Base Handbuch

Verwalten Sie Ihre Daten

LibreOffice ist ein eingetragenes Markenzeichen der The Document Foundation. Weitere Informationen finden Sie unter http://de.libreoffice.org
Copyright

Dieses Dokument unterliegt dem Copyright © 2015. Die Beiträger sind unten aufgeführt. Sie dürfen dieses Dokument unter den Bedingungen der GNU General Public License (http://www.gnu.org/licenses/gpl.html), Version 3 oder höher, oder der Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), Version 3.0 oder höher, verändern und/oder weitergeben.

Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.

Fast alle Hardware- und Softwarebezeichnungen und weitere Stichworte und sonstige Angaben, die in diesem Buch verwendet werden, sind als eingetragene Marken geschützt.

Da es nicht möglich ist, in allen Fällen zeitnah zu ermitteln, ob ein Markenschutz besteht, wird das Symbol (R) in diesem Buch nicht verwendet.

Mitwirkende/Autoren

Titelblatt
Klaus-Jürgen Weghorn
Engr. Original: Drew Jensen
Jean Hollis Weber Christoph Noack
Klaus-Jürgen Weghorn

Vorwort
Klaus-Jürgen Weghorn
Engr. Original: Jean Hollis Weber

Inhalt Base
Robert Großkopf
Jost Lange
Jürgen Thomas
Michael Niedermair
Jochen Schiffers

Glossar
Erhardt Balthasar
Stefan Haas
Christian Kühl
Michael Niedermair
Florian Reisinger
Klaus-Jürgen Weghorn
Jochen Schiffers

Rückmeldung (Feedback)

Kommentare oder Vorschläge zu diesem Dokument können Sie in deutscher Sprache an die Adresse discuss@de.libreoffice.org senden.

Vorsicht

Alles, was an eine Mailingliste geschickt wird, inklusive der E-Mail-Adresse und anderer persönlicher Daten, die die E-Mail enthält, wird öffentlich archiviert und kann nicht gelöscht werden. Also, schreiben Sie mit Bedacht!

Datum der Veröffentlichung und Softwareversion

Veröffentlicht am 1.8.2015. Basierend auf der LibreOffice Version 5.0
Inhalt

Vorwort................................................................................................................................................11
Für wen ist dieses Buch?..................................................................................................................12
Was finden Sie in diesem Buch?......................................................................................................12
Wo bekomme ich mehr Hilfe?........................................................................................................12
Hilfesystem.......................................................................................................................................12
Freier Onlinesupport.......................................................................................................................12
Bezahlter Support und Schulungen................................................................................................13
Sie sehen vielleicht etwas anderes..................................................................................................13
Anmerkung für Macintosh Nutzer.....................................................................................................13
Verwenden von Tipps, Hinweisen und Warnungen.......................................................................14
Wer hat dieses Buch geschrieben?..................................................................................................14
FAQs (häufig gestellte Fragen)..........................................................................................................15

Einführung in Base............................................................................................................................16

Base – ein Container für Datenbankinhalte......................................................................................18
Formulare – Start für die Dateneingabe............................................................................................20
Tabellen – Grundlagen für die Dateneingabe...................................................................................21
Abfragen – Auswertungsmöglichkeiten für eingegebene Daten.....................................................23
Berichte – Präsentationen der Datenauswertung.............................................................................23
Sicherer Umgang mit einer Base-Datei..............................................................................................26

Eine einfache Datenbank – Testbeispiel im Detail.........................................................................26
Tabellenerstellung...............................................................................................................................27
Eingabeformular.................................................................................................................................35
Mit dem Tabulator zum Unterformular..............................................................................................42
Navigationsleiste des Hauptformulars auch im Unterformular aktivieren.......................................43
Eingabe eines Kontrollfeldes einschränken.......................................................................................43

Abfrage..............................................................................................................................................45
Ausdruck im Bericht............................................................................................................................49
Abstände zwischen den Berichtsereichen einstellen.......................................................................54
Beeinflussung eines Textfeldinhaltes durch eine Formel.................................................................55
Umstellung der Formatierung eines Textfeldes..............................................................................55
Verschieben von Feldern im Report-Builder...................................................................................56
Erweiterungen der Beispieldatenbank..............................................................................................56
Weiteres Einführungs- und Beispielmaterial....................................................................................57

Datenbank erstellen............................................................................................................................58
Allgemeines bezüglich der Erstellung einer Datenbank....................................................................59
Neue Datenbank als interne Datenbank............................................................................................59

Zugriff auf externe Datenbanken.......................................................................................................61
MySQL/MariaDB-Datenbanken...........................................................................................................62
Erstellen eines Nutzers und einer Datenbank..................................................................................62
MySQL-Verbindung direkt mit der Extension...................................................................................63
MySQL-Verbindung über JDBC.........................................................................................................63
MySQL-Verbindung über ODBC.........................................................................................................63
odbinst.ini.........................................................................................................................................64
odbc.ini...........................................................................................................................................64

Verbindung zur MySQL-Datenbank über den Datenbank-Assistenten............................................64
Die direkte Verbindung......................................................................................................................66
Die ODBC-Verbindung.....................................................................................................................70
Die JDBC-Verbindung.......................................................................................................................71

PostgreSQL.......................................................................................................................................72
Erstellen eines Nutzers und einer Datenbank ........................................ 72
Direkte Verbindung zu Base..................................................................... 72
dBase-Datenbanken............................................................................... 74
Tabellendokumente............................................................................... 76
Thunderbird Adressbuch........................................................................ 77
Texttabellen.............................................................................................. 78
Texttabellen innerhalb einer internen HSQLDB-Datenbank...................... 78
Texttabellen als Grundlage für eine eigenständige Datenbankdatei......... 81
Firebird..................................................................................................... 83
Erstellen eines Nutzers und einer Datenbank........................................ 83
Firebird-Verbindung über JDBC.............................................................. 84
Firebird-Verbindung über ODBC.............................................................. 85
odbinst.ini........................................................................................... 85
odbc.ini............................................................................................... 85
Nachträgliche Bearbeitung der Verbindungseigenschaften.................... 85

Tabellen..................................................................................................... 90
Allgemeines zu Tabellen......................................................................... 91
Beziehungen von Tabellen....................................................................... 91
Beziehungen zwischen Tabellen allgemein............................................ 91
Tabellen und Beziehungen der Beispieldatenbank................................. 94
Tabellen Medienaufnahme.................................................................... 94
Tabellen Ausleihe.................................................................................... 96
Tabellen Nutzerverwaltung.................................................................... 96

Erstellung von Tabellen........................................................................... 97
Erstellung mit der grafischen Benutzeroberfläche................................... 98
Primärschlüssel....................................................................................... 102
Formatierung von Feldern....................................................................... 103
Einstellung eines Indexes........................................................................ 105
Probleme bei der Änderung von Tabellen.............................................. 107
Mängel der grafischen Tabellenanordnung............................................ 109
Direkte Eingabe von SQL-Befehlen....................................................... 110
Tabellenerezeugung.............................................................................. 111
Tabellenänderung.................................................................................. 114
Tabellen löschen.................................................................................... 117

Verknüpfung von Tabellen....................................................................... 117
Eingabe von Daten in Tabellen............................................................... 122
Eingabe über die grafische Benutzeroberfläche der Tabelle.................... 122
Sortieren von Tabellen.......................................................................... 124
Suchen in Tabellen................................................................................ 125
Filtern von Tabellen............................................................................. 127
Eingabemöglichkeiten über SQL direkt................................................. 129
Neue Datensätze einfügen...................................................................... 129
Bestehende Datensätze ändern.............................................................. 129
Bestehende Datensätze löschen............................................................ 130
Import von Daten aus anderen Datenquellen....................................... 130
Importierte Daten an bestehende Daten einer Tabelle anfügen.............. 132
Neue Tabelle beim Import erstellen...................................................... 133
Daten Aufsplitten beim Import........................................................... 135
Mängel dieser Eingabemöglichkeiten.................................................... 136

Formulare................................................................................................ 137
Formulare als Eingabebeerleichterung................................................... 138
Erstellung von Formularen.................................................................... 138
Einfaches Formular.............................................................................. 138
Symbolleisten des Formularentwurfs.................................................... 140
Formulargrundung über den Navigator.................................................................140
Formulargrundung über ein Formularfeld..........................................................141
Externe Formulare..................................................................................................143
Formular-Eigenschaften..........................................................................................143
Eigenschaften der Kontrollfelder............................................................................146
Standardeinstellungen vieler Kontrollfelder.........................................................147
Textfeld.....................................................................................................................153
Numerisches Feld.....................................................................................................154
Datumsfeld...............................................................................................................155
Zeitfeld......................................................................................................................155
Währungsfeld..........................................................................................................156
Formattedes Feld.....................................................................................................157
Listenfeld..................................................................................................................158
Kombinationsfeld....................................................................................................163
Markierfeld..............................................................................................................166
Optionsfeld..............................................................................................................168
Grafisches Kontrollfeld..........................................................................................168
Maskiertes Feld........................................................................................................169
Tabellen-Kontrollfeld.............................................................................................170
Beschriftungsfeld....................................................................................................172
Gruppierungsrahmen............................................................................................173
Schaltfläche.............................................................................................................176
Grafische Schaltfläche...........................................................................................178
Navigationsleiste....................................................................................................178
Drehfeld und Bildlaufleiste.....................................................................................180
Verstecktes Steuerelement.....................................................................................181
Mehrfachselektion..................................................................................................182
Einfaches Formular komplett erstellt.....................................................................183
Felder als Gruppe hinzufügen................................................................................184
Felder anpassen.......................................................................................................185
Felder einzeln hinzufügen.......................................................................................188
Tabellenkontrollfeld...............................................................................................189
Hauptformular und Unterformular.........................................................................194
Eine Ansicht – viele Formulare.............................................................................207
Fehlermeldungen bei der Eingabe in Formulare....................................................214
Suchen und Filtern in Formularen über die Navigationsleiste..............................215
Datensatzsuche mit Parametern............................................................................215
Filterung mit dem Autofilter..................................................................................216
Filterung mit dem formularbasierten Filter..........................................................218
Filterung über den Standardfilter........................................................................220
Fazit.........................................................................................................................221
Dateneingabe und Navigation...............................................................................221
Drucken aus Formularen......................................................................................223
Abfragen..................................................................................................................224
Allgemeines zu Abfragen......................................................................................225
Eingabemöglichkeiten für Abfragen.......................................................................225
Abfrageerstellung mit der grafischen Benutzeroberfläche.....................................225
Funktionen in der Abfrage......................................................................................233
Beziehungseigenschaften definieren......................................................................236
Abfrageerweiterungen im SQL-Modus.................................................................240
Abfrageerweiterungen im SQL-Modus.................................................................242
Verwendung eines Alias in Abfragen.................................................................252
Abfragen für die Erstellung von Listenfeldern.....................................................253
Abfragen als Grundlage von Zusatzinformationen in Formularen.......................256
Berichte.................................................................273
Berichte mit dem Report-Designer..........................274
Die Benutzeroberfläche des Report-Designer ..........274
   Allgemeine Eigenschaften von Feldern...............282
      Besondere Eigenschaften des grafischen Kontrollfeldes........284
      Diagramme im Bericht einbinden......................286
Dateneigenschaften von Feldern............................290
Funktionen im Report-Designer..............................291
   Formeleingaben...............................................291
   Benutzerdefinierte Funktionen..........................298
   Formeleingabe für ein Feld...............................300
   Bedingte Anzeige.............................................300
   Bedingte Formatierung.....................................301
Beispiele für Berichte mit dem Report-Designer.........301
   Rechnungserstellung........................................302
   Ausdruck von Berichten zum aktuellen Datensatz des Formulars..309
      Aufbau der Filtertabelle................................310
      Aufbau des Makros zum Start des gefilterten Berichtes......310
   Wechselnde Einfärbung von Zeilen........................311
   Zweispaltige Berichte......................................313
Fehlerquellen bei Berichten..................................317
   Der Inhalt eines Feldes aus einer Abfrage erscheint nicht......317
   Ein Bericht lässt sich nicht ausführen..................317
Datenbank-Anbindung...........................................319
   Allgemeines zur Datenbank-Anbindung..................320
   Anmeldung der Datenbank................................320
   Datenquellenbrowser........................................320
      Daten in Text.............................................322
      Daten in Felder...........................................326
   Seriendruck...................................................326
      Aktuelle Dokument-Datenquelle.........................327
      Explorer ein/aus.........................................327
   Serienbrieffrstellung.........................................327
   Erstellung von Etiketten....................................335
   Serienbriebe und Etiketten direkt erstellen............338
   Serienbrieffelder mit der Maus erstellen................338
   Serienbrieffelder über Feldbefehle erstellen............339
Externe Formulare...............................................341
   Vorteil externer Formulare...............................342
   Nachteil externer Formulare..............................342
Datenbanknutzung in Calc....................................342
   Daten in Calc einfügen....................................343
   Daten aus Calc in eine Datenbank exportieren.........345
Inhalt eines Kontrollfelds ändern ................................................................. 397
Zeile einer Datenmenge ändern ................................................................... 397
Zeilen anlegen, ändern, löschen ................................................................. 397
Kontrollfelder prüfen und ändern ................................................................. 398
Englische Bezeichner in Makros ................................................................. 399
Eigenschaften bei Formularen und Kontrollfeldern ..................................... 399
Schrift ........................................................................................................ 399
Formular ................................................................................................. 400
Einheitlich für alle Arten von Kontrollfeld ................................................ 400
Einheitlich für viele Arten von Kontrollfeld ............................................... 400
Textfeld – weitere Angaben ...................................................................... 400
Numerisches Feld .................................................................................... 401
Datumsfeld .............................................................................................. 401
Zeitfeld ..................................................................................................... 402
Währungsfeld .......................................................................................... 402
Formatiertes Feld .................................................................................... 402
Listenfeld .................................................................................................. 403
Kombinationsfeld ..................................................................................... 404
Markierfeld, Optionsfeld .......................................................................... 404
Maskiertes Feld ....................................................................................... 405
Tabellenkontrollfeld .................................................................................. 405
Beschriftungsfeld ...................................................................................... 405
Gruppierungsrahmen ............................................................................... 405
Schaltfläche .............................................................................................. 405
Navigationsleiste ..................................................................................... 406
Methoden bei Formularen und Kontrollfeldern ......................................... 406
In einer Datenmenge navigieren ............................................................... 406
Datenzeilen bearbeiten ............................................................................ 407
Einzelne Werte bearbeiten ....................................................................... 408
Parameter für vorbereitete SQL-Befehle ............................................... 410
Bedienbarkeit verbessern .......................................................................... 410
Automatisches Aktualisieren von Formularen ......................................... 410
Filtern von Datensätzen .......................................................................... 411
daten über den Formularfilter filtern ........................................................ 414
Daten aus Textfeldern auf SQL-Tauglichkeit vorbereiten ....................... 415
Werte in einem Formular vorausberechnen ............................................. 415
Die aktuelle Office-Version ermitteln ...................................................... 416
Wert von Listenfeldern ermitteln .............................................................. 417
Listenfelder durch Eingabe von Anfangsbuchstaben einschränken .......... 418
Datumswert aus einem Formularwert in eine Datumsvariable umwandeln 419
Suchen von Datensätzen .......................................................................... 420
Suchen in Formularen und Ergebnisse farbig hervorheben ....................... 422
Rechtschreibkontrolle während der Eingabe ............................................ 426
Kombinationsfelder als Listenfelder mit Eingabemöglichkeit .................... 428
Textanzeige im Kombinationsfeld ............................................................. 429
Fremdschlüsselwert vom Kombinationsfeld zum numerischen Feld übertragen 431
Kontrollfunktion für die Zeichenlänge der Kombinationsfelder ................ 437
Datensatzaktion erzeugen ....................................................................... 438
Navigation von einem Formular zum anderen ........................................ 438
Hierarchische Listenfelder ....................................................................... 439
Zeiteingaben mit Millisekunden ............................................................... 443
Ein Ereignis – mehrere Implementationen .............................................. 444
Abspeichern mit Nachfrage ..................................................................... 444
Primärschlüssel aus Nummerierung und Jahreszahl ............................... 445
Datenbankaufgaben mit Makros erweitert ................................................. 446
Verbindung mit Datenbanken erzeugen .................................................. 446
Daten von einer Datenbank in eine andere kopieren ................................ 447
Zugriff auf Abfragen. ..............................................................................................................448
Datenbanksicherungen erstellen ..........................................................................................449
Datenbanken komprimieren ..............................................................................................452
Tabellenindex heruntersetzen bei Autowert-Feldern ..........................................................453
Drucken aus Base heraus .....................................................................................................453
  Druck von Berichten aus einem internen Formular heraus ................................................454
  Start, Formatierung, direkter Druck und Schließen des Berichts .....................................454
  Druck von Berichten aus einem externen Formular heraus .............................................456
  Serienbriefdruck aus Base heraus ......................................................................................457
  Drucken über Textfelder ..................................................................................................457
Aufruf von Anwendungen zum Öffnen von Dateien ............................................................458
Aufruf eines Mailprogramms mit Inhaltsvorgaben ..............................................................459
Mauszeiger beim Überfahren eines Links ändern ...............................................................461
Formulare ohne Symbolleisten präsentieren ......................................................................461
  Formulare ohne Symbolleisten in einem Fenster ..........................................................461
  Formulare im Vollbildmodus .........................................................................................463
  Formular direkt beim Öffnen der Datenbankdatei starten .............................................463
MySQL-Datenbank mit Makros ansprechen ......................................................................464
MySQL-Code in Makros .....................................................................................................464
Temporäre Tabelle als individueller Zwischenspeicher ........................................................464
Dialoge ..................................................................................................................................464
  Dialoge starten und beenden .........................................................................................465
  Einfacher Dialog zur Eingabe neuer Datensätze ............................................................466
  Dialog zum Bearbeiten von Daten in einer Tabelle ........................................................468
  Fehlinträge von Tabellen mit Hilfe eines Dialogs bereinigen ........................................473
Makrozugriff mit Access2Base ............................................................................................481
Wartung .............................................................................................................................482
  Allgemeines zur Wartung von Datenbanken .................................................................483
  Datenbank komprimieren ..............................................................................................483
  Autowerte neu einstellen ...............................................................................................483
  Datenbankeigenschaften abfragen ..................................................................................483
  Daten exportieren ..........................................................................................................484
  Tabellen auf unnötige Einträge überprüfen .................................................................485
    Einträge durch Beziehungsdefinition kontrollieren ......................................................485
    Einträge durch Formular und Unterformular bearbeiten ..........................................486
    Verwaiste Einträge durch Abfrage ermitteln .............................................................487
  Datenbankgeschwindigkeit .............................................................................................487
    Einfluss von Abfragen ..................................................................................................487
    Einfluss von Listenfeldern und Kombinationsfeldern ...............................................488
    Einfluss des verwendeten Datenbanksystems ............................................................488
Anhang ..............................................................................................................................489
Barcode ..............................................................................................................................490
Datentypen des Tabelleneditors ..........................................................................................490
  Ganzzahlen ....................................................................................................................490
  Fließkommazahlen ........................................................................................................490
  Text ...............................................................................................................................491
  Zeit ..................................................................................................................................491
  Sonstige ........................................................................................................................491
Datentypen in StarBasic ......................................................................................................492
  Zahlen ...........................................................................................................................492
  Sonstige ........................................................................................................................492
Einbaute Funktionen und abgespeicherte Prozeduren .........................................................492
  Numerisch .....................................................................................................................493
Vorwort
Für wen ist dieses Buch?
Jeder, der sich in das Modul Base von LibreOffice einarbeiten und tiefer einsteigen will, findet hier die Möglichkeit. Sei es, dass Sie noch nie mit Datenbanken und damit in einem Datenbanksystem mit DBMS (Database management system) gearbeitet haben oder sei es, dass Sie ein anderes Datenbanksystem aus einer OfficeSuite oder ein eigenständiges Datenbanksystem wie beispielsweise MySQL gewohnt sind.

