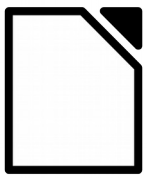


LibreOffice
The Document Foundation

Base Handbook

Chapter 7
Linking to Databases

Copyright

Contributors

Jochen Schiffers
Robert Großkopf
Jost Lange
Hazel Russman
Jean Hollis Weber

Feedback

Please direct any comments or suggestions about this document to the Documentation Team's mailing list: documentation@global.libreoffice.org

Note: Everything you send to a mailing list, including your email address and any other personal information that is written in the message, is publicly archived and cannot be deleted.

Acknowledgments

This chapter is based on an original German document and was translated by Hazel Russman.

Publication date and software version

Published 21 December 2015. Based on LibreOffice 5.0.

Note for Mac users

Some keystrokes and menu items are different on a Mac from those used in Windows and Linux. The table below gives some common substitutions for the instructions in this chapter. For a more detailed list, see the application Help.

Windows or Linux	***Mac equivalent***	***Effect***
Tools > Options menu selection	**LibreOffice > Preferences**	Access setup options
Right-click	*Control+click*	Open a context menu
Ctrl (Control)	*⌘ (Command)*	Used with other keys
F5	*Shift+⌘+F5*	Open the Navigator
F11	*⌘+T*	Open the Styles and Formatting window

Contents

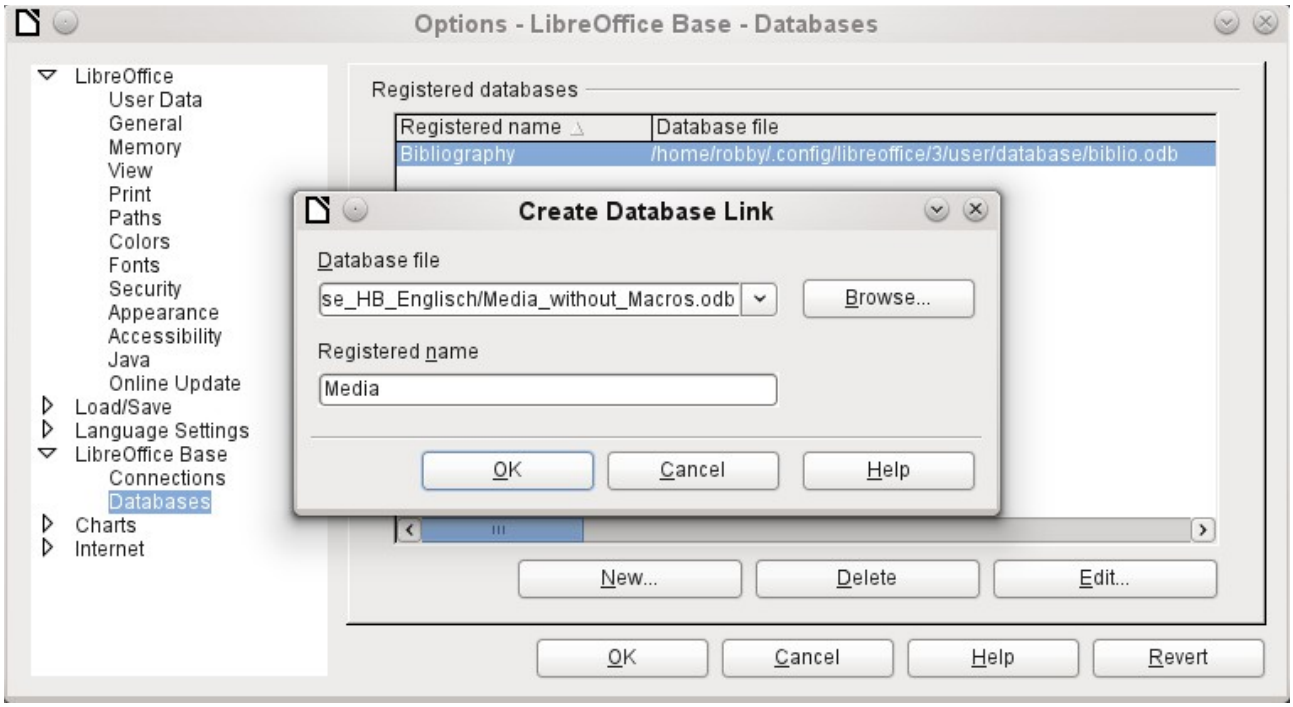
General notes on database linkage

With Base, you can use documents in LibreOffice Writer and Calc in various ways as data sources. This means that the use of Base is not necessarily tied to the registration of databases in the configuration of LibreOffice. External forms can also interact directly with Base, provided that the path to the data sources is supplied.

Registration of databases

Many functions, such as printing labels or using data for form letters, require the registration of a database in the configuration of LibreOffice.

Using **Tools > Options > LibreOffice Base > Databases > New**, you can register a database for subsequent use by other LibreOffice components.

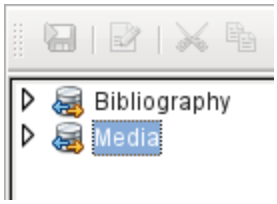


The database is found using the file browser and connected to LibreOffice in a similar way as for a simple form. Give the database itself a suitably informative name, for example the name of the database file. The name serves as an alias, which can also be used in queries to the database.

Data source browser

The data source browser provides access to tables and queries of all registered databases under their registered names. To open the browser, use **View > Data sources**, or press the *F4* key, or click the icon on the standard toolbar.

Registered data sources are shown on the left side of the data source browser. The Bibliography data source is included in LibreOffice by default. The other data sources are listed by their registered names.

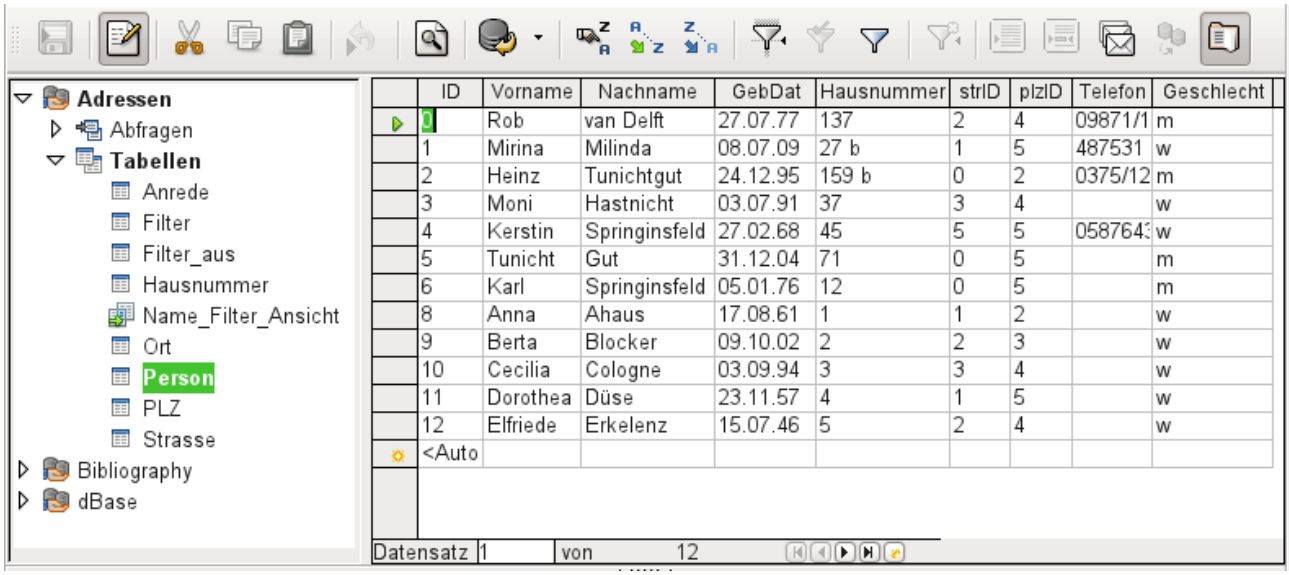


Click on the expansion sign in front of the database name to open the database and show sub-folders for queries and tables. Other sub-folders of the database are not made available here. Internal forms and reports can only be accessed by opening the database itself.

Only when you click on the Tables folder is the database actually accessed. For databases protected by a password, the password must be entered at this point.

To the right of the name tree, you can see the table you have selected. It can be edited just as in Base. However, direct entry into tables should be carried out with caution in very complex relational databases, as the tables are linked together with foreign keys. For example, the database shown below has separate tables for street names, postcodes, and towns.

For a proper view of the data (but without the ability to edit), queries or views are more suitable.



Many of the icons in the toolbar (Figure 1) will be familiar from data entry into tables. The main new ones are those in the last section: *Data to Text, Data to Fields, Mail Merge, Data Source of Current Document, Explorer on/off*.

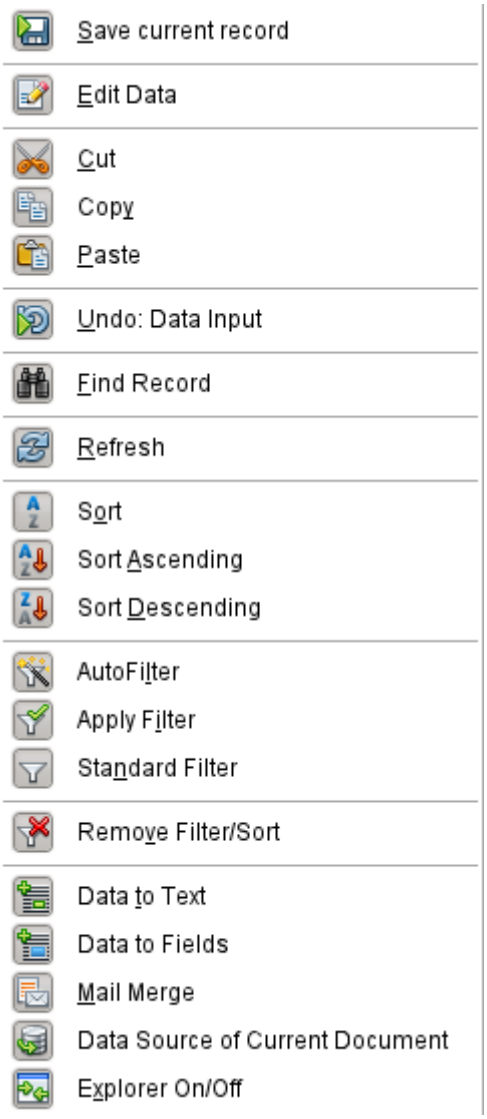


Figure 1: Data source browser toolbar

Data to Text

The *Data to Text* function is available as soon as a record is selected.

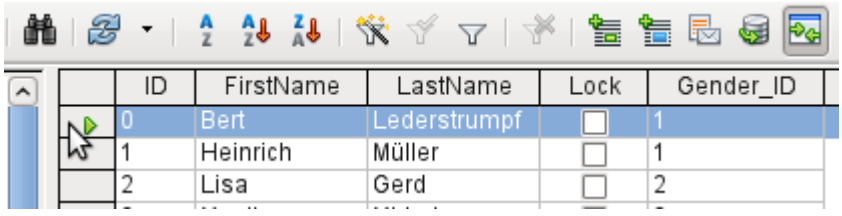


Figure 1: Selecting a data record

If you now choose Data to Text, a Wizard appears to carry out the necessary formatting.

There are several possibilities for entering data as text: as a table, as single fields, or as ordinary text.

Figure 2 shows the option **Insert Data as Table**. In the case of numeric and date fields, the database format can be changed to a chosen format. Otherwise, formatting is carried out automatically when the table fields are selected. The sequence of fields is adjusted by using the arrow keys.

