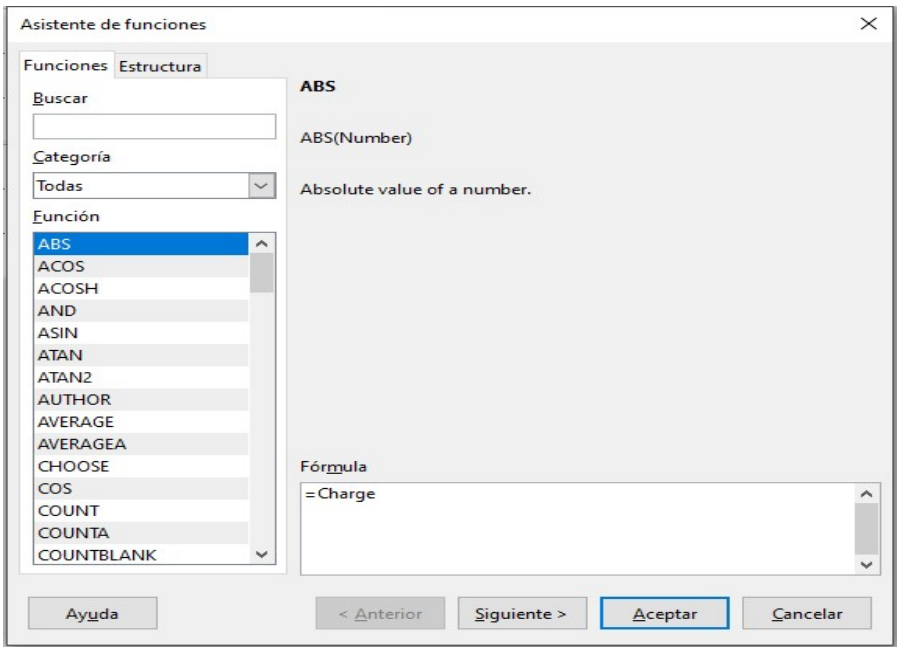
Principios básicos:

Las fórmulas comienzan con un signo igual.	`=`
Las referencias a los campos de datos se colocan entre corchetes.	`[Nombre del campo ]`
Si los campos de datos contienen caracteres especiales (incluidos espacios), el nombre del campo también debe estar entre comillas.	`["Este nombre de campo debe estar entre comillas "]`
La entrada de texto siempre debe estar entre comillas dobles.	`"Entrada de texto "`
Los siguientes operadores están permitidos.	+, -, * (Multiplicación), / (División), % (dividir el número anterior por 100), ^ (Elevar a potencia del siguiente número), & (concatenar texto),
Las siguientes relaciones son posibles.	`= , <> , < , <= , > , >=`
Se permiten paréntesis.	`( )`
Mensaje de error predeterminado .	`NA` (Not available)
Mensaje de error para un campo vacío que se definió como numérico.	`NaN` (Quizás "not a number"?)

Toda entrada de fórmula se aplica solo al registro actual. Por lo tanto, las relaciones con registros anteriores o siguientes no son posibles.



Al lado del campo de datos hay un botón con tres puntos cada vez que se puede ingresar una fórmula. Este botón invoca al *Asistente de funciones*.



Sin embargo, hay muchas menos funciones que en Calc. Muchas funciones tienen equivalentes de Calc. Ahí el Asistente calcula el resultado de la función directamente.

El asistente de funciones no siempre funciona a la perfección. Por ejemplo, las entradas de texto no se utilizan con comillas dobles. Sin embargo, solo las entradas con comillas dobles se procesan al iniciar la función.

Las siguientes funciones están disponibles:

Función	***Descripción***
Funciones de fecha y hora	
`DATE`	Produce una fecha válida a partir de valores numéricos para el año, el mes y el día.
`DATEDIF (DAY | MONTH | YEAR)`	Devuelve el total de años, meses o días entre dos valores de fecha.
`DATEVALUE`	Convierte una entrada de fecha estadounidense en forma de texto (citada) en un valor de fecha. La variante estadounidense que se produce puede ser reformateada.
`DAY`	Devuelve el día del mes para una fecha determinada. `DAY`([campo de fecha])
`DAYS`	Devuelve el número de días entre dos fechas.
`HOUR`	Devuelve las horas de un tiempo determinado en formato de 24 horas. `HOUR ([DateTimeField])` calcula las horas en el campo.
`MINUTE`	Devuelve los minutos de una fecha en formato numérico interno `MINUTE([Campo de tiempo])` calcula los minutos del tiempo.
`MONTH`	Devuelve el mes para una fecha ingresada como un número. `MONTH([campo de fecha])`
`NOW`	Devuelve la fecha y hora actuales.

SECOND	Devuelve los segundos de fecha en formato numérico interno SECOND(NOW()) muestra la parte de segundos del tiempo en que se ejecuta el comando.
TIME	Muestra la hora actual.
TIMEVALUE	Convierte una entrada de texto de tiempo en un valor de tiempo para cálculos.
TODAY	Muestra la fecha actual.
WEEKDAY	Devuelve el día de la semana como un número. El día número 1 es el domingo.
YEAR	Devuelve la parte del año de una entrada de fecha.
Funciones lógicas	
AND	Produce TRUE cuando todos sus argumentos son VERDADEROS.
FALSE	Define el valor lógico como FALSE.
IF	Si una condición es TRUE, entonces este valor, en caso contrario otro valor.
IFNA	
NOT	Invierte el valor lógico de un argumento.
OR	Produce TRUE cuando alguna de sus condiciones es VERDADERA.
TRUE	Define el valor lógico como TRUE.
XOR	Produce TRUE cuando solo uno de los valores vinculados es VERDADERO.
Funciones de redondeo	
INT	Redondea hacia abajo al entero anterior.
Funciones matemáticas	
ABS	Devuelve el valor absoluto (no negativo) de un número.
ACOS	Calcula el arcocoseno de un número. - argumentos entre -1 y 1.
ACOSH	Calcula el area-coseno (coseno hiperbólico inverso) – argumento >= 1.
ASIN	Calcula el arcoseno de un número: argumento entre -1 y 1.
ATAN	Calcula el arcotangente de un número.
ATAN2	Calcula el arcotangente de una coordenada x y una coordenada y.
AVERAGE	Da el promedio de los valores ingresados.
AVERAGEA	Da el promedio de los valores ingresados. El texto se trata como cero.
COS	El argumento es el ángulo en radianes cuyo coseno debe calcularse.

EVEN	Redondea un número positivo hacia arriba o un número negativo hacia abajo al siguiente entero par.
EXP	Calcula la función exponencial (Base 'e').
LN	Calcula el logaritmo natural de un número.
LibreOfficeG10	Calcula el logaritmo de un número (Base '10').
MAX	Devuelve el máximo de una serie de números.
MAXA	Devuelve el valor máximo en una fila. Cualquier texto se establece en cero.
MIN	Devuelve el más pequeño de una serie de valores.
MINA	Devuelve el valor mínimo en una fila. Cualquier texto se establece en cero.
MOD	Devuelve el resto de una división cuando ingresa el dividendo y el divisor.
ODD	Redondea un número positivo hacia arriba o un número negativo hacia abajo al siguiente entero impar.
PI	Da el valor del número 'π'.
POWER	Eleva la base al poder del exponente.
SIN	Calcula el seno de un número.
SQRT	Calcula la raíz cuadrada de un número.
SUM	Suma una lista de valores numéricos.
SUMA	Suma una lista de valores numéricos. Se permiten campos de texto y Sí / No. Lamentablemente, esta función (todavía) termina con un mensaje de error.
VAR	Calcula la varianza, a partir de una muestra.

Funciones de texto	
EXACT	Muestra si dos cadenas de texto son exactamente iguales.
FIND	Da el desplazamiento de una cadena de texto dentro de otra cadena .
LEFT	El número especificado de caracteres de una cadena de texto se reproduce a partir de la izquierda.
LEN	Da el número de caracteres en una cadena de texto.
LOWER	Convierte texto a minúsculas.
MESSAGE	Formatea el valor en el formato de salida dado.
MID	El número especificado de caracteres de una cadena de texto se reproduce a partir de una posición de caracteres especificada.
REPLACE	Reemplaza una subcadena por una subcadena diferente. Se debe indicar la posición inicial y la longitud de la subcadena a reemplazar.

REPT	Repite el texto un número específico de veces.
RIGHT	El número especificado de caracteres de una cadena de texto se reproduce a partir de la derecha.
SUBSTITUTE	Reemplaza partes específicas de una cadena de texto dada por texto nuevo. Además, puede especificar cuál de varias ocurrencias de la cadena de destino se reemplazará.
T	Devuelve el texto o una cadena de texto vacía si el valor no es texto (por ejemplo, un número).
TEXT	Conversión de números u horas en texto.
TRIM	Elimina espacios iniciales y espacios terminales, y reduce múltiples espacios en un solo espacio.
UNICHAR	Convierte un número decimal Unicode en un carácter Unicode. Por ejemplo, 196 se convierte en 'Ä' ('Ä' tiene el valor hexadecimal 00C4, que es 196 en decimales sin ceros a la izquierda).
UNICODE	Convierte un carácter Unicode en un número decimal Unicode. 'Ä' se convierte en 196.
UPPER	Devuelve una cadena de texto en mayúsculas.
URLENCODE	Convierte un texto dado en uno que se ajuste a una URL válida. Si no se especifica ningún estándar particular, se sigue ISO-8859-1.
Funciones de información	
CHOOSE	El primer argumento es un índice, seguido de una lista de valores. Se devuelve el valor representado por el índice. CHOOSE(2;"Apple";"Pear";"Banana") returns Pear. CHOOSE([age_level_field];"Milk";"Cola";"Beer") devuelve una posible bebida para el 'age_level_field' dado.
COUNT	Solo se cuentan los campos que tienen un número o una fecha. COUNT ([hora]; [número]) devuelve 2, si ambos campos contienen un valor (no NULL) o si no 1 o 0.
COUNTA	Incluye también campos que contienen texto. Incluso se cuenta NULL, junto con campos booleanos.
COUNTBLANK	Cuenta los campos vacíos en una región.
HASCHANGED	Comprueba si la columna nombrada ha cambiado. Sin embargo, no se proporciona información sobre la columna.
INDEX	Trabaja con regiones
ISBLANK	Comprueba si el campo es NULL (vacío).
ISERR	Devuelve TRUE si la entrada tiene un error distinto de NA. ISERR (1/0) da TRUE
ISERROR	Al igual que ISERR, excepto que NA también devuelve TRUE.

ISEVEN	Comprueba si un número es par.
ISLOGICAL (ISTLOG)	Comprueba si este es un valor Sí / No. ISLOGICAL (TRUE ()) o ISLOGICAL (FALSE ()) producen TRUE, los valores de texto como ISLOGICAL ("TRUE") producen FALSE.
ISNA	Comprueba si la expresión es un error de tipo NA.
ISNONTEXT	Comprueba si el valor no es texto.
ISNUMBER	Comprueba si algo es numérico. ISNUMBER(1) produce TRUE, ISNUMBER("1") produce FALSE
ISODD	Comprueba si un número es impar.
ISREF	Comprueba si algo es una referencia de campo. ISREF ([Nombre del campo]) produce VERDADERO, ISREF (1) produce FALSO.
ISTEXT	Comprueba si el contenido del campo es texto.
NA (NV)	Devuelve el código de error NA.
VALUE	
Definido por el usuario	
CSVARRAY	Convierte texto CSV en una matriz.
CSVTEXT	Convierte una matriz en texto CSV.
NORMALIZEARRAY	
NULL	Devuelve NULL.
PARSEDATE	Convierte texto en una fecha. Utiliza el SimpleDateFormat. Requiere una fecha en el texto como se describe en este formato de fecha. Ejemplo: PARSEDATE ("9.10.2012"; "dd.MM.yyyy") produce el número interno utilizable para la fecha.
Información del Documento	
AUTHOR	Autor, como se lee en **Herramientas> Opciones> LibreOffice> Datos del usuario**. Por lo tanto, este no es el autor real sino el usuario actual de la base de datos.
TITLE	Devuelve el título del informe.

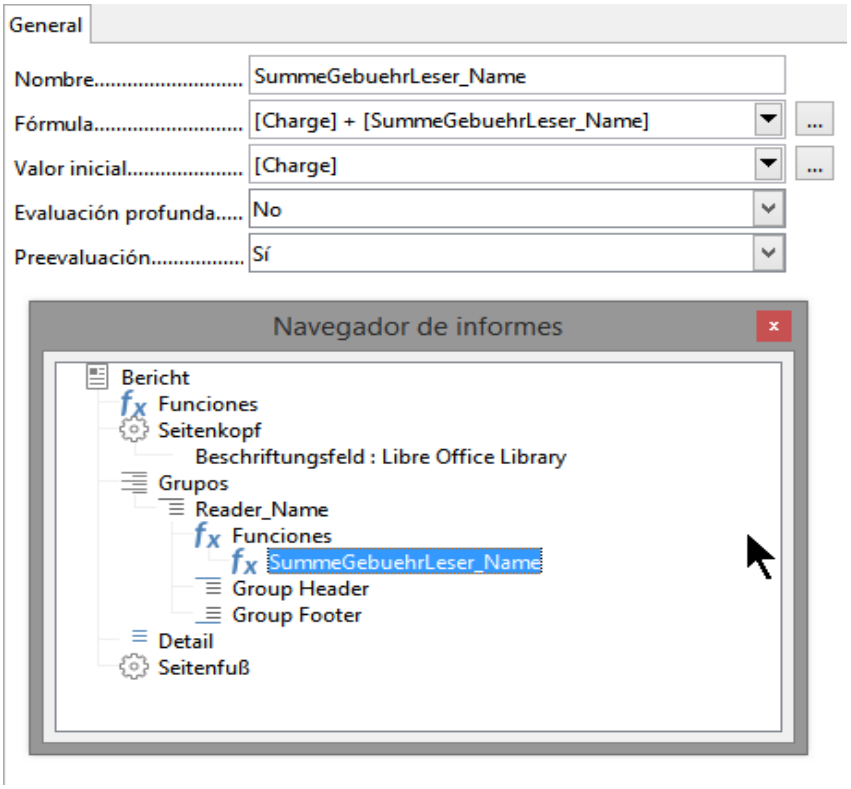
Funciones definidas por el usuario

Puede usar funciones definidas por el usuario para devolver resultados intermedios específicos para un grupo de registros. En el ejemplo anterior, se utilizó una función de este tipo para calcular las multas en el área Reader_Name_Footer.

En el Navegador de Informes, la función se muestra en el grupo Reader_Name. Al hacer clic con el botón derecho en esta función, puede definir funciones adicionales por nombre

Las propiedades de la función SummeGebuehrLeser_Name se muestran arriba. La fórmula agrega el campo Cargo al valor ya almacenado en la función misma. El valor inicial es el valor del campo Cargo en el primer recorrido del grupo. Este valor se almacena en la función bajo el

nombre de la función y se reutiliza en la fórmula, hasta que finaliza el ciclo y se escribe el pie de página del grupo.



El recorrido profundo parece no tener ninguna función por ahora, a menos que los gráficos se traten aquí como subinformes.

Si la evaluación previa está activada para la función, el resultado también se puede colocar en el encabezado del grupo. De lo contrario, el encabezado del grupo contiene solo el valor correspondiente del primer campo del grupo.

Las funciones definidas por el usuario también pueden hacer referencia a otras funciones definidas por el usuario. En ese caso, debe asegurarse de que las funciones utilizadas ya se hayan creado. Se debe excluir el cálculo previo en funciones que se refieren a otras funciones.

```
[SumMarksClass] / ([ClassNumber]+1)
```

Se refiere al grupo de clase. El contenido del campo Marcas se suma y se devuelve la suma de todos los registros. La suma de las marcas se divide por la suma de los registros. Para obtener el número correcto, se debe agregar 1 como se muestra con [ClassNumber]. Esto producirá las calificaciones promedio.

Entrada de fórmula para un campo

Con el campo **Datos> Campo de Datos**, puede ingresar fórmulas que afectan solo a un campo en el área Detalles.

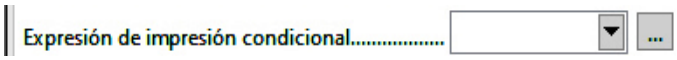
```
IF([boolean_field];"Yes";"No")
```

establece los valores permitidos en "Sí" o "No" en lugar de TRUE y FALSE.

Puede suceder que en un campo con una entrada de fórmula, solo aparezca un número. En los campos de texto esto es un cero.

Para solucionar esto, debe cambiar el campo de texto, de “Number” predeterminado a "Text".

Impresión condicional



Las propiedades generales de los encabezados de grupo, pies de página de grupo y campos incluyen un campo de *Expresión de impresión condicional*. Las fórmulas que se escriben en este campo influyen en el contenido de un campo o en la visualización de una región completa.

```
[Fieldname]="true"
```

hace que el contenido del campo con nombre se muestre solo si es verdadero.[7]

Muchas formas de visualización condicional no están completamente determinadas por las propiedades especificadas. Por ejemplo, si se va a insertar una línea de separación gráfica después del décimo lugar de una lista de resultados de la competencia, no puede simplemente usar el siguiente comando de visualización condicional para el gráfico:

```
[Place]=10
```

Este comando no funciona. En cambio, el gráfico continuará apareciendo en la sección Detalles después de cada registro posterior. Ver error 73707.

Si solo desea una forma rectangular superpuesta en esta ubicación, puede hacerlo utilizando un control gráfico, al que se le puede dar la dirección de un archivo gráfico (monocromo). En las propiedades generales de este control, **Escalado> Autom** debe ser seleccionado. Entonces, el gráfico se ajustará al formulario y se cumplirá la condición.

Es más seguro vincular la visualización condicional a un pie de página de grupo que al gráfico, si esto no es necesario. La línea se coloca en el pie de página del grupo. Entonces la línea realmente aparece después del décimo lugar, cuando se formula como se indica arriba. Debido a que la condición está asociada con el pie de página del grupo, la línea aparece solo si realmente es necesaria. De lo contrario, la visualización condicional puede provocar que aparezca un espacio en blanco en lugar de la línea.

Aquí también puede usar las formas proporcionadas por **Insertar> Formas> Formas estándar**, por ejemplo, la línea horizontal para combinar con el pie de página del grupo.

Formato condicional

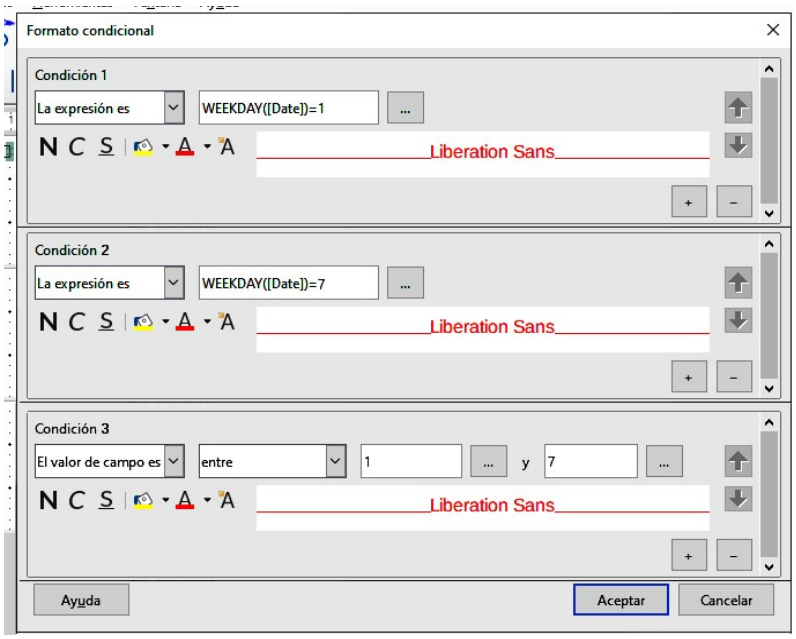
El formato condicional se puede utilizar, por ejemplo, para formatear un calendario de modo que los fines de semana se muestren de manera diferente. Elija **Formato> Formato condicional** e introduzca:

```
WEEKDAY([Date])=1
```

y el formato correspondiente para los domingos.

Si usa "Expresión es" en formato condicional, puede ingresar una fórmula. Como es habitual en Calc, se pueden formular varias condiciones y se evalúan secuencialmente. En el ejemplo anterior, se prueba el primer domingo y luego el sábado. Finalmente, puede haber una consulta sobre el contenido del campo. Entonces, por ejemplo, el contenido "Vacaciones" llevaría a un formato diferente.

7 Consulte también la base de datos `Example_Report_conditional_Overlay_Graphics.odb`, que se incluye en las bases de datos de ejemplo de este libro.



Nota

Si se producen errores adicionales que no se pueden corregir (fórmulas no implementadas, texto demasiado largo que se muestra como un campo vacío, etc.), a veces es necesario eliminar partes del informe o simplemente crearlo de nuevo.

Ejemplos de informes creados con el Generador de informes

El generador de informes tiene un uso algo traicionero, ya que ciertas funciones están disponibles en teoría pero no funcionan en la práctica. Además, la Ayuda de LibreOffice tiene muy poco que decir al respecto. Por esa razón, se proporcionan algunos ejemplos aquí, que muestran cómo se puede usar el Generador de informes para varios tipos de informes.

Impresión de facturas

Para crear una factura se requieren las siguientes consideraciones:

- Los artículos individuales deben estar numerados.
- Las facturas que requieren más de una página deben tener números de página.
- Las facturas que requieren más de una página deben tener un total acumulado en cada página, que se transfiere a la página siguiente.

Varios errores actuales parecen hacer esto imposible:

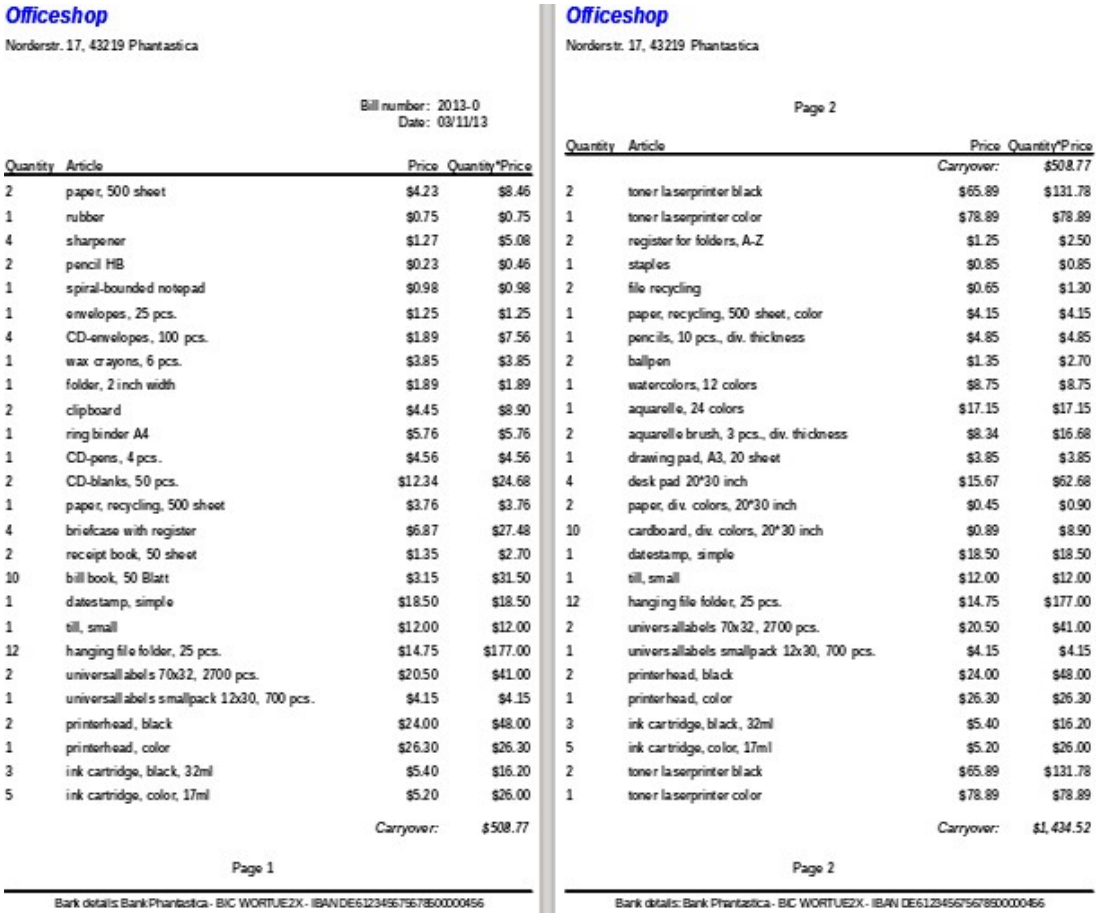
Error 51452: Si un grupo está configurado en "Repetir sección", se inserta un salto de página automáticamente antes y después del grupo.

Error 51453: Los grupos con paginación individual están permitidos pero en realidad no funcionan.

Error 51959: Un pie de página grupal no puede repetirse. Solo puede ocurrir al final del grupo, no al final de cada página. Y si elige "*Repetir sección*", desaparece por completo.

Además, hay problemas al insertar líneas en los informes. Las líneas horizontales y verticales integradas solo aparecen en las versiones de LibreOffice 4.0.5 y 4.1.1 respectivamente. Puede usar rectángulos como sustituto, pero no se pueden colocar correctamente cuando hay un salto de página en la sección.

El informe en una forma sencilla debería verse así:



Officeshop

Norderstr. 17, 43219 Phantastica

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
2	paper, 500 sheet	$4.23	$8.46
1	rubber	$0.75	$0.75
4	sharpener	$1.27	$5.08
2	pencil HB	$0.23	$0.46
1	spiral-bounded notepad	$0.98	$0.98
1	envelopes, 25 pcs.	$1.25	$1.25
4	CD-envelopes, 100 pcs.	$1.89	$7.56
1	wax crayons, 6 pcs.	$3.85	$3.85
1	folder, 2 inch width	$1.89	$1.89
2	clipboard	$4.45	$8.90
1	ring binder A4	$5.76	$5.76
1	CD-pens, 4 pcs.	$4.56	$4.56
2	CD-blanks, 50 pcs.	$12.34	$24.68
1	paper, recycling, 500 sheet	$3.76	$3.76
4	briefcase with register	$6.87	$27.48
2	receipt book, 50 sheet	$1.35	$2.70
10	bill book, 50 Blatt	$3.15	$31.50
1	datestamp, simple	$18.50	$18.50
1	till, small	$12.00	$12.00
12	hanging file folder, 25 pcs.	$14.75	$177.00
2	universallabels 70x32, 2700 pcs.	$20.50	$41.00
1	universallabels smallpack 12x30, 700 pcs.	$4.15	$4.15
2	printerhead, black	$24.00	$48.00
1	printerhead, color	$26.30	$26.30
3	ink cartridge, black, 32ml	$5.40	$16.20
5	ink cartridge, color, 17ml	$5.20	$26.00
		Carryover:	*$508.77*

Page 1

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567850000456

Officeshop

Norderstr. 17, 43219 Phantastica

Page 2

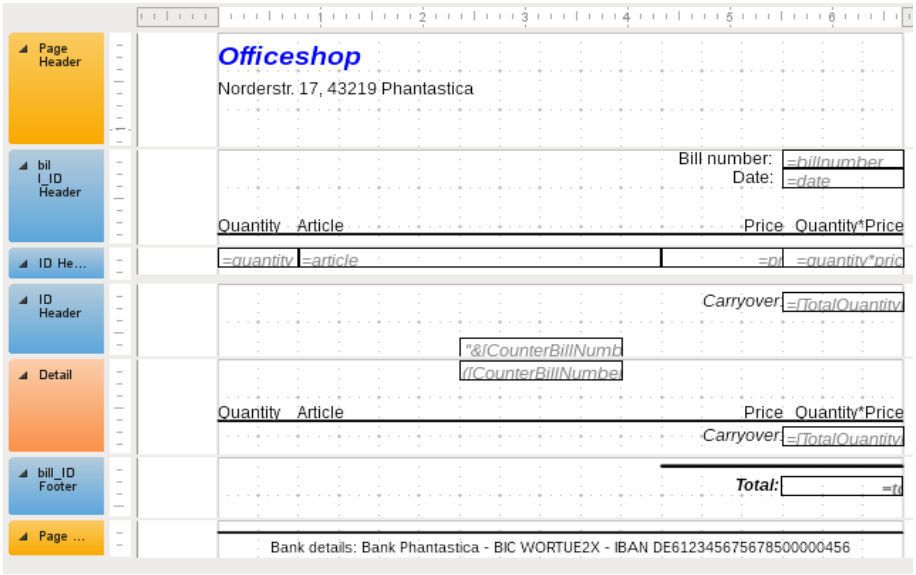
Quantity	Article	Price	Quantity*Price
		Carryover:	*$508.77*
2	toner laserprinter black	$65.89	$131.78
1	toner laserprinter color	$78.89	$78.89
2	register for folders, A-Z	$1.25	$2.50
1	staples	$0.85	$0.85
2	file recycling	$0.65	$1.30
1	paper, recycling, 500 sheet, color	$4.15	$4.15
1	pencils, 10 pcs., div. thickness	$4.85	$4.85
2	ballpen	$1.35	$2.70
1	watercolors, 12 colors	$8.75	$8.75
1	aquarelle, 24 colors	$17.15	$17.15
2	aquarelle brush, 3 pcs., div. thickness	$8.34	$16.68
1	drawing pad, A3, 20 sheet	$3.85	$3.85
4	desk pad 20*30 inch	$15.67	$62.68
2	paper, div. colors, 20*30 inch	$0.45	$0.90
10	cardboard, div. colors, 20*30 inch	$0.89	$8.90
1	datestamp, simple	$18.50	$18.50
1	till, small	$12.00	$12.00
12	hanging file folder, 25 pcs.	$14.75	$177.00
2	universallabels 70x32, 2700 pcs.	$20.50	$41.00
1	universallabels smallpack 12x30, 700 pcs.	$4.15	$4.15
2	printerhead, black	$24.00	$48.00
1	printerhead, color	$26.30	$26.30
3	ink cartridge, black, 32ml	$5.40	$16.20
5	ink cartridge, color, 17ml	$5.20	$26.00
2	toner laserprinter black	$65.89	$131.78
1	toner laserprinter color	$78.89	$78.89
		Carryover:	*$1,434.52*

Page 2

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567850000456

Para superar las restricciones descritas anteriormente, la creación de la factura correspondiente requiere una atención exacta a las medidas de la página en el documento impreso final. Este ejemplo se construye desde un formato DIN-A4. La altura total de cada página es, por lo tanto, de 29,7 cm.

El informe debe dividirse en varios grupos. Dos grupos se relacionan con el mismo campo de tabla y contienen el mismo elemento de tabla.



La siguiente tabla muestra la división de la página en las diferentes secciones del informe.

A	Margen superior	2,00 cm
B	Encabezado de página (aparece en cada página, no contiene contenido de la base de datos, solo material como el logotipo de la empresa o la dirección del proveedor).	3,00 cm
C	Encabezado de grupo para la factura (solo los elementos que pertenecen a un número de factura deben agregarse más adelante. El encabezado de grupo aparece solo al comienzo de la factura).	2,50 cm
D	Encabezado de grupo para los elementos individuales. (La sección Detalles es necesaria para un propósito diferente. Por lo tanto, aquí viene un grupo que también clasifica el contenido después de la entrada. Este grupo contiene solo un valor).	0,70 cm
E	Encabezado de grupo, también vinculado a los elementos. (Esta sección solo se muestra si se producen tantos elementos que es necesaria una página adicional. Contiene el total acumulado y el número de página en la parte inferior de la página. Hay un salto de página después de esta sección).	2,00 cm
F	Sección de detalles. (Esta sección solo se muestra si se producen tantos elementos que se necesita una página adicional. Contiene la suma transferida y el número de página en la parte superior de la página).	2,50 cm
G	Pie de grupo para el número de factura. (Aquí sigue la suma total de la factura, posiblemente con el IVA agregado. El pie de página del grupo aparece solo en la última página de la factura).	1,60 cm
H	Pie de página (por ejemplo, datos bancarios)	1,00 cm
I	Margen de la página	1,00 cm

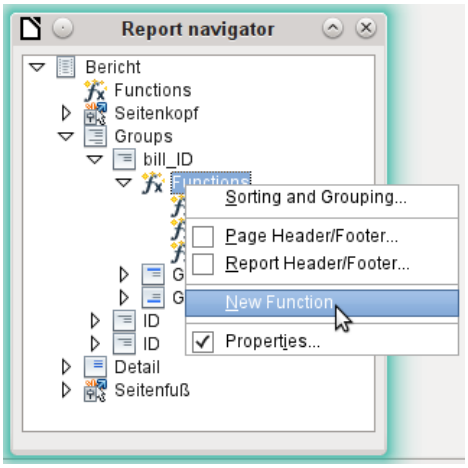
Sigue un salto de página solo si hay demasiados elementos. Para los artículos, tenemos los siguientes espacios libres:

	29,70 cm	(DIN A 4)
-	2,00 cm	(Pos. A)
-	3,00 cm	(Pos. B)
-	2,50 cm	(Pos. C)
-	1,60 cm	(Pos. G)
-	1,00 cm	(Pos. H)
-	1,00 cm	(Pos. I)
=	18,60 cm	

El espacio libre restante es, por lo tanto, como máximo 18,60 cm / 0,70 cm = 26,57. El redondeo da 26 líneas de artículos.

Tan pronto como aparezca el elemento 27, debe seguir inmediatamente un salto de página. Esto indica que se debe mostrar el encabezado de grupo E y la sección Detalles. Entonces necesitamos un contador para los artículos (sección C). Cuando este contador llega a 27, se muestra la sección Detalles (F).

El contador de los artículos se define de la siguiente manera



Usando el navegador de informes, buscamos el grupo Bill_ID. Nuestra nueva función se llama CounterBillNumber. La fórmula es [CounterBillNumber] + 1. El valor inicial es 1. Ningún subinforme está vinculado (no existe tal función). Tampoco se calcula por adelantado. Para cálculos anticipados, usamos un contador separado, CounterTotal.

El encabezado de grupo E y la sección Detalles F se muestran si hay un total de más de 26 artículos en la factura y la posición de registro actual ha alcanzado 26. Por lo tanto, la expresión para la visualización condicional es la misma para ambos:

```
AND([CounterBillNumber]=26;[CounterTotal]>26)
```

Por lo tanto, el contenido de esta sección solo aparece si se esperan al menos 27 elementos. El encabezado de grupo E aparece en la primera página. Sigue un salto de página y el contenido de la sección Detalles aparece en la página siguiente.
Ahora debemos calcular el número de secciones que deben aparecer en la primera página.

	29,70 cm	(DIN A 4)
-	2,00 cm	(Pos. A)
-	3,00 cm	(Pos. B)
-	2,50 cm	(Pos. C)
-	18,20 cm	(Pos. D * 26)
-	1,00 cm	(Pos. H)
-	1,00 cm	(Pos. I)
=	2,00 cm	

No se requiere el pie de página del grupo para la primera página. Hay 26 líneas de artículos en total. Por lo tanto, el encabezado de grupo E puede ocupar como máximo 2 cm en la primera página. Estos 2 cm deben acomodar el total acumulado y el número de página. Para garantizar un salto de página correcto en todas las circunstancias, esta sección debería ser un poco más pequeña que 2 cm. En el ejemplo, usamos 1,90 cm.

La sección Detalles viene en la parte superior de la página siguiente. Como el encabezado de grupo para los elementos (C) no aparece en esta página, la sección Detalles puede ocupar tanto espacio como el encabezado, es decir, 2.50 cm. Luego comience el siguiente lote de artículos con el mismo arreglo que en la página anterior. La suma transferida se calcula simplemente sumando los elementos anteriores.

Navegador de Informes se utiliza para encontrar el grupo Bill_number. La nueva función que se creará se llamará TotalPrice. La fórmula es [Precio] + [Precio total]. El valor inicial es [Precio]. No se vincula ningún subinforme. Tampoco es necesario calcular esta cifra por adelantado.

La suma a transferir se muestra tanto en el encabezado E del grupo como en la sección Detalles. En el encabezado del grupo, está justo en la parte superior de la página. En la primera página aparece en la parte inferior. En la sección Detalles, la suma está justo en la parte inferior. Aparece en la página 2 directamente debajo del encabezado de la tabla.
La consulta para determinar el número de página es similar a la de determinar la visualización del encabezado del grupo y la sección de detalles:

```
IF([CounterBillNumber]=26;"Page 1";"")
```

Esto le da el número de página para la primera página. Se pueden usar más consultas IF para las otras páginas.
El número de página para la siguiente página se configura fácilmente en “Página 2” de la misma manera.
Si las fórmulas continúan de la misma manera, pueden cubrir tantas páginas como desee.
La expresión para *visualización condicional* cambia

de

```
AND([CounterBillNumber]=26;[CounterBillComplete]>26)
```

a

```
AND(MOD([CounterBillNumber];26)=0; [CounterBillComplete]>[CounterBillNumber])
```

El encabezado de grupo E y la sección Detalles F aparecen solo cuando dividir el contador de artículos por 26 no proporciona ningún resto y el número total de artículos es mayor que el contador de artículos.

La expresión para el *número de página* cambia de

```
IF ([CounterBillNumber] =26; "Página 1"; "")
```

a

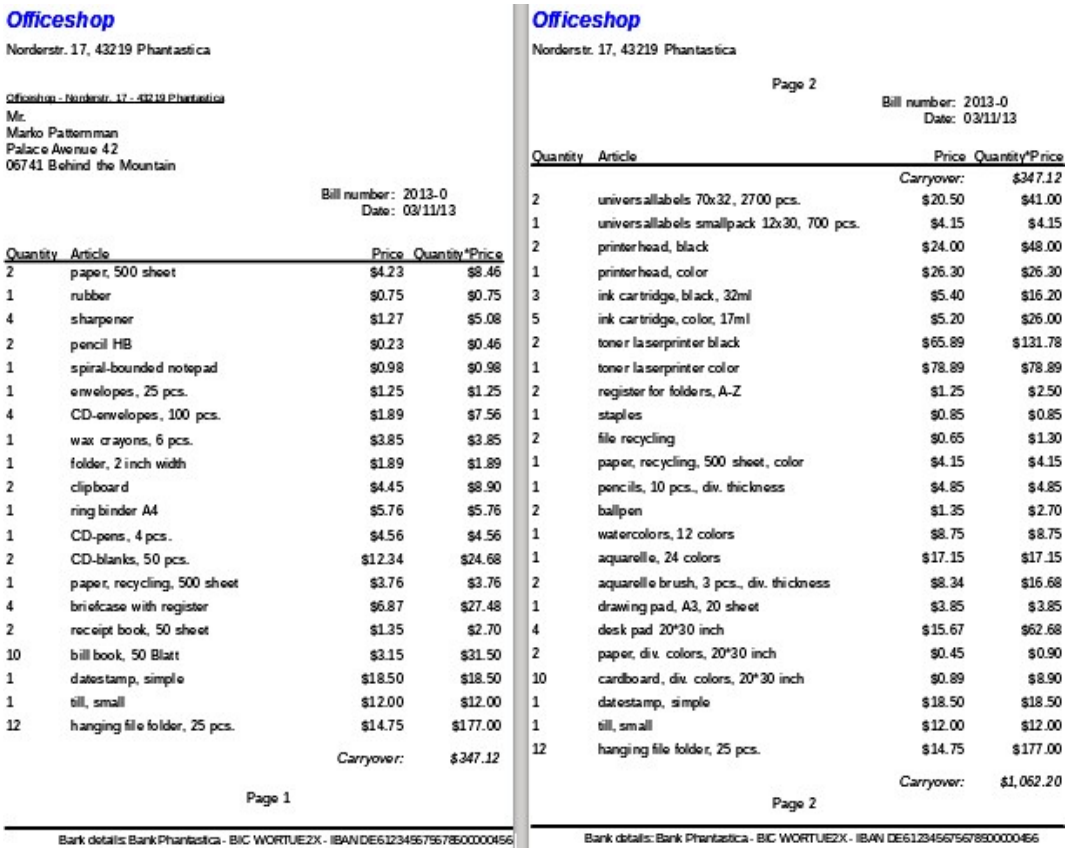
```
"Pag"&[CounterBillNumber]/26
```

para la página actual y

```
"Página"&([CounterBillNumber]/26)+1
```

para la siguiente página.

La siguiente impresión del informe con esta configuración aún no está lista para usar:



Officeshop

Norderstr. 17, 43219 Phantastica

Officeshop - Norderstr. 17 - 43219 Phantastica
Mr.
Marko Pattermman
Palace Avenue 42
06741 Behind the Mountain

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
2	paper, 500 sheet	$4.23	$8.46
1	rubber	$0.75	$0.75
4	sharpener	$1.27	$5.08
2	pencil HB	$0.23	$0.46
1	spiral-bounded notepad	$0.98	$0.98
1	envelopes, 25 pcs.	$1.25	$1.25
4	CD-envelopes, 100 pcs.	$1.89	$7.56
1	wax crayons, 6 pcs.	$3.85	$3.85
1	folder, 2 inch width	$1.89	$1.89
2	clipboard	$4.45	$8.90
1	ring binder A4	$5.76	$5.76
1	CD-pens, 4 pcs.	$4.56	$4.56
2	CD-blanks, 50 pcs.	$12.34	$24.68
1	paper, recycling, 500 sheet	$3.76	$3.76
4	briefcase with register	$6.87	$27.48
2	receipt book, 50 sheet	$1.35	$2.70
10	bill book, 50 Blatt	$3.15	$31.50
1	datestamp, simple	$18.50	$18.50
1	till, small	$12.00	$12.00
12	hanging file folder, 25 pcs.	$14.75	$177.00
		Carryover:	*$347.12*

Page 1

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567800000456

Officeshop

Norderstr. 17, 43219 Phantastica

Page 2

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
		Carryover:	*$347.12*
2	universallabels 70x32, 2700 pcs.	$20.50	$41.00
1	universallabels smallpack 12x30, 700 pcs.	$4.15	$4.15
2	printerhead, black	$24.00	$48.00
1	printerhead, color	$26.30	$26.30
3	ink cartridge, black, 32ml	$5.40	$16.20
5	ink cartridge, color, 17ml	$5.20	$26.00
2	toner laserprinter black	$65.89	$131.78
1	toner laserprinter color	$78.89	$78.89
2	register for folders, A-Z	$1.25	$2.50
1	staples	$0.85	$0.85
2	file recycling	$0.65	$1.30
1	paper, recycling, 500 sheet, color	$4.15	$4.15
1	pencils, 10 pcs., div. thickness	$4.85	$4.85
2	ballpen	$1.35	$2.70
1	watercolors, 12 colors	$8.75	$8.75
1	aquarelle, 24 colors	$17.15	$17.15
2	aquarelle brush, 3 pcs., div. thickness	$8.34	$16.68
1	drawing pad, A3, 20 sheet	$3.85	$3.85
4	desk pad 20*30 inch	$15.67	$62.68
2	paper, div. colors, 20*30 inch	$0.45	$0.90
10	cardboard, div. colors, 20*30 inch	$0.89	$8.90
1	datestamp, simple	$18.50	$18.50
1	till, small	$12.00	$12.00
12	hanging file folder, 25 pcs.	$14.75	$177.00
		Carryover:	*$1,062.20*

Page 2

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567800000456

Debido al campo de dirección, hay menos elementos en la primera página de la factura que en la segunda página. Por lo tanto, la sección Detalles en la parte superior de la segunda página es significativamente más pequeña que el encabezado del grupo para la factura (C).

Para permitir diferente número de elementos en las dos primeras páginas, se ajustan las fórmulas.

El siguiente cálculo garantiza que las secciones correspondientes se muestren correctamente.

```
AND(MOD([CounterBillNumber]-20;24)=0;
[CounterBillComplete]>[CounterBillNumber])
```

Restamos el número de elementos en la primera página del contador de elementos. Esta diferencia se divide por el número total posible de elementos en la segunda página. Si el resto es cero, la primera condición para visualizar el encabezado de grupo E y la sección Detalles F se ha cumplido. Además, como se vio antes, el valor actual del contador de artículos será menor que el total esperado. De lo contrario, habría espacio para la suma total calculada en la página actual. El posible número total de artículos en la segunda página ahora es menor porque esta página ahora contiene el número de factura y la fecha.

El número de página ahora se puede calcular de manera más simple:

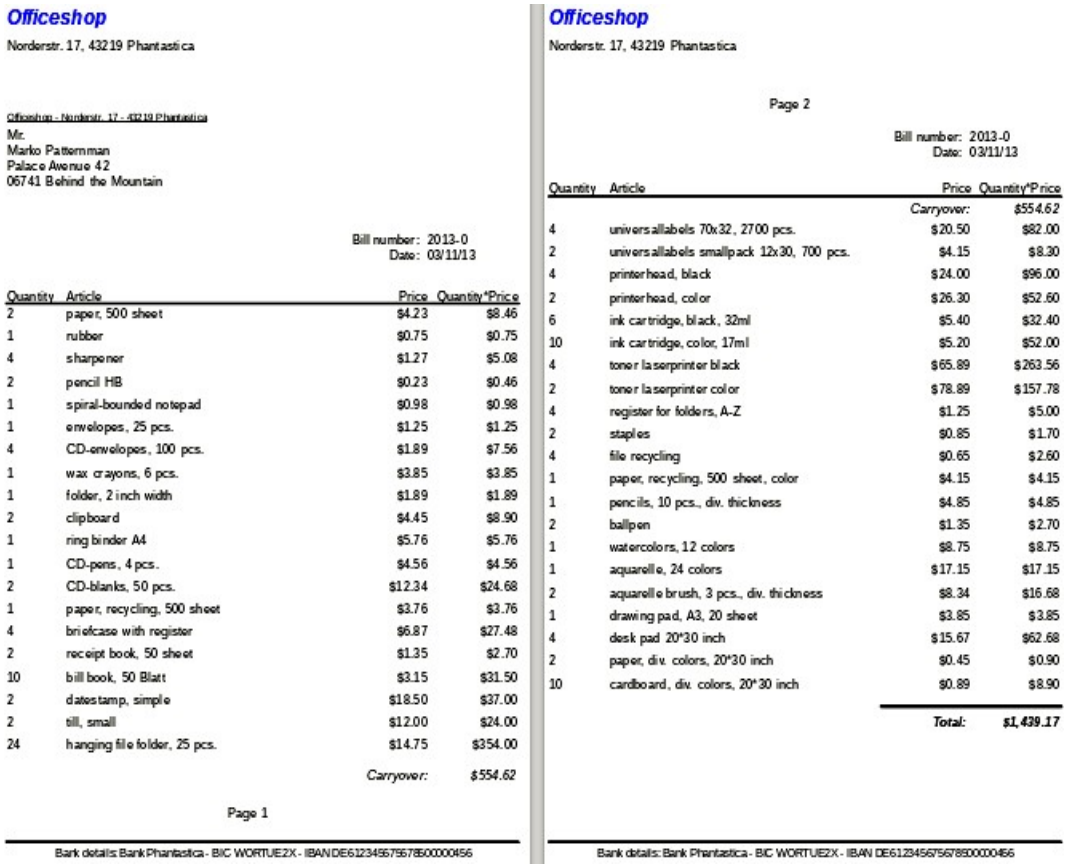
```
"Page "&INT([CounterBillNumber]/24)+1
```

La función INT redondea hacia abajo al entero más cercano. La primera página tiene un máximo de 20 elementos. Por lo tanto, la división da un resultado <1 para esta primera página. Se redondea a cero. Por lo tanto, debemos agregar 1 al número de página calculado para que aparezca 1 en la primera página. Del mismo modo para la página 2.

El informe todavía muestra un fallo estético. Una mirada cuidadosa a los elementos de la factura muestra que los tres elementos inferiores de las páginas son iguales. Esto se debe a que los registros simplemente se han copiado. En realidad, no son los mismos registros, sino diferentes para el mismo producto que se han procesado independientemente uno del otro. Si se agrupara por tipo de producto, cada producto aparecería sólo una vez con el número total ordenado.

Se debe intentar eliminar tantos cálculos y agrupaciones como sea posible del Generador. En lugar de usar los grupos del Generador de informes, es mejor usar las funciones de agrupación en el editor de consultas. Para que el Generador de informes procese la consulta fácilmente, conviértala en una vista. De lo contrario, el Generador de informes intentará mejorar la consulta con sus propias funciones de agrupación y clasificación, lo que puede conducir rápidamente a una codificación poco práctica.

El resultado final es el siguiente:



Officeshop

Norderstr. 17, 43219 Phantastica

Officeshop - Norderstr. 17 - 43219 Phantastica

Mr.
Marko Patternman
Palace Avenue 42
06741 Behind the Mountain

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
2	paper, 500 sheet	$4.23	$8.46
1	rubber	$0.75	$0.75
4	sharpener	$1.27	$5.08
2	pencil HB	$0.23	$0.46
1	spiral-bounded notepad	$0.98	$0.98
1	envelopes, 25 pcs.	$1.25	$1.25
4	CD-envelopes, 100 pcs.	$1.89	$7.56
1	wax crayons, 6 pcs.	$3.85	$3.85
1	folder, 2 inch width	$1.89	$1.89
2	clipboard	$4.45	$8.90
1	ring binder A4	$5.76	$5.76
1	CD-pens, 4 pcs.	$4.56	$4.56
2	CD-blanks, 50 pcs.	$12.34	$24.68
1	paper, recycling, 500 sheet	$3.76	$3.76
4	briefcase with register	$6.87	$27.48
2	receipt book, 50 sheet	$1.35	$2.70
10	bill book, 50 Blatt	$3.15	$31.50
2	datestamp, simple	$18.50	$37.00
2	till, small	$12.00	$24.00
24	hanging file folder, 25 pcs.	$14.75	$354.00
		Carryover:	*$554.62*

Page 1

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567800000456

Officeshop

Norderstr. 17, 43219 Phantastica

Page 2

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
		Carryover:	*$554.62*
4	universallabels 70x32, 2700 pcs.	$20.50	$82.00
2	universallabels smallpack 12x30, 700 pcs.	$4.15	$8.30
4	printerhead, black	$24.00	$96.00
2	printerhead, color	$26.30	$52.60
6	ink cartridge, black, 32ml	$5.40	$32.40
10	ink cartridge, color, 17ml	$5.20	$52.00
4	toner laserprinter black	$65.89	$263.56
2	toner laserprinter color	$78.89	$157.78
4	register for folders, A-Z	$1.25	$5.00
2	staples	$0.85	$1.70
4	file recycling	$0.65	$2.60
1	paper, recycling, 500 sheet, color	$4.15	$4.15
1	pencils, 10 pcs., div. thickness	$4.85	$4.85
2	ballpen	$1.35	$2.70
1	watercolors, 12 colors	$8.75	$8.75
1	aquarelle, 24 colors	$17.15	$17.15
2	aquarelle brush, 3 pcs., div. thickness	$8.34	$16.68
1	drawing pad, A3, 20 sheet	$3.85	$3.85
4	desk pad 20*30 inch	$15.67	$62.68
2	paper, div. colors, 20*30 inch	$0.45	$0.90
10	cardboard, div. colors, 20*30 inch	$0.89	$8.90
		Total:	*$1,439.17*

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567800000456

Aquí todos los ítems ocurren solo una vez. Las 12 carpetas colgantes originales en la parte inferior de la página 1 ahora son 24 carpetas colgantes.

Cantidad*precio han sido recalculados en consecuencia.

Impresión de informes para el registro actual en un formulario

Especialmente en el tipo de producción de facturas que se muestra en el ejemplo anterior, puede ser útil preparar y previsualizar una nueva impresión después de cada entrada de un artículo. El contenido de la factura debe determinarse utilizando un formulario, y a continuación sigue la impresión de los documentos individuales.

Los informes no se pueden iniciar con un filtro utilizando macros. Sin embargo, la consulta que se utiliza para hacer el informe se puede filtrar de antemano. Esto se hace mediante una consulta parametrizada usando la fórmula:

```
SELECT * FROM "bill" WHERE "ID" = :ID
```

o mediante una consulta que se suministra con datos utilizando una tabla de filtro de una línea:

```
SELECT * FROM "bill" WHERE "ID" = (SELECT "Integer" FROM "Filter" WHERE "ID" = TRUE)
```

En una consulta parametrizada, el contenido debe insertarse en el campo de diálogo correspondiente después de iniciarlo.

Cuando se usa una tabla de filtro para controlar el proceso, su contenido debe ser escrito por una macro. Por lo tanto, ya no es necesaria una entrada separada, lo que facilita la vida del usuario. El proceso se describe a continuación.

Construir la tabla de filtros

La tabla filtro debe contener solo un registro a la vez. Esto significa que su clave principal puede ser un campo Sí / No. Otros campos en la tabla se nombran de tal manera que quede claro qué tipo de contenido contienen. En el ejemplo, el campo que debe filtrar la clave primaria de la tabla *Bill* se llama Integer, ya que la clave en sí es de este tipo. Para otras posibilidades de filtrado, se pueden agregar otros campos más adelante. El filtro Integer puede usarse en diferentes momentos para varias tablas diferentes, ya que el contenido antiguo siempre se sobrescribe con el nuevo antes de imprimir. Pero este uso múltiple solo funciona en una base de datos de un solo usuario (Base con HSQLDB interno). En una base de datos multiusuario, siempre existe la posibilidad de que algún otro usuario cambie el valor del filtro en una de las tablas normales mientras se realiza la consulta que usa el filtro,

Nombre de campo	***Tipo de campo***
ID	Sí/No [BOOLEAN]
Integer	Entero [INTEGER]

Esta tabla se llena al principio con un registro.

Para este propósito, el campo ID debe establecerse para que tenga el valor *Sí* (“TRUE” en SQL).

Crear la macro para presentar el informe filtrado

Para llamar un solo informe, el formulario debe contener en algún lugar la clave principal de la tabla Bill. Esta clave primaria se lee y se transfiere a la tabla filtro mediante una macro. Luego, la macro inicia el informe deseado.

```
SUB Filter_and_Print
   DIM oDoc AS OBJECT
   DIM oDrawpage AS OBJECT
   DIM oForm AS OBJECT
   DIM oField AS OBJECT
   DIM oDatasorce AS OBJECT
   DIM oConnection AS OBJECT
   DIM oSQL_Command AS OBJECT
   DIM stSQL AS STRING
   oDoc = thisComponent
   oDrawpage = oDoc.Drawpage
   oForm = oDrawpage.Forms.getByName("MainForm")
   oField = oForm.getByName("fmtID")
   oDatasource = ThisComponent.Parent.CurrentController
   If NOT (oDatasorce.isConnected()) THEN
      oDatasource.connect()
   END IF
   oConnection = oDatasource.ActiveConnection()
   oSQL_Command = oConnection.createStatement()
   stSql = "UPDATE ""Filter"" SET ""Integer"" = '"+oField.GetCurrentValue()+"' WHERE
      ""ID"" = TRUE"
   oSQL_Command.executeUpdate(stSql)
   ThisDatabaseDocument.ReportDocuments.getByName("bill").open
END SUB
```

En este ejemplo, el formulario se llama MainForm. La clave primaria se llama fmtID. Este campo clave no tiene que estar visible para que la macro acceda a él. El valor del campo se lee y el comando ACTUALIZAR lo escribe en la tabla. Luego se lanza el informe. La vista a la que se refiere el informe se expande en una condición:

```
… WHERE "bill_ID" = IFNULL((SELECT "Integer" FROM "Filter" WHERE "ID" =
```

```
TRUE),"bill_ID") …
```

El campo entero se lee. Si no tiene valor, se establece en bill_ID. Esto significa que se muestran todos los registros, no solo el registro del filtro. Entonces, desde la misma vista, se pueden imprimir todos los registros almacenados.

Coloración alternativa de fondo de líneas

Cuando está leyendo líneas individuales en una tabla dentro de un informe, es fácil para el ojo deslizarse hacia arriba o hacia abajo en una línea. Colorear el fondo de al menos una línea ayuda a prevenir esto. En el siguiente ejemplo1, las líneas simplemente se colorean alternativamente. El resultado se ve así:

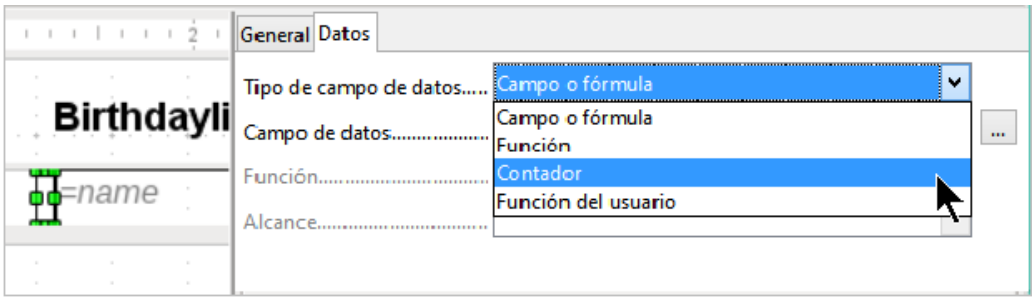
Birthdaylist

Kathrin	01/17/84
Sally	02/15/91
Mick	03/03/53
Hanne	04/13/70
Meike	04/13/71
Lara	04/23/85

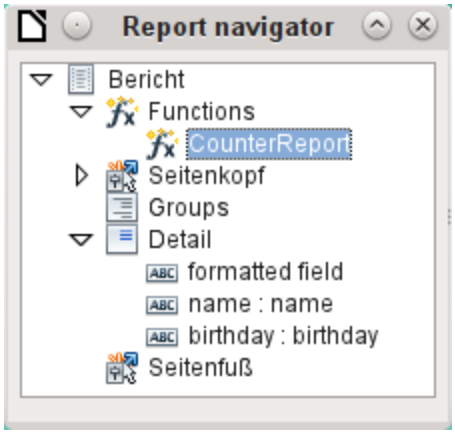
La base del informe es una consulta con nombres y fechas. La tabla original se consulta ordenando por fecha (mes y día) para mostrar la secuencia de cumpleaños a lo largo del año. Esto se hace por:

```
... ORDER BY MONTH("birthday") ASC, DAY("birthday") ASC
```

Para obtener los colores alternos, necesitamos crear una función que luego pueda usar algún valor para establecer condiciones, que a su vez determinan el color de fondo. Creamos un campo de texto en el informe y, usando **Propiedades> Datos> Tipo de campo de datos**, definimos un contador.

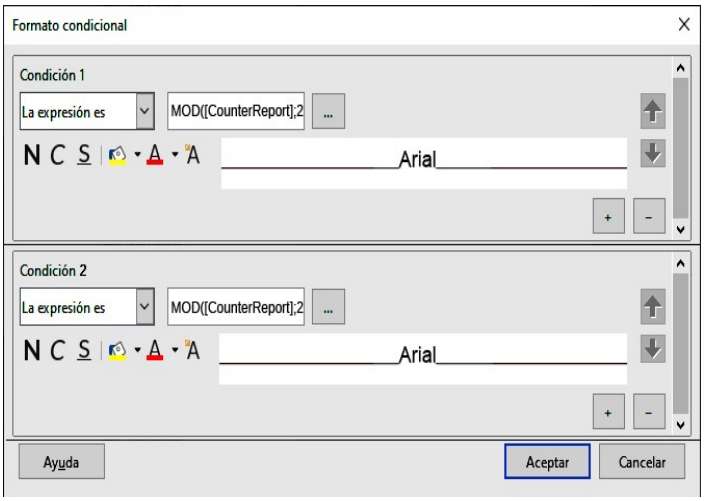


Necesitamos el nombre de la función para el formato condicional. El contador real no necesita aparecer en la expresión. El nombre se puede leer directamente desde el campo. Si el campo se elimina nuevamente, se puede acceder al nombre de la función usando **Ver> Navegador de informes.**



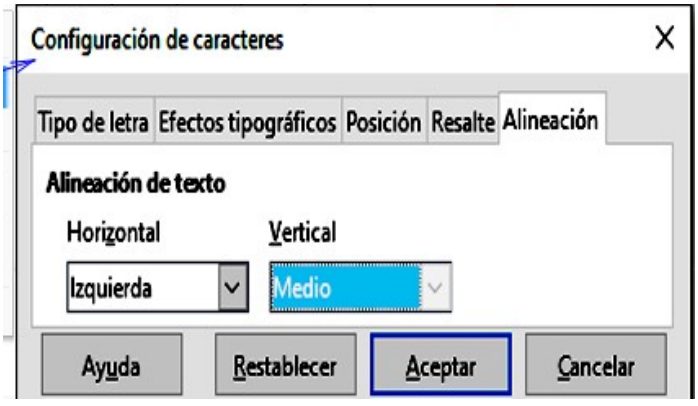
Ahora el contador debe usarse para dar a cada campo de texto su propio formato. La condición es una expresión que no está directamente conectada al campo. Por lo tanto, se configura como Condition 1 > Expression is > MOD([CounterReport];2)>0.

MOD calcula el resto de una división. Para todos los números impares, esto es mayor que 0; para números pares es 0 precisamente. Por lo tanto, a las líneas 1, 3 y 5 se les asignará este formato.

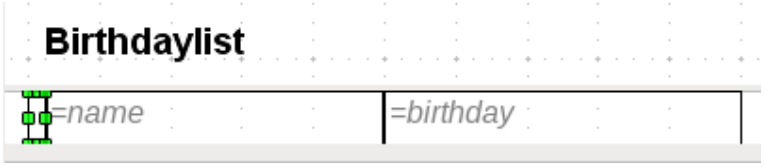


La segunda condición se formula como exactamente lo contrario y se le asigna un formato correspondiente. En realidad, esta segunda condición podría omitirse y establecerse un formato predeterminado en las propiedades de cada campo. El formato condicional especificado en la primera condición reemplazaría este valor predeterminado cada vez que se satisficiera la condición.

Como el formato condicional sobrescribe todo el formato predeterminado, se debe incluir una alineación de texto para este formato utilizando la configuración de caracteres.



Aquí las letras deben centrarse verticalmente en el campo de texto coloreado. La alineación horizontal es la estándar, pero se debe agregar una sangría para que las letras no se amontonen en el lado izquierdo del campo de texto.
Los experimentos con la adición de espacios al contenido de una consulta o fórmula no han llevado a una sangría adecuada del texto. Los espacios principales son simplemente recortados.



Un método más exitoso es colocar un campo de texto antes del texto real pero sin vincularlo a un campo de datos. Este campo de texto está formateado condicionalmente de la misma manera que los demás, de modo que aparece una sangría visible consistente en la impresión.

Informes en dos columnas

Con técnicas de consulta inteligentes, puede hacer un informe con varias columnas de modo que la secuencia horizontal corresponda a registros sucesivos:

Birthdaylist

Kathrin	01/17/84	Sally	02/15/91
Mick	03/03/53	Hanne	04/13/70
Meike	04/13/71	Lara	04/23/85
Monika	06/05/86	Karl	07/01/67
Paul	07/11/89	Egon	07/23/67
Susanne	08/02/65	Georg	08/28/95
Ysabelle	09/17/89	Maik	10/28/93
John	11/19/97	Erkan	12/17/75
Johann	12/23/91		

El primer registro va a la columna izquierda, el segundo a la derecha. Los registros se ordenan según la secuencia de cumpleaños dentro del año.

Ordenar por fecha de nacimiento hace que la consulta para este informe sea bastante larga. Si el ordenamiento fuera por la clave primaria de la tabla subyacente, sería mucho más corto. El criterio de clasificación utiliza un bloque de texto recurrente que se explica más adelante.

He aquí la consulta que subyace al informe:

```
SELECT "T1"."name" "name1", "T1"."birthday" "birthday1", "T2"."name" "name2",
   "T2"."birthday" "birthday2"
FROM ( SELECT "name", "birthday", "rowsNr" AS "row"
   FROM ( SELECT "a".*,
   ( SELECT COUNT( "ID" ) FROM "birthdays"
      WHERE RIGHT( '0' || MONTH( "birthday" ), 2 ) ||
      RIGHT( '0' || DAY  ( "birthday" ), 2 ) || "ID"
      <= RIGHT( '0' || MONTH( "a"."birthday" ), 2 ) ||
      RIGHT( '0' || DAY  ( "a"."birthday" ), 2 ) || "a"."ID" )
      AS "rowsNr"
      FROM "birthdays" AS "a" )
WHERE MOD( "rowsNr", 2 ) > 0 )   AS "T1"
   LEFT JOIN ( SELECT "name", "birthday", "rowsNr" - 1 AS "row"
   FROM  ( SELECT "a".*,
   ( SELECT COUNT( "ID" ) FROM "birthdays"
      WHERE RIGHT( '0' || MONTH( "birthday" ), 2 ) ||
      RIGHT( '0' || DAY( "birthday" ), 2 ) || "ID"
      <= RIGHT( '0' || MONTH( "a"."birthday" ), 2 ) ||
```

```
      RIGHT( '0' || DAY( "a"."birthday" ), 2 ) || "a"."ID" )
      AS "rowsNr"
FROM "birthdays" AS "a" )
   WHERE MOD( "rowsNr", 2 ) = 0 ) AS "T2"
   ON   "T1"."row" = "T2"."row"
   ORDER BY "T1"."row"
```

Hay dos subconsultas idénticas en esta consulta. Las dos primeras columnas se relacionan con la subconsulta con el alias T1 y las dos últimas con la subconsulta T2.

Las subconsultas proporcionan otro campo (filasNr) además de los de la tabla *Birthdays*, lo que permite distinguir líneas y ordenar los registros. La consulta principal usa esto junto con la función de numeración de línea en las consultas (consulte el Capítulo 5, Consultas).

```
RIGHT( '0' || MONTH( "birthday" ), 2 ) || RIGHT( '0' || DAY( "birthday" ), 2 ) || "ID"
```

Esta fórmula permite establecer una secuencia única de registros. Como los registros de ejemplo se ordenarán por fecha, sería fácil pensar que solo se debe usar la fecha en la comparación. Es más difícil en la práctica porque no son las fechas de nacimiento en sí mismas sino su secuencia dentro de un solo año lo que cuenta. Los problemas adicionales son causados por valores de fecha idénticos que impiden que se establezca una secuencia única. Por lo tanto, la clasificación no es solo por mes y día, sino también por la clave primaria única. Y para evitar que el mes 10 llegue antes del mes 2, se coloca un cero a la izquierda antes de cada mes usando || cuando los criterios de clasificación se pongan juntos; si el mes ya tiene 2 dígitos, esto se elimina posteriormente utilizando RIGHT(… , 2).

SELECT COUNT ("ID") proporciona el número de registros cuya combinación de mes, día y clave primaria es menor o igual que la del registro actual en la tabla Cumpleaños. Aquí tenemos una subconsulta de correlación (consulte el capítulo sobre "Consultas").

MOD ("rowsNr", 2) determina si este número es par o impar. MOD da el resto de una división, en el ejemplo, división por 2. Esto hace que las filasNr den los valores "1" y "0" alternativamente. Esto a su vez distingue las consultas para T1 y T2.

En el siguiente nivel de consulta de T2, Fila se define como "rowsNr" –1. De esta manera, T1 y T2 son directamente comparables.

T1 está vinculado a T2 usando LEFT JOIN, de modo que todas las fechas en la tabla *Birthdays* también se muestran para números de registro impares. Cuando se combinan T1 y T2, ahora se puede hacer referencia directa a las filas reales: "T1". "Row" = "T2". "Row".

Finalmente, todo el contenido se ordena según el valor de la fila, que es lo mismo para T1 y T2. El informe también puede usar esto directamente usando su función de agrupación.

Birthdaylist

January		February	
Kathrin	01/17/84	Sally	02/15/91
March		**April**	
Mick	03/03/53	Hanne	04/13/70
		Meike	04/13/71
		Lara	04/23/85
May		**June**	
		Monika	06/05/86
July		**August**	
Karl	07/01/67	Susanne	08/02/65
Paul	07/11/89	Georg	08/28/95
Egon	07/23/67		
September		**October**	
Ysabelle	09/17/89	Maik	10/28/93
November		**December**	
John	11/19/97	Erkan	12/17/75
		Johann	12/23/91

Es mucho más complicado crear técnicas de consulta para hacer subdivisiones además del formato de 2 columnas. En tales casos, las líneas vacías aparecen en el medio del informe, mientras que en el ejemplo anterior de 2 columnas, solo aparecen al final. El Generador de informes no funciona bien con este tipo de consulta. Por lo tanto, en lugar de esto, utilizaremos dos vistas vinculadas.

Primero crearemos la siguiente vista:

```
SELECT "a"."ID", "a"."name", "a"."birthday",
       MONTH( "a"."birthday" )  AS "monthnumber",
            ( SELECT COUNT( "ID" ) FROM "birthdays"
              WHERE MONTH( "birthday" ) = MONTH( "a"."birthday" )
              AND RIGHT( '0' || DAY( "birthday" ), 2 ) || "ID" <=
              RIGHT( '0' || DAY( "a"."birthday" ), 2 ) || "a"."ID" )
       AS "monthcounter"
FROM "birthdays" AS "a"
```

Todos los campos de la tabla *Birthdays* están incluidos. Además, el mes se incluye como un número. A la tabla *Birthdays* se le asigna el alias "a", para que se pueda acceder a la tabla con una subconsulta correlacionada.

La subconsulta cuenta todos los registros dentro de un mes cuyos valores de fecha tienen un número de día menor o igual. Para fechas idénticas, la clave primaria determina qué es lo primero. Esta técnica es la misma que en el ejemplo anterior.

La vista report_month_two_columns accede a la vista mes-número. Aquí solo se dan extractos de esta vista:

```
SELECT
      "Tab1"."name" AS "name1",
      "Tab1"."birthday" AS "birthday1",
      1 AS "monthnumber1",
      IFNULL  ("Tab1"."monthcounter",999) AS "monthcounter1",
      'January' AS "month1",
```

```
        "Tab2"."name"  AS "name2",
        "Tab2"."birthday" AS "birthday2",
        2 AS "monthnumber2",
        IFNULL  ("Tab2"."monthcounter",999) AS "monthcounter2",
        'February' AS "month2"
FROM
        (SELECT * FROM "monthnumber" WHERE "monthnumber" = 1) AS "Tab1"
        RIGHT JOIN(SELECT * FROM "monthnumber" WHERE "monthnumber" = 2) AS "Tab2" ON
        "Tab1"."monthcounter" = "Tab2"."monthcounter"
UNION
        SELECT "Tab1"."name" AS "name1",
        "Tab1"."birthday" AS "birthday1",
        1 AS "monthnumber1",
        IFNULL ("Tab1"."monthcounter",999) AS "monthcounter1",
        'January' AS "month1",
        "Tab2"."name" AS "name2",
        "Tab2"."birthday" AS "birthday2",
        2 AS "monthnumber2",
        IFNULL ("Tab2"."monthcounter",999) AS "monthcounter2",
        'February'  AS "month2"
FROM
        (SELECT * FROM "monthnumber" WHERE "monthnumber" = 1) AS "Tab1"
        LEFT JOIN(SELECT * FROM "monthnumber" WHERE "monthnumber" = 2) AS "Tab2"
        ON "Tab1"."monthcounter" = "Tab2"."monthcounter"
UNION
...
ORDER BY "monthnumber1", "monthcounter1", "monthcounter2"
```

Primero, la subconsulta lee del mes número todas las fechas para las que mes = 1. Esta selección recibe el alias Tab1. Al mismo tiempo, todas las fechas para mes número = 2 se leen en Tab2. Estas tablas están vinculadas con una RIGHT JOIN (UNIÓN DERECHA), de modo que se muestran todos los registros de Tab2 pero solo aquellos registros de Tab1 con el mismo contador de mes que Tab2.
Las columnas de la vista deben tener nombres diferentes para que cada columna tenga un alias. También se ingresa un 1 directamente como el número de mes para Tab1 y enero como el mes1. Estas entradas también aparecen cuando no hay registros de Tab1 pero aún algunos registros en Tab2. Si no hay un contador de mes disponible, se ingresa el valor 999.

Como Tab1 está vinculada a Tab2 mediante una RIGHT JOIN (UNIÓN DERECHA), puede suceder que cuando haya menos registros en Tab1, se muestren campos vacíos. Dado que la clasificación posterior pondría dichos registros por delante de todos los registros con contenido, en su lugar se ingresa un número muy alto.

La creación de columnas para Tab2 es similar. Aquí, aunque no es necesario usar el código IFNULL ("Tab2". "Monthcounter", 999), ya que, con una RIGHT JOIN (UNIÓN DERECHA) para Tab2, se mostrarán todas las filas de Tab2, pero ya no se dará más el supuesto de que se puedan crear más filas desde Tab1 que desde Tab2.

Precisamente este problema se resuelve mediante la vinculación de dos consultas. Con UNION, se muestran todos los registros de la primera consulta y todos los registros de la segunda consulta. Los registros de la segunda consulta aparecen solo si no son idénticos a un registro anterior. Esto significa que UNION funciona como DISTINCT.

Usando UNION, la misma consulta se vuelve a preguntar, solo Tab1 y Tab2 ahora están vinculadas con una LEFT JOIN (UNIÓN IZQUIERDA) Esto hace que se muestren todos los registros de Tab1 incluso si Tab2 contiene menos registros que Tab1.

Para los meses 3 y 4, 5 y 6, y así sucesivamente, se utilizan consultas exactamente equivalentes y se vuelven a unir a la consulta anterior utilizando UNION.

Finalmente, el resultado de la vista se ordena por mes número1, mes contador1 y mes contador2. No es necesario ordenar por mes número2, porque mes número1 ya da la misma secuencia de registro.

Fuentes de errores en los informes.

El generador de informes a veces oculta errores cuya causa exacta no puede determinarse fácilmente después. Aquí hay algunas fuentes de error y contramedidas útiles.

No aparece el contenido de un campo de una consulta

Se configura una base de datos para simular la venta de existencias. Una consulta calcula un precio total a partir del número de artículos comprados y el precio unitario.

```
SELECT "sales"."sum", "stock"."stock", "stock"."Price", "sales"."sum"*"stock"."Price" FROM
"sales", "stock"
WHERE "sales"."stock_ID" = "stock"."ID"
```

Esta consulta sirve como base del informe.

Sin embargo, si el campo `"sales"."sum"*"stock"."Price"` aparece en el informe, permanece vacío. Si, por otro lado, se proporciona un alias en la consulta, el Generador de informes puede acceder fácilmente a él:

```
SELECT "sales"."sum", "stock"."stock", "stock"."Price", "sales"."sum"*"stock"."Price" AS "TPrice"

FROM "sales", "stock" WHERE "sales"."stock_ID" = "stock"."ID"
```

El campo ahora accede a "Tprice" y presenta el valor correspondiente.

No se puede generar un informe.

A veces, se puede preparar un informe, pero luego no se puede generar o incluso guardar. Aparece un mensaje de error poco informativo:

```
“El informe no pudo ser producido. Se descubrió una excepción del tipo
com.sun.star.lang.WrappedTargetExeption "
```

Aquí puede ser útil leer la información adicional adjunta al mensaje. Si ve una referencia a SQL, significa que el Generador de informes no puede interpretar correctamente el código SQL en la fuente de datos.

Puede ser útil usar Navegador de Informes para configurar **Datos> Analizar orden SQL> No**. Desafortunadamente, esta solución hace que los grupos existentes dejen de funcionar.

Una mejor manera es utilizar una vista en lugar de una consulta como base para su informe. Esto se configura de tal manera que parece una tabla para el Generador de informes y se puede editar sin ninguna dificultad. Incluso los requisitos de clasificación de la vista funcionan sin problemas.

LibreOffice

Guía de Base

Capítulo 7
Vincular bases de datos

Notas generales sobre enlace en bases de datos

Con Base, puede usar documentos en LibreOffice Writer y Calc de varias maneras como fuentes de datos. Esto significa que el uso de Base no está necesariamente vinculado al registro de bases de datos en la configuración de LibreOffice. Los formularios externos también pueden interactuar directamente con Base, siempre que se proporcione la ruta a las fuentes de datos.

Registro de bases de datos.

Muchas funciones, como imprimir etiquetas o usar datos para cartas de formulario, requieren el registro previo de una base de datos en la configuración de LibreOffice.

Usando **Herramientas > Opciones > LibreOffice Base > Bases de datos > Nuevo**, puede registrar una base de datos para su uso posterior por otros componentes de LibreOffice.

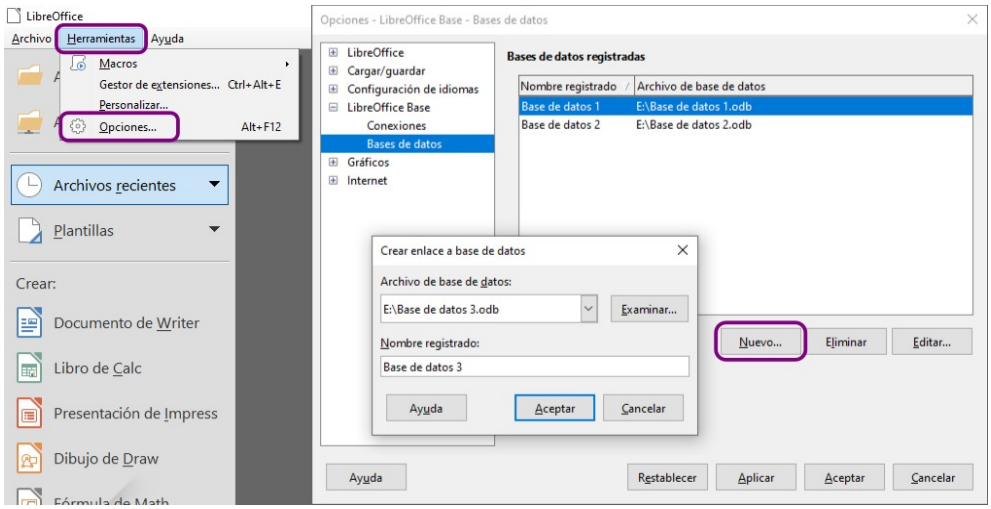


Figura 185

Encuentre la base de datos utilizando el explorador de archivos y conéctela a LibreOffice de forma similar a un formulario simple. Dé a la base de datos un nombre registrado adecuadamente e informativo, por ejemplo, el nombre del archivo de la base de datos. El nombre sirve como un alias, que también se puede usar en consultas a la base de datos.

Buscador de orígenes de datos

El navegador de orígenes de datos en Writer y Calc proporciona acceso a tablas y consultas de todas las bases de datos ingresadas bajo sus nombres registrados. Para abrir el navegador en Calc use **Ver > Orígenes de datos** o presione las teclas *Ctrl+Mayús+F4* o haga clic en el icono en la barra de herramientas *Estándar*.

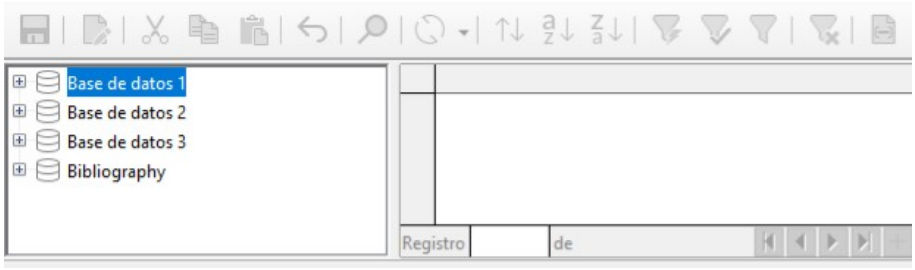


Figura 186

El icono de *Orígenes de datos* no suele verse en la barra de herramientas *Estándar*. Para hacerlo visible, haga clic con el botón derecho en el botón *Guardar* para abrir esta barra de herramientas. Desplácese hacia abajo hasta *Botones visibles*. Esto abre una lista de todos los botones.

Desplácese hacia abajo hasta cerca de la parte inferior para encontrar el botón *Orígenes de datos*. Haga clic para que sea visible en el extremo derecho de la barra de herramientas. El botón aparecerá visible en la barra de herramientas (ver Figura 187).

Figura 187

Consejo

Si está utilizando una computadora portátil, es posible que deba presionar *Ctrl+Fn+Mayús+F4*. La tecla *Fn* (función) permite usar las teclas *F1*,*F2*,… *F12* como teclas de función.

Los orígenes de datos registrados se muestran en el lado izquierdo del navegador de orígenes de datos, que se encuentra por defecto en la parte superior del espacio de trabajo (ver Figura 186). El origen de datos *Bibliography*, correspondiente a la bibliografía, se incluye en LibreOffice por defecto. Otros Orígenes de datos se listarán por sus nombres registrados.

Haga clic en el signo [+] a la izquierda del nombre de la base de datos para expandir el contenido y mostrar sus subcarpetas de consultas y tablas. Algunas subcarpetas de la base de datos no están disponibles aquí. Solo se puede acceder a los formularios e informes internos abriendo la base de datos.