Was finden Sie in diesem Buch?
Dieses Buch führt Sie in die gebräuchlichsten Funktionen von LibreOffice-Base ein:

- Einführung
- Datenbank erstellen
- Ein- und Ausgabe der Datenbank: Tabellen, Formulare, Abfragen, Berichte
- Aufgaben einer Datenbank
- Makros
- Pflege und Wartung von Datenbanken
- und einiges mehr

Wo bekomme ich mehr Hilfe?
Dieses Buch, wie auch die anderen LibreOffice-Handbücher, das eingebaute Hilfesystem und die Benutzer-Supportsysteme setzen voraus, dass Sie mit Ihrem Computer und den Basisfunktionen wie das Starten eines Programms, Öffnen und Speichern von Dateien vertraut sind.

Hilfesystem
LibreOffice besitzt ein umfangreiches Hilfesystem.
Um zu dem Hilfesystem zu gelangen, drücken Sie F1 oder wählen Sie LibreOffice Hilfe aus dem Hilfemenü. Zusätzlich können Sie wählen, ob Sie Tipps, Erweiterte Tipps und den Office-Assistenten einschalten (Extras → Optionen → LibreOffice → Allgemein).
Wenn die Tipps eingeschaltet sind, platziern Sie den Mauszeiger über eines der Symbole um eine kleine Box («Tooltip») angezeigt zu bekommen. Darin befindet sich eine kurze Erklärung der Funktion des Symbols. Um noch mehr Erklärungen zu erhalten, wählen Sie Hilfe → Direkthilfe und halten den Mauszeiger über das Symbol.

Freier Onlinesupport
Tabelle 1: Kostenlose Unterstützung für Nutzer von LibreOffice

<table>
<thead>
<tr>
<th>Freier LibreOffice-Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• FAQ</td>
<td>• Antworten auf häufig gestellte Fragen (Frequently Asked Questions):</td>
</tr>
<tr>
<td></td>
<td><a href="http://de.libreoffice.org/discover/faq/">http://de.libreoffice.org/discover/faq/</a></td>
</tr>
<tr>
<td></td>
<td>• Dokumentation</td>
</tr>
<tr>
<td></td>
<td>• Handbücher und andere Dokumentationen:</td>
</tr>
<tr>
<td></td>
<td><a href="http://de.libreoffice.org/get-help/documentation/">http://de.libreoffice.org/get-help/documentation/</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://wiki.documentfoundation.org/Documentation/de">http://wiki.documentfoundation.org/Documentation/de</a></td>
</tr>
<tr>
<td></td>
<td>• Mailinglisten</td>
</tr>
<tr>
<td></td>
<td>• Freier Community-Support durch ein Netzwerk an unabhängigen, erfahrenen Benutzern:</td>
</tr>
<tr>
<td></td>
<td><a href="http://de.libreoffice.org/get-help/mailing-lists/">http://de.libreoffice.org/get-help/mailing-lists/</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://de.libreoffice.org/get-help/nabble/">http://de.libreoffice.org/get-help/nabble/</a></td>
</tr>
<tr>
<td></td>
<td>• Forum</td>
</tr>
<tr>
<td></td>
<td>• Wir unterhalten ein eigenes Forum:</td>
</tr>
<tr>
<td></td>
<td><a href="http://ask.libreoffice.org/de/questions/">http://ask.libreoffice.org/de/questions/</a></td>
</tr>
<tr>
<td></td>
<td>• Internationaler Support</td>
</tr>
<tr>
<td></td>
<td>• Die LibreOffice-Webseite in Ihrer Sprache</td>
</tr>
<tr>
<td></td>
<td><a href="http://de.libreoffice.org/">http://de.libreoffice.org/</a></td>
</tr>
<tr>
<td></td>
<td>• Die internationalen Mailinglisten:</td>
</tr>
<tr>
<td></td>
<td><a href="http://wiki.documentfoundation.org/Local_Mailing_Lists">http://wiki.documentfoundation.org/Local_Mailing_Lists</a></td>
</tr>
</tbody>
</table>

Benutzer können umfassenden Onlinesupport aus der Community über Mailinglisten und Foren bekommen. Andere Webseiten, die von Nutzern geführt werden, bieten auch kostenlose Tipps und Anleitungen.

**Bezahlter Support und Schulungen**


**Sie sehen vielleicht etwas anderes**

LibreOffice läuft auf Windows, Linux, Mac OS X, FreeBSD und anderen Unix-Varianten, von denen jedes Betriebssystem unterschiedlichste Versionen hat, und kann von den Nutzern (Schriftarten, Farben, Themen) angepasst werden.


**Anmerkung für Macintosh Nutzer**

<table>
<thead>
<tr>
<th><strong>Windows/Linux entspricht am Mac</strong></th>
<th><strong>Effekt</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menü-Auswahl</strong> Extras → Optionen</td>
<td>LibreOffice → Einstellungen</td>
</tr>
<tr>
<td><strong>Rechts-Klick</strong></td>
<td>Control+Klick</td>
</tr>
<tr>
<td><strong>Ctrl (Control)</strong> oder <strong>Strg (Steuerung)</strong></td>
<td>⌘ (Command)</td>
</tr>
<tr>
<td><strong>F5</strong></td>
<td>Shift+⌘+F5</td>
</tr>
<tr>
<td><strong>F11</strong></td>
<td>⌘+T</td>
</tr>
</tbody>
</table>

**Verwenden von Tipps, Hinweisen und Warnungen**

In den Handbüchern werden besondere Information zusätzlich gekennzeichnet: Tipps, Hinweise und Warnungen.

| **Tipp** | Ein Tipp beschreibt eine praktische aber nicht wesentliche Information, die nicht in den Textfluss passt. |
| **Hinweis** | Ein Hinweis enthält Informationen mit Bezug auf den Text. Er erklärt, beschreibt oder kommentiert eine Aussage, um die Aufmerksamkeit des Lesers darauf zu richten. |
| **Vorsicht** | Warnungen (Vorsicht) zeigen Operationen, die zu Datenverlust führen können. |

**Wer hat dieses Buch geschrieben?**

FAQs (häufig gestellte Fragen)

Zu LibreOffice gibt es immer wieder häufig gestellte, allgemeine Fragen. Hier ein paar Antworten:

Wie ist LibreOffice lizenziert?
LibreOffice wird unter der von der Open Source Initiative (OSI) anerkannten Lesser General Public License (LGPL) vertrieben. Die LGPL-Lizenz ist auf der LibreOffice Website verfügbar: http://www.libreoffice.org/download/license/

Darf ich LibreOffice an jeden vertreiben?
Ja.

Auf wie vielen Computern kann ich LibreOffice installieren?
Auf so vielen, wie Sie möchten.

Darf ich es verkaufen?
Ja.

Darf ich LibreOffice in meinem Unternehmen kostenlos verwenden?
Ja.

Ist LibreOffice in meiner Sprache verfügbar?

Wie können Sie es kostenlos anbieten?
LibreOffice wird von Freiwilligen entwickelt und gepflegt und hat die Unterstützung von mehreren Unternehmen.

Ich schreibe eine Software-Anwendung, darf ich Programmcode von LibreOffice in meinem Programm einbauen?
Sie können dies im Rahmen der Parameter, die in der LGPL gesetzt sind, tun. Lesen Sie hierzu die Lizenzvereinbarung: http://www.libreoffice.org/download/license/

Wozu brauche ich Java, um LibreOffice laufen zu lassen? Ist es in Java geschrieben?
LibreOffice ist nicht in Java geschrieben, es wird in der Sprache C++ geschrieben. Java ist eine von mehreren Sprachen, die verwendet werden, um die Software zu erweitern. Das Java JDK / JRE ist nur für einige Funktionen erforderlich. Am wichtigsten davon ist die relationale Datenbank-Engine HSQLDB.


Wie kann ich zu LibreOffice beitragen?
Einführung in Base
Einführung

Im täglichen Büroeinsatz werden häufig Tabellenkalkulationen dazu benutzt, Datensammlungen zu erstellen um anschließend damit eventuell noch kleine Berechnungen durchzuführen. Die Tabellenicht ist sofort da, der Inhalt einfach einzugeben – da fragen sich dann viele Nutzer, warum es denn eine Datenbank sein sollte. Dieses Handbuch versucht den Unterschied zwischen Tabellenkalkulation und Datenbank herauszuarbeiten. Zu Beginn wird erst einmal kurz dargestellt, was eine Datenbank denn überhaupt leisten kann.

Hinweis

In der Fachsprache wird statt von Datenbanken von einer Benutzeroberfläche und einem «Datenbanksystem» gesprochen. Dieser Begriff umfasst das «Datenbankmanagementsystem» (DBMS) und den eigentlichen Datenbestand, die «Datenbank».

Base hingegen bietet einen Zugriff auf verschiedene Datenbanksysteme über eine grafische Benutzeroberfläche. Base arbeitet standardmäßig mit dem eingebetteten Datenbanksystem «HSQLDB».

Das gesamte Handbuch bezieht sich, in der Hauptsache auf zwei Beispieldatenbanken. Die eine Datenbank hat die Bezeichnung «Medien_ohne_Makros.odb», die andere Datenbank ist entsprechend mit Makros erweitert worden und trägt die Bezeichnung «Medien_mit_Makros.odb». Beide Datenbanken sollen einen Bibliotheksbetrieb ermöglichen: Medienaufnahme, Nutzeraufnahme, Medienverleih und alles, was damit verbunden ist wie z. B. das Anmahnen säumiger EntleiherInnen.

Daneben sind viele Beispieldatenbanken1 beigefügt, die nur auf spezielle Probleme zugeschnitten sind:

- Beispiel_Sport.odb (Kapitel «Einführung in Base»)
- Beispiel_CSV_Einbindung.odb (Kapitel «Datenbank erstellen»)
- Beispiel_Cursorsprung_Subform_Mainform.odb (Kapitel «Formulare»)
- Beispiel_Bericht_bedingte_Einblendung_von_Grafiken.odb (Kapitel «Berichte»)
- Beispiel_Bericht_Rechnung.odb (Kapitel «Berichte»)
- Beispiel_Bericht_Zeilen_Farbwechsel_Spalten.odb (Kapitel «Berichte»)
- Beispiel_PDFFormular_Import.odb (Kapitel «Datenbank-Anbindung»)
- Beispiel_Autotext_Suchmarkierung_Rechtschreibung.odb (Kapitel «Datenbank-Aufgaben»)
- Beispiel_Dokumente_einlesen_auslesen.odb (Kapitel «Datenbank-Aufgaben»)
- Beispiel_Suchen_und_Filtern.odb (Kapitel «Makros»)
- Beispiel_Direktberechnung_im_Formular.odb (Kapitel «Makros»)
- Beispiel_Combobox_Listfeld.odb (Kapitel «Makros»)
- Beispiel_Fortlaufende_Nummer_Jahr.odb (Kapitel «Makros»)
- Beispiel_Datenbank_Serienbrief_direkt.odb (Kapitel «Makros»)
- Beispiel_Mailstart_Dateiaufruf.odb (Kapitel «Makros»)
- Beispiel_Dialoge.odb (Kapitel «Makros»)

1 Alle Beispieldatenbanken können auf http://de.libreoffice.org/get-help/documentation/ herunter geladen werden.

Einführung in Base: Einführung
Wie jede Software läuft auch LO-Base nicht vollkommen fehlerfrei. Besonders ärgerlich sind hier die «Regressionen», also Rückschritte von einer vorhergehenden Version zur gerade aktuellen Version. Der folgende Link führt zu den momentan noch offenen Regressionen:

Offene Bugs von LibreOffice-Base, die nicht in vorherigen Versionen von LO auftauchen

Ein Blick auf die Bug-Liste kann also helfen, Unterschiede zwischen Dokumentation und eigener Programmversion zu verstehen.

In diesem Handbuch werden Anführungszeichen mit unterschiedlichen Intentionen verwendet:

"Tabellenname"."Feldname" → Tabellen und Felder in Datenbanken werden in doppelten Anführungszeichen oben wieder gegeben.

'Wert' → Werte, die in die Datenbank eingegeben oder aus ihr gelesen werden, sollten in einfachen Anführungszeichen erscheinen.

«Hervorhebungen» → Alles, was sonst hervorgehoben werden soll, erscheint in der französischen Form der Anführungszeichen.

Base – ein Container für Datenbankinhalte

Eine Base-Datei ist eigentlich nur ein gepacktes Verzeichnis, in dem Informationen für die verschiedenen Arbeitsbereiche von Base stecken. In der täglichen Nutzung startet eine Base-Datei erst einmal mit der folgenden Ansicht:

In den Arbeitsbereichen «Formulare» und «Berichte» können die jeweiligen Elemente auch innerhalb einer Verzeichnisstruktur angeordnet werden:

Dies geht entweder direkt beim Abspeichern über den Speicherdialog oder durch Neugründung von Verzeichnissen über Einfügen → Ordner.

Obwohl die Basis für eine Datenbank durch Tabellen gebildet wird, startet Base mit der Formularansicht, weil Formulare in der Regel die Elemente sind, mit denen die tägliche Datenbankarbeit vonstatten geht. Über die Formulare werden Einträge in die Tabellen vorgenommen und Inhalte aus den Tabellen ausgewertet.
Einfache Formulare bilden lediglich eine Tabelle wie im oben sichtbaren Tabellenkontrollfeld mit den Namen ab. Dieses Formular erfüllt durch seine Struktur einige zusätzliche Punkte.

- Neue Mediennutzer können direkt in das Tabellenkontrollfeld eingegeben werden.
• Das Entleihdatum wird mit dem aktuellen Datum vorgegeben. In dem daneben stehenden Listenfeld werden die zu entliehenden Medien ausgewählt. Dabei können keine Medien ausgewählt werden, die zur Zeit noch entliehen sind.

• Die ausgewählten Daten werden mit dem Button Aktualisieren in das Tabellenkontrollfeld für den aktuellen Ausleihvorgang übernommen.

• Im Tabellenkontrollfeld für die Rückgabe ist es nicht möglich, einen Datensatz einfach zu löschen. Nur die Felder «Rück-Datum» und «Verlängerung» können bearbeitet werden. War ein Medienutzer vorübergehend gesperrt und hat seine entliehenen Medien zurückgegeben, so lässt sich die Ausleihe über den Button Aktualisieren wieder freischalten.

Alle diese Funktionen können ohne den Einsatz von zusätzlicher Programmierung mit Makros gewährleistet werden, wenn entsprechend an den Formularen gefeilt wird.

**Tabellen – Grundlagen für die Dateneingabe**


Die Tabelle "Ausleihe" steht in direkter Beziehung zu den Tabellen "Medien" und "Leser".


Die Beziehungen zwischen den Tabellen haben hier den Vorteil, dass die Schreibarbeit im Formular erheblich reduziert wird. Statt bei jeder Ausleihe ein Medium zumindest unverwechselbar und für jeden erkennbar aufzuschreiben und Leser mit Vor- und Nachnamen zu notieren, wird einfach im Inhalt der anderen Tabellen nachgesehen. Schließlich wird dasselbe Medium später noch öfter ausgeliehen und derselbe Leser kann schon beim ersten Entleihvorgang mehrere Medien entleihen.
Die Tabellengrundlage für das oben vorgestellte Formular sieht erst einmal recht nüchtern aus. In der oben abgebildeten Tabelle werden bis auf die Nutzerneueingabe alle Eingaben gemacht, die in dem Formular möglich sind. Dabei werden im Formular die Verbindungen dieser Tabelle zu den anderen Tabellen der Datenbank genutzt.

- Das erste Feld zeigt den für die meisten Datenbanken unabdingbaren, unverwechselbaren Inhalt, den Primärschlüssel ("ID"), der automatisch geschrieben wird. Mehr dazu im Kapitel «Beziehungen zwischen Tabellen allgemein».

- Das zweite Feld "Medien_ID" speichert den Primärschlüssel der Tabelle "Medien". Es weist über die Nummer auf den entsprechenden Inhalt in dieser Tabelle. Solch ein Verweis auf einen Primärschlüssel wird als Fremdschlüssel bezeichnet. Im Formular werden statt des Fremdschlüssels über ein Listenfeld der Fremdschlüssel, der Titel und der Verfasser angezeigt. Das Listenfeld gibt im Hintergrund den Fremdschlüssel an die Tabelle weiter.

- Das dritte Feld "Leser_ID" speichert den Primärschlüssel der Tabelle "Leser", also nur eine Nummer, die auf den Leser verweist. Im Formular wird aber der Nachname und der Vorname angegeben. Wie in der Tabelle zu sehen ist, hat der Leser mit der Primärschlüsselnummer '0' sehr viele Medien entliehen. Die Tabelle kann beliebig oft den einzigartigen Primärschlüssel der Tabelle "Leser" als Fremdschlüssel "Leser_ID" abspeichern. Auf keinen Fall darf aber ein Leser, der in dieser Tabelle über den Fremdschlüssel verzeichnet ist, gelöscht werden. Sonst wäre nicht mehr nachvollziehbar, wer denn nun das jeweilige Medium entliehen hat. Die Datenbank macht in den Standardeinstellungen so ein Löschen unmöglich. Der Fachbegriff dafür ist die Gewährung der «referentiellen Integrität».

- Im dritten Feld wird das Ausleihdatum abgespeichert. Ist dieses Datum abgespeichert und entspricht das Datum nicht dem aktuellen Datum, so erscheint der entsprechende Datensatz zu dem entsprechenden Leser im Formular im untersten Tabellenkontrollfeld zu Rückgabe der Medien.
• Im letzten Feld wird eine Verlängerung ermöglicht. Was hier eine 1, 2 usw. bedeutet wird an anderer Stelle festgelegt. Dafür enthält die Datenbank eine gesonderte Tabelle mit der Bezeichnung "Einstellungen".

Diese Eingaben reichen aus, um letztlich einen Bibliotheksbetrieb in Gang zu halten.

**Abfragen – Auswertungsmöglichkeiten für eingegebene Daten**

Abfragen zeigen eine Sicht auf die Tabellen. Sie bringen Inhalte mehrerer Tabellen zusammen in einer Übersicht. Abfragen werden nur in der Abfragesprache «SQL» gespeichert. Sie sind also keine neuen Tabellen, auch wenn sie von der Ansicht her in Base erst einmal gleich erscheinen.


• Alle Medien, bei denen das Rückgabedatum in der Tabelle "Ausleihe" nicht ausgefüllt ist, werden aufgelistet. Die Medien sind hier als zusätzliche Übersicht auch noch entsprechend benannt.

• Der Bezug zu den Lesern ist hergestellt über den Primärschlüssel der Tabelle "Leser"

• Aus dem Ausleihdatum "Leih_Datum" und dem aktuellen Datum wird die Zeitdifferenz in Tagen als "Leihzeit" angegeben.

• Von der Ausleihtzeit, die je nach Medienart unterschiedlich sein kann, wird die Leerzeit abgezogen. Daraus wird die verbleibende Restzeit gebildet.

• Bei einer Verlängerung wurde in der Tabelle "Einstellungen" vermerkt, dass der Wert ‘1’ für eine Verlängerung von 7 Tagen steht. Im vorletzten Datensatz mit der "Medien_ID" ’2’ ist so eine Verlängerung bei der Restzeit berücksichtigt worden.