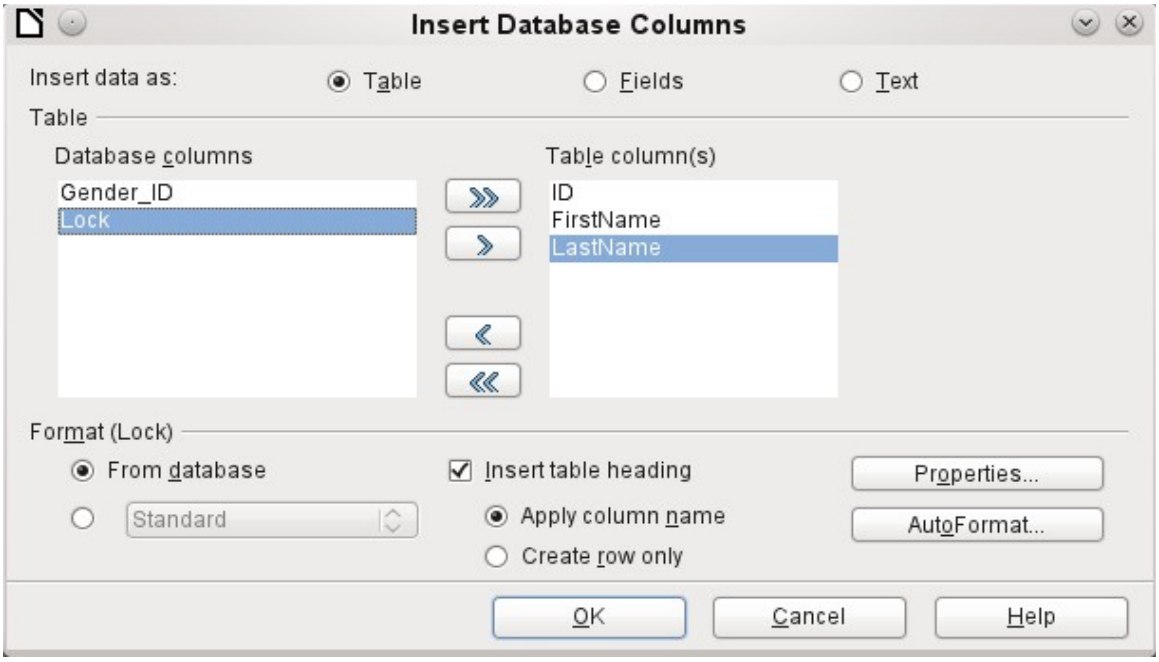


Figure 2: Data entry as table

As soon as table columns have been selected, the Properties button for the table is activated. This allows you to set the usual table properties for Writer (table width, column width, and so on).

The checkbox determines if a table heading is required. If it is not checked, no separate row will be reserved for headings.

The row chosen for the table heading can be taken from the column names, or the record may be written out with space left for the headings to be edited in later.

The AutoFormat button provides several pre-formatted table views. Apart from the suggested *Default* format, all the formats can be renamed. You can also add autoformats; to do this, , first create a table in the required format. Then select the table and click the Add button to add its format to the list.

The table is finally created with the selected columns.

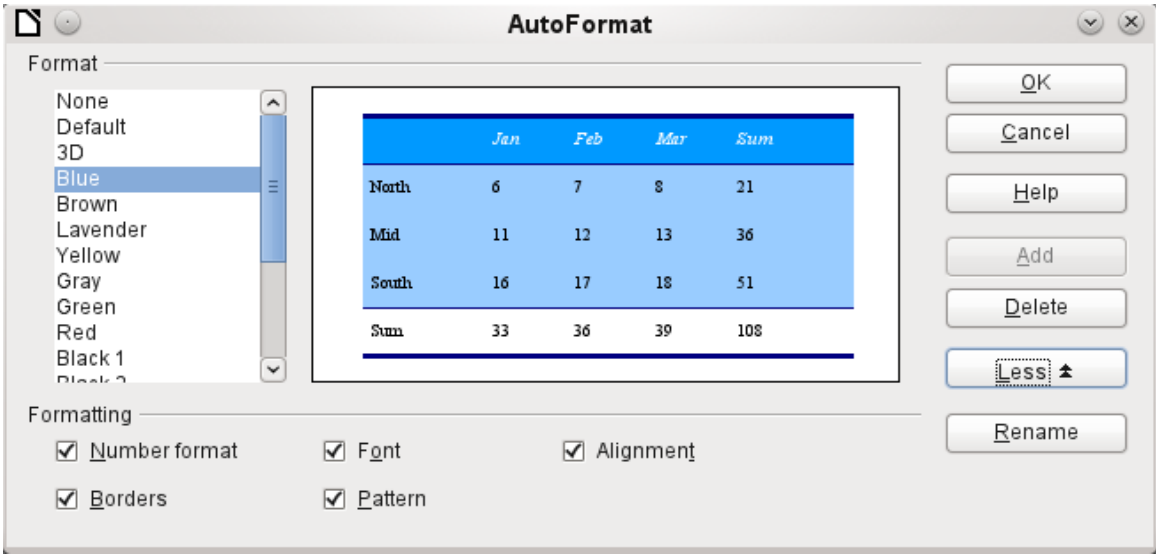


Figure 2: AutoFormat provides a choice of table formats

Insert data as Fields provides the possibility of using a mini-editor to position the various table fields successively in the text. The text created in this way can also be provided with a paragraph style. In this case too, the formatting of dates and numbers can be specified separately, or can be read directly from the table settings in the database.

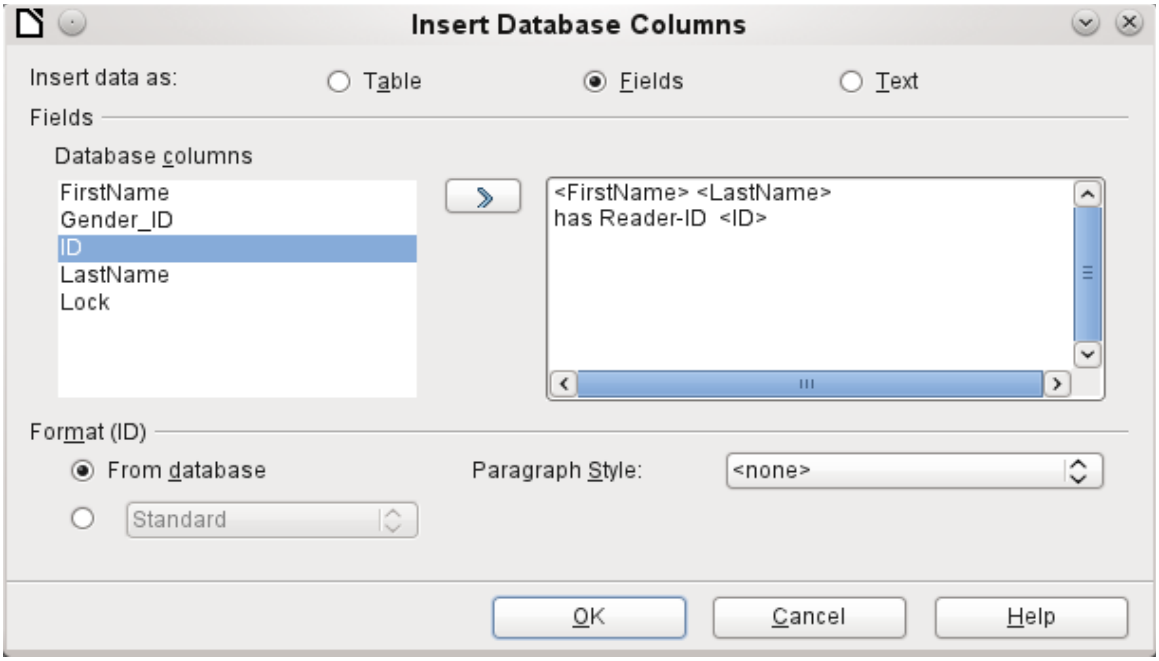


Figure 3: Insert data as Fields – corresponds also to the dialog for Insert data as Text

The fields inserted into the text in this way can subsequently be deleted singly or used for a mail merge.

If you choose **Insert data as Text**, the only difference from using fields is that fields remain linked to the database. When you insert as text, only the content of the specified fields is transferred and not the link to the actual database.

The results of the two procedures are compared below.

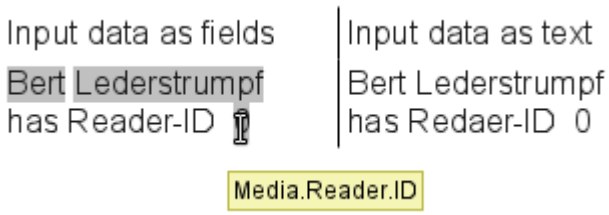


Figure 4: Comparison of Data as Fields and Data as Text

The fields have a gray background. If you hover the mouse cursor over the fields, a tooltip shows that the fields are linked to the *Media* database, to the table *Reader* and, within this table, to the field *ID*.

So, for example, a double-click on the field *ID* opens the following overview. This makes it clear which field was created through the *Insert Data as Fields* procedure. It is the same field type that is shown by **Insert > Fields > Other > Database**.

It is simpler to create such a field by selecting the column header of the table in the data source browser and dragging it into the document with the mouse. You can create a form letter directly in this way.

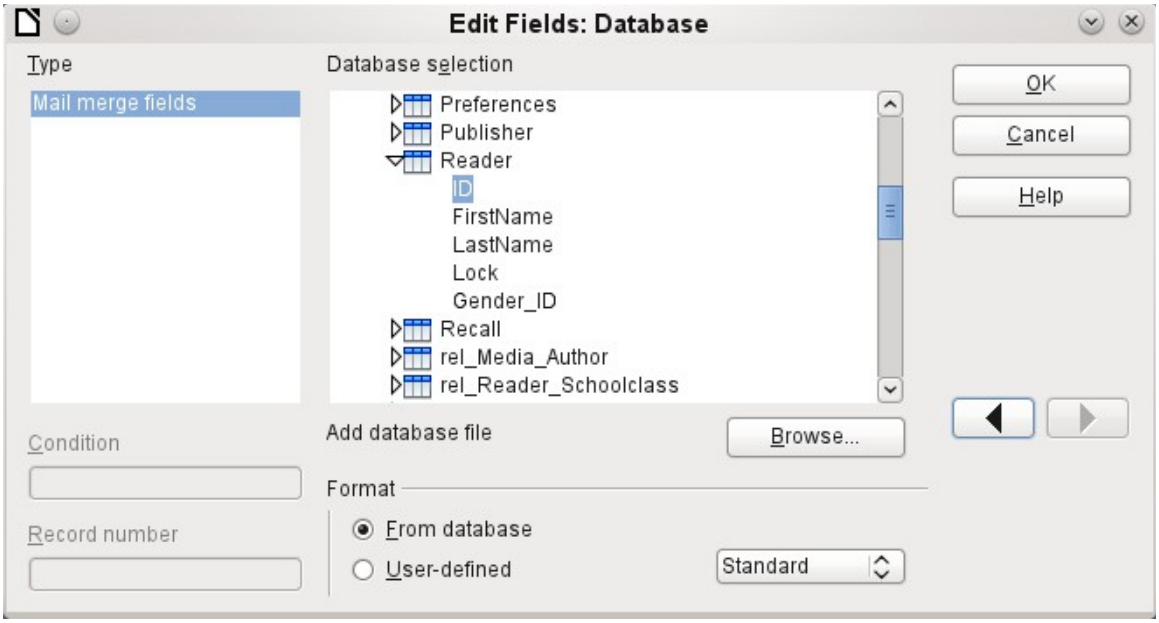


Figure 5: Double-click on an inserted field to show the properties of the Mail Merge fields

Data as Fields

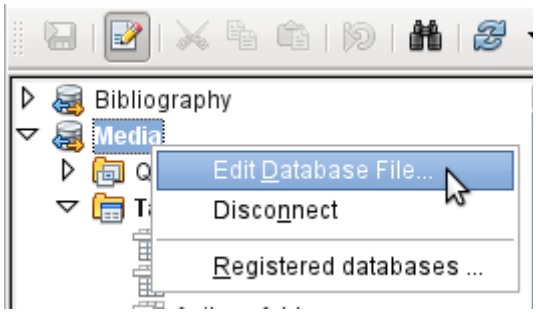
Data as Fields, as described in the previous section, is used to create mail merge fields in a Writer document. If you now select a different record in the data source browser and then choose *Data as Fields*, the previously inserted data are replaced by the new data.

Input data as fields	Input data as text
Lisa Gerd has Reader-ID 2	Bert Lederstrumpf has Redaer-ID 0

Here another record has been selected. While the *Insert data as Fields* option leads to the previous values being changed to the values for the new record, in the case of *Insert data as Text*, the existing text remains unchanged.

Mail merge

The *Mail Merge* button launches the Mail Merge Wizard. A form letter assembles its data from different tables, so you need first to launch the database. In the database, you then create a new query to make the required data available.



To launch the database, right-click on the database itself or on one of its tables or queries; this immediately refreshes the display in the data source browser. After that the Mail Merge Wizard can be called up by using the corresponding button.

Data source of current document

Click on the *Data Source of Current Document* button to open a direct view of the table which forms the basis for the data inserted into the document. In the above example, the *Person* table from the *Addresses* database *appears*.