Solo cuando hace clic en la carpeta *Tablas* se accede realmente a la base de datos. Para las bases de datos protegidas por una contraseña, se debe ingresar esta última en este punto.

A la derecha del árbol de nombres, puede ver la tabla que ha seleccionado. Se puede editar igual que en Base. Sin embargo, la entrada directa en las tablas debe realizarse con precaución en bases de datos relacionales muy complejas, ya que las tablas están vinculadas con claves foráneas. Por ejemplo, la base de datos que se muestra a continuación tiene tablas separadas para nombres de calles, códigos postales y ciudades.

Para una vista adecuada de los datos (pero sin la capacidad de editar), las consultas o vistas son más adecuadas.

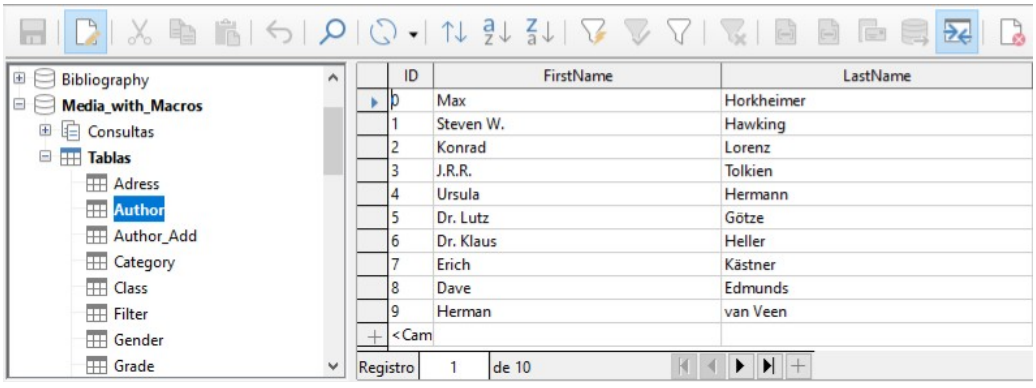


Figura 188

Muchos de los iconos en la barra de herramientas (en la parte superior de la Figura 188) serán familiares por la entrada de datos en las tablas. (Los iconos en su pantalla pueden ser diferentes de los que se muestran en la figura).

Los nuevos iconos principales son los de la última sección: *Datos en texto*, *Datos en campos*, *Combinar correspondencia*, *Origen de datos del documento actual*, *Mostrar/Ocultar Explorador*. Su uso se describe a continuación, utilizando la tabla *Reader* de la base de datos de *Media*.

Datos en texto

Consejo

Con este método, los datos se pueden insertar directamente en lugares específicos en un documento de texto o celdas específicas de una hoja de cálculo. Si bien los datos podrían escribirse en estas ubicaciones, insertarlos garantiza su precisión. Esto es importante cuando se utiliza la combinación de correspondencia, que se analiza más adelante.

Al enviar el mismo documento a diferentes personas, esto garantiza que todos recibirán exactamente los mismos datos.

Seleccione uno o más registros para activar las funciones *Datos en texto* y *Datos en campos*.

1	Datos en texto	2	Datos en campos	3	Combinar correspondencia

Figura 189

Si se elige *Datos en texto*, aparece un asistente para realizar el formateo necesario (Figura 190). Las tres opciones para ingresar datos como texto son: como *Tabla*, *Campos* o *Texto*.

La Figura 190 muestra la opción *Insertar Datos como: Tabla*. En el caso de los campos numéricos y de fecha, el formato de la base de datos se puede cambiar a otro formato elegido. De lo contrario, el formateo se realiza automáticamente cuando se seleccionan los campos de la tabla. La secuencia de campos se ajusta utilizando los botones con flechas.

Tan pronto como se seleccionan las columnas de la tabla, se activa el botón *Propiedades*. Esto le permite establecer las propiedades de tabla habituales para Writer (ancho de tabla, ancho de columna, etc.).

La casilla de verificación *Insertar título de tabla* determina si se requiere un encabezado de tabla. Si no está marcado, no se reservará una fila separada para los encabezados.

La fila elegida para el título de la tabla puede tomarse de los nombres de las columnas o el registro puede escribirse con el espacio restante para que los encabezados se puedan editar más adelante. Elija la opción S*olo crear fila*.

Puede usar el botón *Formato automático* para abrir un diálogo (Figura 191) con varios estilos de tabla predeterminados. Además del estilo predeterminado sugerido, todos los formatos se pueden renombrar. También puede agregar autoformatos; para hacer esto, primero cree una tabla en el formato requerido. Luego seleccione la tabla y haga clic derecho y vaya a **Estilo > Estilos de formato automático** y pulse en el botón *Añadir* para agregar su formato a la lista.

También puede formatear la tabla en Writer seleccionándola y elegir un formato de la lista *Estilos de tabla* en sección *Estilos* de la *Barra lateral*; la lista de estilos de tabla es la misma que la lista de formatos en el diálogo *Autoformato*.

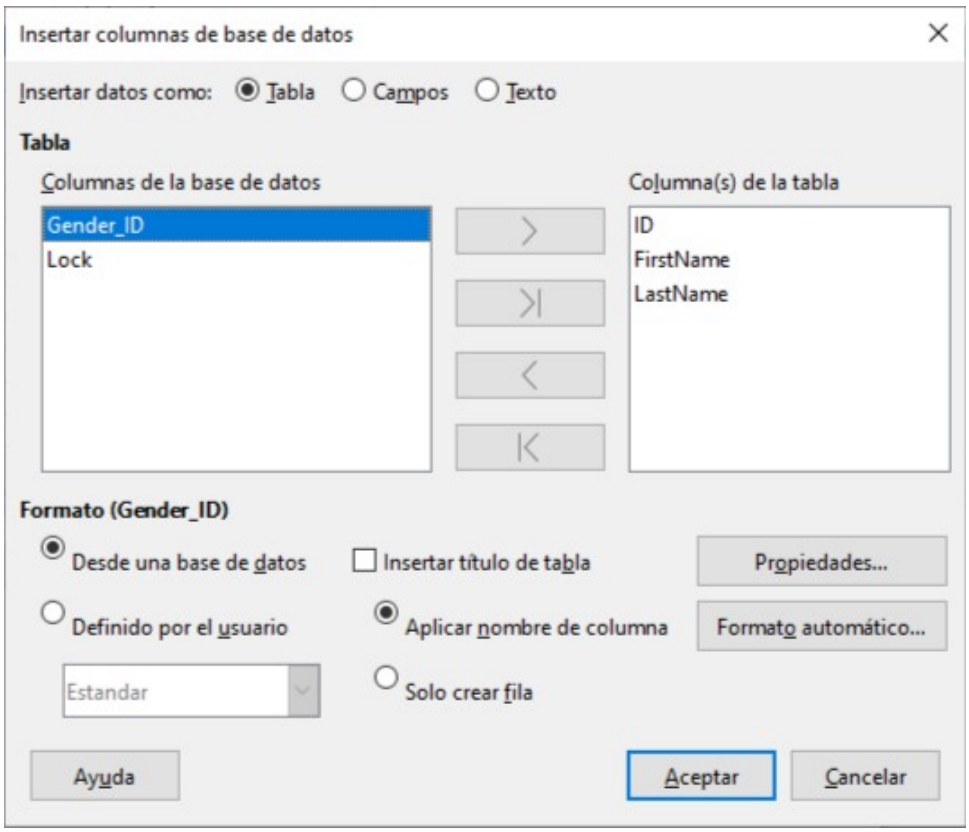


Figura 190

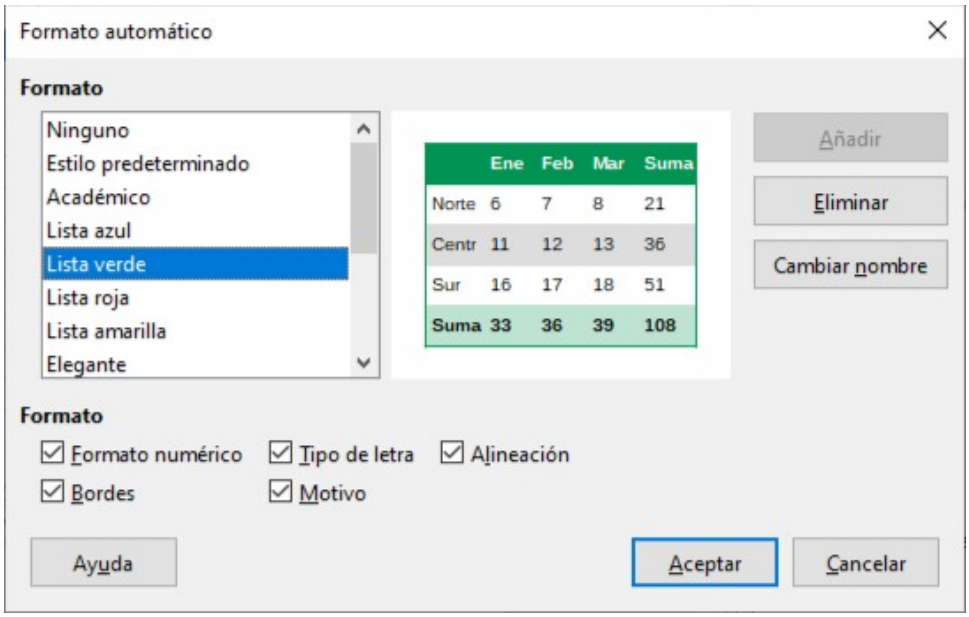


Figura 191

Para crear la tabla con los registros y columnas seleccionados, haga clic en *Aceptar* en el diálogo *Insertar columnas* de base de datos.

En segundo lugar, *Insertar Datos como: Campos* ofrece la posibilidad de utilizar un minieditor para colocar los diversos campos de la tabla sucesivamente en el texto (ver Figura 192). El texto creado de esta manera también se puede proporcionar con un estilo de párrafo. También en este caso, el formato de fechas y números se puede especificar por separado o se puede leer directamente desde la configuración de la tabla en la base de datos.

Los campos insertados en el texto de esta manera pueden eliminarse posteriormente individualmente o usarse para una combinación de correspondencia.

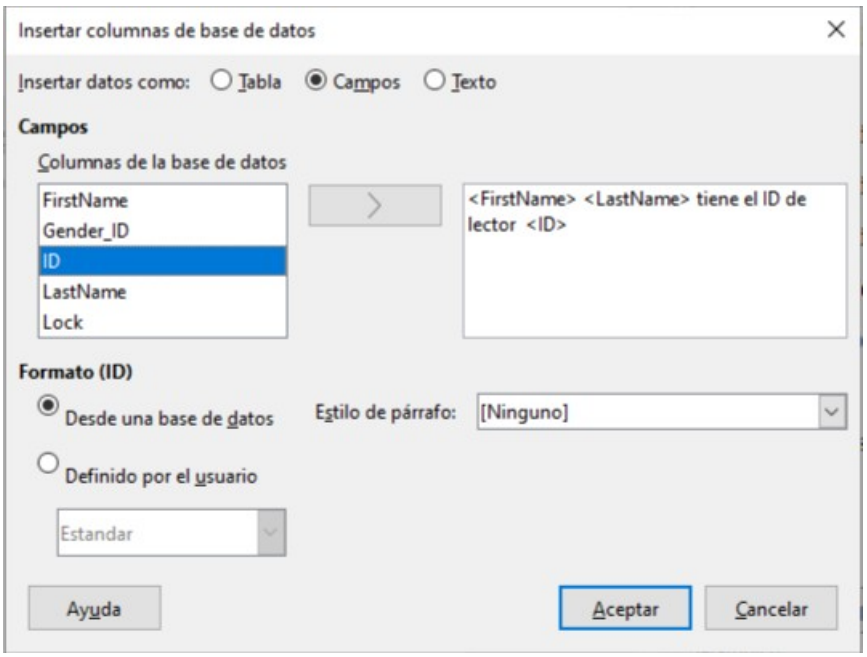


Figura 192

En tercer lugar, si elige *Insertar datos como: Texto,* la única diferencia que verá es que los datos no permanecen vinculados a la base de datos. Cuando los inserta como texto, solo se transfiere el contenido de los campos especificados y no hay un enlace real a la base de datos.

Los resultados de los dos últimos procedimientos se comparan a continuación.

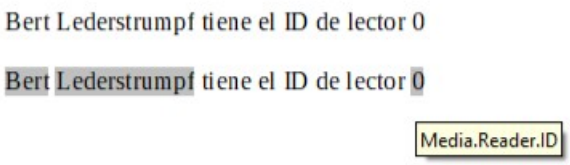


Figura 193: Arriba datos como texto; abajo datos como campos

Los campos tienen un fondo gris. Si coloca el cursor del ratón sobre los campos, se muestra que los campos están vinculados a la base de datos de *Media*, a la tabla *Reader* y, dentro de esta tabla, por ejemplo, el campo «0» está ligado al campo *ID*.

Entonces, por ejemplo, un doble clic en el campo *ID* abre la ventana de la Figura 194. Esto deja en claro qué campo se creó mediante el procedimiento *Insertar datos como campos*. Es el mismo tipo de campo que se muestra en **Insertar > Campo > Más campos > Base de datos**.

Es más sencillo crear dicho campo seleccionando el encabezado de columna de la tabla en el navegador de origen de datos y arrastrándolo al documento con el ratón. Puede crear una carta de formulario directamente de esta manera.

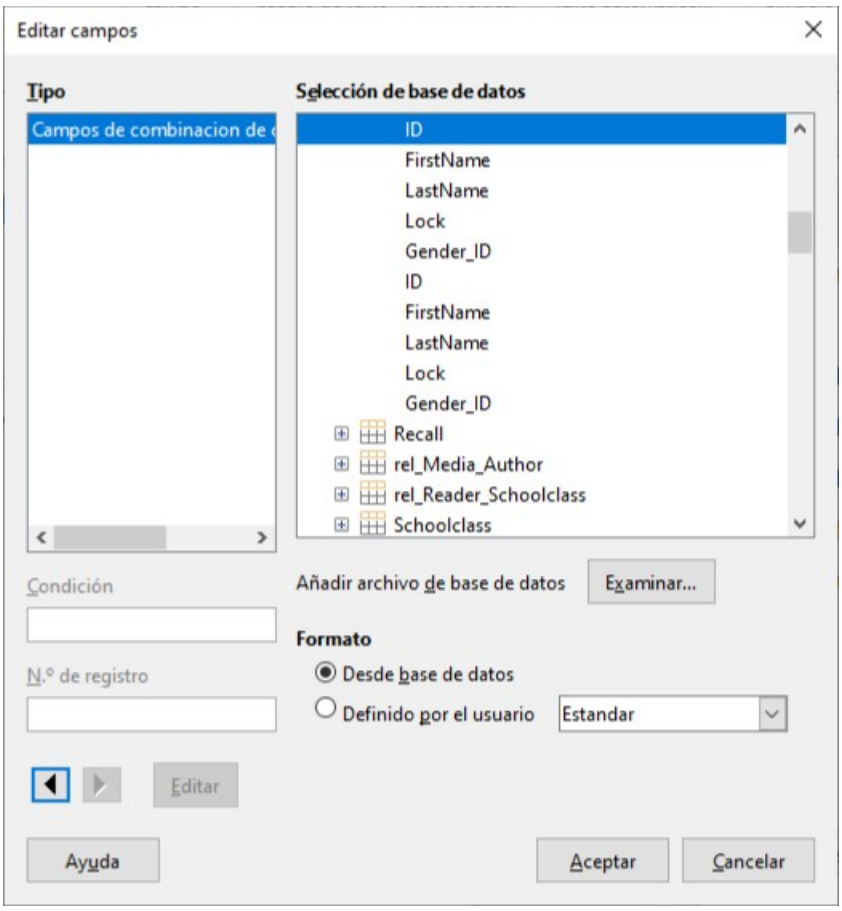


Figura 194

Datos en campos

Consejo

Este método es útil cuando se envía un documento a varias personas en el que a cada uno se le enviarán datos específicos solo para ellos. Esto se actualiza a través del botón *Datos en campos* (figura 189) y se hace a través de la combinación de correspondencia.

Por ejemplo, las bibliotecas envían avisos a las personas enumerando los medios (artículos) que han prestado y no han devuelto a tiempo. La lista generalmente será diferente para cada persona, pero todos recibirán una advertencia de algún tipo. Por supuesto, el tipo de advertencia dependerá del tiempo transcurrido desde la fecha de vencimiento del préstamo de los medios. Todos los que caigan en un período de tiempo particular recibirán la misma advertencia.

Figura 195

Para insertar datos en los campos:

3) Haga clic a la izquierda de una de las filas de prueba para resaltarla completamente.
4) Haga clic en el botón *Datos en texto* para abrir el asistente *Insertar columnas de base de datos* (ver figura 190).

5) Elija en la parte superior el botón de opción *Campos*.
6) Mueva los campos de la base de datos que desea usar de la lista izquierda a la derecha en el orden que desee. Se puede agregar texto así como en la figura 192.
7) Para aplicar un estilo de párrafo específico a estos campos, selecciónelo de la lista desplegable *Estilo de párrafo*.
8) Haga clic en *Aceptar*.

Como se anticipó, una vez que se ha insertado un campo, los valores que muestra se pueden cambiar. Para esto, seleccione un registro (fila) distinto y luego haga clic en el botón *Datos en campos*.

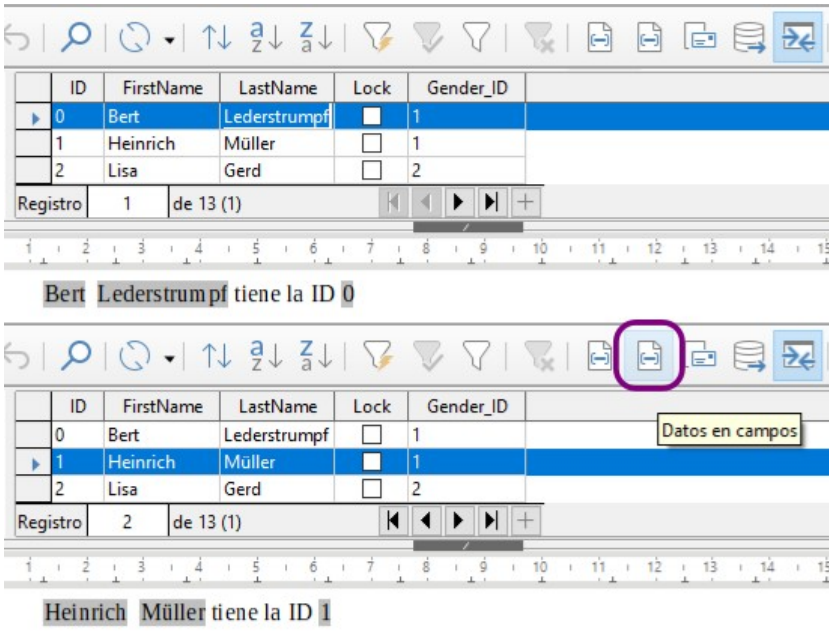


Figura 196: Arriba, texto ingresado mediante Insertar datos como campos. Abajo, texto cambiado mediante el botón Datos en campos.

Combinar correspondencia

El botón *Combinar correspondencia* (ver figura 197) inicia el *Asistente para combinar correspondencia*. Una carta de formulario reúne los datos de diferentes tablas, por lo que primero debe iniciar la base de datos. En la base de datos, cree una nueva consulta para que los datos necesarios estén disponibles.

Figura 197

Para iniciar la base de datos, haga clic derecho en la base de datos o en una de sus tablas o consultas. Esto actualiza inmediatamente la pantalla en el navegador de origen de datos. Después de eso, puede abrir el asistente para combinar correspondencia haciendo clic en el botón correspondiente.

Origen de datos del documento actual

Haga clic en el botón *Origen de datos del documento actual* para abrir una vista directa de la tabla que forma la base de los datos insertados en el documento. En el ejemplo anterior, se mostrará la tabla *Reader* de la base de datos de *Media*.

Mostrar u ocultar el explorador

Al activar o desactivar el botón *Mostrar/ocultar Explorador*, se muestra u oculta el árbol de directorios a la izquierda de la vista de tabla. Esto permite tener más espacio, si es necesario, para la visualización de los datos. Para acceder a otra tabla, debe volver a presionar el icono.

Crear documentos de combinación de correspondencia

También se puede acceder al *Asistente para combinar correspondencia* desde el navegador de la base de datos. Este asistente permite que el campo de dirección y el saludo se construyan a partir de una fuente de datos en pequeños pasos. En principio, puede crear estos campos sin utilizar el asistente. En el ejemplo trabajaremos a través de los pasos del asistente.

Esta vez, la fuente de datos será una consulta, específicamente *Readeradresses* de *Media*. Búsquelo en la lista desplegable *Consultas* como lo hizo anteriormente en *Tablas* con *Reader*.

Consejo

Cierre cualquier documento de texto en Writer si tiene enlaces a una base de datos. No puede establecer un nuevo enlace al documento cuando el antiguo todavía está activo. Comience con un nuevo documento, que puede ser sin título o una plantilla de carta que no contenga enlaces.

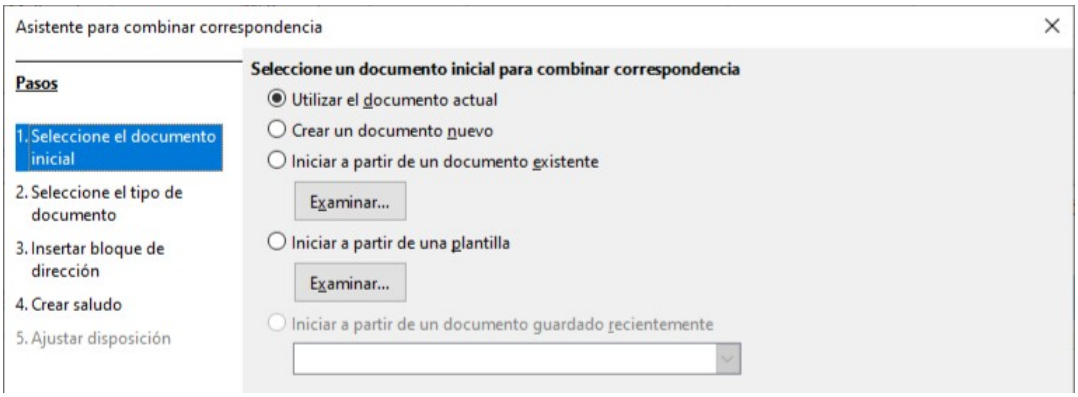


Figura 198

El ejemplo que se mostrará a continuación se realiza desde el asistente de Writer **Herramientas > Asistente para combinar correspondencia**, pero puede hacerse desde el botón *Combinar correspondencia* de la figura 197.

El documento inicial del formulario es el documento al que se vincularán los campos de la base de datos.

El documento combinado es el que contiene los datos de las distintas personas que recibirán los formularios. En el documento combinado no hay vinculación con la fuente de datos. Es similar a la salida *Insertar datos como texto*.

El *Asistente para combinar correspondencia* puede generar cartas o correos electrónicos utilizando registros de la base de datos. En este ejemplo crearemos cartas usando la tabla *Reader* de la base de datos de *Media*.

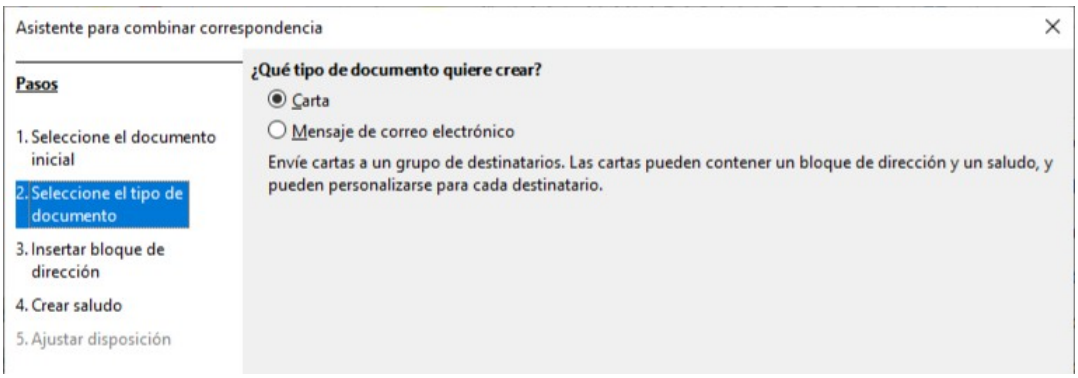


Figura 199

Insertar bloque de dirección permite una configuración más extensa. En el punto 1, la lista de direcciones sugeridas proviene de la consulta o tabla actualmente seleccionada en la base de datos actual. Asegúrese de escoger la consulta *Readeradresses* de la base de datos *Media.*

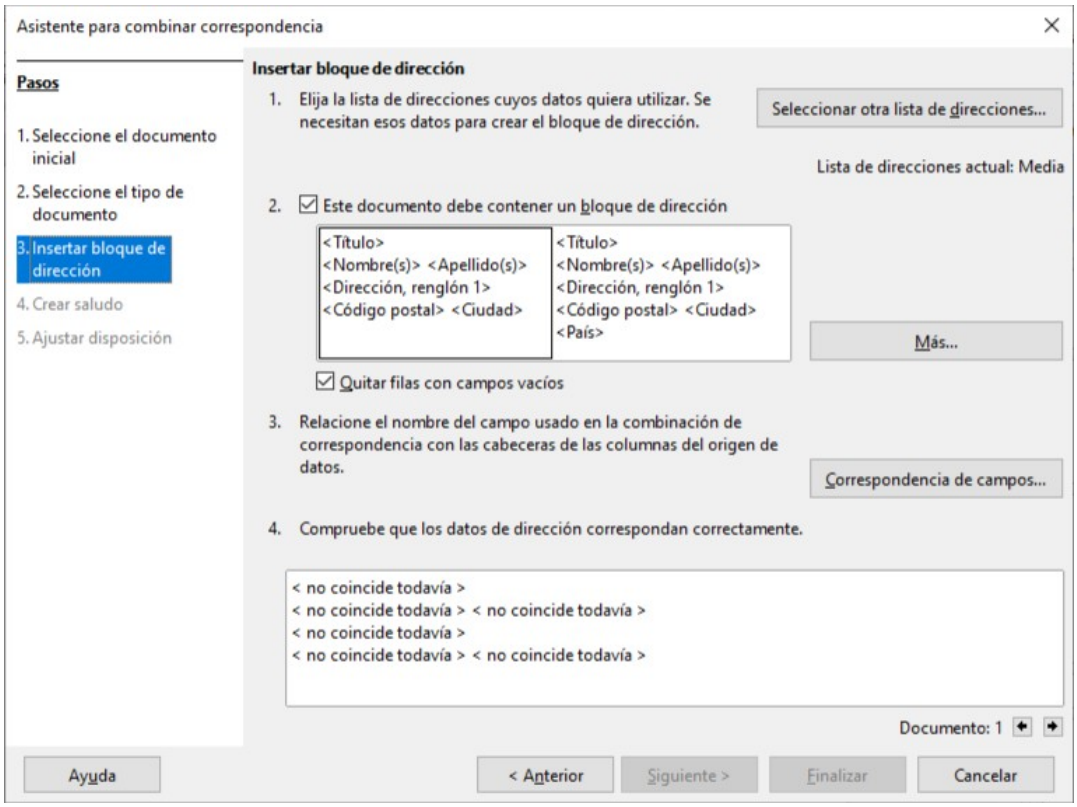


Figura 200

El punto 2 establece el aspecto general del bloque de direcciones, pero se puede personalizar aún más haciendo clic en el botón *Más*. Consulte la figura 200: el bloque de dirección de la izquierda ya está seleccionado y se usará.

Se debe agregar un elemento con la forma <Dirección, renglón 2>. Para hacerlo:

1) Vaya a **Más > Editar**.
2) Arrastre el elemento <Dirección, renglón 2> en la lista de *Elementos de la dirección* para colocarlo a la derecha de <Dirección, renglón 1>. (O pulsar en la flecha de añadir ➡).
3) Si no hay espacio entre estos dos elementos, haga clic en la flecha derecha en el lado derecho del diálogo para crear el espacio.
4) Haga clic en *Aceptar*.
5) Seleccione *Bloque de dirección*: haga clic en *Aceptar*.

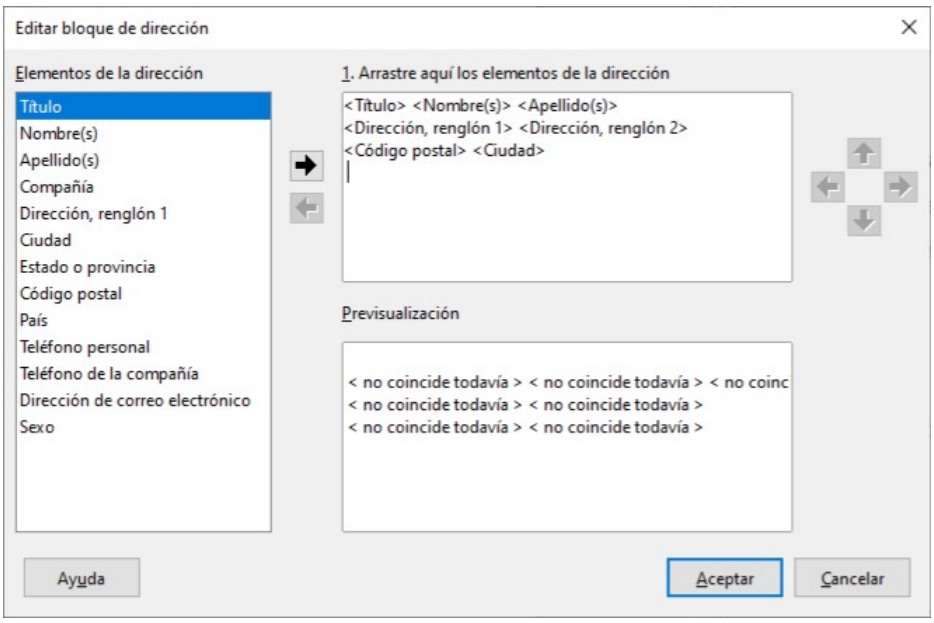


Figura 201

El elemento <Título> debe moverse una línea hacia abajo colocándolo antes de <Nombre(s)> <Apellido(s)>. Asegúrese de poner un espacio entre <Título> y <Nombre(s)>. Use las cuatro flechas a la derecha para mover primero <Apellido(s)> y luego <Nombre(s)> (figura 201).

El punto 3 de la figura 200 sirve para vincular los campos con nombre en el *Bloque de direcciones* a los campos correctos en la base de datos. El asistente inicialmente reconoce solo aquellos campos de la base de datos que tienen exactamente los mismos nombres que los que usa el asistente. En este ejemplo, ninguno de los campos coincide, por lo que todos deberán seleccionarse de las listas desplegables en este paso.

- Para <Título>, seleccione *Saludation*.
- Para <Nombre(s)> seleccione *FirstName*.
- Para <Apellido(s)> seleccione *LastName*.
- Para <Dirección, renglón 1> seleccione *Street*.
- Para <Dirección, renglón 2> seleccione *No*.
- Para <Ciudad> seleccione *Town*.
- Para <Código postal> seleccione *Postal Code*.

Aquí los elementos de dirección están asociados con los elementos correspondientes de la consulta *Readeradresses* de la base de datos transferida con éxito por el *Asistente para combinar correspondencia*. Nuevamente, el primer registro de la consulta se usa para la vista previa (ver figura 202).

La configuración de la base de datos termina esencialmente con el paso 4. Aquí solo es cuestión de elegir de qué campo se debe tomar el género del destinatario. Este campo ya ha sido nombrado, por lo que solo se necesita especificar el contenido del campo para un destinatario femenino.

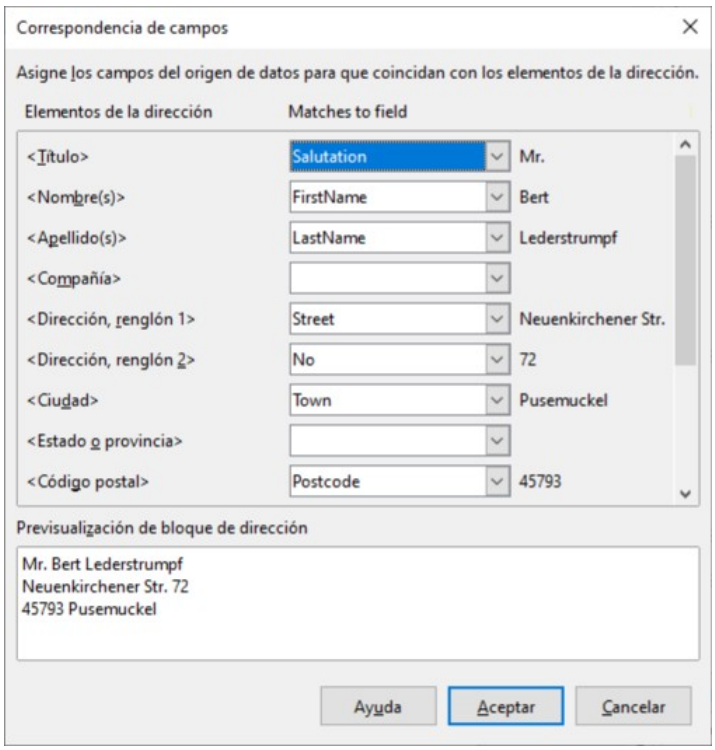


Figura 202

Nota

Debido a que el asistente tiene un error en este punto, el saludo personal se crea utilizando *Datos a texto* como se describe anteriormente. Específicamente, el campo *Saludo*, proporcionará el título apropiado para cada persona. El resto del saludo se crea al escribirlo en el documento combinado.

1) Para finalizar esta página, desmarque *Insertar saludo personalizado*.
2) No hacer cambios en el saludo general. Será reemplazado más tarde, pero es necesario para identificar dónde debe estar el saludo en la carta.

Haga clic en *Siguiente*. En el Paso 5, puede ajustar la posición del bloque de direcciones y el del saludo. (ver figura 203) Luego haga clic en *Finalizar*.

Ahora vamos a finalizar el diseño del documento de combinación de correspondencia. Este ya contiene los campos del Bloque de direcciones donde los ha colocado.

Ahora use **Datos en Texto > Insertar datos como campos** para reemplazar el saludo.

1) Reemplace `A quien corresponda` por `Estimado/a.`
2) Arrastre y suelte <Salutation> a un espacio después de Estimado/a.
3) Arrastre y suelte <FirstName> a un espacio después de <Salutation> para ingresar el nombre.
4) Arrastre y suelte <LastName> a un espacio después de <FirstName> para ingresar el apellido luego del nombre. El resultado puede verse en la figura 204.

Seleccione el registro que le interese en la ventana *Origen de datos* que se mostró al principio. Luego haga clic en el botón *Datos en campo* para ver los datos ingresados en los campos.

Ahora tiene un documento de Writer en el que puede escribir el contenido de la carta. Para fusionar los campos e imprimir las cartas, elija **Archivo > Imprimir** en la barra de menú.

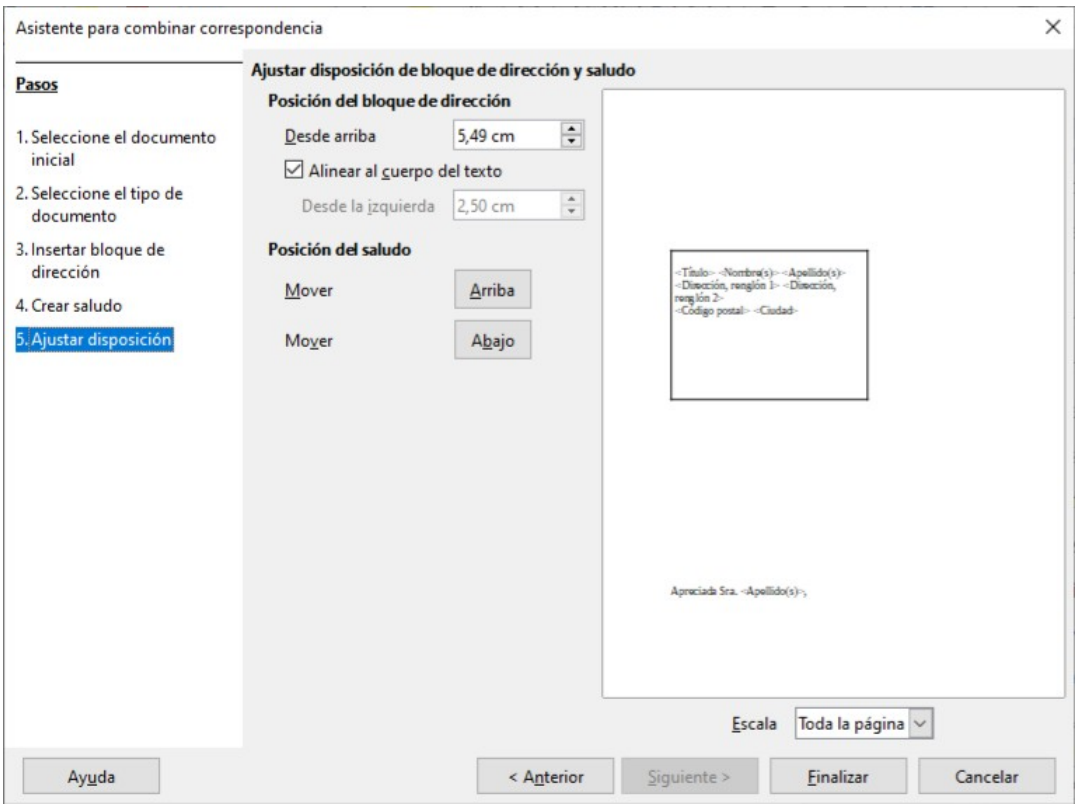


Figura 203

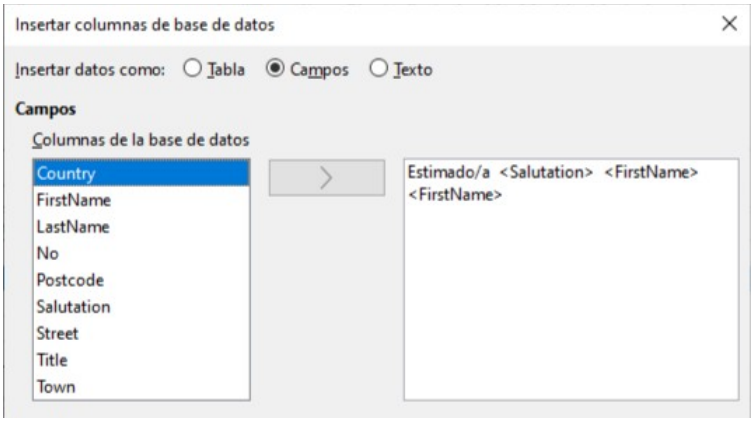


Figura 204

Con el diálogo *Combinar correspondencia* puede seleccionar registros opcionalmente para incluir o excluir y elegir imprimir las cartas o guardarlas en un archivo. Para obtener más detalles, consulte el capítulo «Combinar correspondencia» en la *Guía de Writer*.

Imprimir etiquetas.

Archivo > Nuevo > Etiquetas inicia el *Asistente de etiquetas*. Abre un diálogo, que incluye todas las preguntas de formato y contenido para las etiquetas, antes de que se produzcan las mismas. La configuración de este diálogo se guarda en la configuración personal del usuario.

La configuración básica del contenido se encuentra en la pestaña *Etiquetas* (figura 205). Si para el texto de la etiqueta marca la casilla *Dirección*, todas las etiquetas tendrán el mismo contenido, tomado de la configuración de LibreOffice para el usuario del programa.

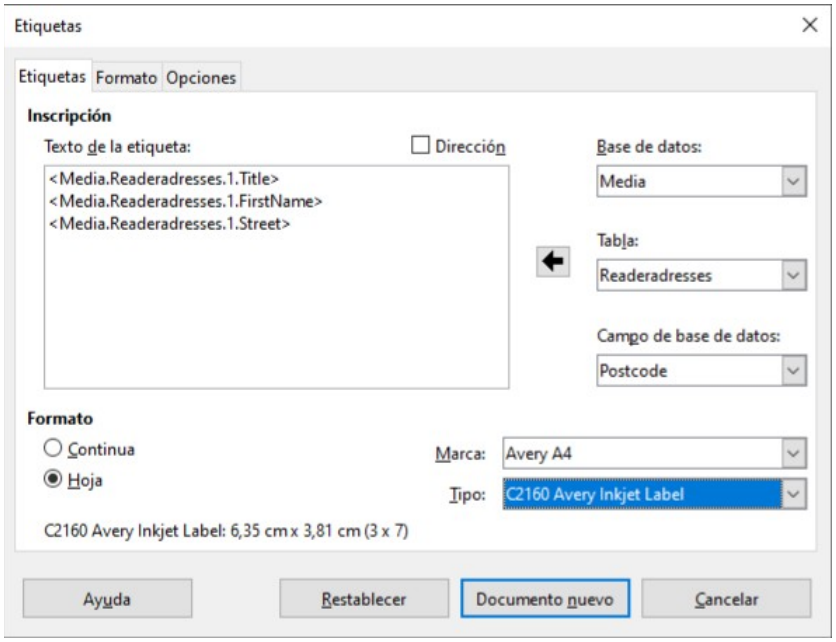


Figura 205

Como ejemplo, utilizaremos nuevamente la consulta *Readeradresses*. Aunque el siguiente campo desplegable de selección está encabezado por la etiqueta «Tablas:», se listan aquí tanto las tablas como las consultas, al igual que en el navegador de origen de datos.

Use los botones de flecha para insertar campos de bases de datos individuales en el editor. El nombre del campo de la base de datos Apellido se establece aquí en <Addresses.MailMergeQuery.1.Surname>. La secuencia es, por lo tanto, <BasedeDatos.Tabla.1.Campo de la base de datos>.

Se puede trabajar con el teclado en el editor. Así, por ejemplo, puede insertar un salto de línea al principio, de modo que las etiquetas no se impriman directamente en el borde superior sino que el contenido se pueda imprimir completamente y claramente visible.

El formato se puede seleccionar en la pestaña *Etiquetas*. Aquí se incorporan muchos modelos de etiqueta para que la mayoría de las otras configuraciones en la pestaña *Formato* no sean necesarias.

Use la pestaña *Formato* (figura 206) para establecer el tamaño de la etiqueta con precisión. La configuración solo es significativa cuando no se conoce el modelo y el tipo de las etiquetas. Tenga en cuenta que, para imprimir etiquetas de 7.00 cm de ancho, necesita un ancho de página un poco más grande que 3 * 7.00 cm = 21.00 cm. Solo entonces se imprimirán tres etiquetas en una fila en la página.

En la pestaña *Opciones* (figura 207), puede especificar si solo se generará una sola etiqueta o una página completa de etiquetas. La página se llenará con datos de registros sucesivos de la base de datos, comenzando con el primer registro. Si hay más registros de los que caben en la página, la página siguiente se completará automáticamente con el siguiente conjunto de registros.

La casilla de verificación *Sincronizar contenido* vincula todas las etiquetas para que los cambios posteriores en el diseño de cualquier etiqueta se apliquen a todas las demás. Para transferir el contenido editado, use el botón *Sincronizar*, que aparece durante la producción de etiquetas si ha seleccionado esta casilla de verificación.

Use el botón *Documento nuevo* para crear un documento que contenga los campos seleccionados.

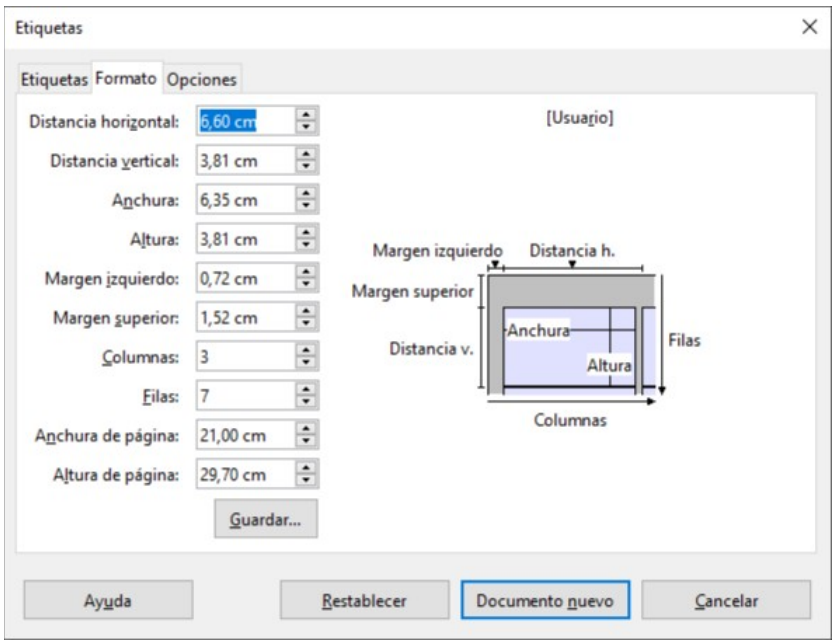


Figura 206

Figura 207

El origen de los datos a usar en la impresión de las etiquetas no se encuentra automáticamente; debe seleccionar previamente la base de datos. La consulta de la base de datos debe ser especificada por el usuario.

Cuando se seleccione la consulta y se elijan los registros correspondientes (en este caso, *Todos*), podrá comenzar la impresión. Es aconsejable, especialmente para las primeras pruebas, elegir *Salida a un archivo*, en el diálogo *Combinar correspondencia* (figura 208), que guardará las etiquetas como un documento. La opción de guardar en varios documentos no es apropiada para la impresión de etiquetas, sino para cartas a diferentes destinatarios en las que luego se puede trabajar.

Producción directa de combinación de correspondencia y documentos de etiquetas.

En lugar de usar el asistente, puede producir una combinación de correspondencia y documentos de etiquetas directamente.

Combinar correspondencia usando el ratón

Los campos de combinación de correspondencia se pueden tomar del navegador de la base de datos con el ratón.

Seleccione el encabezado de la tabla con el botón izquierdo del ratón. Mantenga presionado el botón y arrastre el cursor al documento de texto. El cursor cambia su forma a un símbolo de

inserción. El campo se inserta en el documento de texto. Puede ver el nombre de los campos usando **Ver > Nombres de campo**.

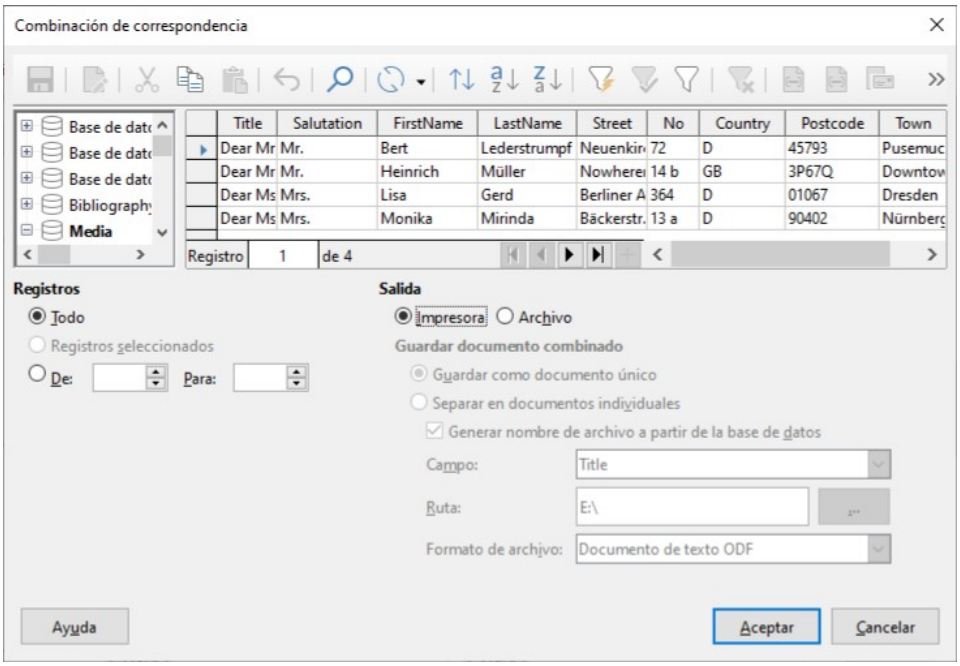


Figura 208

Crear formulario de cartas seleccionando campos

Los campos de combinación de correspondencia se pueden insertar usando **Insertar** > **Campo > Más campos > Base de datos**.

Aquí están disponibles todas las tablas y consultas en la base de datos seleccionada. Usando el botón *Insertar*, se pueden insertar los distintos campos uno tras otro directamente en el texto en la posición actual del cursor (ver figura 209).

Si se desea crear un saludo, que es habitual en cartas, se puede usar un párrafo oculto o texto oculto: **Insertar > Campo > Más campos > Funciones > Párrafo oculto**. Para ambas variantes, sea cuidadoso con que no se cumpla la condición que formula, ya que desea que el párrafo sea visible.

Para que la fórmula Estimada Mrs. <Apellido>, aparezca solo cuando la persona es mujer, una condición suficiente es:

```
[Media.Readeraddresses.Salutation]! = "Mrs."
```

Ahora el único problema que queda es que puede que no haya apellido. En estas circunstancias, debe aparecer "Estimado señor / señora", por lo que esta es la condición que debe insertar. La condición general es:

```
[Media.Readeraddresses.Salutation]! "Mrs." OR NOT
[Media.Readeraddresses.Salutation]
```

Eso excluye la posibilidad de que este párrafo aparezca cuando la persona no es mujer o no hay un apellido ingresado.

Del mismo modo, puede crear entradas para el género masculino y para las entradas que faltan para los dos tipos restantes de saludo.

Naturalmente, puede crear un saludo en el campo superior de la dirección exactamente de la misma manera, independientemente de que se especifique el género.

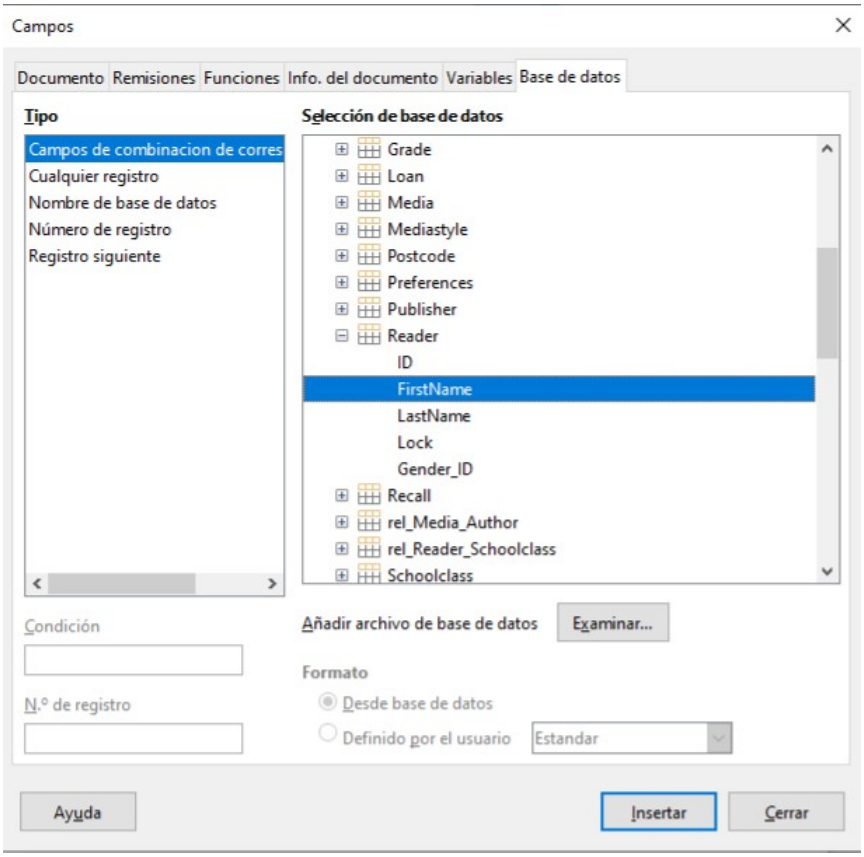


Figura 209

Se proporciona más información en la *Ayuda de LibreOffice* sobre *Párrafo oculto, Texto oculto* y *Texto condicional*.

Por supuesto, sería aún más simple si alguien que comprende las bases de datos incluyera todo el saludo en la consulta. Esto se puede hacer utilizando una subconsulta correlacionada (consulte el «Capítulo 5, Consultas», en este libro).

Particularmente interesante para las etiquetas es el tipo de campo *Registro siguiente*. Si se elige este tipo de campo al final de una etiqueta, la siguiente etiqueta se completará con los datos del siguiente registro. Las etiquetas típicas para la impresión secuencial de etiquetas se parecen a la figura 210 cuando use **Ver > Nombres de campo** para hacer visibles las designaciones de campo correspondientes:

Media.Readeradresses.Salutation Media.Readeradresses.FirstName Media.Readeradresses.LastName
Media.Readeradresses.Street Media.Readeradresses.No
Media.Readeradresses.Postcode Media.Readeradresses.Town Registro siguiente:Media.Readeradresses

Figura 210: Selección de campos para etiquetas con contenido secuencial

En la última etiqueta en la página, debe tener en cuenta el hecho de que el siguiente registro se llama automáticamente después de un salto de página. Aquí no debe aparecer el tipo de campo *Registro siguiente*. De lo contrario, se perderá un registro porque se produce un salto de doble registro.

Consejo

La creación de cartas de correspondencia combinadas también es posible desde un formulario de base de datos. El único requisito es que la base de datos esté registrada en LibreOffice.

Cuando se realice una combinación de correspondencia, asegúrese de elegir **Ver > Normal**. Esto permite asegurar que los elementos están correctamente posicionados en la página. Si se imprime un formulario, aparece la consulta de combinar correspondencia habitual (figura 208).

Este tipo de combinación tiene la ventaja de que no necesita ningún archivo que no sea el archivo *.odb para imprimir.

Formularios externos

Si las propiedades de formulario simples disponibles en LibreOffice se van a utilizar en otros componentes como Writer y Calc, solo necesita mostrar la barra de herramientas *Diseño de formulario*, usando **Ver > Barras de herramientas > Diseño de formularios** y luego abrir el *Navegador de formularios*. Puede crear un formulario o como se describe en el «Capítulo 4, Formularios», crear un campo de formulario. La pestaña *Datos* del diálogo *Propiedades del formulario* se ve un poco diferente de la que ve cuando los formularios se crean directamente en un archivo de base de datos ODB.

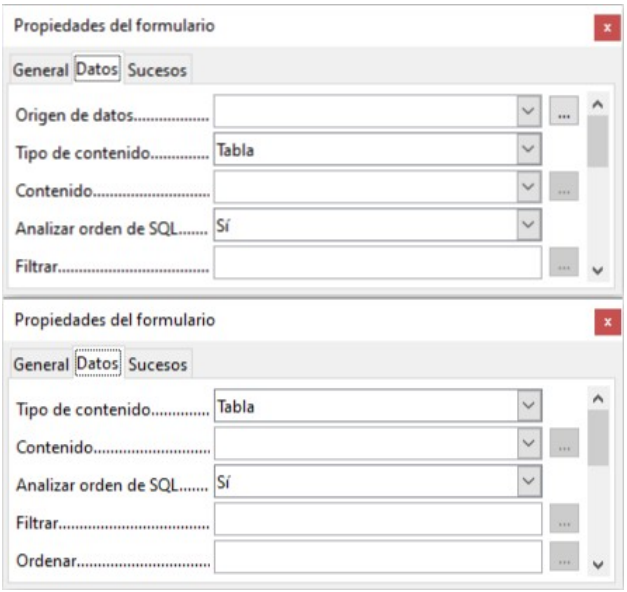


Figura 211: Arriba, formulario con origen de datos externos; abajo, formulario con origen de datos internos.

La fuente de datos debe seleccionarse por separado cuando se utiliza un formulario externo. Use el botón a la derecha de la lista desplegable *Origen de datos* para abrir el explorador de archivos. Se puede seleccionar cualquier archivo ODB. Además, el campo para el origen de datos contiene un enlace, que comienza con `file:///`.

Si, en cambio, busca en el contenido desde la lista desplegable, verá las bases de datos ya añadidas en LibreOffice con sus nombres registrados.

Los formularios se crean exactamente de la misma manera que en Base.

Los formularios producidos de esta manera se muestran por defecto en modo edición cada vez que se abre el archivo y no están protegidos contra escritura como en Base. Para evitar la modificación accidental del formulario, puede usar **Archivo > Propiedades > Seguridad** para abrir el archivo de solo lectura. Incluso puede proteger el archivo de alteraciones usando una contraseña. En los sistemas de oficina, también es posible declarar todo el archivo como protegido contra escritura. Esto todavía permite la entrada de información en los campos del formulario, pero no el movimiento o la entrada de texto entre ellos.

Consejo

Los formularios también se pueden crear rápidamente mediante instrucciones de «arrastrar y soltar». Para ello, abra la base de datos, busque la tabla o consulta relevante y seleccione los encabezados de la tabla.

En Writer, seleccione los encabezados de campo apropiados con el botón izquierdo del ratón, mantenga presionadas las teclas *Shift+Ctrl* y el cursor del ratón se convertirá en un símbolo de enlace. Luego arrastre los encabezados al documento.

Puede agregar los campos a los archivos de Calc sin el uso de combinaciones de teclas. El símbolo de «copiar» aparece en el cursor del ratón.

En ambos casos, se crea un campo de entrada con su etiqueta asociada. El enlace al origen de datos se crea con la primera entrada de datos «vacía», por lo que la entrada de datos en dicho formulario puede comenzar inmediatamente después de la operación de arrastrar y soltar.

Ventajas de formularios externos

La base de datos no necesita abrirse primero para trabajar con ella. Por lo tanto, no necesita una ventana adicional en segundo plano.

En una base de datos que ya está completa, a los usuarios de las bases de datos se les puede enviar el formulario mejorados sin problemas. Pueden continuar usando la base de datos durante el desarrollo de formularios adicionales y no necesitan copiar formularios externos de una base de datos a otra.

Los formularios para una base de datos pueden variar para adaptarse al tipo de usuario. Los usuarios que no tienen los permisos para corregir datos o realizar nuevas entradas pueden recibir un conjunto de datos actual de otros usuarios y simplemente reemplazar su archivo *.odb para tener una vista actualizada. Esto podría ser útil, por ejemplo, en una base de datos para una organización donde todos los miembros del comité tienen la base de datos, pero solo una persona puede editar los datos; los demás aún pueden ver las direcciones de sus respectivos departamentos.

Desventajas de formularios externos

Los usuarios siempre deben instalar formularios y Base con la misma estructura de directorios. Esa es la única forma en que el acceso a la base de datos puede estar libre de errores. Como los enlaces se almacenan en relación con el formulario, es suficiente almacenar la base de datos y sus formularios en un directorio común.

Solo los formularios pueden ser creados externamente, no las consultas o los informes. Una simple mirada a una consulta, por lo tanto, debe pasar por un formulario. Un informe, por otro lado, requiere la apertura de la base de datos. Alternativamente, podría ser posible crear un informe, al menos parcialmente, utilizando la combinación de correspondencia.

Uso de bases de datos en Calc

Los datos se pueden usar en Calc con fines de cálculo. Para este propósito, primero es necesario hacer que los datos sean accesibles en una hoja de cálculo de Calc.

Introducir datos en Calc

Hay varias formas de ingresar datos en Calc.

Seleccione una tabla con el botón izquierdo del ratón y arrástrela a una hoja de cálculo Calc. El cursor establece la esquina superior izquierda de la tabla. La tabla se crea completa con los

nombres de campo. El navegador de origen de datos en este caso no ofrece las opciones de *Datos a texto* o *Datos a campos*.

Los datos arrastrados a Calc de esta manera muestran las siguientes propiedades:
No solo se importan los datos, sino que también se leen las propiedades del campo y actúan durante la importación. Los campos como los números de casa, que se declararon como campos de texto, se formatean como texto después de la inserción en Calc.

La importación se convierte en un intervalo de base de datos de Calc, que por defecto se le asigna el nombre Importar1. Posteriormente se puede acceder a los datos utilizando este intervalo. **Datos > Actualizar intervalo** permite, como dice su nombre, que el intervalo se actualice con los nuevos datos de la base cuando sea necesario.

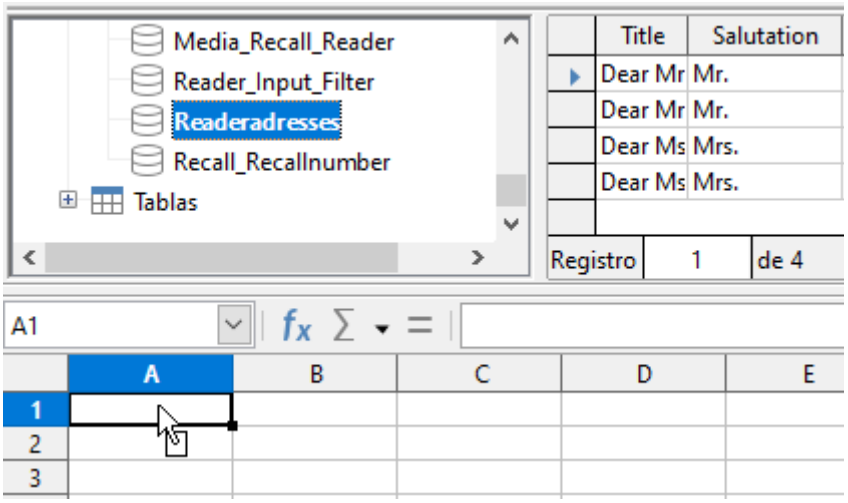


Figura 212: Insertando la consulta desde el navegador de orígenes de datos.

Los datos importados no tienen formato, a menos que los campos de la base de datos así lo requieran.

También puede usar el menú contextual de una tabla para hacer una «copia» de los datos (figura 213). En este caso, sin embargo, no hay importación sino simplemente una copia. Las propiedades de los campos de datos no se leen con ellos, sino que son determinados por Calc. Además, los nombres de campo serán los encabezados de tabla.

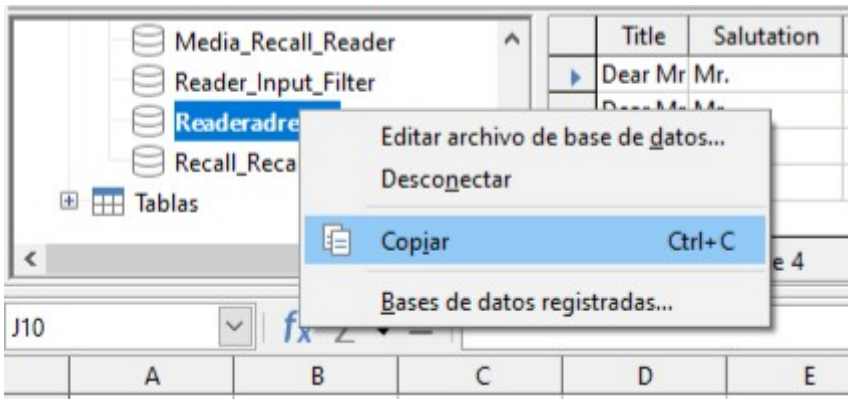


Figura 213

Verá la diferencia, especialmente en los campos de la base de datos con formato de texto. En la importación, Calc los convierte en campos de texto, que están alineados a la izquierda como cualquier otro texto. Estos números ya no se pueden usar en los cálculos. Si los exporta nuevamente, los datos permanecerán como estaban.

	A	B	C	D
1	IMPORTAR		COPIAR	
2	Street	No	Street	No
3	Neuenkirchener Str.	72	Neuenkirchener Str.	72
4	Nowhereroad	14 b	Nowhereroad	14 b
5	Berliner Allee	364	Berliner Allee	364
6	Bäckerstr.	13 a	Bäckerstr.	13 a
7				

Figura 214

Consejo

Importar datos a Calc sobrescribe los contenidos anteriores y también cualquier formato anterior. Si los datos se van a exportar consistentemente a la misma tabla, debe usar una hoja separada para la importación de datos. Luego, los datos se leen en la otra hoja utilizando el término `tablename.fieldname`. Los campos en esta hoja pueden formatearse adecuadamente sin riesgo de sobrescribir el formato.

Consejo

Si una tabla o consulta se arrastra a una hoja de cálculo de Calc, se inserta todo el contenido. Si se abre la tabla o consulta y se seleccionan uno o más registros, solo estos registros junto con los nombres de campo se copian cuando los arrastra.

Exportar datos de Calc a una base de datos

Seleccione los datos en la hoja de cálculo Calc. Mantenga presionado el botón izquierdo del ratón y arrastre los datos que desea convertir en una base de datos al área de tabla del navegador de la base de datos. El cursor cambia su apariencia, mostrando que se puede insertar algo.

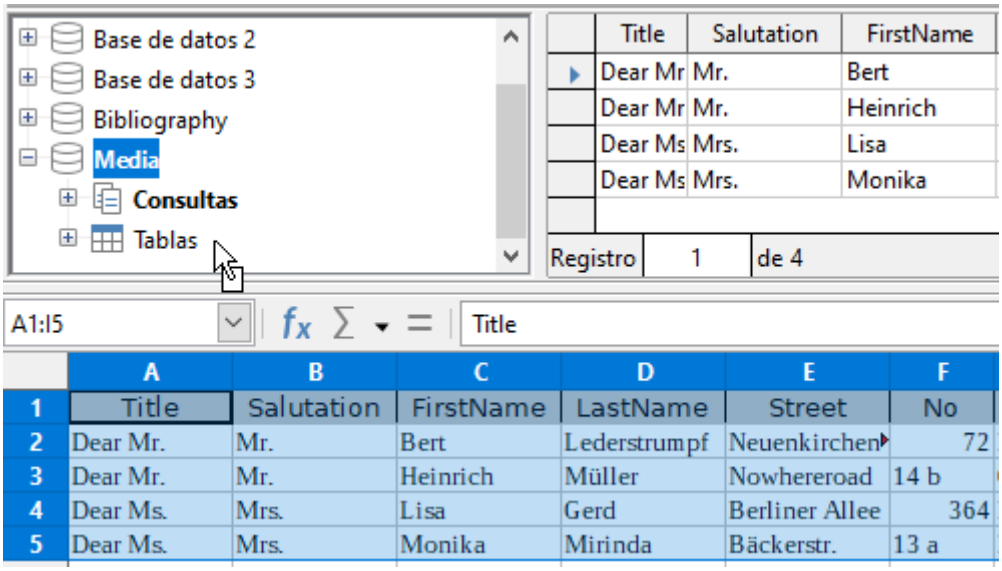


Figura 215

Se abre la primera ventana del asistente de importación. Los pasos adicionales con el asistente se describen en el *Capítulo 3, Tablas*, en la sección *Importar datos de otras fuentes*.

Convertir datos de una base de datos a otra

En el explorador del navegador de origen de datos, las tablas se pueden copiar de una base de datos a otra seleccionando la tabla de origen con el botón izquierdo del ratón, manteniendo presionado el botón y arrastrándola a la base de datos de destino en el contenedor de la tabla. Esto hace que se muestre el diálogo para copiar tablas.

De esta manera, por ejemplo, las bases de datos de solo lectura (fuentes de datos como las libretas de direcciones de un programa de correo electrónico o una tabla de hoja de cálculo) se pueden usar como cimiento para una base de datos en la que los datos se vuelven editables.

Además, los datos se pueden copiar directamente al cambiar a otro programa de base de datos (por ejemplo, cambiar de PostgreSQL a MySQL).

Si desea que la nueva base de datos tenga relaciones diferentes a las que tenía en la original, puede hacer esto mediante las consultas apropiadas. Aquellos que no son lo suficientemente expertos pueden usar Calc. Simplemente arrastre los datos a una hoja de cálculo y prepárelos para importarlos a la base de datos de destino utilizando las características que proporciona Calc.

Consejo

Para la importación más limpia posible en una nueva base de datos, las tablas deben prepararse con anticipación. Esto permite que los problemas de formateo y los que surgen en la creación de claves primarias se reconozcan con suficiente antelación.

Importar registros a una tabla usando el portapapeles

Si los registros están disponibles en forma de tabla, se pueden insertar en Base utilizando el portapapeles y el asistente.

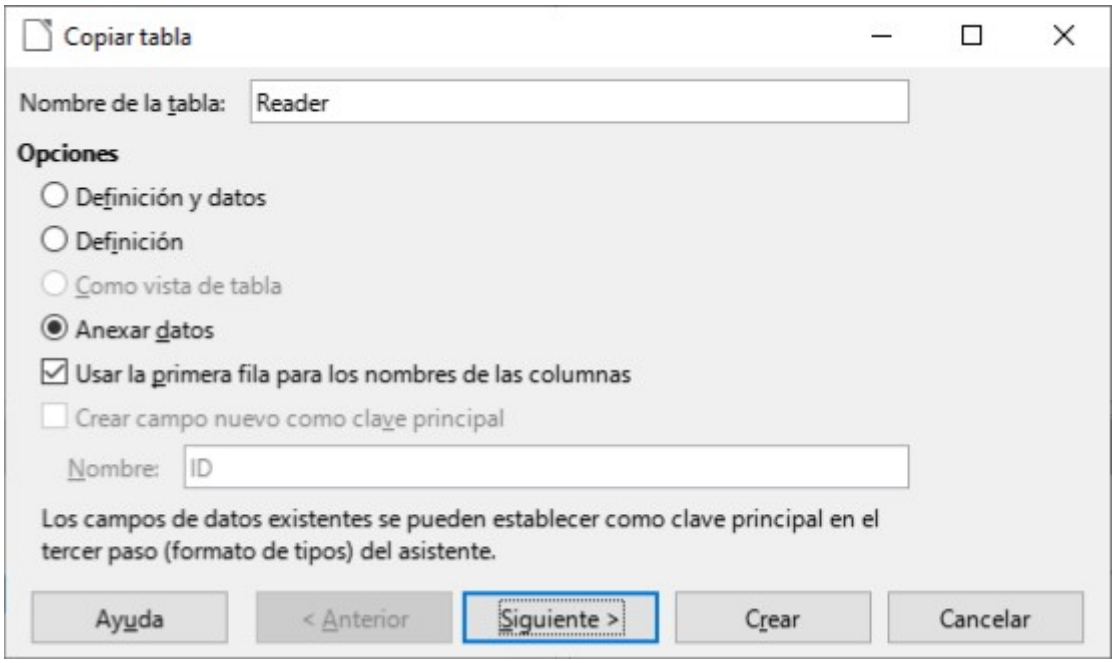


Figura 216

En Base, un clic derecho en la tabla de destino comienza la importación. En el menú contextual *Copiar tabla* (ver Figura 216) están los mandatos *Definición* y *Definición y datos*. Si elige *Crear*, el *Asistente de importación* ya habrá seleccionado la tabla y los datos a anexar. Pegado especial solo proporciona una consulta para un filtro de importación. Las opciones disponibles son HTML y RTF.

Si, en cambio, hace clic con el botón derecho en el contenedor de la tabla, el *Asistente de importación* solo le ofrece la opción de crear una nueva tabla.

Importar registros PDF

Si desea importar datos de varias fuentes externas, es mejor elegir un formato que evite que su formulario se modifique durante la entrada de datos. Con Writer, puede crear formularios en formato PDF, ponerlos en línea y que le devuelvan los formularios completos, por ejemplo, como archivos adjuntos de correo electrónico. Lo único que falta es que la introducción de los datos en Base sea lo más sencilla posible. El ejemplo ilustra esta forma de importación.

Crear un formulario PDF

Se crea un formulario PDF como un formulario externo sin enlace a la base de datos.

Usando **Ver > Barras de herramientas > Controles de formulario**, se muestran los elementos necesarios para el formulario y se pueden insertar según sea necesario.

Lamentablemente, el formato PDF no distingue entre campos numéricos, campos de fecha y campos de texto. Para el ejemplo proporcionado aquí, es suficiente usar campos de texto para todas las entradas. Otros formatos de campo dentro del formulario Writer se perderán inevitablemente durante la exportación de PDF.

Básicamente, los formularios PDF pueden tener los siguientes campos:

- Botones
- Cuadros de texto
- Casillas de verificación
- Cuadros combinados
- Listados

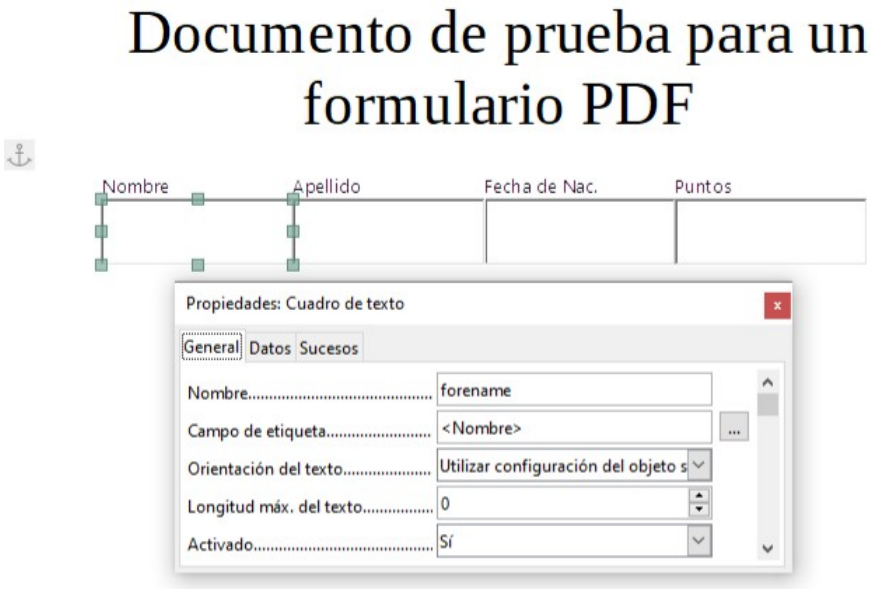


Figura 217

El formulario de la figura 217 contiene un total de 4 campos de texto. En **Propiedades: Cuadro de texto > General > Nombre**, siempre debe elegir el nombre del campo utilizado en la tabla de la base de datos cuando utilice este método de importación, para evitar problemas con los nombres y el contenido del campo.

Los mensajes de ayuda se muestran cuando se leen los registros, pero no aparecen en todos los visores de PDF.

Para asegurarse de que el formulario realmente contenga los registros, debe guardarse en el visor de PDF después de la entrada de datos, utilizando generalmente la opción de menú **Archivo > Guardar como**. El mandato real para hacerlo puede variar entre los usuarios. Sin este procedimiento, el visor mostrará los registros después de que se haya abierto el formulario en su propia computadora, pero en realidad los lee del archivo de almacenamiento temporal del visor y no directamente del archivo PDF. Si el formulario se transfiere a otra computadora, estará vacío.

Leer los registros del formulario PDF

El formulario para la base de datos Base es muy simple en apariencia. Está vinculado a la tabla y muestra los registros que se acaban de leer. Las entradas más recientes se muestran en el control de la tabla anterior.

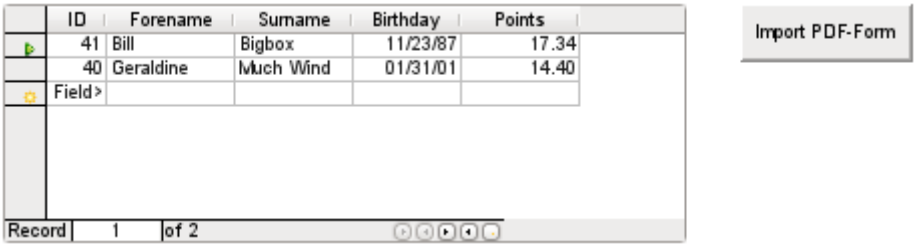
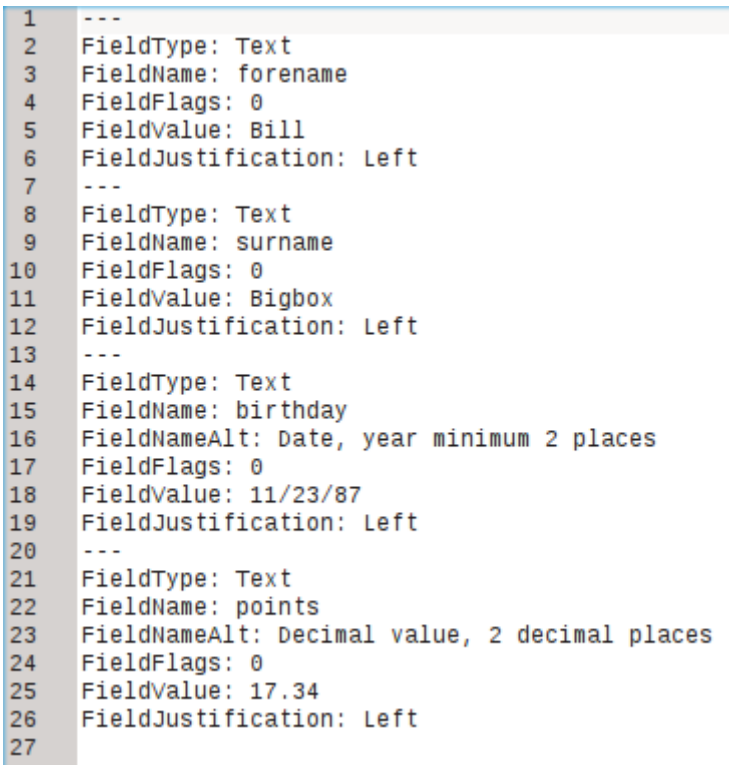


Figura 218

La macro para leer los registros se ingresa en **Propiedades: Botón > Eventos > Ejecutar acción**.

Para leer los registros, usamos el programa de código abierto `pdftk`. El programa está disponible gratuitamente para Windows y Linux. Las distribuciones de Linux lo tienen principalmente como un paquete en sus repositorios. Los usuarios de Windows lo encontrarán en https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

Los registros leídos usando `pdftk` se escriben en un archivo de texto, que se ve como en la Figura 219.



```
---
FieldType: Text
FieldName: forename
FieldFlags: 0
FieldValue: Bill
FieldJustification: Left
---
FieldType: Text
FieldName: surname
FieldFlags: 0
FieldValue: Bigbox
FieldJustification: Left
---
FieldType: Text
FieldName: birthday
FieldNameAlt: Date, year minimum 2 places
FieldFlags: 0
FieldValue: 11/23/87
FieldJustification: Left
---
FieldType: Text
FieldName: points
FieldNameAlt: Decimal value, 2 decimal places
FieldFlags: 0
FieldValue: 17.34
FieldJustification: Left

```

Figura 219

Cada campo está representado por cinco a seis líneas en el archivo. Para la macro, las líneas importantes son `FieldName` (debe ser el mismo que el campo de nombre en la tabla de destino), *FieldValue* (contenido del campo después de guardar el archivo PDF) y `FieldJustification` (última línea de la entrada).

Todo el proceso de importación está controlado por macros. El formulario PDF debe estar en la misma ruta que la base de datos. Los registros se leen en el archivo de texto y luego se leen en la base de datos desde este archivo de texto. Esto se hace así para todos los archivos PDF en la carpeta. Por lo tanto, los registros antiguos deben eliminarse de la carpeta en la medida de lo posible, porque la función no verifica la duplicación.

```
SUB PDF_Form_Import(oEvent AS OBJECT)
  DIM inNumber AS INTEGER
  DIM stRow AS STRING
  DIM i AS INTEGER
  DIM k AS INTEGER
  DIM oDatasource AS OBJECT
  DIM oConnection AS OBJECT
  DIM oSQL_Command AS OBJECT
  DIM oResult AS OBJECT
  DIM stSql AS STRING
  DIM oDB AS OBJECT
  DIM oFileAccess AS OBJECT
  DIM inFields AS INTEGER
  DIM stFieldName AS STRING
  DIM stFieldValue AS STRING
  DIM stFieldType AS STRING
  DIM stDir AS STRING
```

```
DIM stDir2 AS STRING
DIM stPDFForm AS STRING
DIM stFile AS STRING
DIM stTable AS STRING
DIM inNull AS INTEGER
DIM aFiles()
DIM aNull()
DIM stCommand AS STRING
DIM stParameter AS STRING
DIM oShell AS OBJECT
```

Después de declarar las variables, se proporciona el número de campos en el formulario PDF. El recuento comienza en 0, por lo que un valor de 3 en realidad significa un total de cuatro campos. Con este recuento, se puede determinar si se han leído todos los datos de un registro, de modo que estén listos para transferirse a la tabla.

```
inFields = 3
stTable = "Name"
oDatasource = ThisComponent.Parent.CurrentController
If NOT (oDatasource.isConnected()) THEN
  oDatasource.connect()
END IF
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
```

La conexión a la base de datos está hecha. Se lee la ruta al archivo de la base de datos en el sistema de archivos. Usando esta ruta, el contenido de la carpeta se lee en la matriz `aFiles`. Un bucle verifica cada nombre de archivo en la matriz para ver si termina en `.pdf`. Las mayúsculas y minúsculas no se distinguen, ya que los resultados de la búsqueda se convierten en minúsculas usando `Lcase`.

```
oDB = ThisComponent.Parent
stDir = Left(oDB.Location,Len(oDB.Location)-Len(oDB.Title))
oFileAccess = createUnoService("com.sun.star.ucb.SimpleFileAccess")
aFiles = oFileAccess.getFolderContents(stDir, False)
FOR k = 0 TO uBound(aFiles())
  IF LCase(Right(aFiles(k),4)) = ".pdf" THEN
    stDir2 = ConvertFromUrl(stDir)
    stPDFForm = ConvertFromUrl(aFiles(k))
```

Para determinar la orden para leer los datos, es necesario comprender las convenciones de dirección de archivo del sistema operativo. Por ello, la URL original que comienza con `file://` tiene que ser adaptada al sistema usado. La orden para iniciar el programa `pdftk` depende del sistema operativo. Puede llevar el sufijo `.exe` o quizás una ruta completa al programa como `C:\Archivos de programa (x86)\pdftk\pdftk.exe` o el sufijo podría no ser necesario. `GetGuiType` se usa para determinar el tipo de sistema en uso: 1 significa Windows, 3 para macOS y 4 para Linux. Los siguientes pasos solo distinguen entre Windows y el resto.