**Berichte – Präsentationen der Datenauswertung**

Über Berichte werden die Daten so aufbereitet, dass sie sinnvoll ausgedruckt werden können. Formulare wie das folgende sind nicht dazu geeignet, z. B. einen sauber formatierten Brief auszugeben.
Bevor ein aktueller Bericht in Form einer Mahnung ausgedruckt werden kann müssen in diesem Formular erst einmal die Mahnungen bestätigt werden. Im oberen Tabellenkontrollfeld stehen dafür alle Namen der Leser, die ein Medium entliehen haben und für die eine negative Restzeit vermerkt ist.


Dieses Formular benötigt in der Fassung ohne Makros noch eine Eingabe durch den Nutzer. In der Fassung, die mit Makros arbeitet, wird einfach das Datum automatisch geschrieben und anschließend der Bericht zum Mahnungsdruck aufgerufen.
Mit Hilfe einer Abfrage lässt sich aus den getätigten Eingaben solch eine Mahnung zum Ausdruck fertig erstellen. Der Nutzer der Datenbank braucht dazu lediglich bei den Berichten den Bericht Mahnung auszuwählen und kann dann einen entsprechenden Mahnbrief an alle die Personen schicken, bei denen im vorher angegebenen Formular eine Mahnung bearbeitet wurde.

In so einem Bericht stehen also gegebenenfalls auf den folgenden Seiten jeweils auf einer Seite weitere Mahnungen für andere Personen. Sollte ein Leser so viele Medien entliehen haben, dass der Platz auf einer Seite nicht ausreicht, so wird die Tabelle auf der Folgeseite einfach fortgesetzt.

Ein so erstellter Bericht ist also umfangreicher als ein Serienbrief, der mit Writer erstellt wird: er stellt automatisch alle Datensätze zusammen, die gedruckt werden sollen und ordnet den zusätzlichen Text entsprechend an.

Ein ähnlicher Brief wie in der obigen Abbildung lässt sich sonst nur unter Zuhilfenahme von Makros realisieren, wie es in "Drucken über Textfelder" beschrieben wird.

---

Herrn
Bert Lederstrumpf
Neuenkirchen Str. 72
D 45793 Pusenrückel

22.04.12

Mahnung

Sehr geehrter Herr Lederstrumpf,

Leider haben Sie versäumt, die folgenden Medien rechtzeitig wieder zurück zu geben:

<table>
<thead>
<tr>
<th>Mahndatum</th>
<th>Mahn-Nr.</th>
<th>Medium</th>
<th>Leihdatum</th>
<th>Überzogen</th>
<th>Gebühr</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.04.12</td>
<td>2</td>
<td>3 - Traditionelle und kritische Theorie · von Herkheimer, Max</td>
<td>09.12.11</td>
<td>128 Tage</td>
<td>4,50 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>2</td>
<td>5 - I hear you moaning · von Edmunds, Dave</td>
<td>26.11.11</td>
<td>139 Tage</td>
<td>4,75 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>3</td>
<td>4 - Die neue deutsche Rechtschreibung · von Hermann, Ursula</td>
<td>09.11.11</td>
<td>151 Tage</td>
<td>5,25 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>1</td>
<td>1 - Das sogenannte Böse · von Lorenz, Konrad</td>
<td>04.04.12</td>
<td>4 Tage</td>
<td>0,00 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>1</td>
<td>7 - Das Postfix-Buch · von?</td>
<td>25.02.12</td>
<td>43 Tage</td>
<td>1,50 €</td>
</tr>
</tbody>
</table>

15,00 €

Mit freundlichen Grüßen

(Bibliotheksvorstand)
Sicherer Umgang mit einer Base-Datei


Die folgenden Hinweise sollten deshalb beim Umgang mit einer Base-Datei berücksichtigt werden:


Eine einfache Datenbank – Testbeispiel im Detail

Die Erstellung einer Datenbank wird in «Neue Datenbank als interne Datenbank» abgehandelt. Dort werden neben der internen HSQLDB verschiedene andere Datenbanksysteme eingebunden. Das folgende Beispiel basiert auf der Standarddatenbank, die mit LibreOffice als interne Datenbank installiert wird. Es wird also zuerst eine eingebettete Datenbank erstellt, die nicht angemeldet wird.

Erstes Kennzeichen bei der Erstellung einer Datenbank ist, dass zum Abschluss des Assistenten zuerst einmal ein Speicherort gesucht wird und die Datei abgespeichert wird.

Die Datenbank soll dazu dienen, einen Sportwettkampf in verschiedenen Disziplinen zu organisieren. Als Bezeichnung wurde deshalb «Beispiel_Sport»² gewählt.

2 Die Datenbank «Beispiel_Sport.odb» ist den Beispieldatenbanken für dieses Handbuch beigefügt.
Tabellenerstellung


Tabellen → Aufgaben → Tabelle in der Entwurfsansicht erstellen … wird angeklickt. Es öffnet sich der folgende Bildschirm:
Hier werden zuerst einmal die verschiedenen Feldnamen für die erste Tabelle eingetragen. Die Tabelle soll die Teilnehmer und Teilnehmerinnen enthalten. Sie ist hier erst einmal auf die wesentlichen Bestandteile reduziert.


Hier werden zuerst einmal die verschiedenen Feldnamen für die erste Tabelle eingetragen. Die Tabelle soll die Teilnehmer und Teilnehmerinnen enthalten. Sie ist hier erst einmal auf die wesentlichen Bestandteile reduziert.


Hier werden zuerst einmal die verschiedenen Feldnamen für die erste Tabelle eingetragen. Die Tabelle soll die Teilnehmer und Teilnehmerinnen enthalten. Sie ist hier erst einmal auf die wesentlichen Bestandteile reduziert.


Die Feldeigenschaft des Feldes "ID" wird bearbeitet. Für dieses Feld wird die automatische Einsetzung von aufsteigenden Zahlenwerten aktiviert: **Feldeigenschaften → Auto-Wert → Ja.**

Durch die Einstellung des Auto-Wertes sollte beim Verlassen der Feldtypenauswahl auf dem Zeilenkopf ein Schlüsselsymbol zeigen. Das zeigt an, dass dieses Feld den Primärschlüssel der Tabelle bildet. Wird nicht der Auto-Wert gewählt, so kann der Primärschlüssel auch über das Kontextmenü der Maus (rechte Maustaste → Primärschlüssel) ausgewählt werden.


---

**Einführung in Base: Eine einfache Datenbank – Testbeispiel im Detail**
Die Datenbankdatei ist ein gepacktes Verzeichnis von einzelnen Dateien. Das Speichern eines einzelnen Elementes wie der Tabelle wird deshalb nicht direkt in die Datenbankdatei selbst geschrieben. Deswegen muss nach der Erstellung von Tabel- len, Abfragen, Formularen und Berichten anschließend auch der Speicherbutton für die Datenbankdatei selbst betätigt werden.

Einzig das Abspeichern von eingegebenen Daten funktioniert beim Verlassen der Datenzeile automatisch.

Die Teilnehmer können jetzt eingegeben werden. Folgende Informationen fehlen allerdings auf den ersten Blick:

- Eine Liste der Sportarten, die die Teilnehmer belegen wollen.
- Für Wettkämpfe die Unterscheidung von Teilnehmern und Teilnehmerinnen.

<table>
<thead>
<tr>
<th>Feldname</th>
<th>Feldtyp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Text (VARCHAR)</td>
</tr>
<tr>
<td>Sportart</td>
<td>Text (VARCHAR)</td>
</tr>
</tbody>
</table>

Es wird eine Tabelle "Sportarten" erstellt. Da nicht so sehr viele unterschiedliche Sportarten in Frage kommen wird hier nicht der automatisch hoch zählende Primärschlüssel gewählt. Stattdessen wird der Feldtyp auf «Text» belassen, aber auf 5 Zeichen begrenzt. Die 5 Zeichen reichen aus, um damit auch passende Kürzel für die Sportarten zu finden.

Die Tabelle "Teilnehmer" soll noch einmal zum Bearbeiten, nicht zur Dateneingabe, geöffnet werden. Dies geschieht über das Kontextmenü der Tabelle.
Der Tabelle wird das Feld „Geschlecht“ hinzugefügt. Ein neues Feld kann mit der grafischen Benutzeroberfläche nur am Ende der Tabelle hinzugefügt werden. Über SQL ist auch die Möglichkeit gegeben, neue Felder an einer bestimmten Position einzufügen.

Die Länge des Textes in diesem Feld beschränkt sich auf einen Buchstaben. Es reicht also 'm' und 'w' als Eingabe.

Irgendwie müssen die beiden Tabellen verbunden werden, so dass jedem Teilnehmer mehrere Sportarten und jeder Sportart mehrere Teilnehmer zugeordnet werden können. Dies geschieht durch eine Tabelle, in der die Werte der beiden Primärschlüssel der Tabellen "Teilnehmer" und "Sportart" abgespeichert werden sollen. Da nicht mehr als die Kombination dieser Felder gespeichert werden soll sind beide Felder zusammen der Primärschlüssel für diese Tabelle. Um beiden Feldern einen Primärschlüssel zuzuweisen wird zunächst am Zeilenkopf eine Zeile markiert, danach die Großschreibtaste («Shift») betätigt und die zweite Zeile zusätzlich markiert. Über das Kontextmenü der Maus lässt sich jetzt der Primärschlüssel festlegen.


Die Tabelle wird unter dem Namen "rel_Teilnehmer_Sportart" abgespeichert.

Irgendwie müssen die beiden Tabellen verbunden werden, so dass jedem Teilnehmer mehrere Sportarten und jeder Sportart mehrere Teilnehmer zugeordnet werden können. Dies geschieht durch eine Tabelle, in der die Werte der beiden Primärschlüssel der Tabellen "Teilnehmer" und "Sportart" abgespeichert werden sollen. Da nicht mehr als die Kombination dieser Felder gespeichert werden soll sind beide Felder zusammen der Primärschlüssel für diese Tabelle. Um beiden Feldern einen Primärschlüssel zuzuweisen wird zunächst am Zeilenkopf eine Zeile markiert, danach die Großschreibtaste («Shift») betätigt und die zweite Zeile zusätzlich markiert. Über das Kontextmenü der Maus lässt sich jetzt der Primärschlüssel festlegen.


Die Tabelle wird unter dem Namen "rel_Teilnehmer_Sportart" abgespeichert.

**Tipp**

In diese Verbindungstabelle würden auch z.B. die Ergebnisse eines Wettkampfes gehören. Sollen allerdings mehrere Wettkämpfe abgehalten werden, so ist zusätzlich ein Wettkampfdatum in den gemeinsamen Primärschlüssel mit aufzunehmen.
Sind die Tabellen fertiggestellt so sollte eine entsprechende Verbindung der Tabellen fest definiert werden. So kann verhindert werden, dass in der Tabelle "rel_Teilnehmer_Sportart" z. B. eine Nummer für einen Teilnehmer auftaucht, die gar nicht in der Tabelle "Teilnehmer" verzeichnet ist.

Über Extras → Beziehungen wird das Fenster für die Beziehungsdefinition geöffnet.


Das Feld "rel_Teilnehmer_Sportart"."ID_Teilnehmer" soll nicht verändert werden, wenn "Teilnehmer"."ID" verändert würde. Das ist die Standardeinstellung. "Teilnehmer"."ID" wird sowieso nicht verändert, da es sich um ein automatisch hoch zählendes Feld handelt und dort kein Eingriff vorgenommen werden soll.
Der Datensatz der Tabelle "rel_Teilnehmer_Sportart" soll gelöscht werden, wenn "ID_Teilnehmer" gleich "Teilnehmer"."ID" ist und "Teilnehmer"."ID" gelöscht wird. Wenn also aus der Teilnehmer-Tabelle ein Teilnehmer entfernt wird, dann werden alle entsprechenden Datensätze aus der Tabelle "rel_Teilnehmer_Sportart" entfernt. Dieses Vorgehen wird als «kaskadierendes Löschen» bezeichnet.

Im nächsten Schritt werden "rel_Teilnehmer_Sportart"."ID_Sportart" und "Sportart"."ID" durch Ziehen der Maus bei gedrückter linker Maustaste verbunden. Auch hier soll ein Datensatz gelöscht werden, wenn die entsprechende Sportart gelöscht wird.

Für die Datenänderung bei "Sportart"."ID" ist allerdings eine andere Variante gewählt worden. Wird "Sportart"."ID" geändert, so wird entsprechend "rel_Teilnehmer_Sportart"."ID_Sportart" angepasst. So kann gegebenenfalls das Kürzel von maximal 5 Zeichen für eine Sportart problemlos angepasst werden, auch wenn bereits viele Datensätze in der Tabelle "rel_Teilnehmer_Sportart" stehen. Dieser Vorgang wird als «kaskadierendes Update» bezeichnet.


Eingabeformular

Formulare können direkt in der Entwurfsansicht oder mit Hilfe des Assistenten erstellt werden. Die Erstellung mit dem Assistenten greift dabei auch alten Hasen so gut unter die Arme, dass diese eben schnell den Assistenten durchlaufen lassen und anschließend ein erst einmal vom Grund her erstelltes Formular ihren Wünschen anpassen. Das ist häufig der deutlich zeit sparendere Weg.

Zuerst wird unter «Datenbank» der Container für «Formulare» gewählt. Dann unter «Aufgaben» «Formular unter Verwendung des Assistenten erstellen ...».

In der Hauptsache sollen in die Tabelle "Teilnehmer" Daten eingegeben werden. Die Tabelle "Sportarten" wird direkt mit den wenigen notwendigen Sportarten bestückt und nur selten ergänzt.

Von der Tabelle "Teilnehmer" werden alle Felder bis auf das Primärschlüsselfeld "ID" benötigt. Das Primärschlüsselfeld wird ja automatisch mit einem entsprechend unverwechselbarem Wert gefüllt.

Die Felder im Formular werden über die Pfeiltasten zwischen den beiden Auswahlkästchen bestückt.

Einführung in Base: Eine einfache Datenbank – Testbeispiel im Detail 35
Es soll ein Unterformular eingerichtet werden, in dem die Sportarten dem Teilnehmer zugewiesen werden können. Die Beziehung für das Unterformular wird aus der vorher definierten Beziehung ausgelesen. Allerdings muss sie hier noch durch einen Klick auf die Tabelle "rel_Teilnehmer_Sportart" bestätigt werden.

Aus der Tabelle "rel_Teilnehmer_Sportart" wird nur das Feld "ID_Sportart" benötigt. Das Feld "ID_Teilnehmer" wird über die Verbindung von Hauptformular zu Unterformular mit dem Wert versehen, der in der Teilnehmer-Tabelle für den Primärschlüssel des aktuellen Datensatzes steht.

Wie die Elemente im Hauptformular und wie im Unterformular angeordnet sind ist letztlich egal. Damit aber die Zuordnung von Daten auch für ungeübte einfach zu ersehen ist, sollte für das Hauptformular in unserem Beispiel nicht das Tabellenkontrollfeld gewählt werden. Im Unterformular soll das Feld später alle Sportarten des Teilnehmers anzeigen. Deswegen an dieser Stelle am besten das Tabellenkontrollfeld belassen.

Schritt 6 sollte so bleiben, wie er vorgesehen ist, also Dateneingabe – Das Formular zeigt existierende Daten an. Damit ist dann eine Neuengabe ebenso möglich wie eine Veränderung bereits bestehender Daten.


Zuerst wird mit der rechten Maustaste auf den Tabellenkopf des Tabellenkontrollfeldes geklickt. Dort steht «ID_Sportart». Im Kontextmenü wird Ersetzen durch → Listenfeld ausgewählt.

Anschließend soll das Listenfeld bearbeitet werden, damit es auch die entsprechenden Daten anzeigen kann. Wieder über das Kontextmenü geht es zu Spalte...

Es öffnen sich die Eigenschaften das ausgewählten Listenfeldes. Daten → Art des Listeninhaltes → SQL soll gewählt werden. Mit Hilfe von SQL (StructuredQueryLanguage – Standardabfragesprache für Datenbanken) soll das Feld seinen Inhalt aus der Tabelle "Sportart" erhalten.
Ist SQL ausgewählt, so erscheint in der Zeile Listeninhalt rechts ein Button mit 3 Punkten ⋯. Wird dieser Button gedrückt, so öffnet sich der Editor für die Erstellung von Abfragen. Die im Folgenden zu erstellenden Abfrage wird hier nur zusammengestellt und schließlich in dem Listenfeld selbst gespeichert.

Im Dialog «Tabelle oder Abfrage hinzufügen» erfolgt ein Klick auf Sportart → Hinzufügen → Schließen.

Für die erste Spalte wird in dem Abfrage-Editor das Feld "Sportart" ausgewählt. Dieses Feld soll im Listenfeld sichtbar sein.

Die zweite Spalte soll mit dem Feld "ID" bestückt werden. Dieses Feld gibt nachher seinen Wert an die Tabelle weiter, auf der das Unterformular basiert. Es werden also die ausgeschriebenen Worte dargestellt und die passenden Kürzel abgespeichert.

Die Abfrage wird mit dem Speicherbutton in die Eigenschaften des Listenfeldes übertragen. Der Editor wird anschließend geschlossen.

Im Listeninhalt ist jetzt der folgende Text zu lesen:

\[\text{Einführung in Base: Eine einfache Datenbank – Testbeispiel im Detail}\]
SELECT "Sportart", "ID" FROM "Sportart"


Nun soll noch der Titel geändert werden, der für das Feld "ID_Sportart" eingetragen wurde. Es wird zwar weiterhin "ID_Sportart" abgespeichert. Sichtbar ist aber "Sportart". Deswegen: **Allgemein → Titel → Sportart.**

Soll z.B. das Design bei anderen Feldern angepasst werden, so lassen sich diese am besten über den Formularnavigator ansteuern. Werden Felder direkt angeklickt, so werden nicht nur die Eingabefelder, sondern auch die Beschriftungsfelder markiert. Durch den Assistenten wurden sie zu Gruppen zusammengefasst. Das erfordert dann weitere Aktionen im Kontextmenü der Auswahl.

Der Formularnavigator wird über die Symbolleiste «Formular-Entwurf» am unteren Bildschirmrand gestartet.

| **Hinweis** | Manchmal kann es vorkommen, dass die Symbolleisten für die Erstellung von Formularen beim Bearbeiten der Formulare nicht korrekt geöffnet werden. Der Formularnavigator ist dann nicht zu finden.

Die Symbolleisten lassen sich über **Ansicht → Symbolleisten → Formular-Entwurf** sowie **Formular-Steuerelemente** einblenden. Sollten sie danach auch bei der Dateneingabe sichtbar sein, so muss hier entsprechend wieder die Ansicht ausgeschaltet werden. |

---

Einführung in Base: Eine einfache Datenbank – Testbeispiel im Detail

Wenn die Gestaltung nun beendet wurde kann das Formular gespeichert und geschlossen werden. Auch eine Speicherung der Base-Datei ist jetzt wieder sinnvoll.


Bei der Nutzung des Formulars fallen einige Unannehmlichkeiten auf:
1. Das Unterformular mit den Sportarten ist nicht direkt erreichbar. Wird mit dem Tabulator navigiert, so springt der Tabulator nach der Eingabe des Geschlechtes direkt zum nächsten Teilnehmerdatensatz.

2. Die Navigationsleiste zeigt, wenn sie im Unterformular steht, die Datensatzzahl des Unterformulars an. Sie sollte besser nur durch das Hauptformular navigieren.

3. Das Geschlecht wird zur Zeit auswendig je nach Benutzervorliebe eingegeben. In der Tabelle ist lediglich die Länge des Feldes auf 1 Zeichen begrenzt worden. Hier muss also eine sicherere Eingabe gewährleistet werden.