Explorer on/off

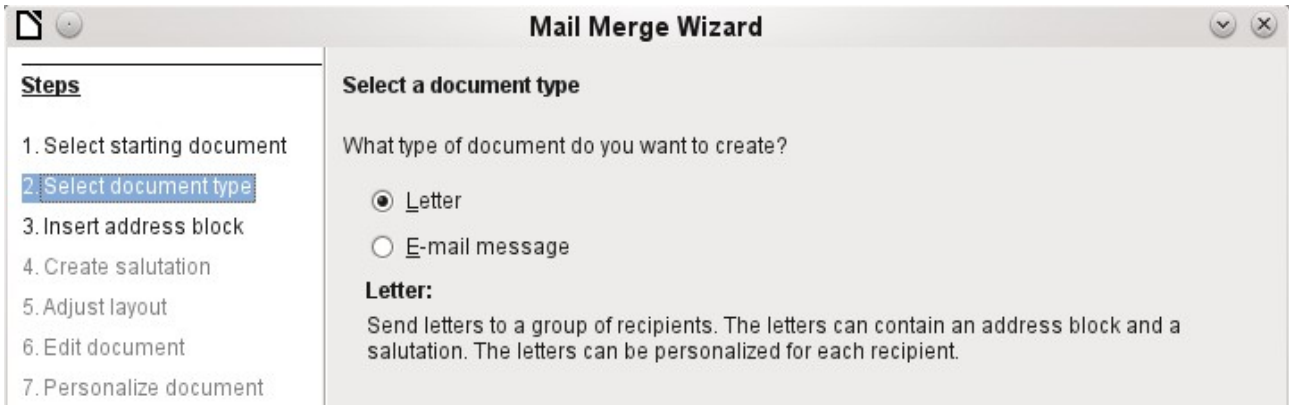
Toggling the *Explorer On/Off* button shows or hides the directory tree to the left of the table view. This allows more space, if necessary, for a display of the data. To access another table, you need to switch the Explorer back on.

Creating mail merge documents

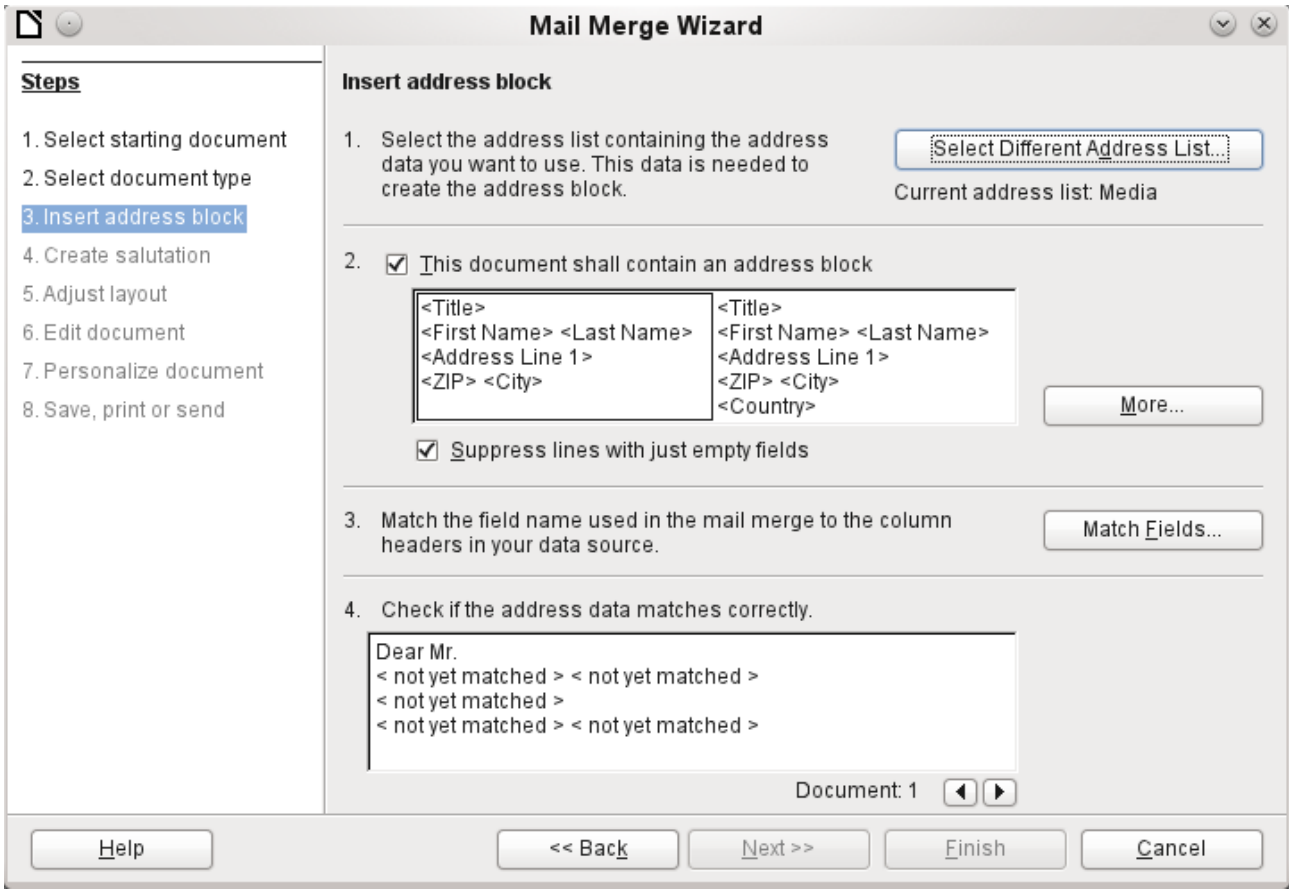
The Mail Merge Wizard is also accessible from the database browser. This Wizard allows the address field and the salutation to be constructed from a data source in small steps. In principle you can create these fields without using the Wizard. Here we will work through the steps of the Wizard as an example.

The **Starting document** for the form letter is the document to which the **database fields are linked**.

The **Merged document** is the one containing the data for the various people who are to receive the form letters. In the merged document there is **no linkage** to the data source. It is similar to the output of *Insert Data as Text*.



The Mail Merge Wizard can produce either letters or emails using records from the database.



The entry of the address block allows the most extensive configuration. The suggested address list comes from the currently selected query or table in the currently selected database.

Step 2 determines the overall look of the address block, which can be customized further using the *More* button. See the following figure.

Step 3 serves to link the named fields in the address block to the correct fields in the database. The Wizard initially recognizes only those database fields which have exactly the same names as those the Wizard uses.

In Step 4, the addresses are displayed. You can choose which addresses to take from the database by using the arrow keys. In the displayed addresses two elements require further editing:

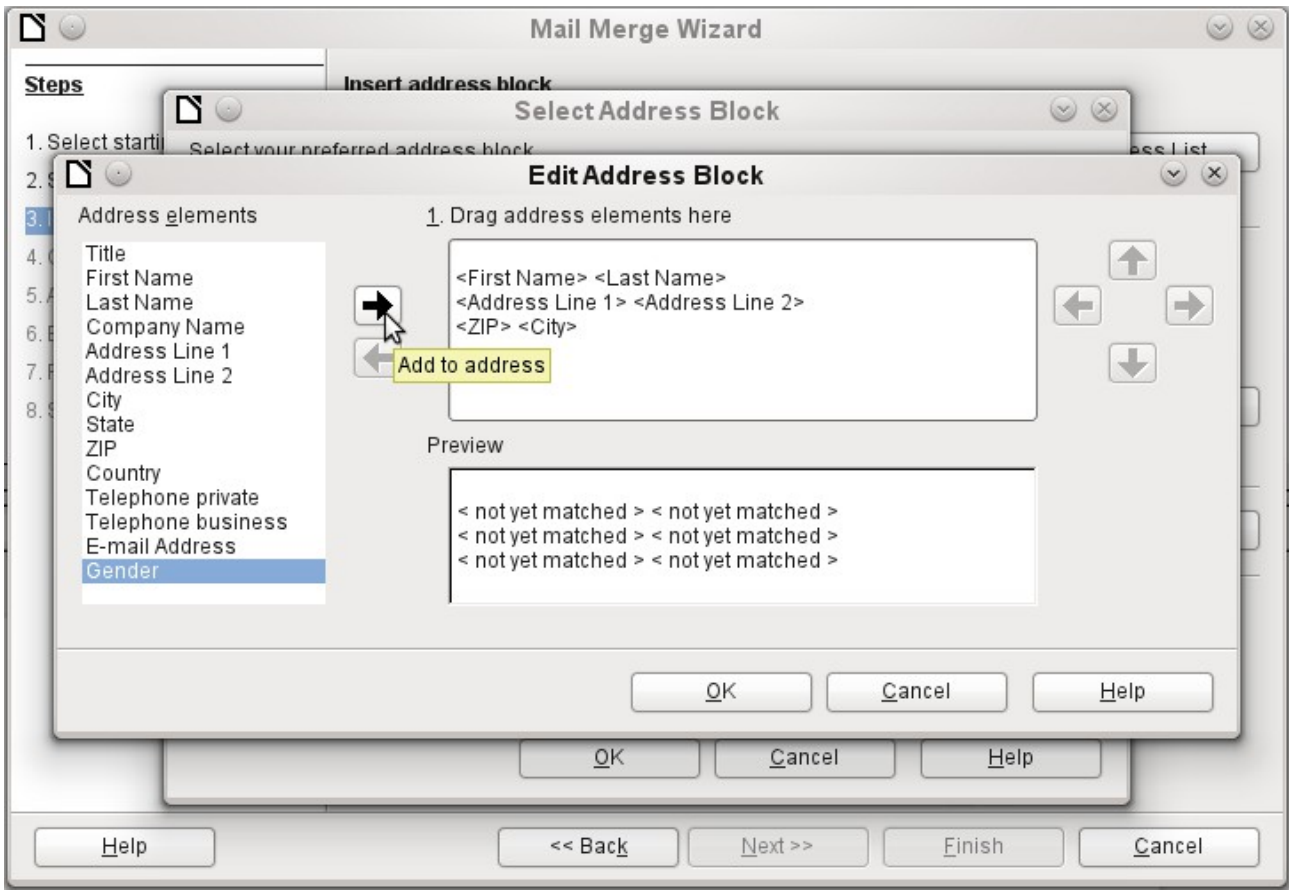
- There is no salutation.
- Apart from the first name, all the other fields are < *not yet allocated* > , because the field names in the database are different from the names that the Wizard initially uses.

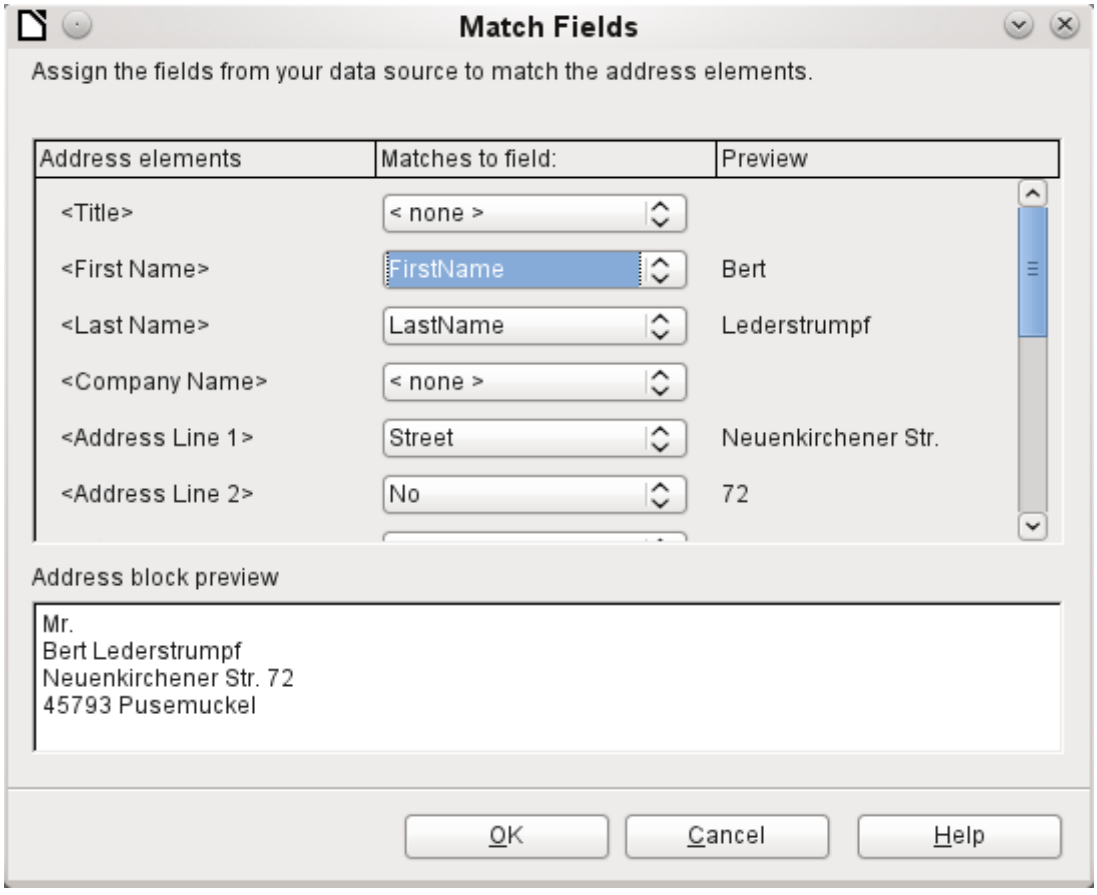
To correct these errors, the address block from Step 2 must be made editable.