Después la función `Shell()` se usa para pasar la orden de inicio apropiada para `pdftk` a la consola. El argumento `True` asegura que LibreOffice esperará hasta que finalice el proceso de shell.

```
    IF GetGuiType = 1 THEN '()
```

```
    stCommand = "pdftk.exe"
  ELSE
    stCommand = "pdftk"
  END IF
  stParameter = stPDFForm & " dump_data_fields_utf8 output "
    & stDir2 & "PDF_Form_Data.txt"
  Shell(stCommand,0,stParameter,True)
  stFile = stDir & "PDF_Form_Data.txt"
  i = -1
  inNumber = FreeFile
```

La función `FreeFile` determina cuál es el siguiente canal de datos libre disponible en el sistema operativo. Este canal se lee como un número entero y se utiliza para conectarse directamente al archivo de datos PDF que se acaba de crear. La instrucción `INPUT` se usa para leer el archivo. Eso tiene lugar fuera de LibreOffice. Los registros externos se leen en LibreOffice.

```
  OPEN stFile FOR INPUT AS inNumber
    DO WHILE NOT Eof(inNumber)
      LINE INPUT #inNumber, stRow
```

El archivo de datos PDF ahora se lee línea por línea. Cada vez que aparece el término `FieldName`, el contenido restante de la línea se toma como el nombre del campo en el formulario PDF y también, debido a la forma en que se definió el formulario, el nombre del campo de la base de datos en el que se deben escribir los datos.

Todos los nombres de campo se combinan directamente para su uso en las órdenes SQL posteriores. Lo que esto significa en la práctica es que los nombres de campo están encerrados entre comillas dobles y separados por comas.

Además, para cada nombre de campo, una consulta determina el tipo de campo en la tabla. Los valores de fecha y decimales deben transferirse de forma diferente al texto.

```
        IF instr(stRow, "FieldName: ") THEN
          IF stFieldName = "" THEN
           stFieldName = """"+mid(stRow,12)+""""
          ELSE
           stFieldName = stFieldName & ","""" + mid(stRow,12)+""""
          END IF
          stSql = "SELECT TYPE_NAME FROM
INFORMATION_SCHEMA.SYSTEM_COLUMNS
           WHERE TABLE_NAME = '" + stTable + "' AND
           COLUMN_NAME = '" + mid(stRow,12) + "'"
          oResult = oSQL_Command.executeQuery(stSql)
          WHILE oResult.next
           stFieldType = oResult.getString(1)
          WEND
        END IF
```

Como con los nombres de campos, también se hace para los campos de valores. Sin embargo, estos no deben estar entre comillas dobles, sino que deben prepararse de acuerdo con los requisitos del código SQL. Esto significa que el texto debe estar entre comillas simples, las fechas convertidas para cumplir con las convenciones de SQL, etc. Esto se realiza mediante la función `SQL_Value` externa adicional.

```
            IF instr(stRow, "FieldValue: ") THEN
              IF stFieldValue = "" THEN
               stFieldValue = SQL_Value(mid(stRow,13), stFieldType)
              ELSE
               stFieldValue = stFieldValue & "," &
                  SQL_Value(mid(stRow,13), stFieldType)
              END IF
            END IF
```

Si se encuentra el término `FieldJustification`, esto marca el final del bloque combinado de nombre de campo y propiedades. Por lo tanto, el contador `i`, que posteriormente se comparará con el contador de campo declarado previamente en `InFields`, se incrementa en 1.

Cuando `i` e `inFields` se vuelven iguales, la orden SQL se puede juntar. Sin embargo, debe asegurarse de que no se creen registros vacíos a partir de formularios vacíos. Por lo tanto, hay una verificación previa para todos los valores de campo que son NULL. En tales casos, la orden SQL se inicia inmediatamente. De lo contrario, el registro se incluye para su inserción en la tabla Nombre. Después de esto, las variables se restauran a sus valores

```
            IF instr(stRow, "FieldJustification:") THEN
              i = i + 1
            END IF
            IF i = inFields THEN
              aNull = Split(stFieldValue,",")
              FOR n = 0 TO Ubound(aNull())
               IF aNull(n) = "NULL" THEN inNull = inNull + 1
              NEXT
              IF inNull < inFields THEN
               stSql ="INSERT INTO """ + stTable + """(" + stFieldName +
")"
               stSql = stSql + "VALUES (" + stFieldValue + ")"
               oSQL_Command.executeUpdate(stSql)
              END IF
              stFieldName = ""
              stFieldValue = ""
              stFieldType = ""
              i = -1
              inNull = 0
            END IF
          LOOP
        CLOSE inNumber
```

Al final del procedimiento, queda un archivo `PDF_Form_Data.txt`. Se elimina. Luego, el formulario se vuelve a cargar para que se puedan mostrar los registros que se leyeron.

```
        Kill(stFile)
      END IF
    NEXT
    oEvent.Source.Model.Parent.reload()
END SUB
```

Si el texto contiene un "'", se verá como un marcador de fin de texto durante la inserción por SQL. El código SQL para la orden de inserción falla si sigue más texto sin estar entre comillas simples. Para evitar esto, cada comilla simple dentro del texto debe estar enmascarada por otra comilla simple. Este es el trabajo de la función `String_to_SQL`.

```
FUNCTION String_to_SQL(st AS STRING) AS STRING
  IF InStr(st,"'") THEN
    st = Join(Split(st,"'"),"''")
  END IF
  String_to_SQL = st
END FUNCTION
```

Las fechas en el archivo PDF se leen como texto. No se pueden verificar por adelantado para la entrada correcta.

Cuando las fechas se escriben en inglés, el día, mes y año están separados por puntos o, más a menudo, guiones. El día y el mes pueden tener un solo dígito o dos. El año puede tener dos dígitos o cuatro.

En el código SQL, las fechas deben comenzar con un año de cuatro dígitos y escribirse AAAA-MM-DD. Por lo tanto, las fechas ingresadas deben pasar por un proceso de conversión.

La fecha ingresada se divide en partes de día, mes y año. El día y el mes reciben un cero inicial y luego se truncan a la derecha a dos dígitos. Esto asegura una cifra de dos dígitos en todos los casos.

Si la parte del año ya tiene cuatro dígitos (mayor que 1000), el valor no cambia. De lo contrario, si el año es mayor a 30, se supone que la fecha pertenece al siglo pasado y necesita agregar 1900. Todas las demás fechas se asignan al siglo actual.

```
FUNCTION Date_to_SQLDate(st AS STRING) AS STRING
  DIM stDay AS STRING
  DIM stMonth AS STRING
  DIM stDate AS STRING
  DIM inYear AS INTEGER
  stDay = Right("0" & Day(CDate(st)), 2)
  stMonth = Right("0" & Month(CDate(st)), 2)
  inYear = Year(CDate(st))
  IF inYear = 0 THEN
    inYear = Year(Now())
  END IF
  IF inYear > 1000 THEN
  ELSEIF inYear > 30 THEN
    inYear = 1900 + inYear
  ELSE
    inYear = 2000 + inYear
  END IF
  stDate = inYear & "-" & stMonth & "-" & stDay
  Date_to_SQLDate = stDate
END FUNCTION
```

La función `SQL_Value` combina esta función con la configuración NULL que se muestra a continuación y, por lo tanto, proporciona valores formateados correctamente para la entrada en la base de datos a su función de llamada.

Los campos vacíos producen un valor NULL. El campo correspondiente en la tabla también estará vacío.

```
FUNCTION SQL_Value(st AS STRING, stType AS STRING) AS STRING
  DIM stValue AS STRING
  IF st = "" THEN
    SQL_Value = "NULL"
```

Si este es un campo de fecha y el contenido debe ser reconocible como una fecha, su contenido debe convertirse al formato de fecha SQL. Si no es reconocible como una fecha, el campo debe permanecer vacío.

```
  ELSEIF stType = "DATE" THEN
    IF isDate(st) THEN
      SQL_Value = "'" & Date_to_SQLDate(st) & "'"
    ELSE
      SQL_Value = "NULL"
    END IF
```

Un campo decimal puede contener comas en lugar de puntos decimales, con lugares decimales a continuación. En Basic y SQL, el separador decimal siempre es un punto. Por lo tanto, los números que contienen una coma deben convertirse. El campo debe contener un número, por lo que se deben eliminar otros caracteres, como las unidades. Esto se lleva a cabo mediante la función `Val()`.

```
  ELSEIF stType = "DECIMAL" THEN
    stValue = Str(Val(Join(Split(st,","),".")))
```

El resto del contenido se trata como texto. Las comillas simples se enmascaran con una comilla simple adicional y el término completo se encierra nuevamente entre comillas simples.

```
  ELSE
    SQL_Value = "'" & String_to_SQL(st) & "'"
  END IF
END FUNCTION
```

Para más detalles sobre la construcción macro, vea el *Capítulo 9* de este libro. Este ejemplo simplemente muestra que es posible transferir datos desde formularios PDF a Base sin tener que copiar los valores campo por campo utilizando el portapapeles. La construcción del procedimiento anterior se ha mantenido deliberadamente muy general y necesitaría adaptarse a situaciones particulares.

LibreOffice

Guía de Base

Capítulo 8, Tareas en las bases de datos

Observaciones generales sobre las tareas de la base de datos

Este capítulo describe algunas soluciones para los problemas que surgen a muchos usuarios de bases de datos.

Filtrado de datos

El filtrado de datos con la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se describe en el “Capítulo 3, Tablas”. Aquí describimos una solución a un problema que muchos usuarios han planteado: cómo usar las *listas desplegables* como un filtro para buscar el contenido de los campos de las tablas, para que luego aparezcan estos campos filtrados en la sección de formulario subyacente y pueden editarse.

Consulta editable

La base para este filtrado es una consulta editable (consulte el “Capítulo 5, Consultas” y la base de datos `Media_with_Macros.odb`[8]) y una tabla adicional en la que se almacenan los datos que se filtrarán. La consulta muestra, a partir de su tabla subyacente, solo los registros que corresponden a los valores del filtro. Si no se proporciona ningún valor de filtro, la consulta muestra todos los registros.

El siguiente ejemplo comienza a partir de la tabla *Media* que incluye, entre otros, los siguientes campos: *ID* (clave principal), *Title*, y *Categoy_ID*. Los tipos de campo son INTEGER, VARCHAR e INTEGER respectivamente.

Primero necesitamos una tabla *Filter*. Esta tabla contiene una clave principal y tres campos de filtro (por supuesto, puede tener más si lo desea): *ID* (clave principal), *Filter*, *Integer y Date*. Los campos *Filter* e *Integer* deben ser del mismo tipo que *Title* y *Category_ID*; en este caso serán VARCHAR e INTEGER, respectivamente. *ID* puede ser el tipo numérico más pequeño, TINYINT, porque la tabla *Filter* nunca contendrá más de un registro.

También puede filtrar los campos que aparecen en la tabla *Media* solo como claves foráneas. En ese caso, debe proporcionar a los campos correspondientes en la tabla *Filter* el tipo apropiado para las claves foráneas, generalmente INTEGER.

La siguiente consulta es ciertamente editable (vaya al panel *Base de datos* y elija el botón *Consultas* y luego haga clic en *Crear consulta en modo SQL…*). Escriba en la ventana que se abre y ejecute pulsando la tecla *F5*:

```
SELECT * FROM "Media"
```

Se muestran todos los registros de la tabla *Media*, incluida la clave primaria.

```
SELECT * FROM "Media"
WHERE "Title" = IFNULL( ( SELECT "Filter" FROM "Filter" ), "Title" )
```

Si el campo *Filter* no es NULL, se muestran aquellos registros para los cuales *Title* coincide con el valor de *Filter*. Si el campo *Filter* es NULL, se utiliza el valor del campo *Title*. Como *Title* es igual al filtro "Title", se muestran todos los registros. Sin embargo, esta suposición no se cumple si el campo *Title* de cualquier registro está vacío (contiene NULL). Eso significa que esos registros nunca se mostrarán sin entrada de título. Por lo tanto, necesitamos mejorar la consulta:

```
SELECT
   Media.* ,
   IFNULL( "Title", '' ) AS "T"
FROM "Media"
WHERE "T" = IFNULL( ( SELECT "Filter" FROM "Filter" ), "T" )
```

8 Vaya a https://documentation.libreoffice.org/assets/Uploads/Documentation/es/Base-62/SampleDatabases62.zip para descargar el archivo que contiene las bases de datos de ejemplo.

Consejo

IFNULL (expresión, valor) requiere que la expresión tenga el mismo tipo de campo que el valor.

Si la expresión tiene el tipo de campo VARCHAR, use dos comillas simples '' como valor. Si tiene DATE como su tipo de campo, ingrese una fecha como el valor que no está contenido en el campo de la tabla a filtrar. Utilice este formato: {D 'AAAA-MM-DD'}. Si es cualquiera de los tipos de campo numéricos, use el tipo de campo NUMÉRICO para el valor. Ingrese un número que no aparece en el campo de la tabla a filtrar.

Esta variante conducirá a la meta deseada. En lugar de filtrar *Title* directamente, se filtra un campo que lleva el alias *T*. Este campo tampoco tiene contenido pero no es NULL. En las condiciones de la instrucción SQL, solo se considera el campo *T*. Por lo tanto, todos los registros se muestran incluso si *Title* es NULL.

Desafortunadamente no puede hacer esto usando la interfaz gráfica de usuario. Esta instrucción solo está disponible directamente con SQL. Para que sea editable en la interfaz gráfica de usuario, se requieren modificaciones adicionales:

```
SELECT
    "Media".* ,
    IFNULL( "Media"."Title", '' ) AS "T"
FROM "Media"
WHERE "T" = IFNULL( ( SELECT "Filter" FROM "Filter" ), "T" )
```

Si la relación de la tabla con los campos ahora está configurada, la consulta se puede editar en la interfaz gráfica de usuario.

Como prueba, puede escribir un título en "Filter"."Filter". Como "Filter"."ID" establece el valor '0', el registro se guarda y se puede comprender el filtrado. Si "Filter"."Filter" se queda vacío, la interfaz gráfica de usuario lo trata como NULL.

Una nueva prueba produce una visualización de todos los medios de la tabla. En cualquier caso, antes de crear y probar un formulario, solo se debe ingresar un registro con una clave principal en la tabla *Filter*. Tiene que ser solo un registro, ya que las subconsultas, como se muestra arriba, solo pueden transmitir un valor.

La consulta ahora se puede ampliar para filtrar dos campos:

```
SELECT
    "Media".*,
    IFNULL( "Media"."Title", '' ) AS "T" ,
    IFNULL( "Media"."Category_ID", 100 ) AS "K"
FROM "Media"
WHERE "T" = IFNULL( ( SELECT "Filter" FROM "Filter"), "T")
AND "K" = IFNULL( ( SELECT "Integer" FROM "Filter" ), "K")
```

Esto concluye la creación de la consulta editable. Ahora, para la consulta básica para las dos listas desplegables (consideraremos una lista desplegable por filtro):

Consulta básica de dos listas desplegables

```
SELECT
    DISTINCT "Title",
    "Title"
FROM "Media"
```

```
ORDER BY "Title" ASC
```

Una de las listas desplegables debe mostrar todos los valores de *Título* y luego transmitir el *Title* seleccionado al campo *Filter* en la tabla *Filter* que subyace en el formulario. Tampoco se deben mostrar valores duplicados (por ello se incluye la condición DISTINCT). Y, por supuesto, todo debe mostrarse en el orden correcto.

Luego se crea una consulta correspondiente para el campo *Category_ID*, que es escribir sus datos en el campo *Integer* en la tabla *Filter*.

Si uno de estos campos contiene una clave externa, la consulta se adapta para que la clave externa se pase a la tabla *Filter* subyacente.

El formulario consta de dos partes. El formulario 1 es el formulario basado en la tabla *Filter*. El formulario 2 es el formulario basado en la consulta. El formulario 1 no tiene barra de navegación y el ciclo se establece en *Registro actual*. Además, la propiedad *Permitir adiciones* está establecida en *No*. El primer y único registro para este formulario ya existe.

El formulario 1 contiene dos listas desplegables con las etiquetas apropiadas. La lista desplegable 1 devuelve valores para el campo *Filter* de la tabla *Filter* y está vinculado a la consulta del campo *Title*. La lista desplegable 2 devuelve valores para el campo *Integer* de la tabla *Filter* y se relaciona con la consulta para el campo *Category_ID*.

El formulario 2 contiene un campo de control de tabla en el que se pueden enumerar todos los campos de la consulta, excepto los campos *T* y *K*. El formulario aún funcionaría si estos campos estuvieran presentes; se omiten para evitar una duplicación confusa de los contenidos de los campos. Además, el formulario 2 contiene un botón vinculado a la función *Actualizar formulario*. Se puede incorporar una barra de navegación adicional para evitar el parpadeo de la pantalla cada vez que cambia el formulario, debido a que la barra de navegación está presente en un formulario y no en el otro.

Una vez finalizado el formulario, comienza la fase de prueba. Cuando cambia una lista desplegable, el botón del formulario 2 se usa para almacenar este valor y actualizar el formulario 2. Esto ahora se relaciona con el valor que proporciona la lista desplegable. El filtrado puede hacerse retrospectivamente eligiendo un campo vacío en la lista desplegable.

Búsqueda de datos

La principal diferencia entre buscar datos y filtrar datos está en la técnica de consulta. El objetivo es entregar, en respuesta a los términos de búsqueda (independientes del idioma), una lista resultante de registros que pueden contener solo parcialmente estos términos reales. En primera instancia describiremos los enfoques similares al de tabla y de formulario.

Búsqueda con LIKE

La tabla para el contenido de búsqueda puede ser la misma que ya contiene los valores de los filtros. La tabla *Filter* simplemente se expande para incluir un campo llamado s*earchtext*. Entonces, si es necesario, se puede acceder a la misma tabla y, utilizando los formularios, filtrar y buscar simultáneamente. s*earchtext* tiene el tipo de campo VARCHAR.

El formulario se construye igual que para el del filtrado de la sección anterior. En lugar de una lista desplegable, necesitamos un campo de entrada de texto para el término de búsqueda y también, quizás, un campo de etiqueta con el título *Searchitem*. El campo para el término de búsqueda puede estar solo en el formulario o junto con los campos para el filtrado, si se desean ambas funciones.

Como ya se indicó antes, la diferencia entre filtrar y buscar radica en la técnica de consulta. Mientras que el filtrado utiliza un término que ya aparece en la tabla subyacente, la búsqueda utiliza entradas arbitrarias. (Después de todo, la lista desplegable se construye a partir del contenido de la tabla).

```
SELECT * FROM "Media"
WHERE "Title" = ( SELECT "searchtext" FROM "Filter" )
```

Esta consulta normalmente conduce a una lista de resultados vacía por estos motivos:

- Al ingresar términos de búsqueda, las personas rara vez saben completamente y con precisión cuál es el título que buscan. Por lo tanto, no se mostrará el título correcto con la instrucción SQL anterior. Para encontrar el libro *"Autoestopista a través de la galaxia"* debería ser suficiente poner *"Autoestopista"* en el campo de búsqueda o incluso simplemente *"Autoe"*.
- Si el campo *searchtext* está vacío, solo se muestran los registros en los que no hay título. Esto solo sucede si el artículo no tiene título o si alguien no ingresó su título.

El último motivo que conduce a resultados vacíos se puede eliminar si la condición de filtrado es:

```
SELECT * FROM "Media"
WHERE "Title" = IFNULL( ( SELECT "searchtext" FROM "Filter" ), "Title" )
```

Con este refinamiento del filtrado (¿qué sucede si el título es NULL?) obtenemos un resultado más acorde con las expectativas. Pero el primero motivo de resultados vacíos aún no se subsana. La búsqueda debería funcionar bien cuando solo hay conocimiento fragmentario disponible. Por lo tanto, la técnica de consulta debe usar la condición LIKE:

```
SELECT * FROM "Media"
WHERE "Title" LIKE ( SELECT '%' || "searchtext" || '%' FROM "Filter" )
```

o mejor aún:

```
SELECT * FROM "Media"
WHERE "Title" LIKE IFNULL( ( SELECT '%' || "searchtext" || '%' FROM "Filter" ), "Title" )
```

LIKE, junto con %, significa que se muestran todos los registros que tienen el término de búsqueda en cualquier lugar dentro de ellos. % es un comodín para cualquier número de caracteres antes o después del término de búsqueda. Aún quedan varias preguntas después de que se haya creado esta versión de la consulta:

- Es común usar letras minúsculas para los términos de búsqueda. Entonces, cómo se puede obtener un resultado si se escribe *"enganche"* en lugar de *"Enganche"*.
- Qué otras convenciones escritas deben considerarse.
- Qué pasa con los campos que no están formateados como campos de texto. Y si se pueden buscar o bien fechas o o bien números con el mismo campo de búsqueda.
- Y si, como en el caso del filtro, se desea evitar que los valores NULL en el campo hagan que se muestren todos los registros.

La siguiente variante cubre una o dos de estas posibilidades:

```
SELECT * FROM "Media"
WHERE LOWER("Title") LIKE IFNULL( ( SELECT '%' || LOWER("searchtext") || '%' FROM "Filter" ),
LOWER("Title") )
```

La condición cambia el término de búsqueda y el contenido del campo a minúsculas. Esto también permite comparar oraciones completas.

```
SELECT * FROM "Media"
WHERE
   LOWER("Title") LIKE IFNULL( ( SELECT '%' || LOWER("searchtext") || '%' FROM "Filter" ),
   LOWER("Title") ) OR
   LOWER("Category") LIKE ( SELECT '%' || LOWER("searchtext") || '%' FROM "Filter" )
```

La función IFNULL debe ocurrir solo una vez, de modo que cuando el término de búsqueda sea NULL, se consulta LOWER ("Title") LIKE LOWER ("Title"). Y como *Title* debe ser un campo que no

puede ser NULL, en tales casos se muestran todos los registros. Por supuesto, para múltiples búsquedas de campo, este código se vuelve correspondientemente más largo. En tales casos, es mejor usar una macro, para permitir que el código cubra todos los campos de una vez.

Pero el código todavía funciona con campos que no son texto. Aunque la condición LIKE está realmente adaptada al texto, también funciona para números, fechas y horas sin necesidad de modificaciones. Entonces, de hecho, la conversión de texto no necesita llevarse a cabo. Sin embargo, un campo de tipo tiempo, que es una mezcla de texto y números, no puede interactuar con los resultados de la búsqueda a menos que la consulta se amplíe, de modo que un solo término de búsqueda se subdivida en todos los espacios entre el texto y los números. Esto, sin embargo, aumentará significativamente la complejidad de la consulta.

Consejo

Las consultas que se utilizan para filtrar y buscar registros se pueden incrustar directamente en el formulario.

Toda la condición que se ha reunido anteriormente se puede ingresar en el filtro de fila utilizando las propiedades del formulario:

```
SELECT * FROM "Media"
    WHERE "Title" = IFNULL( ( SELECT "searchtext" FROM "Filter" ), "Title" )
```

…luego se convierte en un formulario que usa el contenido de la tabla *Media*.

Bajo "Filter" tenemos

```
("Media"."Title" = IFNULL( ( SELECT "searchtext" FROM "Filter" ), "Media"."Title" ))
```

En la entrada del filtro asegúrese de que la condición se ponga entre paréntesis y funcione con la convención de nombre de campos "NombreDeTabla"."NombreDeCampo".

La ventaja de esta variante es que el filtro se puede activar y desactivar cuando el formulario está abierto.

Búsqueda con LOCATE

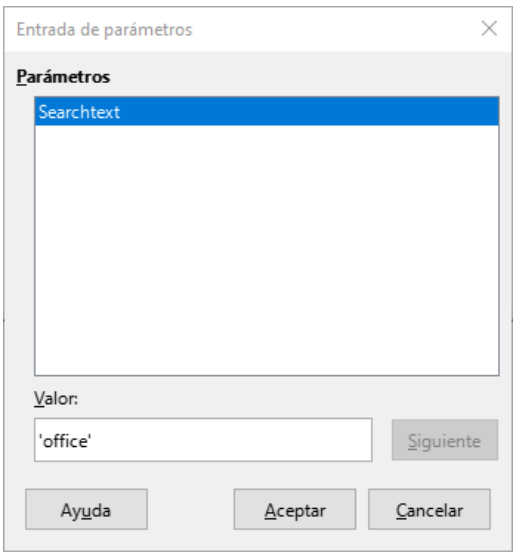
La búsqueda con LIKE suele ser satisfactoria para bases de datos con campos que contienen texto en cantidades que pueden escanearse fácilmente a simple vista. Pero no pasa lo mismo con los campos tipo LONGVARCHAR, que pueden contener varias páginas de texto. Luego, la búsqueda debe determinar dónde se puede encontrar el texto especificado.

Para localizar el texto exactamente, HSQLDB tiene la función LOCATE, la cual toma un término de búsqueda (el texto que desea buscar) como argumento. También puede agregar una posición para buscar.

En resumen: LOCATE (término de búsqueda, campo de texto de la base de datos, posición).

La siguiente explicación utiliza una tabla llamada *table*. La clave primaria se llama *ID* y debe ser única. También hay un campo llamado *memo* que se creó como un campo de tipo LONGVARCHAR. Este campo *memo* contiene algunas oraciones de este manual.[9]

9 Las capturas de pantalla para este capítulo provienen de la base de datos `Example_Autotext_Searchmark_Spelling.odb`, que se incluye en las bases de datos de ejemplo de este libro y que puede descargar desde https://documentation.libreoffice.org/assets/Uploads/Documentation/es/Base-62/SampleDatabases62.zip



Los ejemplos de consultas con esta función se presentan como **consultas parametrizadas**[10]. El texto de búsqueda a ingresar en estos ejemplos siempre es *"office"*.

	ID	memo
▶	1	What are all these things called? The terms used in LibreOffice for most parts of the user interface (the parts of the program
	2	Introduction In everyday office operation, spreadsheets are regularly used to aggregate sets of data
	5	General notes on the creation of a database The basics of creating a database in LibreOffice are described in Chapter 8 of the Getting
*		

Record 1 of 3

```
SELECT "ID", "memo" FROM "table"
WHERE LOWER ( "memo" ) LIKE '%' || LOWER ( :Searchtext ) || '%'
```

Primero usamos la función LIKE. Esta función solo puede usarse en sentencia de las condiciones de la instrucción SQL. Si el texto de búsqueda se encuentra en alguna parte, se muestra el registro correspondiente. La comparación es entre una versión en minúsculas del contenido del campo, usando LOWER ("memo"), y una versión en minúsculas del texto de búsqueda, usando LOWER (: Searchtext), para que la búsqueda no distinga entre mayúsculas y minúsculas. Cuanto más largo sea el texto en el campo *memo*, más difícil será ver el término en el texto recuperado.

LOCATE le muestra con mayor precisión dónde se encuentra su término de búsqueda. En los registros 1 y 2, el término no aparece. En este caso, LOCATE devuelve la posición como "0". Es fácil confirmar la cifra dada para el registro 5: la cadena "Office" comienza en la posición 6. Naturalmente, también sería posible mostrar los resultados de LOCATE de la misma manera que para LIKE.

En la columna *hit*, que se agrega también a la tabla, los resultados de la búsqueda se muestran con mayor precisión. La consulta anterior se ha utilizado como base para esta. Esto permite que la palabra *"position"* se use en la consulta externa en lugar de tener que repetir LOCATE (LOWER (:Searchtext), LOWER ("memo")) cada vez. En principio, esto no es diferente de guardar la consulta anterior y usarla como fuente para esta.

10 Las **consultas parametrizadas** son aquellas que tienen sus argumentos (o parámetros) como una variable cuyo valor puede cambiar en el transcurso de la sesión de trabajo. También conocidas como declaraciones preparadas, son un medio de precompilar una declaración SQL para que todo lo que se necesite proporcionar sean los parámetros (piense en variables) que deben insertarse en la instrucción para que se ejecute.

	ID	memo	position
▶	1	What are all these things called? The terms used in LibreOffice for most parts of the user interface (the	58
	2	Introduction In everyday office operation, spreadsheets are regularly used to aggregate	26
	3	The Base environment contains four work areas: Tables, Queries, Forms, and Reports. Depending on the work area selected, various tasks -	0
	4	Reports – presentation of data Before an actual report in the form of a recall notice can be printed, the	0
	5	General notes on the creation of a database The basics of creating a database in LibreOffice are described in Chapter	87
	6	Accessing external databases An external database must exist before it can be accessed. Assuming that	0
✱			

Record 1 of 6

```
SELECT "ID", "memo",
    LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ) ) AS "position"
FROM "table"
```

"position" = 0 significa que no hay resultado. En este caso, se muestra '** not found **'.

"position" <10 significa que el término de búsqueda aparece justo al comienzo del texto. Se pueden escanear fácilmente 10 caracteres a simple vista. Por lo tanto, se muestra todo el texto. Aquí en lugar de SUBSTRING ("memo", 1), podríamos haber usado solo "memo".

Para todos los demás resultados, la búsqueda busca un espacio ' ' de hasta 10 caracteres antes del término de búsqueda. El texto que se muestra no comienza en el medio de una palabra sino después de un espacio. SUBSTRING ("memo", LOCATE (', "memo", "position" -10) +1) asegura que el texto comience al comienzo de una palabra que contenga 10 caracteres como máximo antes del término de búsqueda.

En la práctica, querríamos usar más caracteres, ya que hay muchas palabras más largas que eso, y el término de búsqueda podría estar dentro de otra palabra con más de 10 caracteres en frente. *LibreOffice* contiene el término de búsqueda *"Office"* con la *"O"* como sexto carácter. Si el término de búsqueda hubiera sido *"hand"*, el registro 4 habría sido fatal para la visualización, pues contiene la palabra *"LibreOffice-Handbooks"* que tiene 12 caracteres a la izquierda de *"hand"*. Si se buscaran espacios para un máximo de 10 caracteres a la izquierda, el primero encontrado habría sido el carácter que sigue a la coma. Esto se mostraría en la columna *hit* como comenzando con *"the built-in help system…"*

	ID	memo	position	hit
▶	1	What are all these things called?	58	in LibreOffice for most p
	2	Introduction	26	everyday office operation
	3	The Base environment contains four work	0	**not found**
	4	Reports – presentation of data	0	**not found**
	5	General notes on the creation of a database	87	in LibreOffice are descri
	6	Accessing external databases	0	**not found**

Record 1 of 6

```
SELECT "ID", "memo", "position",
    CASE
        WHEN "position" = 0 THEN '**not found**'
        WHEN "position" < 10 THEN SUBSTRING ( "memo", 1, 25 )
        ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position" - 10 ) + 1, 25 )
    END AS "hit"
FROM
    (SELECT "ID", "memo", LOCATE(LOWER ( :Searchtext ), LOWER("memo")) AS "position"
    FROM "table" )
```

La técnica de consulta es la misma que para la consulta anterior. Solo la longitud de la coincidencia mostrada (*hit*) se ha reducido a 25 caracteres. La función SUBSTRING requiere como argumentos: primero, uno que busque el texto, luego la posición inicial para el resultado y, como tercer argumento opcional, la longitud de la cadena de texto que se mostrará. Aquí se ha establecido bastante corto para fines de demostración.

Una ventaja de acortarlo es que se reducen los requisitos de almacenamiento para un gran

número de registros y se puede ver fácilmente la ubicación. Una desventaja visible de este tipo de acortamiento de cadena es que el corte se realiza estrictamente de acuerdo con el límite de 25 caracteres, sin tener en cuenta dónde comienzan las palabras.

ID	memo	position	hit
1	What are all these things called?	58	in LibreOffice for most parts
2	Introduction	26	everyday office operation, spreadsheets
3	The Base environment contains four work areas:	0	**not found**
4	Reports – presentation of data	0	**not found**
5	General notes on the creation of a database	87	in LibreOffice are described
6	Accessing external databases	0	**not found**

Record 1 of 6

```
SELECT "ID", "memo", "position",
    CASE
        WHEN "position" = 0 THEN '**not found**'
        WHEN "position" < 10 THEN SUBSTRING ( "memo", 1, LOCATE(' ',"memo", 25 ) )
        ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position" - 10 ) + 1,
            ( LOCATE(' ',"memo", "position" + 20 ) -
                ( LOCATE(' ',"memo", "position" - 10 ) + 1 ) )
            )
    END AS "hit"
FROM
    (SELECT "ID", "memo", LOCATE(LOWER ( :Searchtext ), LOWER("memo")) AS "position"
    FROM "table" )
```

Aquí buscamos desde el carácter en la posición 25 en los *hits* hasta el siguiente lugar con este carácter. El contenido que se mostrará se encuentra entre estas dos posiciones.

Es mucho más simple si la coincidencia aparece al comienzo del campo. Aquí `LOCATE (' ', "memo", 25)` proporciona la posición exacta donde comienza el texto. Dado que queremos que el texto se muestre desde el principio, esto cumple exactamente con la duración del término que se visualizará en el formulario.

La búsqueda del espacio que sigue al término de búsqueda no es más complicada si el término se encuentra más adelante en el campo. La búsqueda simplemente comienza donde está la coincidencia. Luego se cuentan otros 20 caracteres, que se deben seguir en todos los casos. El siguiente espacio después de eso se ubica utilizando `LOCATE (' ', "memo", "position" +20)`. Esto proporciona solo la ubicación concreta dentro del campo, no la longitud de la cadena que se mostrará. Para eso, necesitamos restar la posición en la que debe comenzar la visualización del texto coincidente. Esto ya ha sido establecido dentro de la consulta con `LOCATE (' ', "memo", "position" -10) +1`. De esta manera, se puede encontrar la longitud correcta del texto.

La misma técnica se puede utilizar para encadenar consultas juntas. La consulta anterior ahora se convierte en la fuente de datos para la nueva. Se ha insertado, encerrado entre paréntesis, bajo el término `FROM`. Solo se cambia el nombre de los campos hasta cierto punto, ya que ahora hay múltiples posiciones y coincidencias. Además, la siguiente posición recibe una referencia usando `LOCATE (LOWER (: Searchtext), LOWER ("memo"), "position01" +1)`. Significa que la búsqueda comienza nuevamente en la posición después del texto coincidente anterior.

	ID	memo	position01	hit01	position02	hit02	position03
	2	Introduction	26	everyday office operation,	0	**not found**	0
▶	3	The Base	0	**not found**	0	**not found**	0
	4	Reports –	0	**not found**	0	**not found**	0
	5	General notes on the	87	in LibreOffice are described	209	of LibreOffice, called	307
	6	Accessing external	0	**not found**	0	**not found**	0

Record 3 of 6

```
SELECT "ID", "memo", "position01", "hit01", "position02",
    CASE
        WHEN "position02" = 0 THEN '**not found**'
        WHEN "position02" < 10 THEN SUBSTRING ( "memo", 1, LOCATE( ' ', "memo", 25 ) )
        ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position02" - 10 ) + 1,
            ( LOCATE( ' ', "memo", "position02" + 20 ) -
                ( LOCATE( ' ', "memo", "position02" - 10 ) + 1 ) )
            )
    END AS "hit02",
    CASE
        WHEN "position02" = 0 THEN 0
        ELSE LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ), "position02" + 1 )
    END AS "position03"
FROM
    (SELECT "ID", "memo", "position01",
        CASE
            WHEN "position01" = 0 THEN '**not found**'
            WHEN "position01" < 10 THEN SUBSTRING ("memo", 1, LOCATE(' ', "memo",25 ) )
            ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position01" - 10 ) + 1,
                ( LOCATE( ' ', "memo", "position01" + 20 ) -
                    ( LOCATE( ' ', "memo", "position01" - 10 ) + 1 ) )
                )
        END AS "hit01",
        CASE
            WHEN "position01" = 0 THEN 0
            ELSE LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ), "position01" + 1 )
        END AS "position02"
    FROM
        (SELECT "ID", "memo", LOCATE(LOWER( :Searchtext ), LOWER("memo")) As "position01"
        FROM "table")
    )
```

La consulta más externa establece los campos correspondientes para las otras dos consultas y también proporciona *hit02,* utilizando el mismo método que se utilizó anteriormente para *hit01*. Además, esta consulta más externa determina si hay más coincidencias. La posición correspondiente se da como "position03". Solo el registro 5 tiene más coincidencias, y estas se pueden encontrar en otra subconsulta.

El apilamiento de consultas que se muestran aquí podría llevarse más lejos si se desea. Sin embargo, la adición de cada nueva consulta externa pone una carga adicional en el sistema. Sería necesario realizar algunas pruebas para determinar qué tan lejos fue útil y realista llegar. El “Capítulo 9, Macros” muestra cómo se pueden usar las macros para encontrar todas las cadenas de texto coincidentes en un campo mediante el uso de un formulario.

Manejo de imágenes y documentos en Base

Los formularios en Base usan controles gráficos para manejar las imágenes. Si está utilizando una base de datos interna HSQLDB, los controles gráficos son la única forma de leer imágenes de la base de datos sin usar macros. También se pueden usar como enlaces a imágenes fuera del archivo de base de datos.

En los ejemplos de esta sección se usará la base de datos de ejemplo `Example_Documents_Import_Export.odb`[11].

Leer imágenes en la base de datos

La base de datos requiere una tabla que cumpla al menos las siguientes condiciones:

Nombre del campo	***Tipo de campo***	***Descripción***
ID	INTEGER	ID es la clave principal de esta tabla.
Image	IMAGE	Contiene la imagen como dato binario.

11 Puede descargar esta base de datos de ejemplo desde https://documentation.libreoffice.org/assets/Uploads/Documentation/es/Base-62/SampleDatabases62.zip

Una clave principal (primaria) **tiene** que estar presente, pero no tiene que ser un número entero. Se deben agregar otros campos que agreguen información sobre la imagen.

Los datos que se leerán en el campo de imagen no son visibles en una tabla. En su lugar, verá la palabra <OBJECT>. Del mismo modo, las imágenes no se pueden ingresar directamente en una tabla. Debe usar un formulario que contenga un *control de imagen*, el cual se abre cuando se hace clic para mostrar un diálogo de selección de archivos. Posteriormente muestra la imagen que se leyó desde el archivo seleccionado.

Las imágenes que se deben insertar directamente en la base de datos deben ser lo más pequeñas posible. Como Base no proporciona ninguna manera (excepto mediante el uso de macros) para exportar imágenes en su tamaño original, tiene sentido usar solo el tamaño necesario, por ejemplo, el necesario para imprimir en un informe. Las imágenes cuyos tamaños oscilan los megapíxeles son completamente innecesarias e hinchan la base de datos. Después de agregar solo unas pocas imágenes, el HSQLDB interno proporciona una excepción `Java.NullPointerException` y ya no puede almacenar el registro actual. Incluso si las imágenes no son tan grandes, puede suceder que la base de datos quede inutilizable.

Además, las imágenes no deben integrarse en tablas que se diseñaron para usarse en búsquedas. Si, por ejemplo, tiene una base de datos de personal y se van a incluir imágenes para usar en pases, es mejor almacenarlos en una tabla separada con una clave foránea en la tabla principal. Esto significa que en la tabla principal se puede buscar significativamente más rápido, ya que la tabla en sí no requiere tanta memoria.

Vinculación de imágenes y documentos

Con una estructura de carpetas cuidadosamente diseñada, es más conveniente acceder a archivos externos directamente. Los archivos fuera de la base de datos pueden ser tan grandes como sea necesario, sin tener ningún efecto en el funcionamiento de la base de datos. Desafortunadamente, esto también significa que cambiar el nombre de una carpeta en su propio ordenador o en Internet puede hacer que se pierda el acceso al archivo.

Si no desea leer imágenes directamente en la base de datos, sino solo vincularlas, debe hacer un pequeño cambio en la tabla anterior:

Nombre del campo	***Tipo de campo***	***Descripción***
ID	INTEGER	ID es la clave principal de esta tabla.
Image	TEXT	Contiene la ruta a la imagen.

Si el tipo de campo se establece como TEXT, el *control de imagen* en el formulario transmitirá la ruta al archivo. El *control de imagen* aún puede acceder a la imagen exactamente como una imagen interna.

Lamentablemente, no puede hacer lo mismo con un documento. Ni siquiera es posible leer la ruta, ya que los controles gráficos están diseñados para imágenes gráficas y el diálogo de selección de archivos muestra solo los archivos con un formato gráfico.

Con una imagen, el contenido se puede ver al menos en el control gráfico si se utiliza la ruta al archivo. Un documento no se puede mostrar, incluso si la ruta está almacenada en una tabla. Primero, debemos ampliar un poco la tabla para que al menos una pequeña cantidad de información sobre el documento pueda hacerse visible.

Nombre del campo	*Tipo de campo*	*Descripción*
ID	INTEGER	ID es la clave principal de esta tabla.
Description	TEXT	Descripción del documento, términos de búsqueda.
File	TEXT	Contiene la ruta al documento.

Para hacer visible la ruta al documento, necesitamos incluir un control *selección de archivo* en el formulario.

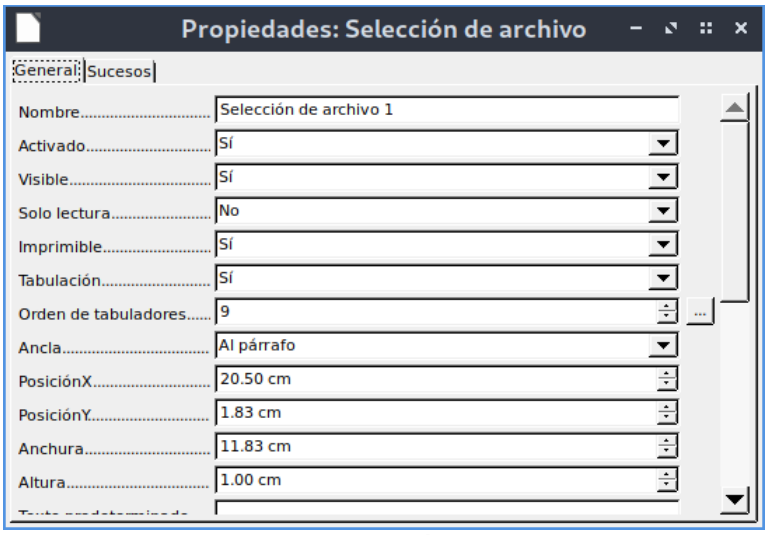


Figura 220: Control para selección de archivos. Puede insertarse en un formulario desde la barra de herramientas Más controles

Un control de *selección de archivo* no tiene pestaña *Datos* en su diálogo *Propiedades*. Por lo tanto, no está vinculado a ningún campo en la tabla subyacente.

Vinculación de documentos con una ruta absoluta

Si se usa el control de *selección de archivo*, la ruta puede mostrarse pero no almacenarse. Para esto es necesario un procedimiento especial que está vinculado a **Sucesos > Texto modificado** en el diálogo *Propiedades* del control:

```
SUB PathRead(oEvent AS OBJECT)
   DIM oForm AS OBJECT
   DIM oField AS OBJECT
   DIM oField2 AS OBJECT
   DIM stUrl AS STRING
   oField = oEvent.Source.Model
   oForm = oField.Parent
   oField2 = oForm.getByName("graphical_control")
   IF oField.Text <> "" THEN
      stUrl = ConvertToUrl(oField.Text)
      oField2.BoundField.updateString(stUrl)
   END IF
END SUB
```

Se pasa al control el evento que desencadena el procedimiento y ayuda a encontrar el formulario y el control en el que se almacenará la ruta. El uso de `oEvent AS OBJECT` simplifica el acceso

cuando otro usuario quiere usar una macro con el mismo nombre en un subformulario. Hace que el control de *selección de archivos* sea accesible a través de `oEvent.Source.Model`. Se accede al formulario como el padre del control de *selección de archivos*. El nombre del formulario es, por lo tanto, irrelevante.

Desde el formulario, ahora se puede acceder al control llamado `"graphical_control"`. Este control se usa normalmente para almacenar las rutas a los archivos de imagen. En este caso, la URL del archivo seleccionado se escribe en él. Para asegurarse de que la URL funciona con las convenciones del sistema operativo, el texto en el control de *selección de archivos* se convierte en una forma generalmente válida mediante `ConvertToUrl`.

La tabla de la base de datos ahora contiene una ruta con el formato absoluto: *file:///…*

Si el ingreso de la ruta a los archivos se leen usando un control gráfico, esto generará una ruta relativa. Para que esto sea utilizable, debe mejorarse. El procedimiento para hacerlo es mucho más largo, ya que implica una comparación entre la ruta de entrada y la real.

Vinculación de documentos con una ruta relacionada

La siguiente macro está vinculada a la propiedad *Texto modificado* del control de *selección de archivos*. (El código se intercala con la explicación de su funcionamiento).

```
SUB PathRead
   DIM oDoc AS OBJECT
   DIM oDrawpage AS OBJECT
   DIM oForm AS OBJECT
   DIM oField AS OBJECT
   DIM oField2 AS OBJECT
   DIM arUrl_Start()
   DIM ar()
   DIM ar1()
   DIM ar2()
   DIM stText AS STRING
   DIM stUrl_complete AS STRING
   DIM stUrl_Text AS STRING
   DIM stUrl AS STRING
   DIM stUrl_cut AS STRING
   DIM ink AS INTEGER
   DIM i AS INTEGER
   oDoc = thisComponent
   oDrawpage = oDoc.Drawpage
   oForm = oDrawpage.Forms.getByName("Form")
   oField = oForm.getByName("graphical_control")
   oField2 = oForm.getByName("filecontrol")
```

Primero, como en todos los procedimientos, se declaran las variables. Luego se buscan los campos que son importantes para la entrada de rutas.

Todo el siguiente código se lleva a cabo solo si realmente hay algo en el campo de selección de archivos, es decir, no ha sido vaciado por un cambio de registro.

```
   IF oField2.Text <> "" THEN
   arUrl_Start = split( oDoc.Parent.Url, oDoc.Parent.Title )
```

```
ar = split( ConvertToUrl(oFeld2.Text), "/" )
stText = ""
```

Se lee la ruta al archivo de la base de datos. Esto se lleva a cabo, como se muestra arriba, primero leyendo toda la URL, luego dividiéndolo en una matriz para que el primer elemento de la matriz contenga la ruta directa.

Luego, todos los elementos de la ruta que se encuentran en el control de *selección de archivos* se leen en la matriz `ar`. El separador es `/`. Esto se puede hacer directamente en Linux. En Windows, el contenido de `oField2` debe convertirse en una URL que utilizará una barra diagonal (`\`) y no una barra diagonal inversa como delimitador de ruta.

El propósito de la división es obtener la ruta al archivo simplemente cortando el nombre de archivo al final. Por lo tanto, en el siguiente paso, la ruta al archivo se vuelve a unir y se coloca en la variable `stText`. El bucle termina no con el último elemento en la matriz `ar`, sino con el elemento anterior.

```
FOR i = LBound(ar()) TO UBound(ar()) - 1
   stText = stText & ar(i) & "/"
NEXT
stText = Left(stText,Len(stText)-1)
arUrl_Start(0) = Left(arUrl_Start(0),Len(arUrl_Start(0))-1)
```

El separador final `/` se elimina nuevamente, ya que de lo contrario aparecería un valor de matriz vacío en la siguiente matriz, lo que interferiría con la comparación de la ruta. Para una comparación correcta, el texto debe convertirse en una URL adecuada que comience con `file:///`.

Finalmente, la ruta al archivo de la base de datos se compara con la ruta que se ha creado.

```
stUrl_Text = ConvertToUrl(stText)
ar1 = split(stUrl_Text,"/")
ar2 = split(arUrl_Start(0),"/")
stUrl = ""
ink = 0
stUrl_cut = ""
```

La matriz `ar1ar2` se compara paso a paso en un bucle.

```
FOR i = LBound(ar2()) TO UBound(ar2())
   IF i <= UBound(ar1()) THEN
```

El siguiente código se ejecuta solo si el número `i` no es mayor que el número de elementos en `ar1`. Si el valor en `ar2` es el mismo que el valor correspondiente en `ar1`, y hasta el momento no se ha encontrado ningún valor incompatible, el contenido común se almacena en una variable que finalmente se puede cortar del valor de la ruta.

```
      IF ar2(i) = ar1(i) AND ink = 0
         THEN  tUrl_cut = stUrl_cut & ar1(i) & "/"
      ELSE
```

Si hay una diferencia en cualquier punto entre las dos matrices, entonces, para cada valor diferente, el signo para subir un directorio se agregará a la variable stUrl.

```
         stUrl = stUrl & "../"
         ink = 1
      END IF
```

Tan pronto como el índice almacenado en `i` sea mayor que el número de elementos en `ar1`, cada valor adicional en `ar2` hará que se almacenen más `../` en la variable `stUrl`.

```
      ELSE
         stUrl = stUrl & "../"
      END IF
      NEXT
      stUrl_complete = ConvertToUrl(oFeld2.Text)
      oFeld.boundField.UpdateString(stUrl & Right(stUrl_complete,Len(stUrl_complete)-
Len(stUrl_cut)))
   END IF
END SUB
```

Cuando se completa el ciclo a través de `ar2`, hemos establecido si y en qué medida el archivo al que se va a acceder es más alto en el árbol que el archivo de base de datos. Ahora se puede crear `stUrl_complete` a partir del texto en el control de *selección de archivos*. Esto también contiene el nombre del archivo. Finalmente, el valor se transfiere al control gráfico. El valor de la URL comienza con `stUrl`, que contiene el número necesario de puntos (`../`). Luego, se corta el comienzo de `stUrl_complete`, la parte que resultó ser la misma para la base de datos y el archivo externo. La forma de cortar la cadena se almacena en `stUrl_cut`.

Mostrar imágenes y documentos vinculados

Las imágenes vinculadas se pueden mostrar directamente en un control gráfico. Pero una pantalla más grande sería mejor para mostrar detalles.

Los documentos normalmente no son visibles en Base.

Para hacer posible este tipo de visualización, nuevamente necesitamos usar macros. Esta macro se inicia utilizando un botón en el formulario que contiene el control gráfico. (El código se intercala con la explicación de su funcionamiento).

```
SUB View(oEvent AS OBJECT)
   DIM oDoc AS OBJECT
   DIM oForm AS OBJECT
   DIM oField AS OBJECT
   DIM oShell AS OBJECT
   DIM stUrl AS STRING
   DIM stField AS STRING
   DIM arUrl_Start()
   oDoc = thisComponent
   oForm = oEvent.Source.Model.Parent
   oField = oForm.getByName("graphical_control")
   stUrl = oField.BoundField.getString
```

Se localiza el control gráfico en el formulario. Como la tabla no contiene la imagen en sí misma, sino solo una ruta almacenada como una cadena de texto, este texto se recupera mediante `getString`.

Luego se determina la ruta al archivo de la base de datos. Se accede al archivo *.odb*, el contenedor de los formularios, utilizando `oDoc.Parent`. La URL completa, incluido el nombre del archivo, se lee con `oDoc.Parent.Url`. El nombre de archivo también se almacena en `oDoc.Parent.Title`. El texto se separa utilizando la función de división con el nombre del

archivo como separador. Esto proporciona la ruta al archivo de la base de datos como el primer y único elemento de la matriz.

```
   arUrl_Start = split(oDoc.Parent.Url,oDoc.Parent.Title)
   oShell = createUnoService("com.sun.star.system.SystemShellExecute")
   stField = convertToUrl(arUrl_Start(0) + stUrl)
   oShell.execute(stField,,0)
END SUB
```

Los programas externos se pueden iniciar utilizando la estructura `com.sun.star.system.SystemShellExecute`. La ruta absoluta al archivo, unida desde la ruta al archivo de la base de datos y la ruta relativa almacenada internamente desde el archivo de la base de datos, se pasa al programa externo. La interfaz gráfica del sistema operativo determina qué programa se llama para abrir el archivo.

El mandato `oShell.execute` toma tres argumentos. El primero es un archivo ejecutable o la ruta a un archivo de datos que está vinculado a un programa por el sistema. El segundo es una lista de argumentos para el programa. El tercero es un número que determina cómo se deben informar los errores. Las posibilidades son `0` (mensaje de error predeterminado), `1` (sin mensaje) y `2` (solo permiten la apertura de URL absolutas).

Leer documentos en la base de datos

Al leer los documentos, siempre se deben observar las siguientes condiciones:

- Cuanto más grandes sean los documentos, más difícil de manejar será la base de datos. Por lo tanto, para documentos grandes, una base de datos externa es mejor que la interna.
- Al igual que las imágenes, los documentos no se pueden buscar. Se almacenan como datos binarios y, por lo tanto, se pueden colocar en un *contro de imagen*.
- Los documentos que deban leerse desde la base de datos interna de HSQLDB solo se pueden leer con macros. No puede hacerse solo con SQL.

  Las siguientes macros para leer dentro y fuera dependen de una tabla que incluye una descripción de los datos y el nombre de archivo original, así como una versión binaria del archivo. El nombre de archivo no se almacena automáticamente junto con el archivo, pero puede proporcionar información útil sobre el tipo de datos almacenados en un archivo que se va a leer. Solo entonces el archivo puede ser leído con seguridad por otros programas.

  La tabla contiene los siguientes campos:

Nombre del campo	***Tipo de campo***	***Descripción***
ID	INTEGER	ID es la clave principal de esta tabla.
Description	TEXT	Descripción del documento, términos de búsqueda
File	IMAGE	La imagen o archivo en forma binaria.
Filename	TEXT	El nombre del archivo, incluido el sufijo del archivo. Importante para la lectura posterior.

El formulario formulario *fileimport_name* (de la base de datos de ejemplo `Example_Documents_Import_Export.odb`[12]) para leer archivos dentro y fuera se ve así:

12 Puede descargar esta base de datos de ejemplo desde https://documentation.libreoffice.org/assets/Uploads/Documentation/es/Base-62/SampleDatabases62.zip

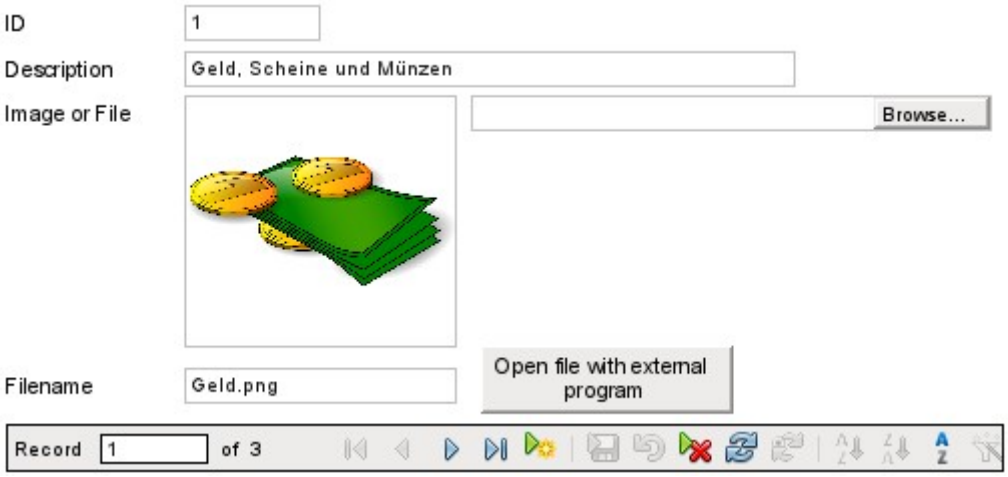


Figura 221: Formulario fileimport_name de la base de datos de ejemplo Example_Documents_Import_Export.odb

Si los archivos de imagen están presentes en la base de datos, se pueden ver en el control gráfico del formulario. Todos los demás tipos de archivo son invisibles.

La siguiente macro para leer un archivo se activa mediante el diálogo *Propiedades* del control *selección de archivos*: **Sucesos > Texto modificado**.

```
SUB FileInput_withName(oEvent AS OBJECT)
   DIM oForm AS OBJECT
   DIM oField AS OBJECT
   DIM oField2 AS OBJECT
   DIM oField3 AS OBJECT
   DIM oStream AS OBJECT
   DIM oSimpleFileAccess AS OBJECT
   DIM stUrl AS STRING
   DIM stName AS STRING
   oField = oEvent.Source.Model
   oForm = oField.Parent
   oField2 = oForm.getByName("txt_filename")
   oField3 = oForm.getByName("graphical_control")
   IF oField.Text <> "" THEN
      stUrl = ConvertToUrl(oField.Text)
      ar = split(stUrl,"/")
      stName = ar(UBound(ar))
      oField2.BoundField.updateString(stName)
      oSimpleFileAccess = createUnoService("com.sun.star.ucb.SimpleFileAccess")
      oStream = oSimpleFileAccess.openFileRead(stUrl)
      oField3.BoundField.updateBinaryStream(oStream, oStream.getLength())
   END IF
END SUB
```

Como el evento desencadenante de la macro proporciona el nombre de otro campo de formulario, no es necesario verificar si los campos son parte del formulario principal o de un subformulario. Todo lo que es necesario saber es que todos los controles deben estar en el mismo formulario.

El control `"txt_filename"` almacena el nombre del archivo a buscar. En el caso de las imágenes, este nombre debe ingresarse a mano sin usar una macro. Aquí, en cambio, el nombre de archivo se determina a través de una URL y se ingresa automáticamente cuando se leen los datos.

El campo `"graphical_control"` almacena los datos reales, tanto para imágenes como para otros archivos.

La ruta completa, incluido el nombre del archivo, se lee desde el control de *selección de archivos* utilizando `oFeld.Text`. Para garantizar que la URL no se vea afectada por condiciones específicas del sistema operativo, el texto que se ha leído se convierte al formato de URL estándar mediante `ConvertToUrl`. Esta URL universalmente válida se divide en una matriz. El separador es `/`. El último elemento de la ruta es el nombre del archivo. `Ubound(ar)` da el índice para este último elemento. El nombre de archivo real puede leerse usando `ar(Ubound(ar))` y transferirse al campo como una cadena.

Para leer en el archivo en sí requiere `UnoService com.sun.star.ucb.SimpleFileAccess`. Este servicio puede leer el contenido del archivo como una secuencia de datos. Esto se almacena temporalmente en el objeto `oStream` y luego se inserta como una secuencia de datos en el control vinculado al campo *File* en la tabla. Esto requiere que se proporcione la longitud del flujo de datos, así como el objeto `oStream`.

Los datos ahora están dentro del control del formulario como con una entrada normal. Sin embargo, si el formulario simplemente se cierra en este punto, los datos no se almacenan. El almacenamiento requiere que se presione el botón *Guardar registro* en la barra de navegación. También ocurre automáticamente al pasar al siguiente registro.

Determinar los nombres de los archivos de imagen.

En el método anterior se mencionó brevemente que el nombre del archivo utilizado para la entrada en un control gráfico no se puede determinar directamente. Aquí hay una macro para determinar este nombre de archivo, la cual se ajusta al formulario anterior. El nombre de archivo no puede determinarse con certeza por un suceso directamente vinculado al control gráfico. Por lo tanto, la macro se inicia utilizando el diálogo **Propiedades del formulario > Sucesos > Antes de la acción en el registro de datos**.

```
SUB ImagenameRead(oEvent AS OBJECT)
   oForm = oEvent.Source
   IF InStr(oForm.ImplementationName, "ODatabaseForm") THEN
      oField = oForm.getByName("graphical_control")
      oField2 = oForm.getByName("txt_filename")
      IF oField.ImageUrl <> "" THEN
         stUrl = ConvertToUrl(oFeld.ImageUrl)
         ar = split(stUrl,"/")
         stName = ar(UBound(ar))
         oField2.BoundField.updateString(stName)
      END IF
   END IF
END SUB
```

Antes de la acción en el registro de datos se llevan a cabo dos implementaciones con sus respectivos diferentes nombres. Se puede acceder fácilmente al formulario utilizando la implementación `"ODatabaseForm"`.

En el control gráfico se puede acceder a la URL de la fuente de datos utilizando `ImageUrl`. Esta URL se lee y el nombre del archivo se determina utilizando el procedimiento anterior `FileInput_withName` para luego transferirse al campo `"txt_filename"`.

Eliminar nombres de archivos de imagen de la memoria

Si después de ejecutar la macro anterior pasa al siguiente registro, la ruta a la imagen original aún está disponible. Si ahora se lee un archivo que no es de imagen usando el control *selección de archivos*, el nombre de archivo de la imagen sobrescribirá el nombre de ese archivo, a menos que use la siguiente macro.

Lamentablemente, la macro anterior no puede eliminar la ruta, ya que el archivo de imagen solo se lee cuando se guarda el registro. Eliminar la ruta en ese punto eliminaría la imagen.

La macro se inicia utilizando el diálogo **Propiedades del formulario> Sucesos> Después de la acción en el registro de datos**.

```
SUB ImagenameReset(oEvent AS OBJECT)
   oForm = oEvent.Source
   IF InStr(oForm.ImplementationName, "ODatabaseForm") THEN
      oField = oForm.getByName("graphical_control")
      IF oField.ImageUrl <> "" THEN
         oField.ImageUrl = ""
      END IF
   END IF
END SUB
```

Como en el procedimiento `"ImagenameRead"`, en esta macro se accede al control gráfico. Si hay una entrada en `ImageUrl`, se elimina.

Lectura y visualización de imágenes y documentos

Tanto para archivos no gráficos como para imágenes de tamaño original, se debe presionar el botón *Open file with external program* del formulario *fileimport_name* de la base de datos de ejemplo `Example_Documents_Import_Export.odb`[13]. Luego, los archivos en la carpeta temporal se pueden leer y mostrar usando el programa vinculado al sufijo de archivo en el sistema operativo.

La macro se inicia usando el diálogo *Propiedades:Botón* y yendo a Sucesos > Ejecutar una acción.

```
SUB FileDisplay_withName(oEvent AS OBJECT)
   DIM oDoc AS OBJECT
   DIM oDrawpage AS OBJECT
   DIM oForm AS OBJECT
   DIM oField AS OBJECT
   DIM oField2 AS OBJECT
   DIM oStream AS OBJECT
   DIM oShell AS OBJECT
   DIM oPath AS OBJECT
   DIM oSimpleFileAccess AS OBJECT
```

13 Puede descargar esta base de datos de ejemplo desde `https://wiki.documentfoundation.org/images/6/62/SampleDatabases62.zip`

```
   DIM stName AS STRING
   DIM stPath AS STRING
   DIM stField AS STRING
   oForm = oEvent.Source.Model.Parent
   oField = oForm.getByName("graphical_control")
   oField2 = oForm.getByName("txt_filename")
   stName = oField2.Text
   IF stName = "" THEN
      stName = "file"
   END IF
   oStream = oField.BoundField.getBinaryStream
   oPath = createUnoService("com.sun.star.util.PathSettings")
   stPath = oPath.Temp & "/" & stName
   oSimpleFileAccess = createUnoService("com.sun.star.ucb.SimpleFileAccess")
   oSimpleFileAccess.writeFile(stPath, oStream)
   oShell = createUnoService("com.sun.star.system.SystemShellExecute")
   stField = convertToUrl(stPath)
   oShell.execute(stField,,0)
END SUB
```

La posición de los otros controles afectados en el formulario viene dada por el botón. Si falta un nombre de archivo, el archivo simplemente recibe el nombre `"file"`.

El contenido del control de formulario `"graphical_control"` corresponde al del campo *File* en la tabla. Se lee como una secuencia de datos. La ruta a la carpeta temporal se usa como ruta para estos datos; se puede configurar usando **Herramientas > Opciones > LibreOffice > Rutas**.

Si los datos se utilizarán posteriormente para otros fines, y no solo se van a mostrar, se pueden copiar desde esta ruta. Dentro de la macro, el archivo se abre directamente después de una lectura exitosa, utilizando el programa que ha sido vinculado al sufijo del archivo por la interfaz gráfica de usuario del sistema operativo.

Fragmentos de código

Estos fragmentos de código proceden de consultas en las listas de correo. Surgen de problemas particulares que tal vez podrían ser útiles como soluciones para sus propios experimentos de bases de datos.

Obtener la edad actual de alguien

Una consulta debe calcular la edad real de una persona a partir de una fecha de nacimiento. Consulte también las funciones en el Apéndice de esta guía.

```
SELECT DATEDIFF( 'yy', "Birthdate", CURDATE()) AS "Age" FROM "Person"
```

Esta consulta da la edad como una diferencia en años. Pero, la edad de un niño nacido el 31 de diciembre de 2011 se daría como 1 año si la fecha actual fuera el 1 de enero de 2012. Por lo tanto, también debemos considerar la posición del día dentro del año. Esto es posible si se usa la función DAYOFYEAR(). Otra función llevará a cabo la comparación.

```
SELECT CASEWHEN (
   DAYOFYEAR( "Birthdate" ) > DAYOFYEAR( CURDATE() ),
   DATEDIFF ( 'yy', "Birthdate", CURDATE() ) - 1,
```

```
    DATEDIFF( 'yy', "Birthdate", CURDATE() )
)
AS "Age" FROM "Person"
```

Ahora tenemos la edad actual correcta en años.

CASEWHEN() también se puede usar para hacer que el texto "Birthdate today" aparezca en otro campo, si `DAYOFYEAR( "Birthdate" ) = DAYOFYEAR( CURDATE() )`.

Ahora podría surgir una objeción sutil: ¿Qué pasa con los años bisiestos?. Para las personas nacidas después del 28 de febrero, habrá un error de un día. No es un problema grave en el uso diario, pero deberíamos esforzarnos por la precisión.

```
CASEWHEN (
    ( MONTH( "Birthdate" ) > MONTH( CURDATE() )
    ) OR
    (
    ( MONTH( "Birthdate" ) = MONTH ( CURDATE() ) )
    AND ( DAY( "Birthdate" ) > DAY (CURDATE() ) )
    ),
DATEDIFF( 'yy', "Birthdate", CURDATE() ) - 1,
DATEDIFF( 'yy', "Birthdate", CURDATE() )
)
```

El código anterior logra este objetivo. Mientras el mes de la fecha de nacimiento sea mayor que el mes actual, la función de diferencia de año restará un año. Igualmente, se restará un año cuando los dos meses sean iguales, pero el día del mes para la fecha de nacimiento es mayor que el día en la fecha actual. Lamentablemente, esta fórmula no es comprensible para la GUI.

Solo con el diálogo *Ejecutar instrucción SQL* (que se accede desde **Herramientas > SQL…**) manejará esta consulta con éxito y eso evitaría que nuestra consulta sea editada. Pero la consulta debe ser editable, así que aquí está cómo engañar a la interfaz gráfica de usuario (GUI, por sus siglas en inglés):

```
CASE
    WHEN MONTH("Birthdate") > MONTH(CURDATE())
    THEN DATEDIFF('yy',"Birthdate",CURDATE())-1
WHEN (MONTH("Birthdate") = MONTH(CURDATE()) AND DAY("Birthdate") > DAY(CURDATE()))
    THEN DATEDIFF('yy',"Birthdate",CURDATE())-1
    ELSE DATEDIFF('yy',"Birthdate",CURDATE())
END
```

Con esta instrucción, la interfaz gráfica de usuario ya no reacciona con un mensaje de error. La edad ahora se da con precisión incluso en años bisiestos y la consulta sigue siendo editable.

Mostrar los cumpleaños que ocurrirán en los próximos días

Con un pequeño fragmento de cálculo, podemos determinar a partir de la tabla quién celebrará sus cumpleaños dentro de los próximos ocho días.

```
SELECT * FROM "Table"
WHERE
    DAYOFYEAR( "Date" ) BETWEEN DAYOFYEAR( CURDATE() )
    AND DAYOFYEAR( CURDATE() ) + 7
    OR DAYOFYEAR( "Date" ) < 7 -
        DAYOFYEAR( CAST( YEAR( CURDATE() ) || '-12-31' AS DATE) ) +
        DAYOFYEAR( CURDATE() )
```

La consulta muestra todos los registros cuya entrada de fecha se encuentra entre el día actual del año y los siguientes 7 días.

Para mostrar 8 días, incluso al final de un año, el día en que comenzó el año debe verificarse a fondo. Esta comprobación se produce solo para los números de día que son como máximo 7 días posteriores al número del último día para el año actual (generalmente 365) más el número de día para la fecha actual. Si la fecha actual es más de 7 días desde el final del año, el total es < 1. Ningún registro en la tabla tiene una fecha como esa, por lo que en tales casos esta condición parcial no se cumple.

En la fórmula anterior, los años bisiestos darán un resultado incorrecto, ya que sus fechas se desplazan por la ocurrencia del 29 de febrero. El código debe ser más extenso para evitar este error:

```
SELECT * FROM "Table"
WHERE
   CASE
      WHEN
         DAYOFYEAR(CAST(YEAR("Date")||'-12-31' AS DATE)) = 366
         AND DAYOFYEAR("Date") > 60 THEN DAYOFYEAR("Date") - 1
      ELSE
         DAYOFYEAR("Date")
   END
BETWEEN
   CASE
      WHEN
         DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) = 366
         AND DAYOFYEAR(CURDATE()) > 60 THEN DAYOFYEAR(CURDATE()) - 1
      ELSE
         DAYOFYEAR(CURDATE())
   END
AND
   CASE
      WHEN
         DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) = 366
         AND DAYOFYEAR(CURDATE()) > 60 THEN DAYOFYEAR(CURDATE()) + 6
      ELSE
         DAYOFYEAR(CURDATE()) + 7
   END
OR DAYOFYEAR("Datum") < 7 -
   DAYOFYEAR(CAST(YEAR(CURDATE())||'-12-31' AS DATE)) +
   DAYOFYEAR(CURDATE())
```

Los años bisiestos se pueden reconocer teniendo 366 como el número total de días en lugar de 365. Esto se usa para la determinación correspondiente.

Por un lado, cada valor de fecha debe probarse para ver si se encuentra en un año bisiesto, y también para el recuento correcto para el día 60 (31 días en enero y 29 en febrero). En este caso, todos los siguientes valores de DAYOFYEAR para la fecha deben incrementarse en 1. Luego, el 1 de marzo en un año bisiesto corresponderá exactamente al 1 de marzo en un año normal.

Por otro lado, el año actual (CURDATE ()) debe probarse para ver si de hecho es un año bisiesto. Aquí también el número de días debe incrementarse en 1.

Mostrar el valor final para los próximos 8 días tampoco es tan simple, ya que el año todavía no está incluido en la consulta. Sin embargo, esta sería una condición fácil de agregar para el siguiente:

YEAR("Date") = YEAR(CURDATE()) para el actual o YEAR("Date") = YEAR(CURDATE()) + 1

Agregar días al valor fecha

Al prestar medios, la biblioteca puede querer saber el día exacto en que debe devolverse el medio. Desafortunadamente, el HSQLDB interno no proporciona la función DATEADD() que está disponible en muchas bases de datos externas y también en el Firebird interno de LibreOffice Base. Aquí sigue una forma indirecta de lograr esto por un período de tiempo limitado.

Primero se crea una tabla que contiene una secuencia de fechas que cubren el lapso de tiempo deseado. Para este propósito, se abre Calc, se coloca la cadena *ID* en la celda A1 y *Date* en la celda B1. En la celda A2 ingresamos 1 y en la celda B2 la fecha de inicio, por ejemplo, 15/01/2015. Seleccione A2 y B2 y arrástrelos hacia abajo. Esto creará una secuencia de números en la columna A y una secuencia de fechas en la columna B.

Luego, toda esta tabla, incluidos los encabezados de columna, se selecciona, copia e importa a Base: haga clic con el botón derecho en el panel inferior *Tablas* y vaya al menú contextual *Pegar…* En el diálogo *Copiar tabla* que se abre, escriba *Date* en el campo *Nombre de tabla*. En Opciones, seleccione Definición y Datos y marque la casilla Usar la primera fila para los nombres de las columnas. Pulse el botón *Siguiente*. Transfiera todas las columnas, que en este caso solo son *ID* y *Date*, al panel de la derecha en el paso *Aplicar columnas*. Pulse *Siguiente* otra vez. Después de eso, en el paso *Formatos de tipos*, asegúrese de que el campo *ID* tenga el tipo INTEGER y el campo *Date* el tipo DATE. No es necesaria una clave principal, ya que los registros no se modificarán más adelante. Como no se ha definido una clave primaria, la tabla está protegida contra escritura. Pulse el botón *Crear*. Verá la nueva tabla en el panel *Tablas* de Base.

Consejo

También puede usar una técnica de consulta para crear dicha vista. Si usa una tabla de filtro, incluso puede controlar la fecha de inicio y el rango de valores de fecha.

```
SELECT DISTINCT CAST
    ( "Y"."Nr" + (SELECT "Year" FROM "Filter" WHERE "ID" = True) - 1
|| '-' ||
    CASEWHEN( "M"."Nr" < 10, '0' || "M"."Nr", '' || "M"."Nr" ) ||
'-' ||
    CASEWHEN( "D"."Nr" < 10, '0' || "D"."Nr", '' || "D"."Nr" )
    AS DATE ) AS "Date"
FROM "Nrto31" AS "D", "Nrto31" AS "M", "Nrto31" AS "Y"
WHERE "Y"."Nr" <= (SELECT "Year" FROM "Filter" WHERE "ID" = True)
AND "M"."Nr" <= 12 AND "D"."Nr" <= 31
```

Esta vista accede a una tabla que contiene solo los números del 1 al 31 y está protegida contra escritura. Otra tabla de filtros contiene el año de inicio y el intervalo de años que la vista debería cubrir. La fecha se junta a partir de estos valores, creando una expresión de fecha (año, mes, día) en texto, que luego se puede convertir en una fecha. HSQLDB acepta todos los días hasta 31 por mes y cadenas con formato como 31/02/2015. Sin embargo, el 31/02/2015 se transmite como el 3/03/2015. Por lo tanto, al preparar la vista, debe usar DISTINCT para excluir valores de fecha duplicados.

Aquí, la siguiente vista es efectiva:

```
SELECT "a"."Date",
    (SELECT COUNT(*) FROM "View_date" WHERE "Date" <= "a"."Date")
```

```
    AS "lfdNr"
FROM "View_Date" AS "a"
```

Usando la numeración de línea, el valor de la fecha se convierte en un número. Como no puede eliminar datos en una vista, no se necesita protección adicional contra escritura.

Mediante una consulta, ahora podemos determinar una fecha específica, por ejemplo, la fecha dentro de 14 días:

```
SELECT "a"."Loan_Date",
   (SELECT "Date" FROM "Date" WHERE "ID" =
      (SELECT "ID" FROM "Date" WHERE "Date" = "a"."Loan_Date")+14)
      AS "Returndate"
FROM "Loans" AS "a"
```

La primera columna muestra la fecha del préstamo. Se accede a esta columna mediante una subconsulta correlacionada que nuevamente se divide en dos consultas. `SELECT "ID" FROM "Date"` proporciona el valor del campo `ID`, correspondiente a la fecha de emisión, y 14 días se agrega al valor. El resultado es asignado al campo `ID` por la subconsulta externa. Esta nueva identificación determina qué fecha entra en el campo de fecha.

Desafortunadamente, en la visualización de esta consulta, el tipo de fecha no se reconoce automáticamente, por lo que se hace necesario utilizar el formato. En un formulario, la visualización correspondiente se puede almacenar, de modo que cada consulta arroje un valor de fecha.

Es posible una variante directa para determinar el valor de la fecha utilizando una forma más corta:

```
SELECT "Loan_Date",
   DATEDIFF( 'dd', '1899-12-30', "Loan_Date" ) + 14
      AS "Returndate"
FROM "Table"
```

El valor numérico devuelto puede formatearse dentro de un formulario como una fecha, utilizando un campo formateado. Sin embargo, se necesita mucho trabajo para que esté disponible para el procesamiento posterior de SQL en una consulta.

Agregar una hora a un intervalo de tiempo

MySQL tiene una función llamada `TIMESTAMPADD()`. Una función similar no existe en HSQLDB. Pero el valor numérico interno de la marca de tiempo se puede usar para sumar o restar, usando un campo formateado en un formulario.

A diferencia de la adición de días a una fecha, los tiempos causan un problema que puede no ser obvio al principio.

```
SELECT "DateTime"
   DATEDIFF('ss', '1899-12-30', "DateTime" ) / 86400.0000000000 +
      36/24 AS "DateTime+36hours"
FROM "Table"
```

El nuevo tiempo calculado se basa en la diferencia con el tiempo cero del sistema. Como en los cálculos de fechas, esta es la fecha del 30/12/1899.

Nota

Se supone que la fecha cero del 30/12/1899 fue elegida porque el año 1900, a diferencia de la mayoría de los años divisible por 4, no fue un año bisiesto. Entonces, la etiqueta "1" del cálculo interno se movió de nuevo al 31/12/1899 y no al 01/01/1900.

La diferencia se expresa en segundos, pero el número interno cuenta los días como números antes del punto decimal, y las horas, minutos y segundos como lugares decimales. Como un día contiene 60 * 60 * 24 segundos, el segundo recuento debe dividirse por 86400 para poder calcular los días y fracciones de días correctamente. Si el HSQLDB interno debe proporcionar decimales, debe incluirse en el cálculo, por lo que en lugar de 86400, debemos dividir por 86400.0000000000.

Los lugares decimales en una consulta deben usar un punto decimal como separador, independientemente de las convenciones locales. El resultado tendrá 10 decimales después del punto.

A este resultado se deben agregar las horas totales como una parte fraccionaria de un día. La figura calculada, con el formato adecuado, se puede crear en la consulta. Lamentablemente, el formato no se guarda, pero se puede transferir con el formato correcto utilizando un campo formateado en un formulario o informe.

Si se van a agregar minutos o segundos, tenga cuidado de que se suministren como fracciones de un día.

Si la fecha cae dentro de noviembre, diciembre, enero, etc., no hay problemas con el cálculo. Parecen bastante precisos: agregar 36 horas a una marca de tiempo del 20/01/2015 13:00:00 da 22/01/2015 00:00:00. Pero las cosas son diferentes para el 20/04/2015 13:00:00. El resultado es 22/04/2015 00:00:00. El cálculo sale mal debido al horario de verano. La hora "perdida" o "ganada" por el cambio de hora no se tiene en cuenta. Dentro de una sola zona horaria, hay varias formas de obtener un resultado "correcto". Aquí hay una variación simple:

```
SELECT "DateTime"
DATEDIFF( 'dd', 1899-12-30', "DateTime" ) +    HOUR  ( "DateTime" ) / 24.0000000000 +
   MINUTE( "DateTime" ) / 1440.0000000000 + SECOND( "DateTime" ) / 86400.0000000000 +
   36/24
AS "DateTime+36hours"  FROM "Table"
```

En lugar de contar horas, minutos y segundos desde el origen de la fecha, se cuentan a partir de la fecha actual. El 20/05/2015 la hora es 13:00 pero sin horario de verano, se mostrará como 12:00. La función HOUR tiene en cuenta el horario de verano y proporciona 13 horas como parte del tiempo por hora. Esto se puede agregar correctamente a la parte diaria. Los minutos y segundos se tratan exactamente de la misma manera. Finalmente, las horas adicionales se agregan como una parte fraccionaria de un día y todo se muestra como una marca de tiempo calculada usando el formato de celda.

Deben tenerse en cuenta dos cosas en este cálculo:

- Al pasar del horario de invierno al horario de verano, los valores por hora no salen correctamente. Esto puede corregirse usando una tabla auxiliar, que toma las fechas para el comienzo y el final del horario de verano y corrige el conteo por hora. Un negocio algo complicado.
- La visualización de tiempos solo es posible con campos formateados. El resultado es un número decimal, no una marca de tiempo que podría almacenarse directamente como tal en la base de datos. Debe copiarse dentro del formulario o convertirse de un número decimal a una marca de tiempo mediante una consulta complicada. El punto de ruptura en la conversión es el valor de la fecha, ya que pueden estar involucrados años bisiestos o meses con diferentes números de días.