Zur Lösung dieser Probleme muss das Formular nicht für die Dateneingabe, sondern zum Bearbeiten geöffnet werden:

**Mit dem Tabulator zum Unterformular**

Um mit dem Tabulator nach der Eingabe in das Feld "Geschlecht" nicht gleich weiter zum nächsten Teilnehmer-Datensatz zu rutschen ist einmal die Bearbeitung der Aktivierungsreihenfolge, erreichbar über die Symbolleiste «Formular-Entwurf», notwendig.

Mit der Auswahl **Auto Sortierung** erfolgt nicht nur die Sortierung der angezeigten Steuerelemente, sondern auch die automatische Weiterleitung in das Unterformular. Das ist zwar aus dem Dialog nicht ersichtlich, wird aber im Hintergrund so geregelt.
Navigationsleiste des Hauptformulars auch im Unterformular aktivieren

Der Formularnavigator wird geöffnet. Die Eigenschaften des Unterformulars werden aufgesucht.


Eingabe eines Kontrollfeldes einschränken

Für die Einschränkung der Eingabe darf das Kontrollfeld kein einfaches Texteingabefeld sein. Im Hauptkapitel Formulare findet sich für die Eingabe des Geschlechtes die Lösung über einen «Gruppierungsrahmen». Hier soll deshalb eine andere Lösung, nämlich noch einmal über ein Listenfeld, vorgestellt werden.

Daten → Art des Listeninhalts bleibt auf dem Wert «Werteliste» stehen. In Daten → Listeninhalt werden die Kürzel 'm' und 'w' untereinander (erreicht durch Shift + Eingabetaste) eingegeben. Diese Kürzel sind die Daten, die an die Tabelle "Teilnehmer" weitergegeben werden.

In Allgemein → Listen-Einträge wird das angegeben, was durch das Listenfeld angezeigt werden soll. Das kann auch 'm' und 'w' sein. Es muss auf jeden Fall die gleiche Reihenfolge haben wie die Einträge unter Daten.


Damit ist das Eingabeformular so weit, dass die auffälligsten Unannehmlichkeiten beseitigt wurden. Das Formular und die Datenbankdatei müssen wieder abgespeichert werden. Die Eingabe von Teilnehmern und Teilnehmerinnen sowie die Zuordnung zu Sportarten kann beginnen.

Damit die folgenden Schritte sinnvoll sind sollten erst einmal einige Datensätze eingegeben werden. Dabei sollte darauf geachtet werden, dass die Teilnehmer und Teilnehmerinnen vom Alter und von der Sportart auch einmal in Konkurrenz zueinander treten können. Sonst sehen die anschließenden Abfragen und Berichte doch recht nichtssagend aus.
Abfrage

In einer Abfrage können z. B. Inhalte verschiedener Tabellen zusammengefasst werden. Mit dem ersten Zugriff sollen erst einmal alle Teilnehmer mit den Sportarten dargestellt werden, die sie belegt haben.

Der Start dazu geht über **Abfragen → Abfrage in der Entwurfsansicht erstellen** … . Es erscheint ein Dialog zur Tabellenauswahl, aus dem wir alle Tabellen für die Abfrage auswählen. Am übersichtlichsten ist es, wenn dabei die Tabelle "rel_Teilnehmer_Sportarten" als zweite Tabelle ausgewählt wird. Dann sind auch gleich die Verbindungen gut zu erkennen.

Die Felder, die in der Abfrage dargestellt werden sollen, können entweder durch einen Doppelklick auf die Feldnamen in den Tabellen oder durch Auswahl in der Zeile "Feld" eingestellt werden. Bei der Auswahl in der Zeile werden die Felder sämtlicher aufgeführter Tabellen aufgelistet. Um die Felder den Tabellen richtig zuordnen zu können werden sie in Abfragen nach "Tabellenname"."Feldname" bezeichnet. Wird statt des Feldnamen ein «*» eingesetzt, so bedeutet dies, dass alle Felder der entsprechenden Tabelle angezeigt werden sollen.

Eine Abfrage sollte grundsätzlich vor dem Abspeichern auch einmal ausgeführt werden um zu sehen, ob sie denn wirklich das erhoffte Ergebnis liefert. Hierzu entweder direkt den oben gezeigten Button anklicken. Es kann auch F5 gedrückt oder der Weg über **Bearbeiten → Abfrage ausführen** gewählt werden.
Die Abfrage zeigt jetzt alle Kombinationen der Teilnehmer und Sportarten an. Hat ein Teilnehmer mehrere Sportarten belegt, so werden mehrere Datensätze angezeigt. Teilnehmer ohne Sportarten erscheinen nicht.

Um bei einem Wettkampf unterschiedliche Altersgruppen zusammen zu stellen ist der Geburtsjahrgang entscheidend. Es kommt darauf an, wie alt eine Person in dem jeweiligen Jahr wird.

Der Geburtsjahrgang kann mit unterschiedlichen Funktionen ermittelt werden. Die folgende Eingabe zeigt die Variante, die für Personen mit Englischkenntnissen vielleicht auch so erschließbar ist:

\[
\text{YEAR(NOW()) - YEAR("Geburtstag")}
\]

\text{NOW()}

steht für «Jetzt», also den aktuellen Zeitpunkt. Mit \text{YEAR(NOW())} wird also das aktuelle Jahr ausgelesen. \text{YEAR("Geburtstag")} liest das Jahr aus dem Geburtstagsdatum aus. Aus beiden wird hier eine Differenz gebildet, die angibt, wie alt diese Person in dem aktuellen Jahr wird oder bereits geworden ist.

Diese und viele andere Funktionen, die mit der eingebauten HSQLDB zusammen arbeiten, sind im Anhang des Handbuches beschrieben.

Wird die Abfrage ausgeführt, so erscheint die entsprechende Berechnung für das Jahr 2014, da diese Abfrage eben in dem entsprechenden Jahr ausgeführt wurde.


<table>
<thead>
<tr>
<th>Sportart</th>
<th>Sportalter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kurzstrecken</td>
<td>29</td>
</tr>
<tr>
<td>Hochsprung</td>
<td>29</td>
</tr>
<tr>
<td>Kugelstoßer</td>
<td>29</td>
</tr>
<tr>
<td>Kurzstrecken</td>
<td>36</td>
</tr>
<tr>
<td>Weitsprung</td>
<td>36</td>
</tr>
</tbody>
</table>

Wird die Abfrage erneut ausgeführt, so erscheint in dem Spaltenkopf nicht mehr die Formel sondern die Bezeichnung «Sportalter».


Auf das Feld "Sportalter" soll nun zugegriffen werden. Das Sportalter bestimmt ja, in welcher Altersklasse die jeweiligen Personen starten.

Damit die Ermittlung nicht allzu kompliziert wird soll hier folgende Zuordnung nachvollzogen werden: Unter 20 Jahren werden die Teilnehmer immer in zwei Jahrgängen, beginnend mit 0, zusammen gefasst. Ab 20 Jahren werden Gruppen für jeweils 10 Jahre gebildet, also von 20-30 Jahren usw.


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

In die deutsche Sprache übertragen bedeutet das so viel wie:
Ist das Sportalter unter 19 Jahre, so nehme den Wert des Sportalters und subtrahiere davon den Rest, der sich ergibt, wenn das Sportalter durch 2 dividiert wird.


Dieser Formel wurde wieder ein Alias zugewiesen, in diesem Falle der Alias "Altersklasse".

Die Abfrage kann jetzt unter dem Namen "Meldung" gespeichert werden.

Im Code sind Felder und Tabellen in doppelte Anführungszeichen gesetzt und in einem ockerfarbenen Farbton dargestellt. Begriffe des SQL-Codes sind in blau gehalten, Funktionen der Datenbank in grün.

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

Im Code sind Felder und Tabellen in doppelte Anführungszeichen gesetzt und in einem ockerfarbenen Farbton dargestellt. Begriffe des SQL-Codes sind in blau gehalten, Funktionen der Datenbank in grün.

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

Der Teil mit der Formel wurde ja schon erwähnt. Hier jetzt der Rahmen.

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

Wähle aus der Tabelle "Sportalter" alle Datensätze aus und zusätzlich das, was durch die Formel bestimmt wird. Das, was durch die Formel bestimmt wird, bezeichne als "Altersklasse".


Sobald in einer Abfrage bei ausgeschalteter Design-Ansicht der Button SQL (für SQL-Kommando direkt ausführen) gedrückt wird, wird die Datenbank mit einer Fehlermeldung reagieren. Die eingebettete HSQLDB kennt die Abfrage "Sportalter" nicht, auf die sich die aktuellen Abfrage "Meldung" bezieht.

Ausdruck im Bericht

Über Berichte → Bericht in der Entwurfsansicht erstellen ... wird ein Bericht erstellt, der eine übersichtliche Teilnehmerliste, sortiert nach Sportart, Geschlecht und Altersklasse darstellen soll.

Wird der Editor gestartet, so ist zuerst der Dialog «Feld hinzufügen» im Vordergrund. Für diesen Dialog wird die von der alphabetischen Sortierung her erste Tabelle als Basis genommen. Es muss für den Bericht aber die Abfrage als Datenquelle herausgesucht werden.
Im Berichtsfenster befindet sich auf der rechten Seite die Übersicht über die Eigenschaften des gerade aktiven Objektes. Sollte die Leiste nicht sichtbar sein, so muss sie über Ansicht → Properties ausgewählt werden.


Aus den Eigenschaften wird im Reiter Daten → Art des Inhaltes → Abfrage ausgewählt. Als Inhalt wird dann die Abfrage "Meldung" gewählt, die als letzte zusammenfassende Abfrage erstellt wurde.

Der Report-Builder zeigt jetzt sämtliche Felder der ausgesuchten Abfrage an. Im nächsten Schritt wird der Sortier-Dialog ausgewählt:
Die Sortierung kann auch über Ansicht → Sorting and Grouping angezeigt werden.

Im Sortierungs- und Gruppierungs-Dialog werden nacheinander die Felder "Sportart", "Geschlecht" und "Altersklasse" ausgewählt.

Auf der linken Seite des Berichtes erscheint bei jeder Sortierungsauswahl ein Gruppenkopf für die Sortierung. Die Eigenschaften der Sortierung und Gruppierung werden einfach erst einmal so gelassen wie im Standard vorgesehen.

Der Dialog «Sortierung und Gruppierung» kann nun geschlossen werden.

Mit dem gleichen Verfahren werden die Felder «Geschlecht» und «Alter» den passenden Gruppenköpfen zugewiesen.

Sämtliche verbleibenden Felder werden in den Bereich «Detail» eingefügt.

Der Berichtsentwurf dürfte jetzt wie folgt aussehen:
Anschließend sollte der Bericht erst einmal abgespeichert werden. Als Name würde sich so etwas wie «Teilnehmerliste» anbieten.

Auch die Datenbankdatei selbst sollte abgespeichert werden, da sonst die Speicherung des Berichtes erst einmal nur temporär vorgenommen wird.

Gerade beim Gestalten eines Berichtes mit dem Report-Builder fallen oft Instabilitäten des Programms auf. Ist ein wichtiger Schritt gelungen so sollte auf jeden Fall sowohl der Bericht als auch die Datenbankdatei gespeichert werden. Die spätere Ausführung eines Berichtes ist von diesen Instabilitäten zum Glück nicht betroffen.

Wird dieser Bericht mit entsprechenden Daten ausgeführt, dann gibt sich in etwa das folgende Bild:
Der Beginn des Berichtes zeigt hier zwei männliche Teilnehmer an, die Hochsprung machen wollen und der Altersklasse 20 angehören.

Bei der Ausführung des Berichtes fallen einige Designmängel auf:

1. Die Abstände zwischen den Gruppenköpfen und dem Inhalt des Detailbereichs ist viel zu groß.
2. Das Geschlecht wird mit «m» bzw. «w» angegeben. Besser wäre eine Bezeichnung wie «Herren» und «Damen».

Abstände zwischen den Berichtsbereichen einstellen


Es ist nicht möglich, einen Bereich kleiner zu wählen als die darin existierenden Beschriftungs- und Textfelder sind.

Der Bereich «Altersklasse» sollte dabei so groß gewählt werden, dass er noch die Beschriftungsfelder aus dem Bereich «Detail» aufnehmen kann.


Einführung in Base: Eine einfache Datenbank – Testbeispiel im Detail
**Beeinflussung eines Textfeldinhaltes durch eine Formel**

Die Bezeichnung des Geschlechtes aus der Tabelle reicht für die Ansprüchen einer Teilnehmerliste nicht aus. Es könnte eine entsprechende Umbenennung bereits in der Abfrage erfolgen. Da die Abfrage aber bereits erstellt wurde, werden jetzt die Möglichkeiten des Berichtes genutzt.


In der Kategorie «Logical» steckt die Funktion «IF». Die Vorgehensweise bei dieser Funktion ist gleich der Funktion «WENN» aus LO-Calc.


Die Eingabe wird mit OK bestätigt.

Damit ist die Formulierung des Feldes entsprechend umgestellt.

**Umstellung der Formatierung eines Textfeldes**

Das Feld, das den Inhalt der Datenbank aufnehmen soll, wird zwar in Berichten als «Textfeld» bezeichnet. Es kann aber genau so formatiert werden wie Felder in Tabellen im Writer oder in Calc.


Die Formatierung der Anzeige der ist nun von Text auf Zahl umgestellt.
Verschieben von Feldern im Report-Builder

Felder können beim Report-Builder auch über die Grenzen eines Bereiches hinweg verschoben werden. Im Zielbereich muss allerdings genügend Platz für das Feld vorhanden sein. Es kann nämlich nicht ein Feld über einem anderen liegen.

Die Positionierung mit der Maus ist je nach System sogar so ungenau, dass am besten reichlich Platz vorhanden sein sollte. Sind erst einmal die Felder in einem Bereich wie hier dem Gruppenkopf «Altersklasse», so lassen sie sich auch mit den Pfeilfeldern und mit Tastatureingaben genau positionieren.


Hier wurde noch ein Beschriftungsfeld für die Überschrift hinzugefügt. Die Eingabe des Textes für das Beschriftungsfeld erfolgt in den Eigenschaften des Feldes.

Der Seitenfuß wurde in den Eigenschaften auf Sichtbar → Nein eingestellt. Er nimmt sonst am unteren Rand des Dokumentes zu viel Platz weg. Hierbei sollte bedacht werden, dass die verfügbare Höhe bereits durch die Größe der Seitenränder zusätzlich verringert wird. Standardmäßig sind alle Seitenränder bei einer DIN-A4-Seite auf 2 cm eingestellt.


Erweiterungen der Beispieldatenbank

Das vorgestellte Beispiel ist natürlich nur der erste Schritt für eine Datenbank im Sportbereich. Zuerst einmal sollte auf jeden Fall ein Feld ergänzt werden, das die jeweiligen Leistungen aufnimmt. Solch ein Feld wäre gut in der Tabelle "rel_Teilnehmer_Sportart" unter zu bringen.

Sollen die Eingaben allerdings für mehrere Wettkämpfe in gleichen Sportarten gelten, so ist außerdem noch ein Datumsfeld oder ein anderes Feld, das dem jeweiligen Wettkampf zugeordnet werden kann, in die Tabelle "rel_Teilnehmer_Sportart" aufzunehmen. Das Feld muss dann auch Teil des gemeinsamen Primärschlüssels sein.

Vielleicht käme noch die Vereinszugehörigkeit hinzu. Dazu würde ein Feld in der Tabelle "Teilnehmer" ausreichen. Bei vielen gleichen Vereinen bietet sich aber auch eine separate Tabelle "Vereine" sowie der entsprechende Fremdschlüssel in der Tabelle "Teilnehmer“ an.
Dann muss natürlich, wie bei allen Wettbewerben, ermittelt werden, wer denn nun in der jeweiligen Altersklasse auf welcher Platzierung gelandet ist. Sortierung ist hier gefragt, die vielleicht wieder als Bericht in einer Ergebnisliste landet.

Schön wäre dann, wenn jeder Teilnehmer (wieder über einen Bericht) eine schön gestaltete Urkunde mit den persönlichen Leistungen und der Platzierung erhält.

Solche Erweiterungen sind ohne weiteres möglich. Das sollen die kommenden Kapitel des Handbuches zeigen.

**Weiteres Einführungs- und Beispielmateriel**


Datenbank erstellen
Allgemeines bezüglich der Erstellung einer Datenbank


Neue Datenbank als interne Datenbank

Sofern nicht von vornherein eine Multiuserdatenbank geplant ist oder erst einmal erste Erfahrungen mit einer Datenbank gesammelt werden sollen, ist die interne Datenbank gut nutzbar. Diese lässt sich später auch mit ein paar Kniffen noch zur externen HSQLDB umwandeln, auf die dann auch mehrere User gleichzeitig zugreifen können, wenn der HSQLDB-Server läuft. Eine Beschreibung dazu erfolgt im Anhang dieses Handbuches im Kapitel «Datenbankverbindung zu einer externen HSQLDB».

Die Erstellung einer internen Datenbank erfolgt direkt über den Eingangsbildschirm von LO mit einem Klick auf den Button Datenbank oder über Datei → Neu → Datenbank. Der Datenbank-Assistent startet mit dem folgenden Bildschirm:

Das Optionsfeld «Neue Datenbank erstellen» ist für eine neue interne Datenbank angewählt. Standardmäßig ist dies die eingebettete HSQLDB. Ab LO 4.2 steht zusätzlich die interne Firebird-Datenbank zur Verfügung. Diese muss allerdings in LO 4.2 und 4.3 extra aktiviert werden.

Die weiteren Optionen dienen dazu, eine bestehende Datei zu öffnen oder eine Verbindung zu einer externen Datenbank wie z.B. einem Adressbuch, einer MySQL-Datenbank o.ä. aufzubauen.


Die Datenbank wird fertiggestellt, indem direkt noch einem Namen und einem Speicherort gefragt wird. An der entsprechenden Astelle wird dann die *.odb-Datei erzeugt, die zur Aufnahme von Daten aus der internen Datenbank, zur Speicherung von Abfragen, Formularen und Berichten gedacht ist.

Im Gegensatz zu den anderen Programmteilen von LibreOffice wird also die Datei abgespeichert, bevor der Nutzer überhaupt irgendwelche für ihn sichtbare Eingaben getätigt hat.

Mit der Version LibreOffice 4.2 wurde zusätzlich zu der internen HSQLDB eine interne Firebird-Datenbank aufgenommen. Um diese bisher noch nicht komplett integrierte Datenbank zu nutzen müssen über Extras → Optionen → LibreOffice → Erweitert → Optionale (instabile) Einstellungen die experimentellen Funktionen aktiviert werden:

Danach erscheint in dem Datenbankassistenten neben der internen HSQLDB auch die interne Firebird-Datenbank zur Auswahl.
Die eingebettete Firebird-Datenbank ist allerdings in der Version 4.2 noch weit entfernt von ihren tatsächlichen Möglichkeiten. SQL-Erweiterungen zu der HSQLDB sind noch nicht implementiert; auch die Erstellung automatisch hoch zählender Primärschlüssel klappt noch nicht.

Die HSQLDB ist auch in LibreOffice 4.2 und 4.3 die Standarddatenbank für jeden Nutzer; die Firebird-Datenbank steht aber bereits für erste Experimente bereit. Das Handbuch bezieht sich aus diesem Grunde vorerst weiter auf die interne HSQLDB. Für die interne Firebird-Datenbank ist zusätzlich ein Vergleich der Funktionen und Datentypen mit denen der internen HSQLDB zu dem Paket der Datenbank-Beispiele hinzugefügt worden.

Zugriff auf externe Datenbanken

Alle externen Datenbanken müssen zuerst einmal existieren. Angenommen es soll der Zugriff auf die Datenbank einer eigenen Homepage erfolgen. Dort muss zuerst einmal die Datenbank selbst mit einem Namen und Zugangsdaten gegründet worden sein, bevor externe Programme darauf zugreifen können.