You can see in the background that, when you choose to edit, you are first presented with an enlarged list of address blocks. Here you can select the most suitable address block to start with, and then edit it.

The address block cannot be edited directly. Instead, the arrangement of the fields is carried out by using the arrow buttons visible to the right to move fields into or out of the address block

For the salutation the *Salutation* field is inserted. All the other fields except FirstName must now be entered appropriately.

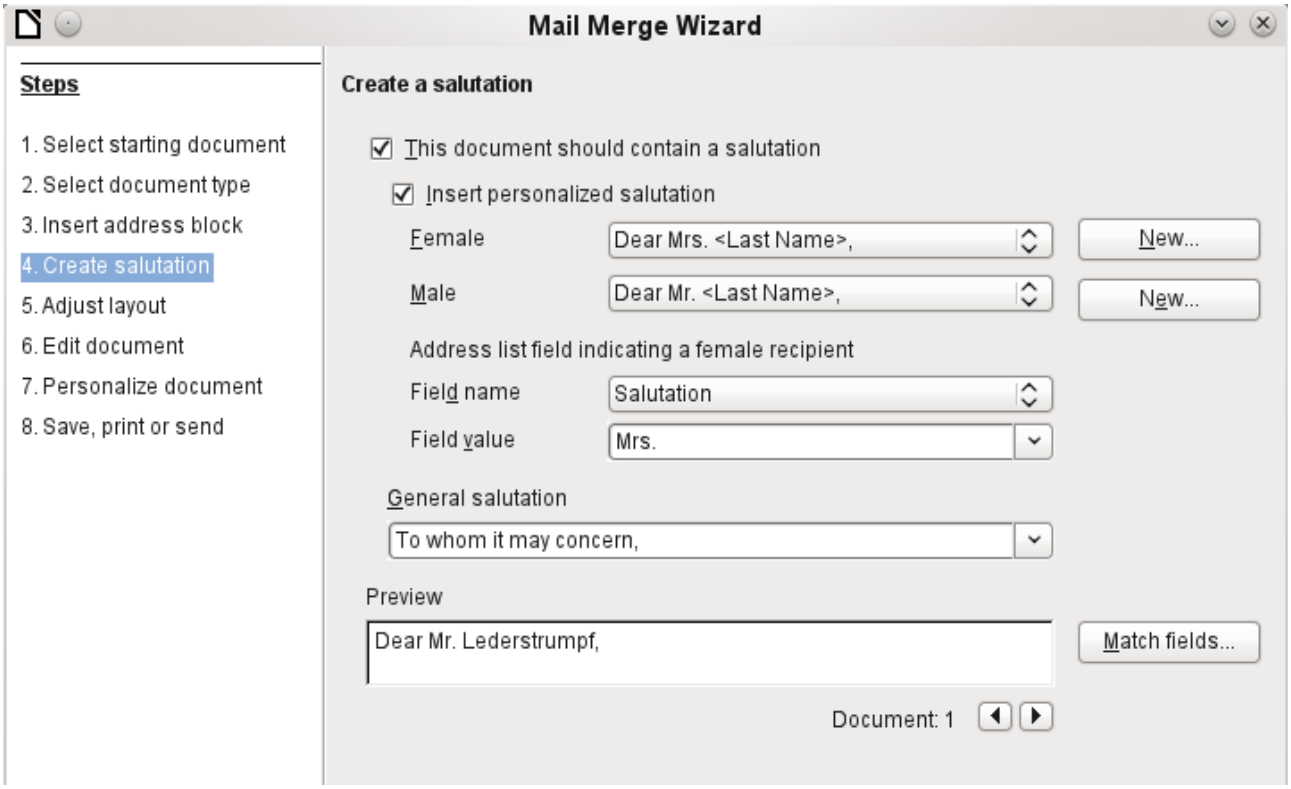

Mail Merge Wizard
Steps
Insert address block
Select Address Block
Edit Address Block
Address elements
Title
First Name
Last Name
Company Name
Address Line 1
Address Line 2
City
State
ZIP
Country
Telephone private
Telephone business
E-mail Address
Gender
1. Drag address elements here
<First Name> <Last Name>
<Address Line 1> <Address Line 2>
<ZIP> <City>
Add to address
Preview
< not yet matched > < not yet matched >
< not yet matched > < not yet matched >
< not yet matched > < not yet matched >
OK
Cancel
Help
<< Back
Next >>
Finish

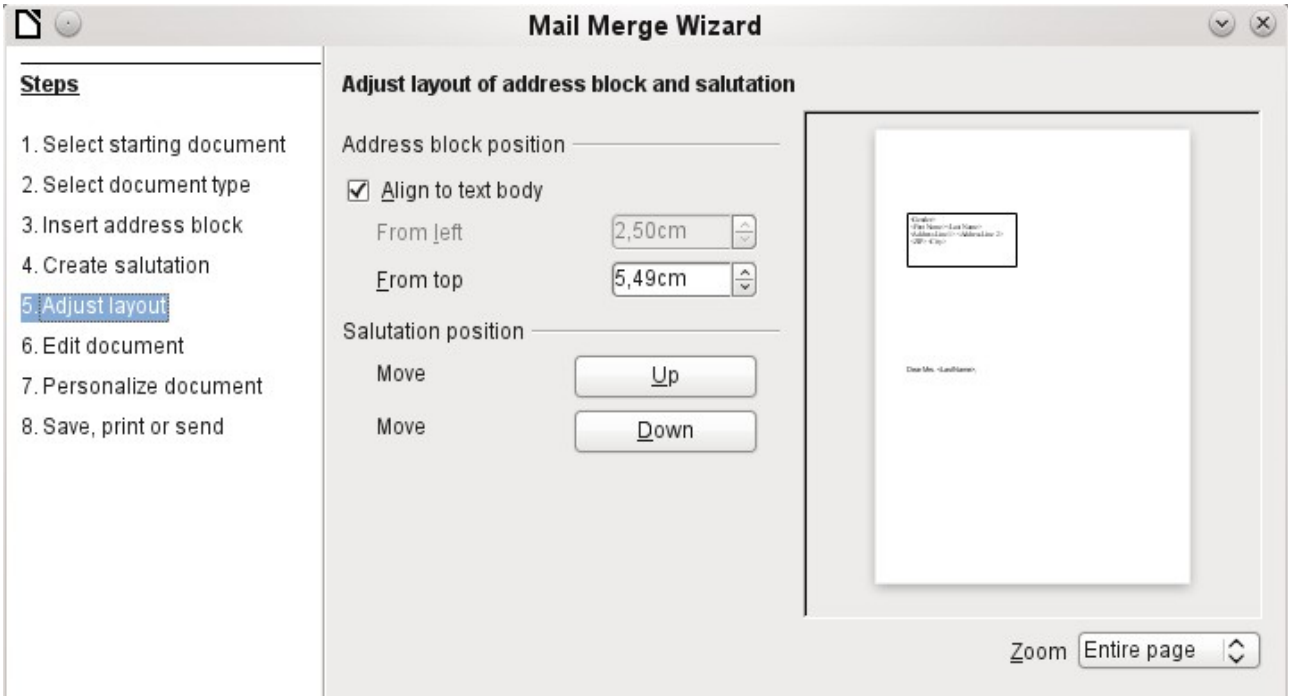

Match Fields
Assign the fields from your data source to match the address elements.
Address elements
Matches to field:
Preview
<Title>
< none >
<First Name>
FirstName
Bert
<Last Name>
LastName
Lederstrumpf
<Company Name>
< none >
<Address Line 1>
Street
Neuenkirchener Str.
<Address Line 2>
No
72
Address block preview
Mr.
Bert Lederstrumpf
Neuenkirchener Str. 72
45793 Pusemuckel
OK
Cancel
Help

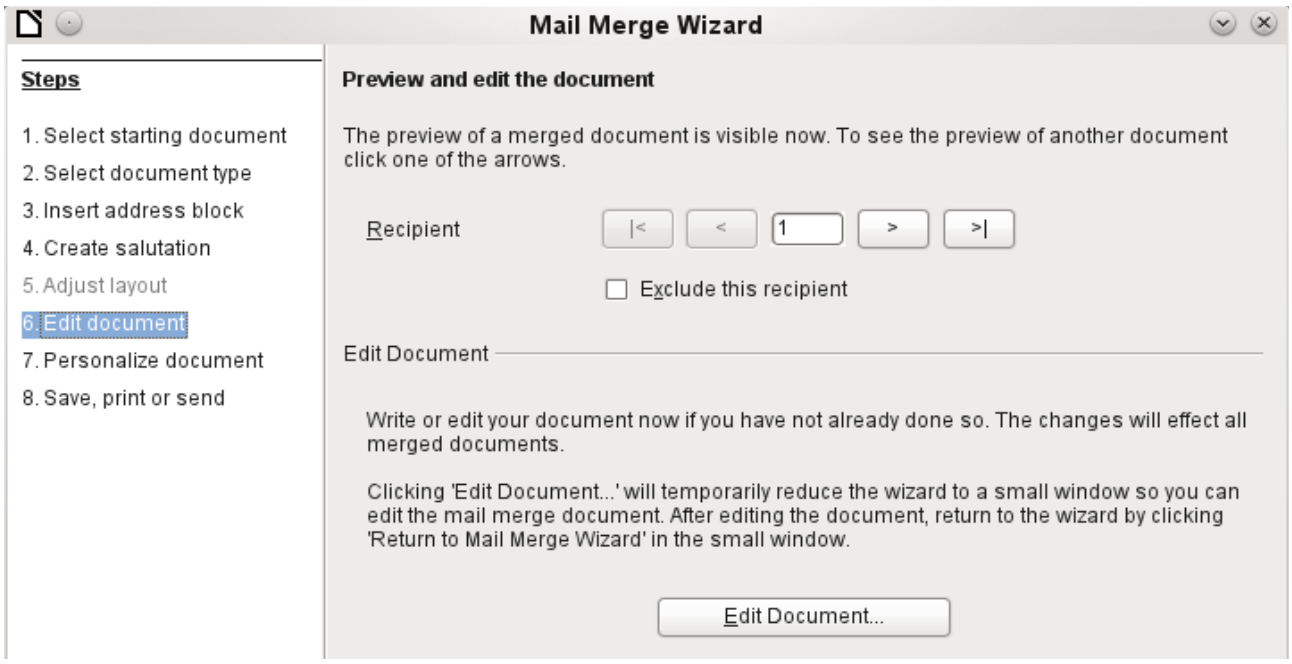
Here the address elements are associated with the corresponding elements from the query of the database successfully transferred by the Mail Merge Wizard. Again the first record in the query is used for the preview.

The database settings are essentially ended with Step 4. Here, it is just a matter of choosing which field the gender of the recipient should be taken from. This field has already been named, so that only the field content for a female recipient still needs to be specified.