Obtener un saldo de cuenta corriente por conceptos

En lugar de usar una agenda doméstica, una base de datos en un PC puede simplificar el trabajo agotador de sumar gastos para alimentos, ropa, transporte, etc. Queremos que la mayoría de estos detalles sean visibles de inmediato en la base de datos, por lo que nuestro ejemplo supone que los ingresos y los gastos se almacenarán como valores señalados en un campo llamado *Amount*. En principio, todo se puede ampliar para cubrir campos separados y una suma relevante para cada uno.

```
SELECT "ID", "Amount", (SELECT SUM("Amount")
FROM "Cash"
WHERE "ID" <= "a"."ID") AS "Balance" FROM "Cash" AS "a"
ORDER BY "ID" ASC
```

Esta consulta provoca para cada nuevo registro un cálculo directo del saldo de la cuenta corriente. Al mismo tiempo, la consulta sigue siendo editable porque el campo *Balance* se crea a través de una subconsulta de correlación. La consulta depende del *ID* de clave principal creada automáticamente para calcular el estado de la cuenta. Sin embargo, los saldos generalmente se calculan diariamente. Entonces necesitamos una consulta de fecha.

```
SELECT "ID", "Date", "Amount", ( SELECT SUM( "Amount" )
FROM "Cash"
WHERE "Date" <= "a"."Date" ) AS "Balance" FROM "Cash" AS "a"
ORDER BY "Date", "ID" ASC
```

El gasto ahora aparece ordenado y sumado por fecha. Todavía queda la cuestión de la categoría, ya que queremos los saldos correspondientes para las categorías individuales de gasto.

```
SELECT "ID", "Date", "Amount", "Acct_ID",
( SELECT "Acct" FROM "Acct" WHERE "ID" = "a"."Acct_ID" ) AS "Acct_name",
( SELECT SUM( "Amount" ) FROM "Cash" WHERE "Date" <= "a"."Date" AND "Acct_ID" =
"a"."Acct_ID" ) AS "Balance",
( SELECT SUM( "Amount" ) FROM "Cash" WHERE "Date" <= "a"."Date" ) AS "Total_balance"
FROM "Cash" AS "a"
ORDER BY "Date", "ID" ASC
```

Esto crea una consulta editable en la que, además de los campos de entrada (*Date, Amount, Acct_ID*), el nombre de la cuenta, el saldo relevante y el saldo total aparecen juntos. Como las subconsultas de correlación se basan parcialmente en entradas anteriores (`"Date" <= "a"."Date"`), solo las nuevas entradas pasarán sin problemas. Las alteraciones a un registro anterior son inicialmente detectables solo en ese registro. La consulta debe actualizarse si se van a realizar cálculos posteriores que dependen de ella.

Numerar líneas

Los campos que se incrementan automáticamente están bien. Sin embargo, no le dicen definitivamente cuántos registros están presentes en la base de datos o están realmente disponibles para ser consultados. Los registros a menudo se eliminan y muchos usuarios intentan en vano determinar qué números ya no están presentes para que el número corrido coincida.

```
SELECT "ID", ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <= "a"."ID" ) AS "Nr."
FROM "Table" AS "a"
```

El campo *ID* se lee, y el segundo campo se determina mediante una subconsulta de correlación, que busca determinar cuántos valores de campo en *ID* son menores o iguales al valor de campo actual. A partir de esto se crea un número de línea en ejecución.

Cada registro al que desea aplicar esta consulta contiene campos. Para aplicar esta consulta a los registros, primero debe agregar estos campos a la consulta. Puede colocarlos en el orden que

desee en la cláusula SELECT. Si tiene los registros en un formulario, debe modificar el formulario para que los datos del formulario provengan de esta consulta.

Por ejemplo, el registro contiene *field1, field2* y *field3*. La consulta completa sería:

```
SELECT "ID", "field1", "field2", "field3", ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <=
"a"."ID" ) AS "Nr." FROM "Table" AS "a"
```

También es posible una numeración para una agrupación correspondiente:

```
SELECT "ID", "Calculation", ( SELECT COUNT( "ID" ) FROM "Table" WHERE
"ID" <= "a"."ID" AND "Calculation" = "a"."Calculation" ) AS "Nr." FROM
"Table" AS "a" ORDER BY "ID" ASC, "Nr." ASC
```

Aquí una tabla contiene diferentes números calculados. ("Calculation"). Para cada número calculado, "Nr." se expresa por separado en orden ascendente después de ordenar en el campo *ID*. Esto produce una numeración de 1 en adelante.

Si el orden de clasificación real dentro de la consulta está de acuerdo con los números de línea, se debe asignar un tipo apropiado de clasificación. Para este propósito, el campo de clasificación debe tener un valor único en todos los registros. De lo contrario, dos números de lugar tendrán el mismo valor. Esto realmente puede ser útil si, por ejemplo, se representa el orden de lugar en una competición, ya que resultados idénticos conducirán a una posición conjunta. Para que el orden de lugar se exprese de tal manera que, en caso de posiciones conjuntas, se omita el valor siguiente, la consulta debe construirse de manera algo diferente:

```
SELECT "ID",
   ( SELECT COUNT( "ID" ) + 1 FROM "Table" WHERE "Time" < "a"."Time" ) AS "Place"
FROM "Table" AS "a"
```

Se evalúan todas las entradas para las cuales el campo *Time* tiene un valor menor. Eso cubre a todos los atletas que llegaron al puesto ganador antes que el atleta actual. A este valor se agrega el número 1. Esto determina el lugar del atleta actual. Si el tiempo es idéntico al de otro atleta, se colocan conjuntamente. Esto hace posible situar los puestos como 1er lugar, 2º lugar, 2º lugar, 4º lugar.

Sería más problemático si se necesitaran los números de línea y el lugar al mismo tiempo. Podría ser útil si se necesita combinar varios registros en una línea.

```
SELECT "ID", ( SELECT COUNT( "ID" ) + 1 FROM "Table"
   WHERE "Time" < "a"."Time" ) AS "Place",
CASE WHEN(SELECT COUNT("ID")+1 FROM "Table" WHERE "Time"="a"."Time" ) = 1
   THEN (SELECT COUNT("ID")+1 FROM "Table" WHERE "Time"<"a"."Time")
   ELSE (SELECT(SELECT COUNT("ID")+1 FROM "Table" WHERE "Time"<"a"."Time")
   + COUNT("ID") FROM "Table" WHERE "Time"="a"."Time" "ID"<"a"."ID"
END
AS "LineNumber" FROM "Table" AS "a"
```

La segunda columna todavía da el orden del puesto. La tercera columna verifica primero si solo una persona cruzó la línea esta vez. Si es así, el orden de puestos se convierte directamente en un número de línea. De lo contrario, se agrega un valor adicional al puesto. Por el mismo tiempo ("Time" = "a"."Time") se agrega al menos 1, si hay otra persona con la *ID* de clave primaria, cuya clave primaria es más pequeña que la clave primaria en el registro actual ("ID"<"a"."ID"). Por lo tanto, esta consulta produce valores idénticos para el puesto siempre que no exista una segunda persona con el mismo tiempo. Si existe una segunda persona con el mismo tiempo, la identificación determina qué persona tiene el número de línea menor.

Por cierto, esta clasificación por número de línea puede servir para cualquier propósito que deseen los usuarios de la base de datos. Por ejemplo, si una serie de registros se ordena por nombre, los registros con el mismo nombre no se ordenan al azar, sino de acuerdo con su clave

principal, que por supuesto es única. También de esta manera, la numeración puede conducir a una clasificación de registros.

La numeración de líneas también es un buen preludio para la combinación de registros individuales en un solo registro. Si se crea una consulta de numeración de líneas como una vista, se puede aplicar una consulta adicional sin crear ningún problema. Como ejemplo simple, aquí una vez más es la primera consulta de numeración con un campo adicional:

```
SELECT "ID", "Name",
   ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <= "a"."ID" ) AS "Nr."
FROM "Table" AS "a"
```

Esta consulta se convierte en la vista *View1*. La consulta se puede usar, por ejemplo, para poner los primeros tres nombres juntos en una línea:

```
SELECT "Name" AS "Name_1",
   (SELECT "Name" FROM "View1" WHERE "Nr."=2) AS "Name_2",
   (SELECT "Name" FROM "View1" WHERE "Nr."=3) AS "Name_3"
FROM "View1" WHERE "Nr."=1
```

De esta forma, se pueden convertir varios registros en campos adyacentes. Esta numeración se ejecuta desde el primero hasta el último registro. Si a todas estas personas se les debe asignar el mismo apellido, esto se puede llevar a cabo de la siguiente manera:

```
SELECT "ID", "Name", "Surname",
   (SELECT COUNT("ID") FROM "Table" WHERE "ID"<="a"."ID" AND "Surname"="a"."Surname")
AS "Nr." FROM "Table" AS "a"
```

Ahora que se ha creado la vista, la familia se puede reunir.

```
SELECT "Surname", "Name" AS "Name_1",
( SELECT "Name" FROM "View1" WHERE "Nr." = 2 AND "Surname" = "a"."Surname") AS
"Name_2",
( SELECT "Name" FROM "View1" WHERE "Nr." = 3 AND "Surname" = "a"."Surname") AS
"Name_3"
FROM "View1" AS "a" WHERE "Nr." = 1
```

Así, en una libreta de direcciones, todos los miembros de una familia (`"Surname"`) se pueden reunir para que cada dirección se tenga en cuenta solo una vez al enviar una carta, pero todos los que deberían recibirla se enumeran.

Hay que tener cuidado aquí, pues no se quiere una función de bucle sin fin. La consulta en el ejemplo anterior limita los registros paralelos que se convertirán en campos a 3. Este límite se eligió deliberadamente. No aparecerán más nombres incluso si el valor de `"Nr."` es mayor que 3. En algunos casos, ese límite es claramente comprensible. Por ejemplo, si estamos creando un calendario, las líneas pueden representar las semanas del año y las columnas los días de la semana. Como en el calendario original solamente la fecha determina el contenido del campo, la numeración de línea se usa para numerar los días de cada semana de forma continua y luego las semanas del año se convierten en registros. Esto requiere que la tabla contenga un campo de fecha con fechas continuas y un campo para los eventos. Además, la fecha más temprana siempre creará un `"Nr." = 1`. Entonces, si desea que el calendario comience el lunes, la fecha más temprana debe ser el lunes. La columna 1 es el lunes, la 2 el martes y así sucesivamente. La subconsulta luego termina en `"Nr." = 7`. De esta forma, los siete días de la semana se pueden mostrar uno al lado del otro y se puede crear una vista de calendario correspondiente.

Obtener un salto de línea a través de una consulta

A veces es útil agrupar varios campos mediante una consulta y separarlos por saltos de línea, por ejemplo, al leer una dirección completa en un informe.

El salto de línea dentro de la consulta está representado por Char (13). Ejemplo:

```
SELECT "Firstname" || ' ' || "Surname" || Char(13) || "Road" || Char(13) || "Town" FROM "Table"
```

Esto produce:

Firstname Surname
Road
Town

Dicha consulta, con una línea que numera hasta 3, le permite imprimir etiquetas de dirección en tres columnas mediante la creación de un informe. La numeración es necesaria a este respecto para que se puedan colocar tres direcciones una al lado de la otra en un registro. Esa es la única forma en que permanecerán uno al lado del otro cuando se lean en el informe.

En algunos sistemas operativos es necesario incluir Char(10) junto con Char (13) en el código.

```
SELECT "Firstname" || ' ' || "Surname" || Char(13) || Char(10) || "Street" || Char(13) ||Char(10) ||
"Town" FROM "Table"
```

Agrupar y resumir

Para otras bases de datos y para versiones nuevas de HSQLDB, el mandato Group_Concat() está disponible. Se puede usar para agrupar campos individuales en un registro en un campo. Entonces, por ejemplo, es posible almacenar nombres y apellidos en una tabla, luego presentar los datos de tal manera que un campo muestre los apellidos como apellidos, mientras que un segundo campo contiene todos los nombres relevantes secuencialmente, separados por comas.

Este ejemplo es similar en muchos aspectos a la numeración de líneas. La agrupación en un campo común es un tipo de complemento a esto.

Surname	***First name***
Müller	Karin
Schneider	Gerd
Müller	Egon
Schneider	Volker
Müller	Monika
Müller	Rita

es convertido por la consulta a:

Surname	***First name***
Müller	Karin, Egon, Monika, Rita
Schneider	Gerd, Volker

Este procedimiento puede, dentro de ciertos límites, expresarse en HSQLDB.

El siguiente ejemplo se refiere a una tabla llamada *Name* con los campos *ID, Firstname* y *Surname*. La siguiente consulta se ejecuta primero en la tabla y se guarda como una vista llamada *View_Group*.

```
SELECT "Surname", "Firstname",
   ( SELECT COUNT( "ID" ) FROM "Name" WHERE "ID" <= "a"."ID"
   AND "Surname" = "a"."Surname" ) AS "GroupNr"
FROM "Name" AS "a"
```

Puede leer en el “Capítulo 5, Consultas” cómo esta consulta accede al contenido del campo en la misma línea de consulta. Produce una secuencia numerada ascendente, agrupada por apellido. Esta numeración es necesaria para la siguiente consulta, de modo que en el ejemplo se enumere un máximo de 5 nombres.

```
SELECT "Surname",
( SELECT "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname" AND "GroupNr"
= 1 ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname"
AND "GroupNr" = 2 ), '' ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname"
AND "GroupNr" = 3 ), '' ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname"
AND "GroupNr" = 4 ), '' ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname"
AND "GroupNr" = 5 ), '' )
AS "Firstnames"
FROM "View_Group" AS "a"
```

Usando subconsultas, los nombres de los miembros del grupo se buscan uno tras otro y se combinan.

A partir de la segunda subconsulta, debe asegurarse de que los valores `'NULL'` no establezcan la combinación completa en `'NULL'`. Es por eso que se muestra un resultado de `''` en lugar de `'NULL'`.

LibreOffice

Guía de Base

Capítulo 9
Macros

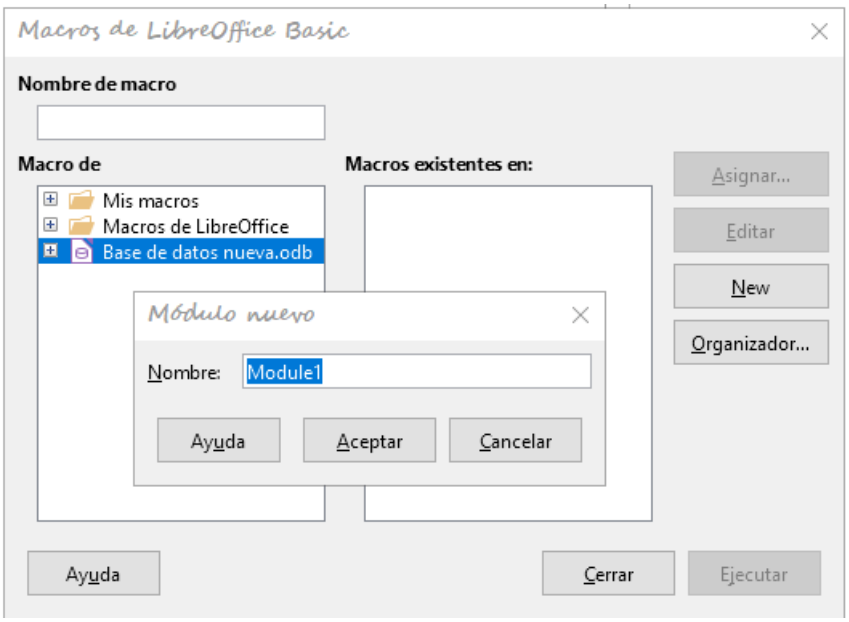
Observaciones generales sobre macros

En principio, una base de datos en Base se puede administrar sin macros. Aunque a veces, pueden ser necesarios para:

- Prevención más efectiva de errores de entrada.
- Simplificar ciertas tareas de procesamiento (cambiar de un formulario a otro, actualizar datos después de ingresarlos en un formulario, etc.).
- Permitir que ciertas órdenes SQL se ejecuten más fácilmente que con el editor SQL.

Debe decidir por sí mismo con qué intensidad desea usar macros en Base. Las macros pueden facilitar el uso, pero siempre están asociadas con pequeñas reducciones en la velocidad de respuesta del programa y, a veces, ralentizan (cuando están mal codificadas). Siempre es mejor comenzar utilizando todas las posibilidades de la base de datos y sus capacidades para configurar los formularios antes de intentar proporcionar una funcionalidad adicional con macros. Las macros siempre deben probarse en bases de datos grandes para determinar su efecto en el rendimiento.

Para la creación y el acceso a las macros se utiliza el diálogo *Macros de LibreOffice Basic* >al que se accede mediante **Herramientas > Macros > Organizar macros > LibreOffice Basic** del menú principal.



Para que las macros estén disponibles en todo momento para la base de datos, en la que está trabajando, es necesario que se guarden en el archivo de la base de datos seleccionándolo en el área de la izquierda (*Macro de*).

Pulse en el botón *Nuevo* en el diálogo y se abrirá el diálogo *Módulo nuevo*, que solicita el nombre del módulo (el contenedor dónde se escribirán los procedimientos).

En cuanto se hace esto, aparece la *interfaz de programación Basic*. En el *área de edición* aparecerá el inicio y el fin para que escriba un procedimiento:

```
REM ***** BASIC *****
Sub Main
...
End Sub
```

Para poder utilizar las macros, en LibreOffice, es necesario seguir los siguientes pasos:

- En **Herramientas > Opciones > Seguridad > Seguridad de macros:** >El *Nivel de seguridad* debe reducirse a *Medio*. También es posible usar la pestaña *Orígenes de*

confianza para elegir certificados de confianza o establecer la ruta a sus archivos de macro y así evitar las ventanas emergentes relativas a la activación de macros al abrir los archivos.

- Después de la creación del primer módulo, debe guardar el archivo, cerrarlo y luego volver a abrirlo.

Algunos principios básicos para el uso del código basic de LibreOffice:

- Las Instrucciones deben terminar con un retorno de línea. Aunque las líneas no están numeradas de forma predeterminada (existe una opción para activar la numeración).
- Las funciones, expresiones reservadas y elementos similares no distinguen entre mayúsculas y minúsculas. Así pues, *String* es lo mismo que *STRING*, *string* o cualquier otra combinación de mayúsculas y minúsculas (el uso de mayúsculas se debe emplear para mejorar la legibilidad). Sin embargo, los nombres de constantes y enumeraciones si distinguen entre mayúsculas y minúsculas la primera vez que los ve el compilador de macros, por lo que es mejor acostumbrarse a escribir siempre de una manera adecuada.
- Se puede distinguir entre *Procedimientos* (que comienzan con `Sub`) y *Funciones* (que comienzan con `Function`). Los procedimientos fueron originalmente trozos de programa sin un valor de retorno, mientras que las funciones devuelven valores que se pueden procesar posteriormente. Pero esta distinción se está volviendo cada vez más irrelevante. Hoy en día, las personas usan términos como *método* o *rutina* para designar a los procedimientos, tanto si hay un valor de retorno o no. Un procedimiento también puede tener un valor de retorno (distinto del tipo `Variant`).

```
Sub miProcedimiento As Integer
…
End Sub
```

Para obtener información más detallada, consulte el «Capítulo 13, Introducción a las macros», en la *Guía de primeros pasos con Base*.

Nota

Las macros en las versiones PDF y ODT de este capítulo están coloreadas de acuerdo con las reglas del editor de macros LibreOffice:

```
Comentario
Identificador
Expresión reservada
Cadena de texto
Número
Operador
```

Macros en Base

Usar macros

Aunque es posible utilizar la forma directa para ejecutar las macros, usando **Herramientas > Macros > Ejecutar macro**, esto no es habitual para las macros que se usan en Base. Normalmente la macro se asigna a un suceso y esta se inicia cuando se produce el desencadenante.

Las macros se usan para:

- Gestionar los sucesos en formularios
- Editar datos dentro de un formulario

- Cambiar entre controles de formulario
- Reaccionar ante una acción del usuario dentro de un control.

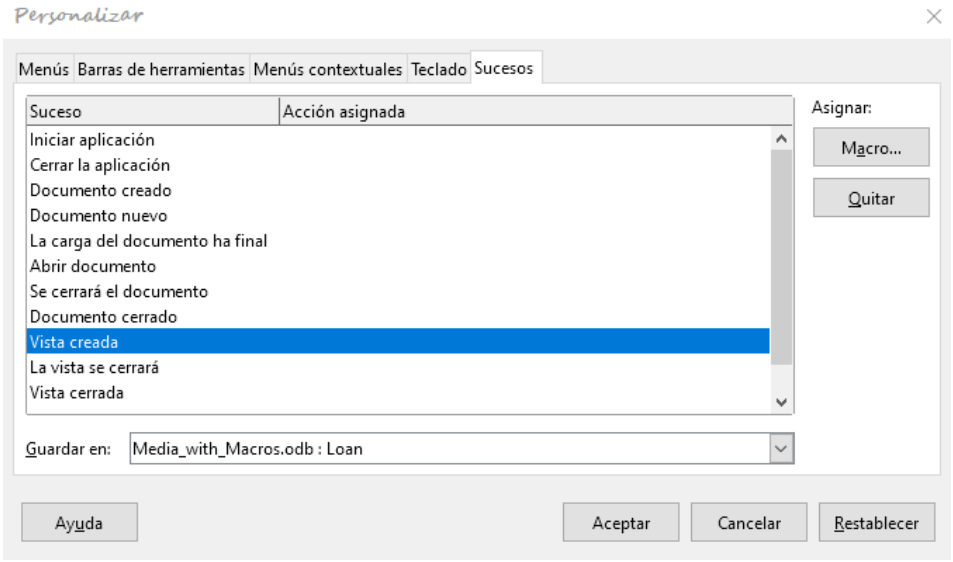
Cuando se utiliza alguno de los objetos de `ThisComponent` u `oEvent` no es posible utilizar la forma directa, ni siquiera para pruebas, (consulte «Acceso a formularios» en la página 348 y también «Acceso a elementos de formulario» en la página 349).

Asignar macros

Si se quiere iniciar una macro, respondiendo a un suceso, primero se tiene que definir esa macro, de manera que se pueda asignar a ese suceso. Se puede acceder a los sucesos a través de dos ubicaciones, en función de su desencadenante:

Sucesos que se desencadenan en un formulario cuando la ventana se abre o se cierra

Las acciones que tienen lugar cuando se abre o cierra un formulario se registran de la siguiente manera:



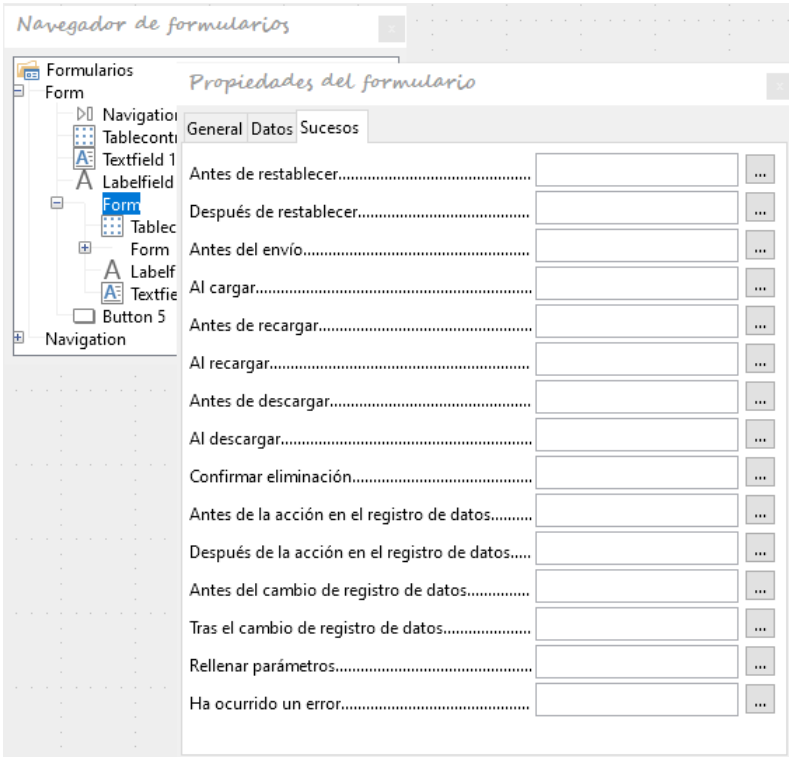
5) Durante la creación de un formulario, utilice el menú **Herramientas > Personalizar** y elija la pestaña *Sucesos*.
6) Elija el suceso apropiado. Algunas macros solo se pueden iniciar cuando se elige el suceso *Vista creada*. Otras macros, por ejemplo para crear un formulario en pantalla completa se deben iniciar con el suceso *Abrir documento*.
7) Use el botón *Macro* para buscar la macro que desea aplicar y confirme su elección.
8) En la parte inferior *Guardar en*, indique el nombre del formulario.
9) Finalmente, confirme con *Aceptar*.

Sucesos dentro de una ventana de un formulario

Una vez abierta la ventana para editar el contenido general del formulario, se puede acceder a los elementos individuales del mismo. Incluidos los elementos que haya asignado al formulario.

Se puede acceder a los elementos del formulario utilizando el *Navegador de formularios*, como se muestra en la siguiente ilustración, o también mediante el menú contextual de los controles dentro de la interfaz del formulario.

En la pestaña de *Sucesos* de *Propiedades del formulario*, los Sucesos enumerados tienen lugar mientras la ventana del formulario está abierta. Se puede configurar por separado para cada formulario o subformulario durante la edición del formulario.



Nota

Desafortunadamente, Base usa la palabra formulario tanto para una ventana que se abre para la entrada de datos, como para los algunos elementos dentro de esta ventana que están vinculados a un origen de datos específico (tabla o consulta).

Una sola ventana de formulario puede contener varios subformularios con diferentes fuentes de datos. En el *Navegador de formularios*, siempre se visualiza primero el término *Formularios* con diversas entradas de formularios y controles. En el caso de un formulario simple contiene solo una entrada subordinada.

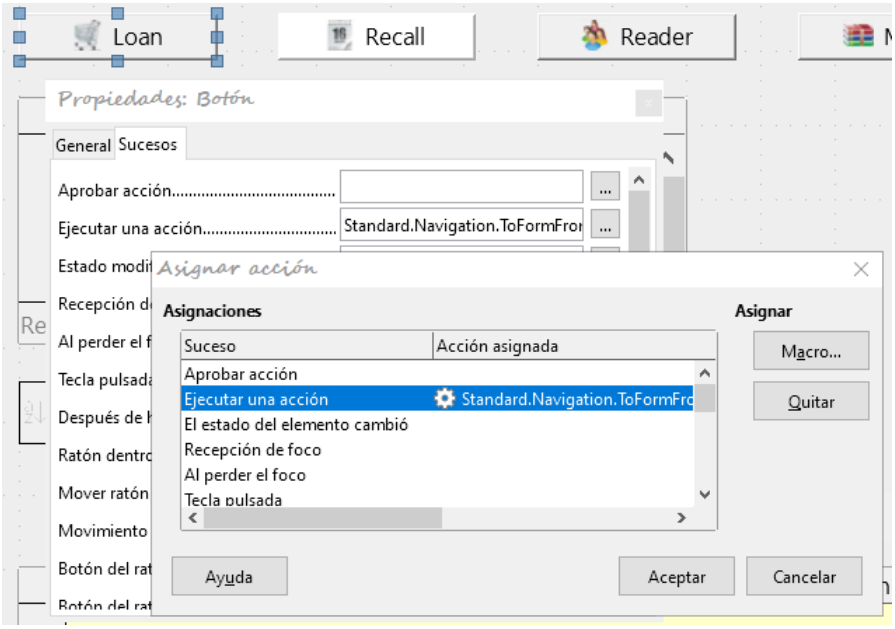
Sucesos dentro de un formulario

Todas las demás macros se registran utilizando las propiedades de subformularios y controles a través de la pestaña Sucesos.

1) Abra la ventana para las propiedades del control.
2) Elija un suceso adecuado en la pestaña Sucesos.
3) Para los encargados de editar la fuente de datos, use sucesos que se refieran a *Grabar*, *Actualizar*, o *Restablecer*.

   Para los botones, o las opciones dentro de los campos de lista u opción, utilice *Ejecutar una acción*.

   Todos los demás sucesos dependen del tipo de control y la acción deseada.
4) Haga clic en el botón (…) a la derecha del suceso para abrir el diálogo *Asignar acción*.
5) Haga clic en el botón *Macro* para elegir la macro específica para esa acción.
6) Haga clic en *Aceptar* para confirmar la asignación.



Componentes de macros

Esta sección explica parte del lenguaje Basic para macros que se usa comúnmente en Base, especialmente dentro de los formularios. En las siguientes secciones se muestran varios ejemplos en la medida de lo posible (y razonable).

La estructura de una macro

La definición de un procedimiento comienza con su tipo `Sub` o `Function` y termina con `End Sub` o `End Function` respectivamente. Una macro asignada a un suceso puede recibir argumentos (valores); el único útil es el argumento `oEvent`. Todos los procedimientos a los que puede llamar una macro pueden definirse con o sin un valor de retorno, dependiendo de su propósito, y estar provista de argumentos si es necesario.

```
Sub update_loan
End Sub

Sub from_Form_to_Form(oEvent As Object)
End Sub

Function confirm_delete(oEvent As Object) As Boolean
   confirm_delete = False
End Function
```

Es útil empezar con este marco (inicio y final) e incluir el contenido después. No olvide agregar comentarios para explicar la macro, recordando la regla: «*Tantos como sea necesario, tan pocos como sea posible*».

Basic no distingue entre mayúsculas y minúsculas salvo en casos especiales. Por lo general, los términos fijos como `SUB` se escriben preferiblemente en mayúsculas o mayúscula inicial, otros conceptos con mayúsculas y minúsculas.

Uso de variables

En el siguiente paso, al comienzo de la rutina, la instrucción `Dim` se usa para definir las variables que se utilizarán en el procedimiento, cada una con su tipo de datos apropiado. Aunque la programación en Basic no necesita las definiciones (acepta cualquier nueva variable que se cree).

El código de la macro es más seguro si se declaran las variables, y especialmente si se indica el tipo de datos.

Muchos programadores utilizan `option Explicit` al principio de un módulo. Con esta opción se aseguran que todas las variables tengan que estar declaradas de antemano, si una variable no se ha declarado el programa mostrará un error y detendrá su ejecución.

```
Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm As Object
Dim sName As String
Dim bOKEnabled As Boolean
Dim iCounter As Integer
Dim dBirthday As Date
```

Los nombres de las variables deben empezar siempre por un carácter alfabético, solo se pueden usar caracteres alfabéticos (A-Z o a-z), números y el carácter de subrayado (_). No se permiten caracteres especiales y aunque se permiten espacios bajo determinadas condiciones, es mejor evitarlos.

Una práctica común y muy recomendable, es utilizar el primer carácter para especificar el tipo de datos[1]. De esta manera se distingue claramente el tipo de datos que contiene la variable en cualquier lugar que aparezca, evitando errores al procesarla. También se recomienda usar nombres explicativos, para que el contenido de la variable sea obvio.

Puede encontrar una lista de tipos de datos admisibles por *Star Basic* en el «Apéndice A» de esta guía. Difieren de algún modo de los tipos propios de la base de datos y de la API de LibreOffice. Estos cambios se verán más claros en los ejemplos

Uso de matrices

Para bases de datos en particular, el agrupamiento de varias variables en un registro es importante. Una matriz es un conjunto de varias variables que se almacenan juntas en una única ubicación común. Se debe definir una matriz antes de que se puedan escribir datos en ella.

```
Dim arData()
```

Crea una matriz vacía.

```
arData = Array("Lisa","Schmidt")
```

Crea una matriz de un tamaño específico (2 elementos) y le proporciona valores.

```
Print arData(0), arData(1)
```

Muestra los dos elementos asignados en pantalla. El recuento de elementos comienza con 0.

```
Dim arData(2)
arData(0) = "Lisa"
arData(1) = "Schmidt"
arData(2) = "Cologne"
```

Crea una matriz en la que se pueden almacenar tres elementos de cualquier tipo, por ejemplo, un registro que contiene `"Lisa" "Schmidt" "Cologne"`. No puede poner más de tres elementos en esta matriz.

1 Algunas veces, una sóla letra no le permite distinguir entre los tipos de datos (Double y Data o "Single" y String). Utilice las necesarias para evitar confusiones.

Si desea almacenar más elementos, debe agrandar la matriz con `ReDim` . Sin embargo, si el tamaño de una matriz se redefine mientras se ejecuta una macro, la matriz queda vacía como si fuera una nueva matriz.

```
ReDim Preserve arData(3)
arData(3) = "18.07.2003"
```

Al agregar `Preserve` se conservan los datos anteriores y la matriz se extenderá agregando un cuarto elemento : la fecha (en este caso en forma de texto).

La matriz que se muestra arriba solo puede almacenar un registro. Si se desea almacenar varios registros, como lo hace una tabla, hay que definir una matriz bidimensional.

```
Dim arData(2,1) '3 elementos, en cada fila(2 filas) >
arData(0,0) = "Lisa" 'primer elemento de la primera fila
arData(1,0) = "Schmidt" 'segundo elemento de la primera fila
arData(2,0) = "Cologne"
arData(0,1) = "Egon" 'primer elemento de la segunda fila
arData(1,1) = "Müller"
arData(2,1) = "Hamburg"
```

Aquí también es posible extender la matriz previamente definida y preservar los contenidos existentes utilizando `Preserve`.

Acceso a formularios

El formulario se encuentra en el documento actualmente activo. La región que se representa se llama *drawpage*. El contenedor en el que se guardan todos los formularios se llama *forms*; en el *Navegador de formularios*, se muestra como el encabezado principal con todos los formularios individuales adjuntos. Las variables mencionadas anteriormente reciben sus valores de esta manera:

```
oDoc = thisComponent 'acceso al documento
oDrawpage = oDoc.drawpage 'acceso a la región de formularios
oForms = oDrawpage.forms 'acceso a los formularios
oForm = oForms.getByName("Filter") 'acceso al formulario filter
```

Todo esto se puede resumir en una sola variable:

```
oForm = ThisComponent.drawpage.forms.getByName("Filter")
```

El formulario al que se accede en esta ocasión se llama *Filter*. Este es el nombre que está visible en el nivel superior del Navegador de formularios (de forma predeterminada, el primer formulario se llama *MainForm*).

Los subformularios se encuentran en orden jerárquico dentro del formulario principal y se puede llegar paso a paso:

```
Dim oSubForm As Object
Dim oSubSubForm As Object
oSubForm = oForm.getByName("Readerselect")
oSubSubForm = oSubForm.getByName("Readerdisplay")
```

En lugar de usar valores intermedios, puede ir directamente a un formulario concreto. Un objeto intermedio, que se puede usar más de una vez, debe declararse y asignarle un valor separado. En el siguiente ejemplo, *oSubForm* ya no se usa.

```
oForm = thisComponent.drawpage.forms.getByName("Filter")
oSubSubForm =
```

```
oForm.getByName("readerselect").getByName("readerdisplay")
```

Nota

Si un nombre de formulario contiene únicamente letras `ASCII` y números sin espacios ni caracteres especiales, se puede usar directamente en asignación a una variable.

```
oForm = thisComponent.drawpage.forms.Filter
oSubSubForm = oForm.readerselect.readerdisplay
```

Contrariamente al uso normal en Basic, los nombres de formularios ,y de los controles son sensibles al uso de mayúsculas.

El suceso que desencadena la macro proporciona un modo diferente de acceso al formulario. Si se inicia una macro desde un suceso de formulario, como por ejemplo *Antes de la acción de registro*, se puede acceder al formulario utilizando el método *oEvent* de la siguiente manera:

```
Sub MacroexampleCalc(oEvent As Object)
  oForm = oEvent.Source
  ...
End Sub
```

Si la macro se inicia desde un suceso en un control de formulario, como por ejemplo en un *Cuadro de texto* , *Al perder el foco*, tanto el formulario como el campo serán accesibles:

```
Sub MacroexampleCalc(oEvent As Object)
  oField = oEvent.Source.Model
  oForm = oField.Parent
  ...
End Sub
```

El acceso a los sucesos tiene la ventaja de que no necesita preocuparse si se trata de un formulario principal o un subformulario. Además, el nombre del formulario no es importante para el funcionamiento de la macro.

Acceso a elementos de formulario

Se accede a los elementos dentro de los formularios de manera similar: declare una variable como objeto y asigne el control apropiado dentro del formulario:

```
  Dim btnOK As Object 'se declara la variable para el botón OK
  btnOK = oSubSubForm.getByName("button 1") ' pertenece al
formulario readerdisplay
```

Este método siempre funciona cuando se sabe con qué elemento debe iniciarse la macro. Sin embargo, cuando se debe determinar qué suceso inició la macro, el método `oEvent` mencionado anteriormente es más útil.

La variable se declara en la definición de la macro y se le asigna un valor cuando se inicia la macro. La propiedad `Source` siempre devuelve el elemento que lanzó la macro, mientras que la propiedad `Model` describe el control en detalle:

```
Sub confirm_choice(oEvent As Object)
  Dim btnOK As Object
  btnOK = oEvent.Source.Model
End Sub
```

Si se desea, se pueden realizar más acciones con el objeto obtenido por este método.Tenga en cuenta que los subformularios cuentan como componentes de un formulario.

Acceso a la base de datos

Normalmente, el acceso a la base de datos está controlado por *formularios*, *consultas*, *informes* o la *combinación de correspondencia*, como se describe en los capítulos anteriores. Si estas posibilidades resultan insuficientes, una macro puede acceder específicamente a la base de datos de varias maneras.

Conectar con la base de datos

El método más sencillo usa la misma conexión que el formulario. *oForm* se define como se indicó antes.

```
Dim oConnection As Object
oConnection = oForm.activeConnection()
```

O se puede buscar la fuente de datos (es decir, la base de datos) a través del documento y usar su conexión existente en la macro.

```
Dim oDatasource As Object
Dim oConnection As Object
oDatasource = thisComponent.Parent.dataSource
oConnection = oDatasource.getConnection("","")
```

Una forma adicional que permite que la conexión se cree sobre la marcha sería:

```
Dim oDatasource As Object
Dim oConnection As Object
oDatasource = thisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then oDatasource.connect()
oConnection = oDatasource.ActiveConnection()
```

Aquí la condición `If` utiliza solo una línea, por lo que no se necesita acabarla con `End If`.

Si la macro se va a iniciar a través de la interfaz de usuario y no desde un suceso en un formulario, se puede usar la siguiente variante:

```
Dim oDatasource As Object
Dim oConnection As Object
oDatasource = thisDatabaseDocument.CurrentController
If Not (oDatasource.isConnected()) Then oDatasource.connect()
oConnection = oDatasource.ActiveConnection()
```

Se puede acceder a otras bases de datos que estén registradas en LibreOffice de la siguiente manera:

```
Dim oDatabaseContext As Object
Dim oDatasource As Object
Dim oConnection As Object
oDatabaseContext =
createUnoService("com.sun.star.sdb.DatabaseContext")
oDatasource = oDatabaseContext.getByName("nombre registrado de la
Base de datos")
oConnection = oDatasource.GetConnection("","")
```

También son posibles las conexiones a bases de datos no registradas con LibreOffice. En estos casos, en lugar del nombre registrado, se debe proporcionar la ruta y nombre de la base de datos como archivo: `///……/database.odb`.

Pueden verse más explicaciones sobre las conexiones de la base de datos en «Crear una conexión a una base de datos» en la página 409.

Órdenes SQL

Para trabajar con la base de datos se utilizan órdenes SQL. Estas órdenes deben ser creadas y enviadas a la base de datos; (el resultado se determina según el tipo de orden) los resultados obtenidos se pueden seguir procesando. La directiva `createStatement` crea un objeto adecuado para este propósito.

```
Dim oSQL_Statement As Object ' el objeto que llevará a cabo la orden
SQL
Dim stSql As String ' Texto de la actual orden SQL
Dim oResult As Object ' resultado de executeQuery
Dim iResult As Integer ' resultado de executeUpdate
oSQL_Statement = oConnection.createStatement()
```

Para consultar datos, utilice el método `executeQuery`. Los nombres de tablas y campos suelen estar entre comillas dobles. La macro debe enmascararlos con comillas dobles adicionales para garantizar que aparezcan en la orden.

```
stSql = "SELECT * FROM ""Table1"""
oResult = oSQL_Statement.executeQuery(stSql)
```

Para modificar datos, es decir, `INSERT`, `UPDATE` o `DELETE`, o para influir en la estructura de la base de datos, llame al método `executeUpdate`. Dependiendo de la orden y la base de datos, esto no produce nada útil, o un cero, o el número de registros modificados.

```
stSql = "DROP TABLE ""Suchtmp"" IF EXISTS"
iResult = oSQL_Statement.executeUpdate(stSql)
```

En aras de la exhaustividad, hay un caso especial más que se debe mencionar: el método `execute` que es util si la directiva `createStatement` se va a utilizar de diferentes maneras para `SELECT` o para otros fines, pero no lo usaremos aquí. Para obtener más información, consulte la Referencia de la API.

Órdenes SQL previamente preparadas con parámetros

En todos los casos en que la entrada manual de un usuario debe transferirse a una instrucción SQL, es más fácil y seguro no crear la orden como una cadena de caracteres porque puede ser muy larga, sino prepararla de antemano y usarla con parámetros. Esto facilita el formateo de números, fechas y cadenas, se evitan las repeticiones de las comillas dobles y evita que una frase mal formada provoque la pérdida de datos.

Para usar este método, se crea y se prepara un objeto para una orden SQL determinada :

```
Dim oSQL_Statement As Object ' el objeto, para ejecutar la orden SQL
Dim stSql As String ' Texto de la orden SQL
stSql = "UPDATE author " _
 & "SET lastname = ?, firstname = ?" _
 & "WHERE ID = ?"
oSQL_Statement = oConnection.prepareStatement(stSql)
```

El objeto se crea con `prepareStatement` para que la orden SQL se conozca de antemano. Cada signo de interrogación indica una posición que, antes de ejecutar la orden, recibirá un valor

real. Debido a que la orden se prepara por adelantado, la base de datos sabe cual es el tipo de entrada, en este caso, se esperan dos cadenas de texto y un número. Las distintas entradas se distinguen por el número de su posición (contando desde 1).

Luego, los valores se transfieren con declaraciones adecuadas y se ejecuta la orden SQL. Los valores se toman de los controles de formulario, pero también pueden originarse en otras macros o darse como texto sin formato:

```
oSQL_Statement.setString(1, oTextfeld1.Text) ' Texto para lastname
oSQL_Statement.setString(2, oTextfeld2.Text) ' Texto para first name
oSQL_Statement.setLong(3, oNumericfield1.Value) ' valor para el ID
iResult = oSQL_Statement.executeUpdate
```

La lista completa de métodos está en “Parámetros para órdenes SQL preparadas" (página 369).

Para obtener más información sobre las ventajas de este método, consulte los siguientes enlaces externos:

- SQL-Injection (Wikipedia)
- Inyección SQL
- Why use PreparedStatement (Java JDBC)
- SQL-commands (Introduction to SQL)
- MySQL Gestión Lenguajes de consulta y extracción de datos

Leer y usar registros

Existen varias formas de transferir información de una base de datos a una macro para que esta información se pueda procesar.

Tenga en cuenta que las referencias a un formulario incluyen subformularios. se entiende que ese formulario o parte del formulario está vinculado a una fuente de datos en particular.

Usar formularios

El registro actual y sus datos siempre están disponibles a través del formulario que muestra los datos relevantes (*tabla*, *consulta*, *SELECT*). Existen varios métodos para obtener los datos (*getdata_ type*), en este ejemplo:

```
Dim ID As Long
Dim sName As String
Dim dValue AS Currency
Dim dEntry As New com.sun.star.util.Date
ID = oForm.getLong(1)
sName = oForm.getString(2)
dValue = oForm.getDouble(4)
dEntry = oForm.getDate(7)
```

Todos estos métodos requieren el número de columna de un origen de datos; su cuenta comienza en 1.

Nota

Para todos los métodos que funcionan con bases de datos, se empiezan a contar desde 1. Esto se aplica tanto a las columnas como a las filas.

Si se prefiere usar nombres de columna en lugar de números para trabajar con la fuente de datos (*tabla*, *consulta*, *vista*), el número de columna se puede sustituir por su nombre, usando `findColumn`. Un ejemplo para utilizar la columna llamada *Name*:

```
Dim sName As String
nName = oForm.findColumn("Name") 'accede a la columna Name
sName = oForm.getString(nName) ' obtiene el contenido (texto) de la columna
```

El tipo de valor devuelto siempre coincide con el tipo de método, pero deben tenerse en cuenta los siguientes casos especiales:

- No existen métodos para los datos de los tipos `Decimal`, `Currency`, etc. que se utilizan para cálculos comerciales exactos. Como Basic realiza automáticamente la conversión adecuada, puede usar `getDouble`.
- Al usar `getBoolean`, debe tener en cuenta que `TRUE` y `FALSE` se definen en la base de datos. Las definiciones habituales (valores lógicos: 1 = TRUE) se procesan correctamente.
- Los valores de fecha se pueden definir no solo con el tipo de datos `Date`, sino también (como se indicó anteriormente) como `util.Date`. Esto facilita la lectura y modificación del año, mes y día.
- Con números enteros, hay que tener cuidado con los diferentes tipos de datos. El ejemplo anterior usa `getLong`; La clave primaria *>ID* en la variable Basic también debe tener el tipo de datos `Long`, que coincide con el tipo `Integer` en la base de datos.

La lista completa de estos métodos se encuentra en “Ediciónde filas de datos (registros)" (página 366).

Consejo

Si los valores de un formulario se van a utilizar directamente para su posterior procesamiento en SQL (por ejemplo, para ingresar en otra tabla), es mucho más simple no tener que consultar el tipo de campo.

La siguiente macro, que está vinculada a las propiedades del Botón **Sucesos > Ejecutar una acción**, lee el primer campo en el formulario independientemente del tipo necesario para el procesamiento futuro en Basic.

```
SUB ReadValues(oEvent As Object)
 Dim oForm As Object
 Dim stFeld1 As String
 oForm = oEvent.Source.Model.Parent
 stFeld1 = oForm.getString(1)
End Sub
```

Si los campos se leen usando `getString()`, se conserva todo el formato necesario para el procesamiento posterior de SQL. Una fecha que se muestra como 08.03.19 se lee en el formato 2019-03-08 y se puede usar directamente en SQL.

Leer en un formato correspondiente al tipo de datos solo es obligatorio si el valor se va a procesar posteriormente dentro de la macro, por ejemplo, en un cálculo.

Resultado de una consulta

El conjunto de resultados de una consulta se puede usar de la misma manera. En la Sección de «Órdenes SQL» en la página 351, encontrará la variable `oResult` para este conjunto de resultados, que en general se lee de la siguiente manera:

```
While oResult.next ' bucle para obtener un registro tras otro
  REM transferir el resultado a variables
  stVar = oResult.getString(1)
  inVar = oResult.getLong(2)
```

```
    boVar = oResult.getBoolean(3)
    REM hacer algo con estos valores
  Wend ' fin del bucle
```

Según el tipo de orden SQL, el resultado esperado y su propósito, el ciclo `WHILE` se puede acortar o eliminar por completo. Pero básicamente un conjunto de resultados siempre se puede evaluar de esta manera:

Si solo se va a evaluar el primer registro, con

```
oResult.next
```

se accede a la fila para este registro, y con .

```
stVar = oResult.getString(1)
```

se lee el contenido del primer campo. El ciclo termina aquí.

La consulta para el ejemplo anterior tiene texto en la primera columna, un número entero en la segunda (`Integer` en la base de datos corresponde a `Long` en Basic) y un campo booleano (Sí/No) en la tercera. Se accede a los campos a través de un índice de campo que, a diferencia de un índice de matriz, comienza en 1.

Navegar a partir de tal resultado no es posible. Solo se permiten pasos individuales para el siguiente registro. Para navegar dentro del registro, se debe conocer `ResultSetType` cuando se crea la consulta. A esto se accede usando:

```
oSQL_Result.ResultSetType = 1004 o bien SQL_Result.ResultSetType =
1005
```

El Tipo `1004 – SCROLL_INTENSIVE` permite navegar libremente, pero no detecta cambios en los datos originales. El Tipo `1005 – SCROLL_SENSITIVE` reconoce los cambios en los datos originales que pueden afectar el resultado de la consulta.

El número total de filas en el conjunto de resultados se puede determinar solo después de que se haya especificado un tipo numérico para el resultado. Se lleva a cabo de la siguiente manera:

```
Dim iResult As Long
If oResult.last ' va al último registro si es posible
  iResult = oResult.getRow ' el número resultante es el total de
filas
Else
  iResult = 0
End If
```

Uso de un control

Si un control está vinculado a una fuente de datos, el valor se puede leer directamente, como se describe en la siguiente sección. Sin embargo, esto puede conducir a problemas. Es más seguro usar el procedimiento descrito en «Usar formularios» (página 352) o bien el siguiente método, que se muestra para varios tipos diferentes de control:

```
sValue = oTextField.BoundField.Text ' ejemplo para un cuadro de
texto
nValue = oNumericField.BoundField.Value ' ejemplo para un campo
numérico
dValue = oDateField.BoundField.Date ' ejemplo para un campo Fecha
```

`BoundField` representa el enlace entre el control visible y el contenido del conjunto de datos.

Navegar en un conjunto de datos

En el último ejemplo, el método `Next` se utilizó para pasar de una registro del conjunto de resultados al siguiente. Existen otros métodos y pruebas similares que se pueden usar tanto para los datos en un formulario, representado por la variable `oForm`, como para un conjunto de resultados.

Por ejemplo, usando el método descrito en “Actualización automática de formularios” (página 369), el registro anterior se puede volver a seleccionar:

```
Dim loRow As Long
loRow = oForm.getRow() ' guarda el número del registro
oForm.reload() ' recarga el conjunto de registros
oForm.absolute(loRow) ' vuelve al mismo registro
```

La sección “Actualización automática de formularios” muestra los métodos adecuados para ello.

Nota

Desafortunadamente, desde el comienzo de LibreOffice, ha habido un error (transferido desde OpenOffice) que afecta los formularios: establece a "0" el número de registro en uso cuando los datos se alteran dentro de un formulario.Vea: https://bugs.documentfoundation.org/show_bug.cgi?id=82591. Para obtener el número de fila actual correcto, vincule la siguiente macro al las propiedades del Formulario: **Sucesos > Después del cambio de registro**.

```
Global loRow As Long
Sub RowCounter(oEvent As Object)
 loRow = oEvent.Source.Row
End Sub
```

El nuevo número de fila se lee y se asigna a la variable global `loRow`. Esta variable se debe colocar en todos los módulos, al comienzo de todos los procedimientos y retendrá su contenido hasta que salga de Base o cambie el valor llamando a `RowCounter` de nuevo.

Editar registros: agregar, modificar, eliminar

Para editar registros, tienen que cumplirse conjuntamente varias condiciones:

- El usuario tiene que ingresar la información en un control, utilizando el teclado.
- El conjunto de datos contenidos en el formulario tiene que ser informado sobre el cambio. Sucede cuando el foco se mueve de un campo a otro.
- La base de datos en sí tiene que ser modificada. Sucede cuando se mueve un registro a otro.

Cuando se hace a través de una macro, todos estos pasos parciales deben ser rigurosos. Si falta alguno de ellos o se lleva a cabo incorrectamente, los cambios se perderán y no se reflejarán en la base de datos. En primer lugar, el cambio no tiene por qué estar en el valor mostrado del control sino en el conjunto de datos en sí. Esto hace que no tenga sentido cambiar la propiedad `Text` de un control.

Tenga en cuenta que las tablas son los únicos conjuntos de datos que se pueden modificar sin causar problemas. Para otros conjuntos de datos (consultas, por ejemplo), la edición solo es posible bajo circunstancias especiales.

Cambiar el contenido de un control

Si desea cambiar un solo valor, la propiedad `BoundField` del control se puede usar con un método apropiado. Posteriormente, el cambio tiene que transmitirse a la base de datos.

Aquí hay un ejemplo para un campo de fecha en el que se debe ingresar la fecha actual:

```
Dim unoDate As New com.sun.star.util.Date
unoDate.Year = Year(Date)
unoDate.Month = Month(Date)
unoDate.Day = Day(Date)
oDateField.BoundField.updateDate( unoDate )
oForm.updateRow() ' el cambio se transmite a la base de datos
```

Para `BoundField`, se debe utilizar el método `updateXxx` que coincida con el tipo de datos del campo. En este ejemplo, el campo es un campo `Date`. El nuevo valor se pasa como argumento, en este caso la fecha actual, convertida al formato que requiere la macro.

Alterar registros en un archivo de datos

El método anterior no es adecuado cuando es necesario cambiar varios valores en un registro. Por un lado, tendría que existir un control en el formulario para cada campo, lo que a menudo no es útil o no se desea. Además, se tiene que asignar un objeto para cada campo. La forma simple y directa usa un formulario de esta manera:

```
Dim unoDate As New com.sun.star.util.Date
unoDate.Year = Year(Date)
unoDate.Month = Month(Date)
unoDate.Day = Day(Date)
oForm.updateDate(3, unoDate )
oForm.updateString(4, "un Texto")
oForm.updateDouble(6, 3.14)
oForm.updateInt(7, 16)
oForm.updateRow()
```

Para cada columna del conjunto de datos, se llama al método `updateXxx` apropiado para su tipo. Los argumentos son el número de columna (contando desde 1) y el valor deseado. Luego, las modificaciones se tienen que pasar a la base de datos.

Crear, modificar y eliminar registros

Los cambios aquí nombrados se refieren al registro en uso del conjunto de datos subyacente al formulario. En algunos casos, es necesario llamar a un método desde «Navegaciónr en los conjuntos de datos» (página 365). Son necesarios los siguientes pasos:

1) Elegir el registro deseado.
2) Cambiar los valores como se describe en la sección anterior.
3) Confirmar el cambio del registro con la siguiente orden:

```
oForm.updateRow()
```

4) En casos especiales, es posible cancelar y volver al estado anterior:

```
oForm.cancelRowUpdates()
```

Para un nuevo registro hay un método especial, comparable con el cambio a una nueva fila en un control de tabla. Se hace de la siguiente manera:

1) Preparar para crear un nuevo registro:

```
oForm.moveToInsertRow()
```

2) Ingresar todos los valores deseados/necesarios usando los métodos `updateXxx` como se indica en la sección anterior.

3) Confirmar la creación de un nuevo registro con los nuevos datos con la siguiente orden:

```
oForm.insertRow()
```

4) La nueva entrada no se puede revertir fácilmente. En lugar de ello, tendrá que eliminarse el nuevo registro.

Hay una orden simple para eliminar un registro:

1) Elegir el registro deseado y actualizarlo, como se hace para una modificación.

2) Usar la siguiente orden para eliminarlo:

```
oForm.deleteRow()
```

Consejo

Para garantizar que los cambios se transfieran a la base de datos, tienen que confirmarse explícitamente con `updateRow` o `insertRow` según corresponda. Mientras se presiona el botón *Guardar*, se usará automáticamente la función apropiada, con una macro tiene que determinar antes de guardar si el registro es nuevo (`Insert`) o es una modificación de uno existente (`Update`).

```
If oForm.isNew Then
 oForm.insertRow()
Else
 oForm.updateRow()
End If
```

Prueba y cambio de controles

Además del contenido de un conjunto de datos, se puede leer editar y modificar mucha más información en un control,. Esto es particularmente cierto para las propiedades, como se describe en el «Capítulo 4, Formularios».

En «Mejorar la facilidad de uso» (página 369) encontrará varios ejemplos que usan la información adicional del control:

```
Dim stTag As String
stTag = oEvent.Source.Model.Tag
```

Como se mencionó en la sección anterior, la propiedad `Text` solo se puede modificar de manera útil si el control no está vinculado a un conjunto de datos. Sin embargo, hay otras propiedades que se establecen como parte de la definición del formulario, pero que se pueden adaptar en tiempo de ejecución.

Por ejemplo, una etiqueta podría recibir un color de texto diferente si tiene que representar una advertencia en lugar de una información:

```
Sub showWarning(oField As Object, iType As Integer)
  Select Case iType
    Case 1
      oField.TextColor = RGB(0,0,255) ' 1 = azul
    Case 2
      oField.TextColor = RGB(255,0,0) ' 2 = rojo
    Case Else
      oField.TextColor = RGB(0,255,0) ' 0 = verde (ni 1 ni 2)
  End Select
```

```
End Sub
```

Nombres de las propiedades de los controles en macros

Mientras que el diseñador de un formulario puede usar las designaciones en su idioma nativo para las propiedades y acceso a datos, en Basic solo se pueden usar términos en inglés. Se exponen en la siguiente sinopsis.

Las propiedades que normalmente solo se establecen en la definición del formulario no se incluyen aquí. Tampoco los métodos (funciones y/o procedimientos) que rara vez se usan o solo se requieren para declaraciones más complejas.

La sinopsis incluye lo siguiente:

Nombre	Nombre que se utilizará para la propiedad en el código de la macro
Tipo	Tipo de datos de Basic. Para funciones, el tipo de retorno.No incluido para procedimientos
R/W	Indica cómo puede usarse el valor • R solo lectura • W >solo escritura (modificar) • (R) Lectura posible, inadecuada para procesamiento posterior • (W) Escritura posible pero no útil • R+W Adecuado para lectura y escritura

Se puede encontrar más información en la «Referencia de la interfaz de programación (API)» buscando el nombre del control en inglés. También puede utilizar una herramienta bastante útil llamada *XrayTool* que muestra las propiedades y métodos disponibles para un objeto. Puede obtenerla en el enlace externo: https://berma.pagesperso-orange.fr/index2.html

```
Sub Main(oEvent)
  Xray(oEvent)
End Sub
```

Con este procedimiento se inicia la macro *XrayTool* para el argumento (objeto que se quiera inspeccionar, en este caso *oEvent*) .

Propiedades de formularios y controles.

El modelo de un control describe sus propiedades. Según la situación, se puede acceder al valor de una propiedad de solo lectura o de solo escritura. El orden sigue al de las listas de «Propiedades de los campos de control» en el «Capítulo 4, Formularios».

Además de las propiedades genéricas que se indican al principio, se enumeran las propiedades específicas para cada tipo de control.

Fuente (Font)

En cada control que muestra texto, las propiedades de fuente se pueden personalizar.

Nombre	***Tipo***	***R/W***	***Propiedad***
FontName	string	R+W	Nombre de la fuente
FontHeight	single	R+W	Tamaño de la fuente
FontWeight	single	R+W	Negrita o normal
FontSlant	integer	R+W	Itálica o cursiva
FontUnderline	integer	R+W	Subrayada
FontStrikeout	integer	R+W	Tachada

Formulario (Form)

Nombre	*Tipo*	*R/W*	*Propiedad*
ApplyFilter	boolean	R+W	Filtro aplicado
Filter	string	R+W	Filtro para el registro
FetchSize	long	R+W	Número de registros cargados a la vez
Row	long	R	Número de fila (registro) en uso
RowCount	long	R	Número de filas (registros)

Propiedades que se aplican a todos los controles (Control)

Consulte también *FormComponent*

Nombre	*Tipo*	*R/W*	*Propiedad*
Name	string	R+(W)	Nombre del control
Enabled	boolean	R+W	Activo (el control se puede seleccionar)
EnableVisible	boolean	R+W	Mostrar el control
ReadOnly	boolean	R+W	El contenido del control no se puede cambiar
TabStop	boolean	R+W	Se puede acceder al control con la tecla Tab
Align	integer	R+W	Alineación horizontal: 0 = izquierda, 1 = centrado, 2 = derecha
BackgroundColor	long	R+W	Color de fondo
Tag	string	R+W	Información Adicional
HelpText	string	R+W	Texto de la descripción emergente

Propiedades para muchos tipos de controles

Nombre	*Tipo*	*R/W*	*Propiedad*
Text	string	(R+W)	Contenido visualizado en el control. En los controles de texto, se puede leer y procesar, pero no suele funcionar para otros controles.
Spin	boolean	R+W	Botón de incremento o para desplegar el contenido en un campo formateado.
TextColor	long	R+W	Color del texto (primer plano).
DataField	string	R	Nombre del campo en el conjunto de datos.
BoundField	object	R	Conexión con el conjunto de datos (campo enlazado), proporciona acceso al contenido del campo.

Cuadro de texto (TextField)

Nombre	*Tipo*	*R/W*	*Propiedad*
String	string	R+W	Contenido visual del control.
MaxTextLen	integer	R+W	Longitud máxima del texto.

Nombre	*Tipo*	*R/W*	*Propiedad*
DefaultText	string	R+W	Texto predeterminado.
MultiLine	boolean	R+W	Indica si hay más de una línea.
EchoChar	(integer)	R+W	Carácter mostrado durante la entrada de contraseña.

Campo numérico (NumericField)

Nombre	*Tipo*	*R/W*	*Propiedad*
ValueMin	double	R+W	Valor mínimo aceptable
ValueMax	double	R+W	Valor máximo aceptable
Value	double	R+(W)	Valor actual (No lo use para valores del conjunto de datos).
ValueStep	double	R+W	Intervalo de un clic para la rueda del ratón o botón de incremento /decremento del control
DefaultValue	double	R+W	Valor predeterminado.
DecimalAccuracy	integer	R+W	Número de lugares decimales.
ShowThousandsSeparator	boolean	R+W	Permite mostrar el separador de miles (se usa el establecido en la configuración regional).

Campo de moneda (CurrencyField)

Un *Campo de moneda* es un *Campo numérico* con las siguientes posibilidades adicionales.

Nombre	*Tipo*	*R/W*	*Propiedad*
CurrencySymbol	string	R+W	Símbolo de moneda solo para visualización.
PrependCurrencySymbol	boolean	R+W	Muestra el símbolo de moneda antes de la cantidad.

Campo de fecha (DateField)

Los valores de fecha están definidos por el tipo de datos `long` y se muestran en formato ISO: AAAAMMDD, por ejemplo 20190304 para el 04 de marzo de 2019. Para usar este tipo con `getDate` y `updateDate`, y con el tipo `com.sun.star.util.Date`, consulte los ejemplos.

Nombre	*Tipo*	*TipoTipo de datos desde LO 4.1.1*	*R/W*	*Propiedad*
DateMin	long	com.sun.star.util.Date	R+W	Valor de entrada mínimo aceptable
DateMax	long	com.sun.star.util.Date	R+W	Valor de entrada máximo aceptable
Date	long	com.sun.star.util.Date	R+(W)	Fecha

Nombre	*Tipo*	*TipoTipo de datos desde LO 4.1.1*	*R/W*	*Propiedad*
DateFormat	integer		R+W	Formato de fecha específico del sistema operativo: 0 = fecha corta (sencilla) 1 = fecha corta dd.mm.aa (año de 2 dígitos) 2 = fecha corta d.mm.aaaa (año de 4 dígitos) 3 = Fecha larga (con día de la semana y el nombre del mes) Se pueden encontrar más posibilidades en la definición del formulario o en la referencia de la API.
DefaultDate	long	com.sun.star.util.Date	R+W	Valor predeterminado.
DropDown	boolean		R+W	Muestra un calendario mensual desplegable.

Campo de hora (TimeField)

Los valores de hora también son de tipo `long`.

Nombre	*Tipo*	*Tipo desde LO 4.1.1*	*R/W*	*Propiedad*
TimeMin	long	com.sun.star.util.Time	R+W	Valor de entrada mínimo aceptable
TimeMax	long	com.sun.star.util.Time	R+W	Valor de entrada máximo aceptable
Time	long	com.sun.star.util.Time	R+(W)	Valor actual (No lo use para valores del conjunto de datos).
TimeFormat	integer		R+W	Formato de hora: 0 = corto hh: mm (horas, minutos, reloj de 24 horas) 1 = largo hh: mm: ss (reloj de 24 horas) 2 = corto >hh: mm (reloj de 12 horas con AM / PM) 3 = largo hh: mm: ss (reloj de 12 horas con AM / PM) 4 = entrada corta durante un tiempo 5 = entrada larga durante un tiempo
DefaultTime	long	com.sun.star.util.Time	R+W	Valor predeterminado.

Campo formateado (FormattedControl)

Un control de *campo formateado* se puede usar como se desee para números, moneda o fecha / hora. Muchas de las propiedades ya descritas se aplican aquí pero con nombres diferentes.

Nombre	*Tipo*	*R/W*	*Propiedad*
CurrentValue	variant	R	Valor actual del contenido. El tipo de datos real depende del contenido y el formato.
EffectiveValue		R+(W)	
EffectiveMin	double	R+W	Valor de entrada mínimo aceptable
EffectiveMax	double	R+W	Valor de entrada máximo aceptable
EffectiveDefault	variant	R+W	Valor predeterminado.
FormatKey	long	R+(W)	Formato para visualización y entrada. No hay una manera fácil de alterarlo usando una macro.
EnforceFormat	boolean	R+W	El formato se comprueba durante la entrada. Solo se permiten ciertos caracteres y combinaciones.

Listado (ListBox)

El acceso de lectura y escritura al valor relativo a la línea seleccionada, es algo complicado pero posible.

Nombre	*Tipo*	*R/W*	*Propiedad*
ListSource	array of string	R+W	Fuente de datos: fuente de los contenidos del listado o nombre del conjunto de datos.
ListSourceType	integer	R+W	Tipo de fuente de datos: 0 = Lista de valores. 1 = tabla. 2 = consulta. 3 = Conjunto de resultados de una orden SQL. 4 = Resultado de un comando de base de datos. 5 = Nombres de los campos de una tabla.
StringItemList	array of string	R	Lista de opciones disponibles para la selección.
ItemCount	integer	R	Número de opciones disponibles en la lista
ValueItemList	array of string	R	Lista de valores que se pasarán del formulario a la tabla.
DropDown	boolean	R+W	Lista desplegable.
LineCount	integer	R+W	Total de líneas mostradas cuando está totalmente desplegado.
MultiSelection	boolean	R+W	Selección múltiple.
SelectedItems	array of integer	R+W	Lista de opciones seleccionadas como una lista de posiciones en la lista general de entradas.

El primer elemento seleccionado del control de listado se obtiene así:

```
oControl = oForm.getByName("Nombre_del_listado")
sEintrag = oControl.ValueItemList( oControl.SelectedItems(0) )
```

Nota

Desde LibreOffice 4.1, el valor pasado a la base de datos se puede determinar directamente.

```
oControl = oForm.getByName("Nombre_del_Listado")
iD = oControl.getCurrentValue() getCurrentValue()
```

devuelve el valor que se almacenará en la tabla de la base de datos.

En los listados, depende del campo al que están vinculados (`BoundField`). Hasta LibreOffice 4.0 inclusive, esta función devolvía el contenido mostrado, no el valor subyacente en la tabla.

Tenga en cuenta que la entrada es una "matriz de cadenas de texto", la consulta de un control de listado debería cambiarse para restringir una opción de selección:

```
Sub Listenfeldfilter
  Dim stSql(0) As String
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm As Object
  Dim oFeld As Object
  oDoc = thisComponent
  oDrawpage = oDoc.drawpage
  oForm = oDrawpage.forms.getByName("MainForm")
  oFeld = oForm.getByname("Nombre_del_Listado")
  stSql(0) = "SELECT ""Name"", ""ID"" FROM ""Filter_Name"" ORDER BY
""Name"""
  oFeld.ListSource = stSql
  oFeld.refresh
End Sub
```

Cuadro combinado (ComboBox)

A pesar de tener una funcionalidad similar a los listados, las propiedades de los cuadros combinados son algo diferentes. Vea el ejemplo «Cuadros combinados, como listados con opción de entrada» en la página 388.

Nombre	*Tipo*	*R/W*	*Propiedad*
Autocomplete	boolean	R+W	Rellenar automáticamente.
StringItemList	array of string	R+W	Lista de opciones disponibles para su uso.
ItemCount	integer	R	Número de opciones de lista disponibles.
DropDown	boolean	R+W	Lista desplegable.
LineCount	integer	R+W	Número de filas que se muestran al desplegarse.
Text	string	R+W	Texto visualizado actualmente.
DefaultText	string	R+W	Opción predeterminada
ListSource	string	R+W	Nombre de la fuente de datos que proporciona las opciones de la lista.

Nombre	Tipo	R/W	Propiedad
ListSourceType	integer	R+W	Tipo de fuente de datos. Las mismas que para los listados (solo se ignora la elección de la lista de valores).

Casillas de verificación (CheckBox) y Botón de opción (RadioButton)

Nombre	Tipo	R/W	Propiedad
Label	string	R+W	Título (etiqueta)
State	short	R+W	Estado 0 = no seleccionado 1 = seleccionado 2 = indefinido
MultiLine	boolean	R+W	Indica si hay más de una linea.

Campo enmascarado (PatternField)

Además de las propiedades de cuadro de texto tiene >las siguientes:

Nombre	Tipo	R/W	Propiedad
EditMask	string	R+W	Máscara de entrada.
LiteralMask	string	R+W	Máscara de caracteres
StrictFormat	boolean	R+W	Comprobación de formato durante la entrada.

Control de Tablas (GridControl)

Nombre	Tipo	R/W	Propiedad
Count	long	R	Número de columnas.
ElementNames	array of string	R	Lista de nombres de columna.
HasNavigation Bar	boolean	R+W	Barra de navegación.
RowHeight	long	R+W	Altura de la fila.

Etiqueta (Label)

Nombre	Tipo	R/W	Propiedad
Label	string	R+W	Texto mostrado.
MultiLine	boolean	R+W	Indica si hay más de una línea.

Cuadro de Grupo (GroupBox)

No hay propiedades para los cuadros de grupo que normalmente se procesen con macros. Lo que realmente importa es el estado de los *campos de opcion* individuales.

Botones (Button)

Botón o Botón con imagen

Nombre	Tipo	R/W	Propiedad
Label	string	R+W	Título (texto de la etiqueta).

Nombre	*Tipo*	*R/W*	*Propiedad*
State	short	R+W	Estado predeterminado para una selección alternativa.
MultiLine	boolean	R+W	Indica si hay más de una línea.
DefaultButton	boolean	R+W	Si ha de ser el botón predeterminado.

Barra de navegación (NavigationBar)

Las propiedades y métodos adicionales asociados con la navegación, por ejemplo, filtros y cambio del puntero de registro, se controlan mediante el formulario.

Nombre	*Tipo*	*R/W*	*Propiedad*
IconSize	short	R+W	Tamaño de los iconos.
ShowPosition	boolean	R+W	Si la posición se puede ingresar y mostrar.
ShowNavigation	boolean	R+W	Si se permite la navegación.
ShowRecordActions	boolean	R+W	Si se permite grabar acciones.
ShowFilterSort	boolean	R+W	Si se permite la ordenación por filtros.

Métodos para formularios y controles

El tipo de datos del parámetro se indica mediante una abreviatura:

- número de columna para el campo deseado en el archivo de datos, contando desde 1
- valor numérico: podría ser un número entero o decimal
- s String (cadena de texto); La longitud máxima depende de la definición de la tabla.
- b booleano (lógico): verdadero o falso
- d Valor de fecha

Navegaciónr en los conjuntos de datos

Estos métodos funcionan tanto en formularios como en el conjunto de resultados de una consulta.

Cursor en la descripción significa el puntero de registro (**|**).

Nombre	*Tipo*	*Descripción*
Probar la posición del cursor.		
isBeforeFirst	boolean	El cursor está antes del primer registro. Este es el caso si aún no se ha restablecido después de la entrada.
isFirst	boolean	Muestra si el cursor está en la primera entrada.
isLast	boolean	Muestra si el cursor está en la última entrada.
isAfterLast	boolean	El cursor está después de la última fila cuando se mueve hacia la siguiente.
getRow	long	Número de fila en que está el cursor.
Mover el cursor Para los tipos de datos booleanos, True significa que la navegación se realizó.		
beforeFirst	–	Se mueve antes de la primera fila.
first	boolean	Se mueve a la primera fila.

Nombre	*Tipo*	*Descripción*
previous	boolean	Retrocede una fila.
next	boolean	Avanza una fila.
last	boolean	Va al último registro.
afterLast	–	Va después del último registro.
absolute(n)	boolean	Va a la fila con número especificado.
relative(n)	boolean	Avanza o retrocede la cantidad especifica. Hacia adelante para argumentos positivos y hacia atrás para argumentos negativos.
Métodos que afectan el estado actual del registro		
refreshRow	–	Lee de nuevo los valores originales de la fila.
rowInserted	boolean	Indica si es una fila nueva.
rowUpdated	boolean	Indica si la fila actual ha sido alterada.
rowDeleted	boolean	Indica si la fila actual se ha eliminado.

Ediciónde filas de datos (registros)

Los métodos utilizados para la lectura están disponibles para cualquier formulario o para un conjunto de resultados. Los métodos de alteración y almacenamiento solo se pueden usar para conjuntos de datos editables (generalmente tablas, no consultas).

Nombre	*Tipo*	*Descripción*
Métodos para toda la fila		
insertRow	–	Inserta un registro nuevo.
updateRow	–	Actualiza la alteración del registro en curso.
deleteRow	–	Elimina el registro indicado.
cancelRowUpdates	–	Deshace los cambios en el registro correspondiente.
moveToInsertRow	–	Mueve el cursor al registro que se ha insertado.
moveToCurrentRow	–	Tras insertar un nuevo registro, vuelve cursor a su posición anterior.
Valores de lectura		
getString(c)	string	Obtiene el contenido de la columna como una cadena de caracteres.
getBoolean(c)	boolean	Obtiene el contenido de la columna como un valor booleano.
getByte(c)	byte	Obtiene el contenido de la columna como un solo byte.
getShort(c) getInt(c) getLong(c)	Short integer long	Obtiene el contenido de la columna como un entero.

Nombre	*Tipo*	*Descripción*
getFloat(c)	float	Obtiene el contenido de la columna como un único valor de precisión decimal.
getDouble(c)	double	Obtiene el contenido de la columna como un número decimal de doble precisión. Las conversiones automáticas hechas por Basic hacen de este un tipo adecuado para campos decimales y de moneda.
getBytes(c)	array of bytes	Obtiene el contenido de la columna como una matriz de bytes simples.
getDate(c)	Date	Obtiene el contenido de la columna como una fecha.
getTime(c)	Time	Obtiene el contenido de la columna como un valor de hora.
getTimestamp(c)	DateTime	Obtiene el contenido de la columna como marca de tiempo (fecha y hora).
En Basic, los valores de fecha y hora reciben el tipo `DATE`. Para acceder a las fechas en los conjuntos de datos, hay varios tipos: `com.sun.star.util.Date` para una fecha, `com.sun.star.util.Time` para unidades horarias y `com.sun.star.util.DateTime` para una marca de tiempo (fecha y hora).		
wasNull	boolean	Indica si el valor más reciente leído en una columna no tiene datos (`NULL`).
Almacenar valores		
updateNull(c)	–	Establece el contenido de la columna c como vacío (`NULL`).
updateBoolean(c,b)	–	Almacena el contenido booleano (b) en la columna c.
updateByte(c,x)	–	Almacena el byte (x) en la columna c.
updateShort(c,n)updateInt(c,n) updateLong(c,n)	–	Almacena el número entero (n) en la columna c.
updateFloat(c,n) updateDouble(c,n)	–	Almacena el número decimal (n) en la columna c.
updateString(c,s)	–	Almacena la cadena de texto (s) en la columna c.
updateBytes(c,x)	–	Almacena la matriz de bytes (x) en la columna c.
updateDate(c,d)	–	Almacena la fecha (d) en la columna c.
updateTime(c,d)	–	Almacena la hora (d) en la columna c.
updateTimestamp(c,d)	–	Almacena la marca de tiempo (d) en la columna c.

Edición individual de valores

Con estos métodos, el contenido de la columna relevante se lee o se cambia desde un campo de control mediante su enlace con un campo (`BoundField`). Es prácticamente el mismo que el método descrito en la sección anterior, excepto que no se utiliza el número de columna.

Nombre	*Tipo*	*Descripción*
Valores de lectura		
getString	string	Obtiene el contenido del campo como una cadena de texto.
getBoolean	boolean	Obtiene el contenido del campo como un valor lógico.
getByte	byte	Obtiene el contenido del campo como un solo byte.
GetShort getInt getLong	Short integer long	Obtiene el contenido del campo como un entero.
getFloat	float	Obtiene el contenido como un valor decimal de precisión simple.
getDouble	double	Obtiene el contenido del campo como un número decimal de doble precisión. Las conversiones automáticas realizadas por Basic hacen de este un tipo adecuado para campos decimales y de moneda.
getBytes	array of bytes	Obtiene el contenido del campo como una matriz de bytes.
getDate	Date	Obtiene el contenido del campo como una fecha.
getTime	Time	Obtiene el contenido del campo como una hora.
getTimestamp	DateTime	Obtiene el contenido del campo como una marca de tiempo.
En Basic, los valores de fecha y hora reciben el tipo DATE. Para acceder a las fechas, en los conjuntos de datos, hay varios tipos: `com.sun.star.util.Date` para una fecha, `com.sun.star.util.Time` para una expresión horaria y com.sun.star.util.DateTime para una marca de tiempo.		
WasNull	boolean	Indica si el valor más leído en una columna no tiene datos (NULL).
Almacenar valores		
updateNull	–	Establece el contenido de la columna en vacío (NULL).
updateBoolean(b)	–	Establece el contenido de la columna en el valor lógico 'b'.
updateByte(x)	–	Almacena el byte 'x' en la columna.
updateShort(n) updateInt(n) updateLong(n)	– – –	Almacena el entero 'n' en la columna.
updateFloat(n) updateDouble(n)	– –	Almacena el número decimal 'n' en la columna.
updateString(s)	–	Almacena la cadena de caracteres 's' en la columna.
updateBytes(x)	–	Almacena la matriz de bytes 'x' en la columna.
updateDate(d)	–	Almacena la fecha 'd' en la columna.

Nombre	Tipo	Descripción
updateTime(d)	–	Almacena la hora 'd' en la columna.
updateTimestamp(d)	–	Almacena la marca de tiempo 'd' en la columna.

Parámetros para órdenes SQL preparadas

Los métodos que transfieren el valor de una orden SQL preparado previamente (consulte «Órdenes SQL previamente preparadas con parámetros» en la página 351) son similares a los de la sección anterior.

El primer parámetro (indicado por i) es una posición numerada dentro de la orden SQL.

Nombre	Tipo	Descripción
setNull(i, n)	–	Establece el contenido de la columna 'i' en NULL. 'n' es el tipo de datos SQL proporcionado en la Referencia de API.
setBoolean(i, b)	–	Pone el valor lógico 'b' en la orden SQL.
setByte(i, x)	–	Pone el byte 'x' en la orden SQL.
setShort(i, n) setInt(i, n) setLong(i, n)	– – –	Pone el entero 'n' en la orden SQL.
setFloat(i, n) setDouble(i, n)	–	Pone el número decimal 'n' en la orden SQL.
setString(i, s)	–	Pone la cadena de caracteres 's' en la orden SQL.
setBytes(i, x)	–	Pone el conjunto de bytes 'x' en la orden SQL.
setDate(i, d)	–	Pone la fecha 'd' en la orden SQL.
setTime(i, d)	–	Pone la hora 'd' en la orden SQL.
setTimestamp(i, d)	–	Pone la marca de tiempo 'd' en la orden SQL.
clearParameters	–	Elimina valores anteriores de todos los parámetros en la orden SQL.

Mejorar la facilidad de uso

Como primera categoría, se presentan varias opciones que sirven para mejorar la facilidad de uso de los formularios de Base.

Actualización automática de formularios

A menudo, se modifica algo en un formulario y se necesita que esta modificación aparezca en un segundo formulario de la misma página. El siguiente fragmento de código llama al método de actualización del segundo formulario.

```
Sub Update
```

Primero se nombra el procedimiento. La designación predeterminada para un procedimiento es `Sub`. Puede escribirse en mayúsculas o minúsculas. `Sub` incica un procedimiento que normalmente no devuelve ningún valor. Más abajo, se describe una función que puede devolver un valor para su uso posterior.

La macro tiene el nombre `Update`. No necesita declarar variables porque LibreOffice Basic declara automáticamente variables cuando se usan. Si escribe mal una variable, LibreOffice Basic crea silenciosamente una nueva variable sin quejarse.

Utilice `Option Explicit` para evitar que LibreoOffice Basic declare automáticamente variables; (Recomendado por la mayoría de los programadores).