Existiert nun diese externe Datenbank, so kann sie, je nach vorhandener Treibersoftware, zur Erstellung von Tabellen und anschließender Dateneingabe und -abfrage genutzt werden.

Über Datei → Neu → Datenbank wird der Datenbankassistent geöffnet und die Verbindung zu einer bestehenden Datenbank hergestellt. Die Liste der hier aufgeführten Treiber, die mit Datenbanken verbinden, variiert etwas je nach Betriebssystem und Benutzeroberfläche. Immer dabei sind aber

- dBase
- JDBC
Je nach gewählter Datenbank variieren nun die Verbindungseinstellungen. Diese können auch gegebenenfalls später noch korrigiert werden, wenn erst einmal die *.odb-Datei erstellt worden ist.


Hier soll nur kurz an ein paar Beispielen vorgestellt werden, wie der Kontakt zu anderen externen Datenbanken hergestellt wird.

MySQL/MariaDB-Datenbanken

MySQL-Datenbanken oder auch MariaDB-Datenbanken können auf drei verschiedene Weisen mit Base verbunden werden. Die einfachste und schnellste Art ist die direkte Verbindung mit dem MySQL-Connector. Daneben steht noch die Verbindung über JDBC und ODBC zur Verfügung.

Erstellen eines Nutzers und einer Datenbank

Nachdem MySQL bzw. MariaDB installiert ist sollten nacheinander die folgenden Schritte vollzogen werden:


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

Am Anfang ist kein Passwort vorhanden, so dass bei der Eingabeaufforderung auch nur Enter gedrückt wird.

Es erscheint die Eingabeaufforderung

```bash
mysql>
```

Alle folgenden Eingaben werden jetzt auf dieser mysql-Konsole erstellt. Die Passwörter können sich unterscheiden, je nachdem, ob die Anmeldung von dem momentanen Rechner («localhost») oder von einem anderen Rechner zum MySQL-Serverrechner («host») erfolgt.

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

Windows-Nutzer geben statt der zweiten Zeile

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

ein.

2. Als Sicherheitsmaßnahme werden alle eventuell bestehenden anonymen Nutzer gelöscht.

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


```sql
CREATE DATABASE libretest;
```

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

Damit ist die Datenbank vorhanden, mit der die folgenden Verbindungen aufgebaut werden.

**MySQL-Verbindung direkt mit der Extension**

Für die MySQL-Datenbank existieren Erweiterungen zur direkten Verbindung mit Base. Diese Erweiterungen tragen den Namen MySQL-Connector.

- Die mit der 3.3.4 funktionierende Erweiterung («Extension») ist unter http://extensions.services.openoffice.org/en/project/mysql_connector zu finden.

**Vorsicht**

Ein Kontakt zu LO 4.1.* kommt zwar mit dem AOO-Connector zustande, aber da sich ab der Version LO 4.1 die Beschreibung der Sekundenbruchteile von LO und AOO unterscheiden, werden falsche Werte im Bereich der Zeiten und auch von kompletten Timestamp-Feldern angezeigt.

Verwenden sie also bitte immer die Version, die lt. Beschreibung auch zur LO-Version passt. Achten sie besonders beim Wechsel auf LO 4.1 darauf, ob überhaupt ein entsprechender Connector zur Verfügung steht.

**MySQL-Verbindung über JDBC**


Gegebenenfalls kann das Java-Archiv auch separat über Extras → Optionen → LibreOffice → Erweitert → Java Optionen → Class-Pfad in den Class-Path aus jeder beliebigen Stelle der Festplatte übernommen werden. Für diese Option sind dann auch keine Systemverwalterrechte notwendig.

**MySQL-Verbindung über ODBC**

Für eine Verbindung über ODBC muss natürlich erst einmal die entsprechende ODBC-Software installiert sein. Details dazu werden hier nicht weiter beschrieben.

Nach Installation der entsprechenden Software kann es passieren, dass LO den Dienst verweigert, weil es die libodbc.so nicht findet. Hier existiert in den meisten Systemen inzwischen die lib-
odbc.so.2. Auf diese Datei muss ein entsprechender Verweis mit dem Namen libodbc.so in dem gleichen Verzeichnis abgespeichert werden.

In den für das System notwendigen Dateien odbcinst.ini und odbc.ini müssen Einträge ähnlich den folgenden existieren:

**odbcinst.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
```

Diese beiden Dateien liegen bei einem Linux-System im Verzeichnis `/etc/unixODBC`. Ohne den Eintrag zur Art des verwendeten Zeichensatzes kommt es bei Umlauten zu Problemen, auch wenn die Einstellungen in MySQL/MariaDB und Base übereinstimmen.

Details zu den Verbindungsparametern sind im *MySQL-Referenzhandbuch* zu finden.

**Verbindung zur MySQL-Datenbank über den Datenbank-Assistenten**

Der Zugriff auf eine existierende MySQL-Datenbank erfolgt mit der direkten Verbindung in den folgenden Schritten:

Hier wird aus den, teilweise betriebssystemspezifischen, Datenbanken die MySQL-Variante ausgewählt.
Die direkte Verbindung

Der Kontakt zu MySQL kann über ODBC oder JDBC hergestellt werden, solange kein nativer MySQL-Connector als direkte Verbindung installiert ist oder unterstützt wird. Ist die Erweiterung für eine direkte Verbindung installiert, so wird «Direkte Verbindung» standardmäßig voreingestellt.

Die direkte Verbindung ist diejenige, die von der Geschwindigkeit und von der Funktionalität her am besten gewählt werden sollte.

Der Datenbankname muss bekannt sein. Befindet sich der Server auf dem gleichen Rechner wie die Benutzeroberfläche, unter der die Datenbank erstellt wird, so kann als Server «localhost» gewählt werden. Ansonsten kann die IP-Adresse (z.B. 192.168.0.1) oder auch je nach Netzwerkstruktur der Rechnername oder gar eine Internetadresse angegeben werden. Es ist also ohne weiteres möglich, mit Base auf die Datenbank zuzugreifen, die vielleicht auf der eigenen Homepage bei irgendeinem Provider liegt.

3 «nativ» – unverändert, unmodifiziert
Bei der Arbeit mit Base über das Internet sollte sich der Nutzer allerdings darüber bewusst sein, wie seine Verbindung zu der Datenbank gestaltet ist. Gibt es eine verschlüsselte Verbindung? Wie erfolgt die Passwortübertragung?

Jede über das Netz erreichbare Datenbank sollte mit einem Benutzernamen und einem Kennwort geschützt sein. Hier wird direkt getestet, ob die Verbindung klappt. Wichtig ist natürlich, dass der entsprechende Benutzer in MySQL bzw. der MariaDB entsprechend für den benannten Server eingerichtet wurde.

Dieses Fenster erscheint anschließend bei jedem Start der Datenbankdatei, wenn das erste Mal auf die MySQL-Datenbank zugegriffen wird.

Der Verbindungstest startet die Authentifizierung mit vorgegebenem Benutzernamen. Nach Passworteingabe erfolgt die Meldung, ob die Verbindung erfolgreich hergestellt werden kann. Läuft z.B. MySQL zur Zeit nicht, so kommt natürlich an dieser Stelle eine Fehlermeldung.
Auch hier wird die Datenbank nicht angemeldet, da sie nur zu ersten Tests aufgebaut wurde. Eine Anmeldung ist erst notwendig, wenn andere Programme wie z.B. der Writer für einen Serienbrief auf die Daten zugreifen sollen.

Der Assistent beendet die Verbindungserstellung mit dem Abspeichern der gewünschten Datenbankverbindung. In dieser *.odb-Datei liegen jetzt lediglich diese Verbindungsinformationen, die bei jedem Datenbankstart ausgelesen werden, so dass auf die Tabellen der MySQL-Datenbank zugegriffen werden kann.
Ansicht der geöffneten Datenbankdatei mit Tabellenübersicht und in der Fußzeile der Benennung des verwendeten Treibers «MySQL (nativ)», des Datenbanknamens «libretest», des Nutzers der Datenbank «lotest» und des Servers, auf dem die Datenbank läuft, nämlich «localhost».

Über «Fertigstellen» wird die Base-Datei erstellt und die Ansicht auf die Tabellen der MySQL-Datenbank geöffnet. Die Tabellen der Datenbank werden unter dem Namen der Datenbank selbst aufgeführt.

Beim manchen Treibern wird nur die Datenbank «libretest» angezeigt, für die auch die Verbindung bestimmt war. Andere Treiber von LO bieten auch andere MySQL- bzw. MariaDB-Datenbanken auf dem gleichen Server zur Auswahl.

Auch mit den Treibern für nur eine Datenbank ist aber ein Zugriff auf die anderen Tabellen z.B. für Abfragen möglich, sofern natürlich der angegebene Datenbanknutzer, im obigen Fall also «lotest», mit seinem Passwort auf die Daten zugreifen kann. Im Gegensatz zu den bisherigen nativen LO-Treibern ist hier allerdings kein schreibender Zugriff auf andere Datenbanken des gleichen MySQL-Datenbankservers möglich.

Im Unterschied zu der internen Datenbank von Base taucht bei Abfragen entsprechend in MySQL für die Definition der Tabelle immer auch der Datenbankname auf, hier z.B.

```
... FROM "test"."Klasse" AS "Klasse" ...
```

Hier ist es also auf jeden Fall notwendig, der Kombination aus Datenbankname und Tabellenname mit dem Zusatz «AS» einen alternativen Alias-Namen zuzuweisen. Genaueres siehe dazu in dem Kapitel «Verwendung eines Alias in Abfragen».

Die ODBC-Verbindung

Die ersten Schritte zur ODBC-Verbindung sind gleich denen zur direkten Verbindung. Wird beim zweiten Schritt dann die ODBC-Verbindung für MySQL gewählt, dann erscheint das folgende Fenster des Datenbank-Assistenten:


Datenbank erstellen: Zugriff auf externe Datenbanken
Beim Test erscheint der Name aus der «odbc.ini». Auch wenn hier wieder nur mit einer Datenbank verknüpft wird, können andere Tabellen des MySQL-Servers sehr wohl gelesen werden.

Schritt 4 und 5 laufen wieder identisch zur Direktverbindung ab.

**Die JDBC-Verbindung**

Auch bei der JDBC-Verbindung sind die ersten Schritte gleich hier unterscheidet sich wieder nur Schritt 3 von den anderen Schritten des Assistenten:

Der Assistent fragt hier die gleichen Informationen ab wie bei der direkten Verbindung. Der Datenbankname ist der, der auch in MySQL selbst verwandt wird.


Alle weiteren Schritte sind wieder identisch zu denen der vorherigen Verbindungen. Die Verbindungen zu anderen Datenbanken des gleichen MySQL-Datenbankservers funktionieren ebenfalls nur lesend.
**PostgreSQL**

Für die PostgreSQL-Datenbank gibt es einen direkten Treiber von LO, der von vornherein mit installiert wird. Damit der Kontakt auch sicher passt zuerst aber eine kurze Anleitung zu den ersten Schritten nach der Installation von PostgreSQL.

### Erstellen eines Nutzers und einer Datenbank

Die folgenden Schritte sind nach einer Installation über den Paketmanager in OpenSUSE erforderlich. Sie ähneln vermutlich denen unter anderen Betriebssystemen.

2. Der Postgre-Server muss vom Administrator gestartet werden, z. B. `service postgresql start` oder `rcpostgresql start`.
3. Der Nutzer «postgres» muss sich mit `su postgres` auf der Konsole anmelden.
7. Der Systemnutzer «postgres» erstellt die Datenbank «libretest»:

   ```sql
   CREATE DATABASE libretest;
   ```

### Direkte Verbindung zu Base


### Datenbank-Assistent

**Schritte**

1. **Datenbank auswählen**
2. **Verbindungseinstellung an**
3. **Benutzer-Authentifizierung einrichten**
4. **Fertigstellen und fortfahren**

**URL der Datenquelle**

`dbname=libretest host=localhost`

Die Benutzer-Authentifizierung sieht genauso aus wie die unter MySQL.


Hinweis

Werden Tabellen z.B. aus der internen HSQLDB nach PostgreSQL kopiert, so schlägt der Importassistent den einfachen Tabellennamen aus der HSQLDB, z.B. «Tabelle1», vor. Der Import mit diesem Namen schlägt allerdings fehl. Stattdessen muss vor den Namen die Schemabezeichnung ergänzt werden: aus «Tabelle1» wird «public.Tabelle1».

In der Tabellenübersicht von PostgreSQL erscheinen die unterschiedlichen Schemata. Im Schema «public» ist die abgespeicherte Tabelle «Name» zu sehen.

dBase-Datenbanken


Hinweis

Zur Zeit werden nur dBase-Dateien berücksichtigt, die mit der in Kleinbuchstaben geschriebene Dateiendung *.dbf versehen sind. Andere Endungen wie z.B. *.DBF werden nicht ausgelesen. (Bug 46180)

Datenbank-Assistent

Schritte
1. Datenbank auswählen
2. Verbindung zu dBASE-Dateien einrichten
3. Fertigstellen und fortfahren

Verbindung zu dBASE-Dateien einrichten

Bitte wählen Sie den Ordner, in dem sich die dBASE-Dateien befinden.

Erfad zu den dBASE Dateien

Durchsuchen
Die Verbindung wird zu einem Verzeichnis hergestellt. Alle *.dbf-Dateien, die sich in diesem Verzeichnis befinden, werden anschließend angezeigt und können über Abfragen oder Formulare miteinander verbunden werden.

Tabellen in dBase haben kein unverwechselbares Feld für jeden Datensatz («Primärschlüssel»). Sie können also vom Prinzip her so beschrieben werden wie Tabellenblätter in Calc.

Tabellen können in Base neu erstellt werden. Sie werden dann einfach als neue Dateien in das angegebene Verzeichnis geschrieben.

Für neue Tabellen stehen deutlich weniger Feldtypen zur Verfügung als für die interne HSQLDB. Dabei sind noch einige Bezeichnungen der folgenden Abbildung als Verdoppelungen anzusehen.

---

Datenbank erstellen: Zugriff auf externe Datenbanken
Dbase bietet sich besonders an, wenn es um Weitergabe und vielfältige Verarbeitungsmöglichkeit von Daten geht. Schließlich können auch Tabellenkalkulationen dBase-Tabellen direkt einlesen. 

Base übernimmt einfach die Codierung, die dem Betriebssystem entspricht. Alte dBase-Dateien weisen dadurch leicht Fehler beim Import von Sonderzeichen auf. Der Zeichensatz kann anschließend über Bearbeiten → Datenbank → Eigenschaften → Zusätzliche Einstellungen entsprechend korrigiert werden:

Datenbankegenschaften

[Diagramm]

Hinweis

Der Importassistent für dBase hat Probleme, numerische Feldtypen und Ja/Nein-Felder automatisch richtig zu erkennen. (Bug 53027) Hier muss entsprechend nachgebessert werden.

Tabellendokumente

Tabellenquellen für Datenbanken können auch Tabellendokumente sein. Wird aber eine Calc-Tabelle als Grundlage genommen, so ist jedes Editieren der Tabelle ausgeschlossen. Selbst wenn das Calc-Dokument auch noch geöffnet wird, wird dort ein Schreibschutz gesetzt.
Die einzige Abfrage ist letztlich, an welcher Position das Tabellendokument liegt und ob die Tabelle passwortgeschützt ist. Base öffnet dann das Tabellendokument und übernimmt alle Tabellenblätter mit ihrem Inhalt. Aus der ersten Zeile bildet Base die Feldbezeichnungen, aus den Tabellenblattbezeichnungen die Tabellenbezeichnungen.

Beziehungen zwischen den Tabellen lassen sich in Base nicht erstellen, da Calc nicht Grundlage für eine relationale Datenbank ist. Auch lassen sich Abfragen nicht über eine Tabelle hinaus erstellen.

Thunderbird Adressbuch

Die Verbindung zu einem Adressbuch wie z.B. dem Thunderbird-Adressbuch sucht der Assistent automatisch. Er fragt lediglich nach einem Speicherort für die *.odb-Datei, die er erstellen soll.
Alle Tabellen werden dargestellt. Wie in Calc sind die Tabellen aber nicht editierbar. Base macht die Adressen damit nur für Abfragen und z. B. für die Verwendung in Serienbriefen zugänglich.

Hinweis
Seit der Version LO 4.* wird unter Linux und MacOSX nur das Adressbuch «Persönliches Adressbuch» als «AddressBook» ausgelesen. Dadurch werden die Daten, die standardmäßig im Adressbuch «Gesammelte Adressen» liegen, nicht zugänglich. (Bug 64741)

Texttabellen
In Base gibt es die Möglichkeit, eine komplette Datenbank mit Zugriff auf Texttabellen zu erstellen. Gleichzeitig können auch Texttabellen innerhalb einer internen Datenbank angesprochen werden.

Texttabellen innerhalb einer internen HSQLDB-Datenbank

Soll zum Beispiel der Inhalt eines Adressbuchs, das nicht direkt von den Treibern von Base unterstützt wird, eingelesen werden, so kann dies entweder über einen Import einer solchen *.csv-Datei erfolgen (hierzü ist gegebenenfalls ein Zwischenschritt über Calc notwendig) oder die Datei wird direkt als Texttabelle in die Datenbank eingefügt. Um dort bearbeitet werden zu können benötigt die *.csv-Datei allerdings ein Feld, das als eindeutiges Feld («Primärschlüssel») erkennbar ist.
Der Zugriff zu einer Texttabelle kann nicht über die grafische Benutzeroberfläche erstellt werden. Stattdessen wird über Extras → SQL eine Texttabelle erstellt. Die Felder in der Texttabelle müssen dem entsprechen, was die Texttabelle in der entsprechenden Reihenfolge liefert. Das Feld "ID" muss z. B. positive Ganzzahlen enthalten, das Feld "Geburtsdatum" z. B. ein Datum in der Schreibweise Jahr – Monat – Tag.

Die Tabelle ist nicht direkt auf der Benutzeroberfläche sichtbar. Falls jetzt noch eine Ergänzung erfolgen soll muss über Ansicht → Tabellen aktualisieren die Tabelle verfügbar gemacht werden. Das Symbol der Tabelle macht deutlich, dass es sich nicht um eine «normale» Tabelle der Datenbank handelt.

Diesem Handbuch liegt die Datenbank «Beispiel_CSV_Einbindung.odb» bei.
Die Tabelle wurde zum Bearbeiten geöffnet und das Primärschlüsselfeld auf einen automatisch hoch zählendes Feld umgestellt.

Über Extras → SQL muss jetzt noch die Verbindung zur externen Texttabelle erstellt werden. Diese Texttabelle liegt im gleichen Verzeichnis wie die Datenbankdatei selbst.