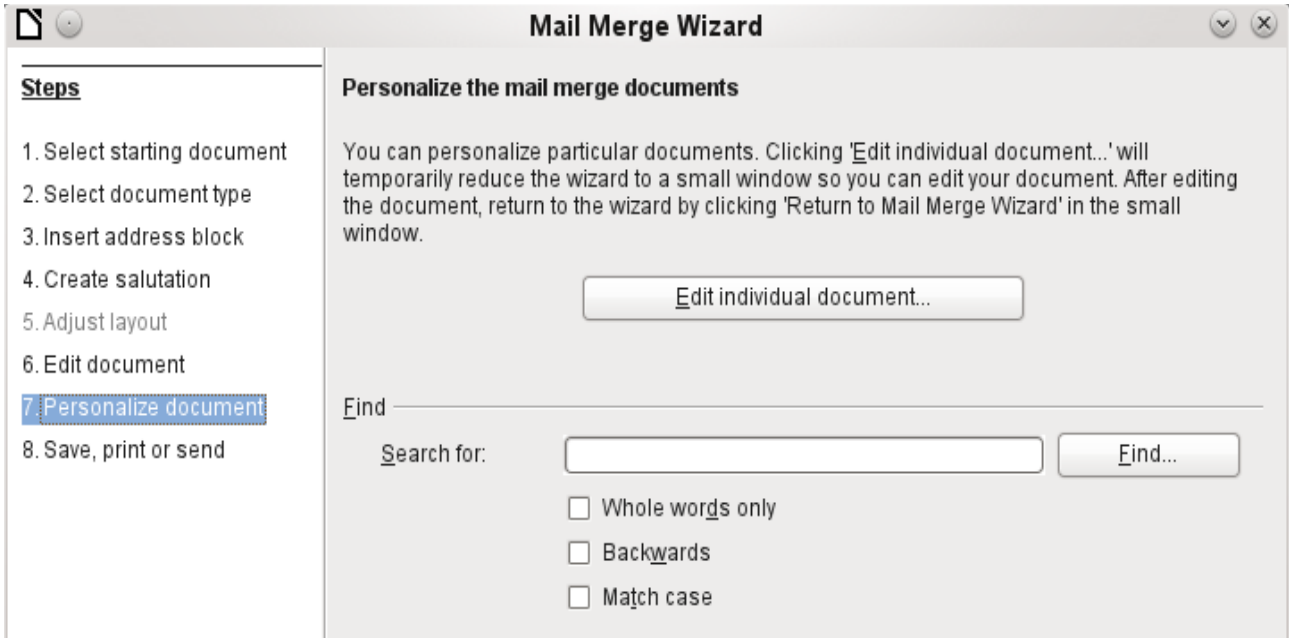
Three different salutations are to be produced. All records with 'f' start with *Dear Ms…*, all those with 'm' with *Dear Mr…* If there is no gender given, *Dear Sir/Madam* is selected.







Normally the document is initially a rough sketch, which can be further edited in Writer in Step 6.



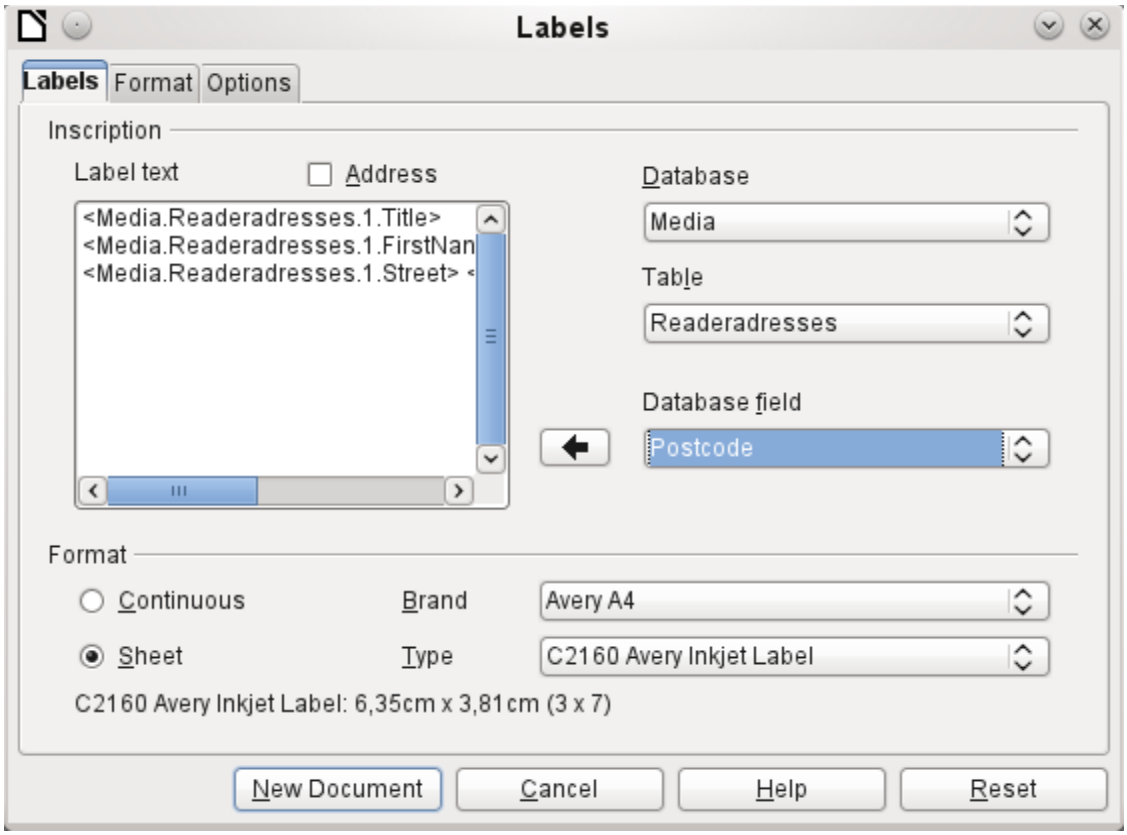
Up to now all the documents have been identical except for the different content of the fields read from the database. This can be changed in Step 7.

The **Starting document** is the document in which the **field properties** and the **linkage to the database** are stored. In the background meanwhile, you can see the original document with the contents of the first record that is to be converted into the form letter. This is called the Mail Merge document.

Only when one of the options is actually carried out (in the above example to save the starting document) does the Mail Merge Wizard terminate.

Label printing

Files > New > Labels launches the Label Printing Wizard. It opens a dialog, which includes all questions of formating and content for labels, before the labels themselves are produced. The settings in this dialog are saved in the personal settings for the user.



The basic settings for the content are in the *Labels* tab. If for Label text you check the Address box, all the labels will have the same content, taken from the LibreOffice settings for the user of the program.

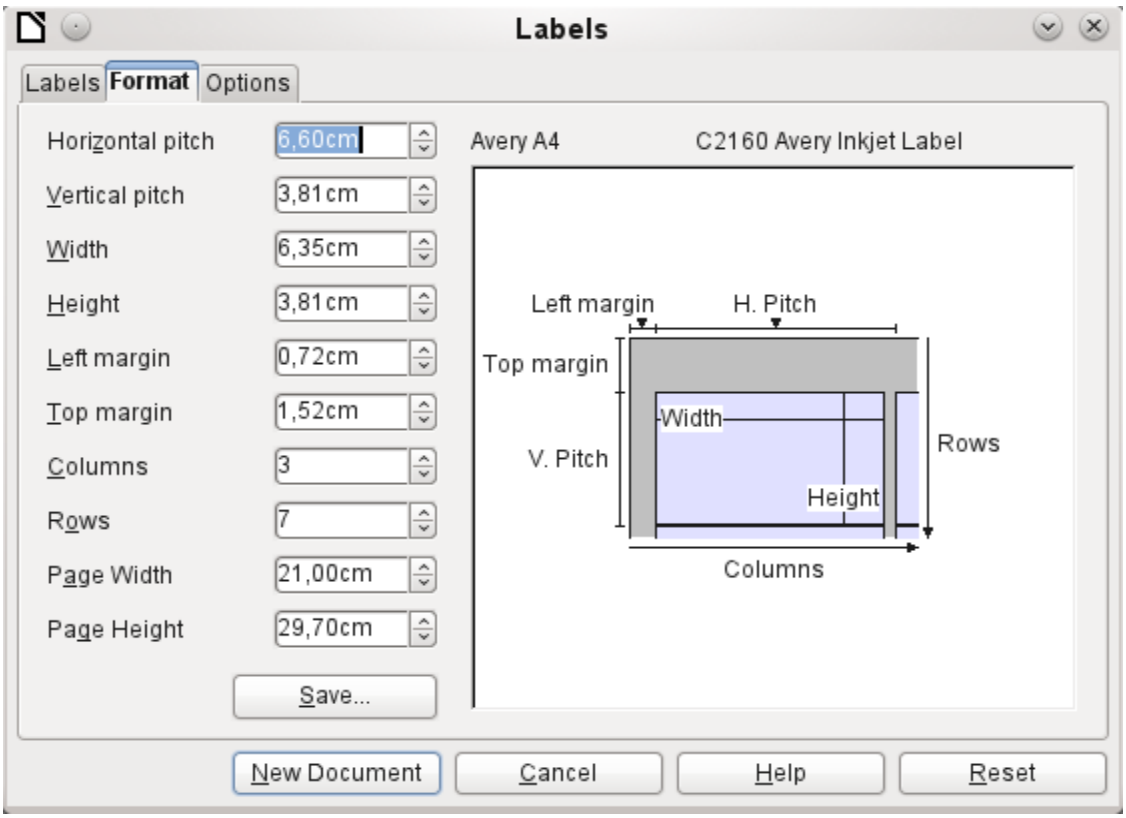
As an example, we will again use the *Addresses* database. Although the next selection field is headed *Tables*, **Tables and Queries** are both listed here, just as in the data source browser.

Use the arrow buttons to insert individual database fields into the editor. The name for the database field *Surname* is set here to <*Addresses.MailMergeQuery.1.Surname*>. The sequence is thus <*database.Table.1.database field*>.

You can work with the keyboard in the editor. So for example, you can insert a line break at the beginning, so that the labels will not be printed directly on the top edge but the content can be printed as completely and clearly visible.

The format can be selected in the *Labels* tab. Here many label brands are incorporated so that most other settings in the *Format* tab are not necessary.

Note	In versions 3.4.x to 3.5.2, due to a change in the basic settings in the label wizard, display errors occurred when the labels were the same width as the page width. Under these conditions, the last label simply slides down one line. In version 3.3.x the default page width for DIN-A4-label sheets was slightly increased from 21.00 cm to 21.04 cm. A subsequent widening of the page width can also solve this problem for current label sheets. In version 3.5.3 page settings were added in the Format tab.



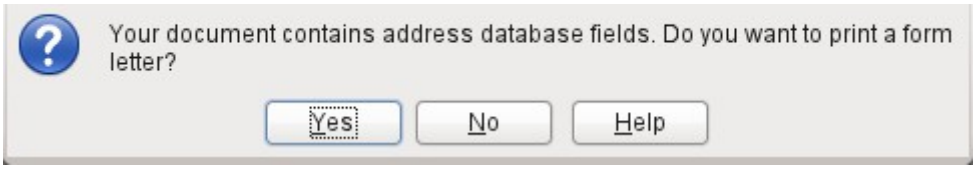
Use the *Format* tab to set the label size accurately. The settings are only significant when the make and type of the labels is not known. Note that, to print labels 7.00 cm wide, you need a page width a little bigger than 3*7.00 cm = 21.00 cm. Only then will three labels be printed in a row on the page.

On the *Options* tab you can specify whether only a single label or a whole page of labels will be produced. The page will then be filled with data from successive records of the database, beginning with the first record. If there are more records than will fit on the page, the next page will automatically be filled with the next set of records.

The *Synchronize contents* checkbox links all the labels together so that subsequent changes in layout of any label will be applied to all the others. To transfer the edited content, use the *Synchronize* button, which appears during label production if you have selected this checkbox.

Use the **New Document** button to create a document containing the selected fields.