Generalmente se empieza declarando las variables. Todas las variables declaradas aquí son objetos (no números ni texto), por lo que agregamos `As Object` al final de la declaración. Para recordarnos más tarde el tipo de variables, anteponemos a sus nombres una "o". Sin embargo, en principio, puede elegir cualquier nombre de variable que desee.

```
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oForm As Object
```

El formulario se encuentra en el documento actualmente activo. El contenedor, en el que se almacenan todos los formularios, se denomina `drawpage`. En el *Navegador de formularios*, este es el concepto de nivel superior, del que todos los formularios son subsidiarios.

En este ejemplo, el formulario se accede al formulario *Display*. Este es el nombre que se ha dado al formulario y se verá en el navegador de formularios. Para el primer formulario, si no se indica un nombre, Base le asigna el nombre predeterminado: *Form1*.

```
    oDoc = thisComponent
    oDrawpage = oDoc.drawpage
    oForm = oDrawpage.forms.getByName("Display")
```

Dado que el formulario se ha hecho accesible y el punto en el que se puede acceder se guarda en la variable `oForm`, ahora se vuelve a cargar (actualizar) con la instrucción `reload()`.

```
    oForm.reload()
End Sub
```

Al ser un procedimiento (comienza con `SUB`, debe terminar con `End Sub`.

Se puede seleccionar este procedimiento para que se ejecute cuando se guarda un formulario.

Por ejemplo, en una caja registradora, si el número total de artículos vendidos y sus números de identificación en un inventario (leídos por un escáner de código de barras) se ingresan en un formulario, otro formulario en la misma ventana abierta (`drawpage`)puede mostrar los nombres de todos los artículos, y el costo total, cuando se actualice el formulario.

Filtrar registros

Un filtro puede funcionar perfectamente en la forma descrita en el «Capítulo 8, Tareas de base de datos». La variante que se muestra a continuación se encuentra en la base de datos de ejemplo `Example_Search_and_Filter.odb` reemplaza al botón *Guardar* utilizado en el formulario *filter_ without_macros* (de este archivo) y vuelve a leer los controles de listado, de modo que un filtro elegido para un control de listado puede restringir las opciones disponibles en el otro control de listado.

```
Sub Filter
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oForm1 As Object
    Dim oForm2 As Object
    Dim oFieldList1 As Object
    Dim oFieldList2 As Object
```

```
    oDoc = thisComponent
    oDrawpage = oDoc.drawpage
```

Primero, las variables se definen y configuran para acceder al conjunto de formularios. Este conjunto comprende los dos formularios *filter* y *display*. Los controles de listado están en el formulario *filter* y tienen los nombres *listbox1* y *listbox2*.

```
    oForm1 = oDrawpage.forms.getByName("filter")
    oForm2 = oDrawpage.forms.getByName("display")
    oFieldList1 = oForm1.getByName("listbox1")
    oFieldList2 = oForm1.getByName("listbox2")
```

Primero se transfiere el contenido de los Listados al formulario subyacente mediante `commit()`. La transferencia es necesaria, porque de lo contrario el cambio en un listado no se tendrá en cuenta cuando se actualice. La instrucción `commit()` solo debe aplicarse al listado al que se acaba de activar. Después de eso, el registro se guarda usando `updateRow()`.

En principio, nuestra tabla de filtros contiene un único registro, que se escribe una vez al principio. Este registro se sobrescribe continuamente con una instrucción de actualización.

```
    oFieldList1.commit()
    oFieldList2.commit()
    oForm1.updateRow()
```

Los Listados están destinados a influenciarse entre sí. Por ejemplo, si se usa un listado para restringir los artículos visualizados a *CD*, el otro listado no debe incluir a los escritores de libros en su lista de autores. Una selección demasiado frecuente en el segundo listado daría como resultado un filtro vacío. Es por eso que los Listados deben leerse nuevamente. En sentido estricto, la instrucción `refresh()` solo debe aplicarse en el listado al que no se ha accedido.

Después de esto, se lee nuevamente *form2*, que debería mostrar el contenido filtrado.

```
    oFieldList1.refresh()
    oFieldList2.refresh()
    oForm2.reload()
End Sub
```

Mediante varias consultas se puede suministrar el contenido a los listados (influenciados con este método).

La variante más sencilla es hacer que el listado obtenga su contenido de los resultados de un filtro. Posteriormente, otro filtro determinará qué datos serán nuevamente filtrados.

```
SELECT "Field_1" || ' - ' || "Count" AS "Display", "Field_1"
FROM ( SELECT COUNT( "ID" ) AS "Count", "Field_1" FROM "searchtable" GROUP BY
"Field_1" )
ORDER BY "Field1"
```

Se muestra el contenido del campo y el número de resultados. Para obtener el número de resultados, se utiliza una subconsulta. Esto es necesario, ya que de lo contrario solo se mostrará el número de resultados, sin más información que el campo relacionado, en el listado.

Mediante este procecimiento, la macro crea rápidamente listados que solo se llenan con un valor. Si un listado no está vacío (`NULL`), se tiene en cuenta durante el filtrado. Después de activar el segundo listado, solo están disponibles los campos vacíos y un valor mostrado para ambos listados.

Eso puede parecer práctico para una búsqueda limitada. Pero, ¿qué sucede si el catálogo de una biblioteca muestra claramente la ordenación de un elemento, pero no estuviera claro si se trata de un libro, un CD o un DVD?

Una vez seleccionado el sistema de ordenación en el primer listado, si en el segundo listado se selecciona *CD*, debe restablecerse a NULL para realizar una búsqueda posterior que incluya libros. Sería más práctico si el segundo Listado mostrara directamente los diversos tipos de artículos disponibles, con el cómputo de resultados correspondiente.

Para lograr este objetivo, se construye la siguiente consulta, que ya no se alimenta directamente de los resultados del filtro. El número de resultados debe obtenerse de una manera diferente.

```
SELECT
IFNULL( "Field_1" || ' - ' || "Count", 'empty - ' || "Count" ) AS "Display",
"Field_1"
FROM
   ( SELECT COUNT( "ID" ) AS "Count", "Field_1" FROM "Table"
      WHERE "ID" IN
      ( SELECT "Table"."ID" FROM "Filter", "Table"
         WHERE "Table"."Field_2" = IFNULL( "Filter"."Filter_2",
         "Table"."Field_2" ) )
   GROUP BY "Field_1" )
ORDER BY "Field_1"
```

Esta consulta tan compleja puede desglosarse. En la práctica, es común usar una consulta en modo vista para la subconsulta. El listado recibe su contenido de una consulta relacionada con esta consulta(vista) .

La consulta en detalle: la consulta presenta dos columnas. La primera columna contiene la vista visualizada por una persona que tiene el formulario abierto. Esta vista muestra el contenido del campo y, separados por un guion, la cantidad de resultados de este campo. La segunda columna transfiere su contenido a la tabla subyacente del formulario. Aquí solo tenemos el contenido del campo. Los listados extraen su contenido de la consulta, que se presenta como el resultado del filtro en el formulario. Solo estos campos están disponibles para su posterior filtrado.

La tabla de la que se extrae esta información es en realidad una consulta. En esta consulta, se cuentan los campos de la clave primaria (SELECT COUNT("ID") AS "Count"). Se agrupa por el término de búsqueda en el campo (GROUP BY "Field_1"). Esta consulta presenta el término en el campo mismo como la segunda columna y a su vez se basa en una subconsulta adicional:

```
SELECT "Table"."ID" FROM "Filter", "Table"
WHERE "Table"."Field_2" =
   IFNULL( "Filter"."Filter_2", "Table"."Field_2" )
```

Esta subconsulta trata con el otro campo a filtrar. En principio, este campo también debe coincidir con la clave primaria. Si se utilizan más filtros, esta consulta se puede extender:

```
SELECT "Table"."ID" FROM "Filter", "Table" WHERE
"Table"."Field_2" = IFNULL( "Filter"."Filter_2", "Table"."Field_2" )
AND
"Table"."Field_3" = IFNULL( "Filter"."Filter_3", "Table"."Field_3" )
```

Esto permite que cualquier otro campo que se filtre controle lo que finalmente aparece en el listado del primer campo, *Field_1*.

Finalmente, toda la consulta se ordena por el campo subyacente.

En el «Capítulo 8, Tareas de base de datos», se puede ver cómo se muestra realmente la consulta final subyacente al formulario mostrado.

La siguiente macro puede controlar mediante un listado qué listados deben guardarse y cuáles deben leerse nuevamente.

El siguiente procedimiento asume que la propiedad de *Información adicional* de cada listado contiene una lista separada por comas de todos los nombres de listado sin espacios. El primer nombre en la lista debe ser el nombre de ese listado.

```
Sub Filter_more_info(oEvent As Object)
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm1 As Object
  Dim oForm2 As Object
  Dim sTag As String
  sTag = oEvent.Source.Model.Tag
```

Se establece una matriz (una colección de datos accesibles a través de un número de índice) y se llena con los nombres de campo de los listados. El primer nombre en la lista es el nombre del listado vinculado al suceso.

```
  aList() = Split(sTag, ",")
  oDoc = thisComponent
  oDrawpage = oDoc.drawpage
  oForm1 = oDrawpage.forms.getByName("filter")
  oForm2 = oDrawpage.forms.getByName("display")
```

La matriz se ejecuta desde su límite inferior (`Lbound()`) hasta su límite superior (`Ubound()`) en un solo bucle. Todos los valores que estaban separados por comas en la información adicional, ahora se transfieren sucesivamente.

```
  For i = LBound(aList()) To UBound(aList())
    If i = 0 Then
```

El listado que activó la macro debe guardarse. Se encuentra en la variable `aList(0)`. Primero, la información del listado se transfiere a la tabla subyacente y luego se guarda el registro.

```
      oForm1.getByName(aList(i)).commit()
      oForm1.updateRow()
    Else
```

Los otros listados tienen que actualizarse, ya que ahora contienen valores diferentes dependientes del primer listado.

```
      oForm1.getByName(aList(i)).refresh()
    End If
  Next
  oForm2.reload()
End Sub
```

Las consultas para esta macro más operativa (mayor facilidad de uso) son naturalmente las mismas que las presentadas en la sección anterior.

Preparar datos para campos de texto que se ajusten a convenciones SQL

Cuando los datos se almacenan en una orden SQL, los apóstrofos en nombres como "O'Connor" pueden causar problemas. Esto se debe a que las comillas simples (`''`) se usan para encerrar el texto que se ingresará en los registros. En estos casos, necesitamos una función intermedia para preparar los datos adecuadamente.

```
Function String_to_SQL(st As StringString)
```

```
  If InStr(st,"'") Then
    st = Join(Split(st,"'"),"''")
  End If
  String_to_SQL = st
End Function
```

Tenga en cuenta que esta es una función, no un procedimiento. Una función puede tomar un valor como argumento y luego devolver un valor procesado.

Primero se busca el texto a transferir para ver si contiene un apóstrofo. Si este es el caso, el texto se divide en este punto, el apóstrofo en sí mismo es el delimitador de la división, y se une nuevamente con dos apóstrofos. Esto enmascara el código SQL. La función produce su resultado a través de la siguiente llamada:

```
stTextnew = String_to_SQL(stTextold)
```

Esto significa simplemente que la variable `stTextold` se vuelve a procesar y el resultado se almacena en `stTextnew`. Las dos variables en realidad no necesitan tener nombres diferentes. La llamada se puede hacer con:

```
stText = String_to_SQL(stText)
```

Esta función se usa repetidamente en las siguientes macros para que los apóstrofos también se puedan almacenar utilizando órdenes SQL.

Calcular valores anticipadamente

Normalmente, los valores que se pueden calcular utilizando funciones de la base de datos no se almacenan en la base de datos. El cálculo no se lleva a cabo durante la entrada de datos en el formulario, sino al guardar el registro.

Si el formulario consta de un control de tabla,es un poco diferente. El valor calculado se puede leer inmediatamente después de la entrada de datos. Pero cuando los formularios tienen un conjunto de campos individuales, el cálculo correspondiente puede no ser visible.

En estos supuestos, es lógico que los valores que se calculan mediante fórmulas en la base de datos se reflejen en los controles apropiados del formulario. Los siguientes procedimientos que aparecen el la base de datos de ejemplo `Example_direct_Calculation_Form.odb` muestran cómo se pueden hacer visibles estos cálculos.

El primer procedimiento está vinculado a la pérdida del foco del campo *price*.

```
Sub Calculation_without_Tax(oEvent As Object)
  Dim oForm As Object
  Dim oField As Object
  Dim oField2 As Object
  oField = oEvent.Source.Model
  oForm = oField.Parent
  oField2 = oForm.getByName("price_without_tax")
  oField2.BoundField.UpdateDouble(oField.getCurrentValue / 1.19)
  If Not IsEmpty(oForm.getByName("quantity").getCurrentValue()) Then
    total_calc2(oForm.getByName("quantity"))
  End If
End Sub
```

Si se ingresa un valor en el control *price*, la macro se inicia al salir de ese campo. En el mismo formulario, el control *price_without_tax* se actualiza mediante `BoundField.UpdateDouble` calculando el precio sin Impuestos. El campo de datos se deriva de la consulta (vinculada al

formulario), que, en principio, realiza el mismo cálculo. De esta manera, el valor calculado es visible, durante la entrada de datos (mediante la macro), y también más tarde (con la consulta), durante la navegación a través de los registros almacenados.

Si el control de cantidad (*quantity*) contiene un valor, se debe realizar un cálculo adicional para los campos vinculados *total* y *total_without_tax*.

```
Sub Calculation_Total(oEvent As Object)
  oField = oEvent.Source.Model
  Calculation_Total2(oField)
End Sub
```

Este breve procedimiento sirve para transmitir el valor de la cantidad al siguiente procedimiento, cuando se abandona el control *quantity* del formulario.

```
Sub Calculation_Total2(oFeld As Object)
  Dim oForm As Object
  Dim oField2 As Object
  Dim oField3 As Object
  Dim oField4 As Object
  oForm = oFeld.Parent
  oField2 = oForm.getByName("price")
  oField3 = oForm.getByName("total")
  oField4 = oForm.getByName("tax_total")
  oField3.BoundField.UpdateDouble(oField.getCurrentValue *
oField2.getCurrentValue)
  oField4.BoundField.UpdateDouble(oField.getCurrentValue *
oField2.getCurrentValue -
    oField.getCurrentValue * oField2.getCurrentValue / 1.19)
End Sub
```

Este procedimiento afecta a varios campos a la vez. Se inicia desde el campo *quantity* que contiene la cantidad de productos. Con este campo y el campo de precio, se calcula el precio total y el total con Impuestos y se transfieren a los campos correspondientes.

Este diseño tiene un inconveniente: la tasa de Impuestos está formulada dentro del los procedimientos y la consulta. Sería mejor utilizar un argumento, relacionado con el precio del producto, ya que los impuestos pueden variar o no ser los mismos para todos los productos. El valor apropiado para los Impuestos debería leerse de un control en el formulario y estar reflejado en la tabla correspondiente.

Proporcionar la versión actual de LibreOffice

LibreOffice en su versión 4.1 trajo algunos cambios, en los campos de lista y los valores de fecha, que hacen necesario determinar cuál es la versión que se está usando al ejecutar macros en estas circunstancias. El siguiente código sirve para este propósito:

```
Function OfficeVersion()
  Dim aSettings, aConfigProvider
  Dim aParams2(0) As New com.sun.star.beans.PropertyValue
  Dim sProvider$, sAccess$
  sProvider = "com.sun.star.configuration.ConfigurationProvider"
  sAccess = "com.sun.star.configuration.ConfigurationAccess"
  aConfigProvider = createUnoService(sProvider)
  aParams2(0).Name = "nodepath"
```

```
  aParams2(0).Value = "/org.openoffice.Setup/Product"
  aSettings = aConfigProvider.createInstanceWithArguments(sAccess,
aParams2())
  OfficeVersion() =
Array(aSettings.ooName,aSettings.ooSetupVersionAboutBox)
End Function
```

Esta función devuelve una matriz en la que el primer elemento es LibreOffice y el segundo es el número de versión completo, por ejemplo 4.1.5.2.

Obtener el valor devuelto por los listados

Desde LibreOffice 4.1, el valor devuelto por un listado a la base de datos se almacena en `CurrentValue`. Este no era el caso en versiones anteriores, ni en OpenOffice ni en Apache OpenOffice. La siguiente función determinará como usar esta característica para que sea compatible con las distintas versiones. Para ello verifica la versión de LibreOffice.

```
Function ID_Determination(oField As Object) As Integer
  a() = OfficeVersion()
  If a(0) = "LibreOffice" And (LEFT(a(1),1) = 4 And
RIGHT(LEFT(a(1),3),1) > 0) Or     LEFT(a(1),1) > 4 Then
    stContent = oField.currentValue
  Else
```

Antes de LibreOffice 4.1, el valor que se pasaba se leía de la lista de valores del listado. El registro visiblemente elegido es `SelectedItems (0)`. Se utilizaba `0` porque podrían seleccionarse varios valores adicionales en un listado.

```
    stContent = oField.ValueItemList(oField.SelectedItems(0))
  End If
  If IsEmpty(stContent) Then
```

-1 es un valor que no se utiliza como Valor Automático y, por lo tanto, no existirá en la mayoría de las tablas como clave foránea.

```
    ID_Determination = -1
  Else
    ID_Determination = Cint(stContent) 'convertir a entero
  End If
End Function
```

La función transmite el valor como un entero. La mayoría de las claves primarias son enteros de incremento automático. Cuando una clave foránea no cumple este criterio, el valor de retorno debe ajustarse al tipo apropiado.

El valor visualizado de un listado se puede determinar adicionalmente utilizando la propiedad de presentación del campo.

```
Sub Listfielddisplay
  Dim oDoc As Object
  Dim oForm As Object
  Dim oListbox As Object
  Dim oController As Object
  Dim oView As Object
  oDoc = thisComponent
```

```
  oForm = oDoc.Drawpage.Forms(0)
  oListbox = oForm.getByName("Listbox")
  oController = oDoc.getCurrentController()
  oView = oController.getControl(oListbox)
  print "Displayed content: " & oView.SelectedItem
End Sub
```

El controlador *oController* se utiliza para acceder a la vista del formulario. Esto determina lo que aparece en la interfaz visual. El valor seleccionado es `SelectedItem`.

Limitar el contenido de los listados ingresando letras iniciales

A veces el contenido de los listados es demasiado grande para manejarlo. Para agilizar la búsqueda en estos casos, es útil limitar el contenido del listado a los valores indicados al ingresar uno o más caracteres iniciales. El listado en sí se proporciona con una orden SQL que sirve como marcador de posición. Podría ser:

```
SELECT "Name", "ID" FROM "Table" ORDER BY "Name" LIMIT 5
```

Evita que Base tenga que leer una gran lista de valores cuando se abre el formulario.

La siguiente macro está vinculada a las propiedades del listado **Sucesos > Clave liberada**.

```
Global stListStart As String
Global lTime As Long
```

Primero, se crean variables globales. Estas variables son necesarias para permitir la búsqueda no solo de una sola letra, sino también, tras pulsar más teclas, para combinaciones de letras.

Las letras ingresadas se almacenan secuencialmente en la variable global `stListStart`.

La variable global `lTime` se usa para almacenar la hora actual en segundos. Si hay una pausa larga entre las pulsaciones de teclas, la variable `stListStart` debe restablecerse. Por esta razón, se consulta la diferencia horaria entre entradas sucesivas.

```
Sub ListFilter(oEvent As Object)
  oField = oEvent.Source.Model
  If oEvent.KeyCode < 538 Then
```

La macro se inicia con una pulsación de tecla. Dentro de la interfaz de programación (API), cada tecla tiene un código numérico (`KeyCode`) que se puede buscar en el siguiente enlace de internet: com::sun::star::awt::Key.

Los caracteres especiales como ä, ö y ü tienen el `KeyCode 0`. Todas las demás letras y números tienen un `KeyCode` menor que 538.

Es importante verificar el `KeyCode` porque al presionar la tecla *Tab* para moverse a otro campo también se iniciará la macro. El `KeyCode` para la tecla *Tab* es 1282, por lo que no se ejecutará ningún código adicional en la macro.

```
    Dim stSql(0) As String
```

El código SQL para el listado se almacena en una matriz. Sin embargo, las órdenes SQL cuentan como elementos de datos únicos, por lo que la matriz se dimensiona como `stSql(0)`.

Cuando lea el código SQL del listado, tenga en cuenta que el código SQL no es accesible directamente como texto. En cambio, el código está disponible como un único elemento de matriz: `oField.ListSource(0)`.

Después de declarar variables para uso futuro, la orden SQL se divide. Para obtener el campo que se va a filtrar, dividimos el código en la primera coma. Por lo tanto, el campo debe colocarse al

principio de la orden. Luego, este código se divide nuevamente en el primer carácter de comillas dobles, que introduce el nombre del campo. Se hace usando matrices simples. La variable `stField` necesita que las comillas vuelvan a aparecer al principio. Además se usa `Rtrim` para evitar que se produzca un espacio al final de la expresión.

```
Dim stText As String
Dim stField As String
Dim stQuery As String
Dim ar0()
Dim ar1()
ar0() = Split(oField.ListSource(0),",", 2)
ar1() = Split(ar0(0),"""", 2)
stFeld = """ & Rtrim(ar1(1))
```

Se espera una instrucción de ordenación a continuación en el código SQL. Sin embargo, las órdenes en SQL pueden estar en mayúsculas, minúsculas o mixtas, por lo que se usa la función `inStr` en lugar de `Split` para encontrar la cadena de caracteres *ORDER*. El último parámetro para esta función es `1`, lo que indica que la búsqueda no distingue entre mayúsculas y minúsculas. Todo a la izquierda de la cadena *ORDER* se debe utilizar para construir el nuevo código SQL. Esto garantiza que el código también pueda servir para listados provenientes de diferentes tablas o que se han definido en el código SQL utilizando condiciones.

```
  stQuery = Left(oField.ListSource(0), inStr(1,oField.ListSource(0),
"ORDER",1)-1)
  If inStr(stQuery, "LOWER") > 0 Then
    stQuery = Left(stQuery, inStr(stQuery, "LOWER")-1)
  ElseIf inStr(1,stQuery, "WHERE",1) > 0 Then
    stQuery = stQuery & " AND "
  Else
    stQuery = stQuery & " WHERE "
  End If
```

Si la consulta contiene el término `LOWER`, significa que se creó utilizando este procedimiento `ListFilter`. Por lo tanto, al construir la nueva consulta, necesitamos ir solo hasta esta posición.

Si este no es el caso, y la consulta ya contiene el término `WHERE` (en mayúscula o minúscula), cualquier condición adicional a la consulta debe anteponerse con `AND`.

Si no se cumple ninguna de las dos condiciones, se añade un `WHERE` al código existente.

```
If lTime > 0 And Timer() - lTime < 5 Then
  stListStart = stListStart & oEvent.KeyChar
Else
  stListStart = oEvent.KeyChar
End If
lTime = Timer()
```

Si se ha almacenado un valor de hora en la variable global, y la diferencia entre este valor y la hora actual es inferior a 5 segundos, la letra ingresada se une a la anterior. De lo contrario, la letra se trata como una nueva entrada de una sola letra. El control de listado se volverá a filtrar de acuerdo con esta entrada. Después de esto, la hora actual se almacena en `lTime`.

```
stText = LCase( stListStart & "%")
stSql(0) = stQuery + "LOWER("+stField+") LIKE '"+stText+"' ORDER
BY "+stField+""
```

```
      oFeld.ListSource = stSql
      oField.refresh
    End If
  End Sub
```

El código SQL finalmente se une. La versión en minúsculas del contenido del campo se compara con la versión en minúsculas de las letras ingresadas. El código se inserta en el listado y el campo se actualiza para que solo se pueda buscar el contenido filtrado.

Convertir fechas de un formulario en una variable de tipo fecha

```
  Function DateValue(oField As Object) As Date
    a() = OfficeVersion()
    If a(0) = "LibreOffice" And (LEFT(a(1),1) = 4 And
RIGHT(LEFT(a(1),3),1) > 0)
      Or LEFT(a(1),1) > 4 Then
```

Aquí se interceptan todas las versiones de LibreOffice desde la 4.1 en adelante. Para este propósito, el número de versión se divide en sus elementos individuales, y se verifican los números de versión mayor y menor. Esto funcionará hasta LibreOffice 9.

```
      Dim stMonth As String
      Dim stDay As String
      stMonth = Right(Str(0) & Str(oField.CurrentValue.Month),2)
      stDay = Right(Str(0) & Str(oField.CurrentValue.Day),2)
      Datumswert = CDateFromIso(oField.CurrentValue.Year & stMonth &
stDay)
    Else
      DateValue = CDateFromIso(oField.CurrentValue)
    End If
  End Function
```

Desde LibreOffice 4.1.2, las fechas se han almacenado como matrices dentro de los controles de formulario. Esto significa que el valor actual del control no se puede usar para acceder a la fecha en sí. La fecha debe recrearse a partir del día, mes y año si se va a seguir usando en macros.

Buscar registros de datos

Se pueden buscar registros en la base de datos sin usar una macro. Sin embargo, configurar una consulta compleja puede llegar a ser una tarea muy complicada. Una macro puede resolver este problema con un simple bucle.

El procedimiento siguiente lee los campos en una tabla, crea una consulta interna y, finalmente, escribe una lista de los números de las claves primarias de registros en la tabla que se recuperan con este término de búsqueda. En la siguiente descripción, hay una tabla llamada *Searchtmp*, que consiste en un campo de clave primaria (*ID*) de incremento automático y un campo llamado *Nr*. que contiene todas las claves primarias recuperadas de la tabla en la que se busca. El nombre de la tabla se proporciona inicialmente como una variable.

Para obtener un resultado correcto, la tabla tiene que incluir el contenido que está buscando como texto y no como claves foráneas. Si es necesario, puede crear una consulta en modo vista para que la macro la use.[14]

```
  Sub Searching(stTable As String)
    Dim oDataSource As Object
```

14 Consulte la base de datos >`Example_Search_and_Filter.odb` >asociada a esta guía.

```
Dim oConnection As Object
Dim oSQL_Command As Object
Dim stSql As String
Dim oResult As Object
Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm As Object
Dim oForm2 As Object
Dim oField As Object
Dim stContent As String
Dim arContent() As String
Dim inI As Integer
Dim inK As Integer
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("searchform")
oField = oForm.getByName("searchtext")
stContent = oField.getCurrentValue()
stContent = LCase(stContent)
```

El contenido del cuadro de texto de búsqueda se convierte inicialmente en minúsculas, de modo que la función de búsqueda posterior solo necesita comparar lo ingresado en minúsculas.

```
oDataSource = ThisComponent.Parent.DataSource
oConnection = oDataSource.GetConnection("","")
oSQL_Command = oConnection.createStatement()
```

Primero se debe determinar si realmente se ha ingresado un término de búsqueda. Si el control está vacío, se supone que no se está haciendo una búsqueda y se mostrarán todos los registros.

Si se ingresó un término de búsqueda, los nombres de columna se leen de la tabla en la que se está buscando, de modo que la consulta pueda acceder a los campos.

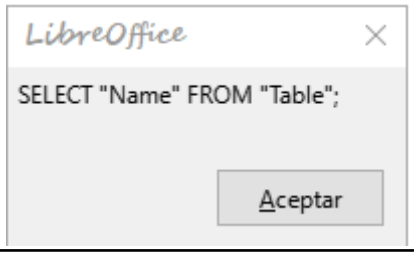
```
If stContent <> "" Then
  stSql = "SELECT ""COLUMN_NAME"" FROM
""INFORMATION_SCHEMA"".""SYSTEM_COLUMNS"" WHERE ""TABLE_NAME"" = '"
+ stTable + "' ORDER BY ""ORDINAL_POSITION"""
  oResult = oSQL_Statement.executeQuery(stSql)
```

Nota

Las fórmulas SQL en macros primero deben colocarse entre comillas dobles como cadenas de caracteres normales. Los nombres de campo y de tabla ya están entre comillas dobles dentro de la fórmula SQL. Para crear un código final que transmita las comillas dobles correctamente, los nombres de campo y de tabla deben recibir dos conjuntos de estas comillas.

```
StSql =" SELECT ""Name"" FROM ""Table""; "
```

Esto convierte la variable *StSql* a la orden SQL correcta, se puede comprobar usando la función Msgbox de Basic: `MsgBox stSql .`



El índice de la matriz, en el que se escriben los nombres de los campos, se establece inicialmente en `0`. Luego, la consulta comienza a leerse. Como se desconoce el tamaño de la matriz, debe ajustarse continuamente. Por eso el ciclo comienza con `ReDim Preserve arContent(inl)` para establecer el tamaño de la matriz y al mismo tiempo preservar el contenido existente. A continuación, se leen los campos y el índice de la matriz se incrementa en `1`. Luego, la matriz se dimensiona nuevamente y se puede almacenar un valor adicional.

```
    InI = 0
    While oResult.next
      ReDim Preserve arContent(inI)
      arContent(inI) = oResult.getString(1)
      inI = inI + 1
    Wend
    stSql = "DROP TABLE ""searchtmp"" IF EXISTS"
    oSQL_Command.executeUpdate (stSql)
```

La consulta se reúne dentro de un bucle y posteriormente se aplica a la tabla definida al principio. Se permiten todas las combinaciones de mayúsculas y minúsculas, ya que el contenido del campo en la consulta se convierte a minúsculas.

La consulta se construye de manera que los resultados terminen en la tabla *searchtmp*. Se supone que la clave primaria es el primer campo de la tabla (`arContent(0)`).

```
    stSql = "SELECT """+arContent(0)+""" INTO ""searchtmp"" FROM """ +
stTable _
 + """ WHERE "
    For inK = 0 To (inI - 1)
      stSql = stSql+"LCase("""+arContent(inK)+""") LIKE
'%"+stContent+"%'"
      If inK < (inI - 1) Then
        stSql = stSql+" OR "
      End If
    Next
    oSQL_Command.executeQuery(stSql)
  Else
    stSql = "DELETE FROM ""searchtmp"""
    oSQL_Command.executeUpdate (stSql)
  End If
```

El formulario *display* tiene que recargarse. Su fuente de datos es una consulta, en este ejemplo *Searchquery*.

```
    oForm2 = oDrawpage.forms.getByName("display")
    oForm2.reload()
  End Sub
```

Esto crea una tabla que debe ser evaluada por la consulta. En la medida de lo posible, la consulta debe construirse de modo que pueda editarse posteriormente. Se muestra una consulta como ejemplo:

```
SELECT * FROM "searchtable" WHERE "Nr." IN ( SELECT "Nr." FROM "searchtmp" ) OR "Nr."
= CASE WHEN ( SELECT COUNT( "Nr." ) FROM "searchtmp" ) > 0 THEN '0' ELSE "Nr." END
```

Se incluyen todos los elementos de *searchtable*, incluida la clave primaria. Ninguna otra tabla aparece en la consulta directa; por lo tanto, no se necesita ninguna clave primaria de otra tabla y el resultado de la consulta sigue pudiéndose editar.

La clave primaria se guarda en este ejemplo con el nombre *Nr*. La macro lee precisamente este campo. Hay una verificación inicial para ver si el contenido del campo *Nr*. aparece en la tabla *searchtmp*. El operador `IN` es compatible con múltiples valores. La subconsulta también puede generar varios registros.

Para grandes cantidades de datos, la coincidencia de valores mediante el uso del operador `IN` se ralentiza notablemente. Por lo tanto, no es una buena idea usar un campo de búsqueda vacío simplemente para transferir todos los campos de clave primaria de *searchtable* a la tabla *searchtmp* y luego ver los datos de la misma manera. En cambio, un campo de búsqueda vacío crea una tabla *searchtmp* vacía, de modo que no hay registros disponibles. Este es el propósito de la segunda parte de la condición:

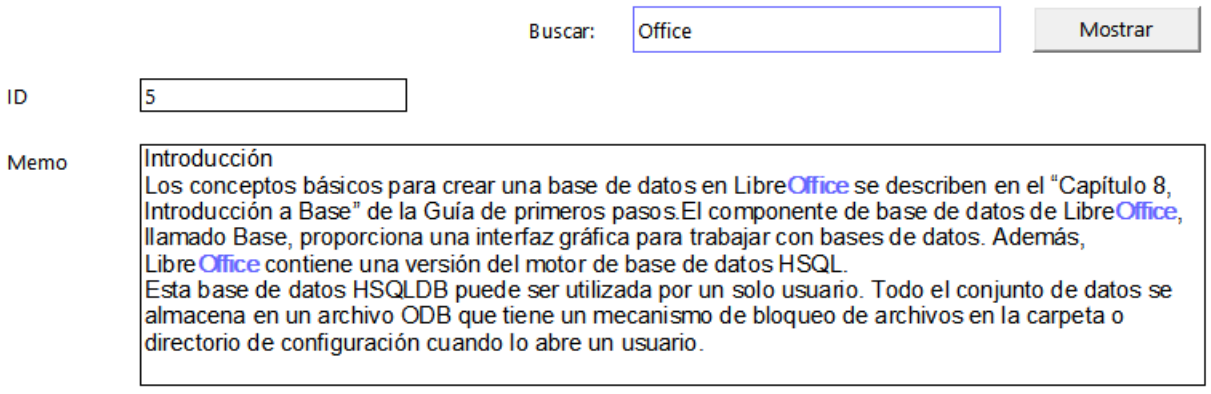
```
OR "Nr." = CASE WHEN ( SELECT COUNT( "Nr." ) FROM "searchtmp" ) > 0 THEN '-1' ELSE
"Nr." END
```

Si se encuentra un registro en la tabla *searchtmp*, significa que el resultado de la primera consulta es mayor que 0. En este caso: `"Nr." = '-1'` (aquí necesitamos un número que no puede aparecer como una clave primaria, por eso `'-1'`es un buen valor).

Si la consulta arroja exactamente `0` (que será el caso si no hay registros presentes), entonces `"Nr." = "Nr."`. Esto enumerará cada registro que tenga un `Nr.` , al ser `Nr.` la clave primaria, esto significa todos los registros.

Resaltar términos de búsqueda en formularios y resultados

Con un cuadro de texto grande, a menudo no está claro dónde ocurren las coincidencias de un término de búsqueda. Sería más útil si el formulario pudiera resaltar las ocurrencias.



Para que un formulario funcione de esta manera, necesitamos un par de elementos adicionales en nuestra caja de trucos. Esta macro está incluida en las bases de datos de ejemplo, en el archivo `Example_autotext_Searchmark_Spelling.odt`.

El funcionamiento de un campo de búsqueda ya se ha explicado. Se crea una tabla de filtro y se usa un formulario para escribir los valores actuales de un único registro en esta tabla. Al >formulario principal se le proporciona contenido mediante una consulta:

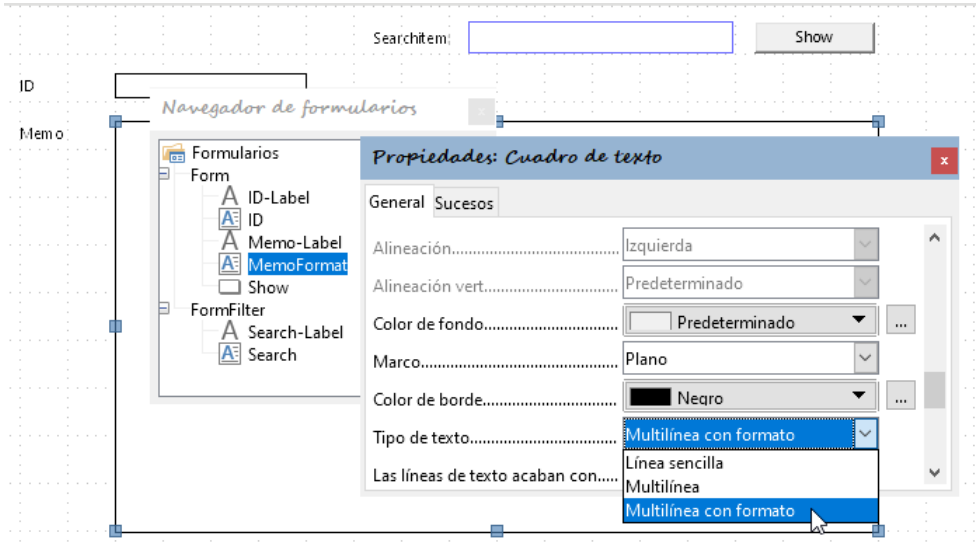
```
SELECT "ID", "memo"
```

```
FROM "table"
WHERE LOWER ( "memo" ) LIKE '%' || LOWER (
   ( SELECT "searchtext" FROM "filter" WHERE "ID" = TRUE ) ) || '%'
```

Cuando se ingresa el texto de búsqueda, y se pulsa el botón mostrar, se puede acceder a todos los registros en la tabla *table* que contienen el texto ingresado en el campo *memo*.

Se podrá usar los botones de navegación de formulario para acceder solo a los registros que cumplan esa condición. La búsqueda se configura para que no distinga entre mayúsculas y minúsculas.

Si no se ingresa ningún texto de búsqueda, se puede acceder a todos los registros de la tabla mediante los botones de navegación. Como la clave primaria de esta tabla se incluye en la consulta, esta consulta se puede editar.



En el formulario *Form*, además del campo *ID* para la clave primaria, hay un campo *MemoFormat* que se puede configurar para que muestre texto fomateado usando en sus propiedades generales: **Tipo de texto > Multirenglón con formato** dejaremos su color predeterminado (negro).

Al fijarnos en las propiedades del cuadro de texto, la pestaña *Datos* ahora ha desaparecido. Esto se debe a que no se pueden ingresar texto con formato en un campo de la base de datos, la base de datos no lo puede almacenar. Sin embargo, todavía es posible obtener texto en este campo, marcarlo y transferirlo a la base de datos después de una actualización mediante el uso de una macro.

El procedimiento `ContentRead` sirve para transferir el contenido del campo *memo* de la base de datos al cuadro de texto formateado *MemoFormat*, y para formatearlo, en este caso resaltando las coincidencias con el texto ingresado en el campo de búsqueda.

El procedimiento está ligado a las propiedades del Formulario **Sucesos > Tras el cambio de registro de datos.**

```
Sub ContentRead(oEvent As Object)
  Dim inMemo As Integer
  Dim oField As Object
  Dim stSearchtext As String
  Dim oCursor As Object
  Dim inSearch As Integer
  Dim inSearchOld As Integer
  Dim inLen As Integer
```

```
oForm = oEvent.Source
inMemo = oForm.findColumn("memo")
oField = oForm.getByName("MemoFormat")
oField.Text = oForm.getString(inMemo)
```

Primero se definen las variables. Luego, se busca el campo *memo* de tabla (mediante su relación con el formulario), la función `getString()` se usa para leer el texto de la columna numerada. Esto se transfiere al campo que puede formatearse pero que no tiene enlace con la base de datos (*MemoFormat)*.

Las pruebas iniciales mostraron que el formulario se abrió, pero la barra de herramientas del formulario en la parte inferior ya no se creó. Para lo que se generó una espera muy corta de 5 milisegundos.

Después de esto, el contenido que se va a buscar se obtiene del formulario *FormFilter* (paralelo al formulario *Form* en la jerarquía de formularios).

```
Wait 5
stSearchtext =
oForm.Parent.getByName("FormFilter").getByName("Search").Text
```

Para poder formatear el texto, se tiene que crear un cursor de texto invisible (`TextCursor`) en el control que contiene el texto (*MemoFormat*). El formato predeterminado del campo utiliza una fuente serif de 12 puntos que puede no aparecer en otras partes del formulario y no puede personalizarse directamente con las propiedades de control del formulario. En este procedimiento, el texto se establece en la apariencia deseada desde el principio. Si esto no se hace, las diferencias en el formato pueden hacer que se corte el límite superior del texto en el campo. En las primeras pruebas, solo 2/3 de la primera línea eran legibles.

Para que el cursor marque el texto, se establece su inicio al principio del campo y luego al se selecciona hasta el final. El argumento en ambos casos es `true`. Después se establecen las propiedades de la fuente (tipo de fuente, color y tamaño) y luego, el cursor vuelve al principio.

```
oCursor = oField.createTextCursor()
oCursor.gotoStart(true)
oCursor.gotoEnd(true)
oCursor.CharHeight = 10
oCursor.CharFontName = "Arial, Helvetica, Tahoma"
oCursor.CharColor = RGB(0,0,0)
oCursor.CharWeight = 100.000000 'com::sun::star::awt::FontWeight
oCursor.gotoStart(false)
```

Si hay texto en el campo formateado y se ha realizado una entrada en el control de búsqueda solicitando una búsqueda, se recorre el contenido para encontrar la cadena de texto. El condicional `If` comprueba primero si se cumplen estas condiciones; el bucle `Do While` comprueba si la cadena buscada está realmente en el texto del campo *MemoFormat*. En realidad, esta configuración podría omitirse, ya que la consulta en la que se basa el formulario solo muestra texto que cumple estas condiciones.

```
If oField.Text <> "" And stSearchtext <> "" Then
  If inStr(oField.Text, stSearchtext) Then
    inSearch = 1
    inSearchOld = 0
    inLen = Len(stSearchtext)
```

Se busca en el texto la cadena introducida. Mediante el bucle que termina cuando no hay más coincidencias. La función `inStr()` devuelve la ubicación del primer carácter de la cadena de

búsqueda en el formato de visualización especificado. El ciclo está controlado por el requisito de que al final de cada ciclo, el valor de `inSearch` se haya incrementado en 1 (-1 en la primera línea del ciclo y +2 en la última línea). Para cada ciclo, el cursor se mueve a la posición inicial sin seleccionar el texto usando `oCursor.goRight(Posición_destino,false)`, y luego a la derecha seleccionando el texto (determinado por la longitud de la cadena de búsqueda) `oCursor.goRight(inLen,true)` y luego se resalta el texto encontrado con el formato deseado (color azul o algo más destacado) `oCursor.CharColor = RGB(102,102,255)`. El cursor se mueve de regreso a su nuevo punto de partida para la próxima ejecución.

```
        Do While inStr(inSearch, oField.Text, stSearchtext) > 0
          inSearch = inStr(inSearch, oField.Text, stSearchtext) - 1
          oCursor.goRight(inSearch-inSearchOld,false)
          oCursor.goRight(inLen,true)
          oCursor.CharColor = RGB(102,102,255)
          oCursor.CharWeight = 110.000000
          oCursor.goLeft(inLen,false)
          inSearchOld = inSearch
          inSearch = inSearch + 2
        Loop
      End If
    End If
End Sub
```

El procedimiento `ContentWrite` sirve para transferir el contenido del cuadro de texto formateado *MemoFormat* a la base de datos independientemente de si se produce o no alguna alteración.

El procedimiento está vinculado a las propiedades del formulario **Sucesos > Antes del cambio de registro**.

```
Sub ContentWrite(oEvent As Object)
  Dim oForm As Object
  Dim inMemo As Integer
  Dim loID As Long
  Dim oField As Object
  Dim stMemo As String
  oForm = oEvent.Source
  If InStr(oForm.ImplementationName, "ODatabaseForm") Then
```

El suceso desencadenante se implementa dos veces. Solo el nombre de implementación que termina con `OdatabaseForm` proporciona el acceso correcto al registro (las implementaciones se explican en la página 406).

```
If Not oForm.isBeforeFirst() And Not oForm.isAfterLast() Then
```

Cuando el formulario se lee o se vuelve a cargar, el cursor se coloca antes del registro actual. Luego, si se realiza un intento, aparece el mensaje "*Estado de cursor no válido*".

```
        inMemo = oForm.findColumn("memo")
        loID = oForm.findColumn("ID")
        oField = oForm.getByName("MemoFormat")
        stMemo = oField.Text
        If stMemo <> "" Then
          oForm.updateString(inMemo,stMemo)
        End If
```

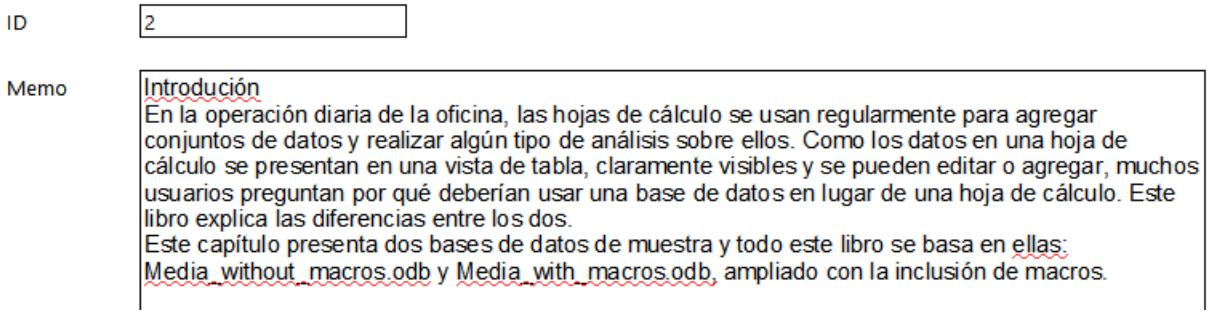
```
        If stMemo <> "" And oForm.getString(loID) <> "" Then
          oForm.UpdateRow()
        End If
      End If
    End If
  End Sub
```

Desde la fuente de datos enlazada al formulario se obtienen los campos *memo* y el campo *ID*. Si el control *MemoFormat* contiene texto se transfiere al campo *memo* mediante `oForm.updateString()`. Solo si el control *ID* del formulario contiene un número (se ha establecido un dato para la clave primaria) se actualiza el registro. De lo contrario, se crea un nuevo registro mediante el funcionamiento normal del formulario. El formulario reconoce el cambio y lo almacena de manera independiente.

Revisar la ortografía durante la entrada de datos

Esta macro se puede usar para cuadros de texto *Multirenglón con formato*. Como en el capítulo anterior,está incluida en `Example_autotext_Searchmark_Spelling.odt`.

Primero se debe escribir el contenido de cada registro y luego se puede cargar el nuevo registro en el control de formulario. Los procedimientos *ContentRead* y *ContentWrite* difieren solo en el punto en el que la función de búsqueda se puede excluir.



El corrector ortográfico se inicia en el formulario anterior al pulsar la *barra espaciadora* o la tecla *Entrar* dentro del control de texto formateado del formulario. Se ejecuta al final de cada palabra. Y posiblemente se podría combinar con la pérdida de foco del control para garantizar que se verifique la última palabra.

El procedimiento está vinculado con las propiedades del cuadro de texto *Memoformat* **Sucesos > Despues de haber pulsado la tecla.**

```
SUB MarkWrongWordsDirect(oEvent As Object)
  GlobalScope.BasicLibraries.LoadLibrary("Tools") 'carga la biblioteca
auxiliar "Tools" con funciones utilizadas en el procedimiento
```

La función `RTrimStr` se usa para eliminar cualquier signo de puntuación al final de la cadena, de lo contrario, todas las palabras que terminaban con una coma, punto final u otro signo de puntuación aparecerían como errores ortográficos. Además, `LTrimChar` se usa para eliminar paréntesis de apertura al comienzo de las palabras.

```
  Dim aProp() As New com.sun.star.beans.PropertyValue
  Dim oLinuSvcMgr As Object
  Dim oSpellChk As Object
  Dim oField As Object
  Dim arText()
  Dim stWord As String
```

```
  Dim inlenWord As Integer
  Dim ink As Integer
  Dim i As Integer
  Dim oCursor As Object
  Dim stText As Object
  oLinguSvcMgr =
createUnoService("com.sun.star.linguistic2.LinguServiceManager")
  If Not IsNull(oLinguSvcMgr) Then
    oSpellChk = oLinguSvcMgr.getSpellChecker()
  End If
```

Primero se declaran todas las variables. Luego se accede al módulo básico de corrección ortográfica `SpellChecker`. Será este módulo el que realmente verificará la corrección de las palabras individuales.

```
  oField = oEvent.Source.Model
  ink = 0
  If oEvent.KeyCode = 1280 Or oEvent.KeyCode = 1284 Then
```

El suceso que inicia la macro es una pulsación de tecla. Este suceso incluye un código, `KeyCode`, para cada clave individual. El `KeyCode` para la tecla *Entrar* es 1280, el de *espacio* es 1284. Al igual que muchos otros datos, estos elementos se pueden inspeccionar mediante la herramienta *XrayTool*. Si se pulsa el *espacio* o el *retorno*, se verifica la ortografía. Se lanza, en otras palabras, al final de cada palabra. Solo la prueba de la última palabra no se produce automáticamente.

Cada vez que se ejecuta la macro, se verifican todas las palabras del texto. También se podría hacer una verificación individual de palabras, pero sería mucho más compleja.

El texto se divide en palabras individuales. El delimitador es el carácter del espacio. Antes de eso, las palabras divididas por saltos de línea deben unirse nuevamente, o los fragmentos de texto podrían confundirse con palabras completas.

```
    stText = Join(Split(oField.Text,CHR(10))," ")
    stText = Join(Split(stText,CHR(13))," ")
    arText = Split(RTrim(stText)," ")
    For i = LBound(arText) To Ubound(arText)
      stWord = arText(i)
      inlenWord = len(stWord)
      stWord = Trim( RtrimStr( RtrimStr( RtrimStr( RtrimStr(
RtrimStr(                              RtrimStr(stWord,","),
"."),"?"),"!"),"."),")"))
      stWord =
Trim(LTrimChar(LTrimChar(LTrimChar(stWord,"("),"¿"),"¡"))
```

Se leen las palabras individuales, su longitud (sin recortar) es necesaria para el siguiente paso de edición. Solo así se puede determinar la posición de la palabra dentro del texto completo (necesario para el resaltado específico de los errores ortográficos).

La función `Trim` se usa para eliminar espacios, mientras que `RTrimStr` elimina comas y signos de puntuación al final del texto y `LTrimChar` cualquier signo de puntuación al principio.

```
    If stWord <> "" Then
      oCursor = oField.createTextCursor()
      oCursor.gotoStart(false)
      oCursor.goRight(ink,false)
```

```
            oCursor.goRight(inLenWord,true)
            If Not oSpellChk.isValid(stWord, "en", aProp()) Then
             oCursor.CharUnderline = 9
             oCursor.CharUnderlineHasColor = True
             oCursor.CharUnderlineColor = RGB(255,51,51)
            Else
             oCursor.CharUnderline = 0
            End If
          End If
          ink = ink + inLenWord + 1
        Next
      End If
    End Sub
```

Si la palabra no está vacía, se crea un cursor de texto. Este cursor se mueve sin resaltar al comienzo del texto en el campo de entrada. Luego salta hacia la derecha, aún sin resaltar, al término almacenado en la variable `ink`. Esta variable comienza con 0, pero después de que se haya ejecutado el primer bucle, es igual a la longitud de la palabra (+1 para el siguiente espacio).

Luego, el cursor se mueve hacia la derecha según la longitud de la palabra actual. Las propiedades de la fuente se modifican para crear el resaltado.

Se lanza el corrector ortográfico (`spellchecker`) que necesita la palabra a comprobar y el código del país como argumentos; si no se indica un código de país, lo tomará como correcto. El argumento de matriz generalmente está vacío.

Si la palabra no está en el diccionario, está subrayada con una línea roja ondulada. Este tipo de subrayado está representado por el número 9.

Si a palabra está en el diccionario, se elimina el subrayado (0). Este paso es necesario porque de lo contrario una palabra reconocida como falsa y luego corregida continuaría resaltada con la línea ondulada roja. Nunca se eliminaría si no se proporciona el formato para una ortografía correcta.

Cuadros combinados, como listados con opción de entrada

Mientras el control de *Listado* nos permite unicamente elegir un valor dentro de un listado, el control de *Cuadro combinado* funciona como un control de listado, pero además brinda la posibilidad de introducir datos que se pueden almacenar en una tabla.

Mediante el uso de cuadros combinados y un control de campo numérico (invisibles) se puede usar una tabla con dos campos: la clave primaria que servirá como referente de la elección del usuario y el otro para mostrar las entradas correspondientes y almacenar nuevos datos.

En nuestro ejemplo, `Example_Combobox_Listfield.odb`, usamos una tabla principal (*name*)para todos los datos de unos clientes y dos cuadros combinados que obtienen y actualizan la información de otras tablas: la calle y código postal y ciudad. Los cuadros combinados del formulario deben transferir la claves primarias y el contenido relacionado las tablas subsidiarias y además almacenar esa clave primaria en la tabla principal para, con una consulta, obtener todos los datos del cliente. Las claves primarias que se deben transferir se obtienen mediante los controles de campos numéricos invisibles.

Los valores de los controles invisibles se leen cuando se carga el formulario y en el cuadro combinado se puede presentar el contenido relacionado con ese valor. Si se cambia el contenido del cuadro combinado se guarda el contenido mostrado y el valor relacionado (clave primaria).

Si se utilizan consultas que puedan ser editadas en lugar de tablas, el texto que se mostrará en los campos combinados se puede obtener directamente a partir de la consulta, con lo que no se requiere una macro para este paso.

Para el funcionamiento de la macro se entiende que la clave primaria de fuente de datos para el cuadro combinado es un entero de incremento automático y tiene el nombre *ID*.

Visualizar texto en cuadros combinados

El siguiente procedimiento sirve para mostrar texto en el cuadro combinado de acuerdo con el valor de la clave proporcionado por el campo invisible. También se puede usar para listados que hacen referencia a dos tablas diferentes. En este caso, el código postal de una ciudad se almacena en una tabla separada de la tabla de la ciudad. El código postal se lee de una tabla que contiene tres campos: la clave principal, el código postal y una clave foránea para la ciudad. El cuadro combinado debe mostrar el código postal y la ciudad juntos.

```
Sub ShowText(oEvent As Object)
```

Este procedimiento se debe vincular al suceso de formulario *Tras el cambio de registro de datos*.

La macro se llama directamente desde el formulario. El suceso desencadenante es el cambio de datos en los cuadros combinados. Algunas variables ya se han declarado globalmente en un módulo separado y no se declaran aquí nuevamente.

```
Dim oForm As Object
Dim oFieldList As Object
Dim stFieldValue As String
Dim inCom As Integer
Dim stQuery As String
Dim stFieldValue AS String
```

En el formulario hay un control oculto desde el cual se pueden obtener los nombres de los cuadros combinados, estos cuadros combinados son procesados por la macro.

La *Información adicional* (Tag) de las propiedades del control oculto contiene esta lista de nombres de cuadros combinados, separados por comas. Los nombres se escriben en una matriz y todo el proceso tiene lugar dentro de un bucle que finaliza con `NEXT inCom`.

```
oForm = oEvent.Source
aComboboxes() = Split(oForm.getByName("combofields").Tag,",")
For inCom = LBound(aComboboxes) TO Ubound(aComboboxes)
  ...
Next inCom
```

El cuadro combinado se identifica con `oFieldList`. Para obtener el valor de la clave foránea, necesitamos la columna correcta en la tabla correspondiente. Se puede acceder a él utilizando el nombre del campo de la tabla `stFieldID`, que se almacena en la información adicional del cuadro combinado.

```
oFieldList = oForm.getByName(Trim(aComboboxes(inCom)))
stFieldID = oForm.getString(oForm.findColumn(oFieldList.Tag))
oFieldList.Refresh()
```

El cuadro combinado se vuelve a leer usando `Refresh()` en caso de que el contenido del campo haya cambiado por la entrada de datos nuevos.

La consulta necesaria para proporcionar el contenido visible al cuadro combinado se basa en el campo subyacente al control y el valor determinado por la clave foránea. Para que el código SQL sea utilizable, se procesa, eliminando cualquier operación de ordenación que pueda estar presente. Luego se realiza una comprobación para cualquier definición de relación (que comience con la palabra WHERE). De manera predeterminada, la función `InStr()` no distingue entre mayúsculas y minúsculas, por lo que cualquier combinación de mayúsculas y minúsculas están cubiertas. Si hay una relación, significa que la consulta contiene campos de dos tablas diferentes.

Necesitamos encontrar la tabla que proporciona la clave foránea para el enlace. La macro depende aquí del hecho de que la clave primaria en todas las tablas se llama *ID*.

Si no hay una relación definida, la consulta solo accede a una tabla. La información de la tabla se puede descartar y la condición se puede formular directamente utilizando el valor de la clave foránea.

```
If stFieldID <> "" Then
  stQuery = oFieldList.ListSource
  If InStr(stQuery,"order by") > 0 Then
    stSql = Left(stQuery, InStr(stQuery,"order by")-1)
  Else
    stSql = stQuery
  End If
  If InStr(stSql,"where") Then
    st = Right(stSql, Len(stSql)-InStr(stSql,"where")-4)
    If InStr(Left(st, InStr(st,"=")),"."ID"") Then
     a() = Split(Right(st, Len(st)-InStr(st,"=")-1),".")
    Else
     a() = Split(Left(st, InStr(st,"=")-1),".")
    End If
    stSql = stSql + "AND "+a(0)+"."ID" = "+stFieldID
  Else
    stSql = stSql + "WHERE "ID" = "+stFieldID
  End If
```

Cada campo y nombre de tabla tiene que referenciarse en la orden SQL con dos conjuntos de comillas. Basic normalmente interpreta las comillas como delimitadores de cadena de texto, por lo que no aparecen cuando el código se pasa a SQL. Duplicar las comillas asegura que se pase un conjunto. `""ID""` significa que se accederá al campo `"ID"` en la consulta, (el conjunto único de comillas que utiliza SQL).

La consulta almacenada en la variable `stSql` y tras verificar la conexión con la base de datos se lleva a cabo. Su resultado se guarda en `oResult`.

```
oDatasource = ThisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then
  oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
oResult = oSQL_Command.executeQuery(stSql)
```

El resultado de la consulta se lee en otro bucle. Al igual que cuando se hace una consulta en la interfaz, se pueden mostrar varios campos y registros. Pero la construcción de esta consulta requiere solo un resultado, que se encontrará en la primera columna (1) del conjunto de resultados de la consulta. Es el registro que proporciona el contenido visualizado del cuadro combinado. El contenido es texto (`getString()`), de ahí el comando `oResult.getString(1)`.

```
While oResult.next
  stFieldValue = oResult.getString(1)
Wend
```

El cuadro combinado ahora tiene que contener el valor del texto recuperado por la consulta.

```
        oFieldList.Text = stFieldValue
      Else
```

Si no hay ningún valor en el campo para la clave foránea `oField`, la consulta ha fallado y en el cuadro combinado se escribe una cadena sin texto.

```
        oFieldList.Text = ""
      End If
    Next inCom
  End Sub
```

Este procedimiento gestiona el contacto entre el cuadro combinado y la clave foránea disponible en un campo de la fuente de datos del formulario. Esto debería ser suficiente para mostrar los valores correctos en los cuadros combinados. El almacenamiento de nuevos valores requiere un procedimiento adicional.

Transferir un valor de clave foránea de un cuadro combinado a un campo numérico

Si se ingresa un nuevo valor en el cuadro combinado (propósito para el cual se construyó esta macro), la clave primaria correspondiente debe ingresarse en la tabla subyacente del formulario como una clave foránea.

```
  Sub TextSelectionSaveValue(oEvent As Object)
```

Este procedimiento se vincula al suceso de formulario: *Antes de la acción en el registro de datos*.

Después de que declarar las variables (no se muestran aquí), Se tiene que determinar exactamente qué suceso debe iniciar la macro. El suceso que hemos vinculado, hace llamada a dos implementaciones en sucesión. Es importante que el procedimiento identifique al formulario para obtener y procesar sus elementos. Se puede hacer con ambas implementaciones pero de diferentes maneras. Aquí se utiliza la implementación *OdatabaseForm*. Para más información sobre las implementaciones >consulte «Un suceso: varias implementaciones» en la página 406.

```
    If InStr(oEvent.Source.ImplementationName,"ODatabaseForm") Then
      ...
    End If
  End Sub
```

El siguiente bucle se construye con el mismo principio que el procedimiento empleado para mostrar texto en cuadros combinados:

```
    oForm = oEvent.Source
    aComboboxes() = Split(oForm.getByName("combofields").Tag,",")
    For inCom = LBound(aComboboxes) To Ubound(aComboboxes)
      ...
    Next inCom
```

El objeto `oFieldList` contiene el texto que se mostrará. Si el cuadro combinado está dentro de un control de tabla, no es posible acceder directamente desde el formulario. En ese supuesto, la información adicional del control oculto debe contener la ruta al campo mediante el cuadro combinado del control de tabla (control de tabla > campo). Con la función `Split` conseguiremos acceder al cuadro combinado.

```
    a() = Split(Trim(aComboboxes(inCom)),">")
      If Ubound(a) > 0 Then
        oFieldList = oForm.getByName(a(0)).getByName(a(1))
      Else
        oFieldList = oForm.getByName(a(0))
```

```
End If
```

A continuación, la consulta se lee desde el cuadro combinado y se divide en partes individuales. En un cuadro combinado sencillo, los elementos de información necesarios son el nombre del campo y el nombre de la tabla:

```
SELECT "Field" FROM "Table"
```

En algunos casos, esto podría aumentarse mediante una instrucción de ordenación. Siempre que se junten dos campos en el cuadro combinado, se requerirá más trabajo para separarlos.

```
SELECT "Field1"||' '||"Field2" FROM "Table"
```

Esta consulta une dos campos con un espacio entre ellos. Como el separador es un espacio, la macro lo buscará y dividirá el texto en dos partes en consecuencia. Naturalmente, esto solo funcionará de manera fiable si *Field1* no contiene espacios en el texto. Como ejemplo, si el primer nombre es "Anne Marie" y el apellido "Müller", la macro tratará a "Anne" como el primer nombre y "Marie Müller" como el apellido. Para esto, se debe utilizar un separador más adecuado, que pueda procesar la macro. En el caso de los nombres, esto podría ser "Apellido, Nombre de pila".

Las cosas se complican aún más si los dos campos provienen de tablas diferentes:

```
SELECT "Table1"."Field1"||' > '||"Table2"."Field2"
FROM "Table1", "Table2"
WHERE "Table1"."ID" = "Table2"."ForeignID"
ORDER BY "Table1"."Field1"||' > '||"Table2"."Field2" ASC
```

Los campos se tienen que separar uno de otro, se tiene que indicar la tabla a la que pertenece cada uno y determinar las claves foráneas correspondientes.

```
stQuery = oFieldList.ListSource
aFields() = Split(stQuery, """")
stContent = ""
For i=LBound(aFields)+1 To UBound(aFields)
```

El contenido de la consulta es despojado de lastre innecesario. Las partes se vuelven a montar en una matriz con una combinación de caracteres inusual como separador. FROM separa la visualización del campo visible de los nombres de las tablas. WHERE separa la condición de los nombres de las tablas. Las uniones mediante JOIN no son compatibles.

```
  If Trim(UCASE(aFields(i))) = "ORDER BY" Then
   Exit For
  ElseIf Trim(UCASE(aFields(i))) = "FROM" Then
   stContent = stcontent+" §§ "
  ElseIf Trim(UCASE(aFields(i))) = "WHERE" Then
   stContent = stcontent+" §§ "
  Else
  stContent = stContent+Trim(aFields(i))
  End If
Next i
aContent() = Split(stContent, " §§ ")
```

En algunos casos, el contenido de la visualización del campo visible proviene de diferentes campos:

```
aFirst() = Split(aContent(0),"||")
If UBound(aFirst) > 0 Then
  If UBound(aContent) > 1 Then
```

La primera parte contiene al menos dos campos. Los campos comienzan con un nombre de tabla. La segunda parte contiene dos nombres de tabla, que se pueden determinar a partir de la primera. La tercera parte contiene una relación con una clave foránea, separada por un igual (**=**):

```
aTest() = Split(aFirst(0),".")
NameTable1 = aTest(0)
NameTableField1 = aTest(1)
Erase aTest
stFieldSeperator = Join(Split(aFirst(1),"'"),"")
aTest() = Split(aFirst(2),".")
NameTable2 = aTest(0)
NameTableField2 = aTest(1)
Erase aTest
aTest() = Split(aContent(2),"=")
aTest1() = Split(aTest(0),".")
If aTest1(1) <> "ID" Then
   NameTab12ID = aTest1(1)
   IF aTest1(0) = NameTable1 Then
      Position = 2
   Else
      Position = 1
   End If
Else
   Erase aTest1
   aTest1() = Split(aTest(1),".")
   NameTab12ID = aTest1(1)
   If aTest1(0) = NameTable1 Then
      Position = 2
   Else
      Position = 1
   End If
End If
Else
```

La primera parte contiene dos nombres de campo sin nombres de tabla, posiblemente con separadores. La segunda los nombres de las tablas. No hay una tercera parte:

```
If UBound(aFirst) > 1 Then
   NameTableField1 = aFirst(0)
   stFieldSeparator = Join(Split(aFirst(1),"'"),"")
   NameTableField2 = aFirst(2)
Else
   NameTableField1 = aFirst(0)
   NameTableField2 = aFirst(1)
End If
NameTable1 = aContent(1)
End If
Else
```

Solo hay un campo de una tabla:

```
    NameTableField1 = aFirst(0)
    NameTable1 = aContent(1)
  End If
```

La longitud máxima de caracteres que puede tener una entrada viene dada por la función `ColumnSize`. El cuadro combinado no se puede usar para limitar el tamaño, ya que puede necesitar contener dos campos al mismo tiempo.

```
  LengthField1 = ColumnSize(NameTable1,NameTableField1)
  If NameTableField2 <> "" Then
    If NameTable2 <> "" Then
     LengthField2 = ColumnSize(NameTable2,NameTableField2)
    Else
     LengthField2 = ColumnSize(NameTable1,NameTableField2)
    End If
  Else
    LengthField2 = 0
  End If
```

El contenido del cuadro combinado se lee:

```
  stContent = oFieldList.getCurrentValue()
```

Los espacios iniciales y finales y los caracteres de control se eliminan si es necesario.

```
stContent = Trim(stContent)
If stContent <> "" Then
  If NameTableField2 <> "" Then
```

En algunos casos no se detecta el espacio, con las siguientes tres líneas se corrige este fallo

```
IF ASC(stFieldSeparator) = 32 Then
  stFieldSeparator = " "
End If
```

Si existe un segundo campo de tabla, el contenido del cuadro combinado tiene que dividirse. Para determinar dónde se producirá la división, usamos el separador de campo como argumento para la función.

```
  a_stParts = Split(stContent, stFieldSeparator, 2)
```

El último parámetro significa que el número máximo de partes es 2.

Dependiendo de qué entrada corresponde al campo 1 y cuál al campo 2, el contenido del cuadro combinado ahora se asigna a las variables individuales. `Position = 2` >sirve aquí como una señal de que la segunda parte del contenido representa el Campo 2.

```
  If Position = 2 Then
    stContent = Trim(a_stParts(0))
    If UBound(a_stParts()) > 0 Then
     stContentField2 = Trim(a_stParts(1))
    Else
     stContentField2 = ""
    End If
    stContentField2 = Trim(a_stParts(1))
  Else
    stContentField2 = Trim(a_stParts(0))
```

```
            If UBound(a_stParts()) > 0 Then
             stContent = Trim(a_stParts(1))
            Else
             stContent = ""
            End If
            stContent = Trim(a_stParts(1))
          End If
        End If
```

Puede suceder que con dos contenidos separables, el tamaño previsto del cuadro combinado (longitud del texto) no se ajuste a los campos de la tabla que deben ser guardados. Para los cuadros combinados que representan un solo campo, esto normalmente se maneja configurando adecuadamente el control de formulario. Aquí, por el contrario, necesitamos alguna forma de detectar estos errores. Se verifica la longitud máxima permitida del campo relevante.

```
  If (LengthField1 > 0 And Len(stContent) > LengthField1) Or
(LengthField2 > 0 And Len(stContentField2) > LengthField2) Then
```

Si la longitud del campo de la primera o segunda parte es demasiado grande, se almacena una cadena predeterminada en una de las variables. El carácter `Chr(13)` se usa para insertar un salto de línea.

```
    stmsgbox1 = "El campo " + NameTableField1 + " no debe exceder de
" + Field1Length + "caracteres de longitud." + Chr(13)
    stmsgbox2 = "El campo " + NameTableField2 + " no debe exceder de
" + Field2Length + "caracteres de longitud." + Chr(13)
```

Si ambos contenidos de campo son demasiado largos, se muestran ambos textos.

```
    If (LengthField1 > 0 And Len(stContent) > LengthField1) And
(LengthField2 > 0 And Len(stContentField2) > LengthField2) Then
      MsgBox("El texto introducido es demasiado largo. " + Chr(13) +
stmsgbox1 + stmsgbox2 + "Reduzca su longitud.",64,"Entrada
Inválida")
```

El procecimiento utiliza la función `MsgBox()` para presentar un mensaje. Esta función necesita como primer argumento una cadena de texto (mensaje a mostrar), luego, opcionalmente, un número (que determina el tipo de diálogo de mensaje que se muestra) y, finalmente, una cadena de texto opcional como título para la ventana. Por lo tanto, el diálogo del mensaje tendrá el título *Entrada Inválida* . El número `64` proporciona un icono con el símbolo de Información.

El siguiente código cubre cualquier otro caso de texto excesivamente largo que pueda surgir.

```
    ElseIf (Field1Length > 0 And Len(stContent) > Field1Length) Then
      MsgBox("El texto introducido es demasiado largo. " + Chr(13) +
stmsgbox1 + "Reduzca su longitud.",64,"Entrada Inválida")
    Else
      MsgBox("El texto introducido es demasiado largo. " + Chr(13) +
stmsgbox2 + "Reduzca su longitud.",64,"Entrada inválida")
    End If
  Else
```

Si no hay texto excesivamente largo, el procedimiento continua sin presentar el mensaje.

Ahora las entradas están enmascaradas para que las comillas que puedan estar presentes no generen un error.

```
            stContent = String_to_SQL(stContent)
            If stContentField2 <> "" Then
               stContentField2 = String_to_SQL(stContentField2)
            End If
```

Las primeras variables se asignan previamente y la consulta puede modificarlas posteriormente. Las variables `inID1` e `inID2` almacenan el contenido de los campos de clave primaria de las dos tablas.

Las variables se establecen en -1

Al iniciar variables numéricas (de tipo integer), Basic les asigna el valor de 0. Si una consulta no produce resultados el valor devuelto por Basic sería 0, lo que también podría indicar que la consulta ha tenido éxito. (la consulta devuelve una clave primaria 0). Al establecer la variable en -1 se desambigua esta salida, puesto que HSQLDB no puede establecer este valor para un campo de clave primaria con numeración automática.

A continuación, se establece la conexión con la base de datos, si no se hubiera establecido.

```
      inID1 = -1
      inID2 = -1
      oDatasource = ThisComponent.Parent.CurrentController
      If Not (oDatasource.isConnected()) Then
        oDatasource.connect()
      End If
      oConnection = oDatasource.ActiveConnection()
      oSQL_Command = oConnection.createStatement()
      If NameTableField2 <> "" And Not IsEmpty(stContentField2) And
NameTable2 <> "" Then
```

Si existe un segundo campo de tabla, se tiene que declarar una segunda dependencia.

```
        stSql = "SELECT ""ID"" FROM """ + NameTable2 + """ WHERE """ +
NameTableField2 + """='" + stContentField2 + "'"
        oResult = oSQL_Command.executeQuery(stSql)
          While oResult.next
           inID2 = oResult.getInt(1)
          Wend
        If inID2 = -1 Then
          stSql = "INSERT INTO """ + NameTable2 + """ (""" +
NameTableField2 + """) VALUES ('" + stContentField2 + "') "
          oSQL_Command.executeUpdate(stSql)
          stSql = "CALL IDENTITY()"
```

Si el contenido dentro del cuadro combinado no está presente en la tabla correspondiente, se inserta allí. Luego se lee el valor de la clave primaria resultante. Si está presente, la clave primaria >se lee de la misma manera. La función utiliza los campos de clave primaria generados automáticamente (`IDENTITY`).

```
          oResult = oSQL_Command.executeQuery(stSql)
           While oResult.next
              inID2 = oResult.getInt(1)
           Wend
        End If
```

La clave primaria para el segundo valor se almacena temporalmente en la variable `inID2` y luego se escribe como clave primaria en la tabla correspondiente al primer valor. En función de si el registro de la primera tabla ya estaba disponible, el contenido se guarda (`INSERT`) o se altera (`UPDATE`).

```
        If inID1 = -1 Then
          stSql = "INSERT INTO """ + NameTable1 + """ (""" +
    NameTableField1 + """,""" + NameTab12ID + """) VALUES ('" + stContent
    + "','" + inID2 + "') "
          oSQL_Command.executeUpdate(stSql)
```

Y el correspondiente *ID* se lee directamente:

```
          stSql = "CALL IDENTITY()"
          oResult = oSQL_Command.executeQuery(stSql)
           While oResult.next
              inID1 = oResult.getInt(1)
           Wend
```

La clave primaria para la primera tabla se tiene que leer nuevamente para que se pueda transferir a la tabla subyacente del formulario.

```
        Else
          stSql = "UPDATE """ + NameTable1 + """ SET """ + NameTab12ID
    + """='" + inID2 + "' WHERE """ + NameTableField1 + """ = '" +
    stContent + "'"
          oSQL_Command.executeUpdate(stSql)
        End If
      End If
```

En el caso de que ambos campos subyacentes al cuadro combinado estén en la misma tabla (por ejemplo, *Apellido* y *Nombre* de la tabla *Nombre*), se necesita una consulta diferente:

```
      If NameTableField2 <> "" And NameTable2 = "" Then
        stSql = "SELECT ""ID"" FROM """ + NameTable1 + """ WHERE """ +
    NameTableField1 + """='" + stContent + "' AND """ + NameTableField2 +
    """='" + stContentField2 + "'"
        oResult = oSQL_Command.executeQuery(stSql)
          While oResult.next
           inID1 = oResult.getInt(1)
          Wend
        If inID1 = -1 Then
```

y no existe una segunda tabla:

```
        stSql = "INSERT INTO """ + NameTable1 + """ (""" +
    NameTableField1 + """,""" + NameTableField2 + """) VALUES ('" +
    stContent + "','" + stContentField2 + "') "
        oSQL_Command.executeUpdate(stSql)
```

La clave primaria se lee de nuevo.

```
        stSql = "CALL IDENTITY()"
        oResult = oSQL_Command.executeQuery(stSql)
          While oResult.next
```

```
            inID1 = oResult.getInt(1)
          Wend
      End If
    End If
    IF NameTableField2 = "" Then
```

Ahora se considera el caso más simple: el segundo campo de la tabla no existe y la entrada aún no está presente en la tabla. En otras palabras, se ha ingresado un nuevo valor en el cuadro combinado.

```
      stSql = "SELECT ""ID"" FROM """ + NameTable1 + """ WHERE """ +
NameTableField1 + """='" + stContent + "'"
      oResult = oSQL_Command.executeQuery(stSql)
        While oResult.next
          inID1 = oResult.getInt(1)
        Wend
      If inID1 = -1 Then
```

Si no hay un segundo campo, el contenido del cuadro combinado se inserta como un nuevo registro.

```
        stSql = "INSERT INTO """ + NameTable1 + """ (""" +
NameTableField1 + """) VALUES ('" + stContent + "') "
        oSQL_Command.executeUpdate(stSql)
```

y el `ID` resultante se lee directamente.

```
        stSql = "CALL IDENTITY()"
        oResult = oSQL_Command.executeQuery(stSql)
          While oResult.next
           inID1 = oResult.getInt(1)
          Wend
      End If
    End If
```

El valor del campo de clave primaria tiene que determinarse, de modo que se pueda transferir a la parte principal del formulario.

A continuación, el valor de la clave primaria resultante de todos estos bucles se transfiere al campo invisible en la tabla principal y la base de datos subyacente. El campo de tabla vinculado al campo de formulario se obtiene mediante `BoundField` y mediante `updateInt` se inserta un número entero en este campo (consulte las definiciones de tipo numérico).

```
        oForm.updateLong(oForm.findColumn(oFeldList.Tag),inID1)
      End If
    ELSE
```

Si no se va a ingresar una clave primaria, porque no hubo entrada en el cuadro combinado o esa entrada se eliminó, el contenido del campo invisible también debe eliminarse. La instrucción `updateNull()` se usa para llenar el campo con la expresión para un campo vacío (`NULL`).

```
          oForm.updateNULL(oForm.findColumn(oFeldList.Tag),NULL)
        End If
      NEXT inCom
    End If
  End Sub
```

Función para medir la longitud de la entrada del cuadro combinado

La siguiente función proporciona el número de caracteres en la columna de la tabla respectiva, de modo que las entradas que son demasiado largas no se trunquen. Se elige una función para proporcionar valores de retorno, puesto que un procedimiento no proporciona ningún valor de retorno que pueda procesarse en otro lugar.

```
Function ColumnSize(Tablename As String, Fieldname As String) As
Integer
  oDatasource = ThisComponent.Parent.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If
  oConnection = oDataSource.ActiveConnection()
  oSQL_Command = oConnection.createStatement()
  stSql = "SELECT ""COLUMN_SIZE"" FROM
""INFORMATION_SCHEMA"".""SYSTEM_COLUMNS"" WHERE ""TABLE_NAME"" = '"
+ Tablename + "' AND ""COLUMN_NAME"" = '" + Fieldname + "'"
  oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
      i = oResult.getInt(1)
    Wend
  ColumnSize = i
End Function
```

Generar acciones de base de datos

```
Sub GenerateRecordAction(oEvent As Object)
```

Esta macro debe estar vinculada al suceso *Texto modificado* del cuadro combinado. Es necesario que en todos los casos en que se cambia el contenido ese cambio se almacene. Sin esta macro, no habría cambios en la tabla real que Base pudiera reconocer, ya que el cuadro combinado no está vinculado al formulario.

Esta macro altera directamente las propiedades del formulario:

```
  Dim oForm As Object
  oForm = oEvent.Source.Model.Parent
  oForm.IsModified = TRUE
End Sub
```

Esta macro no es necesaria para formularios que utilizan consultas para el contenido de cuadros combinados. Los cambios en los cuadros combinados se registran directamente.

Navegar de un formulario a otro

Se abrirá un formulario cuando ocurra un suceso particular.

En las propiedades de control de formulario, en la línea *Información adicional* (Tag), ingrese el nombre del formulario. También se puede ingresar más información aquí, y luego separarla mediante la función `Split()`.

```
Sub From_form_to_form(oEvent As Object)
  Dim stTag As String
  stTag = oEvent.Source.Model.Tag
  aForm() = Split(stTag, ",")
```

La matriz se declara y se llena con los nombres de los formularios, primero el formulario que se abrirá y, en segundo lugar, el formulario actual, que se cerrará después de que se haya abierto el otro.

```
    ThisDatabaseDocument.FormDocuments.getByName(
Trim(aForm(0)) ).open
    ThisDatabaseDocument.FormDocuments.getByName(
Trim(aForm(1)) ).close
End Sub
```

Si, en cambio, el otro formulario solo se debe abrir cuando el actual está cerrado, por ejemplo, cuando existe un formulario principal y todos los demás formularios se controlan a través de botones, la siguiente macro debe vincularse al formulario con **Herramientas > Personalizar > Sucesos > Documento cerrado**:

```
Sub Mainform_open
  ThisDatabaseDocument.FormDocuments.getByName( "Mainform" ).open
End Sub
```

Si los documentos del formulario se ordenan dentro del archivo ODB en directorios, el procedimiento para cambiar el formulario debe ser más extenso:

```
Sub From_form_to_form_with_folders(oEvent As Object)
  REM El formulario que se debe abrir se indica en primer lugar.
  REM Si un formulario está en una carpeta, use "/" para definir la
relación
  REM para que se pueda encontrar la subcarpeta.
  Dim stTag As String
  stTag = oEvent.Source.Model.Tag 'La etiqueta se ingresa en la
información adicional
  aForms() = Split(stTag, ",") 'Aquí viene primero el nombre del
formulario nuevo, luego el del formulario anterior
  aForms1() = Split(aForms(0),"/")
  aForms2() = Split(aForms(1),"/")
  If UBound(aForms1()) = 0 Then


ThisDatabaseDocument.FormDocuments.getByName( Trim(aForms1(0)) ).ope
n
  Else
    ThisDatabaseDocument.FormDocuments.getByName(
Trim(aForms1(0)) ).getByName( Trim(aForms1(1)) ).open
  End If
  If UBound(aForms2()) = 0 Then


ThisDatabaseDocument.FormDocuments.getByName( Trim(aForms2(0)) ).clo
se
  Else
    ThisDatabaseDocument.FormDocuments.getByName(
Trim(aForms2(0)) ).getByName( Trim(aForms2(1)) ).close
  End If
```

```
End Sub
```

Los documentos de formulario que se encuentran en un directorio se ingresan en el campo Información adicional como directorio / formulario. Tienen que convertirse a:

```
...getByName("Directorio").getByName("Form")
```

Listados jerárquicos

La configuración en un campo de listado tiene la intención de influir directamente en la configuración de otro. Para casos simples, esto ya se ha descrito anteriormente en la sección sobre filtrado de registros. Pero supongamos que el primer listado está destinado a afectar el contenido del segundo listado, que luego afecta el contenido de un tercer listado, y así sucesivamente.