```
SET TABLE "Adressen" SOURCE "Adressen.csv;encoding=UTF-8";
```

Anschließend steht die Texttabelle ganz normal zur Eingabe zur Verfügung. Allerdings ist hierbei Vorsicht geboten:

- Die Texttabelle kann gleichzeitig auch von externen Textprogrammen oder Tabellenkalkulationen geöffnet und bearbeitet werden. Datenverlust ist dann nicht auszuschließen.
- Änderungen an bereits geschriebenen Datensätzen führen dazu, dass die entsprechende Zeile in der Originaldatei geleert wird und in der neuen Fassung an den Schluss der Tabelle angefügt wird. Die obige Ansicht präsentiert z.B. sauber 4 beschriftete Zeilen mit richtig sortierter ID. Die Originaldatei zeigt hingegen nach einer Änderung im 2. Datensatz die folgende Reihenfolge, geordnet nach der ID: 1, Leerzeile, 3, 4, 2

Für die Verbindung zur Textdatei stehen verschiedene Parameter zur Verfügung.

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

Mit «ignore_first=true» wird die erste Zeile der Datei nicht eingelesen. Dies ist sinnvoll, wenn es sich dabei um die Feldbezeichnungen handelt. Der interne Standardwert der HSQLDB steht hier auf «false».

Standardmäßig werden von der HSQLDB Texte nur in doppelte Anführungszeichen gesetzt, wenn sie ein Komma enthalten, da das Komma die Standardtrennung der Felder ist. Soll jeder Text in doppelte Anführungszeichen gefasst werden, so ist «all_quoted=true» zu setzen.


Mit

```
SET TABLE "Adressen" READONLY TRUE;
```

wird schließlich davor geschützt, dass in die Tabelle überhaupt etwas geschrieben wird. So steht dann die Tabelle als normale Adressabelle wie eine Tabelle aus Adressbüchern z.B. von Mailprogrammen schreibgeschützt zur Verfügung. Die gesonderte Erstellung eines Schreibschutzes ist allerdings nur notwendig, wenn für die Tabelle erst einmal ein Primärschlüssel erstellt wurde.

---

Texttabellen als Grundlage für eine eigenständige Datenbankdatei

Wie im vorhergehenden Beispiel werden als Datengrundlage in dem folgenden Beispiel *.csv-Dateien genutzt. Über Base wird ein Verzeichnis, in dem die *.csv-Dateien liegen als Datenverzeichnis eingebunden.


Bereits bei der Auswahl ist schon deutlich beschrieben: Auf die Tabellen kann nur lesend zugegriffen werden, nicht schreibend.

Wird eine solche Datenbank genutzt, um einmal kurz in einer *.csv-Datei nach entsprechenden Datensätzen zu suchen oder um eine *.csv-Datei in eine andere Datenbankdatei über die Kopierfunktion einzufügen, so erfüllt diese Text-Datenbank ihren Sinn. Die entsprechende *.csv-Datei wird nur in das definierte Verzeichnis geschoben und kann dann direkt durchsucht oder kopiert werden. Für weitergehende Datenbankaufgaben ist diese Text-Datenbank wohl nicht gedacht.

**Firebird**

Die etwas in die Jahre gekommene alte HSQLDB-Version der internen Datenbank soll durch eine interne Firebird-Datenbank ersetzt werden. Zur Version LO 4.2.* ist die interne Firebird-Datenbank erst einmal nur als«experimentelle Funktion» zusätzlich wählbar. Das liegt einfach daran, dass längst noch nicht alle Funktionen richtig integriert wurden. Um aber zu sehen, was Firebird leisten kann, hier jetzt also die Verbindung zu einer externen Firebird-Datenbank zu testen.

Da die Dokumentation dazu nicht ganz so umfangreich ist wie z.B. zu MySQL oder PostgreSQL hier erst einmal die wichtigsten Schritte bei der Installation.

**Erstellen eines Nutzers und einer Datenbank**

Unter Linux stehen für Firebird entsprechende Pakete in der Paketauswahl bereit. Hier muss nach der Installation eigentlich nur noch der Server eingeschaltet werden. Die folgenden Schritte haben unter OpenSUSE 12.3 zu einer funktionierenden Datenbank mit Firebird geführt:

2. Eine Passwortänderung geht im Terminal über
gsec -user sysdba -pass masterkey -mo sysdba -pw neuespasswort
4. Der Nutzer «firebird» muss sich mit
su firebird
auf der Konsole anmelden.
5. Ein neuer Nutzer sollte gegründet werden - hier gezeigt mit altem Standardpassword des sysdba:
gsec -user sysdba -pass masterkey -add lotest -pw libre
Damit ist ein neuer Nutzer mit dem Namen lotest und dem Passwort libre erstellt.
isql-fb
Es erscheint die folgende Meldung:
Use CONNECT or CREATE DATABASE to specify a database
direkt gefolgt von der Eingabeaufforderung SQL>. Hier entsprechend passende Einträge:
SQL> CREATE DATABASE 'libretest.fdb'
CON> user 'lotest' password 'libre';


**Firebird-Verbindung über JDBC**

Aus der *.zip-Datei muss nur das Archiv jaybird-full-2.2.8.jar (oder aktuellere Version) kopiert und entweder in den Java-Pfad der Installation gelegt oder als Archiv direkt in LO eingebunden werden. Siehe dazu auch die entsprechende Einbindung bei MySQL.

Für die Angaben bei der JDBC-Installation sind folgende Parameter wichtig:
JDBC URL jdbc:firebirdsql://host[(:port)]/<path_or_alias>
Treiberbezeichnung: org.firebirdsql.jdbc.FBDriver
In obigen Beispiel wurde dies zu der URL
jdbc:firebirdsql://localhost/libretest.fdb?charSet=UTF-8
Ohne Zeichensatzangabe erscheint die folgende Fehlermeldung:

Nachträgliche Änderungen z.B. der Feldeigenschaften von Feldern einer Tabelle sind nicht möglich, wohl aber die Erweiterung der Tabelle oder das Löschen der Felder.

**Firebird-Verbindung über ODBC**


In den für das System notwendigen Dateien odbcinst.ini und odbc.ini müssen Einträge ähnlich den folgenden existieren:

**odbcinst.ini**

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

**odbc.ini**

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

Diese beiden Dateien liegen bei einem Linux-System im Verzeichnis `/etc/ unixODBC`. Die Variable SensitiveIdentifier muss auf jeden Fall auf «Yes» stehen, damit die Eingabe in Tabellen funktioniert, deren Namen und Feldbezeichnungen nicht in Großbuchstaben angegeben sind.

Zur Erstellung der Verbindung von LO mit der entsprechenden Datenquelle siehe [Die ODBC-Verbindung](#).


**Nachträgliche Bearbeitung der Verbindungseigenschaften**

Vor allem bei Verbindungen zu externen Datenbanken kann es hin und wieder vorkommen, dass eine Verbindung nicht wie gewünscht einwandfrei funktioniert. Mal ist der Zeichensatz nicht korrekt, ein anderes Mal funktionieren Unterformulare nicht einwandfrei oder es ist einfach etwas an den grundsätzlichen Verbindungseigenschaften zur Datenbank geändert worden.

Die folgenden Screenshots zeigen die Möglichkeit auf, wie z.B. die Verbindungsparameter zu einer externen PostgreSQL-Datenbank im Nachhinein geändert werden können.

---

*Datenbank erstellen: Nachträgliche Bearbeitung der Verbindungseigenschaften* 85

Falls sich der Name der Datenquelle geändert hat, kann dies hier geändert werden. Bei einer Verbindung über ODBC wird ja der Name, mit dem die Datenbank aufgerufen werden kann, in der odbc.ini festgelegt. Der Name ist in der Regel nicht gleich dem eigentlichen Datenbanknamen in PostgreSQL.

---

**Datenbank erstellen: Nachträgliche Bearbeitung der Verbindungseigenschaften** 86

Auch eine zusätzliche spezielle Einstellung des Treibers ist möglich, wenn eventuell ein Parameter ausgeführt werden soll, der zur Zeit nicht bereits in der odbc.ini steht.

Wird die Verbindungsart ausgewählt, so kann der gesamte Kontakt zur Datenquelle geändert werden. Die darauf folgenden Schritte sind gleich denen des Datenbankassistenten ab Schritt 2. So kann z.B. von einer ODBC- auf eine JDBC- oder gar eine direkte Verbindung über die internen Treiber von LO gewechselt werden. Dies ist vor allem dann sinnvoll, wenn zuerst einmal ausgetestet werden soll, welche Verbindungsart zu dem eigenen Projekt am besten passt.
Je nach Datenbanksystem gibt es unterschiedliche Befehle um automatisch hoch zählende Werte zu erzeugen. Soll so etwas erreicht werden, was der Treiber zur Zeit nicht direkt ermöglicht, dann ist an dieser Stelle Handarbeit angesagt. Dabei ist sowohl ein Befehl für das Erstellen eines Auto-Wert-Feldes als auch ein Befehl zur Abfrage des zuletzt erstellten Wertes notwendig.

Die über Bearbeiten → Datenbank → Erweiterte Einstellungen erreichbaren besonderen Einstellungen sind beeinflussen das Zusammenspiel von externer Datenbank und Base auf die unterschiedliche Art und Weise. Manche Einstellungen sind bereits ausgegraut, da sie für die zugrundeliegende Datenbank nicht änderbar sind. In dem obigen Beispiel wurde Benannte Parameter
durch '?' ersetzen ausgewählt. Es hatte sich gezeigt, dass sonst bei PostgreSQL die Weitergabe von Werten von einem Hauptformular zum Unterformular nicht funktionierte. Erst mit dieser Einstellung funktionierten die weitergehenden Formularkonstruktionen aus dem Kapitel «Formulare» dieses Handbuches korrekt.
Tabellen
Allgemeines zu Tabellen

Daten werden in Datenbanken innerhalb von Tabellen gespeichert. Wesentlicher Unterschied zu Tabellen innerhalb einer einfachen Tabellenkalkulation ist, dass die Felder, in die geschrieben wird, klar vordefiniert werden müssen. Eine Datenbank erwartet innerhalb einer Textspalte keine Zahleneingaben, mit denen sie rechnen kann. Sie stellt die Zahlen dann zwar dar, geht aber von einem Wert «0» für diese Zahlen aus. Auch Bilder lassen sich nicht in jeder Feldform ablegen.

Welche Datentypen es im einzelnen gibt, kann bei der grafischen Benutzeroberfläche dem Tabelleneditor entnommen werden. Details dafür im Anhang dieses Handbuches.

Einfache Datenbanken beruhen lediglich auf einer Tabelle. Hier werden alle Daten unabhängig davon eingegeben, ob eventuell mehrfache Eingaben des gleichen Inhaltes gemacht werden müssen. Eine einfache Adressensammlung für den Privatgebrauch lässt sich so erstellen. Die Adressensammlung einer Schule oder eines Sportvereins würde aber so viele Wiederholungen bei Postleitzahl und Ort aufweisen, dass diese Tabellenfelder in eine oder sogar 2 separate Tabellen ausgelagert würden. Die Auslagerung von Informationen in andere Tabellen hilft:

- laufend wiederkehrende Eingaben gleichen Inhaltes zu reduzieren
- Schreibfehler bei diesen laufenden Eingaben zu vermeiden
- Daten besser nach den ausgelagerten Tabellen zu filtern

Bei der Erstellung der Tabellen sollte also immer wieder überlegt werden, ob eventuell viele Wiederholungen vor allem von Texten oder Bildern (hier stecken die Speicherräuber) in den Tabellen vorkommen. Dann empfiehlt sich eine Auslagerung der Tabelle. Wie dies prinzipiell geht ist in der Einführung im Kapitel «Eine einfache Datenbank – Testbeispiel im Detail» beschrieben.

Beziehungen von Tabellen


Beziehungen zwischen Tabellen allgemein

Nur in Ausnahmefällen wird ein Primärschlüssel auch aus mehreren Feldern zusammen gebildet. Dann müssen diese Felder zusammen einzigartig sein.

Tabelle 2 kann ein Feld besitzen, das auf die Inhalte von Tabelle 1 hinweist. Hier wird der Primärschlüssel aus Tabelle 1 als Wert in das Feld der Tabelle 2 geschrieben. Tabelle 2 hat jetzt ein Feld, das auf ein fremdes Schlüsselfeld verweist, also einen Fremdschlüssel. Dieser Fremdschlüssel existiert in Tabelle 2 neben dem Primärschlüssel.

Je mehr Tabellen in Beziehung zueinander stehen, desto komplexer kann der Entwurf sein. Das folgende Bild zeigt die gesamte Tabellenstruktur der Beispieldatenbank in einer Übersicht, die von der Größe her die Seite dieses Dokuments sprengt:

Abbildung 1: Beziehungsentwurf der Beispieldatenbank «Medien_ohne_Makros”

Eins-zu-Viele Beziehungen:
Viele-zu-Viele Beziehungen:


Praktisch wird also die n:m-Beziehung über zwei 1:n-Beziehungen gelöst. In der Mitteltabelle kann die "Medien_ID" mehrmals erscheinen, ebenso die "Verfasser_ID". Dadurch, dass beide zusammen den Primärschlüssel ergeben ist nur ausgeschlossen, dass zu einem Medium wiederholt der gleiche Verfasser gewählt wird.

Eins-zu-Eins-Beziehung:

Die bereits angesprochene Bibliotheks-Datenbank enthält eine Tabelle für die Leser. In dieser Tabelle sind erst einmal nur die direkt notwendig erscheinenden Felder vorgesehen. Für eine


Eine 1:1-Beziehung bedeutet nicht, dass automatisch zu jedem Datensatz der einen Tabelle auch ein Datensatz der anderen Tabelle existiert. Es existiert allerdings höchstens ein Datensatz. Durch die 1:1-Beziehung werden also Felder ausgelagert, die vermutlich nur bei einem Teil der Datensätze mit Inhalt gefüllt sein werden.

Tabellen und Beziehungen der Beispieldatenbank

Die Beispieldatenbank muss drei verschiedene Aufgabenbereiche erfüllen. Zuerst einmal müssen Medien in die Datenbank aufgenommen werden. Dies soll so erfolgen, dass auch eine Bibliothek damit arbeiten kann.

Tabellen Medienaufnahme

Zentrale Tabelle der Medienaufnahme ist die Tabelle Medien. In dieser Tabelle werden alle Felder direkt verwaltet, die vermutlich nicht auch von anderen Medien mit dem gleichen Inhalt belegt werden. Doppelungen sollen also vermieden werden.


Die Tabelle Untertitel soll dazu dienen, z.B. den Inhalt von CDs im Einzelnen aufzunehmen. Da auf einer CD mehrere Musikstücke (Untertitel) vorhanden sind würde eine Aufnahme der Musikstücke in die Haupttabelle dazu führen, dass entweder viele zusätzliche Felder (Untertitel 1, Untertitel 2 usw.) erstellt werden müssten oder das gleiche Medium mehrmals hintereinander eingegeben werden müsste. Die Tabelle Untertitel steht also in einer n:1-Beziehung zu der Tabelle Medien.

Felder der Tabelle Untertitel sind neben dem Untertitel selbst die Nummerierung der Reihenfolge der Titel und die Dauer der Untertitel. Das Feld Dauer ist erst einmal als ein Zeitfeld vorgesehen. So kann gegebenenfalls die Gesamtdauer der CD in einer Übersicht berechnet und ausgegeben werden.


Kategorie, Medienart, Ort und Verlag haben jeweils eine 1:n-Beziehung.

In der Kategorie kann bei kleinen Bibliotheken so etwas stehen wie «Kunst», «Biologie» ... Bei größeren Bibliotheken gibt es gängige Systematiken wie z.B. die ASB (allgemeine Systematik für...

Die **Medienart** ist gekoppelt mit der **Ausleihzeit**. So kann es z.B. sein, dass Video-DVDs grundsätzlich nur eine Ausleihzeit von 1 Woche haben, Bücher aber eine von 3 Wochen. Wird die Ausleihzeit an ein anderes Kriterium gekoppelt, so muss entsprechend anders verfahren werden.

Die Tabelle **Ort** dient nicht nur dazu, die Ortsbezeichnungen aus den Medien aufzunehmen. In ihr werden gleichzeitig die Orte gespeichert, die für die Adressen der Nutzer Bedeutung haben.

Da der **Verlag** vermutlich auch häufiger vorkommt, ist für seine Eingabe ebenfalls eine gesonderte Tabelle vorgesehen.

---

**Abbildung 4: Medienaufnahme**

Die Tabelle Medien hat so insgesamt vier Fremdschlüssel und einen Primärschlüssel, der für 2 Tabellen der Abbildung *Medienaufnahme* zum Fremdschlüssel wird.
Tabellen Ausleihe

Abbildung 5: Ausleihe

Zentrale Tabelle ist die **Ausleihe**. In ihr werden die Tabellen Medien und Leser verknüpft. Da auch später noch nachvollzogen werden soll, wer ein Buch ausgeliehen hat (falls z.B. jemand beim Ausleihen bemerkt, dass das Buch beschädigt ist, oder falls eine Hitliste der Medien erstellt werden soll) wird der Datensatz in der Ausleihe bei der Rückgabe nicht einfach gelöscht. Vielmehr wird ein Rückgabedatum (*Rueck_Datum*) vermerkt.

Ebenso in die Ausleihe integriert ist das Mahnverfahren. Um die Anzahl der Mahnungen zu erfassen wird jede Mahnung separat in der Tabelle **Mahnung** eingetragen.


Schließlich ist noch der **Leser** mit der Ausleihe in Verbindung zu bringen. In der eigentlichen Lesertabelle wird lediglich der Name, eine eventuelle Sperrung und ein Fremdschlüssel für eine Verbindung zur Tabelle Geschlecht vorgesehen.

Tabellen Nutzerverwaltung

In dieser Tabellenkonstruktion werden gleich zwei Szenarien bedient. Der obere Tabellenstrang ist dabei auf Schulen zugeschnitten. Hier werden keine Adressen benötigt, da die Schüler und Schülerinnen über die Schule selbst ansprechbar sind. Mahnungen müssen nicht postalisch zugestellt werden, sondern auf dem internen Wege weitergegeben werden.

Der Adressstrang ist dagegen bei öffentlichen Bibliotheken notwendig. Hier müssen sämtliche Daten erfasst werden, die zu Erstellung eines Mahnbriefes erforderlich sind.

Die Tabelle `rel_Leser_Schulkasse` steht wie die Tabelle Adresse in einer 1:1-Beziehung zu der Tabelle "Leser". Dies ist gewählt worden, weil entweder die eine oder die andere Möglichkeit beschritten werden soll. Sonst könnte die "Schulklasse_ID" direkt in der Tabelle Schuler existieren; gleiches gilt für den gesamten Inhalt der Tabelle Adresse.


Wie eine derartige Tabellenkonstruktion später sinnvoll zu befüllen ist, wird weiter unten im Kapitel Formulare erläutert.

**Erstellung von Tabellen**


Bezeichnungen bei Tabellen:
Die obige Skizze zeigt die allgemein übliche Aufteilung von Tabellen in Spalten und Zeilen. Die entsprechenden Datenbankbezeichnungen sind in Klammern hinzugefügt.


In der grafischen Benutzeroberfläche von Base sind die Begriffe einer Spalte etwas anders umschrieben:


**Hinweis**

Defaultwert: Der Begriff «Defaultwert» in der GUI entspricht nicht dem, was Datenbanknutzer unter Defaultwert verstehen. Die GUI gibt hier einen bestimmten Wert sichtbar vor, der dann mit abgespeichert wird.


**Erstellung mit der grafischen Benutzeroberfläche**

Die Erstellung innerhalb der grafischen Benutzeroberfläche ist wird beispielhaft für die Tabelle Medien Schritt für Schritt erklärt.

Durch einen Klick auf **Tabelle in der Entwurfsansicht erstellen** wird der Tabelleneditor gestartet.

1. **ID-Feld:**
   - In der ersten Spalte wird «ID» als **Feldname** eingegeben. Danach wird die die Tabulatortaste gedrückt, um in die Spalte **Feldtyp** zu wechseln. Alternativ kann auch mit der Maus die Auswahlmenüe der nächsten Spalte angeklickt werden.

Tipp
Die Erstellung einer Verknüpfung zur Auswahl aus der Auswahlliste Feldtyp:

«ID» wird als Primärschlüssel gesetzt, indem mit der rechten Maustaste auf das grüne Dreieck auf der linken Seite der Zeile «ID» geklickt und Primärschlüssel in dem Kontextmenü ausgewählt wird. Dies stellt ein Schlüssel-Symbol vor «ID».

Hinweis
Der Primärschlüssel dient nur einem Zweck – nämlich zur eindeutigen Identifizierung des Datensatzes. Daher kann ein beliebiger Name für dieses Feld verwendet werden. Im Beispiel wurde die allgemein übliche Bezeichnung ID (Identifikation) verwendet.


2. Das nächste Feld ist das Feld «Titel»
   ➢ Der Feldname «Titel» wird in der Spalte «Feldname» eingegeben.
   ➢ Der Feldtyp muss hier nicht geändert werden, da er bereits auf Text [VARCHAR] eingestellt ist.

   ➢ In den Feldeigenschaften muss die Länge des Feldes angepasst werden. Die Standardlänge ist hier bei den neueren LO-Versionen 100, sollte für den Medientitel aber auf 250 Zeichen erweitert werden.
   ➢ In den Feldeigenschaften sollte wenigstens bei diesem Feld Eingabe erforderlich → Ja gewählt werden. Ein Medium ohne Titel wird keinen Sinn machen.
3. **Beschreibung** kann alles sein, die Spalte kann auch leer gelassen werden. Die Beschreibung dient nur zur Erklärung des Feldinhaltes für Personen, die sich später einmal den Tabellenentwurf ansehen wollen.


   ➢ Bei den **Feldeigenschaften** wird eine **Länge** von 6 Zeichen eingestellt. Das dürfte für die Wertangabe der Medien reichen.
   ➢ Die Zahl der **Nachkommastellen** wird auf 2 Zeichen festgelegt. Damit ist jetzt maximal eine Wertangabe von 9999,99 möglich, da das Komma nicht als Stelle mitgerechnet wird.
   ➢ Eine Formatierung der Ausgabe als € ist nicht erforderlich. Diese Einstellung erfolgt im Formular.


<table>
<thead>
<tr>
<th>Verlag_ID</th>
<th>Zahl [INTEGER]</th>
<th>Fremdschlüssel zu Verlag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titel</td>
<td>Zahl [NUMERIC]</td>
<td>Werterfassung in €</td>
</tr>
<tr>
<td>Bild</td>
<td>Bild [LONGVARBINARY]</td>
<td></td>
</tr>
</tbody>
</table>

**Feldereigenschaften**

- Eingabe erforderlich: Nein
- Länge: 13

### Hinweis

Die Reihenfolge der Felder in der Tabelle kann nur bis zum ersten Abspeichern über die grafische Benutzeroberfläche beeinflusst werden. Sie lässt sich später nur bei der Eingabe der Daten direkt in die Tabelle nicht mehr ändern. In Abfragen, Formularen oder Berichten ist die Reihenfolge allerdings weiterhin frei zusammensetzzbar.

### Primärschlüssel


Es können ohne weiteres auch mehrere Felder zum gemeinsamen Primärschlüssel erstellt werden. Hierzu müssen nur die entsprechenden Felder gemeinsam markiert werden (Taste Strg oder Shift gedrückt halten). Dann kann über den Rechtsklick der Primärschlüssel allen markierten Feldern zugewiesen werden.

<table>
<thead>
<tr>
<th>Hinweis</th>
</tr>
</thead>
</table>

- **Hinweis**: Wird nicht direkt beim Anlegen der Tabelle in der grafischen Benutzeroberfläche der Primärschlüssel festgelegt, so ist die anschließend über die grafische Benutzeroberfläche auch nicht mehr möglich. *(Bug 61547)* Dieser Bug ist ab LO 4.0.5 behoben.

Stattdessen muss der Primärschlüssel über Extras → SQL erstellt werden:

```
ALTER TABLE "Tabellenname" ADD PRIMARY KEY ("Feldname")
```

Sollen von einer anderen Tabelle in dieser Tabelle Informationen mitgeführt werden (Beispiel: Adressdatenbank, aber ausgelagert Postleitzahlen und Orte), so ist ein Feld mit dem gleichen Datentyp wie dem des Primärschlüssels der anderen Tabelle in die Tabelle aufzunehmen. Ange-
nommen die Tabelle "PLZ_Ort" hat als Primärschlüssel das Feld "ID", als Datentyp 'Tiny Integer'. In
der Tabelle Adressen erscheint jetzt ein Feld "ID_PLZ_Ort" mit dem Datentyp 'Tiny Integer'. Es wird
also in der Tabelle Adressen immer nur die Zahl eingetragen, die als Primärschlüssel in der Tabelle
"PLZ_Ort" steht. Für die Tabelle "Adresse" heißt das: Sie hat einen Fremdschlüssel zusätzlich zum
eigenen Primärschlüssel bekommen.

Grundlage bei der Namenswahl von Feldern in der Tabelle: Keine 2 Felder dürfen gleich heißen.
Deswegen darf auch nicht ein zweites Feld mit der Bezeichnung "ID" als Fremdschlüssel in der
Tabelle "Adressen" auftauchen.

Die Feldtypen können nur begrenzt geändert werden. Eine Aufstufung (längeres Textfeld, größerer
Zahlenumfang) ist ohne weiteres möglich, da alle eventuell schon eingegebenen Werte diesem
Feldtyp entsprechen. Eine Abstufung wirft eher Probleme auf. Hier droht gegebenenfalls Datenver-
lust.

Zeitfelder in Tabellen können nicht als Felder mit Bruchteilen einer Sekunde dargestellt werden.
Dies geht nur mit einem Timestamp-Feld. Mit der grafischen Benutzeroberfläche wird allerdings
nur ein Timestamp-Feld erzeugt, dass Datum, Stunde, Minute und Sekunde abspeichert. Dieses
Feld muss über Extras → SQL noch entsprechend verändert werden.

    ALTER TABLE "Tabellenname" ALTER COLUMN "Feldname" TIMESTAMP(6)

Mit dem Parameter «6» wird das Timestamp-Feld auch für Bruchteile von Sekunden aufnahmefä-
hig.

Formatierung von Feldern

Die Formatierung stellt die Werte der Datenbank für den Nutzer dar und erlaubt eine Eingabe von
Werten in Abhängigkeit von landesüblichen Eingabeformen. Ohne Formatierung werden Dezimal-
stellen mit einem Punkt abgetrennt (4.21 statt 4,21), Datumswerte im Format 2014-12-22 dar-
gestellt. Bei der Einstellung der Formatierung muss deshalb auf die Landeseinstellung geachtet
werden.

Die Formatierung zeigt die Inhalte nur an. Eine Datumsdarstellung mit einer zweistelligen Jahres-
zahl wird dennoch als vierstellige Jahreszahl gespeichert. Wird ein Feld für eine Zahl mit zwei
Nachkommastellen erstellt, wie es bei der Überziehungsgebühr in dem folgenden Beispiel der Fall
ist, so wird auch eine Zahl mit zwei Nachkommastellen gespeichert, auch wenn irrtümlich die For-
matierung vielleicht ohne Nachkommastellen erstellt wurde. Eine Zahl mit zwei Nachkommastellen
lässt sich sogar in ein Feld eingeben, das ohne Nachkommastellen formatiert wurde. Die Nach-
kommastellen verschwinden scheinbar bei der Eingabe, werden aber wieder sichtbar, wenn die
Formatierung angepasst wird.

Soll nur eine Zeit, aber kein Datum gespeichert werden, so sollte die Tabelle entsprechend z.B. so
formatiert werden, dass nur Minuten, Sekunden und Zehntelsekunden abgefragt werden:
MM:SS,00. Eine Formatierung mit Nachkommastellen ist später in Formularen nur über das forma-
tierte Feld, nicht über das Zeitfeld möglich.

Die Formatierung von Feldern wird bei Erstellung der Tabelle oder auch anschließend in den Feld-
eigenschaften über einen gesonderten Dialog vorgenommen:
Über den Button in **Feldeigenschaften → Format-Beispiel** wird der Dialog zur Änderung des Formats gestartet.
Für die Erstellung von Feldern, die eine Währung aufnehmen sollen, ist darauf zu achten, dass die Zahlenfelder zwei Nachkommastellen haben. Die Formatierung kann in der Tabellenerstellung der grafischen Benutzeroberfläche in der gewünschten Währung für die Eingabe in die Tabelle vorgenommen werden. Dies hat allerdings nur Auswirkungen auf die Eingabe in der Tabelle und auf Abfragen, die den Wert ohne Umrechnungen auslesen. In Formularen muss die Währungsbezeichnung gesondert formatiert werden.

**Hinweis**
Base speichert zur Zeit nur die Formatierungen der Tabelle ab, die entweder beim Erstellen der Tabelle oder bei der Eingabe von Daten über die Spaltenköpfe erfolgt. Dies gilt auch für die Spaltenbreiten bei der Eingabe. Gesonderte Formatierungen von Abfragen werden dagegen nicht gespeichert.

Bei Feldern, die einen Prozentsatz aufnehmen sollen, ist darauf zu achten, dass 1 % bereits als 0,01 gespeichert werden muss. Die Prozentschreibweise beansprucht also schon standardmäßig 2 Nachkommastellen. Sollen Prozentwerte wie 3,45 % abgespeichert werden, so sind also 4 Nachkommastellen bei dem numerischen Wert notwendig.

**Hinweis**

Dieser Bug ist ab der Version LibreOffice 4.3.2 behoben.

**Einstellung eines Indexes**
Manchmal erscheint es sinnvoll, neben dem Primärschlüssel auch andere Felder oder eine Kombination anderer Felder mit einem Index zu versehen. Ein Index dient dazu, Suchergebnisse schneller zu erhalten. Er kann außerdem dazu genutzt werden, Doppeleingaben zu vermeiden.

Jeder Index hat eine fest definierte Sortierreihenfolge. Wird eine Tabelle ohne Sortierung aufgerufen, so richtet sich die Sortierreihenfolge immer nach der Sortierreihenfolge der als Index definierten Felder.

Zuerst muss die Tabelle mit einem rechten Mausklick über das Kontextmenü zum Bearbeiten geöffnet werden. Der Zugriff auf den Indexentwurf erfolgt dann über Extras → Indexentwurf... oder direkt über den entsprechenden Button in der Menüleiste des Tabellenentwurfes.
Abbildung 8: Erstellen eines neuen Indexes

Über "Neuer Index" wird ein Index neben dem des Primärschlüssels erstellt.

Abbildung 9: Der Index wird als "Eindeutig" definiert.

Dem neuen Index wird automatisch die Bezeichnung "index1" zugewiesen. Diese Bezeichnung kann geändert werden. Im Indexfeld wird ausgewählt, welches Feld bzw. welche Felder über den Index verwaltet werden sollen. Dabei wird gleichzeitig eine Sortierung eingestellt.

Ein Index kann prinzipiell auch über Tabellenfelder erstellt werden, die keine eindeutigen Werte haben. Im obigen Bild ist aber das Index-Detail "Eindeutig" gewählt, so dass in das Feld "Nachname" zusammen mit dem Feld "Vorname" nur Werte eingegeben werden können, die dort in der Kombination noch nicht stehen. So ist z.B. Robert Müller und Robert Maier möglich, ebenso Robert Müller und Eva Müller.
Wird ein Index über ein Feld erstellt, so wird die Eindeutigkeit auf ein Feld bezogen. Ein solcher Index ist in der Regel der Primärschlüssel. In diesem Feld darf jeder Wert nur einmal vorkommen. Beim Primärschlüssel darf allerdings zusätzlich das Feld auf keinen Fall NULL sein.

Eine Sonderstellung für einen eindeutigen Index nimmt in einem Feld das Fehlen eines Eintrages, also NULL, ein. Da NULL alle beliebigen Werte annehmen könnte ist es ohne weiteres erlaubt, bei einem Index über zwei Felder in einem Feld mehrmals hintereinander die gleiche Eingabe zu täti-
gen, solange in dem anderen Feld keine weitere Angabe gemacht wird.

Null ist für Datenbanken die Bezeichnung für eine leere Zelle, die nichts enthält. Mit einem Feld, das NULL ist kann also nicht gerechnet werden. Im Gegensatz dazu gehen Tabellenkalkulationen bei leeren Feldern automatisch davon aus, dass der Inhalt «0» ist.

Beispiel: In einer Mediendatenbank wird für die Ausleihe die Mediennummer und das Ausleihdatum eingegeben. Wird das Medium zurückgegeben, so wird dies durch ein Rückgabedatum vermerkt. Nun könnte ein Index über die Felder "Mediennummer" und "Rückgabedatum" doch leicht verhin-
dern, dass das gleiche Medium mehrmals ausgeliehen wird, ohne dass die Rückgabe vermerkt
wurde. Dies funktioniert aber leider nicht, da das Rückgabedatum ja noch nicht mit einem Wert
versehen ist. Der Index verhindert stattdessen, dass ein Medium zweimal mit dem gleichen Datum
zurückgegeben wird – sonst nichts.

Hinweis: In den LO-Versionen 4.* befand sich anfangs ein Bug dieses Index-Entwurfes. Ein
Index kann einwandfrei erstellt, aber anschließend nicht mehr geändert oder
gelöscht werden (Bug 63539). Dieser Bug taucht bei LO 4.2, LO 4.1.2 und LO 4.0.6
nicht mehr auf.

Bei einer der Vorversionen half der folgende Umweg:
Der Index-Entwurf wird geöffnet. Ein zusätzlicher neuer Index wird als erstellt und
ohne Eingaben gleich wieder gelöscht. Jetzt kann auch der vorher erstellte Index
wieder bearbeitet oder gelöscht werden.

Probleme bei der Änderung von Tabellen
Tabellen sollten am besten direkt bei der Erstellung komplett mit allen nötigen Einstellungen verse-
hen werden. Sollen hinterher die Eigenschaften von Feldern verändert werden (Feldname, erfor-
derliche Eingabe usw.), so kann das häufig zu einer Fehlermeldung führen, die nicht Ursache der
GUI ist, sondern in der darunterliegenden Datenbank begründet ist.

Ein Klick auf Ja führt zu einer weiteren Fehlermeldung, da die Datenbank die Löschung nicht zulässt. Ein Klick auf Nein bricht den gesamten Vorgang ab. Häufig wird die Frage nach mehr Informationen gar nicht erst gestellt, weil die Informationen nur mit etwas gründlicherem Wissen auch sinnvoll zu einer anderen Handlungsweise und dann zum gewünschten Ziel führen können.
Die Fehlermeldung **Column is referenced in constraint or view** bedeutet:


Taucht jetzt noch wieder ein Fehler auf, so liegt dieser häufig darin begründet, dass das entsprechende Feld mit einem Feld einer anderen Tabelle in der Beziehungsdefinition verknüpft wurde. Hier muss dann zuerst die Beziehung unter Extras → Beziehungen gelöst werden, bevor die Änderung vorgenommen werden kann.

**Mängel der grafischen Tabellenerstellung**

Die Reihenfolge der Tabellenfelder kann im Anschluss an den Abspeichervorgang nicht mehr geändert werden. Für eine Darstellung in anderer Reihenfolge ist dann eine Abfrage notwendig. Dies gilt, obwohl die grafische Benutzeroberfläche etwas anderes vortäuscht. Hier kann bei der Tabellenerstellung und bei der Tabellenbearbeitung ein Kontextmenü aufgerufen werden, das z.B. anbietet, Felder auszuschneiden und an anderer Stelle einzufügen. Damit sind dann aber nur die Feldbezeichnungen und die Feldtypen gemeint, nicht aber die Inhalte der Tabelle. Die tauchen nach so einer Änderung mit anderer Feldbezeichnung und eventuell auch anderem Feldtyp wieder auf.6

Nur über direkte SQL-Eingabe kann ein neues Feld an eine bestimmte Position innerhalb der Tabelle rutschen. Bereits erstellte Felder sind aber auch hier nicht beweglich.

---

6 **Bug 51605**

Tabellen: Erstellung von Tabellen 109


**Direkte Eingabe von SQL-Befehlen**

Die direkte Eingabe von SQL-Befehlen ist über das Menü **Extras → SQL** erreichbar.

![Menu mit Extras → SQL](image)

Hier werden Befehle im oberen Fensterbereich eingegeben; im unteren Bereich wird der Erfolg oder gegebenenfalls die Gründe für den fehlenden Erfolg (auf Englisch) mitgeteilt. Abfragen können hier unter «Ausgabe» dargestellt werden, wenn das Ankreuzfeld markiert wird.

**Hinweis**

**Tabellenerstellung**
Ein einfacher Befehl um eine gebrauchstüchtige Tabelle zu erstellen, ist z. B.