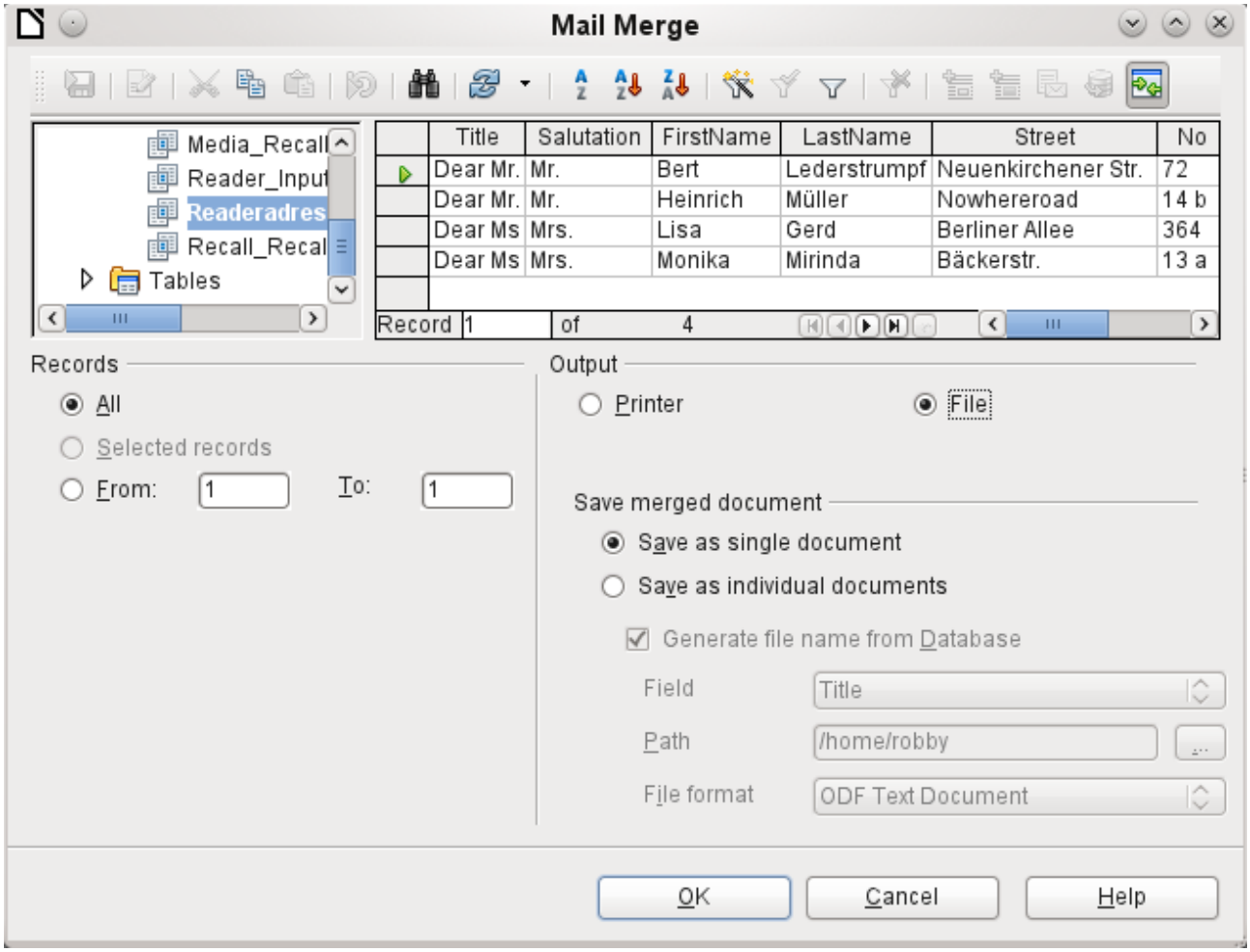
When you initiate the printing process, the following question appears:



Choose **Yes** to fill the address database fields with the corresponding content.

The source of the data for the label printing is not found automatically; only the database is pre-selected. The actual query must be specified by the user, because in this case we are not dealing with a table.

When the query is selected and the corresponding records chosen (in this case *All*), the printing can begin. It is advisable, especially for the first tests, to choose *Output* to a *File*, which will save the labels as a document. The option to save in several documents is not appropriate for label printing but rather for letters to different recipients which can then be worked on subsequently.

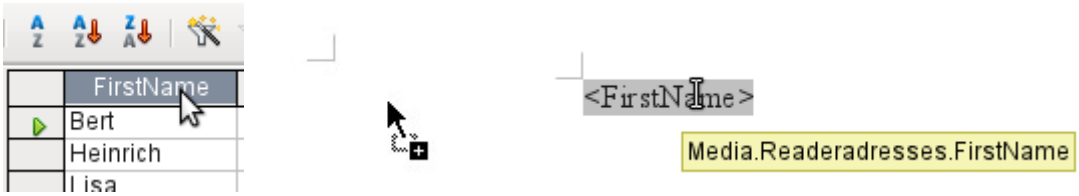


Direct creation of mail merge and label documents

Instead of using the Wizard, you can of course produce mail merge and label documents directly.

Mail merge using the mouse

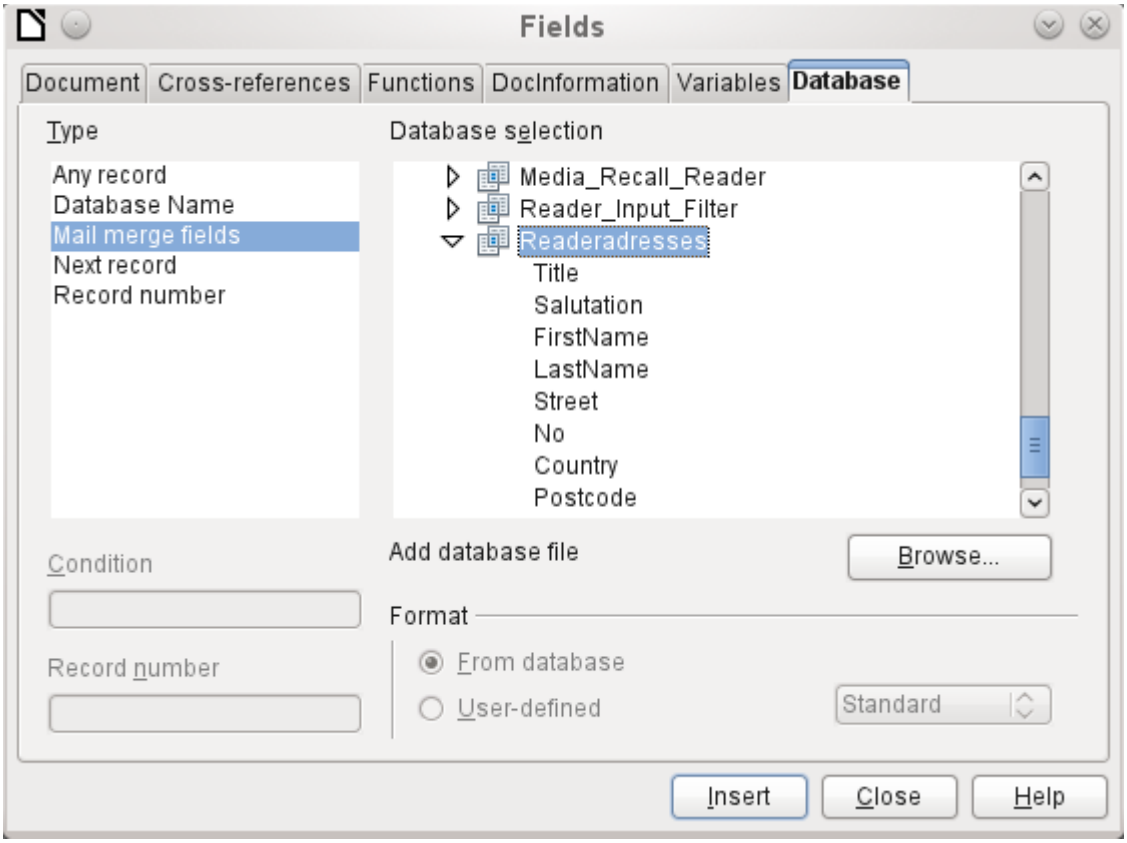
Mail merge fields can be taken from the database browser using the mouse.



Select the table header with the left mouse button. Hold the button down and drag the cursor through the text document. The cursor changes its shape to an insert symbol. The MailMerge field is inserted into the text document, here shown in the complete description which is made visible using **View > Field names**.

Creating form letters by selecting fields

Mail merge fields can be inserted using **Insert > Fields > Other > Database**.



Here all tables and queries in the selected database are available. Using the '*Insert*' button, the various fields can be inserted one after another directly into the text at the current cursor position.

If you want to create a salutation, which is usual in form letters, you can use a hidden paragraph or hidden text: **Insert > Fields > Other > Functions > Hidden paragraph**. For both variants take care that the condition you formulate will **not** be fulfilled, since you want the paragraph to be visible.

For the formula *Dear Mrs <Surname>,* to appear only when the person is female, a sufficient condition is:

```
[Addresses.Mailmergequery.Gender] != "f"
```

Now the only remaining problem is that there may be no surname. Under these circumstances, "*Dear Sir/Madam,*" should appear so this is the condition you must insert. The overall condition is:

```
[Addresses.MailMergeQuery.Gender] != "f" OR NOT
[Addresses.MailMergeQuery.Surname]
```

That excludes the possibility of this paragraph appearing when the person is not female or there is no entered surname.

In the same way you can create entries for the masculine gender and for missing entries for the two remaining types of salutation.

Naturally you can create a salutation in the address field in exactly the same way, wherever the gender is specified.

Further information is given in the LibreOffice Help under *Hidden Text* and *Conditional Text*.

Of course it would be still simpler if someone who understands databases were to put the whole salutation right into the query. This can be done using a correlated subquery (see the chapter on Queries in this Handbook).

Particularly interesting for labels is the field type Next record. If this field type is chosen at the end of a label, the next label will be filled with data from the following record. Typical labels for sequential label printing look like the following figure when you use **View > Field names** to make the corresponding field designations visible:

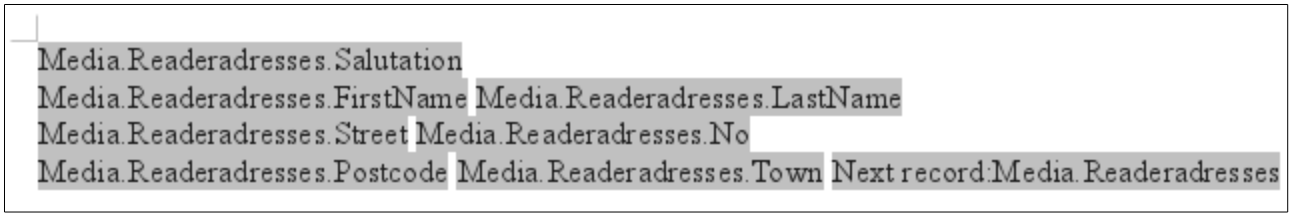


Figure 6: Field selection for labels with sequential content

For the last label on the page, you must allow for the fact that the next record is automatically called up following a page break. Here the field type “Next record” should not occur. Otherwise a record will be missed out because a double record jump occurs.

Tip	Creating mail merge letters is also possible directly from a database form. The only requirement is that the database be registered in LibreOffice. When a mail merge is carried out, be sure to choose **View > Print Layout.** This ensures that the elements are correctly positioned on the page. If a form is then printed, the usual mail merge query appears. This type of mail merge has the advantage that you do not need any files other than the *.odb file in order to print.

External forms

If simple form properties available in LibreOffice are to be used in other program modules such as Writer and Calc, you only need to display the form design toolbar, using **View > Toolbars > Form design**, then open the *Form navigator.* You can build a form or, as described in the Forms chapter, create a form field. The *Data* tab of the Form Properties dialog looks a little different from the one you see when forms are built directly in an ODB database file.

Figure 7: Form with an external data source

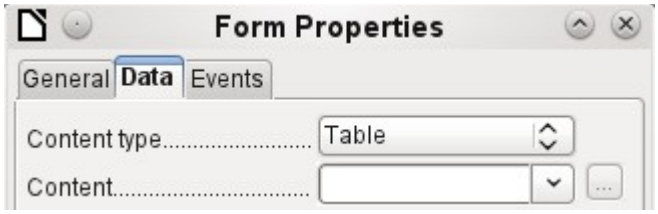


Figure 8: Form with an internal data source.

The *Data source* must be selected separately when using an external form. Use the button to the right of the data source listbox to open the file browser. Any ODB file can be selected. In addition, the field for the data source contains a link, beginning with **`file:///`**.

If instead you look in the listbox contents, you will see databases already registered in LibreOffice under their registered names.

The forms are created in exactly the same way as in Base itself.

The forms produced in this way are by default shown in edit mode each time the file is opened, not write-protected as in Base. To prevent accidental modification of the form, you can use **File > Properties > Security** to open the file read-only. You can even protect the file from alteration using a password. In office systems, it is also possible to declare the whole file as write-protected. This still allows entry into the fields of the form, but not movement of fields or the entering of text between them.

Tip	Forms can also be rapidly created by using drag and drop. To do this, open the database, find the relevant table or query and select the table headers. In Writer, select apropriate field headings with the left mouse button, hold down the *Shift*- and *Ctrl*- keys, and the mouse cursor will turn into a link symbol. Then drag the headings into the Writer document. You can drag the fields into Calc files without the use of the additional keys. The copy symbol appears as a mouse cursor. In both cases an entry field is created with the associated label. The link to the data source is created with the first actual entry of data, So data entry into such a form can commence immediately after the drag-and-drop operation.

Advantages of external forms

Base need not be opened first in order to work with the database. Therefore you do not need an extra open window in the background.

In a database that is already complete, existing database users can subsequently be sent the improved form without problems. They can continue to use the database during the development of further forms, and do not need to copy complicated external forms from one database into another.