Años
1
2
3
4
5
6
7
8
9
10
11
12
13

Clase
a
b
c
d
e
f
g

Nombre
Karl Müller
Evelyn Maier
Maria Gott
Eduard Abgefahren
Kurt Drechsler
Kunigunde Schimmel

Ejemplos de campos de lista para un ordenamiento jerárquico

En este ejemplo, el primer listado *Años* contiene todos los años escolares. El segundo, *Clase* las clases que en cada año están representadas por letras. El tercero *Nombre* contiene los nombres de los miembros de la clase.

En circunstancias normales, el listado Años mostraría los 13 años, el listado Clases todas las letras de clase y el listado Nombres todos los alumnos de la escuela.

Si se trata de listado jerárquicos, la elección de clases se restringe una vez que se ha seleccionado un año. Solo se muestran las letras de clase que están realmente presentes en ese año. Esto puede variar porque, si el número de alumnos aumenta, el número de clases en un año también podría aumentar. El último listado, Nombres, es muy restringido. En lugar de más de 1000 alumnos, mostraría solo 30.

Al principio, solo se puede seleccionar el año. Una vez hecho esto, la lista (restringida) de clases estará disponible. Solo al final se da la lista de nombres.

Si se modifica el listado Años, la secuencia debe comenzar de nuevo. Si solo se modifica el listado Clases, el número de año sigue siendo válido. Para crear dicha función, el formulario debe poder almacenar una variable intermedia. Esto tiene lugar en un control oculto.

La macro está vinculada a un cambio en el contenido de un listado: **Propiedades Listado > Sucesos > Modificado**. Las variables necesarias se almacenan en la información adicional del listado.

He aquí un ejemplo de la información adicional para el control oculto:

```
MainForm,Year,hidden control,Listbox_2
```

El formulario se llama *MainForm*. El listado actual se llama *Listbox1*. Este listado muestra el contenido del campo de la tabla Año y los siguientes listados se tienen que filtrar de acuerdo con

esta entrada. El control oculto se designa mediante `hidden_control` y la existencia de un segundo listado (*Listbox_2)* se pasa al procedimiento de filtrado.

```
Sub Hierarchical_control(oEvent As Object)
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm As Object
  Dim oFieldHidden As Object
  Dim oField As Object
  Dim oField1 As Object
  Dim stSql As String
  Dim acontent()
  Dim stTag As String
  oField = oEvent.Source.Model
  stTag = oField.Tag
  oForm = oField.Parent

  REM la etiqueta va en el campo de información adicional
  REM Contiene:
  REM 0. Nombre del campo para el filtrado de la tabla
  REM 1. Nombre del campo del control oculto que almacenará el valor
filtrado
  REM 2. Un posible listado adicional
  REM La etiqueta se lee del elemento que inicia la macro. La
variable se pasa
  REM al procedimiento y si es necesario a todos los listados
adicionales
  aFilter() = Split(stTag, ",")
  stFilter = ""
```

Una vez que se han declarado los variables, el contenido de la etiqueta se pasa a una matriz, para que se pueda acceder a los elementos individuales. Luego se declara el acceso a los distintos controles del formulario.

Se determina el listado que llamó a la macro y se lee su valor. Solo si este valor no está vacío (`NULL`) se combinará con el nombre del campo que se va a filtrar >para hacer una orden SQL (Año en nuestro ejemplo). De lo contrario, el filtro permanecerá vacío. Si los listados están destinados a filtrar un formulario, no hay ningún control oculto disponible. En este caso, el valor del filtro se almacena directamente en el formulario.

```
    If Trim(aFilter(1)) = "" Then
    If oField.getCurrentValue <> "" Then
      stFilter = """"+Trim(aFilter(0))
+"""='"+oField.getCurrentValue()+"'"
```

Si ya existe un filtro (por ejemplo, uno relacionado con *Listbox_2*, al que ahora se está accediendo), el nuevo contenido se adjunta al contenido anterior almacenado en el control oculto.

```
        If oForm.Filter <> ""
```

Solo puede darse cuando el mismo campo aún no se ha filtrado. Por ejemplo, si se está filtrando por Año, una repetición del filtro no encontrará registros adicionales para el listado *Nombre*. Solo

se puede encontrar a una persona en un año. Por lo tanto, se tiene que excluir la posibilidad de que el nombre del filtro ya se haya utilizado.

```
            And InStr(oForm.Filter, """"+Trim(aFilter(0))+"""='") = 0
    Then
            stFilter = oForm.Filter + " AND " + stFilter
```

Si existe un filtro y el campo que se utilizará para el filtrado ya está presente en el filtro, se tiene que eliminar el filtrado anterior en este nombre de campo y crear un nuevo filtro.

```
          ElseIf oForm.Filter <> "" Then
            stFilter = Left(oForm.Filter,
            InStr(oForm.Filter, """"+Trim(aFilter(0))+"""='")-1) +
    stFilter
          End If
        End If
```

Luego, el filtro se ingresa en el formulario. Este filtro también puede estar vacío si se seleccionó el primer listado y no tiene contenido.

```
        oForm.Filter = stFilter
        oForm.reload()
```

El mismo procedimiento >se ejecutará si el formulario no necesita ser filtrado inmediatamente. En este caso, el valor del filtro se almacena en el control oculto.

```
      Else
        oFieldHidden = oForm.getByName(Trim(aFilter(1)))
        If oField.getCurrentValue <>"" Then
          stFilter = """"+Trim(aFilter(0))
    +"""='"+oField.getCurrentValue()+"'"
          If oFieldHidden.HiddenValue <> ""
          And InStr(oFieldHidden.HiddenValue, """"+Trim(aFilter(0))
    +"""='") = 0 Then
            stFilter = oFieldHidden.HiddenValue + " AND " + stFilter
          ElseIf oFieldHidden.HiddenValue <> "" Then
            stFilter = Left(oFieldHidden.HiddenValue,
             InStr(oFieldHidden.HiddenValue, """"+Trim(aFilter(0))
    +"""='")-1) +               stFilter
          End If
        End If
        oFieldHidden.HiddenValue = stFilter
      End If
```

Si la información adicional tiene una entrada numerada 4 (la numeración comienza en 0), el siguiente listado debe establecerse en la entrada correspondiente del listado que hace la llamada.

```
      If UBound(aFilter()) > 1 Then
        oField1 = oForm.getByName(Trim(aFilter(2)))
        aFilter1() = Split(oField1.Tag,",")
```

Los datos necesarios para el filtrado se leen de la Información adicional (*Tag*) en el listado correspondiente. Desafortunadamente, no es posible escribir solo el código SQL nuevo en el listado y luego leer los valores del listado. Los valores correspondientes a la consulta tienen que escribirse directamente en el listado.

La creación del código se basa en que la tabla a la que se refiere el formulario es la misma a la que se refieren los listados. Tal listado no está diseñado para transferir claves foráneas a la tabla.

```
        If oField.getCurrentValue <> "" Then
          stSql = "SELECT DISTINCT """+Trim(aFilter1(0))+""" FROM
    """+oForm.Command+          """ WHERE "+stFilter+" ORDER BY
    """+Trim(aFilter1(0))+""""
          oDatasource = ThisComponent.Parent.CurrentController
          If Not (oDatasource.isConnected()) Then
            oDatasource.connect()
          End If
          oConnection = oDatasource.ActiveConnection()
          oSQL_Statement = oConnection.createStatement()
          oQuery_result = oSQL_Statement.executeQuery(stSql)
```

Los valores se leen en una matriz. La matriz se transfiere directamente al listado. Los índices correspondientes para la matriz se incrementan dentro de un bucle.

```
          inIndex = 0
          While oQuery_result.next
            ReDim Preserve aContent(inIndex)
            acontent(inIndex) = oQuery_result.getString(1)
            inIndex = inIndex+1
          WEnd
        Else
          aContent(0) = ""
        End If
        oField1.StringItemList = aContent()
```

El contenido del listado se ha modificado y tiene que ser leído de nuevo.

Luego, utilizando la propiedad de información adicional del listado que se ha actualizado, se vacía cada uno de los listados dependientes siguientes, iniciando un bucle para todos los listados siguientes hasta que se llegue a uno que no tenga un cuarto término en su información adicional.

```
        oField1.refresh()
        While UBound(aFilter1()) > 1
          Dim aLeer()
          oField2 = oForm.getByName(Trim(aFilter1(2)))
          Dim aFilter1()
          aFilter1() = Split(oField2.Tag,",")
          oField2.StringItemList = aEmpty()
          oField2.refresh()
        Wend
      End If
    End Sub
```

El contenido visible de los listados se almacena en `oField1.StringItemList`. Si algún valor adicional necesita ser almacenado para su transmisión a la tabla subyacente como una clave foránea,(habitual para los listados en formularios), este valor tiene que pasarse a la consulta por separado y luego almacenarse con `oField1.ValueItemList`.

Dicha extensión requiere variables adicionales, además de la tabla en la que se almacenarán los valores del formulario, la tabla de la que se extraen los contenidos del listado.

Se tiene que tener especial cuidado al formular la consulta de filtro.

```
    stFilter = """"+Trim(aFilter(1))+""""='"+oField.getCurrentValue()
+"'"
```

solo funcionará si la versión de LibreOffice es 4.1 o posterior, ya que es el valor que se debe almacenar lo que se proporciona como `CurrentValue()`, y no el valor que se muestra. Para asegurarse de que funciona en diferentes versiones, establezca la propiedad del listado **Datos > Campo enlazado > '0'.**

Ingresar horas con milisegundos

Para almacenar tiempos a una precisión de milisegundos se requiere un campo de marca de tiempo en la tabla, adaptado por separado por SQL para este propósito (consulte «Creación de tablas» en el «Capítulo 3»). Dicho campo se puede representar en un formulario mediante un campo formateado con el formato `MM:SS.00`. Sin embargo, en el primer intento de escribir en él, la entrada de registro fallará. Esto se puede corregir con el siguiente procedimiento, que debe estar vinculado a la propiedad del formulario **Sucesos > Antes de la acción en el registro de datos**:

```
SUB Timestamp
  Dim unoStmp As New com.sun.star.util.DateTime
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm As Object
  Dim oFeld As Object
  Dim stZeit As String
  Dim ar()
  Dim arMandS()
  Dim loNano As Long
  Dim inSecond As Integer
  Dim inMinute As Integer
  oDoc = thisComponent
  oDrawpage = oDoc.Drawpage
  oForm = oDrawpage.Forms.getByName("MainForm")
  oField = oForm.getByName("Time")
  stTime = oField.Text
```

Las variables se declaran primero. El resto del código se ejecuta solo cuando el campo Hora tiene contenido. De lo contrario, el mecanismo interno del formulario actuará para establecer el campo vacío (`NULL`).

```
    If stTime <> "" Then
      ar() = Split(stTime,".")
      loNano = CLng(ar(1)&"0000000")
      arMandS() = Split(ar(0),":")
      inSecond = CInt(arMandS(1))
      inMinute = Cint(arMandS(0))
```

La entrada en el campo Hora se divide en sus elementos:

Primero, la parte decimal se separa y se rellena a la derecha con caracteres nulos hasta un total de nueve dígitos. Un número tan alto solo puede almacenarse en una variable de tipo `Long`.

Luego, el resto del tiempo se divide en minutos y segundos, utilizando los dos puntos como separador, y estos se convierten en enteros.

```
With unoStmp
   .NanoSeconds = loNano
   .Seconds = inSecond
   .Minutes = inMinute
   .Hours = 0
   .Day = 30
   .Month = 12
   .Year = 1899
End With
```

Los valores de la marca de tiempo se asignan a la fecha estándar de LibreOffice (30.12.1899). Por supuesto, la fecha presente real se puede almacenar junto a ella.

Nota

Obtener y almacenar la fecha presente:

```
Dim now As Date
 now = Now()
 With unoStmp
    .NanoSeconds = loNano
    .Seconds = inSecond
    .Minutes = inMinute
    .Hours = Hour(now)
    .Day = Day(now)
    .Month = Month(now)
    .Year = Year(now)
 End With   oField.BoundField.updateTimestamp(unoStmp)
 End If
End Sub
```

Ahora la marca de tiempo que hemos creado se transfiere al campo usando `updateTimestamp` y se almacena en el formulario.

En tutoriales anteriores, los `NanoSeconds` se llamaban `HundrethSeconds`. Esto no coincide con la Interfaz de Programación (API) de LibreOffice y causará un mensaje de error.

Un suceso: varias implementaciones

Al usar formularios, puede ocurrir que una macro vinculada a un solo suceso se ejecute dos veces. Esto ocurre porque más de un proceso está vinculado simultáneamente a, por ejemplo, el almacenamiento de un registro modificado. Las diferentes causas de tal suceso se pueden determinar de la siguiente manera:

```
Sub Determine_eventcause(oEvent As Object)
  Dim oForm As Object
  oForm = oEvent.Source
  MsgBox oForm.ImplementationName
End Sub
```

Cuando se almacena un registro modificado, hay dos implementaciones involucradas llamadas `org.openoffice.comp.svx.FormController` y `com.sun.star.comp.forms.ODatabaseForm`. Con estos nombres, podemos asegurarnos de

que una macro solo se ejecute a través de su código una vez. Una ejecución duplicada generalmente causa una pausa (pequeña) en la ejecución del programa, pero puede también llevar a situaciones extrañas, como que un cursor devuelva dos registros en lugar de uno. Cada implementación solo permite comandos específicos, por lo que conocer el nombre de la implementación es importante.

Guardar con confirmación

Para alteraciones complicadas de registros, lo lógico es preguntar al usuario antes si el cambio realmente debe llevarse a cabo. Si la respuesta en el diálogo es *No*, se cancela el proceso de guardado, se descarta el cambio y el cursor permanece en el registro actual.

```
Sub Save_confirmation(oEvent As Object)
  Dim oFormFeature As Object
  Dim oFormOperations As Object
  Dim inAnswer As Integer
  oFormFeature = com.sun.star.form.runtime.FormFeature
  Select Case oEvent.Source.ImplementationName
    Case "org.openoffice.comp.svx.FormController"
      inAnswer = MsgBox("¿Desea modificar el registro?" ,4,
"Change_record")
      Select Case inAnswer
        Case 6' Yes, no hay acción posterior
        Case 7' No, se deshace el cambio del registro
         oFormOperations = oEvent.Source.FormOperations
         oFormOperations.execute(oFormFeature.UndoRecordChanges)
        Case Else
      End Select
    Case "com.sun.star.comp.forms.ODatabaseForm"
  End Select
End Sub
```

Hay dos momentos de activación con diferentes nombres de implementación. Estas dos implementaciones se distinguen en `SELECT CASE`. El código se ejecutará solo para la implementación de `FormController`. Esto se debe a que solo `FormController` tiene la variable `FormOperations`.

Además de *Sí* y *No*, el usuario también puede hacer clic en el botón *Cerrar*. Sin embargo, esto produce el mismo valor que No, es decir, **7**.

Si se navega por el formulario con la tecla de tabulación, el usuario solo ve el diálogo con el mensaje de confirmación. Sin embargo, los usuarios que usan la barra de navegación también verán un mensaje que indica que el registro no se alterará.

Clave primaria con año en curso y número

Cuando se preparan las facturas, los saldos anuales se ven afectados. Esto a menudo lleva a un deseo de separar las tablas de facturas de una base de datos por año y comenzar una nueva tabla cada año.

La siguiente solución macro utiliza un método diferente. Escribe automáticamente el valor del campo *ID* en la tabla, pero también tiene en cuenta el campo *año* que existe en la tabla como otra clave . La combinación de estas claves forman la clave primaria de la tabla. Por lo tanto, los campos siguientes deben aparecer en la tabla:

year	*ID*
2014	1
2014	2
2014	3
2015	1
2015	2

De esta forma, se obtiene una descripción general del año más fácilmente para documentos.

En la base de datos de ejemplo: `Example_serial_Number_Year.odb` puede comprobar su funcionamiento.

```
Sub Current_Date_and_ID
  Dim oDatasource As Object
  Dim oConnection As Object
  Dim oSQL_Command As Object
  Dim stSql As String
  Dim oResult As Object
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm As Object
  Dim oField1 As Object
  Dim oField2 As Object
  Dim oField3 As Object
  Dim inIDnew As Integer
  Dim inYear As Integer
  Dim unoDate
  oDoc = thisComponent
  oDrawpage = oDoc.drawpage
  oForm = oDrawpage.forms.getByName("MainForm")
  oField1 = oForm.getByName("fmt_year")
  oField2 = oForm.getByName("fmtID")
  oField3 = oForm.getByName("dat_date")
  If IsEmpty(oField2.getCurrentValue()) Then
    If IsEmpty(oField3.getCurrentValue()) Then
      unoDate = createUnoStruct("com.sun.star.util.Date")
      unoDate.Year = Year(Date)
      unoDate.Month = Month(Date)
      unoDate.Day = Day(Date)
      inYear = Year(Date)
    Else
      inYear = oField3.CurrentValue.Year
    End If
    oDatasource = ThisComponent.Parent.CurrentController
    If Not (oDatasource.isConnected()) Then
      oDatasource.connect()
    End If
    oConnection = oDatasource.ActiveConnection()
```

```
      oSQL_Command = oConnection.createStatement()
      stSql = "SELECT MAX( ""ID"" )+1 FROM ""orders"" WHERE ""year"" =
'"
        + inYear + "'"
      oResult = oSQL_Command.executeQuery(stSql)
      While oResult.next
        inIDnew = oresult.getInt(1)
      Wend
      If inIDnew = 0 Then
        inIDnew = 1
      End If
      oField1.BoundField.updateInt(inYear)
      oField2.BoundField.updateInt(inIDnew)
      If IsEmpty(oField3.getCurrentValue()) Then
        oField3.BoundField.updateDate(unoDate)
      End If
    End If
  End Sub
```

Todas las variables están declaradas. Los controles del formulario principal son accesibles. El resto del código solo se ejecuta si la entrada en el campo *fmtID* todavía está vacía. Si no se ha ingresado una fecha, se crea una estructura de fecha para que, fecha y año presentes se trasladen a los campos relevantes. Y se realiza una conexión a la base de datos, si aún no existe.

El valor más alto del campo *ID* para el año en curso se incrementa en 1. Si el conjunto de resultados está vacío, es que no hay entradas en el campo *ID*. En este punto, se podría ingresar 0 en el control *fmtID*, pero la numeración de las órdenes debe comenzar en 1, por lo que la variable *inIDnew* recibe el valor 1.

El valor devuelto para el año, *ID* y *fecha* (si no se ha ingresado ninguna fecha) se transfieren al formulario. En el formulario, los campos de claves primarias para el *ID* y el *año* están protegidos contra escritura. En consecuencia, solo se les puede dar valores usando esta macro.

Ampliación de tareas de bases de datos mediante macros

Crear una conexión a una base de datos

```
oDataSource = ThisComponent.Parent.DataSource
If Not oDataSource.IsPasswordRequired Then
  oConnection = oDataSource.GetConnection("","")
```

Sería posible proporcionar un nombre de usuario y una contraseña, si fuera necesario. En ese caso, los paréntesis contendrían ("Nombre de usuario", "Contraseña"). En lugar de incluir el nombre de usuario y una contraseña visibles, se abre el diálogo de protección de contraseñas:

```
Else
  oAuthentication =
createUnoService("com.sun.star.sdb.InteractionHandler")
  oConnection = oDataSource.ConnectWithCompletion(oAuthentication)
End If
```

Sin embargo, si un formulario dentro del mismo archivo Base está accediendo a la base de datos, solo necesita:

```
oDataSource = ThisComponent.Parent.CurrentController
If Not (oDataSource.isConnected()) Then
  oDataSource.connect()
End If
oConnection = oDataSource.ActiveConnection()
```

Aquí se conoce la base de datos, por lo que no es necesario un nombre de usuario y una contraseña, ya que estos están desactivados en la configuración básica de HSQLDB para la versión interna.

Para formularios fuera de Base, la conexión se realiza a través del primer formulario:

```
oDataSource = Thiscomponent.Drawpage.Forms(0)
oConnection = oDataSource.activeConnection
```

Copiar datos de una base de datos a otra

La base de datos interna es una base de datos de usuario único. Los registros se almacenan dentro del archivo `*.odb`. El intercambio directo de datos entre diferentes archivos de base de datos no estaba permitido pero, sin embargo, es posible mediante la exportación e importación.

Los archivos `*.odb` se configuran frecuentemente para permitir el intercambio automático de datos entre bases de datos.

El siguiente procedimiento puede ser útil.

Después de declarar las variables, la ruta a la base de datos actual se lee desde un botón en el formulario. El nombre de la base de datos está separado del resto de la ruta. El archivo de destino para los registros también está presente en esta carpeta. El nombre de este archivo se adjunta a la ruta para permitir que se realice una conexión a la base de datos de destino.

La conexión a la base de datos de origen se determina en relación con el formulario que contiene el botón: `ThisComponent.Parent.CurrentController`.

La conexión a la base de datos externa se configura utilizando `DatabaseContext` y la ruta.

```
Sub DataCopy
  Dim oDatabaseContext As Object
  Dim oDatasource As Object
  Dim oDatasourceZiel As Object
  Dim oConnection As Object
  Dim oConnection Ziel As Object
  Dim oDB As Object
  Dim oSQL_Command As Object
  Dim oSQL_CommandTarget As Object
  Dim oResult As Object
  Dim oResultTarget As Object
  Dim stSql As String
  Dim stSqlTarget As String
  Dim inID As Integer
  Dim inIDTarget As Integer
  Dim stName As String
  Dim stTown As String
  oDB = ThisComponent.Parent
  stDir = Left(oDB.Location,Len(oDB.Location)-Len(oDB.Title))
  stDir = ConvertToUrl(stDir & "TargetDB.odb")
```

```
  oDatasource = ThisComponent.Parent.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If
  oConnection = oDatasource.ActiveConnection()
  oDatabaseContext =
createUnoService("com.sun.star.sdb.DatabaseContext")
  oDatasourceTarget = oDatabaseContext.getByName(stDir)
  oConnectionTarget = oDatasourceTarget.GetConnection("","")
  oSQL_Command = oConnection.createStatement()
  stSql = "SELECT * FROM ""table"""
  oResult = oSQL_Command.executeQuery(stSql)
  While oResult.next
    inID = oResult.getInt(1)
    stName = oResult.getString(2)
    stTown = oResult.getString(3)
    oSQL_CommandTarget = oConnectionTarget.createStatement()
    stSqlTarget = "SELECT ""ID"" FROM ""table"" WHERE ""ID"" =
'"+inID+"'"
    oResultTarget = oSQL_CommandZiel.executeQuery(stSqlTarget)
    inIDZiel = - 1
    While oResultTarget.next
      inIDTarget = oResultTarget.getInt(1)
    Wend
    If inIDTarget = - 1 Then
      stSqlTarget = "INSERT INTO ""table""
(""ID"",""name"",""town"") VALUES
        ('"+inID+"','"+stName+"','"+stTown+"')"
      oSQL_CommandTarget.executeUpdate(stSqlZiel)
    End If
  Wend
End Sub
```

Las tablas completas de la base de datos de origen se leen e insertan, línea por línea, en las tablas de la base de datos de destino utilizando la conexión que se ha configurado. Antes de la inserción, se realiza una comprobación para ver si se ha establecido un valor para la clave primaria. Si es así, el registro no se copia.

También se puede organizar que, en lugar de copiar un nuevo registro, se actualice un registro existente. En todos los casos, esto asegura que la base de datos de destino contiene registros con la clave primaria correcta de la base de datos de origen.

Acceso a consultas

Es más fácil crear consultas en la interfaz gráfica de usuario que transferir su texto a macros, con la complicación adicional de las dobles comillas duplicadas para todos los nombres de tablas y campos.

```
Sub aQueryContent
  Dim oDatabaseFile As Object
  Dim oQuery As Object
```

```
  Dim stQuery As String
  oDatabaseFile = ThisComponent.Parent.CurrentController.DataSource
  oQuery = oDatabaseFile.getQueryDefinitions()
  stQuery = oQuery.getByName("Query").Command
  MsgBox stQuery
End Sub
```

Así se accede al contenido de un archivo `*.odb` desde un formulario. A la consulta se accede usando `getQueryDefinitions()`. El código SQL para la consulta está en su campo *Command*. Se puede obtener para utilizar la orden SQL dentro de una macro.

Cuando utiliza el código SQL de la consulta, debe tener cuidado de que el código no haga referencia a otra consulta. Lo que conlleva inevitablemente al mensaje de que la tabla de la base de datos es desconocida (aparentemente). Debido a esto, es más sencillo crear vistas a partir de consultas y luego acceder a las vistas en la macro.

Asegurar la base de datos

En algunas ocasiones puede ocurrir que el archivo `*.odb` se trunque inesperadamente, y de manera especial al crear una base de datos. Por eso, es recomendable guardar con cierta frecuencia la base de datos después de una edición, sobre todo cuando se utilizan Informes.

El archivo de base de datos puede dañarse por un fallo del sistema operativo en el momento en que la base de datos está en uso. Particularmente, cuando se termina el archivo Base, momento en el que el contendido de la base de datos tiene que escribirse en el archivo.

Además, existen los habituales supuestos de archivos que repentinamente no se pueden abrir, por ejemplo por un fallo del disco duro. Por lo tanto, es conveniente tener una copia de seguridad lo más actualizada posible. El estado de los datos no cambia mientras el archivo `*.odb` permanezca abierto.

Por esta razón, los procedimientos de seguridad se pueden vincular directamente a la apertura del archivo. Simplemente copie el archivo utilizando la ruta de respaldo proporcionada en **Herramientas > Opciones > LibreOffice > Rutas**. Esta macro sobrescribe la versión más antigua de la copia de seguridad después de un número específico de copias (`inMax`).

```
Sub Databasebackup(inMax As Integer)
  Dim oPath As Object
  Dim oDoc As Object
  Dim sTitle As String
  Dim sUrl_End As String
  Dim sUrl_Start As String
  Dim i As Integer
  Dim k As Integer
  oDoc = ThisComponent
  sTitle = oDoc.Title
  sUrl_Start = oDoc.URL
  Do While sUrl_Start = ""
    oDoc = oDoc.Parent
    sTitle = oDoc.Title
    sUrl_Start = oDoc.URL
  Loop
```

Si la macro se ejecuta al iniciar el archivo `*.odb`, `sTitle` y `sUrl_Start` serán correctos. En cambio, si la macro se ejecuta mediante un formulario, primero debe determinar si hay una URL

disponible para la copia de seguridad. Si la URL está vacía, se busca en un nivel superior (`oDoc.Parent`).

```
    oPath = createUnoService("com.sun.star.util.PathSettings")
    For i = 1 To inMax + 1
      If Not FileExists(oPath.Backup & "/" & i & "_" & sTitle) Then
        If i > inMax Then
          For k = 1 To inMax - 1 To 1 Step -1
          If FileDateTime(oPath.Backup & "/" & k & "_" & sTitle) <=
FileDateTime(oPath.Backup & "/" & k+1 & "_" & sTitle) Then
           If k = 1 Then
                i = k
                Exit For
           End If
          Else
           i = k + 1
           Exit For
          End If
          Next
        End If
        Exit For
      End If
    Next
    sUrl_End = oPath.Backup & "/" & i &"_" & sTitle
    FileCopy(sUrl_Start,sUrl_End)
End Sub
```

También puede hacer una copia de seguridad mientras Base se está ejecutando, siempre que los datos se puedan escribir en el archivo antes de que se lleve a cabo el procedimiento de copia de seguridad `Databasebackup`. Esto se puede hacer mediante una secuencia de tiempo programada o al presionar un botón en el formulario. Este borrado de memoria intermedia se maneja con el siguiente procedimiento:

```
Sub Write_data_out_of_cache
  Dim oData As Object
  Dim oDataSource As Object
  oData = ThisDatabaseDocument.CurrentController
  If Not ( oData.isConnected() ) Then oData.connect()
  oDataSource = oData.DataSource
  oDataSource.flush
End Sub
```

Si todo esto se inicia desde un solo botón en un formulario, se tiene que programar otro procedimiento que inicie ambos procedimientos:

```
Sub BackupNow
  Write_data_out_of_cache
  DatabaseBackup(10)
End Sub
```

Especialmente para una macro de seguridad, sería más lógico hacer que la macro fuera accesible a través de la barra de herramientas de la base de datos. Esto se hace en la ventana principal del archivo Base usando **Herramientas > Personalizar > Barras de herramientas**.

En el diálogo *Personalizar* que se muestra en la figura *222*, en debe usar el nombre del archivo de la base de datos como *Ámbito*, para que la *Función* se guarde en el archivo, que en este caso es `Media_with_Macros.odb`.

Para que siempre sea visible en la base de datos es recomendable seleccionar la barra de herramientas *Estándar* como *Destino,* puesto que está disponible en todas las ventanas de Base.

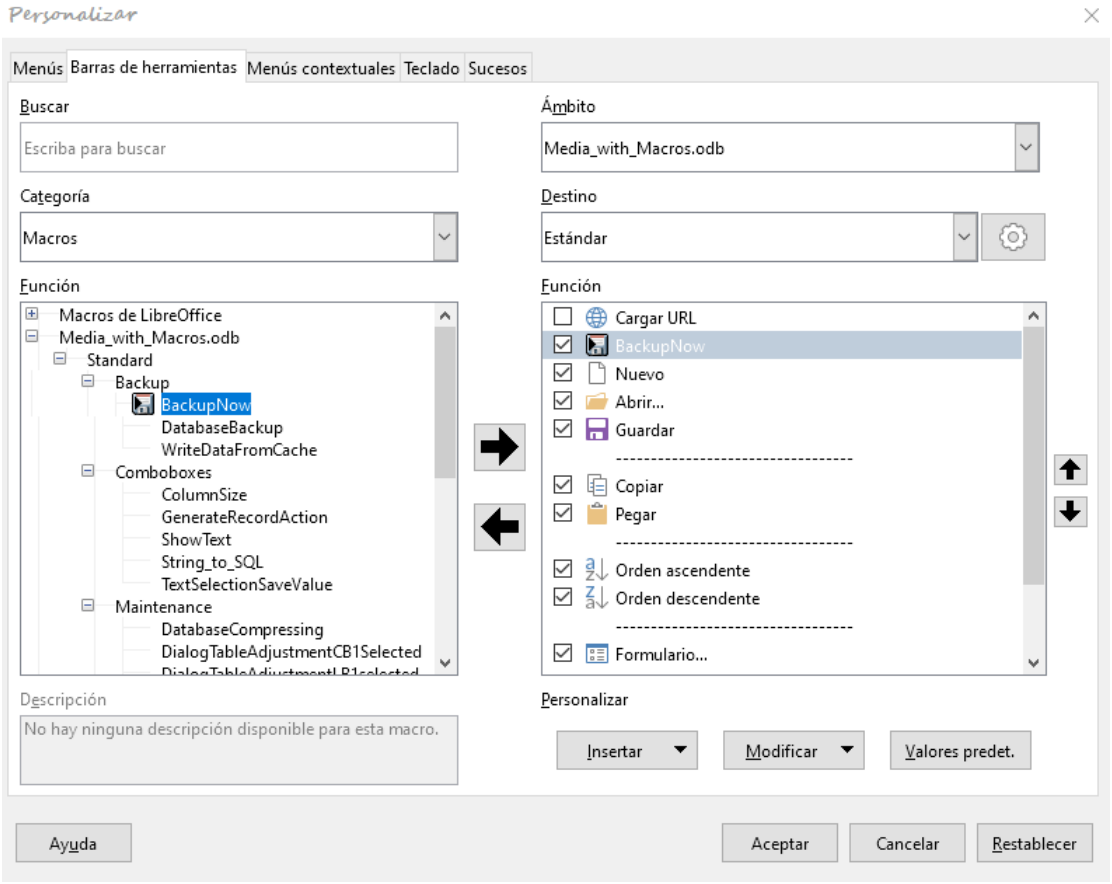
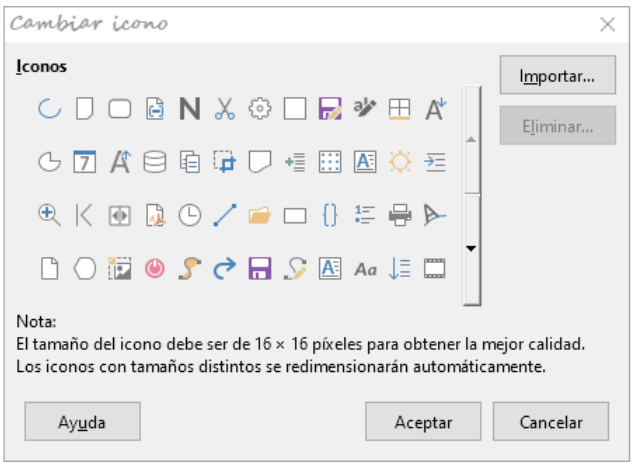


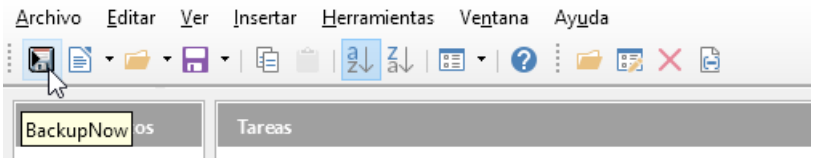
Figura 222: Personalización de la barra de herramientas

El diálogo ahora muestra funciones relevantes, en la lista de la derecha. Seleccione la *categoría >Macros* y elija el procedimiento *BackupNow*.

El comando estará disponible para su uso en la barra de herramientas. Para asignar un icono al comando, elija el nombre que ha asignado al nuevo botón y mediante **Modificar > Cambiar icono** se abrirá el siguiente diálogo.



Seleccione un icono adecuado para el botón (también puede crear y agregar su propio icono).



El icono ahora aparece en lugar del nombre del procedimiento. El nombre se convierte en una información emergente cuando se sitúa el ratón sobre el icono. Para ejecutar el procedimiento, simplemente haga clic en el icono en la barra de herramientas.

Compactar la base de datos

Esto es simplemente una orden SQL (SHUTDOWN COMPACT), que debe ejecutarse de vez en cuando, especialmente después de que se hayan eliminado muchos datos. La base de datos almacena datos nuevos, pero aún reserva el espacio de los datos eliminados. En los casos en que los datos se hayan alterado sustancialmente, necesita por tanto compactar la base de datos.

Nota

Desde LibreOffice versión 3.6, esta orden se ejecuta automáticamente para el HSQLDB interno cuando se cierra la base de datos. Por lo tanto, esta macro ya no es necesaria para una base de datos interna.

Una vez que se lleva a cabo la compactación, las tablas no son accesibles. Para poder acceder al contenido, el archivo tiene que ser reabierto. Por lo tanto, esta macro cierra el formulario desde el que se la llama. Lamentablemente, no puede cerrar el documento en sí mismo sin provocar una recuperación cuando se abre de nuevo. Por eso esta instrucción está comentada.

```
Sub Database_compaction
  Dim stMessage As String
  oDataSource = ThisComponent.Parent.CurrentController ' Accesible
desde el formulario
  If Not (oDataSource.isConnected()) Then
    oDataSource.connect()
  End If
  oConnection = oDataSource.ActiveConnection()
  oSQL_Statement = oConnection.createStatement()
  stSql = "SHUTDOWN COMPACT"' La base de datos se está compactando y
cerrando
  oSQL_Statement.executeQuery(stSql)
```

```
  stMessage = "Se va a compactar la base de datos." + Chr(13) + "Se
cerrará el formulario."
  stMessage = stMessage + Chr(13) + "Después de esto, la base de
datos debe cerrarse"
  stMessage = stMessage + Chr(13) + "Solo tendrá acceso a la base de
datos después de la reapertura del archivo de base de datos ."
  MsgBox stMessage

ThisDatabaseDocument.FormDocuments.getByName( "Maintenance" ).close
  REM El cierre del archivo de base de datos mediante el siguiente
comando provoca una operación de recuperación al abrirla nuevamente.
Por eso se comenta el comando de cierre.
' ThisDatabaseDocument.close(True)
End Sub
```

Disminuir el índice de la tabla para los campos automáticos

Si se eliminan muchos datos de una tabla, a los usuarios a menudo les preocupa que la secuencia de claves primarias generadas automáticamente continúe aumentando desde el último número asignado en lugar de comenzar a partir del último valor más alto de los registros que quedan (si en una tabla tenemos 60 registros y se borran los 10 últimos, el siguiente registro que se inserte tendrá el número 61, en lugar del esperado 51). El siguiente procedimiento lee el valor más alto existente del campo *ID* en una tabla y establece el siguiente valor sumando 1 a este máximo.

Si el campo de la clave primaria tiene un nombre distinto de *ID*, se tiene que editar la macro para utilizar ese nombre.

```
Sub Table_index_down(stTable As String)
  REM Se establece el valor siguiente de la clave primaria ID con el
mínimo valor posible.
  Dim inCount As Integer
  Dim inSequence_Value As Integer
  oDataSource = ThisComponent.Parent.CurrentController ' Accessible
mediante el formulario
  If Not (oDataSource.isConnected()) Then
    oDataSource.connect()
  End If
  oConnection = oDataSource.ActiveConnection()
  oSQL_Statement = oConnection.createStatement()
  stSql = "SELECT MAX(""ID"") FROM """+stTable+""""" ' Se obtiene el
valor más alto de "ID"
  oQuery_result = oSQL_Statement.executeQuery(stSql) ' Se inicia la
consulta y el valor de retorno se almacena en la variable
oQuery_result
  If Not IsNull(oQuery_result) Then
    While oQuery_result.next
    inCount = oQuery_result.getInt(1) ' Se lee el primer campo de
datos
    Wend ' siguiente registro, en este caso ninguno, ya que solo
existe un registro
```

```
    If inCount = "" Then ' Si la tabla está vacía no se obtiene
ningún valor, y el valor >más alto se establece en -1
      inCount = -1
    End If
    inSequence_Value = inCount+1 ' El valor más alto se incrementa
en 1
REM Se prepara una nueva orden para la base de datos. La ID del
siguiente registro comenzará ahora desde inCount + 1.
REM Esta operación no tiene valor de retorno, ya que no se está
leyendo ningún registro
    oSQL_statement = oConnection.createStatement()
    oSQL_statement.executeQuery("ALTER TABLE """ + stTable + """
ALTER COLUMN ""ID"" RESTART WITH " + inSequence_Value + "")
  End If
End Sub
```

Imprimir desde Base

La forma estándar de imprimir un documento desde Base es usar un informe. Alternativamente, las tablas y consultas se pueden copiar y preparar su impresión desde Calc. Por supuesto, también es posible la impresión directa de un formulario desde la pantalla.

Imprimir un informe desde un formulario interno

Normalmente, la generación de informes se realiza desde la interfaz Base de usuario. Un clic en el nombre del informe inicia la preparación del informe. Sería más fácil, por supuesto, si el informe se pudiera lanzar directamente desde un formulario.

```
Sub Reportlaunch
  ThisDatabaseDocument.ReportDocuments.getByName("Report").open
End Sub
```

Se accede a todos los informes por nombre desde su contenedor `ReportDocuments`. Se abren con `Open`. Si un informe está vinculado a una consulta que se filtra a través del formulario, este método permite imprimir el registro seleccionado.

Lanzar, formatear, imprimir directamente y cerrar un informe

Sería aún mejor si el informe se pudiera enviar directamente a la impresora. La siguiente combinación de procedimientos agrega algunos pequeños detalles. Primero selecciona el registro activo en el formulario, vuelve a formatear el informe para que los cuadros de texto se configuren automáticamente para la altura correcta y luego inicia el informe. Finalmente, el informe se imprime y, opcionalmente, se guarda como un *pdf*. Y todo esto sucede casi por completo en segundo plano, ya que el informe se oculta después de que se abre el formulario y se cierra tras la impresión. Andrew Pitonyak, Thomas Krumbein y Lionel Elie Mamane hicieron sugerencias para los diversos procedimientos.

```
Sub ReportStart(oEvent As Object)
  Dim oForm As Object
  Dim stSql As String
  Dim oDatasource As Object
  Dim oConnection As Object
  Dim oSQL_command As Object
  Dim oReport As Object
```

```
  Dim oReportView As Object
  oForm = oEvent.Source.model.parent
  stSql = "UPDATE ""Filter"" SET ""Integer"" = '" +
    oForm.getInt(oForm.findColumn("ID")) + "' WHERE ""ID"" = TRUE"
  oDatasource = ThisComponent.Parent.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If
  oConnection = oDatasource.ActiveConnection()
  oSQL_command = oConnection.createStatement()
  oSQL_command.executeUpdate(stSql)
  oReport =
ThisDatabaseDocument.ReportDocuments.getByName("Reportname").open
  oReportView = oReport.CurrentController.Frame.ContainerWindow
  oReportView.Visible = False
  ReportLineHeightAuto(oReport)
End Sub
```

El procedimiento `ReportStart` está vinculado a un botón en el formulario. Usando este botón, se puede leer la clave primaria del registro seleccionado. Desde el suceso que inicia la macro, podemos llegar al formulario (`oForm`). El nombre del campo de clave primaria en este caso es *ID*. Usando `oForm.getInt(oForm.findColumn("ID"))`, la clave se lee desde el campo como un entero. Este valor se almacena en una tabla de filtro. La tabla de filtro controla una consulta para garantizar que solo se utilizará el registro seleccionado para el informe.

El informe se puede abrir sin referencia al formulario. Se hace accesible como un objeto (`oReport`). La ventana del informe se vuelve invisible. Desafortunadamente, no puede ocultarse durante la activación, por lo que aparece brevemente y se llena con el contenido apropiado en segundo plano.

A continuación, se inicia el procedimiento `ReportLineHeightAuto` que recibe como argumento una referencia al informe abierto.

La altura de la línea del registro se puede configurar automáticamente en el momento de la impresión. Si hay demasiado texto en un campo en particular, el texto se trunca y el resto se indica con un triángulo rojo. Cuando esto no funciona, el siguiente procedimiento garantizará que en todas las tablas con el nombre *Detail*, se active el control automático de altura.

```
Sub ReportLineHeightAuto(oReport As Object)
  Dim oTables As Object
  Dim oTable As Object
  Dim inT As Integer
  Dim inI As Integer
  Dim oRows As Object
  Dim oRow As Object
  oTables = oReport.getTextTables()
  For inT = 0 TO oTables.count() - 1
    oTable = oTables.getByIndex(inT)
    If Left$(oTable.name, 6) = "Detail" Then
      oRows = oTable.Rows
      For inI = 0 To oRows.count - 1
        oRow = oRows.getByIndex(inI)
```

```
            oRow.IsAutoHeight = True
          Next inI
        End If
      Next inT
      PrintCloseReport(oReport)
    End Sub
```

Cuando se crea el informe, se debe tener cuidado de que todos los campos en la misma línea de la sección *Detail* tengan la misma altura. De lo contrario, el control automático de altura puede establecer una línea a doble altura.

Una vez que en todas las tablas con el nombre *Detail* se ha establecido el control automático de altura, el informe se envía a la impresora mediante el procedimiento `PrintCloseReport`.

La matriz *Props* contiene los ajustes de los valores asociados con la impresión del documento. Para el comando de impresión, es necesario el nombre de la impresora de salida. El informe debe permanecer abierto hasta que se complete la impresión. Esto se garantiza dando el nombre de la impresora y `Wait` como argumentos.

```
Sub PrintCloseReport(oReport As Object)
  Dim Props
  Dim stPrinter As String
  Props = oReport.getPrinter()
  stPrinter = Props(0).value
  Dim arg(1) As New com.sun.star.beans.PropertyValue
  arg(0).name = "Name"
  arg(0).value = "<" & stPrinter & ">"
  arg(1).name = "Wait"
  arg(1).value = True
  oReport.print(arg())
  oReport.close(true)
End Sub
```

Solo cuando la impresión se ha enviado completamente a la impresora se cierra el documento.

Para conocer la configuración de la impresora, consulte las secciones «Impresora» y «Configuración de impresión» de la ayuda.

Si, en lugar de (o además de) una copia impresa, desea obtener un una copia de seguridad en formato *pdf* del documento, se puede usar el método `storeToURL()`:

```
Sub ReportPDFstore(oReport As Object)
  Dim stUrl As String
  Dim arg(0) As New com.sun.star.beans.PropertyValue
  arg(0).name = "FilterName"
  arg(0).value = "writer_pdf_Export"
  stUrl = "file:///...."
  oReport.storeToURL(stUrl, arg())
End Sub
```

La URL tiene que ser una dirección URL completa (`file:///Ruta/Nombre_Archivo`).

Si además esta dirección está vinculada a un registro permanente de la base de datos o del documento impreso, como un número de factura se evita que la próxima impresión sobrescriba el archivo PDF de copia de seguridad.

Imprimir informes desde un formulario externo

Hay problemas cuando se utilizan formularios externos. Los informes se encuentran dentro del archivo *.odb y no están disponibles mediante el navegador de origen de datos.

```
Sub Reportstart(oEvent As Object)
  Dim oFeld As Object
  Dim oForm As Object
  Dim oDocument As Object
  Dim oDocView As Object
  Dim Arg()
  oField = oEvent.Source.Model
  oForm = oField.Parent
  sURL = oForm.DataSourceName
  oDocument = StarDesktop.loadComponentFromURL(sURL, "_blank", 0,
Arg() )
  oDocView = oDocument.CurrentController.Frame.ContainerWindow
  oDocView.Visible = False
  oDocument.getCurrentController().connect
  Wait(100)
  oDocument.ReportDocuments.getByName("Report").open
  oDocument.close(True)
End Sub
```

El informe se inicia desde un botón en el formulario externo que traslada al formulario la ruta del archivo: `oForm.DataSourceName`. El archivo se abre mediante `loadComponentFromUrl`, y este debe permanecer en segundo plano, por lo que se accede a la vista del documento y la interfaz se establece en `Visible = False`. En principio, esto debería haberse hecho directamente usando la lista de argumentos `Arg()`, pero lamentablemente, esto no produce el resultado correcto.

El informe no se puede llamar inmediatamente desde el documento abierto, ya que la conexión aún no está establecida. (El informe aparece con un fondo gris y LibreOffice se bloquea). Con una breve espera de aproximadamente 100 milisegundos se puede resolver el problema.(Es posible que necesite hacer pruebas para determinar el tiempo mínimo de espera).

Después se lanza el informe, como el informe estará en un archivo de texto separado, el archivo `*.odb` abierto puede cerrarse nuevamente. El método `oDocument.close(True)` pasa esta instrucción al archivo `*.odb`, que solo se cerrará cuando ya no esté activo, es decir, cuando no se pasen más registros al informe.

Se puede iniciar un acceso similar desde los formularios dentro del archivo `*.odb`, pero en este caso el documento no debe cerrarse.

Utilizando macros combinadas con la función de combinación de correspondencia o cuadros de texto, puede obtener impresiones de buena calidad bastante más rápido que con el Generador de informes.

Combinación de correspondencia desde Base

Desde Base se puede hacer una *Combinación de correspondencia* mediante informes, pero los cuadros de texto de los informes son de uso limitado. Para crear una carta modelo de cierta calidad se puede utilizar Writer.

No es necesario abrir Writer, personalizar la carta y luego imprimir desde este programa, todo este proceso se puede automatizar mediante una macro y una plantilla.

```
Sub MailmergePrint
  Dim oMailMerge As Object
  Dim aProps()
  oMailMerge = CreateUnoService("com.sun.star.text.MailMerge")
```

La base de datos tiene que estar registrada en LibreOffice, y el nombre que se empleó para registrarla será el que se utilice para acceder a los datos. No es necesario que el nombre sea el mismo que el del archivo que se utilice para la combinación de correspondencia. En este ejemplo se utilizará Addresses como el nombre registrado de la base de datos

```
  oMailMerge.DataSourceName = "Addresses"
```

La ruta al archivo de salida (`mailmerge.odt`) debe formatearse para poder usarlo en LibreOffice en este ejemplo se utiliza, una ruta absoluta en un sistema Linux .

```
  oMailMerge.DocumentURL =
ConvertToUrl("home/user/Documentos/mailmerge.odt")
```

Se establece el tipo de comando: 0 representa una tabla, 1 una consulta y 2 una orden SQL.

```
  oMailMerge.CommandType = 1
```

Se ha elegido una consulta con el nombre *MailmergeQuery*.

```
  oMailMerge.Command = "MailmergeQuery"
```

Se utiliza un filtro para determinar qué registros se utilizarán para la impresión de la correspondencia. Este filtro podría, especificarse utilizando un control de formulario y acceder a él desde la macro. El uso de la clave primaria de un registro puede hacer que se imprima un solo documento.

En este ejemplo, se selecciona el campo *Gender* en *MailmergeQuery* y luego se buscan registros que tengan *m* en este campo.

```
  oMailMerge.Filter = """Gender""='m'"
```

Los tipos de salida disponibles son Impresora (1), Archivo (2) y Correo (3). Utilizaremos un archivo de salida para las pruebas que guardaremos en una ruta establecida. Por cada registro se creará un archivo y para evitar la sobreescritura utilizaremos el campo *Surname* (Apellido) que se agregará al nombre del archivo de salida.

```
  oMailMerge.OutputType = 2
  oMailMerge.OutputUrl = ConvertToUrl("home/user/Documents")
  oMailMerge.FileNameFromColumn = True
  oMailMerge.Filenameprefix = "Surname"
  oMailMerge.execute(aProps())
End Sub
```

Si el filtro se proporciona con sus datos a través del formulario posibilita la combinación de correspondencia sin abrir Writer.

Imprimir mediante campos de texto

Crearemos una plantilla en Writer para personalizar la carta modelo Para completar los datos a partir de la base de datos se deben usar campos, Estos campos se insertan usando **Insertar > Campo > Más campos > Funciones > Marcador de posición**.

Los marcadores de posición deben tener los mismos nombres que los nombres de los campos en la base de datos o consulta desde la que se llama a la macro y el tipo de marcador para un caso sencillo debe ser *Texto*.

Se debe indicar la ruta de la plantilla para que el procedimiento para crear las cartas rellene los marcadores de posición en la plantilla con el contenido de el/los registros elegidos por la consulta.

La base de datos de ejemplo `Example_database_mailmerge_direct.odb` muestra cómo se puede generar una factura completa con la ayuda de campos de texto y el acceso a una tabla preparada dentro de la plantilla de documento. Esta manera de creación de documentos no tiene las limitaciones de altura de los campos que nos podemos encontrar en un informe creado con el generador de informes.

Aquí está parte del código, suministrado principalmente por DPunch:

http://de.openoffice.info/viewtopic.php?f=8&t=45868#p194799

```
Sub Filling_Textfields
  oForm = thisComponent.Drawpage.Forms.MainForm
  If oForm.RowCount = 0 Then
    MsgBox "Registro no disponible para la impresión"
    Exit Sub
  End If
```

El formulario principal está activo. El botón que inicia la macro también podría usarse para indicar el formulario. La macro comprueba que el formulario realmente contenga datos imprimibles.

```
  oColumns = oForm.Columns
  oDB = ThisComponent.Parent
```

El acceso directo a la URL de la plantilla desde el formulario no es posible. Tiene que hacerse utilizando la referencia de nivel superior a la base de datos.

```
  stDir = Left(oDB.Location,Len(oDB.Location)-Len(oDB.Title))
```

Con esta instrucción obtenemos la URL del directorio donde se aloja la base de datos (sin el nombre del archivo de Base.

```
  stDir = stDir & "Plantilla_con_Campos.ott" 'Se indica la plantilla y se
abre creando un nuevo documento.
  Dim args(0) As New com.sun.star.beans.PropertyValue
  args(0).Name = "AsTemplate"
  args(0).Value = True
  oNewDoc = StarDesktop.loadComponentFromURL(stDir,"_blank",0,args)
```

Los campos de texto se rellenan con los datos

```
  oTextfields = oNewDoc.Textfields.createEnumeration
  Do While oTextfields.hasMoreElements
    oTextfield = oTextfields.nextElement
    If
oTextfield.supportsService("com.sun.star.text.TextField.JumpEdit")
Then
      stColumnname = oTextfield.PlaceHolder
```

El marcador de posición `oTextfield.PlaceHolder` representa el campo de texto.

```
      If oColumns.hasByName(stColumnname) Then
```

Al especificar el mismo nombre del campo de texto en el documento que en de la columna del conjunto de datos, el contenido de la base de datos se transfiere al campo correspondiente del documento de texto. Se rellenan los todos datos mediante el bucle `Do While`.

```
            inIndex = oForm.findColumn(stColumnname)
            oTextfield.Anchor.String = oForm.getString(inIndex)
          End If
        End If
      Loop
    End Sub
```

Llamar a aplicaciones externas a LibreOffice

Mediante macros también es posible hacer llamadas a programas externos a LibreOffice para abrir archivos, >seguir los enlaces de Internet o iniciar un programa de correo electrónico. Para esta sección vea la base de datos de ejemplo `Example_Mail_File_activate.odb`.

Llamada a aplicaciones para abrir archivos

Este procedimiento permite, con un solo clic en un botón, abrir el programa que está asociado al tipo de archivo en el sistema operativo. Se utiliza en el formulario *File_adtivate*

```
SUB File_activate
  DIM oDoc AS OBJECT
  DIM oDrawpage AS OBJECT
  DIM oForm AS OBJECT
  DIM oField AS OBJECT
  DIM oShell AS OBJECT
  DIM stField AS STRING
  oDoc = thisComponent
  oDrawpage = oDoc.Drawpage
  oForm = oDrawpage.Forms.getByName("form")
  oField = oForm.getByName("fileselection")
  REM Leer el nombre de archivo elegido por el control de selección
  stField = oField.Text
  REM Iniciar la llamada al programa mediante la conexión con el
sistema operativo
  oShell =
createUnoService("com.sun.star.system.SystemShellExecute")
  stField = convertToUrl(stField)
  oShell.execute(stField,,0)
END SUB
```

En el ejemplo se utiliza el control de selección de archivo *fileselection* para elegir un archivo y un botón para iniciar el procedimiento.

Llamada a un programa en función del texto introducido por el usuario

Este procedimiento lanzará el programa asociado al tipo de archivo, el programa de correo electrónico o el navegador web en función del contenido del texto ingresado por el usuario. El siguiente procedimiento se utiliza en el formulario *Mail_Website_activate*

```
Sub Website_Mail_activate
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm As Object
  Dim oField As Object
```

```
  Dim oShell As Object
  Dim stField As String
  oDoc = thisComponent
  oDrawpage = oDoc.Drawpage
  oForm = oDrawpage.Forms.getByName("form")
  oField = oForm.getByName("url_mail")
```

Se lee el contenido del control *url_mail*. Puede ser una dirección web que comience, una dirección de correo electrónico o una ruta a un documento (por ejemplo, una imagen o un archivo almacenado externamente).

```
  stFeld = oField.Text
  If stField = "" Then
    Exit Sub
  End If
```

Al pulsar el botón, si el control de cuadro de texto está vacío, el procedimiento se interrumpe, en caso de que tenga contenido se busca un carácter `@` Esto indicaría una dirección de correo electrónico. El programa de correo electrónico debe iniciarse para enviar correo a esta dirección.

```
If InStr(stField,"@") Then
stField = "mailto:"+stField
```

Si no hay un cáracter `@`, se busca el identificativo de una URL. Si comienza con `http://` se trata de un recurso de Internet que debe buscarse con un navegador web. De lo contrario, la ruta comenzará con el término `file:///` que corresponde a un archivo en el sistema.

```
  ElseIf InStr(stFeld,"http://") Then
    stFeld = convertToUrl(stField)
  Else
    stField = "file:///"+stField
End If
```

Se busca el programa asignado por el sistema operativo a dichos archivos. Para la palabra clave `'mailto:'` es el programa de correo, para `'http://'` el navegador, en cualquier otro caso el sistema tiene que decidir en función del tipo de archivo y su configuración.

```
  oShell =
createUnoService("com.sun.star.system.SystemShellExecute")
  oShell.execute(stField,,0)
End Sub
```

Llamada a un programa de correo con contenido predefinido

El ejemplo anterior se puede ampliar para iniciar un programa de correo con un asunto y contenido predefinidos. El siguiente procedimiento se utiliza en el formulario *Mail_activate*

El programa de correo se inicia usando `'mailto:recipient?subject= &body= &cc= &bcc= '`. Las dos últimas entradas no están presentes en el formulario. Los archivos adjuntos no están previstos en la definición de `'mailto'` pero a veces funciona `'attachment='`.

```
Sub Mail_activate
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm As Object
  Dim oField1 As Object
```

```
  Dim oField2 As Object
  Dim oField3 As Object
  Dim oField4 As Object
  Dim oShell As Object
  Dim stField1 As String
  Dim stField2 As String
  Dim stField3 As String
  Dim stField4 As String
  oDoc = thisComponent
  oDrawpage = oDoc.Drawpage
  oForm = oDrawpage.Forms.getByName("form")
  oField1 = oForm.getByName("mail_to")
  oField2 = oForm.getByName("mail_subject")
  oField3 = oForm.getByName("mail_body")
  stField1 = oField1.Text
  If stField1 = "" Then
    MsgBox "Sin dirección de correo electrónico" & Chr(13) & _
      "No se iniciará el programa de correo electrónico" , 48,
"Enviar mensaje"
    Exit Sub
  End If
```

La conversión a URL en el asunto (*mail_subject*) y el cuerpo del mensaje (*mail_body*)es necesaria para evitar que caracteres especiales y saltos de línea causen errores en el código. Sin embargo, >esta conversión añade `file:///` al principio del texto que se elimina con la función `Mid()`.

```
  stField2 = Mid(ConvertToUrl(oField2.Text),9)
  stField3 = Mid(ConvertToUrl(oField3.Text),9)
```

En contraste a una simple llamada a la ejecución de un programa, Los detalles de la invocación al correo se dan aquí como parte de la llamada de ejecución.

```
  oShell =
createUnoService("com.sun.star.system.SystemShellExecute")
  oShell.execute("mailto:" + stField1 + "?subject=" + stField2 +
"&body=" + stField3,,0)
End Sub
```

Nota

El envío de correo electrónico con la ayuda de un programa de correo también se puede hacer usando el siguiente código, pero el contenido real del correo electrónico no se puede insertar de esta manera.

```
Dim attachs(0)
oMailer =
createUnoService("com.sun.star.system.SimpleSystemMail")
oMailProgram = oMailer.querySimpleMailClient()
oNewmessage = oMailProgramm.createSimpleMailMessage()
oNewmessage.setRecipient(stField1)
oNewmessage.setSubject(stField2)
attachs(0) = "file:///..."
oNewmessage.setAttachement(attachs())
oMailProgram.sendSimpleMailMessage(oNewmessage, 0 )
```

Para posibles parámetros, consulte:
http://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1system_1_1XSimpleMailMessage.html

Cambiar el puntero del ratón al situarse sobre un enlace

En el navegador de internet y el otros programas, al situar el puntero del ratón sobre un enlace externo el puntero del ratón se transforma en una mano, además el texto del enlace suele estar resaltado. Base también emplea esta característica, y su semejanza es perfecta. Cualquier usuario entenderá que puede hacer un clic para abrir un programa externo.

Para esta sección, consulte vea el formulario *Mail_Website_activate_directly* en la base de datos de ejemplo `Example_Mail_File_activate.odb`.

Este breve procedimiento debe estar vinculado al suceso `mouse inside` de un control.

```
Sub mouse_pointer(Event As Object)
  REM Consulte también SwitchmousePointer en el módulo
«ModuleControls» de la biblioteca de macros de LibreOffice «Tools»
  Dim oPointer As Object
  oPointer = createUnoService("com.sun.star.awt.Pointer")
  oPointer.setType(27) 'Tipos (consulte
com.sun.star.awt.SystemPointer en la API)
  Event.Source.Peer.SetPointer(oPointer)
End Sub
```

Mostrar formularios sin una barra de herramientas

Los nuevos usuarios de Base a menudo se irritan porque aparecen algunas barras de herramientas cuyo uso no es util dentro de un formulario. Estas barras de herramientas se pueden ocultar de varias maneras. Las dos maneras, más comunes para ocultar una barra de herramientas en todas las versiones de LibreOffice, se describen a continuación.

Los tamaños de las ventanas y las barras de herramientas generalmente están controlados por una macro que se inicia desde un formulario usando **Herramientas > Personalizar > Sucesos > Abrir documento**. Esto se aplica a todo el documento, no a uno principal o subformulario individual.

Formularios sin barra de herramientas en la ventana

El tamaño de una ventana puede variar. Usando el botón apropiado también se puede cerrar. Estas tareas las lleva a cabo el gestor de ventanas o entorno de escritorio de su sistema operativo. La posición y el tamaño de una ventana en la pantalla se pueden establecer con una macro cuando se inicia el programa.

```
Sub Hide_toolbar
```

```
    Dim oFrame As Object
    Dim oWin As Object
    Dim oLayoutMng As Object
    Dim aElements()
    oFrame = StarDesktop.getCurrentFrame()
```

El título del formulario se mostrará en la barra de título de la ventana.

```
    oFrame.setTitle "Mi Formulario"
    oWin = oFrame.getContainerWindow()
```

La ventana está maximizada. Esto no es lo mismo que el modo de pantalla completa, ya que la barra de tareas todavía está visible. La ventana tiene una barra de título, que se puede usar para cambiar su tamaño o cerrarla.

```
    oWin.IsMaximized = true
```

Es posible crear una ventana con un tamaño y posición específicos. Esto se lleva a cabo con `oWin.setPosSize(0,0,600,400,15)`. Aquí la ventana aparece en la esquina superior izquierda de la pantalla con un ancho de 600 píxeles y una altura de 400. El último número indica los valores introducidos que se tendrán en cuenta. Se llama `Flag`. Este valor se calcula a partir de la suma de los siguientes valores (internos de la API): x = 1, y = 2, ancho = 4, alto = 8. Como todos los parámetros >están expresados, `Flag` tiene que tener el valor 15 (1+2+4+8)

```
    oLayoutMng = oFrame.LayoutManager
    aElements = oLayoutMng.getElements()
    For i = LBound(aElements) To UBound(aElements)
      If aElements(i).ResourceURL =
        "private:resource/toolbar/formsnavigationbar" Then
      Else
        oLayoutMng.hideElement(aElements(i).ResourceURL)
      End If
    Next
  End Sub
```

En el caso de una barra de navegación de formulario, no se debe ocultar, el formulario debe poderse usar en el caso en que no se haya incorporado un control de barra de navegación (lo que provocaría que la barra de navegación esté oculta de todos modos). Solo se deben ocultar las barras de herramientas que no sean la barra de navegación.

Si las barras de herramientas se ocultan, después de abandonar el formulario, seguirán ocultas. Por supuesto, pueden visualizarse usando **Ver > Barras de herramientas**. Pero sería bastante molesto si faltara la barra de herramientas estándar (**Ver > Barras de herramientas > Estándar**) o la barra de estado (**Ver > Barra de estado**).

Este procedimiento muestra (`showElement`) las barras de herramientas que estaban ocultas (`hideElement`). Los comentarios aluden a las barras cuya ausencia puede ser más notable.

```
  Sub Show_toolbar
    Dim oFrame As Object
    Dim oLayoutMng As Object
    Dim aElements()
    oFrame = StarDesktop.getCurrentFrame()
    oLayoutMng = oFrame.LayoutManager
    aElements = oLayoutMng.getElements()
    For i = LBound(aElements) To UBound(aElements)
```

```
    oLayoutMng.showElement(aElements(i).ResourceURL)
  Next
  ' Barras importantes que pueden encontrarse ocultas:
  ' "private:resource/toolbar/standardbar"
  ' "private:resource/statusbar/statusbar"
End Sub
```

Las macros están vinculadas a: **Herramientas > Personalizar > Sucesos > Abrir documento** (*Hide _toolbar*) y **Cerrar documento** (*Show_Toolbar*).

A veces las barras de herramientas no vuelven. En algunos casos, puede ser más útil no utilizar las barras de herramientas >que el administrador de diseño ya conoce, sino primero crear barras de herramientas personalizadas y luego mostrarlas:

```
Sub Hide_toolbar
  Dim oFrame As Object
  Dim oLayoutMng As Object
  Dim i As Integer
  Dim aElements(5) As String
  oFrame = StarDesktop.getCurrentFrame()
  oLayoutMng = oFrame.LayoutManager
  aElements(0) = "private:resource/menubar/menubar"
  aElements(1) = "private:resource/statusbar/statusbar"
  aElements(2) = "private:resource/toolbar/formsnavigationbar"
  aElements(3) = "private:resource/toolbar/standardbar"
  aElements(4) = "private:resource/toolbar/formdesign"
  aElements(5) = "private:resource/toolbar/formcontrols"
  For Each i In aElemente()
    IF Not(oLayoutMng.requestElement(i)) Then
      oLayoutMng.createElement(i)
    End If
  oLayoutMng.showElement(i)
  Next i
End Sub
```

Las barras de herramientas que se crearán se nombran explícitamente. Si la barra de herramientas correspondiente no está disponible para el administrador de diseño, se crea usando `createElement` y luego se muestra usando `showElement`.

Formularios en modo de pantalla completa

En el modo de pantalla completa, el formulario cubre toda la pantalla. No hay barra de tareas u otros elementos que distraigan si se están ejecutando otros programas.

```
Function Fullscreen(boSwitch As Boolean)
  Dim oDispatcher As Object
  Dim Props(0) As New com.sun.star.beans.PropertyValue
  oDispatcher =
createUnoService("com.sun.star.frame.DispatchHelper")
  Props(0).Name = "FullScreen"
  Props(0).Value = boSwitch
  oDispatcher.executeDispatch(ThisComponent.CurrentController.Frame,
```

```
        ".uno:FullScreen", "", 0, Props())
    End Function
```

Esta función se inicia utilizando el siguiente procedimiento (`Fullscreen_on`). En este procedimiento, se ejecuta simultáneamente el procedimiento anterior (`hide_toolbar`) para ocultar las barras de herramientas; de lo contrario, aparecerá la barra de herramientas, y al usarla se desactiva el modo de pantalla completa. En el modo de pantalla completa hay una barra de herramientas, aunque solo tiene un símbolo para volver al modo normal:

```
Sub Fullscreen_on
  Fullscreen(true)
  Hide_toolbar
End Sub
```

Se sale del modo de pantalla completa presionando la tecla ***ESC***. También se puede usar un botón específico al que se asigne el siguiente procedimiento:

```
Sub Fullscreen_off
  Fullscreen(false)
  Show_toolbar
End Sub
```

Lanzar formularios directamente desde la apertura de la base de datos

Cuando las barras de herramientas desaparecen o se debe mostrar un formulario en modo de pantalla completa, el archivo de la base de datos debe iniciar el formulario directamente al abrirse. Desafortunadamente, un comando simple para abrir un formulario no funcionará, ya que la conexión de la base de datos aún no existe cuando se abre el archivo.

La siguiente macro se inicia desde **Herramientas > Personalizar > Sucesos > Abrir documento**. Use la opción *Guardar en...* `Databasefile.odb`.

```
Sub Form_Directstart
  Dim oDatasource As Object
  oDatasource = ThisDatabaseDocument.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If

ThisDatabaseDocument.FormDocuments.getByName("Nombre_Formulario").op
en
End Sub
```

Primero se tiene que hacer una conexión a la base de datos. El controlador es parte de `ThisDatabaseDocument`, igual que el formulario. Tras hacer esta conexión, el formulario se puede iniciar y, al estar conectado, se tiene acceso a los datos.

Acceder a una base de datos MySQL con macros

Todas las macros mostradas hasta ahora han sido parte de una base de datos interna HSQLDB. Cuando se trabaja con bases de datos externas, son necesarios algunos cambios y extensiones.

Código MySQL en macros

Cuando se accede a la base de datos interna mediante macros las tablas y los campos tienen que estar entre comillas dobles duplicadas porque en Basic las comillas dobles se utilizan para delimitar un texto. Una instrucción en SQL:

```
SELECT "Field" FROM "Table"
```

Dentro de una macro tendrá el siguiente aspecto.:

```
stSQL = "SELECT ""Field"" FROM ""Table"""
```

Las consultas MySQL usan otra forma de señalar los campos y tablas: una comilla inclinada.

```
SELECT `Field` FROM `Database`.`Table`
```

Dentro del código Basic la comilla inclinada no necesita estar duplicada, pero sí hay que encerrar la instrucción entre comillas dobles, puesto que se trata de una cadena de texto. La instrucción MySQL anterior, en Basic debe escribirse así:

```
stMySql = "SELECT `Field` FROM `Database`.`Table`"
```

Tablas temporales como almacenamiento intermedio individual

En el capítulo anterior, una tabla de un único registro se usaba con frecuencia para buscar o filtrar tablas. Esto no funcionará en un sistema multiusuario, ya que otros usuarios dependerían del valor del filtro de otra persona. Las tablas temporales en MySQL solo son accesibles para el usuario de la conexión activa, por lo que se pueden crear este tipo de tablas para buscar y filtrar.

Naturalmente, estas tablas no se pueden crear de antemano. Tienen que crearse cuando se abre el archivo Base. Por lo tanto, la siguiente macro debe estar vinculada a la apertura del archivo `*.odb`.

```
Sub CreateTempTable
  oDatasource = thisDatabaseDocument.CurrentController
  If Not (oDatasource.isConnected()) Then oDatasource.connect()
  oConnection = oDatasource.ActiveConnection()
  oSQL_Statement = oConnection.createStatement()
  stSql = "CREATE TEMPORARY TABLE IF NOT EXISTS `Searchtmp` (`ID`
INT PRIMARY KEY,
     `Name` VARCHAR(50))"
  oSQL_Statement.executeUpdate(stSql)
End Sub
```

Cuando se abre el archivo `*.odb` >no hay conexión a una base de datos MySQL externa. La conexión tiene que ser creada y posteriormente configurar una tabla temporal con los campos necesarios.

Diálogos

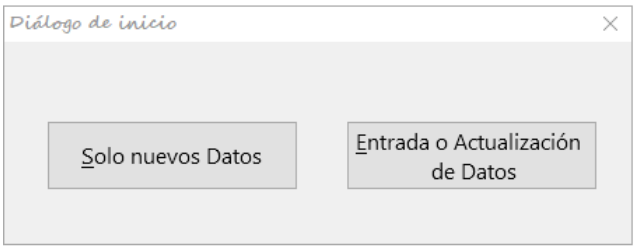
En Base, puede usar diálogos Basic en lugar de formularios para la entrada y modificación de datos o el mantenimiento de la base de datos. Los diálogos se pueden personalizar directamente para el entorno de aplicación en uso, pero, naturalmente, no se definen de manera tan cómoda como los formularios. Aquí se muestra una breve introducción que termina en un ejemplo bastante complicado para usar en el mantenimiento de la base de datos.

El ejemplo que utilizaremos se basa en la base de datos proporcionada `Example_Dialogs.odb` que contiene una tabla *name* con tres campos: *ID*, (clave primaria) *forename* (nombre) y *surname* (apellido)

Iniciar y finalizar diálogos

Primero, se debe crear el diálogo en la computadora en que está el archivo *,odb. Esto se hace usando **Herramientas > Macros > Organizar diálogos > Nombre_de_la_base_de_datos > Estándar > Nuevo**. El diálogo aparece con una superficie gris continua y una barra de título con un icono de cierre. El diálogo creado puede abrirse y luego cerrarse haciendo clic en el icono de cierre.

Cuando se selecciona el diálogo, existe la posibilidad de establecer su tamaño y posición en las propiedades generales. También se puede ingresar el contenido del título, *Diálogo de inicio* en este caso).



La barra de herramientas en el borde inferior de la ventana contiene varios controles para los diálogos. Desde ella se han seleccionado dos botones lo que permite iniciar distintos procedimientos u otros diálogos. La edición de contenido y la vinculación de macros a sucesos se hace igual que para los botones en los formularios. Las declaraciones de variables para los diálogos requiere un cuidado especial. El diálogo se declara como una variable global para que se pueda acceder a él mediante diferentes procedimientos. En este caso, el diálogo se llama oDialog0, porque en el ejemplo habrá más diálogos, aunque puede tener otro nombre.

```
Dim oDialog0 As Object
```

Primero se carga la biblioteca para el diálogo. Está en el directorio Estándar, si no se eligió otro nombre cuando se creó el diálogo. El diálogo en sí se puede alcanzar en esta biblioteca utilizando el nombre Dialog0. `Execute()` inicia el diálogo.

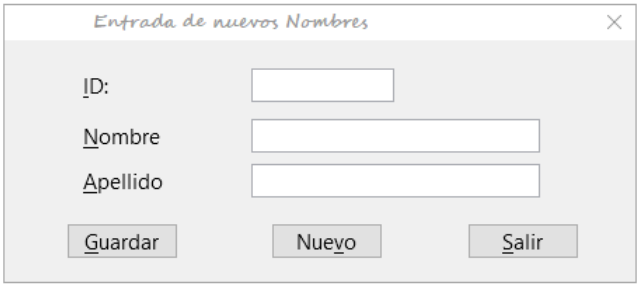
```
Sub Dialog0Start
  DialogLibraries.LoadLibrary("Standard")
  oDialog0 = createUnoDialog(DialogLibraries.Standard.Dialog0)
  oDialog0.Execute()
End Sub
```

En principio, un diálogo puede cerrarse usando el botón Cerrar en el marco. Sin embargo, si desea otro botón específico para esto, el comando `EndExecute()` debe usarse dentro de un procedimiento.

```
Sub Dialog0End
  oDialog0.EndExecute()
End Sub
```

Dentro de este marco, se puede iniciar y cerrar cualquier número de diálogos nuevamente.

Diálogo sencillo para ingresar nuevos registros



Este diálogo es un primer paso para el siguiente diálogo para editar registros. Primero se define el enfoque básico para administrar tablas. Aquí nos ocupamos del almacenamiento de registros con nuevas claves primarias o la nueva entrada completa de registros. ¿Hasta qué punto un pequeño diálogo como este puede ser suficiente para la entrada de dato en una base de datos? – depende de las necesidades del usuario –.

```
Dim oDialog1 As Object
```

Se debe crear una variable global para el diálogo en el nivel superior del módulo antes de cualquier procedimiento para que pueda ser accesible en cualquier momento.

El diálogo se abre y cierra de la misma manera que el diálogo anterior. Solo se cambia el nombre de Dialog0 a Dialog1. El botón *Salir* está vinculado al procedimiento para cerrar el diálogo.

El botón Nuevo se usa para borrar todos los controles del diálogo de entradas anteriores, utilizando el procedimiento `DatafieldsClear`.

```
Sub DatafieldsClear
  oDialog1.getControl("NumericField1").Text = ""
  oDialog1.getControl("TextField1").Text = ""
  oDialog1.getControl("TextField2").Text = ""
End Sub
```

Se puede acceder a cada control que se ha insertado en un diálogo por su nombre. La interfaz de Basic evita que los nombres estén duplicados, cosa que no ocurre con los controles en un formulario.

El método `getControl` se usa con el nombre del control. Los campos numéricos también tienen una propiedad `Text` , usando esta propiedad es la única forma en que se puede vaciar un campo numérico (existe texto vacío, pero no un número vacío). Se debe escribir un 0 en el campo de la clave primaria.

El botón **Guardar** inicia el procedimiento `Data1Save`:

```
Sub Data1Save
  Dim oDatasource As Object
  Dim oConnection As Object
  Dim oSQL_Command As Object
  Dim loID As Long
  Dim stForename As String
  Dim stSurname As String
  loID = oDialog1.getControl("NumericField1").Value
  stForename = oDialog1.getControl("TextField1").Text
  stSurname = oDialog1.getControl("TextField2").Text
  If loID > 0 And stSurname <> "" Then
    oDatasource = thisDatabaseDocument.CurrentController
```

```
      If Not (oDatasource.isConnected()) Then
        oDatasource.connect()
      End If
      oConnection = oDatasource.ActiveConnection()
      oSQL_Command = oConnection.createStatement()
      stSql = "SELECT ""ID"" FROM ""name"" WHERE ""ID"" = '"+loID+"'"
      oResult = oSQL_Command.executeQuery(stSql)
      While oResult.next
        MsgBox ("Ya existe el valor para el campo 'ID'",16,"Valor
Duplicado")
        Exit Sub
      Wend
      stSql = "INSERT INTO ""name"" (""ID"", ""forename"",
""surname"")
        VALUES ('"+loID+"','"+stForename+"','"+stSurname+"')"
      oSQL_Command.executeUpdate(stSql)
      DatafieldsClear
    End If
End Sub
```

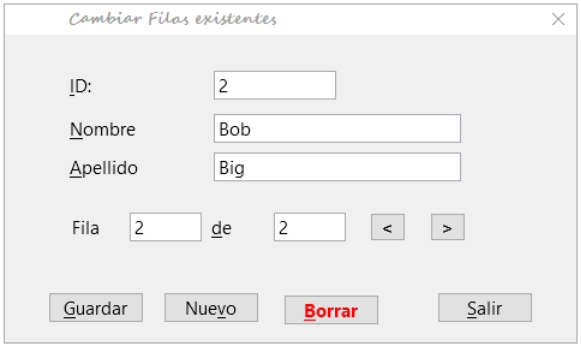
Como en el procedimiento `DatafieldsClear`, se accede a los campos de entrada. Esta vez el acceso es para lectura. Solo si el campo *ID* tiene una entrada mayor que 0 y el campo *surname* también contiene texto, se pasará el registro. Se puede excluir un valor nulo para el *ID* porque una variable numérica para números enteros siempre se inicializa a 0. Por lo tanto, un campo vacío se almacena con un valor cero.

Si ambos campos han recibido contenido, se establece una conexión con la base de datos. Como los controles no están en un formulario, la conexión de la base de datos debe realizarse utilizando `thisDatabaseDocument.CurrentController`.

Primero se consulta la base de datos para ver si ya existe un registro con la misma clave primaria que la ingresada. Si esto ocurre, se muestra un diálogo de mensaje que contiene un símbolo de *Stop* (código: 16) el mensaje *Ya existe valor para el campo ' ID'* >y con título *Valor Duplicado*. Al pulsar *Aceptar* el procedimiento se detiene con la instrucción `Exit SUB`.

Si la consulta no encuentra ningún registro con la misma clave primaria, el nuevo registro se inserta en la base de datos utilizando la orden SQL de inserción. Luego se llama a la rutina `DatafieldsClear` para limpiar el contenido de los controles y poder continuar ingresando datos.

Diálogo para editar registros en una tabla



Este diálogo ofrece más posibilidades que el anterior. Aquí se pueden mostrar todos los registros y puede navegar por ellos, crear o eliminar registros. Naturalmente, el código es más complicado.

El botón Salir está vinculado al procedimiento que se describió en el diálogo anterior, modificado para *Dialog2*. Aquí se describen los botones restantes y sus funciones. La entrada de datos en el diálogo está restringida porque el campo *ID* tiene que tener un valor mínimo de 1. Esta limitación está producida por el manejo de variables en Basic:

Las variables numéricas se inicializan de manera predeterminada en 0. Por lo tanto, si los valores numéricos de los campos vacíos y los de los campos que contienen un 0 se leen, Basic no puede detectar diferencias entre ellos. Esto significa que si se usara un 0 en el campo *ID*, tendría que leerse primero como texto y quizás convertirse más tarde a un número.

El diálogo se crea en las mismas condiciones que antes. Sin embargo, el procedimiento de carga depende de un valor cero para la variable pasada al procedimiento `DataLoad`.

```
Sub Dialog2Start
  DialogLibraries.LoadLibrary("Standard")
  oDialog2 = createUnoDialog(DialogLibraries.Standard.Dialog2)
  DataLoad(0)
End Sub
```

Con este procedimiento se crea el diálogo, y se hace la llamada al procedimiento Dataload() para conectar con la base de datos y cargar los valores en los controles. Una vez cargados los valores se presentará el diálogo al usuario.

```
Sub DataLoad(loID As Long)
  Dim oDatasource As Object
  Dim oConnection As Object
  Dim oSQL_Command As Object
  Dim stForename As String
  Dim stSurname As String
  Dim loRow As Long
  Dim loRowMax As Long
  Dim inStart As Integer
  oDatasource = thisDatabaseDocument.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If
  oConnection = oDatasource.ActiveConnection()
  oSQL_Command = oConnection.createStatement()
  If loID < 1 Then
    stSql = "SELECT MIN(""ID"") FROM ""name"""
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
      loID = oResult.getInt(1)
    Wend
    inStart = 1
  End If
```

Las variables se declaran. La conexión de la base de datos para el diálogo se establece como se describe anteriormente. Al principio, `loID` es 0. Este caso proporciona el valor de clave primaria más bajo permitido por SQL. El registro correspondiente se mostrará más tarde en el diálogo. Al mismo tiempo, la variable `inStart` se establece en 1, de modo que el diálogo se pueda iniciar más tarde. Si la tabla no contiene ningún registro, `loID` seguirá siendo 0. En ese caso, no será necesario buscar el número y el contenido de los registros correspondientes.