Forms for a database can be varied to suit the user. Users who do not have the authority to correct data or make new entries can be sent a current data set by other users, and simply replace their *.odb file to have an up-to-date view. This could, for example, be useful for a database for an association where all committee members get the database but only one person can edit the data; the others can still view the addresses for their respective departments.

Disadvantages of external forms

Users must always install forms and Base with the same directory structure. That is the only way that access to the database can be free from errors. As the links are stored relative to the form, it is sufficient to store the database and its forms in a common directory.

Only forms can be created externally, not queries or reports. A simple glance at a query therefore must go through a form. A report on the other hand requires the opening of the database. Alternatively it might be possible to create it, at least partially, using mail merge.

Database use in Calc

Data can be used in Calc for calculation purposes. For this purpose it is first necessary to make the data accessible in a Calc worksheet.

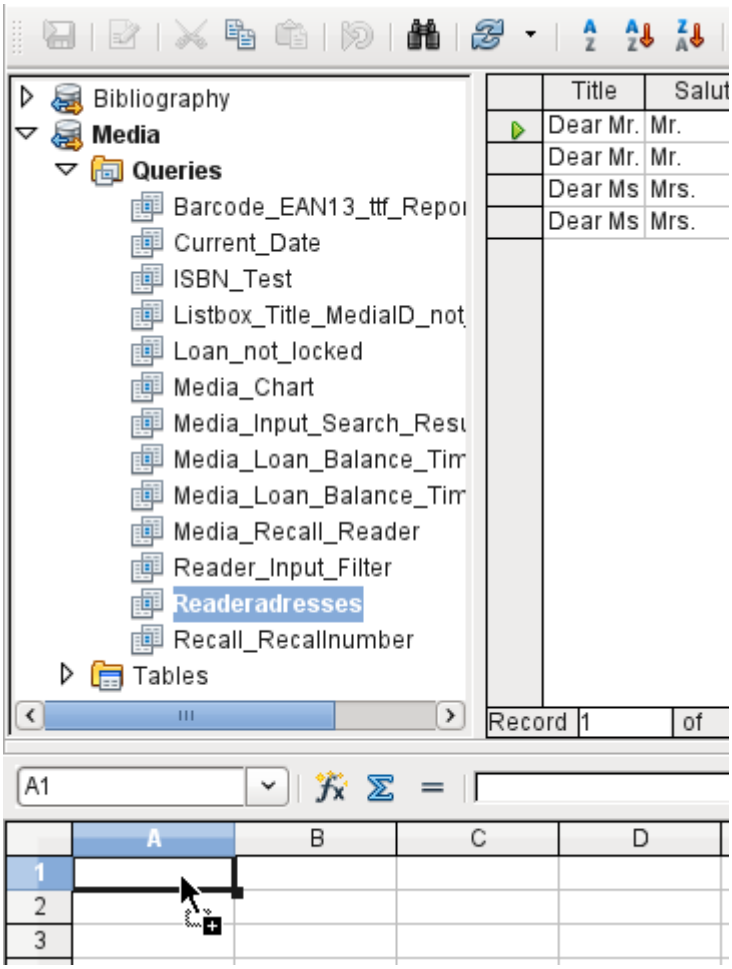
Entering data into Calc

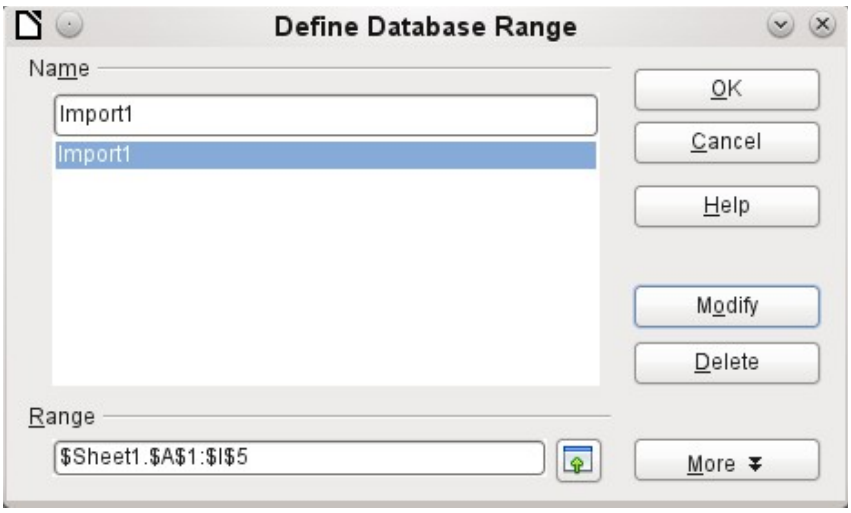
There are various ways of entering data into Calc.

Select a table with the left mouse button and drag it into a Calc worksheet. The cursor sets the left upper corner of the table. The table is created complete with field names. The data source browser in this case does not offer the options of *Data into Text* or *Data into Fields*.

Data dragged into Calc in this way shows the following properties:
Not only the data are imported, but the field properties too are read and acted on during the import. Fields such as house numbers, which were declared as text fields, are formatted as text after insertion into Calc.

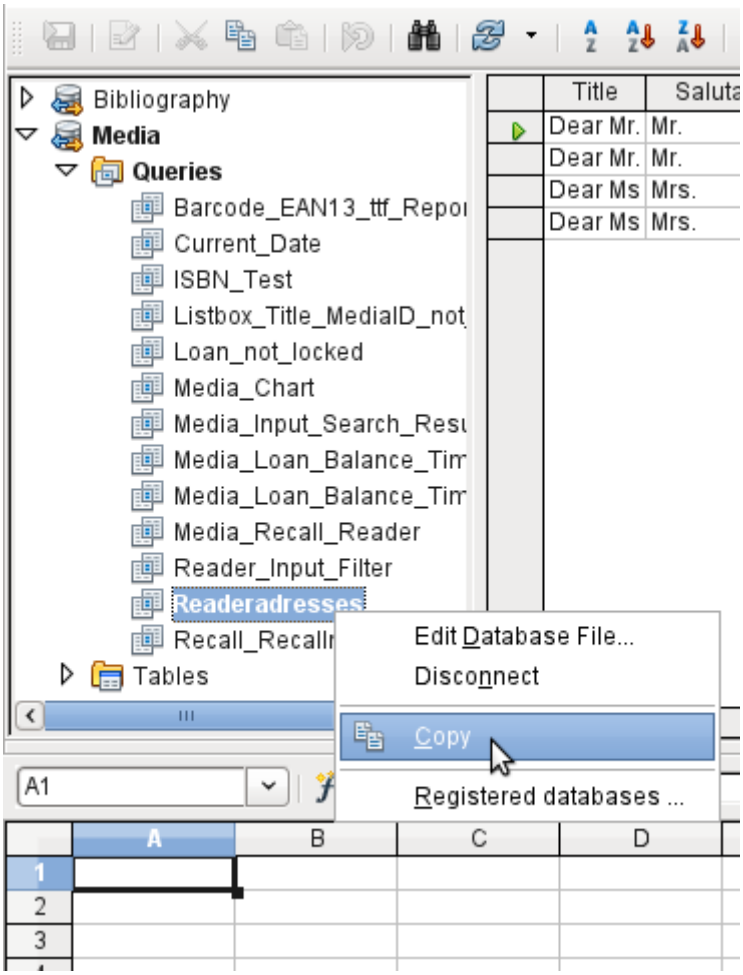
The import becomes a Calc range, which by default is assigned the name Import1. The data can later be accessed using this range. **Data > Refresh range** allows the range, where appropriate, to be supplied with new data from the database.

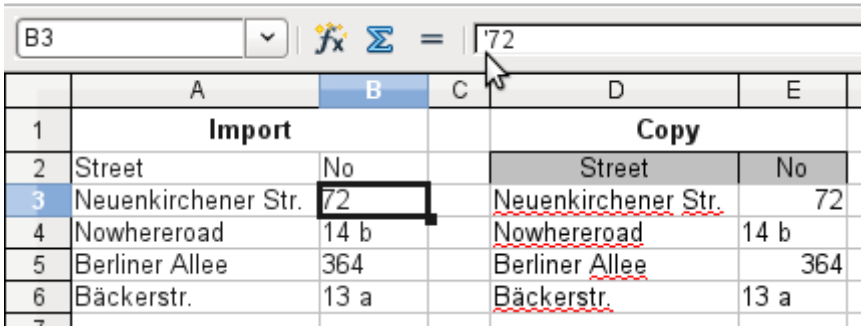




The imported data is not formatted except as the properties of the database fields require.

You can also use the context menu of a table to make a copy of the data. In this case, however, there is no import but merely a copy. The properties of the data fields are not read with them but are determined by Calc. In addition the field names are formatted as table headers.





You see the difference especially in database fields that are formatted as text. On import, Calc turns these into text fields and precedes numbers, which would otherwise be interpreted as such, with a single quote **('72)**. These numbers can then no longer be used in calculations.

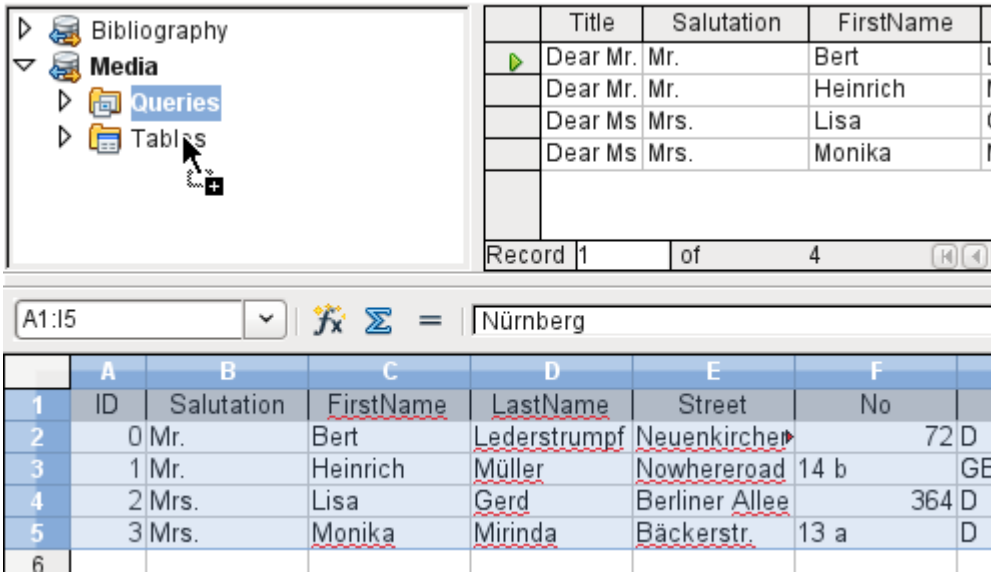
If you export them again, the single quote is removed, so that the data remain as they were.

Tip	Importing data into Calc overwrites the previous contents and also any previous formatting. If data is to be exported consistently into the same table, you should use a separate sheet for data import. The data are then read into the other sheet by using the term tablename.fieldname. The fields in this sheet can be formatted suitably without risk of the formatting being overwritten.

Tip	Records can also be copied directly from the database using the clipboard, or by drag and drop using the mouse. If a table or query is dragged into a Calc spreadsheet, the whole of the content is inserted. If the table or query is opened and one or more records selected, only these records together with the field names are copied when you drag them over.

Exporting data from Calc into a database

Select the data in the Calc worksheet. Hold the left mouse button down and drag the data that you want to turn into a database into the table area of the database browser.



The cursor changes its appearance, showing that something can be inserted.

The first window of the Import Wizard opens. The further steps with the wizard are described in Chapter 3, Tables, in the section “Importing records from other data sources”.

Converting data from one database to another

In the explorer of the data source browser, tables can be copied from one database to another by selecting the source table with the left mouse button, holding the button down, and dragging it into the target database in the table container. This causes the dialog for copying tables to be displayed.

In this way, for example, read-only databases (data sources such as address books from an email program or a spreadsheet table) can be used as a basis for a database in which the data become editable. Also data can be directly copied when changing to another database program (for example changing from PostgreSQL to MySQL).

If you wish the new database to have different relationships from the original one, you can arrange this by using appropriate queries. Those who are not sufficiently expert can instead use Calc. Just drag the data into a spreadsheet and prepare them for import into the target database using the facilities that Calc provides.

For the cleanest possible import into a new database, the tables should be prepared in advance. This allows problems of formatting and those involving the creation of primary keys to be recognized well in advance.

Importing records into a table using the clipboard.

If records are available in tabular form, they can be inserted into Base using the clipboard and the wizard.

In Base, a right-click on the destination table begins the import. In the context menu under Copy are the commands “Import” and “Import content...” If you choose Paste, the Import Wizard will have already selected the table and “Append data”. “Paste special...” gives only a query for an import filter. The available options are HTML and RTF.

If instead you right-click in the table container, the Import Wizard gives you only the choice of creating a new table.

Importing PDF records

If you want to import data from various external sources, it is best to choose a format that prevents your form from being modified during data entry. Using Writer, you can create forms in PDF format, put them online, and have the completed forms returned to you, for example as email attachments. All that is lacking is the simplest possible entry of the data into Base. The example[1] illustrates such a means of import.

Creating a PDF form

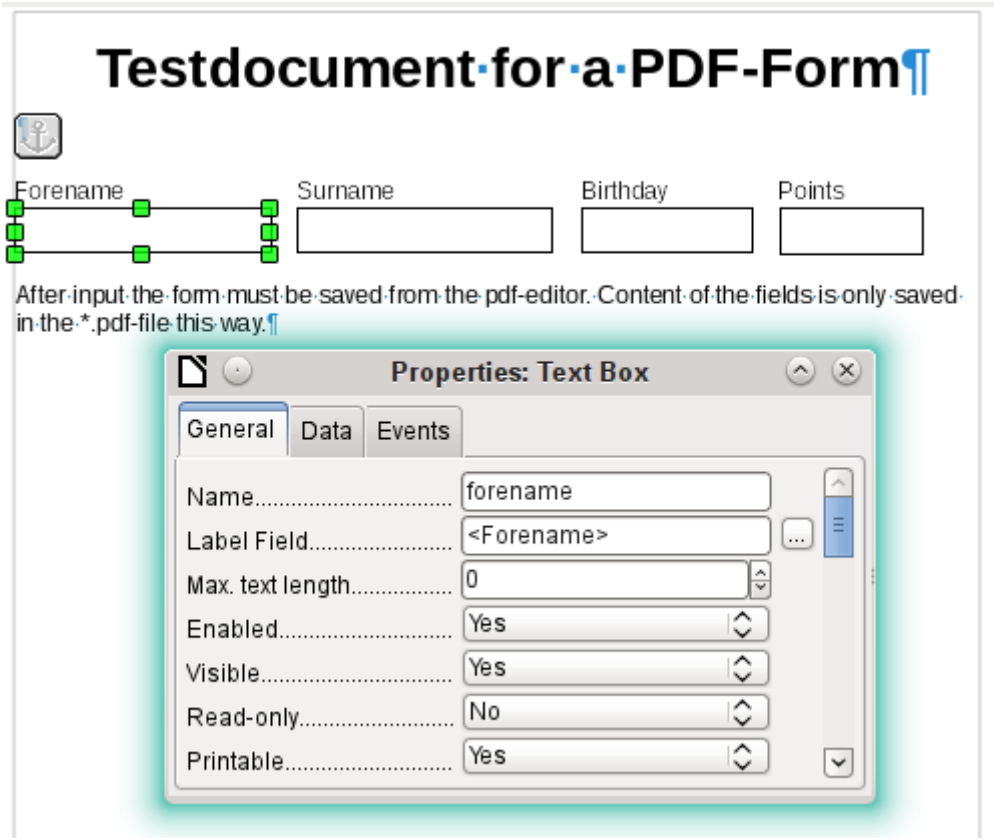
A PDF form is created as an external form with no link to the database. Using **View > Toolbars > Form Controls**, the necessary elements for the form are displayed and can be inserted as required.

Unfortunately PDF format makes no distinction between numeric fields, date fields, and text fields. For the example provided here, it is sufficient to use text fields for all the entries. Other field formats within the Writer form will inevitably be lost during PDF export.

1 The database `Example_PDFFormular_Import.odb` for this report is included in the example database for this handbook.

Basically PDF forms can have the following fields:

- Buttons
- Text fields
- Checkboxes
- Comboboxes and
- Listboxes



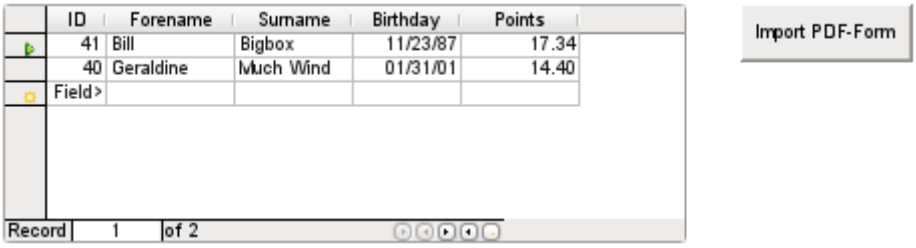
The test form contains a total of 4 text fields. In **Properties: Text box > General > Name,** you should always choose the field name used in the database table when using the following import method in order to avoid problems with fieldnames and field content.

Help messages are shown when the records are read, but do not appear in every PDF viewer.

To ensure that the form actually contains the records, it should be saved in the PDF viewer after data input, using the usual **File > Save as** menu option. The actual command for doing this might vary between viewers. Without this procedure, the viewer will show the records after the form has been opened on your own computer, but it actually reads them from the viewer's temporary storage file and not directly from the PDF file. If the form is then transferred to another computer, it will be found to be empty.

Reading the records from the PDF form

The form for the Base database is very simple in appearance. It is linked to the table and shows the records that have just been read in. The most recent entries are shown in the table control above.



The macro for reading the records in is entered under **Properties : Button > Events > Execute Action**.

To read the records out, we use the open source program **pdftk**. The program is freely available for both Windows and Linux. Linux distributions mostly have it as a package in their repositories. Windows users will find it at https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/

The records read out using **pdftk** are written into a text file, which looks like this:

```
1  ---
2  FieldType: Text
3  FieldName: forename
4  FieldFlags: 0
5  FieldValue: Bill
6  FieldJustification: Left
7  ---
8  FieldType: Text
9  FieldName: surname
10 FieldFlags: 0
11 FieldValue: Bigbox
12 FieldJustification: Left
13 ---
14 FieldType: Text
15 FieldName: birthday
16 FieldNameAlt: Date, year minimum 2 places
17 FieldFlags: 0
18 FieldValue: 11/23/87
19 FieldJustification: Left
20 ---
21 FieldType: Text
22 FieldName: points
23 FieldNameAlt: Decimal value, 2 decimal places
24 FieldFlags: 0
25 FieldValue: 17.34
26 FieldJustification: Left
27
```

Each field is represented by five lines in the file. For the macro, the important lines are “FieldName” (should be the same as FieldName in the destination table), “FieldValue” (content of the field after saving the PDF file), and “FieldJustification” (last line of the entry).

The whole import process is controlled by macros. The PDF form needs to be on the same path as the database. The records are read out of it into the text file, and then read into the database from this text file. This continues for all the PDF files in the folder. Old records should therefore be removed from the folder as far as possible, because the function does not check for duplication.

```
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
```

After the variables have been declared, the number of fields in the PDF form is given. The count begins at 0, so a value of 3 actually means a total of four fields. Using this count, it can be determined if all the data for a record has been read, so that it is ready to be transferred into the table.

```
inFields = 3
stTable = "Name"
oDatasource = ThisComponent.Parent.CurrentController
If NOT (oDatasource.isConnected()) THEN
   oDatasource.connect()
END IF
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
```

The database connection is made. The path to the database file in the file system is read. Using this path, the contents of the folder are read into the array **`aFiles`**. A loop checks each filename in the array to see if it ends in `.pdf`. Upper and lower case are not distinguished, as the results of the search are all converted into lower case using **`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))
```

To determine the command for reading out the data, it is necessary to understand the operating system's file address conventions. Therefore the original URL beginning with **`file://`** must be adapted to the current system. The command for starting the **pdftk** program depends on the operating system. It might carry the suffix .exe or perhaps a complete path to the program like C:\`Program Files (x86)\pdftk\pdftk.exe` or the suffix might not be required at all. **`GetGuiType`** is used to determine the type of system in use: 1 stands for Windows, 3 for MacOS and 4 for Linux. The following steps only distinguish between Windows and the rest.

After this the **`Shell()`** function is used to pass the appropriate launch command for **`pdftk`** to the console. The argument **`True`** ensures that LibreOffice will wait until the shell process terminates.

```
      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
```

The **FreeFile** function determines which is the next free data channel available in the operating system. This channel is read out as an integer number and used to connect directly to the PDF data file that has just been created. The INPUT instruction is used to read the file. This takes place outside LibreOffice. The external records are then read into LibreOffice.

```
OPEN stFile FOR INPUT AS inNumber
   DO WHILE NOT Eof(inNumber)
      LINE INPUT #inNumber, stRow
```

The PDF data file is now read line by line. Whenever the term “FieldName” occurs, the remaining content of the line is taken as the name of the field in the PDF form and also, because of the way the form was defined, the name of the database field into which the data should be written.

All the field names are combined directly for use in the later SQL commands. What this means in practice is that the field names are enclosed in double quotes and separated by commas.

In addition, for each field name a query determines the field type in the table. Date and decimal values must be transferred in a different way to text.

```
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
```

As for the field names, so for the field values. However these must not be double-quoted but must be prepared in accordance with the requirements of SQL code. This means that text must be single-quoted, dates converted to conform with SQL conventions, and so on. This is done by the extra external **SQL_Value** function.

```
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
```

If the term “FieldJustification” is found, this marks the end of the combined block of field name and properties. The counter **i,** which will subsequently be compared with the previously declared field counter **inFields**, is therefore incrememented by 1.

When **i** and **inFields** become equal, the SQL command can be put together. However you must ensure that empty records are not created from empty forms. Therefore there is a previous check for all field values being **NULL**. In such cases, the SQL command is launched immediately. Otherwise, the record is included for insertion into the “Name” table. After this, the variables are restored to their default values and the next PDF form can be read.

```
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
```

At the end of the procedure, one `PDF_Form_Data.txt` file remains. This is deleted. Then the form is reloaded so that the records that were read in can be displayed.

```
            Kill(stFile)
        END IF
    NEXT
    oEvent.Source.Model.Parent.reload()
END SUB
```

If the text contains a “”, this will be seen as an end-of-text marker during insertion by SQL. The SQL code for the insertion command fails if any more text follows without being enclosed in single quotes. To prevent this, each single quote within the text must be masked by another single quote. This is the job of the **`String_to_SQL`** function.

```
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
```

Dates in the PDF file are read as text. They cannot be checked in advance for correct entry.

When dates are written in English, the day, month and year are separated by dots or, more often, hyphens. The day and the month may have a single digit or two. The year may have two digits or four.

In SQL code, dates must begin with a four-digit year and be written YYYY-MM-DD. The entered dates therefore need to go through a conversion process.

The entered date is split up into day, month and year parts. The day and month are given a leading zero and then right-truncated to two digits. This ensures a two-digit figure in all cases.

If the year part already has four digits (greater than 1000), the value is not changed. Otherwise, if the year is greater than 30, the date is assumed to belong to the last century and needs to have 1900 added. All other dates are assigned to the current century.

```
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
```

The **SQL_Value** function combines this function with the **NULL** settings shown below, and so gives correctly formatted values for input into the database to its calling function.

Empty fields yield a NULL value. The corresponding field in the table will also be empty.

```
FUNCTION SQL_Value(st AS STRING, stType AS STRING) AS STRING
    DIM stValue AS STRING
    IF st = "" THEN
        SQL_Value = "NULL"
```

If this is a date field, and the content is to be recognizable as a date, its content must be converted into the SQL date format. If it is not recognizable as a date, the field should remain empty.

```
    ELSEIF stType = "DATE" THEN
        IF isDate(st) THEN
            SQL_Value = "'" & Date_to_SQLDate(st) & "'"
        ELSE
            SQL_Value = "NULL"
        END IF
```

A decimal field might contain commas instead of decimal points, with decimal places following them. In Basic and SQL, the decimal separator is always a dot. Therefore numbers containing a comma must be converted. The field must contain a number, so other characters such as units must be removed. This is carried out by the **Val()** function.

```
    ELSEIF stType = "DECIMAL" THEN
        stValue = Str(Val(Join(Split(st,","),".")))
```

All other content is treated as text. Single quotes are masked with a further single quote and the whole term is enclosed again in single quotes.

```
    ELSE
        SQL_Value = "'" & String_to_SQL(st) & "'"
    END IF
END FUNCTION
```

For further details on macro construction, see Chapter 9 of this handbook. This example simply shows that it is possible to transfer data from PDF forms into Base without having to copy the values field by field using the clipboard. The construction of the above procedure has deliberately been kept very general and would need to be adapted to particular situations.