Solo si `loID` es mayor que 0, se hará una consulta para ver qué registros están disponibles en la base de datos. Luego, una segunda consulta contará todos los registros que se mostrarán. La tercera consulta proporciona la posición del registro en uso contando todos los registros que tengan un número en la clave primaria menor o igual al número introducido.

```
    If loID > 0 Then
      stSql = "SELECT * FROM ""name"" WHERE ""ID"" = '"+loID+"'"
      oResult = oSQL_Command.executeQuery(stSql)
      While oResult.next
        loID = oResult.getInt(1)
        stForename = oResult.getString(2)
        stSurname = oResult.getString(3)
      Wend
      stSql = "SELECT COUNT(""ID"") FROM ""name"""
      oResult = oSQL_Command.executeQuery(stSql)
      While oResult.next
        loRowMax = oResult.getInt(1)
      Wend
      stSql = "SELECT COUNT(""ID"") FROM ""name"" WHERE ""ID"" <=
'"+loID+"'"
      oResult = oSQL_Command.executeQuery(stSql)
      While oResult.next
        loRow = oResult.getInt(1)
      Wend
      oDialog2.getControl("NumericField1").Value = loID
      oDialog2.getControl("TextField1").Text = stForename
      oDialog2.getControl("TextField2").Text = stSurname
    End If
    oDialog2.getControl("NumericField2").Value = loRow
    oDialog2.getControl("NumericField3").Value = loRowMax
    If loRow = 1 Then
      ' Fila (registro) anterior
      oDialog2.getControl("CommandButton4").Model.enabled = False
    Else
      oDialog2.getControl("CommandButton4").Model.enabled = True
    End If
    If loRow <= loRowMax Then
      ' Fila (registro) siguiente| nueva fila | borrar
      oDialog2.getControl("CommandButton5").Model.enabled = True
      oDialog2.getControl("CommandButton2").Model.enabled = True
      oDialog2.getControl("CommandButton6").Model.enabled = True
    Else
      oDialog2.getControl("CommandButton5").Model.enabled = False
      oDialog2.getControl("CommandButton2").Model.enabled = False
      oDialog2.getControl("CommandButton6").Model.enabled = False
    End If
    IF inStart = 1 Then
      oDialog2.Execute()
```

```
    End If
  End Sub
```

Los valores obtenidos se transfieren a los controles del diálogo. Las entradas para el número de registro actual y el número total de registros recuperados siempre se escriben, reemplazando el valor numérico predeterminado de 0.

Los botones de navegación (*CommandButton4* y *CommandButton5*) solo se pueden usar cuando se puede acceder al registro correspondiente. En caso contrario, se desactivan temporalmente con `enabled = False`. Lo mismo ocurre para los botones *Nuevo* y *Borrar*. No deberían estar disponibles cuando el número de registro mostrado sea mayor que el número máximo que se determinó. Esta es la configuración predeterminada de este diálogo al ingresar registros.

Una vez cargados los valores necesarios para los controles se presenta el diálogo al usuario con la instrucción `oDialog2.Execute`.

El botón `<` (`CommandButton4`) se usa para retroceder al registro anterior. Por lo tanto, este botón debe estar desactivado cuando el registro que se muestre sea el primero.

La navegación mediante este botón requiere que la clave primaria para el registro en curso se lea desde el control de cuadro de texto con etiqueta *ID* (*NumericField1*) (donde se refleja el registro en curso). Aquí hay dos casos posibles:

1) El control no tiene ningún valor, En este caso `loId` tiene el valor predeterminado suministrado por Basic a la variable de tipo integer (0)
2) O por el contrario, `loID` contiene un valor mayor que 0. Luego, una consulta puede determinar el valor de *ID* directamente inferior al >actual.

```
Sub PreviousRow
  Dim loID As Long
  Dim loIDnew As Long
  loID = oDialog2.getControl("NumericField1").Value
  oDatasource = thisDatabaseDocument.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If
  oConnection = oDatasource.ActiveConnection()
  oSQL_Command = oConnection.createStatement()
  If loID < 1 Then
    stSql = "SELECT MAX(""ID"") FROM ""name"""
  Else
    stSql = "SELECT MAX(""ID"") FROM ""name"" WHERE ""ID"" <
'"+loID+"'"
  End If
  oResult = oSQL_Command.executeQuery(stSql)
  While oResult.next
    loIDnew = oResult.getInt(1)
  Wend
  If loIDnew > 0 Then
    DataLoad(loIDnew)
  End If
End Sub
```

Si el control *ID* está vacío, la navegación por registros debería cambiar al valor más alto del número de clave primaria. Si, por otro lado, el control *>ID* se refiere a un registro, debe mostrar el valor anterior de ID.

El resultado de esta consulta se utiliza para ejecutar el procedimiento `DataLoad` nuevamente con el valor de la clave primaria correspondiente.

El botón `>` (`CommandButton5`) se usa para navegar al siguiente registro. Esta posibilidad debería existir solo cuando los datos del diálogo no se hayan vaciado para la entrada de un nuevo registro. Naturalmente, este será el caso cuando se inicie el diálogo y también con una tabla vacía.

Un valor en *NumericField1* es obligatorio. A partir de este valor, SQL puede determinar qué clave primaria es la siguiente más alta en la tabla. Si el conjunto de resultados de la consulta está vacío porque no hay un registro correspondiente, el valor de `loIDnew = 0`. De lo contrario, el contenido del siguiente registro se lee usando `DataLoad`.

```
Sub NextRow
  Dim loID As Long
  Dim loIDnew As Long
  loID = oDialog2.getControl("NumericField1").Value
  oDatasource = thisDatabaseDocument.CurrentController
  If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
  End If
  oConnection = oDatasource.ActiveConnection()
  oSQL_Command = oConnection.createStatement()
  stSql = "SELECT MIN(""ID"") FROM ""name"" WHERE ""ID"" >
'"+loID+"'"
  oResult = oSQL_Command.executeQuery(stSql)
  While oResult.next
    loIDnew = oResult.getInt(1)
  Wend
  If loIDnew > 0 Then
    DataLoad(loIDnew)
  Else
    Datafields2Clear
  End If
End Sub
```

Si se ha alcanzado el registro más alto, al navegar al siguiente registro, la tecla de navegación inicia el procedimiento `Datafields2Clear`, que sirve para preparar la entrada de un nuevo registro.

Este procedimiento no solo vacía los datos de los controles. El número del nuevo registro presentado en la zona de navegación se obtiene sumando un uno al número total de registros, dejando en claro que el registro en el que se está trabajando actualmente aún no está incluido en la base de datos.

Tan pronto como se haya lanzado `Datafields2Clear`, se activa la posibilidad de saltar al registro anterior, se salta al siguiente registro y se desactivan los botones *Nuevo* y *Borrar*, puesto que el diálogo ya está preparado para crear un nuevo registro y no se puede borrar un registro que todavía no existe.

```
Sub Datafields2Clear
  loRowMax = oDialog2.getControl("NumericField3").Value
```

```
  oDialog2.getControl("NumericField1").Text = ""
  oDialog2.getControl("TextField1").Text = ""
  oDialog2.getControl("TextField2").Text = ""
  oDialog2.getControl("NumericField2").Value = loRowMax + 1
  oDialog2.getControl("CommandButton4").Model.enabled = True '
Registro anterior
  oDialog2.getControl("CommandButton5").Model.enabled = False   '
Registro siguiente
  oDialog2.getControl("CommandButton2").Model.enabled = False   '
Registro nuevo
  oDialog2.getControl("CommandButton6").Model.enabled = False   '
Borrar
End Sub
```

Guardar registros solo debería ser posible cuando los campos *ID* y *surname* contengan entradas. Si se cumple esta condición, el procedimiento comprueba si se trata de un nuevo registro haciendo uso del puntero de registro que está configurado para que el nuevo registro sean uno más alto que el número total de registros.

Para nuevos registros, se realizan comprobaciones para garantizar que se realice correctamente la operación de guardado. Si el número utilizado para la clave primaria se ha utilizado antes, se muestra una advertencia. Si la pregunta asociada se responde con el botón *Aceptar*, se sobrescribe el registro existente con este número. De lo contrario, se utilizará el número existente. Si no hay entradas existentes en la base de datos (`loRowMax = 0`), esta prueba es innecesaria y el nuevo registro se puede guardar directamente. Para un nuevo registro, el número de registros se incrementa en 1 y los controles se limpian para el siguiente registro.

Los registros existentes simplemente se sobrescriben con un comando de actualización.

```
Sub Data2Save(oEvent As Object)
  Dim oDatasource As Object
  Dim oConnection As Object
  Dim oSQL_Command As Object
  Dim oDlg As Object
  Dim loID As Long
  Dim stForename As String
  Dim stSurname As String
  Dim inMsg As Integer
  Dim loRow As Long
  Dim loRowMax As Long
  Dim stSql As String
  oDlg = oEvent.Source.getContext()
  loID = oDlg.getControl("NumericField1").Value
  stForename = oDlg.getControl("TextField1").Text
  stSurname = oDlg.getControl("TextField2").Text
  If loID > 0 And stSurname <> "" Then
    oDatasource = thisDatabaseDocument.CurrentController
    If Not (oDatasource.isConnected()) Then
      oDatasource.connect()
    End If
    oConnection = oDatasource.ActiveConnection()
```

```
    oSQL_Command = oConnection.createStatement()
    loRow = oDlg.getControl("NumericField2").Value
    loRowMax = oDlg.getControl("NumericField3").Value
    If loRowMax < loRow Then
      If loRowMax > 0 Then
        stSql = "SELECT ""ID"" FROM ""name"" WHERE ""ID"" =
'"+loID+"'"
        oResult = oSQL_Command.executeQuery(stSql)
        While oResult.next
         inMsg = MsgBox ("Ya existe el valor para el campo 'ID'." &
CHR(13) & "¿Desea sobreescribir el registro?",20,"Valor Duplicado")
         If inMsg = 6 Then
            stSql = "UPDATE ""name"" SET
""forename""='"+stForename+"',
               ""surname""='"+stSurname+"' WHERE ""ID"" =
'"+loID+"'"
            oSQL_Command.executeUpdate(stSql)
            DataLoad(loID)  'el contador de registros (Rowcount) se
debe actualizar.
         End If
         Exit Sub
        Wend
      End If
      stSql = "INSERT INTO ""name"" (""ID"", ""forename"",
""surname"") VALUES
        ('"+loID+"','"+stForename+"','"+stSurname+"')"
      oSQL_Command.executeUpdate(stSql)
      oDlg.getControl("NumericField3").Value = loRowMax + 1
        ' Después de la inserción se suma un 1 al número de
registros.
      Datafields2Clear
        ' Se limpian los datos de los controles para la siguiente
entrada.
    Else
      stSql = "UPDATE ""name"" SET ""forename""='"+stForename+"',
        ""surname""='"+stSurname+"' WHERE ""ID"" = '"+loID+"'"
      oSQL_Command.executeUpdate(stSql)
    End If
  End If
End Sub
```

El procedimiento de borrado utiliza una pregunta adicional para evitar eliminaciones accidentales. Dado que el botón está desactivado cuando los campos de entrada están vacíos, nunca debe aparecer un *NumericField1* vacío. Por lo tanto, se puede omitir la verificación `IF loID > 0`.

La eliminación hace que el número de registros disminuya en 1. Esto debe corregirse usando `loRowMax - 1`. Luego se muestra el registro siguiente al que estaba en uso.

```
Sub DataDelete(oEvent As Object)
  Dim oDatasource As Object
```

```
  Dim oConnection As Object
  Dim oSQL_Command As Object
  Dim oDlg As Object
  Dim loID As Long
  oDlg = oEvent.Source.getContext()
  loID = oDlg.getControl("NumericField1").Value
  If loID > 0 Then
    inMsg = MsgBox ("¿Desea borrar el registro actual?",20,"Borrar
registro")
    If inMsg = 6 Then
      oDatasource = thisDatabaseDocument.CurrentController
      If Not (oDatasource.isConnected()) Then
        oDatasource.connect()
      End If
      oConnection = oDatasource.ActiveConnection()
      oSQL_Command = oConnection.createStatement()
      stSql = "DELETE FROM ""name"" WHERE ""ID"" = '"+loID+"'"
      oSQL_Command.executeUpdate(stSql)
      loRowMax = oDlg.getControl("NumericField3").Value
      oDlg.getControl("NumericField3").Value = loRowMax - 1
      NextRow
    End If
  ELSE
    MsgBox ("No se ha borrado ningún registro." & CHR(13) &
      "No se ha seleccionado ningún registro.",64,"No se puede
borrar")
  End If
End Sub
```

Este pequeño diálogo ha demostrado que el uso de código macro puede proporcionar una base para procesar registros. El acceso a través de formularios es mucho más fácil, pero un diálogo puede ser muy flexible para adaptarse a los requisitos del programa. Sin embargo, no es adecuado para la creación rápida de una interfaz de base de datos.

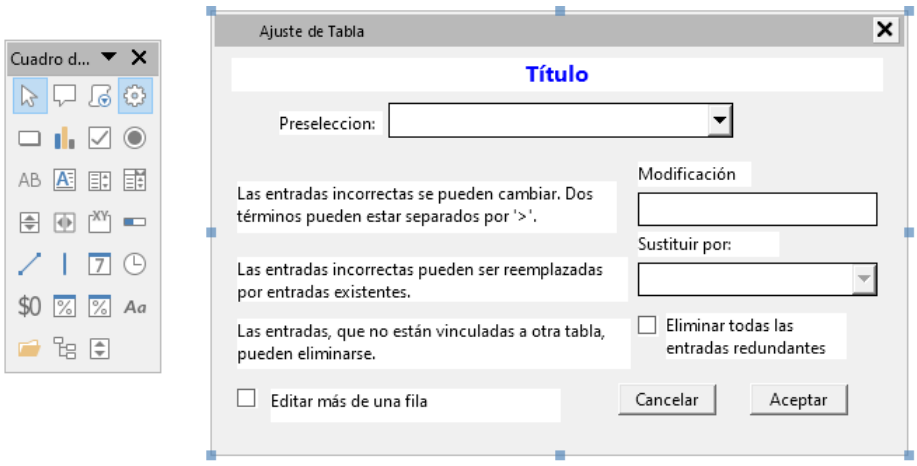
Usar un diálogo para limpiar entradas incorrectas en tablas

Este diálogo más complejo *Dialog_TableAdjustment* se encuentra en la base de datos de ejemplo `Media_with_macros.odb`

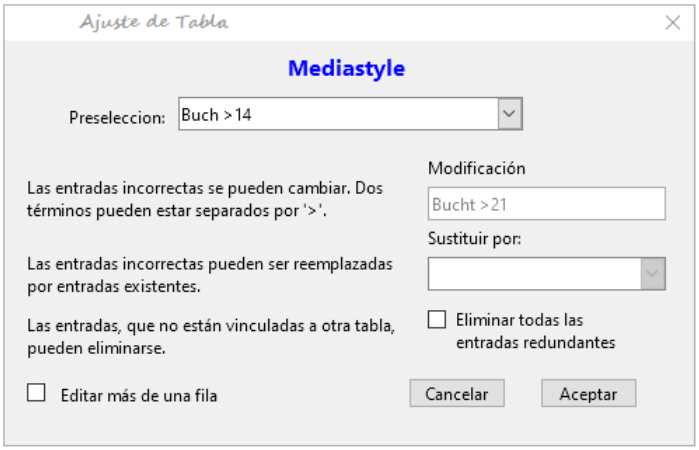
Los errores de entrada en los campos a menudo solo se notan más tarde. A menudo es necesario modificar entradas duplicada en varios registros al mismo tiempo. Es incómodo tener que hacer esto en la vista de tabla normal, especialmente cuando se tienen que editar varios registros, ya que cada registro requiere que se realice una entrada individual.

Los formularios pueden usar macros para hacer este tipo de cosas, pero para hacerlo en varias tablas, necesitaría formularios construidos de manera idéntica.

Los diálogos pueden hacer el trabajo de manera más eficaz. Al principio se puede proporcionar un diálogo con los datos necesarios para las tablas apropiadas y se puede invocar desde distintos formularios.



Los diálogos se guardan junto con los módulos para macros. Su creación es básicamente igual a la de un formulario. Los controles son muy similares, excepto el control de tabla que está presente en formularios pero no existe en los diálogos de Basic.



La apariencia de los diálogos está determinada por la configuración de la interfaz gráfica de usuario.

El diálogo que se muestra arriba sirve en la base de datos de ejemplo para editar tablas que no se utilizan directamente como base de un formulario.

Por ejemplo, solo se puede acceder al tipo de artículo a través de un listado (en la versión macro se utiliza un cuadro combinado). En la versión macro, el contenido del campo puede expandirse con nuevo contenido, pero no es posible alterar el contenido existente. En la versión sin macros, las modificaciones se llevan a cabo utilizando un control de tabla separado.

Si bien las alteraciones en este caso son fáciles de realizar sin macros, es bastante difícil cambiar el tipo de artículo de muchos artículos a la vez. Supongamos que están disponibles los siguientes tipos:

"Libro, tapa blanda", "Libro, tapa dura", "Libro en rústica" y "Libro espiral doble".

Ahora resulta que, después de que la base de datos ha estado en uso durante mucho tiempo, nos damos cuenta que tantos tipos de artículos han vuelto excesiva la tarea de ordenación. Por lo tanto, deseamos reducirlos, preferiblemente a un solo tipo. Sin macros, los registros en la tabla de artículos tendrían que ser encontrados (usando un filtro) y alterados individualmente. Si conoce SQL, puede hacerlo mucho mejor usando una orden SQL. Puede cambiar todos los registros en la tabla *Media* con una sola entrada. Una segunda orden SQL elimina los tipos de medios ya sobrantes que no tienen ningún enlace a la tabla de Medios.

Precisamente, este método se aplica utilizando el diálogo anterior con el control >*Sustituir por :* solo hay que adaptar una orden SQL a la tabla *Mediastyle* utilizando un procedimiento que también puede usarse en otras tablas.

A menudo, las entradas se trasladan a una tabla que, posteriormente, se puede cambiar en el formulario, por lo que ya no son necesarias. Eliminar estas entradas huérfanas puede ser beneficioso, pero son bastante difíciles de encontrar utilizando la interfaz gráfica de usuario. Aquí, de nuevo, es útil una orden SQL adecuada, junto con una instrucción de eliminación. Esta orden para las tablas afectadas se incluye en el diálogo con *Eliminar todas las entradas redundanes*.

Si el diálogo se va a utilizar para llevar a cabo varios cambios, se indica mediante la casilla de verificación *Editar más de una fila*. Entonces el diálogo no terminará simplemente cuando se haga clic en el botón *Aceptar*.

El código de macro para este diálogo se puede ver en su totalidad en la base de datos de ejemplo. Solo algunos extractos se explican a continuación.

```
Sub Table_purge(oEvent As Object)
```

La macro debe iniciarse ingresando en la sección Información adicional para los botones relevantes:

```
0: Form, 1: Subform, 2: SubSubform, 3: Combobox or table control, 4:
Foreign key field in a form, empty for a table control, 5: Table name of
auxiliary table, 6: Table field1 of auxiliary table, 7: Table field2 of
auxiliary table, or 8: Table name of auxiliary table for table field2
```

Las entradas en esta área se enumeran al comienzo de la macro como comentarios. Los números vinculados a ellos se transfieren y la entrada relevante se lee desde una matriz. La macro puede editar listados, que tienen dos entradas, separadas por el signo mayor que (`>`). Estas dos entradas también pueden provenir de tablas diferentes y reunirse mediante una consulta, como por ejemplo en la tabla de Código postal, que solo tiene el campo de clave foránea *Town_ID* para la ciudad, que requiere que la tabla *Town* muestre los nombres de las ciudades.

```
Dim aForeignTable(0, 0 to 1)
Dim aForeignTable2(0, 0 to 1)
```

Entre las variables definidas al principio hay dos matrices. Si bien el comando `Split()`puede crear matrices normales durante la ejecución del procedimiento, las matrices bidimensionales tienen que definirse de antemano. Las matrices bidimensionales son necesarias para almacenar varios registros de una consulta cuando la consulta se refiere a más de un campo. Las dos matrices declaradas anteriormente tienen que poder interpretar las consultas que hacen referencia a dos campos de tabla. Por lo tanto, se definen para dos contenidos diferentes utilizando 0 a 1 para la segunda dimensión.

```
stTag = oEvent.Source.Model.Tag
aTable() = Split(stTag, ", ")
For i = LBound(aTable()) To UBound(aTable())
  aTable(i) = trim(aTable(i))
Next
```

Se leen las variables proporcionadas. La secuencia es la establecida en el comentario anterior. Hay un máximo de nueve entradas y debe declarar si existe una octava entrada para el campo de table2 y una novena entrada para una segunda tabla.

Si los valores se van a eliminar de una tabla, primero es necesario verificar que no existan como claves foráneas en alguna otra tabla. En estructuras de tabla simples, una tabla dada tendrá solo una conexión de clave foránea a otra tabla. Sin embargo, en la base de datos de ejemplo >hay una tabla *Town* que se usa tanto para el lugar de publicación de los artículos como para las

direcciones de la ciudad. Por lo tanto, la clave primaria de la tabla *Town* se ingresa dos veces en diferentes tablas.

Estas tablas y nombres de claves foráneas también se pueden ingresar naturalmente usando el campo *Información adicional*. Sin embargo, sería mejor si se pudieran proporcionar universalmente para todos los casos. Esto se puede hacer usando la siguiente consulta.

```
  stSql = "SELECT ""FKTABLE_NAME"", ""FKCOLUMN_NAME"" FROM
""INFORMATION_SCHEMA""."SYSTEM_CROSSREFERENCE"" WHERE
""PKTABLE_NAME"" = '" + aTable(5) + "'"
```

En la base de datos, el área `INFORMATION_SCHEMA` contiene toda la información sobre las tablas de la base de datos, incluida información sobre claves foráneas. Se puede acceder a las tablas que contienen esta información utilizando `"INFORMATION_SCHEMA"`. `"SYSTEM_CROSSREFERENCE"."PKTABLE_NAME"` proporciona la tabla que proporciona su clave primaria para la conexión. `FKTABLE_NAME` proporciona la tabla que utiliza esta clave primaria como clave foránea. Finalmente, `FKCOLUMN_NAME` proporciona el nombre del campo de clave foránea.

La tabla que proporciona su clave primaria para usar como clave foránea se encuentra en la matriz creada previamente en la posición 6. Al recuento que comienza con 0, el valor se lee de la matriz usando `aTable(5)`.

```
inCount = 0
stForeignIDTab1Tab2 = "ID"
stForeignIDTab2Tab1 = "ID"
stAuxiltable = aTable(5)
```

Antes de que comience la lectura de las matrices, se tienen que establecer algunos valores predeterminados. Estos son: el índice de la matriz en la que se escribirán los valores de la tabla auxiliar, la clave primaria predeterminada si no necesitamos la clave foránea para una segunda tabla y la tabla auxiliar predeterminada, vinculada a la tabla principal, para el código postal y ciudad, la tabla de códigos postales.

Cuando dos campos están vinculados para su visualización en un listado, pueden proceder de dos tablas diferentes. Para la visualización del código postal y la ciudad, la consulta es:

```
SELECT "Postcode"."Postcode" || ' > ' || "Town"."Town" FROM "Postcode", "Town" WHERE
"Postcode"."Town_ID" = "Town"."ID"
```

La tabla para el primer campo (*Postcode*) está vinculada a la segunda tabla mediante una clave foránea. Solo la información de estas dos tablas y los campos *Postcode* y *Town* se pasa a la macro. Todas las claves primarias se llaman por defecto *ID* en la base de datos de ejemplo. Por lo tanto, la clave foránea de *Town* en el código postal tiene que determinarse utilizando el procedimiento.

Del mismo modo, el procedimiento tiene que acceder a cada tabla con la que el contenido del listado está conectado por una clave foránea.

```
oQuery_result = oSQL_Statement.executeQuery(stSql)
If Not IsNull(oQuery_result) Then
  While oQuery_result.next
    ReDim Preserve aForeignTable(inCount,0 to 1)
```

La matriz tiene que estar recién dimensionada cada vez. Para mantener los contenidos existentes, se utiliza la instrucción `Preserve`.

```
    aForeignTables(inCount,0) = oQuery_result.getString(1)
```

Lectura del primer campo con el nombre de la tabla que contiene la clave foránea. El resultado para la tabla de códigos postales es la tabla *Address*.

```
aForeignTables(inCount,1) = oQuery_result.getString(2)
```

Lectura del segundo campo con el nombre del campo de clave foránea. El resultado para la tabla de código postal es el campo `Postcode_ID` en la tabla *Adress*.

En los casos en que una llamada al procedimiento incluye el nombre de una segunda tabla, se ejecuta el siguiente bucle. Solo cuando el nombre de la segunda tabla aparece como la tabla de clave foránea para la primera tabla, se cambia la entrada predeterminada. En nuestro caso, esto no ocurre, ya que la tabla *Town* no tiene clave foránea de la tabla de *Postcode*. Por lo tanto, la entrada predeterminada para la tabla auxiliar sigue siendo *Postcode*; finalmente, la combinación de código postal y ciudad es la base de la tabla de direcciones, que contiene una clave foránea de la tabla de códigos postales.

```
If UBound(aTable()) = 8 Then
  If aTable(8) = aForeignTable(inCount,0) Then
   stForeignIDTab2Tab1 = aForeignTable(inCount,1)
   stAuxiltable = aTable(8)
  End If
End If
inCount = inCount + 1
```

Como es posible que sea necesario leer más valores, el índice se incrementa para redimensionar las matrices y el ciclo termina.

```
  Wend
End If
```

Si, cuando se llama al procedimiento, existe un segundo nombre de tabla, se inicia la misma consulta para esta tabla:

```
If UBound(aTable()) = 8 Then
```

Se ejecuta de manera idéntica, excepto que el bucle prueba si quizás el primer nombre de la tabla aparece como un nombre de tabla de clave foránea. Ese es el caso aquí: la tabla *Postcode*l contiene la clave foránea *Town_ID* de la tabla *Town*. Esta clave foránea ahora se asigna a la variable `stForeignIDTab1Tab2`, de modo que se puede definir la relación entre las tablas.

```
  If aTable(5) = aForeignTable2(inCount,0) Then
   stForeignIDTab1Tab2 = aForeignTable2(inCount,1)
  End If
```

Después de algunas configuraciones adicionales para garantizar un retorno al formulario correcto después de ejecutar el diálogo (determinando el número de línea del formulario, para que podamos volver a ese número de línea después de una nueva lectura), comienza el bucle, que recrea el diálogo cuando se completa la primera acción pero se requiere que el diálogo se mantenga abierto para acciones adicionales. La configuración para la repetición se lleva a cabo utilizando la casilla de verificación correspondiente.

```
Do
```

Antes de que se inicie el diálogo, primero se determina el contenido de los listados. Se debe tener cuidado si los listados representan dos campos de tabla o incluso están relacionados con dos tablas diferentes.

```
If UBound(aTable()) = 6 Then
```

El listado se relaciona solo con una tabla y un campo, ya que la matriz de argumentos termina en `Tablefield1` de la tabla auxiliar.

```
        stSql = "SELECT """ + aTable(6) + """ FROM """ + aTable(5) +
"""" ORDER BY """ + aTable(6) + """"
    ElseIf UBound(aTable()) = 7 Then
```

El listado se relaciona con dos campos de tabla pero solo con una tabla, ya que la matriz de argumentos termina en `Tablefield2` de la tabla auxiliar.

```
        stSql = "SELECT """ + aTable(6) + """||' > '||""" + aTable(7)
+ """ FROM """ + aTable(5) + """ ORDER BY """ + aTable(6) + """"
    Else
```

El listado se basa en dos campos de tabla de dos tablas. Esta consulta corresponde al ejemplo con el código postal y la ciudad.

```
        stSql = "SELECT """ + aTable(5) + """.""" + aTable(6) + """||'
> '||""" + aTable(8) + """.""" + aTable(7) + """ FROM """ +
aTable(5) + """, """ + aTable(8) + """ WHERE """ + aTable(8) +
""".""" + stForeignIDTab2Tab1 + """ = """ + aTable(5) + """.""" +
stForeignIDTab1Tab2 + """ ORDER BY """ + aTable(6) + """"
    End If
```

Tenemos la primera evaluación para determinar las claves foráneas. Las variables `stForeignIDTab2Tab1` y `stForeignIDTab1Tab2` comienzan con el valor *ID*. Para `stForeignIDTab1Tab2`, la evaluación de la consulta anterior arroja un valor diferente, es decir, el valor de *Town_ID*. De esta manera, la construcción de la consulta anterior produce exactamente el contenido ya formulado para el código postal y la ciudad, solo mejorado por la ordenación.

Ahora se tiene que hacer enlace con los listados, para proporcionarles el contenido devuelto por las consultas. Estos listados aún no existen, ya que el diálogo en sí aún no se ha creado. El diálogo se crea primero en la memoria, usando las siguientes líneas, antes de que se visualice en la pantalla.

```
    DialogLibraries.LoadLibrary("Standard")
    oDlg =
CreateUnoDialog(DialogLibraries.Standard.Dialog_Table_purge)
```

Luego vienen las configuraciones para los campos del diálogo. Aquí, por ejemplo, está listado que se proporcionará con los resultados de la consulta anterior:

```
    oCtlList1 = oDlg.GetControl("ListBox1")
    oCtlList1.addItems(aContent(),0)
```

El acceso a los controles del diálogo se logra utilizando `GetControl` con el nombre apropiado. En los diálogos no es posible que dos controles utilicen el mismo nombre, ya que esto crearía problemas al evaluar el diálogo.

El listado se suministra con el contenido de la consulta, que se ha almacenado en la matriz `aContent()`. El listado contiene solo el contenido que se mostrará como un campo, por lo que solo se ocupará la posición 0.

Después de que se hayan completado todos los campos con el contenido deseado, se inicia el diálogo.

```
    Select Case oDlg.Execute()
    Case 1 'Case 1 significa que se ha pulsado el botón "Aceptar"
    Case 0 'Si se ha pulsado el botón "Cancelar"
```

```
        inRepetition = 0
      End Select
    Loop While inRepetition = 1
```

El diálogo se ejecuta repetidamente siempre que el valor de `inRepetition` sea 1. Esto se establece mediante la casilla de verificación correspondiente.

Aquí, en resumen, está el contenido después de hacer clic en el botón *Aceptar*:

```
    Case 1
      stInhalt1 = oCtlList1.getSelectedItem() 'Lee valor de
  Listbox1 ...
      REM ... y determina el valor de ID correspondiente.
```

El valor de ID del primer listado se almacena en la variable `inLB1`.

```
      stText = oCtlText.Text  ' Lee el valor del campo .
```

Si el cuadro de texto no está vacío, se maneja la entrada en el cuadro de texto. No se consideran ni el listado para un valor de reemplazo ni la casilla de verificación para eliminar todos los registros huérfanos. Esto queda claro por el hecho de que la entrada de texto establece que estos otros campos estén inactivos.

```
      If stText <> "" Then
```

Si el cuadro de texto no está vacío, el nuevo valor se escribe en lugar del anterior utilizando el control *ID* que ha tomado el valor de la tabla. Existe la posibilidad de dos entradas, como también es el caso en el listado. El separador es **>**. Para dos entradas en tablas diferentes, se tienen que iniciar dos comandos `UPDATE`, que se crean aquí simultáneamente y se reenvían, separados por un punto y coma.

```
      ElseIf oCtlList2.getSelectedItem() <> "" Then
```

Si el cuadro de texto está vacío y el listado 2 contiene un valor, el valor del listado 1 tiene que reemplazarse por el valor en el listado 2. Esto significa que todos los registros en las tablas para las que los registros en los listados son claves foráneas deben ser verificados y, si es necesario, escrito con una clave foránea alterada.

```
      sContent2 = oCtlList2.getSelectedItem()
      REM Lee valor del listado.
      REM Determina el ID para el valor del listado.
```

El valor de *ID* del segundo listado se almacena en la variable `inLB2`. Aquí también, las cosas se desarrollan de manera diferente dependiendo de si uno o dos campos están contenidos en el listado, y también de si una o dos tablas son la base del contenido del listado.

El proceso de reemplazo depende de qué tabla se define como la tabla que proporciona la clave foránea para la tabla principal. Para el ejemplo anterior, esta es la tabla de códigos postales, ya que *Postcode_ID* es la clave foránea que se reenvía a través de *Listbox 1* y *Listbox 2*.

```
    If stAuxilTable = aTable(5) Then
      For i = LBound(aForeignTables()) To UBound(aForeignTables())
```

Reemplazar el antiguo valor de ID por el nuevo valor de ID se vuelve problemático en las relaciones `n:m`, ya que en tales casos, el mismo valor se puede asignar dos veces. Eso puede ser lo que se desea, pero debe evitarse cuando la clave foránea forme parte de la clave primaria. Así, en la tabla *rel_Media_Author*, un artículo no puede tener el mismo autor dos veces porque la clave primaria se construye a partir de *Media_ID* y *Author_ID*. En la consulta, se buscan todos los campos clave que colectivamente tienen la propiedad UNIQUE o se definieron como claves foráneas con la propiedad UNIQUE utilizando un índice.

Si la clave foránea tiene la propiedad ÚNICA y ya está representada allí como se desea en el futuro en LB2, esa clave no se puede reemplazar.

```
stSql = "SELECT ""COLUMN_NAME"" FROM
""INFORMATION_SCHEMA"".""SYSTEM_INDEXINFO"" WHERE ""TABLE_NAME"" =
'" + aForeignTables(i,0) + "' AND ""NON_UNIQUE"" = False AND
""INDEX_NAME"" = (SELECT ""INDEX_NAME"" FROM
""INFORMATION_SCHEMA"".""SYSTEM_INDEXINFO"" WHERE ""TABLE_NAME"" =
'" + aForeignTables(i,0) + "' AND ""COLUMN_NAME"" = '" +
aForeignTables(i,1) + "')"
```

`'"NON_UNIQUE" = False'` proporciona los nombres de las columnas que son ÚNICAS. Sin embargo, no se necesitan todos los nombres de columna, sino solo aquellos que forman un índice con el campo de clave foránea. La Subselección maneja esto con los mismos nombres de tabla (que contienen la clave foránea) y los nombres de los campos de clave foránea.

Si ahora la clave foránea está presente en el conjunto, el valor de la clave solo se puede reemplazar si se utilizan otros campos para definir el índice correspondiente como UNIQUE. Cuando realice reemplazos, tiene que tener cuidado de que la singularidad de la combinación de índices no se vea comprometida.

```
If aForeignTables(i,1) = stFieldname Then
  inUnique = 1
Else
  ReDim Preserve aColumns(inCount)
  aColumns(inCount) = oQuery_result.getString(1)
  inCount = inCount + 1
End If
```

Todos los nombres de columna, aparte de los nombres de columna conocidos para campos de clave foránea como Índice con la propiedad UNIQUE, se almacenan en la matriz. Como el nombre de columna del campo de clave foránea también pertenece al grupo, se puede usar para determinar si se debe verificar la unicidad durante la modificación de datos.

```
If inUnique = 1 Then
  stSql = "UPDATE """ + aForeignTables(i,0) + """ AS ""a"" SET """ +
aForeignTables(i,1) + """='" + inLB2 + "' WHERE """ +
aForeignTables(i,1) + """='" + inLB1 + "' AND ( SELECT COUNT(*) FROM
""" + aForeignTables(i,0) + """ WHERE """ + aForeignTables(i,1) +
"""='" + inLB2 + "' )"
  If inCount > 0 Then
    stFieldgroup = Join(aColumns(), """|| ||""")
```

Si hay varios campos, aparte del campo de clave foránea, que juntos forman un índice UNIQUE, se combinan aquí para una agrupación SQL. De lo contrario, solo `aColumns(0)` aparece como `stFieldgroup`.

```
    stFieldname = ""
    For ink = LBound(aColumns()) To UBound(aColumns())
      stFieldname = stFieldname + " AND """ + aColumns(ink) + """ =
""a"".""" + aColumns(ink) + """ "
```

Las partes SQL se combinan para una subconsulta correlacionada.

```
    Next ink
  stSql = Left(stSql, Len(stSql) - 1)
```

La consulta anterior termina con un paréntesis. Ahora se debe agregar más contenido a la subconsulta, por lo que este cierre tiene que eliminarse. Después de eso, la consulta se expande con las condiciones adicionales.

```
    stSql = stSql + stFieldName + "GROUP BY (""" + stFieldgroup + """)
) < 1"
End If
```

Si la clave foránea no tiene conexión con la clave primaria o con un índice UNIQUE, no importa si el contenido está duplicado.

```
Else
  stSql = "UPDATE """ + aForeignTables(i,0) + """ SET """ +
aForeignTables(i,1) + """='" + inLB2 + "' WHERE """ +
aForeignTables(i,1) + """='" + inLB1 + "'"
End If
oSQL_Statement.executeQuery(stSql)
NEXT
```

La actualización se lleva a cabo mientras existan diferentes conexiones a otras tablas; es decir, siempre que la tabla actual sea la fuente de una clave foránea en otra tabla. Este es el caso dos veces para la tabla *Town*: en la tabla *Media* y en la tabla *Postcode*l.

Posteriormente, el valor anterior se puede eliminar del listado 1, ya que ya no tiene ninguna conexión con otras tablas.

```
stSql = "DELETE FROM """ + aTable(5) + """ WHERE ""ID""='" + inLB1 +
"'"
oSQL_Statement.executeQuery(stSql)
```

En algunos casos, se tiene que llevar a cabo el mismo método para una segunda tabla que ha proporcionado datos para los listados. En nuestro ejemplo, la primera tabla es la tabla *Postcode* y la segunda es la tabla *Town*.

Si el cuadro de texto está vacío y el listado 2 tampoco contiene nada, verificamos si la casilla correspondiente indica que se deben eliminar todas las entradas excedentes. Esto significa las entradas que no están vinculadas a otras tablas por una clave foránea.

```
ElseIf oCtlCheck1.State = 1 Then
  stCondition = ""
  If stAuxilTable = aTable(5) Then
    For i = LBound(aForeignTables()) To UBound(aForeignTables())
      stCondition = stCondition + """ID"" NOT IN (SELECT """ +
aForeignTables(i,1) + """ FROM """ + aForeignTables(i,0) + """) AND
"
    Next
  Else
    For i = LBound(aForeignTables2()) To UBound(aForeignTables2())
      stCondition = stCondition + """ID"" NOT IN (SELECT """ +
aForeignTables2(i,1) + """ FROM """ + aForeignTables2(i,0) + """)
AND "
    Next
  End If
```

El último AND tiene que eliminarse, ya que de lo contrario la instrucción de eliminación terminaría con AND.

```
    stCondition = Left(stCondition, Len(stCondition) - 4)'
    stSql = "DELETE FROM """ + stAuxilTable + """ WHERE " +
stCondition + ""
    oSQL_Statement.executeQuery(stSql)
```

Como la tabla ya se ha purgado una vez, el índice de la tabla puede verificarse y, opcionalmente, corregirse hacia abajo. Consulte el procedimiento descrito en una de las secciones anteriores.

```
Table_index_down(stAuxilTable)
```

Posteriormente, si es necesario, se puede actualizar el listado en el formulario desde el que se llamó al diálogo *Table_purge*. En algunos casos, se debe volver a leer todo el formulario. Para este fin, el registro actual se determina al comienzo del procedimiento para que después de que se haya actualizado el formulario, el registro actual se pueda restablecer.

```
    oDlg.endExecute() 'Fin del dialogo ...
    oDlg.Dispose() '... y liberación de memoria
End Sub
```

Los diálogos se terminan con el comando endExecute() y se eliminan por completo de la memoria con Dispose().

Escribir macros con Access2Base

Las versiones de LibreOffice de la 4.2 en adelante tienen Access2Base integrado. Esta biblioteca presenta una capa básica con su API (interfaz de programación de aplicaciones) específica entre el código del usuario y la interfaz UNO habitual. La API proporcionada no trae en sí nuevas funcionalidades pero, en muchos casos, es más legible, concisa y más fácil de usar que UNO.

La API se parece mucho a la diseñada por Microsoft para el software Access. Aunque Base y Access tienen mucho en común, no así sus estilos de programación nativos. Access2Base llena el vacío.

Puede encontrar una documentación en inglés con ejemplos en http://www.access2base.com/access2base.html

Para dar algunos ejemplos de cómo Access2Base oculta la complejidad de UNO:

- La propiedad de valor (Access2Base sencillo) de un control tiene en UNO como equivalentes, dependiendo del tipo de control o su ubicación en un formulario, un control de tabla o un diálogo: *CurrentValue*, *Date*, *EffectiveValue*, *HiddenValue*, *ProgressValue*, *RefValue*, *ScrollValue*, *SpinValue*, *State*, *StringItemList*, *Text*, *Time*, *ValueItemList* o … *Value*.
- Para obtener los `N` primeros registros de una tabla o una consulta en una matriz Basic, se puede usar simplemente >el método *GetRows (N)* en un objeto *Recordset*. Comparado con los métodos *getString*, *getNull*, *getDouble*, *getLong*, ... en UNO que debe aplicar en los campos según su tipo y el sistema de base de datos utilizado.

Hay dos categorías principales de objetos manejados por Access2Base, que apuntan a:

- *La interfaz de usuario*: métodos utilizados normalmente desde una aplicación Base
- *Acceso a la base de datos*: métodos utilizados desde una aplicación Base o desde cualquier otra aplicación LibreOffice

Para acceder a la biblioteca, adjunte el siguiente procedimiento de una aplicación Base al suceso `OpenDocument` de su archivo Base:

```
Sub DBOpen(Optional oEvent As Object)
  If GlobalScope.BasicLibraries.hasByName("Access2Base") then
```

```
    GlobalScope.BasicLibraries.loadLibrary("Access2Base")
  End If
  Call Application.OpenConnection(ThisDatabaseDocument)
End Sub
```

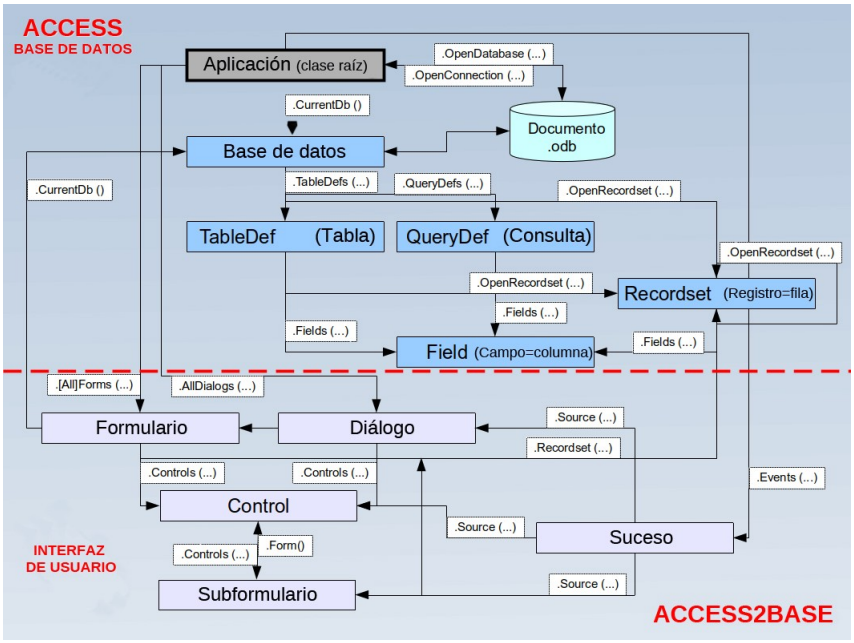
Alternativamente, para obtener acceso a la base de datos desde una aplicación que no sea Base, ejecute:

```
Sub DBOpen()
Dim myDb As Object
  If GlobalScope.BasicLibraries.hasByName("Access2Base") then
    GlobalScope.BasicLibraries.loadLibrary("Access2Base")
  End If
  Set myDb = Application.OpenDatabase(" … Nombre_archivo_de_base de
datos … ")
End Sub
```

No es la intención de esta guía replicar la documentación del sitio web mencionado anteriormente. Utilizaremos solo un resumen de los conceptos principales de la API.

El modelo Objeto

A continuación, a partir del *objeto raíz de la aplicación*, hay un esquema que describe la navegación a través de los objetos más utilizados:



Algunos ejemplos

Imprimir una lista de nombres de tablas y campos

```
Sub ScanTables()
Dim oDatabase As Object, oTable As Object, oField As Object
Dim i As Integer, j As Integer
  Set oDatabase = Application.CurrentDb()
  With odatabase
    For i = 0 To .TableDefs.Count - 1
```

```
      Set oTable = .TableDefs(i) 'Obtiene cada definición de tabla
individual
      DebugPrint oTable.Name
      For j = 0 To oTable.Fields.Count - 1
        Set oField = oTable.Fields(j) 'Obtiene cada campo individual
        DebugPrint "", oField.Name, oField.TypeName
      Next j
    Next i
  End With
End Sub
```

Almacenar los datos producidos por una consulta en una matriz básica

```
Sub LoadQuery()
Dim oRecords As Object, vData As Variant
  Set oRecords = Application.CurrentDb().OpenRecordset("myQuery")
  vData = oRecords.GetRows(1000)
  orecords.mClose()
End Sub
```

Establecer valores predeterminados en entradas de formulario

Para hacer que después de cada entrada de registro se complete previamente algún control con el último valor establecido, asigne el siguiente procedimiento al suceso *Después del cambio de registro del formulario*:

```
Sub SetDefaultNewRec(poEvent As Object)
Dim oForm As Object, oControl As Object
  Set oForm = Application.Events(poEvent).Source 'Obteniene el
formulario en uso
  Set oControl = oForm.Controls("txtCountry")
  oControl.DefaultValue = oControl.Value
End Sub
```

Funciones de base de datos

Se proporciona una colección de funciones para acortar a una sola línea el acceso a los valores de la base de datos: *DLookup, DMax*, *DMin*, *Dsum*. Todos aceptan los mismos argumentos: un nombre de campo o una expresión basada en nombres de campo, un nombre de tabla o consulta y una cláusula SQL-where sin la palabra clave WHERE. Por ejemplo:

```
Function Lookup(psField As String, psSearchField As String,
psSearchValue As String) As Variant
  Lookup = Application.DLookup(psField, "myTable", _
         psSearchField & "='" & psSearchValue & "'")
End Function
```

Comandos especiales

El `DoCmd` (2ª clase raíz) propone un conjunto de funciones que permiten ejecutar en una declaración Basic acciones complejas aunque frecuentes y prácticas. Por nombrar unas pocas:

`CopyObject`	Copia una tabla o una consulta dentro de la misma base de datos o entre dos bases de datos.

`OpenSQL`	Ejecuta una instrucción SQL SELECT dada y muestre el resultado en una hoja de datos.
`OutputTo`	Almacena los datos de una tabla o consulta en un archivo HTML. Almacena el contenido real de un formulario en un archivo PDF.
`SelectObject`	Activa la ventana dada (formulario, informe…)
`SendObject`	Envía por correo adjunto con el formulario.

LibreOffice

Guía de Base

Capítulo 10
Mantenimiento de bases de datos

Observaciones generales sobre el mantenimiento de bases de datos

La alteración frecuente de los datos en una base de datos, en particular muchas eliminaciones, tiene dos efectos. Primero, la base de datos crece de manera constante aunque en realidad no contenga más datos. En segundo lugar, la clave primaria creada automáticamente continúa incrementándose independientemente de si es realmente necesario o no. El mantenimiento importante se describe en este capítulo.

Compactar una base de datos

El comportamiento de HSQLDB es preservar el espacio de almacenamiento para los registros eliminados. Las bases de datos que están llenas de datos de prueba, especialmente si incluyen imágenes, conservan el mismo tamaño incluso si todos estos registros se eliminan posteriormente. Esto se debe a una propiedad de las claves principales de cada tabla. El archivo de la base de datos contiene el último valor utilizado para cada clave primaria. Cuando se crea un nuevo registro dentro de una tabla, se le asigna el siguiente valor.

Para liberar este espacio de almacenamiento, los registros de la base de datos deben reescribirse (tablas, descripciones de tablas, etc.). Esto se puede hacer abriendo cada tabla y eliminando todos sus registros. Se debe tener cuidado al tratar con tablas vinculadas.

Use **Herramientas > Relaciones** para determinar qué tabla debe tener sus datos eliminados. Observe las dos tablas. La que tiene su clave principal como parte de la relación es la tabla cuyos datos deben eliminarse. Cierre el diálogo *Relaciones*. Seleccione el icono *Tablas* en la ventana principal de Base. Luego haga doble clic en la tabla para mostrar los datos. Elimine sus datos. Guarde la tabla y luego la base de datos. Después de hacer esto, se necesita que se escriba en el archivo de documento de la base de datos: para hacer esto, cierre LibreOffice. Esto también compactará los archivos de la base de datos.

Cierre LibreOffice y vuelva a abrirlo si va a usarlo nuevamente.

Restablecer valores automáticos

Se crea una base de datos, todas las funciones posibles se prueban con ejemplos y se realizan correcciones hasta que todo funcione. En este momento, en promedio, muchos valores de la clave primaria habrán aumentado a más de 100. Sería contraproducente si la clave primaria se ha configurado para incrementarse automáticamente, como es común. Si las tablas se vacían en preparación para el uso normal o antes de pasar la base de datos a otra persona, la clave primaria continúa incrementándose desde su posición actual, en lugar de restablecerse a cero.

La siguiente orden SQL, ingresada usando **Herramientas > SQL**, permite restablecer el valor inicial:

```
ALTER TABLE "NombreDeLaTabla" ALTER COLUMN "ID" RESTART WITH New value
```

Esto supone que el campo de la clave primaria tiene por nombre «ID» y se ha definido como un campo de autonumeración. El nuevo valor debe ser el que desea que se cree automáticamente para el próximo nuevo registro. Así, por ejemplo, si los registros actuales suben a 4, el nuevo valor debería ser 5 sin alterar el campo ID. El primer valor de ID será el nuevo valor en la orden SQL anterior.

Consultar propiedades de la base de datos

Toda la información en las tablas de la base de datos se almacena en forma de tabla en una parte separada de HSQLDB. Se puede llegar a esta área separada utilizando el nombre INFORMATION_SCHEMA.

La siguiente consulta se puede utilizar para encontrar nombres de campos, tipos de campos, tamaños de columna y valores predeterminados. Aquí hay un ejemplo para una tabla llamada *Searchtable*.

```
SELECT "COLUMN_NAME", "TYPE_NAME", "COLUMN_SIZE", "COLUMN_DEF" AS "Default Value" FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" WHERE "TABLE_NAME" = 'Searchtable' ORDER BY "ORDINAL_POSITION"
```

Todas las tablas especiales en HSQLDB se describen en el Apéndice A de este libro. La información sobre el contenido de estas tablas se obtiene más fácilmente mediante consultas directas:

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS"
```

El asterisco garantiza que se muestren todas las columnas disponibles de la tabla. La tabla buscada arriba proporciona información esencial sobre las claves principales de las distintas tablas.

Esta información es útil sobre todo para macros. En lugar de tener que proporcionar información detallada sobre cada tabla o base de datos recién creada, los procedimientos se escriben para obtener esta información directamente de la base de datos y, por lo tanto, son de aplicación universal. La base de datos de ejemplo muestra esto, entre otras cosas, en uno de los módulos de mantenimiento, donde se determinan las claves externas.

Exportar datos

A la par con la posibilidad de exportar datos abriendo el archivo `*.odb`, hay un método mucho más simple. Directamente en la interfaz Base, puede usar **Herramientas > SQL** para ingresar una orden simple que, en las bases de datos del servidor, está reservado para el administrador del sistema.

```
SCRIPT 'database name'
```

Esto crea una extracción SQL completa de la base de datos con todas las definiciones de tabla, relaciones entre tablas y registros. No se accede a las consultas y formularios, ya que se crearon en la interfaz de usuario y no se almacenan en la base de datos interna. Sin embargo, todas las vistas están incluidas.

Nota

Este procedimiento se puede usar para actualizar una base de datos integrada para conectarse a la base de datos con HSQLDB 2.50. Nuevamente, las consultas y los formularios tienen que ser reemplazados.

Por defecto, el archivo resultante es un archivo de texto normal. También se puede exportar en binario o comprimido (en formato comprimido), especialmente para grandes bases de datos. Sin embargo, esto hace que volver a importarlo en LibreOffice sea algo más complicado.

El formato del archivo exportado se puede cambiar usando:

```
SET SCRIPTFORMAT {TEXT | BINARY | COMPRESSED};
```

Para exportar el archivo es necesario usar este código SQL línea por línea:

```
SCRIPT ‘database name’;
```

```
SET SCRIPTFORMAT {TEXT | BINARY | COMPRESSED}:
SHUTDOWN SCRIPT;
CHECKPOINT;
```

Esto exporta el archivo de texto *NombreDeLaBaseDeDatos* en la carpeta de inicio con la información de la base de datos.

El archivo se puede leer usando **Herramientas > SQL**, creando una nueva base de datos con los mismos datos. En el caso de una base de datos interna, las siguientes líneas deben eliminarse antes de la importación:

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
SET SCHEMA PUBLIC
```

Estas entradas tratan con el perfil de usuario y otras configuraciones predeterminadas, que ya están configuradas para las bases de datos internas de LibreOffice. Como resultado, aparece un mensaje de error si alguna de estas líneas está presente. Estas se encuentran directamente antes del contenido que se insertará en las tablas con la orden INSERT.

Para importar este archivo, su contenido debe dividirse en múltiples archivos de texto creados por un simple programa de edición de texto. El primer archivo debe contener todas las tablas y vistas de creación. Copie todas las líneas de la primera línea que comienza con CREATE TABLE en la línea que está arriba de la línea que contiene INSERT INTO. Pegue esto en el primer archivo. Luego, copie y pegue el resto del archivo en el segundo archivo.

Hay un límite para el tamaño del segundo archivo: debe ser inferior a 65 kB. Si es más grande, también debe dividirse en archivos de texto más pequeños cortando y pegando. Solo asegúrese de que la línea superior de cada uno de estos archivos nuevos comience con INSERT INTO. Una forma de hacer esto es cortar de abajo hacia arriba a esa línea.

Tablas de prueba para entradas innecesarias

Una base de datos consta de una o más tablas principales, que contienen claves foráneas de otras tablas. En la base de datos de ejemplo, estas son las tablas *Media* y *Address*. En la tabla *Address*, la clave principal del código postal aparece como una clave externa. Si una persona se muda a un nuevo hogar, la dirección cambia. El resultado puede ser que ya no exista una clave externa *Postcode_ID* correspondiente a este código postal. Por lo tanto, en principio, el código postal en sí podría eliminarse. Sin embargo, no es común durante el uso normal que el registro ya no sea necesario. Hay varias formas de evitar que surja este tipo de problema.

Probar entradas utilizando la definición de relación

La integridad de los datos se puede garantizar al definir las relaciones. En otras palabras, puede evitar que la eliminación o alteración de las claves provoque errores en la base de datos. Se puede acceder al siguiente diálogo a través de **Herramientas > Relaciones**, seguido de un clic derecho en el conector entre dos tablas (Ver Figura 457223).

Aquí se consideran las tablas *Address* y *Street*. Todas las acciones especificadas se aplican a la tabla *Address*, que contiene la clave externa *Street_ID*. Las opciones de actualización se refieren a una actualización del campo *ID* en la tabla *Street*. Si la clave numérica en el campo "Street"."ID" es alterada, la opción *Ninguna acción* indica que la base de datos resiste este cambio si aparece un "Street "."ID" con ese número de clave como clave foránea en la tabla *Address*.

Actualización en cascada indica que el número clave simplemente se hereda. Si la calle «Burgring» en la tabla *Street* tiene la ID «3» y también está representada en

"Address"."Street_ID", la ID se puede modificar de forma segura, por ejemplo a «67», los valores correspondientes de "Address"."Street_ID" se cambiarán automáticamente a «67».

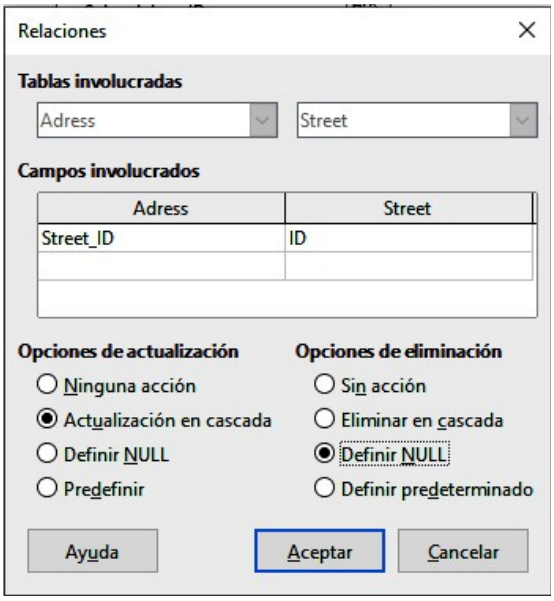


Figura 223

Si se elige *Definir NULL*, la modificación de la ID hace que "Address"."Street_ID" sea un campo vacío.

Las opciones de eliminación a la derecha funcionan de manera similar.

Para ambas opciones, la posibilidad de establecer valores predeterminados no está permitida en la GUI actualmente, ya que las configuraciones predeterminadas de la GUI son diferentes de las de la base de datos. Ver *Capítulo 3, Tablas*.

La definición de relaciones ayuda a mantener las relaciones limpias, pero no elimina los registros innecesarios que proporcionan su clave principal como clave externa en la relación. Para el ejemplo anterior, puede haber cualquier número de calles sin las direcciones correspondientes.

Editar entradas usando formularios y subformularios

En principio, toda la interrelación entre tablas se puede mostrar dentro de los formularios. Esto es más fácil, por supuesto, cuando una tabla está relacionada con solo otra tabla. Por lo tanto, en el siguiente ejemplo, la clave principal de *Author* se convierte en la clave externa en la tabla *rel_Media_Author*. La tabla *rel_Media_Author* también contiene una clave foránea de *Media*, de modo que la siguiente disposición muestra una relación n:m con tres formularios. Cada una se presenta a través de una tabla.

La primera figura muestra que el libro *I hear you knocking* pertenece al autor Dave Edmunds. Por lo tanto, Dave Edmunds no debe eliminarse; de lo contrario, faltará la información requerida para *I hear you knocking*. Sin embargo, Listado le permite elegir un registro diferente en lugar de Dave Edmunds.

Figura 224

En el formulario hay un filtro incorporado cuya activación puede indicarle qué categorías no son necesarias en la tabla *Media* (Ver Figura 225). En el caso que se acaba de describir, casi todos los autores de ejemplo están en uso. Solo el registro de Erich Kästner se puede eliminar sin consecuencias para ningún otro registro en los medios.

Figura 225

El filtro está codificado en este caso. Se encuentra en las propiedades del formulario (Ver Figura 226). Dicho filtro se activa automáticamente cuando se inicia el formulario. Se puede apagar y encender. Si se elimina, se puede acceder nuevamente mediante una recarga completa del formulario.

Esto implica más que solo actualizar los datos; todo el documento del formulario debe cerrarse y luego volverse a abrir.

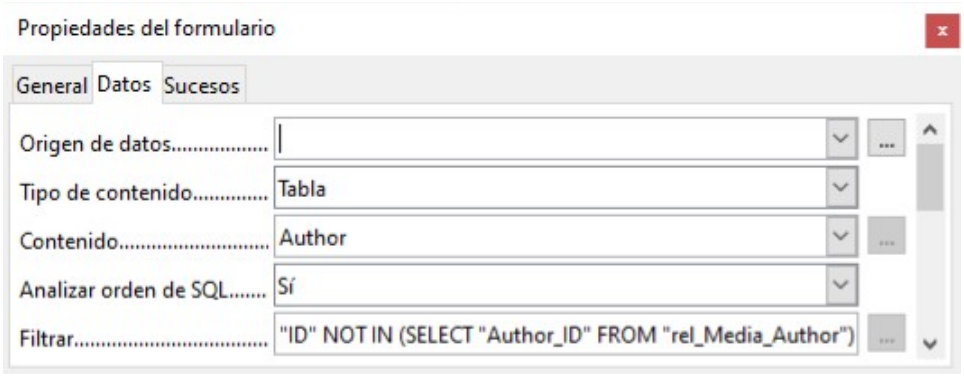


Figura 226

Consultas para encontrar entradas huérfanas

El filtro anterior es parte de una consulta que se puede usar para buscar entradas huérfanas.

```
SELECT "Surname", "Firstname" FROM "Author" WHERE "ID" NOT IN (SELECT "Author_ID“ FROM "rel_Media_Author")
```

Si una tabla contiene claves externas de otras tablas, la consulta debe ampliarse en consecuencia. Esto afecta, por ejemplo, a la tabla *Town*, que tiene claves foráneas tanto en la tabla *Media* como en la tabla *Postcode*. Por lo tanto, los registros de la tabla *Town* que se van a eliminar no deben referenciarse en ninguna de estas tablas. Esto está determinado por la siguiente consulta:

```
SELECT "Town" FROM "Town" WHERE "ID" NOT IN (SELECT "Town_ID“ FROM "Media") AND "ID" NOT IN (SELECT "Town_ID“ FROM "Postcode")
```

Las entradas huérfanas se pueden eliminar seleccionando todas las entradas que pasan el filtro establecido y utilizando la opción *Eliminar* en el menú contextual del puntero de registro, que se activa haciendo clic con el botón derecho.

Velocidad de búsqueda en base de datos

Efecto de consultas

Son solo estas consultas, utilizadas en la sección anterior para filtrar datos, las que resultan insatisfactorias con respecto a la velocidad máxima de búsqueda en una base de datos. El problema es que en bases de datos grandes, la subconsulta recupera una cantidad de datos grande con la que se debe comparar cada registro visualizable. Solo las comparaciones con la relación IN permiten comparar un solo valor con un conjunto de valores. La consulta:

```
... WHERE "ID" NOT IN (SELECT "Author_ID“ FROM "rel_Media_Author")
```

puede contener una gran cantidad de claves foráneas posibles de la tabla *rel_Media_Author*, que primero debe compararse con las claves primarias de la tabla *Authors* para cada registro de esa tabla. Por lo tanto, dicha consulta no es adecuada para el uso cotidiano, pero puede ser necesaria para el mantenimiento de la base de datos. Para el uso diario, las funciones de búsqueda deben construirse de manera diferente para que la búsqueda de datos no sea demasiado larga y no dañe el trabajo con la base de datos.

Efecto de Listados y Cuadros combinados.

Cuantos más controles de formulario de listados estén integrados en un formulario, y cuanto más datos contengan, más tardará en cargarse el formulario, ya que estos listados deben crearse.

Cuanto mejor configure la interfaz gráfica parque inicialmente lea el contenido del listado solo parcialmente, menos demora habrá.

Los listados se crean mediante consultas, y estas consultas deben ejecutarse cuando se carga el formulario para cada listado.

La misma estructura de consulta para más listados se realiza mejor usando una vista común, en lugar de crear campos con la misma sintaxis usando repetidamente las órdenes SQL almacenados en los listados. Las vistas son, sobre todo, preferibles para sistemas de bases de datos externas, ya que aquí el servidor se ejecuta significativamente más rápido que una consulta que debe ser reunida por la GUI y recién puesta en el servidor. El servidor trata las vistas como consultas locales completas.

Influencia del sistema de base de datos utilizado

La base de datos interna HSQLDB está configurada para garantizar que Base y Java funcionen bien juntos.

Desafortunadamente, a partir de la versión 3.5 de LibreOffice, Base tuvo problemas con la velocidad de procesamiento precisamente en esta área. Esto se volvió particularmente notable cuando se trataba de tablas grandes con varios miles de registros. Estos problemas tuvieron varias causas y solo desaparecieron con la versión 4.1.

Las bases de datos externas se ejecutan significativamente más rápido. Las conexiones directas a MySQL o PostgreSQL y las conexiones que utilizan ODBC, se ejecutan prácticamente a la misma velocidad. JDBC también depende de la cooperación con Java, pero aún funciona más rápido que una conexión interna usando HSQLDB.

LibreOffice

Guía de Base

Apéndice A
Tareas comunes de bases de datos

Códigos de barras

Para poder utilizar la función de impresión de código de barras, se debe instalar la fuente ean13.ttf. Esta fuente está disponible gratuitamente.

Los códigos de barras EAN13 se pueden crear usando ean13.ttf de la siguiente manera:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Número	Mayúsculas, A=0 B=1 etc.						*	Minúsculas, a=0 b=1 etc.						+

Vea también la consulta *Barcode_EAN13_ttf_command* en la base de datos de ejemplo: `Media_without_Macros.odb`

Tipos de datos para el editor de tablas

Enteros

Tipo	*Opción*	*HSQLDB*	*Rango*	*Almacenamiento*
Tiny Integer	TINYINT	TINYINT	2^8 = 256 (de -128 a +127)	1 Byte
Small Integer	SMALLINT	SMALLINT	2^{16} = 65536 (de -32768 a +32767)	2 Bytes
Integer	INTEGER	INTEGER | INT	2^{32} = 4294967296 (de -2147483648 a +2147483647)	4 Bytes
BigInt	BIGINT	BIGINT	2^{64}	8 Bytes

Números de punto flotante

Tipo	*Opción*	*HSQLDB*	*Rango*	*Almacenamiento*
Decimal	DECIMAL	DECIMAL	Ilimitado, hasta 50 lugares en la GUI, punto decimal fijo, precisión perfecta	variable
Number	NUMERIC	NUMERIC	Ilimitado, hasta 50 lugares en la GUI, punto decimal fijo, precisión perfecta	variable
Float	FLOAT	(DOUBLE) (lo sustituye)		
Real	REAL	REAL		
Double	DOUBLE	DOUBLE [PRECISION] | FLOAT	Ajustable, no exacto, 15 decimales máximo.	8 Bytes

Texto

Tipo	*Opción*	*HSQLDB*	*Rango*	*Almacenamiento*
Text	VARCHAR	VARCHAR	Ajustable	variable
Text	VARCHAR_IGNORECASE	VARCHAR_IGNORECASE	Ajustable, el rango afecta la ordenación	variable
Text (fix)	CHAR	CHAR | CHARACTER	Ajustable, resto del texto real reemplazado por espacios	fijo
Memo	LONGVARCHAR	LONGVARCHAR		variable

Hora

Tipo	*Opción*	*HSQLDB*	*Rango*	*Almacenamiento*
Date	DATE	DATE		4 Bytes
Time	TIME	TIME		4 Bytes
Date/ Time	TIMESTAMP	TIMESTAMP | DATETIME	Ajustable (0.6 - 6 con milisegundos)	8 Bytes

Otros

Tipo	*Opción*	*HSQLDB*	*Rango*	*Almacenamiento*
Yes/No	BOOLEAN	BOOLEAN | BIT		
Binary field (fix)	BINARY	BINARY	Como entero	fijo
Binary field	VARBINARY	VARBINARY	Como entero	variable
Image	LONGVARBINARY	LONGVARBINARY	Como entero	variable, usado para imágenes grandes
OTHER	OTHER	OTHER | OBJECT		

En las definiciones de la tabla, y cuando los tipos de datos se cambian en las consultas utilizando las funciones `convert` o `cast`, algunos tipos de datos esperan información sobre el número de caracteres (a), la precisión (g, correspondiente al número total de caracteres) y el número de decimales (d). Los tipos son `CHAR(a)`, `VARCHAR(a)`, `DOUBLE(g)`, `NUMERIC(g,d)`, `DECIMAL(g,d)` y `TIMESTAMP(g)`.

`TIMESTAMP(g)` solo puede tener dos valores: "0" y "6". "0" significa que no se almacenarán segundos en la parte decimal (décimas, centésimas …). La precisión de las marcas de tiempo solo se puede proporcionar directamente mediante órdenes SQL. Si está almacenando tiempos de algún tipo de deporte, debe configurar previamente `TIMESTAMP(6)` usando **Herramientas > SQL**.

Tipos de datos en StarBasic

Números

Tipo	*Corresponde a HSQLDB*	*Valor inicial*	*Observaciones*	*Requisitos de almacenamiento*
Integer	SMALLINT	0	2^{16} de -32768 a +32767	2 Bytes
Long	INTEGER	0	2^{32} de -2147483648 a +2147483647	4 Bytes
Single		0.0	Signo Decimal: .	4 Bytes
Double	DOUBLE	0.0	Signo Decimal: .	8 Bytes
Currency	Se asemeja a DECIMAL, NUMERIC	0.0000	4 decimales fijos	8 Bytes

Otros

Tipo	*Corresponde a HSQLDB*	*Valor inicial*	*Observaciones*	*Requisitos de almacenamiento*
Boolean	BOOLEAN	False	1 = sí, otro caso = no.	1 Byte
Date	TIMESTAMP	00:00:00	Fecha y hora	8 Bytes
String	VARCHAR	Cadena vacía	Hasta 65536 caracteres	variable
Object	OTHER	Null		variable
Variant		Vacío	Puede aceptar cualquier (otro) tipo de datos	variable

Existen grandes riesgos en la conversión de datos, especialmente con valores numéricos. Por ejemplo, las claves primarias en las bases de datos son comúnmente del tipo `INTEGER`. Si estos son leídos por una macro, la variable en la que se almacenan debe ser del tipo Long, ya que corresponde en tamaño al tipo `INTEGER` en Base. La instrucción de lectura correspondiente es `getLong`.

Funciones incorporadas y procedimientos almacenados

Las siguientes funciones están disponibles en la base de datos HSQLDB integrada. Lamentablemente, una o dos funciones solo se pueden usar cuando se elige **Ejecutar orden SQL directamente**. Esto evitará que se editen estas consultas.

Las funciones que solo se pueden usar en mandatos SQL directos están marcadas **[SQL directo - no funciona en la GUI]**.

Numérico

Como estamos tratando aquí con números de coma flotante, asegúrese de tener cuidado con la configuración de los campos en las consultas. Principalmente, la visualización de lugares decimales está restringida, por lo que en algunos casos puede haber resultados inesperados. Por ejemplo, la columna 1 puede mostrar 0.00 pero en realidad contiene 0.001, y la columna 2, 1000. Si la columna 3 está configurada para mostrar la Columna 1 * Columna 2, en realidad mostrará 1.

(Todas las funciones incluidas en esta tabla [Funcionan en la GUI]

excepto dos que están marcadas con [SQL directo - no funciona en la GUI]).

ABS(d)	Devuelve el valor absoluto de un número 'd', eliminando un signo menos cuando sea necesario.
ACOS(d)	Devuelve el arco coseno de 'd'.
ASIN(d)	Devuelve el arco-seno de 'd'.
ATAN(d)	Devuelve el arco tangente de 'd'.
ATAN2(a,b)	Devuelve el arco tangente usando coordenadas. 'a' es el valor del eje x, 'b' el valor del eje y.
BITAND(a,b)	Tanto la forma binaria de 'a' como la forma binaria de 'b' deben tener un 1 en la misma posición para producir 1 en el resultado. BITAND (3,5) produce 1: (0011 Y 0101 = 0001).
BITOR(a,b)	La forma binaria de 'a' o la forma binaria de 'b' deben tener 1 en la misma posición para producir 1 en el resultado. BITOR (3,5) produce 7: (0011 O 0101 = 0111).
CEILING(d)	Devuelve el número entero más pequeño que no es menor que 'd'.
COS(d)	Devuelve el coseno de 'd'.
COT(d)	Devuelve la cotangente de 'd'.
DEGREES(d)	Convierte radianes a grados.
EXP(d)	Devuelve e $^{'d'}$ (e: (2.718 …)).
FLOOR(d)	Devuelve el número entero más grande que no es mayor que 'd'.
LOG(d)	Devuelve el logaritmo natural en base e de 'd'.
LOG10(d)	Devuelve el logaritmo en base 10 de 'd'.

MOD(a,b)	Devuelve el resto como un número entero, en la división de 2 números enteros. MOD(11,3) devuelve 2:(3*3+2=11)
PI()	Devuelve π (3.1415…).
POWER(a,b)	Devuelve 'a' elevado a 'b', POWER(2,3) = 8: (2^3 = 8).
RADIANS(d)	Convierte los grados 'd' a radianes.
RAND()	Devuelve un número aleatorio mayor o igual a 0.0 y menor a 1.0.
ROUND(a,b)	Redondea los decimales de 'a' a 'b' decimales.
ROUNDMAGIC(d)	Resuelve problemas de redondeo, que surgen del uso de números de coma flotante. 3.11-3.1-0.01 no es exactamente 0, pero se muestra como 0 en la GUI. ROUNDMAGIC lo convierte en un valor cero real.
SIGN(d)	Devuelve -1, si 'd' es menor que 0, 0 si 'd' es igual a 0 y 1 si 'd' es mayor que 0.
SIN(A)	Devuelve el seno de un ángulo 'A' en radianes.
SQRT(d)	Devuelve la raíz cuadrada de 'd'.
TAN(A)	Devuelve la tangente de un ángulo 'A' en radianes.
TRUNCATE(a,b)	Trunca 'a' a 'b' decimales. TRUNCATE (2.37456,2) = 2.37

Texto

ASCII(s)	Devuelve el código ASCII de la primera letra de la cadena 's'.
BIT_LENGTH(str)	Devuelve la longitud de la cadena de texto 'str' en bits.
CHAR(c)	Devuelve la letra correspondiente al código ASCII de 'c'.
CHAR_LENGTH(str)	Devuelve la cantidad de caracteres del texto 'str'.
CONCAT(str1,str2)	Concatena str1 y str2.
'str1'||'str2'||'str3' o 'str1'+'str2'+'str3'	Concatena str1 + str2 + str3 (alternativa más simple a CONCAT).
DIFFERENCE(s1,s2)	Devuelve la diferencia de sonido entre 's1' y 's2'. Solo se emite un número entero. '0' significa que suenan igual. 'For' y 'four' producen 0, 'king' y 'wing' producen 1, ''see' y 'sea' producen 0.
HEXTORAW(s1)	Traduce el código hexadecimal a otros caracteres.

INSERT(s,start,len,s2)	Devuelve una cadena de texto, con parte del texto reemplazado Del texto 's', comenzando en 'start', se corta una longitud 'len' y se reemplaza por el texto 's2'. INSERT (Bundesbahn, 3, 4, mmel) convierte Bundesbahn en Bummelbahn, (la longitud del texto insertado puede ser mayor que la del texto eliminado sin causar ningún problema). INSERT (Bundesbahn, 3, 5, s und B) produce 'Bus und Bahn'.
LCASE(s)	Convierte una cadena a minúsculas.
LEFT(s,cantidad)	Devuelve cantidad caracteres desde el comienzo del texto s.
LENGTH(s)	Devuelve el número de caracteres del texto 's'.
LOCATE(search,s,[start])	Devuelve la primera coincidencia del término 'search' en el texto 's'. La coincidencia se da como un número: (>0 = posición si encontrado, 0 = no encontrado). Establecer el punto de inicio 'start' es opcional.
LTRIM(s)	Elimina los espacios iniciales y los caracteres de control a la izquierda de una cadena de texto 's'.
OCTET_LENGTH(str)	Devuelve la longitud de una cadena de texto en bytes. (Esto corresponde al doble de la longitud en caracteres).
RAWTOHEX(s1)	Convierte a hexadecimal, inverso de HEXTORAW().
REPEAT(s,count)	Repite una cantidad veces 'count' la cadena de texto 's'.
REPLACE(s,replace,s2)	Reemplaza todas las ocurrencias existentes de 'replace' en la cadena de texto 's' con el texto 's2'.
RIGHT(s,count)	Devuelve el número de caracteres 'count' a la derecha de la cadena de texto 's'.
RTRIM(s)	Elimina todos los espacios y caracteres de control a la derecha de una cadena de texto 's'.
SOUNDEX(s)	Devuelve un código de 4 caracteres, correspondiente al sonido de 's' –Coincide con la función DIFFERENCE().
SPACE(count)	Devuelve el número de espacios.
SUBSTR(s,start[,len])	Abreviatura de SUBSTRING.
SUBSTRING(s,start[,len])	Devuelve el texto 's' desde la posición inicial (1 = izquierda). Si se omite 'len', devuelve toda la cadena.
UCASE(s)	Convierte una cadena de texto a mayúsculas.
LOWER(s)	Como LCASE(s).
UPPER(s)	Como UCASE(s).

Fecha/Hora

CURDATE()	Devuelve la fecha actual.
CURTIME()	Devuelve la hora actual.
DATEDIFF(string, datetime1, datetime2)	Diferencia entre dos fechas: compara valores fecha / hora. 'string' determina las unidades en que se devuelve la diferencia: ms = milisegundo, ss = segundo, mi = minuto, hh = hora, dd = día, mm = mes, aa = año. Tanto las formas largas como las cortas se pueden usar para cadenas de texto.
DAY(date)	Devuelve el día del mes (1-31).
DAYNAME(date)	Devuelve el nombre del día en inglés.
DAYOFMONTH(date)	Devuelve el día del mes (1-31). Sinónimo de DAY().
DAYOFWEEK(date)	Devuelve el día de la semana como un número (1 es domingo).
DAYOFYEAR(date)	Devuelve el día del año (1-366).
HOUR(time)	Devuelve la hora (0-23).
MINUTE(time)	Devuelve el minuto (0-59).
MONTH(date)	Devuelve el mes (1-12).
MONTHNAME(date)	Devuelve el nombre del mes en inglés.
NOW()	Devuelve la fecha y la hora del momento juntas como una marca de tiempo. Alternativamente puede usar CURRENT_TIMESTAMP.
QUARTER(date)	Devuelve el trimestre del año (1-4).
SECOND(time)	Devuelve los segundos de la hora (0-59).
WEEK(date)	Devuelve la semana del año. (1-53).
YEAR(date)	Devuelve la parte del año de una fecha.
CURRENT_DATE	Sinónimo de CURDATE(), SQL-Standard.
CURRENT_TIME	Sinónimo de CURTIME(), SQL-Standard.
CURRENT_TIMESTAMP	Sinónimo de NOW(), SQL-Standard.

Conexión de base de datos

Excepto `IDENTITY ()`, que no tiene significado en Base, todo esto puede llevarse a cabo utilizando SQL directo

DATABASE()	Devuelve el nombre de la base de datos a la que pertenece esta conexión.

USER()	Devuelve el nombre de usuario de esta conexión. **[SQL directo - no funciona en la GUI]**
CURRENT_USER	Función estándar de SQL, sinónimo de USER(). **[Funciona en la GUI]]**
IDENTITY()	Devuelve el último valor para un campo automático, que se creó en la conexión en uso. Esto se utiliza en la codificación de macros para transferir una clave primaria de una tabla que se convierte en una clave externa para otra tabla.

Sistema

IFNULL(exp,value)	Si 'exp' es NULL, 'devuelve ''value'; si no, devuelve 'exp'. Como alternativa, se puede usar COALESCE () 'exp' y 'value' deben tener el mismo tipo de datos.
CASEWHEN(exp,v1,v2)	Si 'exp' es verdadero, devuelve 'v1'; si no, devuelve 'v2'. Se puede usar CASE WHEN. (con espacio) que funciona mejor con la GUI.
CONVERT(term,type)	Convierte el término 'term' en otro tipo de datos (type).
CAST(term AS type)	Sinónimo de CONVERTIR ().
COALESCE(expr1,expr2 ,expr3,...)	Si 'expr1' no es NULL, devuelve 'expr1'; si no, se comprueba 'expr2', luego 'expr3' y así sucesivamente.
NULLIF(v1,v2)	Si 'v1' es igual a 'v2', devuelve NULL; si no, devuelve 'v1'.
CASE v1 WHEN v2 THEN v3 [ELSE v4] END	Si 'v1' es igual a 'v2', devuelve 'v3'. Si no, devuelve 'v4' o NULL, si existe la cláusula ELSE. **[SQL directo - no funciona en la GUI]**
CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END	Si 'expr1' es verdadero, devuelve 'v1' [se pueden establecer condiciones adicionales]. Si no devuelve 'v4' o NULL si no hay otra condición.
EXTRACT ({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM <date or time>)	Puede reemplazar muchas de las funciones de fecha y hora. Devuelve el año, el mes, el día, etc. de una fecha o valor de fecha / hora.
POSITION(<string expression> IN <string expression>)	Si la primera cadena está contenida en la segunda, se da el desplazamiento de la primera, de lo contrario se devuelve 0.
SUBSTRING(<string expression> FROM <numeric expression> [FOR <numeric expression>])	Produce parte de una cadena de texto desde la posición especificada en FROM, y opcionalmente hasta la longitud dada por FOR.

TRIM([{LEADING | TRAILING | BOTH}] FROM <string expression>)	Se eliminan Los espacios y caracteres de control.

Control de caracteres para usar en consultas

En consultas se puecen enlazar o concatenar dos campos, :

```
SELECT "First name", "Surname" FROM "Table"
```

se pueden convertir en un solo campo usando:

```
SELECT "First name"||' '||"Surname" FROM "Table"
```

Aquí se inserta un espacio adicional. Puede ser cualquier carácter; siempre que esté entre comillas simples (' '), se interpretará como texto. A veces es necesario insertar caracteres que de control, como nuevas líneas, por ejemplo, en la preparación de informes. Aquí se muestra una breve lista de caracteres de control, que puede ampliar consultando la página web: https://en.wikipedia.org/wiki/Control_character.

CHAR(9)	Tabulador horizontal	
CHAR(10)	Salto de línea	En Combinación de correspondencia y construcción de Informes, crea un salto de línea (Linux, Unix, Mac)
CHAR(13)	Retorno de carro	Salto de línea cuando se combina con el retorno de carro en Windows CHAR (13) || CHAR (10) También se puede usar en Linux y Mac, de ahí la variante universal.

Algunos comandos «.uno» para usar con un botón

Un botón puede contener varios comandos *.uno* directamente vinculados a él. Para este propósito, debe elegir en las propiedades del botón: **Sucesos > Ejecutar una acción**, abrir un documento o página web y luego, por ejemplo, usar `URL > .uno: RecSearch` para ejecutar la función de búsqueda.

A menudo, tendrá que elegir en la pestaña *General* establecer la propiedad *Activar al hacer clic* en *No* si la instrucción necesita que esté otro control tenga el foco, por ejemplo para usar `.uno:Paste`, que puede insertar el contenido del portapapeles.

La siguiente lista contiene solo unos pocos comandos. Todos los comandos de la barra de herramientas de navegación se pueden usar en el botón, pero también se pueden crear usando los comandos `.uno`. Puede descubrir muchos de estos comandos utilizando la grabadora de macros, que a menudo utiliza *dispatch* para acceder a ellos.

Comando .uno	***Utilizado para …***
.uno:RecSearch	Abre la función de búsqueda en un formulario.
.uno:Paste	Pega desde el portapapeles. Solo cuando Activar al hacer clic = No
.uno:Copy	Copia el contenido seleccionado en el portapapeles. Solo funciona cuando Activar al hacer clic = No
.uno:Print	Abre el diálogo de impresión para el formulario.

.uno:PrintDefault	Imprime directamentecon la impresora predeterminada.

Tablas de información para HSQLDB

Dentro de una base de datos, la información sobre todas las propiedades de la tabla y sus conexiones con otras tablas se almacenan en el área `INFORMATION_SCHEMA`. Esta información permite crear macros base que requieren muy pocos argumentos para sus procedimientos. Se proporciona un procecimiento en la base de datos de ejemplo en el módulo Mantenimiento: el procedimiento `Table_purge` para el control de diálogos.

En una consulta, se pueden proporcionar piezas individuales de información y todos los campos a los que pertenecen de la siguiente manera:

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_ALIASES"
```

A diferencia de una tabla normal, aquí es necesario utilizar `INFORMATION_SCHEMA` como prefijo del nombre apropiado de la siguiente lista:

```
SYSTEM_ALIASES
SYSTEM_ALLTYPEINFO
SYSTEM_BESTROWIDENTIFIER
SYSTEM_CACHEINFO
SYSTEM_CATALOGS
SYSTEM_CHECK_COLUMN_USAGE
SYSTEM_CHECK_CONSTRAINTS
SYSTEM_CHECK_ROUTINE_USAGE
SYSTEM_CHECK_TABLE_USAGE
SYSTEM_CLASSPRIVILEGES
SYSTEM_COLUMNPRIVILEGES
SYSTEM_COLUMNS
SYSTEM_CROSSREFERENCE
SYSTEM_INDEXINFO
SYSTEM_PRIMARYKEYS
SYSTEM_PROCEDURECOLUMNS
SYSTEM_PROCEDURES
SYSTEM_PROPERTIES
SYSTEM_SCHEMAS
SYSTEM_SEQUENCES
SYSTEM_SESSIONINFO
SYSTEM_SESSIONS
SYSTEM_SUPERTABLES
SYSTEM_SUPERTYPES
SYSTEM_TABLEPRIVILEGES
SYSTEM_TABLES
SYSTEM_TABLETYPES
SYSTEM_TABLE_CONSTRAINTS
SYSTEM_TEXTTABLES
SYSTEM_TRIGGERCOLUMNS
SYSTEM_TRIGGERS
SYSTEM_TYPEINFO
SYSTEM_UDTATTRIBUTES
SYSTEM_UDTS
SYSTEM_USAGE_PRIVILEGES
SYSTEM_USERS
```

```
SYSTEM_VERSIONCOLUMNS
SYSTEM_VIEWS
SYSTEM_VIEW_COLUMN_USAGE
SYSTEM_VIEW_ROUTINE_USAGE
SYSTEM_VIEW_TABLE_USAGE
```

La siguiente consulta ofrece una visión general completa de todas las tablas de la base de datos con tipos de campo, claves primarias y claves externas:

```
SELECT
"A"."TABLE_NAME",
"A"."COLUMN_NAME",
"A"."TYPE_NAME",
"A"."NULLABLE",
"B"."KEY_SEQ" AS "PRIMARYKEY",
"C"."PKTABLE_NAME" || '.' || "C"."PKCOLUMN_NAME" AS "FOREIGNKEY FOR"
FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" AS "A"
   LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS" AS "B"
   ON ( "B"."TABLE_NAME" = "A"."TABLE_NAME" AND "B"."COLUMN_NAME" =
   "A"."COLUMN_NAME" )
   LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_CROSSREFERENCE" AS "C"
   ON ( "C"."FKTABLE_NAME" = "A"."TABLE_NAME" AND "C"."FKCOLUMN_NAME" =
"A"."COLUMN_NAME" )
WHERE "A"."TABLE_SCHEM" = 'PUBLIC'
```

*Reparación de bases de datos *.odb*

La copia de seguridad periódica de los datos debe ser una práctica habitual cuando se usa una computadora. Las copias de seguridad son la forma más sencilla de recuperar los datos ante un fallo del sistema. Sin embargo, en la práctica esto a menudo falta.

Los formularios, consultas e informes siempre se pueden copiar utilizando el portapapeles en una nueva base de datos, siempre que se haya guardado una versión anterior de la base de datos. Pero si, por alguna razón, la base de datos ya no se puede abrir, el acceso a los datos es el problema principal.

En el caso de fallos repentinos de la computadora, puede suceder que no se puedan abrir en LibreOffice las bases de datos que en el momento del fallo estaban abiertas (bases de datos internas de HSQLDB). Cuando intenta abrir la base de datos, se le solicita un filtro correspondiente al formato.

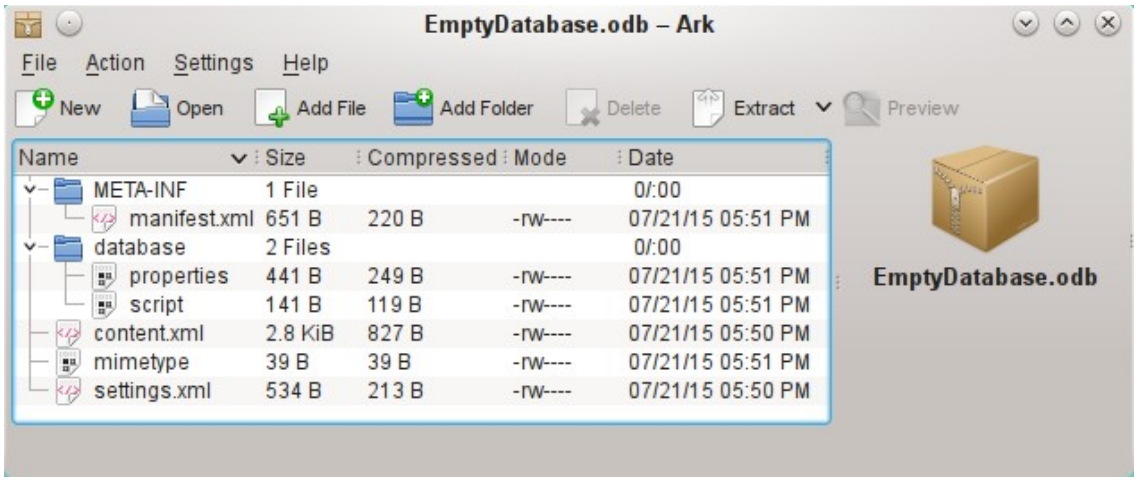
El problema surje porque parte de los datos de una base de datos abierta está en la memoria de trabajo y solo se copia de forma temporal al almacenamiento intermedio (memoria caché). Solo cuando el archivo se cierra, toda la base de datos se vuelve a empaquetar y se escribe en el archivo.

Recuperar el archivo de la base de datos

Para acceder nuevamente a sus datos, puede intentar el siguiente procedimiento:

7) Cree una copia de su base de datos para seguir los pasos indicados.
8) Intente abrir la copia con un programa de compresión de archivos. En el caso de los archivos *.odb, se utiliza un formato comprimido *Zip*. Si el archivo no se puede abrir directamente, intente cambiarle la extensión de `*.odb` a `*.zip`. Si el programa de compresión no puede abrirlo, su base de datos no se pudo guardar.

9) Después de abrir un archivo de base de datos en un programa de compresión, Verá las siguientes carpetas:



10) El archivo de la base de datos debe descomprimirse. La información más importante, en lo que respecta a los datos, se encuentra en carpeta *database*, dentro de los archivos *data* y *script*. (en la imagen no aparece el archivo data porque al tratarse de una base de datos vacía no contiene datos, ese archivo lo encontrará en la siguiente imagen)

11) Puede ser necesario buscar contradicciones en el archivo *script* . Sin embargo, este paso puede dejarse para la etapa de prueba. El archivo *script* contiene las descripciones de la estructura de las tablas.

12) Cree un nuevo archivo de base de datos vacío con LibreOffice y después de cerrarlo, abra este archivo con el programa de compresión.

13) Reemplace los archivos *data* y *script* en el nuevo archivo de base de datos con los archivos desempaquetados en el paso 4.

14) Cierre el programa de compresión. Si fue necesario cambiar la extensión del archivo a `.zip` antes de abrirlo en el programa de compresión (depende de su sistema operativo), cambie la extensión nuevamente a `.odb`.

15) Abra el archivo de la base de datos en LibreOffice. Debería poder acceder a sus tablas nuevamente.

16) Para recuperar consultas, formularios e informes de manera similar puede necesitar pruebas adicionales.

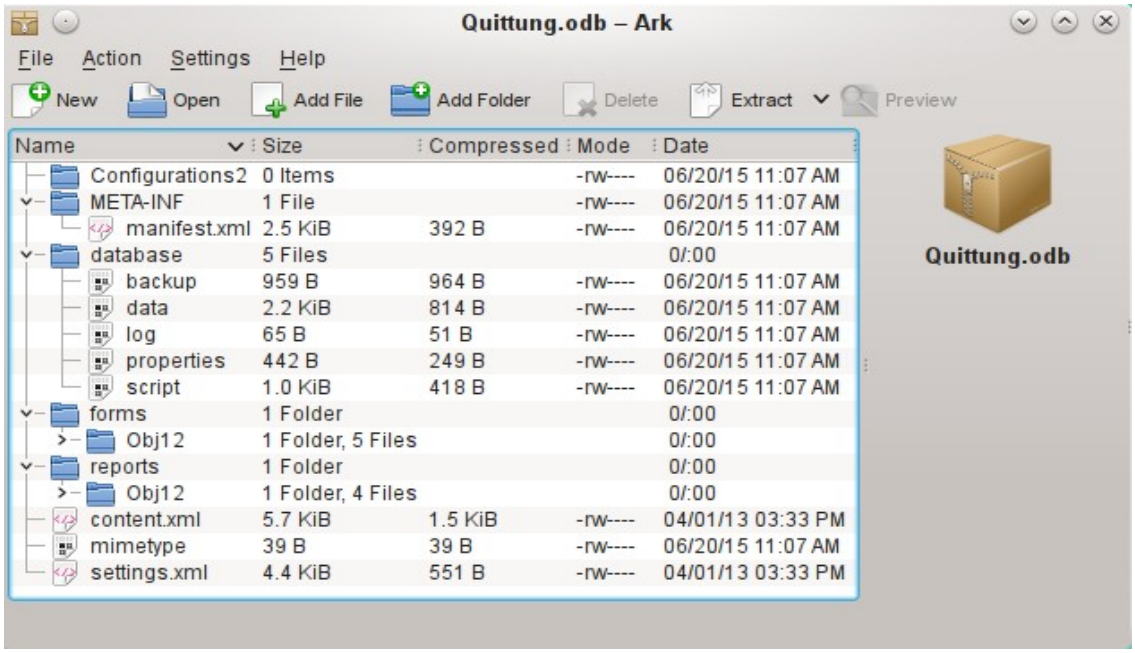
Vea también: http://forum.openoffice.org/en/forum/viewtopic.php?f=83&t=17125

Más información sobre archivos de bases de datos

En la práctica, un archivo de base de datos contiene no solo la carpeta básica para la base de datos y la carpeta `META-INF` que se especifica para el formato OpenDocument, sino también carpetas adicionales para almacenar formularios e informes.

Puede encontrar una descripción de la estructura básica del formato OpenDocument en https://en.wikipedia.org/wiki/OpenDocument_technical_specification.

La siguiente figura muestra una base de datos que contiene tablas, un formulario y un informe. No es evidente que la base de datos también contenga alguna consulta, puesto que las consultas no se almacenan en carpetas separadas sino en el archivo content.xml. La información necesaria para ejecutar una consulta es un simple fragmento de código SQL.



Archivo de base de datos que contiene información almacenada para un formulario y un informe además de la base de datos.

Aquí hay una descripción general de los ficheros que contiene un archivo (comprimido) de base de datos *.odb.

mimetype

```
application/vnd.oasis.opendocument.base
```

Este pequeño archivo de texto contiene solo la información que lo identifica como una base de datos en formato OpenDocument.

content.xml (para una base de datos sin contenido)

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content
xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0"
xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-
compatible:1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0"
xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0"
xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0"
xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0"
xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0"
xmlns:ooo="http://openoffice.org/2004/office"
```

```
xmlns:ooow="http://openoffice.org/2004/writer"
xmlns:oooc="http://openoffice.org/2004/calc"
xmlns:dom="http://www.w3.org/2001/xml-events"
xmlns:db="urn:oasis:names:tc:opendocument:xmlns:database:1.0"
xmlns:xforms="http://www.w3.org/2002/xforms"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rpt="http://openoffice.org/2005/report"
xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2"
xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:grddl="http://www.w3.org/2003/g/data-view#"
xmlns:tableooo="http://openoffice.org/2009/table"
xmlns:drawooo="http://openoffice.org/2010/draw"
xmlns:calcext="urn:org:documentfoundation:names:experimental:calc:xm
lns:calcext:1.0" xmlns:field="urn:openoffice:names:experimental:ooo-
ms-interop:xmlns:field:1.0"
xmlns:formx="urn:openoffice:names:experimental:ooxml-odf-
interop:xmlns:form:1.0"
xmlns:css3t="http://www.w3.org/TR/css3-text/"
office:version="1.2">
  <office:scripts/>
  <office:font-face-decls/>
  <office:automatic-styles/>
  <office:body>
    <office:database>
      <db:data-source>
        <db:connection-data>
         <db:connection-resource xlink:href="sdbc:embedded:hsqldb"/>
         <db:login db:is-password-required="false"/>
        </db:connection-data>
        <db:driver-settings
         db:system-driver-settings=""
         db:base-dn=""
         db:parameter-name-substitution="false"/>
        <db:application-connection-settings
         db:is-table-name-length-limited="false"
         db:append-table-alias-name="false"
         db:max-row-count="100">
         <db:table-filter>
            <db:table-include-filter>
               <db:table-filter-pattern>%</db:table-filter-pattern>
            </db:table-include-filter>
         </db:table-filter>
        </db:application-connection-settings>
      </db:data-source>
    </office:database>
  </office:body>
```

```
</office:document-content>
```

Este archivo comienza con la versión xml y el juego de caracteres utilizado. Todo lo que sigue es en realidad una sola línea. La vista preparada anteriormente debería aclarar las cosas. Los elementos que tienen una relación estrecha están agrupados entre etiquetas.

Las definiciones del principio, que comienzan con xmlns: (espacio de nombres XML) especifican los espacios de nombres a los que se puede acceder desde dentro del archivo. Luego se especifican términos algo más concretos. Aquí queda claro que estamos tratando con una base de datos interna HSQLDB, y que no se requiere una contraseña para acceder.

content.xml (para una base de datos con contenido)

El siguiente fragmento es solo un extracto del archivo `content.xml`, para aclarar su estructura.

```
<office:scripts/>
<office:font-face-decls>
  <style:font-face style:name="F" svg:font-family=""/>
</office:font-face-decls>
<office:automatic-styles>
  <style:style
    style:name="co1"
    style:family="table-column"
    style:data-style-name="N0"/>
  <style:style
    style:name="co2"
    style:family="table-column"
    style:data-style-name="N107"/>
  <style:style style:name="ce1" style:family="table-cell">
    <style:paragraph-properties fo:text-align="start"/>
  </style:style>
  <number:number-style style:name="N0" number:language="de"
      number:country="DE">
    <number:number number:min-integer-digits="1"/>
  </number:number-style>
  <number:currency-style
    style:name="N107P0"
    style:volatile="true"
    number:language="de"
    number:country="DE">
    <number:number
      number:decimal-places="2"
      number:min-integer-digits="1"
      number:grouping="true"/>
    <number:text> </number:text>
    <number:currency-symbol
      number:language="de"
      number:country="DE">€
    </number:currency-symbol>
  </number:currency-style>
```

Aquí un campo se define como un campo de moneda. Se da el número de decimales, la separación entre los números y el símbolo de la moneda, y el símbolo de la moneda.

```
  <number:currency-style
    style:name="N107"
    number:language="de"
    number:country="DE">
    <style:text-properties fo:color="#ff0000"/>
    <number:text>-</number:text>
    <number:number
      number:decimal-places="2"
      number:min-integer-digits="1"
      number:grouping="true"/>
    <number:text> </number:text>
    <number:currency-symbol
      number:language="de"
      number:country="DE">€
    </number:currency-symbol>
    <style:map style:condition="value()&gt;=0" style:apply-style-
name="N107P0"/>
  </number:currency-style>
```

Este segundo extracto establece un estilo para aplicar a la moneda en función de su valor. La moneda debe aparecer en rojo (`"#ff0000"`).

```
</office:automatic-styles>
<office:body>
  <office:database>
    <db:data-source>
```

Esta entrada del archivo `content.xml` anterior, con todas sus subentradas, se corresponde con un archivo de base de datos vacío.

```
    </db:data-source>
    <db:forms>
      <db:component
        db:name="Receipts"
        xlink:href="forms/Obj12"
        db:as-template="false"/>
    </db:forms>
```

El archivo de base de datos contiene una subsección en la que se almacenan los detalles de un formulario. El formulario tiene el nombre *Receipts* en la interfaz de usuario.

```
    <db:reports>
      <db:component
        db: name="Receipts"
        xlink:href="reports/Obj12"
        db:as-template="false"/>
    </db:reports>
```

El archivo de base de datos también contiene una subsección en la que se almacenan los detalles de un informe. El informe también tiene el nombre *Receipts* en la interfaz de usuario.

```
    <db:queries>
    <db:query
        db:name="Sales_calc"
        db:command="SELECT &quot;a&quot;.*, ( SELECT
&quot;Price&quot; *
         &quot;a&quot;.&quot;Total&quot; FROM &quot;Stock&quot;
WHERE
         &quot;ID&quot; = &quot;a&quot;.&quot;Stock_ID&quot; ) AS
         &quot;Total*Price&quot; FROM &quot;Sales&quot; AS
&quot;a&quot;"/>
    </db:queries>
```

Todas las consultas se almacenan directamente en `content.xml`, *&quot* en lenguaje xml significa comillas dobles. La consulta anterior en este ejemplo es en realidad bastante complicada con muchas subconsultas correlacionadas. Se reproduce aquí de forma abreviada.

```
    <db:table-representations>
      <db:table-representation db:name="Receipts"/>
      <db:table-representation db:name="Sales"/>
      <db:table-representation db:name="Stock">
        <db:columns>
         <db:column
            db:name="ID"
            db:style-name="co1"
            db:default-cell-style-name="ce1"/>
         <db:column
            db:name="MWSt"
            db:style-name="co1"
         db:gefault-cell-style-name="ce1"/>
         <db:column
            db:name="Price"
            db:style-name="co2"
            db:default-cell-style-name="ce1"/>
         <db:column
            db:name="Stock"
            db:style-name="co1"
            db:default-cell-style-name="ce1"/>
        </db:columns>
      </db:table-representation>
    </db:table-representations>
```

Esto muestra cómo se muestran varias tablas. Aquí se almacenan las propiedades de visualización de columnas particulares: en este ejemplo, se almacenan las configuraciones para la tabla *Stock* con sus campos: *ID*, *MWSt*, etc. Aparentemente, algo se ha ingresado directamente aquí, cambiando un poco las columnas de la tabla.

```
  </office:database>
</office:body>
```

Básicamente, `content.xml` almacena directamente el contenido de las consultas e información sobre la apariencia visual de las tablas. Además, hay una definición de la conexión de la base de datos. Finalmente llega información sobre formularios e informes.

settings.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-settings
xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
xmlns:svg="http://www.w3.org/2000/svg"
xmlns:config="urn:oasis:names:tc:opendocument:xmlns:config:1.0"
xmlns:ooo="http://openoffice.org/2004/office"
xmlns:db="urn:oasis:names:tc:opendocument:xmlns:database:1.0"
office:version="1.2"/>
```

Para una base de datos sin más contenido, solo se almacenan aquí las definiciones básicas. Con el contenido, también se almacenan varias configuraciones. Después del comienzo de la definición anterior, se almacenan las siguientes configuraciones de la base de datos de ejemplo.

```
<office:settings>
  <config:config-item-set config:name="ooo:view-settings">
    <config:config-item-set config:name="Queries">
      <config:config-item-set config:name="Calculate_sales">
        <config:config-item-set config:name="Tables">
         <config:config-item-set config:name="Table1">
            <config:config-item config:name="WindowName"
               config:type="string">Verkauf</config:config-item>
            <config:config-item config:name="WindowLeft"
               config:type="int">153</config:config-item>
            <config:config-item config:name="ShowAll"
               config:type="boolean">true</config:config-item>
            <config:config-item config:name="WindowTop"
               config:type="int">17</config:config-item>
            <config:config-item config:name="WindowWidth"
               config:type="int">120</config:config-item>
            <config:config-item config:name="WindowHeight"
               config:type="int">120</config:config-item>
            <config:config-item config:name="ComposedName"
               config:type="string">Verkauf</config:config-item>
            <config:config-item config:name="TableName"
               config:type="string">Verkauf</config:config-item>
         </config:config-item-set>
        </config:config-item-set>
        <config:config-item config:name="SplitterPosition"
         config:type="int">105</config:config-item>
        <config:config-item config:name="VisibleRows"
         config:type="int">1024</config:config-item>
      </config:config-item-set>
    </config:config-item-set>
  </config:config-item-set>
  <config:config-item-set config:name="ooo:configuration-settings">
```

```
      <config:config-item-set config:name="layout-settings">
        <config:config-item-set config:name="Tables">
          <config:config-item-set config:name="Table1">
           <config:config-item config:name="WindowName"
              config:type="string">Verkauf</config:config-item>
           <config:config-item config:name="WindowLeft"
              config:type="int">186</config:config-item>
           <config:config-item config:name="ShowAll"
              config:type="boolean">false</config:config-item>
           <config:config-item config:name="WindowTop"
              config:type="int">17</config:config-item>
           <config:config-item config:name="WindowWidth"
              config:type="int">120</config:config-item>
           <config:config-item config:name="WindowHeight"
              config:type="int">120</config:config-item>
           <config:config-item config:name="ComposedName"
              config:type="string">Verkauf</config:config-item>
           <config:config-item config:name="TableName"
              config:type="string">Sales</config:config-item>
          </config:config-item-set>
          <config:config-item-set config:name="Table2">
           ... (identical config:type-Points as "Table1"
           <config:config-item config:name="TableName"
              config:type="string">Ware</config:config-item>
          </config:config-item-set>
          <config:config-item-set config:name="Table3">
           ... (identical config:type-Points as "Table1"
           <config:config-item config:name="TableName"
              config:type="string">Receipts</config:config-item>
          </config:config-item-set>
        </config:config-item-set>
      </config:config-item-set>
    </config:config-item-set>
</office:settings>
```

Toda la visión general se relaciona con diferentes vistas de ventanas para la consulta *Calculate_sales* y las tablas *Sales*, *Stock* y *Receipts*. Los dos últimos se muestran aquí en forma abreviada. Si estas configuraciones estuvieran ausentes en un archivo `.odb` defectuoso, no importaría. Se volverían a crear la próxima vez que se abriera la ventana correspondiente.

META-INF/manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest

xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
  <manifest:file-entry
    manifest:full-path="/"
    manifest:media-type="application/vnd.oasis.opendocument.base"/>
  <manifest:file-entry
    manifest:full-path="database/script"
```

```
    manifest:media-type=""/>
  <manifest:file-entry
    manifest:full-path="database/properties"
    manifest:media-type=""/>
  <manifest:file-entry
    manifest:full-path="settings.xml"
    manifest:media-type="text/xml"/>
  <manifest:file-entry
    manifest:full-path="content.xml"
    manifest:media-type="text/xml"/>
</manifest:manifest>
```

El archivo `manifest.xml`, en la carpeta `META-INF` proporciona la carpeta de contenido para todo el archivo de la base de datos. Como este archivo trata con una base de datos vacía, solo hay cinco entradas de archivo. Un archivo de base de datos que contiene formularios e informes tendrá un archivo `META-INF` mucho más complicado.

database/properties

```
#HSQL Database Engine 1.8.0.10
#Sun Jul 14 18:02:08 CEST 2013
hsqldb.script_format=0
runtime.gc_interval=0
sql.enforce_strict_size=true
hsqldb.cache_size_scale=8
readonly=false
hsqldb.nio_data_file=false
hsqldb.cache_scale=13
version=1.8.0
hsqldb.default_table_type=cached
hsqldb.cache_file_scale=1
hsqldb.lock_file=true
hsqldb.log_size=10
modified=no
hsqldb.cache_version=1.7.0
hsqldb.original_version=1.8.0
hsqldb.compatible_version=1.8.0
```

El archivo `properties` contiene la configuración básica de la base de datos interna HSQLDB.

database/script

```
SET DATABASE COLLATION "German"
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
```

El archivo `script` contiene configuraciones predeterminadas para la conexión a la base de datos, la configuración del idioma, etc. Aquí aparece el *SA* de usuario, que se describirá más adelante.

En una base de datos con contenido, este archivo contiene las definiciones básicas de la tabla.

```
SET DATABASE COLLATION "German"
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
```

Las tablas se definen antes de que se defina el usuario de la base de datos. Primero, las tablas se crean en la memoria caché con sus campos.

```
CREATE CACHED TABLE "Stock"
   ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
   PRIMARY KEY,"Stock" VARCHAR(50),"Price" DECIMAL(8,2),"MWSt" TINYINT)
CREATE CACHED TABLE "Sales"
   ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
   PRIMARY KEY,"Total" TINYINT,"Stock_ID" INTEGER,"Receipt_ID" INTEGER,
   CONSTRAINT SYS_FK_59 FOREIGN KEY("Stock_ID") REFERENCES "Stock"("ID"))
CREATE CACHED TABLE "Receipts"
   ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
   PRIMARY KEY,"Date" DATE)
```

Luego se realizan cambios en la tabla para garantizar que las relaciones (`REFERENCES`) sean consistentes.

```
ALTER TABLE "Sales" ADD CONSTRAINT SYS_FK_76 FOREIGN KEY("Receipt_ID")
   REFERENCES "Receipts"("ID")
SET TABLE "Stock" INDEX'608 20'
SET TABLE "Sales" INDEX'1872 1656 1872 12'
SET TABLE "Receipts" INDEX'2232 1'
```

Después de establecer la posición del índice en el archivo de datos (aparece aquí solo en el archivo `script`, pero nunca se ingresa directamente en SQL), los campos de incremento automático en las tablas se configuran para que proporcionen el siguiente valor a la entrada de un nuevo registro. Suponga que el último valor ingresado en el campo *ID* de la tabla de *Stock* es 19. El incremento automático hará que el siguiente sea 20.

```
ALTER TABLE "Stock" ALTER COLUMN "ID" RESTART WITH 20
ALTER TABLE "Sales" ALTER COLUMN "ID" RESTART WITH 12
ALTER TABLE "Receipts" ALTER COLUMN "ID" RESTART WITH 1
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
```

Resolver problemas de conflicto de versiones

Si, como se describe en las páginas siguientes, se utiliza HSQLDB externo, puede haber otro problema con algunas versiones de LibreOffice en relación al archivo `.odb` conectado. Si se usa HSQLDB externo, la forma más segura de hacerlo es a través del archivo `hsqldb.jar`, que se suministra con LibreOffice. Si se utiliza un archivo diferente, puede hacer que la base de datos interna de repente se vuelva inaccesible. Esto ocurre porque LibreOffice en sus versiones 3.3 y 3.4 tuvo dificultades para distinguir entre HSQLDB interno y externo y generó advertencias de incompatibilidad entre versiones.

Si la base de datos interna ya no se puede abrir, la única solución práctica es usar LibreOffice 3.5 o posterior. Alternativamente, se debe usar una base de datos externa con el archivo `hsqldb.jar` suministrado.

La carpeta de la base de datos debe extraerse del archivo `.odb`. El archivo de `properties` tiene una entrada en la línea 11 que genera este mensaje de error en LO 3.3:

```
version=1.8.1
```

Esta línea debe cambiarse a:

```
version=1.8.0
```

Luego, la carpeta de la base de datos se vuelve a colocar en el archivo (comprimido) `.odb` y la base de datos ya se podrá abrir en LibreOffice 3.3 y 3.4.

Consejos adicionales

Si por alguna razón la base de datos se ha abierto pero no hay acceso a las tablas, puede usar **Herramientas> SQL** para ingresar la orden `SHUTDOWN SCRIPT`. La base de datos se puede cerrar y volver a abrir. Esto no funcionará si ya ha habido un mensaje de error: *Error en el archivo de script*.

Los registros en la base de datos se almacenan en el archivo `.odb` en la carpeta `database`. Aquí encontrará dos archivos llamados *data* y *Backup*. Si el archivo *data* es defectuoso, se puede restaurar desde la copia de seguridad (*Backup*). Para hacer esto, primero debe editar el archivo `properties`, que también está en la carpeta `database`. Este archivo contiene una línea: `modified = no`. Cámbiela a `modified = yes`. Lo que informa al sistema que la base de datos no se cerró correctamente. Cuando reinicie, el archivo de copia de seguridad comprimido regenerará el archivo de datos.

Conectar una base de datos a un HSQLDB externo

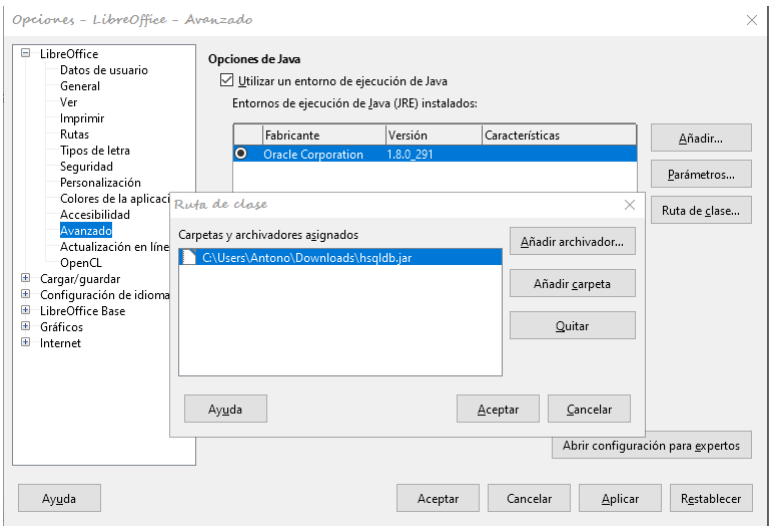
El HSQLDB interno es indistinguible de la variante externa. Si, como en la siguiente descripción, el acceso inicial a la base de datos es externo, no es necesaria ninguna función del servidor. Solo necesita el programa en java que se suministra con LibreOffice. Lo encontrará en la ruta en `/program/classes/hsqldb.jar`. El uso de este archivo es la solución más segura, ya que no tiene problemas con las versiones.

El HSQLDB externo está disponible gratuitamente para su descarga en http://hsqldb.org/. Cuando se instala la base de datos, se deben realizar los siguientes pasos en LibreOffice:

Si el controlador de la base de datos no se encuentra en la ruta de Java-Runtime, debe ingresarse como una *Ruta de clase* en **Herramientas > Opciones > Avanzado.**

La conexión a la base de datos externa usa JDBC. El archivo de la base de datos debe almacenarse en un directorio particular. Este directorio puede ser elegido libremente. En el siguiente ejemplo está en la carpeta de inicio. El resto de la ruta del directorio y el nombre de la base de datos no se proporcionan aquí.

Si los datos en la base de datos deben escribirse usando la interfaz, es importante, que al lado del nombre de la base de datos se escriba: `; default_schema = true`. Esto se puede ampliar con `; shutdown = true`, para que LibreOffice cierre la base de datos.

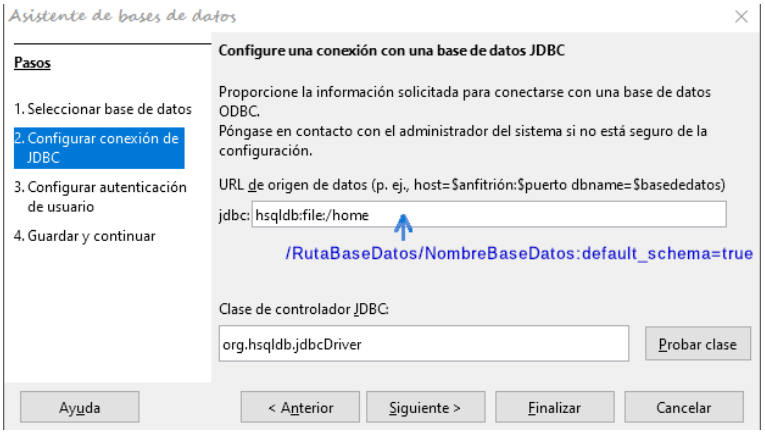


De esta manera:

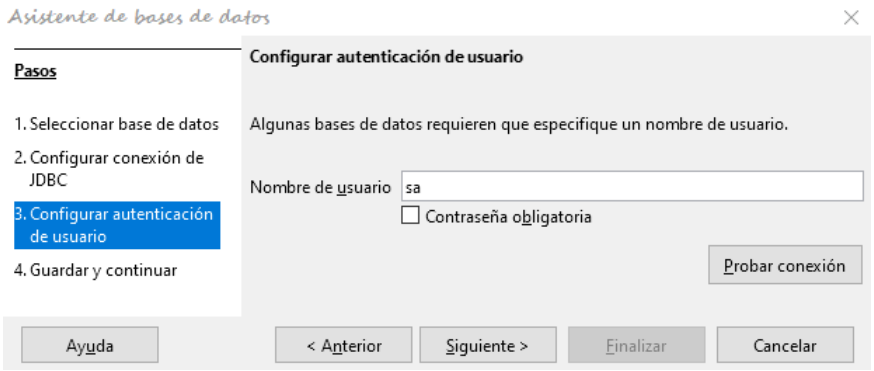
```
jdbc:hsqldb:file:/home/RutaBaseDatos/
NombreBaseDatos;default_schema=true;shutdown=true
```

En la carpeta `database` encontrará los archivos:

```
NombreBasedatos.backup
NombreBasedatos.data
NombreBasedatos.properties
NombreBasedatos.script
NombreBasedatos.log
```



El siguiente paso es dar al usuario predeterminado, si no se cambia nada en la configuración de HSQLDB:



Esto crea la conexión y la base de datos se vuelve accesible.

Precaución

Si una base de datos externa se edita con una versión de HSQLDB 2.x, ya no se puede convertir en una base de datos interna en LibreOffice. Esto se debe a funciones adicionales que no están presentes en la versión 1.8.x. Esto termina la invocación en el caso de la versión 1.8.x mientras se lee el archivo de script de la base de datos.

Del mismo modo, una base de datos externa que una vez se editó con una versión de la segunda serie no se puede editar con la versión 1.8.x, que es compatible con LibreOffice.

Instalación paralela de bases de datos HSQLDB internas y externas.

La inclusión del archivo externo hsqldb.jar en la ruta de clase puede, en algunas versiones, evitar que se abran bases de datos internas. La base se atasca porque los controladores tienen el mismo nombre y trata de usar el controlador externo para la base de datos interna. Funciona la primera vez. Pero una segunda vez, recibirá un mensaje de que la base de datos no se puede abrir ya que se escribió con una versión más nueva de HSQLDB.

Para resolver este problema, **no** coloque el archivo `hsqldb.jar`, que se necesita para bases de datos externas, en la ruta de clase de LibreOffice. La ruta de clase para este archivo debe establecerse mediante una macro, como se muestra a continuación.

```
SUB Start
  Const cPath = "/home/robby/public_html/hsqldb_test/hsqldb.jar"
  DIM oDataSource AS OBJECT
  DIM oSettings AS OBJECT
  DIM sURL AS STRING
  sURL = ConvertToURL(cPath)
  oDataSource = ThisComponent.DataSource
  oSettings = oDataSource.Settings
  oSettings.JavaDriverClassPath = sURL
END SUB
```

Aquí el archivo `hsqldb.jar` en un sistema Linux está en la ruta que se muestra arriba. Esta ruta se pasa a la base de datos que se acaba de abrir como su archivo controlador.

Este procedimiento solo se ejecuta una vez después de abrir el archivo `.odb`. Escribe la conexión correspondiente a la clase Java en el archivo `content.xml` del archivo (comprimido) `.odb`:

```
<db:data-source-settings>
  <db:data-source-setting
    db:data-source-setting-is-list="false"
    db:data-source-setting-name="JavaDriverClass"
    db:data-source-setting-type="string">
  <db:data-source-setting-value>
    org.hsqldb.jdbcDriver
  </db:data-source-setting-value>
  </db:data-source-setting>
  <db:data-source-setting
    db:data-source-setting-is-list="false"
    db:data-source-setting-name="JavaDriverClassPath"
    db:data-source-setting-type="string">
    <db:data-source-setting-value>
      file:///home/robby/public_html/hsqldb_test/hsqldb.jar
    </db:data-source-setting-value>
  </db:data-source-setting>
</db:data-source-settings>
```

La ruta podría escribirse también directamente en el archivo `content.xml` sin usar una macro. Pero, este método no resulta nada cómodo.

Cambiar la conexión de la base de datos a HSQLDB externo

Las bases de datos internas de HSQL tienen la desventaja de que el almacenamiento de datos implica un archivo comprimido. Solo cuando se comprime se escriben todos los datos. Esto puede provocar más fácilmente a la pérdida de datos que cuando se trabaja con una base de datos externa. La siguiente sección muestra los pasos necesarios para cambiar con éxito una base de datos existente de un archivo `.odb` a una versión externa en HSQL.

Haga una copia de la base de datos existente, extraiga el directorio de la base de datos. Copie el contenido en un directorio arbitrario como se describió anteriormente. Agregue el nombre de la base de datos a los nombres de archivo resultantes:

```
Databasename.backup
Databasename.data
Databasename.properties
Databasename.script
Databasename.log
```

El archivo `content.xml` debe extraerse del archivo `.odb`. Use cualquier editor de texto plano para encontrar las siguientes líneas:

```
<db:connection-data><db:connection-resource
xlink:href="sdbc:embedded:hsqldb"/><db:login
db:is-password-required="false"/></db:connection-data><db:driver-settings/>
```

Estas líneas deben reemplazarse con una conexión a una base de datos externa, en este caso una conexión a una base de datos con el nombre *Union*, en el directorio `hsqldb_data`.

```
<db:connection-data><db:connection-resource
xlink:href="jdbc:hsqldb:file:/home/robby/documents/databases/hsqldb_data/
Union;default_schema=true"/><db:login db:user-name="sa" db:is-password-
required="false"/></db:connection-data><db:driver-settings db:java-driver-
class="org.hsqldb.jdbcDriver"/>
```

Si, como se describió anteriormente, la configuración básica de HSQLDB no se dañó, el nombre de usuario y la contraseña opcional también deben coincidir.

Después de cambiar el código de `content.xml` debe volverlo a comprimir en el archivo `.odb`. El directorio de la base de datos en el archivo ahora cumplirá los requisitos. En el futuro se accederá a los datos a través de la base de datos externa.

Cambiar la conexión de la base de datos para el acceso multiusuario

Para el acceso multiusuario, HSQLDB debe estar disponible en un servidor. La forma en que se realiza la instalación del servidor varía según su sistema operativo. Para OpenSuSE, solo es necesario descargar el paquete apropiado e iniciar el servidor centralmente usando *YAST* (configuración de nivel de ejecución). Los usuarios de otros sistemas operativos y otras distribuciones de Linux probablemente puedan encontrar consejos adecuados en Internet.

El directorio de inicio en el servidor (en SuSE, `/var/lib/hsqldb`) contiene, entre otras cosas, un directorio llamado datos, en el que se archivará la base de datos, y un archivo llamado `server.properties`, que controla el acceso a las bases de datos en este directorio.

Las siguientes líneas reproducen el contenido completo de este archivo en una computadora de ejemplo. Controla el acceso a dos bases de datos: la base de datos predeterminada original (que se puede utilizar como una nueva base de datos) y la base de datos que se extrajo del archivo `.odb`.

```
# Hsqldb Server cfg file.
# See the Advanced Topics chapter of the Hsqldb User Guide.
```

```
server.database.0       file:data/db0
server.dbname.0             firstdb
server.urlid.0              db0-url
server.database.1       file:data/union
server.dbname.1             union
server.urlid.1              union-url
server.silent           true
server.trace            false
server.port             9001
server.no_system_exit   true
```

La base de datos.0 se direcciona con el nombre *firstdb*, aunque los archivos individuales en el directorio de datos comienzan con *db0*. Se agregó otra base de datos como base de datos. Aquí el nombre de la base de datos y el archivo comienzan de manera idéntica.

Las dos bases de datos se abordan de la siguiente manera:

```
jdbc:hsqldb:hsql://localhost/firstdb;default_schema=true
username sa
password
jdbc:hsqldb:hsql://localhost/union;default_schema=true
username sa
password
```

El sufijo `; default_schema = true` para la URL, que es necesario para el acceso de escritura utilizando la GUI base, se incluye de forma permanente.

Si realmente necesita trabajar en el servidor, deberá considerar si la base de datos debe estar protegida con contraseña por razones de seguridad.

Ahora puede conectarse al servidor usando LibreOffice.

Con estos datos de acceso, el servidor se puede cargar en su propia computadora. En una red con otras computadoras, debe proporcionar el nombre del host o la dirección IP al servidor.

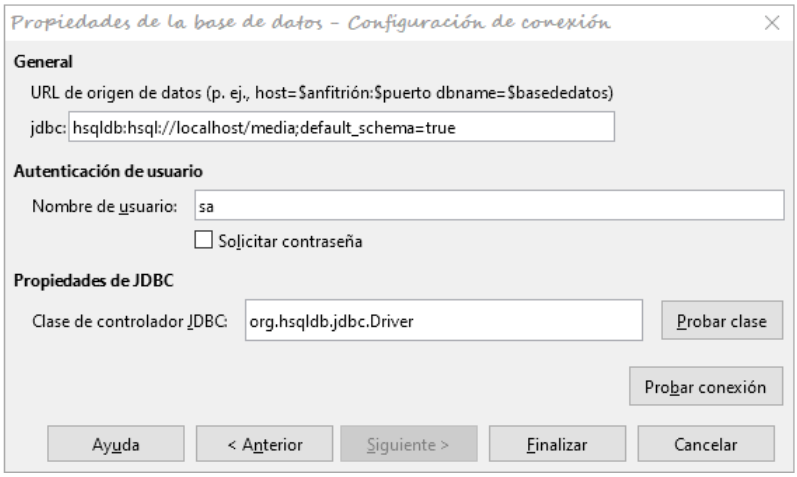
Ejemplo: una computadora tiene la `IP 192.168.0.20` y se conoce en la red con el nombre `lin_serv`. Ahora suponga que se debe ingresar otra computadora para la conexión a la base de datos:

```
jdbc:hsqldb:hsql://192.168.0.20/union;default_schema=true
```

o bien:

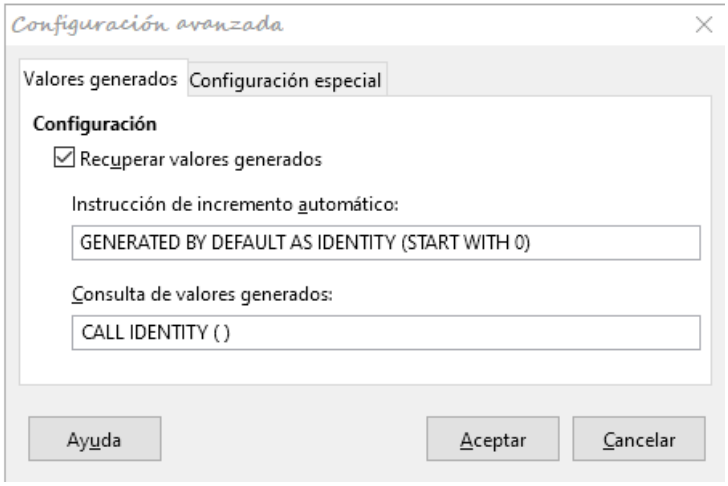
```
jdbc:hsqldb:hsql://lin_serv/union;default_schema=true
```

La base de datos ahora está conectada y podemos escribir en ella. Sin embargo, rápidamente aparece un problema adicional. Los valores generados previamente de forma automática dejan de incrementarse. Para este propósito, necesitamos una configuración adicional.



Incremento automático de valores con HSQLDB externo

Para usar valores automáticos, se necesitan diferentes procedimientos para la configuración de la tabla según la versión de LibreOffice. Común a todos ellos es la siguiente entrada bajo **Edición > Base de datos > Configuración avanzada**



Agregar `GENERATED BY DEFAULT AS IDENTITY (START WITH 0)` hace que se establezca la función de los valores de incremento automático para la clave primaria. La interfaz en LibreOffice obedece a esta orden, pero desafortunadamente antecede a la declaración con `NOT NULL`, por lo que la secuencia de órdenes HSQLDB no es legible. Debe asegurarse de que HSQLDB reciba la orden anterior para que el campo correspondiente contenga la clave primaria.

En todas sus versiones, LibreOffice, usa la orden `CALL IDENTITY ()` para leer el último valor e incrementar al siguiente . Esto le permite crear un archivo minimo `.odb`, probarlo a fondo y luego simplemente eliminar las tablas.

Todas las consultas, formularios e informes seguirán siendo utilizables, ya que la base de datos para el archivo `.odb` accede de la misma manera, y los mandatos SQL específicos se pueden usar para la base de datos HSQLDB externa (real).

Administrar la base de datos interna de Firebird

Por el momento, la base de datos interna de Firebird solo está disponible como una función experimental. Para crear una base de datos de este tipo o editar una que se haya creado, debe seleccionar **Herramientas> Opciones> LibreOffice> Avanzado> Activar funcionalidades experimentales(podrían provocar inestabilidad)**. Esta opción ilustra bien que dicha base de datos no es adecuada para el uso diario.

El siguiente enlace permite estudiar errores importantes en la base de datos interna de Firebird en cooperación con el equipo de LibreOffice: Reporting bugs for Firebird in Base.

Los usuarios notarán las siguientes diferencias con respecto a HSQLDB:

1) Si a un campo se le da el tipo Integer y luego se declara como la clave principal, parece posible darle un valor de incremento automático. Sin embargo, al guardar, esta configuración desaparece sin previo aviso.
2) Cuando se ingresan nuevos registros, no se guardan automáticamente en la base de datos. El botón Guardar debe usarse para cada entrada. En el HSQLDB incorporado, el guardado explícito de registros no es necesario.
3) Los alias se ignoran por completo en las consultas. Se puede crear un alias, pero no aparecerá en el encabezado de la tabla de la consulta.
4) No es posible crear condiciones, aunque las bases de datos externas de Firebird las admiten.
5) Los tipos de datos decimales y numéricos son defectuosos actualmente. Estos son los únicos tipos que garantizan valores precisos, especialmente cuando hay lugares decimales. Por lo tanto, son los campos preferidos para los valores de moneda. En la actualidad, solo se pueden ingresar valores con un decimal como máximo.

Hacer que los valores automáticos estén disponibles

El siguiente código, ingresado usando **Herramientas > SQL**, puede ayudar con el problema de los valores automáticos que no se proporcionan.

```
CREATE TABLE "Table1" ( "ID" INTEGER NOT NULL PRIMARY KEY, "Name" VARCHAR(20)
NOT NULL );
CREATE GENERATOR GEN_T1_ID;
SET GENERATOR GEN_T1_ID TO 0;
```

Después de esto, la ventana de entrada de SQL debe cerrarse y seleccionar **Ver > Actualizar tablas**. Solo cuando aparece la tabla, y en algunos casos solo después de un intento (sin éxito) de crear una entrada, se puede crear el siguiente disparador.

```
CREATE TRIGGER T1_BI FOR "Table1" ACTIVE BEFORE INSERT POSITION 0 AS
BEGIN
IF (NEW.ID IS NULL) THEN NEW.ID = GEN_ID(GEN_T1_ID, 1);
END;
```

Incluso con este disparador, se pueden hacer muchas entradas en la tabla y no se crea la entrada en el campo *ID*. El campo *ID* solo muestra 0. Solo cuando se pulsa *Actualizar*, se muestran los valores reales asignados. El disparador proporciona valores que comienzan con 1.

LibreOffice

Guía de Base

Apéndice B
Comparación de HSQLDB y Firebird

Tipos de datos y funciones

Tipos de datos y funciones en HSQLDB y Firebird

Las tablas en este Apéndice están tomadas de los manuales de HSQLDB y Firebird.

- http://hsqldb.org/doc/guide/
- https://www.firebirdsql.org/en/documentation/

La información para HSQLDB interna es la misma que la mostrada en el Apéndice A de esta guía

La base de datos interna Firebird se clasifica como experimental.

Las tablas proporcionan una comparación de las funciones, especialmente las funciones que son populares en los foros, tales como:

- Añadir un cierto número de días a una fecha (DATEADD)
- Valores de varias líneas de datos agrupados en una línea de datos (LIST)

Actualmente solo están disponibles en la base de datos externa de Firebird, pero no en la versión interna.

Funciones incorporadas y procedimientos almacenados.

Las siguientes funciones están disponibles en las bases de datos integradas. Lamentablemente, una o dos funciones solo se pueden usar cuando se elige *Ejecutar orden SQL directamente*. Lo que evita que se editen estas consultas.

Las funciones que funcionan solo con órdenes SQL directas están marcadas con la frase [SQL directo: no funciona en la GUI].

Las funciones que se pueden emplear con la interfaz gráfica de usuario están marcadas con: [Funciona en la GUI].

Numéricas

Al tratar con números de coma flotante, asegúrese de tener cuidado con la configuración de los campos en las consultas. Mayormente, la visualización de lugares decimales está restringida, por lo que en algunos casos puede haber resultados inesperados. Por ejemplo, la columna 1 puede mostrar 0.00 pero en realidad contiene 0.001 y la columna 2, 1000. Si la columna 3 está configurada para mostrar la Columna 1 * Columna 2, en realidad mostrará 1.

HSQLDB		***Firebird***	
ABS(d)	Devuelve el valor absoluto de un número, eliminando un signo menos si es necesario. [Funciona en la GUI]	ABS(d)	
ACOS(d)	Devuelve el arco coseno. [Funciona en la GUI]	ACOS(d)	
ASIN(d)	Devuelve el arco seno. [Funciona en la GUI]	ASIN(d)	
ATAN(d)	Devuelve el arco tangente. [Funciona en la GUI]	ATAN(d)	
ATAN2(a,b)	Devuelve el arco tangente a través de coordenadas. 'a' es el valor del eje x, 'b' es el valor del eje y. [Funciona en la GUI]	ATAN2(x,y)	
BITAND(a,b)	Tanto la notación binaria de 'a' como la notación binaria de 'b' tienen que tener un '1' en la misma posición para producir '1' en el resultado. BITAND (3,5) devuelve 1; 0011 Y 0101 = 0001 [Funciona en la GUI]	BIN_AND(x,y [,…])	
BITOR(a,b)	La notación binaria de 'a' o la notación binaria de 'b' tienen que tener un '1' en la misma posición para producir '1' en el resultado. BITOR (3,5) devuelve 7; 0011 O 0101 = 0111 [Funciona en la GUI	BIN_OR(x,y [,z...])	
		BIN_SHL(n,exp)	N 2 exp [Funciona en la GUI]

HSQLDB		***Firebird***	
		BIN_SHR(n,exp)	n / 2 exp El resultado se muestra como un entero redondeado. [Funciona en la GUI]
		BIN_XOR(x,y [,z...])	La notación binaria de 'a' o la notación binaria de 'b' tiene que tener un '1' en la misma posición para producir '1' en el resultado. BIN_XOR (3,5) devuelve 6; 0011 X OR 0101 = 0110 [Funciona en la GUI]
CEILING(d)	Especifica el número entero más pequeño que no es menor que d. [Funciona en la GUI]	CEIL(d) CEILING(d)	
COS(d)	Devuelve el coseno. [Funciona en la GUI]	COS(radians)	Los radianes también se pueden representar usando el ángulo o convertir mediante la fórmula: radianes = (2 * PI () * ángulo / 360)
		COSH(d)	
COT(d)	Devuelve la cotangente. [Funciona en la GUI]	COT(d)	
DEGREES(d)	Convierte radianes a grados. [Funciona en la GUI]		
EXP(d)	Devuelve e^d (e: (2.718…)). [Funciona en la GUI]	EXP(d)	
FLOOR(d)	Devuelve el entero más grande que no es mayor que d. [Funciona en la GUI]	FLOOR(d)	
LOG(d)	Devuelve el logaritmo natural en base 'e'. [Funciona en la GUI]	LN(d)	
LOG10(d)	Devuelve el logaritmo en base 10. [Funciona en la GUI]	LOG10(d)	
		LOG(base,d)	Devuelve de nuevo el logaritmo de d en cualquier base.

HSQLDB		***Firebird***	
MOD(a,b)	Devuelve el resto que resulta de dividir 2 enteros como un entero. MOD (11,3) devuelve 2 porque 3 * 3 + 2 = 11 [Funciona en la GUI]	MOD(a,b)	
PI()	Devuelve π (3.1415…) [Funciona en la GUI]	PI()	
POWER(a,b)	Devuelve la potencia a^b , POWER (2,3) = 8, porque $2^3 = 8$ [Funciona en la GUI]	POWER(x,y)	
RADIANS(d)	Convierte grados a radianes. [Funciona en la GUI]		
EDGE()	Devuelve un número aleatorio 'x' mayor o igual que 0.0 y menor que 1.0. [Funciona en la GUI]	EDGE( )	
ROUND(a,b)	Redondea el valor de 'a', con 'b' dígitos después del punto decimal. [Funciona en la GUI]	ROUND (d [,places])	Redondea después del número especificado de dígitos (places) desde el punto decimal. ROUND (123.45, 1) Devuelve 123.50 ROUND (123.45, –2) Devuelve 100.00 [Funciona en la GUI]
ROUNDMAGIC (d)	Resuelve problemas de redondeo causados por números de coma flotante. 3.11–3.1–0.01 puede no ser exactamente 0, pero se muestra como 0 en la GUI. ROUNDMAGIC lo convierte en un valor 0 real. [Funciona en la GUI]		
SIGN(d)	Devuelve –1 si 'd' es menor que 0, 0 si 'd' es igual a 0 y 1 si 'd' es mayor que 0. [Funciona en la GUI]	SIGN(d)	
SIN(A)	Devuelve el seno de un ángulo en radianes. [Funciona en la GUI]	SIN(radians)	
		Sinh(d)	
SQRT(d)	Devuelve la raíz cuadrada. [Funciona en la GUI]	SQRT(d)	

HSQLDB		*Firebird*	
TAN(A)	Devuelve la tangente de un ángulo en radianes. [Funciona en la GUI]	TAN(radians)	
		TANH(d)	
TRUNCATE (a,b)	Corta de 'a' 'b' dígitos después del punto decimal. TRUNCATE(2.37456.2) = 2.37 [Funciona en la GUI]	TRUNC(d[,jobs])	Se establece en 0 después del número especificado de dígitos desde el punto decimal. TRUNC (123.45, 1) Devuelve 123.4 0 [Funciona en la GUI]

Texto

HSQLDB		*Firebird*	
ASCII(str)	Devuelve el código ASCII de la primera letra de la cadena 'str'.[Funciona en la GUI]	ASCII_VAL ('s')	Devuelve el valor numérico que coincide con el carácter 's' ingresado. [Funciona en la GUI]
BIT_LENGTH (str)	Devuelve la longitud del texto 'str' en bits. [Funciona en la GUI]	BIT_LENGTH ('str')	
CHAR(c)	Devuelve la letra del código ASCII expresado. (Tanto de letras, como de caracteres de control). CHAR (13) crea un salto de línea en una consulta, que es visible en campos multi-línea de formulario o de informe. [Funciona en la GUI]	ASCII_CHAR (n)	Devuelve la letra que pertenece al código ASCII. (Tanto de letras como de caracteres de control). [Funciona en la GUI]
CHAR_LENGTH (str)	Devuelve el número de caracteres del texto. [Funciona en la GUI]	CHAR_LENGTH ('str') CHARACTER_LENGTH('str')	
		CHAR_TO_UUID	Convierte un Identificador Universal Único (UUID) de 36 caracteres en un UUID de 16 bytes (genera caracteres ilegibles)

HSQLDB		***Firebird***	
CONCAT (STR1,STR2)	Concatena 'str1' y 'str2'. [Funciona en la GUI]		
DIFFERENCE (s1,s2)	Muestra la diferencia de sonido entre 's1' y 's2'. La salida es un entero. 0 significa el mismo sonido. 'For' y 'four' con 0 aparecen como iguales, el ‘ shortening’ y ‘seasoning’ se establecen en 1, mouth y moon vuelven a 0 [Funciona en la GUI]		
HEXTORAW (s1)	Traduce el código hexadecimal a otros caracteres [Funciona en la GUI]		
INSERT(s, start, len,s2)	Devuelve una cadena de texto 's' con parte del texto reemplazado. Empezando en 'start', se elimina una longitud de texto 'len' que se reemplaza por el texto 's2'. INSERT (Bundesbahn, 3, 4, mmel) convierte Bundesbahn en Bummelbahn, la longitud del texto insertado puede ser mayor que la del texto eliminado sin causar ningún problema. INSERT (Bundesbahn, 3, 5, s und B) produce 'Bus und Bahn'. [Funciona en la GUI]	OVERLAY ('s' PLACING 's2' FROM pos [FOR length])	Si la posición de inicio se establece de modo que sea mayor o igual que el texto 's', la cadena 's2' se agrega directamente a 's'. OVERLAY 'Bundesbahn' PLACING 'mmel' FROM 3 FOR 4) cambia 'Bundesbahn' a 'Bummelbahn', Cuando la longitud del texto es mayor que el texto cortado: OVERLAY ('Bundesbahn'PLACING' s and B' FROM 3 FOR 5) se convierte en 'Bus und Bahn'. [NO Funciona en la GUI]
LCASE(s)	Convierte la cadena en minúsculas.[Funciona en la GUI]		
LEFT(s, count)	Devuelve el número de caracteres especificados por 'count' desde la izquierda del texto 's'. [Funciona en la GUI]	LEFT('s', length)	
LENGTH(s)	Devuelve el número total de caracteres de un texto. [Funciona en la GUI]		

HSQLDB		***Firebird***	
LOCATE (search,s [,start])	Devuelve la posición de la primera coincidencia del término 'search' en el texto 's'. La coincidencia se expresa numéricamente: (1 = izquierda, 0 = no encontrado) La especificación del punto de inicio 'start' es opcional. [Funciona en la GUI]	POSITION ('s2' IN 's') POSITION ('s2', 's' [, Startposition])	
POSITION (<string expression> IN <string expression>)	Si el primer texto está contenido en el segundo, se mostrará la posición del primer texto, de lo contrario 0 Esto podría usarse en lugar de una opción de búsqueda con LIKE. [Funciona en la GUI]		
LTRIM(s)	Elimina espacios y caracteres de control de la izquieda del texto. [Funciona en la GUI]		
OCTET_LENGTH (str)	Devuelve el número de caracteres de una cadena de texto en bytes. En principio, esto corresponde al doble de la longitud de caracteres. [Funciona en la GUI]	OCTET_LENGTH (str)	Devuelve el número de caracteres.(Los espacios también se tienen en cuenta). El número depende del juego de caracteres. UTF-8 necesita dos bytes para caracteres especiales.
RAWTOHEX(s1)	Convierte a notación hexadecimal. [Funciona en la GUI]		
REPEAT(s, count)	Repite la cadena de texto 's' un número de veces 'count'. [Funciona en la GUI]		
REPLACE(s, replace, s2)	Reemplaza todas las ocurrencias de 'replace' que existan en el texto 's' con el texto 's2'. [Funciona en la GUI]	REPLACE('s', 's2', replace)	REPLACE ('Schule', 'ul', 'eib') convierte 'Schule' en 'Scheibe'. Si 's2' no aparece en 's', no se reemplazará nada.

HSQLDB		*Firebird*	
		LPAD('s', totallength [,characters])	LPAD ('Hello',8, '+') = +++Hello Coloca los caracteres 'characters' a la izquierda de un término 's' hasta que se alcanza la longitud total. Si la longitud total es menor que la longitud del término, este se corta a la derecha. Si el tercer parámetro no se expresa, se colocan espacios delante de él.
		RPAD('s', total length [,characters])	RPAD ('Hello',8, '+') = Hello+++ Coloca los caracteres 'characters' a la derecha de un término 's' hasta que se alcanza la longitud total. Si la longitud total es menor que la longitud del término, este se corta a la derecha. Si el tercer parámetro no se expresa, se colocan espacios detrás de él.
		REVERSE('s')	Invierte la cadena 's' por completo. Esto puede ser útil, por ejemplo, si desea ordenar por caracteres finales (ej. finales de dominio).
RIGHT(s, count)	Opuesto a LEFT; devuelve el número de caracteres especificados por 'count' desde la derecha del texto. [Funciona en la GUI]	RIGHT('s', length)	
RTRIM(s)	Elimina todos los espacios y caracteres de control a la derecha del texto. [Funciona en la GUI]		
SOUNDEX(s)	Devuelve un código de 4 caracteres, que corresponde al sonido de 's' – se ajusta a la función DIFFERENCE (). [Funciona en la GUI]		
SPACE(count)	Devuelve el número de espacios especificados en 'count'. [Funciona en la GUI]		

HSQLDB		***Firebird***	
SUBSTR(s, start [,len])	Abreviatura de SUBSTRING. [Funciona en la GUI]		
SUBSTRING (s,start [,len])	Devuelve el texto 's' desde la posición inicial 'start'. (1 = izquierda). Si se omite la longitud, se devuelve todo el texto. [Funciona en la GUI]		
SUBSTRING (<string expression> FROM <numeric expression> [FOR <numeric expression>])	Devuelve la parte de un texto desde la posición inicial especificada en FROM, opcionalmente en la longitud especificada en FOR. SUBSTRING ("Roberta" FROM 3 FOR 3) da como resultado la subcadena 'ber'.[Funciona en la GUI]	SUBSTRING ('s' FROM start position [FOR length])	
TRIM ([{LEADING | TRAILING | BOTH}] FROM <string expression>)	Elimina espacios y caracteres de control. [Funciona en la GUI]	TRIM ([Where 's2' FROM] 's')	Donde: BOTH | LEADING | TRAILING El estándar es BOTH: que el carácter ‘s2’ se eliminará a ambos lados de 's'.de manera predeterminada son espacios (' '), pero también pueden ser otros caracteres. TRIM (TRAILING '!' FROM 'Hallo!') = 'Hallo'
UCASE(s)	Convierte la cadena en mayúsculas. [Funciona en la GUI]		
LOWER(s)	Como LCASE(s). [Funciona en la GUI]	LOWER('s')	
UPPER(s)	Como UCASE(s). [Funciona en la GUI]	UPPER('s')	
		UUID_TO_CHAR ('s')	Convierte un UUID de 16 bytes a un formato ASCII de 36 caracteres.

Fecha/Hora

HSQLDB		*Firebird*	
		DATEADD (n DAY TO date) DATEADD (DAY, n, date)	'n' es un número entero (puede ser negativo). YEAR | MONTH | WEEK | DAY | HOUR | MINUTE| SECOND | MILLISECOND deben usarse como términos para el intervalo de tiempo. Use un campo de fecha para fechas, un campo de hora para horas o una marca de tiempo como término de fecha
DATEDIFF (string, datetime1, datetime2)	Diferencia entre dos fechas u horas. La entrada en cadena decide en qué unidad se muestra la diferencia: 'ms' = 'milisegundos', 'ss' = 'segundo', 'mi' = 'minuto', 'hh' = 'hora', 'dd' = ' day ',' mm '=' mes ',' aa '=' año '. Se puede usar tanto en la versión larga como en la versión corta. [Funciona en la GUI]	DATEDIFF (DAY FROM date TO date) DATEDIFF (DAY, date, date)	Vea DATEADD.
EXTRACT ({YEAR|MONTH |DAY|HOUR| MINUTE| SECOND} FROM <date or time value>)	Puede reemplazar muchas de las funciones de fecha y hora. Devuelve el año, mes, día, etc. de un valor de fecha u hora. EXTRACT (DAY FROM "date") muestra el día del mes. [Funciona en la GUI]	EXTRACT ({YEAR| MONTH | WEEK | DAY | WEEKDAY | YEARDAY | HOUR | MINUTE | SECOND | MILLISECOND } FROM <date or time value>)	El día de la semana empieza en domingo = 0 Primer día del año 1º de enero = 0 Primera semana del año debe tener al menos 4 días
DAY(date)	Devuelve el día del mes. (1-3) [Funciona en la GUI]		

HSQLDB		***Firebird***	
DAYNAME (date)	Devuelve el nombre en inglés del día [Funciona en la GUI]		
DAYOFMONTH (date)	Devuelve el día del mes. (1-31), sinónimo de DÍA(). [Funciona en la GUI]		
DAYOFWEEK (date)	Devuelve el día de la semana como un número. (1 significa domingo.) [Funciona en la GUI]		
DAYOFYEAR (date)	Devuelve el día del año. (1-366) [Funciona en la GUI]		
HOUR(time)	Devuelve la hora. (0-23) [Funciona en la GUI]		
MINUTE(time)	Devuelve el minuto. (0-59) [Funciona en la GUI]		
MONTH(date)	Devuelve el mes. (1-12) [Funciona en la GUI]		
MONTHNAME (date)	Devuelve el nombre en inglés del mes. [Funciona en la GUI]		
QUARTER (date)	Devuelve el trimestre del año. (1-4) [Funciona en la GUI]		
SECOND(time)	Devuelve los segundos de un tiempo. (0-59) [Funciona en la GUI]		
WEEK(date)	Devuelve la semana del año. (1-53) [Funciona en la GUI]		
YEAR(date)	Devuelve el año desde una entrada de fecha. [Funciona en la GUI]		

Conexión con bases de datos

HSQLDB		***Firebird***	
DATABASE()	Devuelve la ruta y el nombre de la base de datos que pertenece a esta conexión. [Funciona en la GUI]		
		CURRENT_TRANSACTION	SELECT CURRENT_TRANSACTION FROM RDB $ DATABASE devuelve el número (único) de la transacción como un entero.
		CURRENT_CONNECTION	SELECT CURRENT_CONNECTION FROM RDB $ DATABASE devuelve un valor entero para la conexión actual.
		CURRENT_ROLE	SELECT CURRENT_ROLE FROM RDB $ DATABASE refleja el rol del usuario actual. Si no se define ningún rol, el resultado es 'NONE'
		RDB $ SET_CONTEXT ('<namespace>', '<variable name>', value | NULL)	Namespace: USER_SESSION | USER_TRANSACTION El nombre de la variable puede tener un máximo de 80 caracteres, el valor puede tener un máximo de 255 caracteres.
CURRENT_USER	Función SQL estándar sinónimo de USER(). [Funciona en la GUI]	CURRENT_USER	
USER()	Devuelve el nombre de usuario de esta conexión. El nombre de usuario es importante si la base de datos se va a convertir en una base de datos externa. [SQL directo: no funciona en la GUI]	USER	

HSQLDB		*Firebird*	
IDENTITY()	Devuelve el último valor para un campo de valor automático creado en la conexión actual. Se usa en programación de macros para crear una clave foránea de otra tabla a partir de una clave primaria de la principal. [Funciona en la GUI]	GEN_ID (generator name, <step>)	Los valores automáticos se crean con un 'generator'. El incemento (step) debe darse como 1. En principio, cualquier valor entero es posible. new.rec_id = gen_id (gen_recnum, 1);

Sistema

HSQLDB		*Firebird*	
IFNULL (exp, value)	Si 'exp' es 'NULL' devuelve 'value'; si no, 'exp'. En cambio, COALESCE () también se puede usar como una extensión. 'Exp' y 'value' deben tener el mismo tipo de datos. IFNULL es una función importante cuando los campos están vinculados entre sí por invocación o CONCAT. El contenido del resultado sería NULL, incluso si solo un valor es NULL. "Apellido" || ',' || "nombre" daría un resultado vacío para las personas que, por ejemplo, carecen de la entrada para "nombre", es decir, NULL. "Apellido" || IFNULL (',' || "Nombre", '') simplemente imprimiría "Apellido". [Funciona en la GUI]		
CASE WHEN (exp, v1, v2)	Si 'exp' es verdadero, se devuelve 'v1'; de lo contrario, 'v2'. También se puede usar CASEWHEN (sin espacio). CASEWHEN ("a"> 10, 'objetivo alcanzado', 'todavía en práctica') devuelve 'objetivo alcanzado' si el contenido del campo "a" es mayor que 10. [Funciona en la GUI]	IIF (<condition>, v1, v2)	

<table>
<tr><th colspan="2">HSQLDB</th><th colspan="3">Firebird</th></tr>
<tr><td>CONVERT (term, type)</td><td>Convierte 'term' a otro tipo de datos 'type'.
CONVERT ("a", DECIMAL (5,2)) hace que el campo "a" sea un campo con 5 dígitos, incluidos los 2 decimales. Si el número es demasiado grande, se genera un error.
[Funciona en la GUI]</td><td></td><td colspan="2"></td></tr>
<tr><td rowspan="5">CAST (term AS type)</td><td rowspan="5">Sinónimo de CONVERT ()
[Funciona en la GUI]</td><td rowspan="5">CAST (term AS type)</td><td>De</td><td>A</td></tr>
<tr><td>Tipos numéricos</td><td>Tipos numéricos [VAR] CHAR
BLOB</td></tr>
<tr><td>[VAR] CHAR
BLOB</td><td>[VAR] CHAR
BLOB
Tipos numéricos
DATE
TIME
TIMESTAMP</td></tr>
<tr><td>DATE
TIME</td><td>[VAR] CHAR
BLOB
TIMESTAMP</td></tr>
<tr><td>TIMESTAMP</td><td>[VAR] CHAR
BLOB
DATE
TIME</td></tr>
</table>

HSQLDB		***Firebird***	
COALESCE (expr1, expr2, expr3…)	Si 'expr1' no es NULL, se muestra 'expr1'; de lo contrario, se comprueba 'expr2', luego 'expr3', etc. Todas las expresiones deben tener al menos un tipo de datos similar. Esta es la representación alternativa de números enteros y de coma flotante, pero **no** de un valor de fecha u hora [Funciona en la GUI]	COALESCE (expr1, expr2 [, expr3…]	
NULLIF (v1, v2)	Si 'v1' es igual a 'v2', se muestra CERO; si no, 'v1'. Los datos deben ser de tipo compatible [Funciona en la GUI]	NULLIF (v1, v2)	
CASE v1 WHEN v2 THEN v3 [ELSE v4] END	Si 'v1' es igual a 'v2', devuelve v3'. De lo contrario, devuelve 'v4' o 'NULL' si no se formula 'ELSE'. [SQL directo: no funciona en la GUI]	DECODE (test expression, expression, result [, expression2, Earnings…] [, default expression])	DECODE (UPPER ("gender"), 'M', 'male', 'F', 'female', 'unknown')
CASE WHEN expr1 THEN v1 [WHEN expr2 THEN v2] [ELSE v4] END	Si 'expr1' es verdadero, devuelve 'v1'. [Se pueden especificar más casos] De lo contrario, devuelve 'v4' o 'NULL' si no se formula 'ELSE'. CASE WHEN DAYOFWEEK ("date") = 1 THEN 'domingo' WHEN DAYOFWEEK ("date") = 2 THEN 'lunes'… END Podría devolver el nombre del día en otro idioma a través de SQL, (en la función solo lo devuelve en inglés).[Funciona en la GUI]		DECODE (EXTRACT (WEEK DAY FROM "date"), 0 , ' domingo', 1 , 'lunes', 'etc. ')
		GEN_UUID ()	Devuelve una ID única como un conjunto de caracteres de 16 bytes.

HSQLDB		*Firebird*	
		HASH (s)	Devuelve el valor *hash* de una cadena de caracteres arbitrariamente larga. Los valores hash de las mismas cadenas de caracteres deben ser los mismos. (hash = serie de caracteres con longitud fija)
		MAXVALUE (expr [, expr ...])	Devuelve el valor máximo de una lista de valores. Funciona con cadenas, valores numéricos, valores de fecha u hora.
		MINVALUE (expr [, expr ...])	Devuelve el valor mínimo de una lista de valores. Funciona con cadenas, valores numéricos, valores de fecha u hora.

Funciones agregadas (especialmente con GROUP BY)

HSQLDB		*Firebird*	
		MAX (expr)	Valor máximo de un campo en una tabla.
		MIN (expr)	Valor mínimo de un campo en una tabla.
		LIST ([ALL | DISTINCT] 's', ['s2'])	Conecta campos de varios registros de datos a un campo con el término de conexión 's2' correspondiente . [Funciona en la GUI]

Variables (dependientes del sistema)

HSQLDB		*Firebird*	
CURRENT_TIME	Sinónimo de CURTIME (), estándar SQL [Funciona en la GUI]	CURRENT_TIME	Tiempo en horas, minutos y segundos. CURRENT_TIME (3) también especifica milisegundos.

HSQLDB		*Firebird*	
CURTIME ()	Devuelve la hora en el momento de la consulta. [Funciona en la GUI]		
CURRENT_TIMESTAMP	Sinónimo de NOW (), estándar SQL [Funciona en la GUI]	CURRENT_TIMESTAMP[(accuracy)]	Especificación de tiempo con fecha y milisegundos La precisión se puede establecer con [0 | 1 | 2 | 3] El estándar es 3 decimales. SELECCT CURRENT_TIMESTAMP (2) FROM RDB $ DATABASE devuelve la marca de tiempo con décimas y centésimas de segundo (2 decimales)
NOW ()	Devuelve la fecha y hora juntas como una marca de tiempo. CURRENT_TIMESTAMP también se puede usar en su lugar. [Funciona en la GUI]	CAST ('NOW' AS DATE | TIME | TIMESTAMP) or DATE 'NOW'	'NOW', escrito solo, se entiende como una cadena. Con la conversión adecuada, se convierte en una fecha, una hora o una marca de tiempo (cada una con 1/1000 s). La forma abreviada no funciona en la GUI.
CURRENT_DATE	Sinónimo de CURDATE (), estándar SQL [Funciona en la GUI]	CURRENT_DATE	
CURDATE ()	Devuelve la fecha actual. [Funciona en la GUI]		

Operadores y declaraciones

Tipos de datos para el editor de tablas

Enteros

Tipo	*Opción*	*HSQLDB*	*Firebird*	*Rango*	*Almacenamiento*
Tiny integer	TINYINT	TINYINT		2^8 = 256 (de -128 a +127)	1 byte

Small integer	SMALLINT	SMALLINT	SMALLINT	2^{16} = 65536 (de -32768 a +32767)	2 bytes
integer	INTEGER	INTEGER | INT	INTEGER	2^{32} = 4294967296 (de -2147483648 a +2147483647)	4 bytes
BigInt	BIGINT	BIGINT	BIGINT	2^{64} (de -2^{63} a $+2^{63}$)	8 bytes

Números de coma flotante

Tipo	*Opción*	*HSQLDB*	*Firebird*	*Extension*	*Memoria necesaria*
Decimal	DECIMAL	DECIMAL	DECIMAL (n, m)	Ilimitado, a través de GUI a 50 dígitos, decimales fijos ajustables, precisión exacta	2,4 u 8 bytes
Number	NUMERIC	NUMERIC	NUMERIC (n, m)	Ilimitado, a través de GUI a 50 dígitos, decimales fijos ajustables, precisión exacta	2,4 u 8 bytes
Float	FLOAT	(DOUBLE is used instead)	FLOAT	De $3.4 * 10^{-38}$ a $3.4 * 10^{38}$ ajustable, no exacto, 7 decimales máximo	4 bytes
Real	REAL	REAL			
Double	DOUBLE	DOUBLE [PRECISION] | FLOAT	DOUBLE PRECISION	De $1,7 * 10^{-308}$ a $1,7 * 10^{308}$ ajustable, no exacto, 15 decimales máximo	8 bytes

Texto

Tipo	*Opción*	*HSQLDB*	*Firebird*	*Extension*	*Memoria necesaria*
Text	VARCHAR	VARCHAR	VARCHAR (n)	Ajustable	Variable Firebird: de 1 a 32767 bytes

Text	VARCHAR_IGNORECASE	VARCHAR_IGNORECASE		Ajustable, afecta la clasificación, ignora las diferencias entre mayúsculas y minúsculas	variable
Text (fixed)	CHAR	CHAR | CHARACTER		Ajustable, el resto del texto real está lleno de espacios	fijo
Memo	LONGVARCHAR	LONGVARCHAR	BLOB (BLOB SUB_TYPE 1)		Variable Firebird: <32 GB

Hora

Tipo	*Opción*	*HSQLDB*	*Firebird*	*Extension*	*Memoria necesaria*
Date	DATE	DATE	DATE		4 bytes
Time	TIME	TIME	TIME	Firebird: de 0:00 a 23:59,9999	4 bytes
Date/Time	TIMESTAMP	TIMESTAMP | DATETIME	TIMESTAMP	Ajustable (HSQLDB: 0, 6 - 6 significa con milisegundos)	8 bytes

Otros

Tipo	*Opción*	*HSQLDB*	*Firebird*	*Extension*	*Memoria necesaria*
Yes No	BOOLEAN	BOOLEAN | BIT			
Binary field (fixed)	BINARY	BINARY		Como entero	fijo
Binary field	VARBINARY	VARBINARY		Como entero	variable
Image	LONGVARBINARY	LONGVARBINARY	BLOB SUB_TYPE 0 BLOB SUB_TYPE binary	Como entero	variable, destinado a imágenes mayores Firebird: <32 GB

OTHER	OTHER	OTHER | OBJECT			

LibreOffice

Equipo de documentación de LibreOffice

Guía de Base

6.2

Gestionar los datos

Acerca de este libro:

Este libro es tanto para principiantes como para usuarios avanzados de LibreOffice Base. Si nunca ha usado Base antes o si necesita una introducción a todos los componentes de LibreOffice quizá le interese leer antes la Guía de primeros pasos con LibreOffice.

Este libro introduce las funciones más comunes de LibreOffice Base:
- Crear bases de datos
- Entrada y salida de datos
- Trabajar con tablas, formularios, consultas e informes
- Usar macros
- Mantenimiento de las bases de datos
- Y mucho más

Acerca de los autores:

Este libro está escrito y traducido por voluntarios de la comunidad LibreOffice. Los beneficios de las ediciones impresas se usarán en favor de la comunidad.

Se puede descargar una versión en PDF de este libro en :
https://documentation.libreoffice.org/es/

Acerca de LibreOffice:

LibreOffice es la suite gratuita, libre y open source de The Document Foundation. Funciona en Windows, macOS, y GNU/Linux. El soporte y la documentación es gratuita en nuestra gran comunidad de usuarios, colaboradores y desarrolladores.
Puede colaborar involucrándose como voluntario en varias áreas: desarrollo, aseguramiento de la calidad, documentación, traducción, ayuda a los usuarios y más.

Puede descargar LibreOffice gratuitamente en https://es.libreoffice.org/descarga/