```
CREATE TABLE "Test" ("ID" INT PRIMARY KEY, "Text" VARCHAR(50));
CREATE TABLE "Test": Erschaffe eine Tabelle mit dem Namen "Test".
```

( ): mit den hierin enthaltenen Feldnamen, Feldtypen und Zusätzen.

---

**Abbildung 10: Fenster für direkte Eingabe von SQL-Befehlen**

CREATE [MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT] TABLE "Tabellenname" ( <Felddefinition> [, ...] [, <Bedingungsdefinition>...] ) [ON COMMIT {DELETE | PRESERVE} ROWS];

[MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT]:
Die Standardeinstellung ist hier MEMORY. Die HSQLDB erstellt also grundsätzlich alle Tabellen im Arbeitsspeicher. Dies gilt auch für die Tabellen, die über LibreOffice Base in der internen Datenbank geschrieben werden. Eine andere Möglichkeit wäre, die Tabellen auf die Festplatte zu schreiben und nur über den Arbeitsspeicher den Zugriff auf die Festplatte puffern zu lassen (CACHED).

Hinweis
CREATE TEXT TABLE "Text" ("ID" INT PRIMARY KEY, "Text" VARCHAR(50));
Eine Texttabelle wird in der HSQLDB erstellt. Sie muss jetzt mit einer externen Textdatei (z. B. einer *.csv-Datei) verbunden werden:

SET TABLE "Text" SOURCE "Text.csv";

Die Datei "Text.csv" muss jetzt natürlich die entsprechenden Felder in der entsprechenden Reihenfolge enthalten. Es können bei der Erstellung der Verbindung zusätzliche Optionen gewählt werden. Details hierzu:
http://hsqldb.org/doc/guide/ch09.html#set_table_source-section

Texttabellen sind nicht gegenüber anderen Programmen schreibgeschützt. Es kann also passieren, dass ein anderes Programm/ein anderer Nutzer gerade die Tabelle ändert, während Base darauf zugreift. Texttabellen taugen in der Hauptsache zum Datenaustausch zwischen verschiedenen Programme.

Auf TEMPORARY bzw. TEMP kann Base nicht zugreifen. Die SQL-Befehle werden hier wohl abgesetzt, die Tabellen aber nicht in der grafischen Benutzeroberfläche angezeigt (und damit auch nicht über die grafische Benutzeroberfläche löschar) und die Eingaben (über SQL) auch nicht anzeigbar, es sei denn die automatische Lösung des Inhaltes nach dem endgültigen Abspeichern ist ausgeschaltet. Eine Abfrage ergibt hier eine Tabelle ohne Inhalt.

Tabellen, die mit SQL direkt gegründet wurden, werden nicht direkt angezeigt. Hier muss entweder über Ansicht → Tabellen aktualisieren eine Auffrischung erfolgen oder die Datenbank einfach geschlossen und erneut geöffnet werden.

<Felddefinition>:
    "Feldname" Datentyp [(Zeichenanzahl[,Nachkommastellen])] [{DEFAULT "Standardwert" | GENERATED BY DEFAULT AS IDENTITY (START WITH <n>[, INCREMENT BY <m>])}] | [[NOT] NULL] [IDENTITY] [PRIMARY KEY]

Erlaubte Standardwerte innerhalb der Felddefinition:

Für Textfelder kann ein Text in einfachen Anführungsstrichen oder NULL eingegeben werden. Die einzige SQL-Funktion, die erlaubt ist, ist CURRENT_USER. Dies ergibt allerdings nur dann einen Sinn, wenn die HSQLDB als externe Serverdatenbank mit mehreren Nutzern betrieben wird.

Für Datums- und Zeitfelder kann ein Datum, eine Zeit oder eine Kombination aus Datum und Zeit in einfachen Anführungsstrichen oder NULL eingegeben werden. Dabei ist zu beachten, dass das Datum amerikanischen Konventionen entspricht (yyyy-mm-dd), die Zeitangabe das Format hh:mm:ss hat und der Datums_Zeit_Wert das Format yyyy-mm-dd hh:mm:ss.

SQL-Funktionen, die erlaubt sind:
   für das aktuelle Datum - CURRENT_DATE, TODAY

Tabellen: Erstellung von Tabellen
für die aktuelle Zeit - **CURRENT_TIME, NOW**
für den aktuellen Datums-Zeit-Wert - **CURRENT_TIMESTAMP, NOW**.

Für boolsche Felder (Ja/Nein) können die Ausdrücke **FALSE, TRUE, NULL** gesetzt werden. Diese sind ohne einfache Anführungszeichen einzugeben.


Für Binärfelder (Bilder etc.) ist jeder gültige Hexadezimalstring in einfachen Anführungsstrichen sowie **NULL** möglich. Beispiel für einen Hexadezimalstring: '0004ff' bedeutet 3 Bytes, zuerst 0, als zweites 4 und zum Schluss 255 (0xff). Da Binärfelder in der Praxis nur für Bilder eingesetzt werden müsste also der Binärkode des Bildes bekannt sein, das den Defaultwert bilden soll.

---

**Hinweis**

Hexadezimalsystem: Zahlen werden in einem Stellenwertsystem von 16 dargestellt. Die Ziffern 0 bis 9 und die Buchstaben a bis f ergeben pro Spalte 16 Ziffern im Mischsystem. Bei zwei Feldern kommen dann 16*16=256 mögliche Werte dabei zustande. Das entspricht schließlich 1 Byte.

**NOT NULL** – der Feldwert kann nicht NULL sein. Diese Bedingung kann lediglich in der Felddefinition mit angegeben werden.

Beispiel:

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


**<Bedingungsdefinition>**:  
[CONSTRAINT "Name"]  
UNIQUE ( "Feldname 1" [,"Feldname 2"...] ) |  
PRIMARY KEY ( "Feldname 1" [,"Feldname 2"...] ) |  
FOREIGN KEY ( "Feldname 1" [,"Feldname 2"...] ) REFERENCES "anderer Tabellenname" ("Feldname 1" [,"Feldname 2"...] ) [ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}] |  
CHECK(<Suchbedingung>)  

Bedingungdefinitionen (Constraints) definieren Bedingungen, die beim Einfügen der Daten erfüllt sein müssen. Die Constraints können mit einem Namen versehen werden.

**UNIQUE ("Feldname")** – der Feldwert muss innerhalb des Feldes einzigartig sein  
**PRIMARY KEY ("Feldname")** – der Feldwert muss einzigartig sein und kann nicht NULL sein (Primärschlüssel)  
**FOREIGN KEY ("Feldname") REFERENCES "anderer Tabellenname" ("Feldname")** – Die aufgeführten Felder dieser Tabelle sind mit den Feldern einer anderen Tabelle verknüpft. Der Feldwert muss auf »Referentielle Integrität« geprüft werden (Fremdschlüssel), d.h. es muss ein entsprechender Primärschlüssel in der anderen Tabelle existieren, wenn hier ein Wert eingetragen wird.  
**[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}]** – Wenn ein Fremdschlüssel besteht, so ist hier zu klären, wie damit verfahren werden soll, wenn z.B. der fremde Datensatz gelöscht wird. Z.B. macht es keinen Sinn, in einer Ausleihtabelle einer Bibliol...
thek eine Nutzernummer weiter zu führen, wenn der Nutzer selbst gar nicht mehr existiert. Die entsprechenden Datensätze müssten behandelt werden, so dass die Beziehung zwischen den Tabellen stimmig bleibt. In der Regel würde der entsprechende Datensatz einfach gelöscht. Dies geschieht mit **ON DELETE CASCADE**.

**CHECK(<Suchbedingung>)** – Wird wie eine **WHERE**-Bedingung formuliert, bezieht sich aber nur auf den aktuellen Datensatz.

```
CREATE TABLE "Zeitmessung" ("ID" INT PRIMARY KEY, "Startzeit" TIME,  
"Zielzeit" TIME, CHECK ("Startzeit" <= "Zielzeit"));
```

Mit der CHECK-Bedingung wird ausgeschlossen, dass eine "Zielzeit" eingegeben wird, die kleiner als die "Startzeit" ist. Wird dies versucht, so erscheint eine englischsprachige Fehlermeldung, die ungefähr folgendermaßen aussieht:

**Check constraint violation SYS_CT_357 table: Zeitmessung ...**

Der Suchbedingung wird hier gleich ein Name zugewiesen, der allerdings nicht sehr aussagekräftig ist. Stattdessen bietet es sich an, direkt bei der Tabellendefinition den Namen zu definieren:

```
CREATE TABLE "Zeitmessung" ("ID" INT PRIMARY KEY, "Startzeit" TIME,  
"Zielzeit" TIME, CONSTRAINT "Startzeit<=Zielzeit" CHECK ("Startzeit" <= "Zielzeit"));
```

Damit wird die Fehlermeldung etwas klarer. Der Name der Bedingung drückt dann wenigstens aus, worauf sie sich bezieht.

Mit Constraints wird vor allem gearbeitet, wenn die Beziehung zwischen Tabellen oder der Index für bestimmte Felder festgelegt werden soll. Die Constraints werden, bis auf die CHECK-Bedingung, in der GUI unter **Extras → Beziehungen** und als Indexentwurf in dem Tabellenentwurf unter **Extras → Indexentwurf** festgelegt.

**[ON COMMIT {DELETE | PRESERVE} ROWS]:**

Der Inhalt von Tabellen des Typs **TEMPORARY** oder **TEMP** wird nach Beendigung der Arbeit mit dem Datensatz standardmäßig gelöscht (**ON COMMIT DELETE ROWS**). Hier kann also nur ein flüchtiger Datensatz erstellt werden, der Informationen für andere Aktionen, die gleichzeitig laufen, vorhält.

Sollen diese Tabellentypen Daten für eine ganze Sitzung (Aufruf einer Datenbank und Schließen einer Datenbank) zur Verfügung stehen, so kann hier **ON COMMIT PRESERVE ROWS** gewählt werden.

**Tabellenänderung**

Manchmal wünscht sich der User, dass ein zusätzliches Feld an einer bestimmten Stelle in die Tabelle eingebaut wird. Angenommen es gibt die Tabelle "Adresse" mit den Feldern "ID", "Name", "Strasse" usw. Jetzt fällt dem Nutzer auf, dass vielleicht eine Unterscheidung in Name und Vorname sinnvoll wäre:

```
ALTER TABLE "Adresse" ADD "Vorname" VARCHAR(25) BEFORE "Name";
```

**ALTER TABLE "Adresse"**: Ändere die Tabelle mit dem Namen "Adresse".

**ADD "Vorname" VARCHAR(25)**: füge das Feld "Vorname" mit einer Länge von 25 Zeichen hinzu.

**BEFORE "Name"**: und zwar vor dem Feld "Name".

Die Möglichkeit, die Position nach dem Erstellen einer Tabelle für zusätzliche Felder zu bestimmen, bietet die GUI nicht.

```
ALTER TABLE "Tabellenname" ADD [COLUMN] <Felddefinition> [BEFORE "bereits_existierender_Feldname"];
```
Die zusätzliche Bezeichnung COLUMN ist dabei nicht erforderlich, da keine Alternativen zur Auswahl stehen.

ALTER TABLE "Tabellenname" DROP [COLUMN] "Feldname";

Das Feld "Feldname" wird aus der Tabelle "Tabellenname" gelöscht. Dies wird allerdings unterbunden, wenn das Feld in einem View oder als Fremdschlüssel in einer anderen Tabelle Bedeutung hat.

ALTER TABLE "Tabellenname" ALTER COLUMN "Feldname" RENAME TO "neuer_Feldname"

Ändert den Namen eines Feldes

ALTER TABLE "Tabellenname" ALTER COLUMN "Feldname" SET DEFAULT <Standardwert>;

Fügt dem Feld einen bestimmten Standardwert hinzu. NULL entfernt einen bestehenden Standardwert.

ALTER TABLE "Tabellenname" ALTER COLUMN "Datum" SET DEFAULT TODAY;

Dies ändert ein Datumsfeld so, dass das momentane Datum eingefügt wird, wenn das Feld "Datum" leer ist.

ALTER TABLE "Tabellenname" ALTER COLUMN "Feldname" SET [NOT] NULL

Setzt oder entfernt eine NOT NULL Bedingung für ein Feld.

ALTER TABLE "Tabellenname" ALTER COLUMN <Felddefinition>;

Die Felddefinition entspricht der aus der Tabellenerstellung mit den folgenden Einschränkungen:


- Wenn das Feld bereits die Eigenschaft IDENTITY hat und sie wird nicht in die Felddefinition erneut aufgenommen, so wird die bereits existierende Eigenschaft IDENTITY entfernt.

- Der Standardwert wird der der neuen Felddefinition sein. Wenn die Definition des Standardwertes leer gelassen wird, so wird ein bereits bestehender entfernt.

- Die Eigenschaft NOT NULL wird in die neue Definition übernommen, wenn nicht anders definiert. Dies entspricht dem Umgang mit dem Standardwert.

- Abhängig von der Art der Änderung muss eventuell die Tabelle leer sein, damit die Änderung durchgeführt werden kann. Auf jeden Fall wird die Änderung dann funktionieren, wenn die Änderung grundsätzlich möglich ist (z.B. Änderung von NOT NULL auf NULL) und die existierenden Werte alle umgewandelt werden können (z.B. von TINYINT zu INTEGER).

ALTER TABLE "Tabellenname" ADD PRIMARY KEY ("Feldname1", "Feldname2" ...);

Dieser Befehl erstellt im Nachhinein einen Primärschlüssel, auch über mehrere Felder.

ALTER TABLE "Tabellenname"
ALTER COLUMN "Feldname" RESTART WITH <neuer_Feldwert>


ALTER TABLE "Tabellenname"
ADD [CONSTRAINT "Bedingungsname"] CHECK (<Suchbedingung>);
Dies fügt eine mit CHECK eingeleitete Suchbedingung hinzu. Solch eine Bedingung wird nicht auf bereits bestehende Datensätze angewandt, sondern bei allen zukünftigen Änderungen und neu erstellten Datensätzen berücksichtigt. Wird kein Bedingungsname definiert, so wird automatisch eine Bezeichnung zugewiesen. Beispiel:

```sql
ALTER TABLE "Ausleihe" ADD CHECK
(IFNULL("Rueckdatum","Leihdatum")>="Leihdatum")
```

Die Tabelle "Ausleihe" soll in Bezug auf Fehleingaben abgesichert werden. Es soll vermieden werden, dass ein Rückgabedatum angegeben wird, das vor dem Ausleihdatum liegt. Taucht jetzt bei der Eingabe des Rückgabedatums dieser Fehler auf, so erscheint die Fehlermeldung **Check constraint violation** ...

```sql
ALTER TABLE "Tabellenname"
ADD [CONSTRAINT "Bedingungsname"] UNIQUE ("Feldname1",
"Feldname2" ...);
```

Hier wird hinzugefügt, dass die benannten Felder nur jeweils verschiedene Werte beinhalten dürfen. Werden mehrere Felder benannt, so gilt dies für die Kombination von Feldern. **NULL** wird hierbei nicht berücksichtigt. Ein Feld kann also ohne weiteres mehrmals die gleichen Werte haben, wenn das andere Feld bei den entsprechenden Datensätzen **NULL** ist.

Der Befehl wird nicht funktionieren, wenn bereits eine **UNIQUE** – Bedingung für die gleiche Felderkombination existiert.

```sql
ALTER TABLE "Tabellenname"
ADD [CONSTRAINT "Bedingungsname"] PRIMARY KEY ("Feldname1",
"Feldname2" ...);
```

Fügt einen Primärschlüssel, gegebenenfalls mit einer Bedingungsdefinition, einer Tabelle hinzu.

Die Syntax der Bedingungsdefinition entspricht der der Erstellung bei einer Tabelle.

```sql
ALTER TABLE "Tabellenname"
ADD [CONSTRAINT "Bedingungsname"] FOREIGN KEY ("Feldname1",
"Feldname2" ...)
REFERENCES "Tabellenname_der_anderen_Tabelle"
("Feldname1_andere_Tabelle", "Feldname2_andere_Tabelle" ...)
[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}];
```

Hiermit wird eine Fremdschlüsselbedingung (**FOREIGN KEY**) zur Tabelle hinzugefügt. Die Syntax ist die gleiche wie bei der Erstellung einer Tabelle.

Das Verfahren wird mit einer Fehlermeldung beendet, wenn nicht für jeden Wert in der Tabelle ein entsprechender Wert aus der Tabelle mit dem entsprechenden Schlüsseldatum vorhanden ist.


```sql
ALTER TABLE "Tabellenname" DROP CONSTRAINT "Bedingungsname";
```

Der Befehl entfernt eine mit Namen versehene Bedingung (**UNIQUE, CHECK, FOREIGN KEY**) aus einer Tabelle.

```sql
ALTER TABLE "Tabellenname" RENAME TO "neuer_Tabellenname";
```

Mit diesem Befehl schließlich wird einfach nur der Name einer Tabelle geändert.
Bei der Änderung einer Tabelle über SQL wird die Änderung zwar in der Datenbank übernommen, nicht aber unbedingt sofort überall in der GUI sichtbar und verfügbar. Wird die Datenbank geschlossen und wieder geöffnet, so werden die Änderungen auch in der GUI angezeigt.
Die Änderungen werden auch dann angezeigt, wenn im Tabellencontainer Ansicht → Tabellen aktualisieren aufgerufen wird.

**Tabellen löschen**

DROP TABLE "Tabellenname" [IF EXISTS] [RESTRICT | CASCADE];
Löscht die Tabelle "Tabellenname".

**IF EXISTS** schließt aus, dass eine Fehlermeldung erscheint, falls diese Tabelle nicht existiert.

**RESTRICT** ist die Standardeinstellung und muss nicht definitiv gewählt werden, d.h. ein Löschen wird dann nicht ausgeführt, wenn die Tabelle mit irgendeiner anderen Tabelle durch einen Fremdschlüssel verbunden wurde oder auf die Tabelle mit einem View (Ansicht) Bezug genommen wird. Abfragen sind davon nicht berührt, da die innerhalb der HSQLDB nicht gespeichert sind.

Wird statt **RESTRICT CASCADE** gewählt, so werden alle Beziehungen zu der Tabelle "Tabellenname" gelöscht. In den verknüpften Tabellen werden dann alle Fremdschlüsselfelder auf NULL gesetzt. Alle Views, in denen auf die entsprechende Tabelle Bezug genommen wird, werden komplett gelöscht.

**Verknüpfung von Tabellen**

Prinzipiell kommt eine Datenbank auch ohne die Verknüpfung von Tabellen aus. Der Nutzer muss dann bei der Eingabe selbst darauf achten, dass die Beziehungen zwischen den Tabellen stimmig bleiben. In der Regel geschieht dies, indem er sich entsprechende Formulare erstellt, die dies bewerkstelligen sollen.


Die folgenden Fehlermeldungen können beim Ziehen so einer Verbindung auftreten:
Die Meldung gibt einen englischen Text sowie das zu der Fehlermeldung führende SQL-Kommando wieder. Eine gute Möglichkeit also, auch an dieser Stelle etwas über die Sprache zu erfahren, mit der die Datenbank arbeitet.

«Column types do not match in statement» - die Spaltentypen stimmen in der SQL-Formulierung nicht überein. Da das SQL-Kommando gleich mitgeliefert wird, müssen das die Spalten Adresse.str_ID und Strasse.ID sein. Zu Testzwecken wurde hier das eine Feld als Integer, das andere als 'Tiny Integer' definiert. So eine Verbindung lässt sich nicht erstellen, da das eine Feld nicht die gleichen Werte annehmen kann wie das andere Feld.

Jetzt stimmen die Spaltentypen überein. Die SQL-Formulierung (statement) ist die gleiche wie beim ersten Zugriff. Aber wieder taucht ein Fehler auf:

Ist die Verbindung erfolgreich erzeugt worden und wird später versucht einen entsprechend fehlerhaften Eintrag in die Tabelle einzugeben, so kommt die folgende Fehlermeldung:

<table>
<thead>
<tr>
<th>ID</th>
<th>str_ID</th>
<th>Nr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>

Also wiederum die Integritätsverletzung. Base weigert sich, für das Feld str_ID nach der Verknüpfung den Wert 1 anzunehmen, weil die Tabelle Strasse so einen Wert im Feld ID nicht enthält.

Abbildung 11: Durch Klick mit der rechten Maustaste können Verknüpfungen bearbeitet werden.

Die Eigenschaften der Verknüpfung können so bearbeitet werden, dass beim Löschen von Datensätzen aus der Tabelle Strasse gleichzeitig eventuell vorhandene Einträge in der Tabelle Adresse auf NULL gesetzt werden.
Die oben abgebildeten Eigenschaften beziehen sich immer auf eine Aktion, die mit einer Änderung des Datensatzes aus der Tabelle zusammenhängt, zu der der betroffene Primärschlüssel gehört. In unserem Fall ist dies die Tabelle *Strasse*. Wird also dort ein Primärschlüssel eines Datensatzes "ID" geändert (Update), so können die folgenden Aktionen ausgeführt werden:

**Keine Aktion:**
Eine Änderung des Primärschlüssels *Strasse.ID* kann in diesem Fall nicht vorgenommen werden, da die Relation ansonsten zerstört wird. Da dies die Standardinstellung der Relation ist gehen Nutzer häufig davon aus, dass eine Änderung des Primärschlüssels unmöglich ist. Die anderen Update-Optionen ermöglichen allerdings so eine Änderung.

**Kask. Update:**

**Null setzen:**
Alle Datensätze, die sich auf den Primärschlüssel bezogen haben, haben jetzt in dem Fremdschlüsseldfeld *Adresse.Strasse_ID* statt dessen keinen Eintrag mehr stehen, sind also NULL.

**Default setzen:**
Wird der Primärschlüssel *Strasse.ID* geändert, so wird der damit ursprünglich verbundene Wert aus *Adresse.Strasse_ID* auf den dafür vorgesehenen Standardwert gesetzt. Hierfür ist allerdings eine eindeutige Definition eines Standardwertes erforderlich. Dies scheint die grafische Benutzeroberfläche bis zur Version LO 3.5 nicht bereitzustellen. Wird per SQL der Default-Wert durch

```
ALTER TABLE "Adresse" ALTER COLUMN "Strasse_ID" SET DEFAULT 1;
```

*Abbildung 12: Dialog «Relationen»*
gesetzt, so übernimmt die Beziehungsdefinition auch die Zuweisung, dass bei einem Update auf diesen Wert zurückgegriffen werden kann. Wird also der Primärschlüssel aus der Tabelle Strasse geändert, so wird in der Tabelle Adresse das Fremdschlüsselfeld auf 1 gesetzt. Dies bietet sich z.B. an, wenn auf jeden Fall jeder Datensatz eine Straßenzuweisung erhalten soll, also nicht NULL sein soll. Aber Achtung: Ist 1 nicht belegt, so wird eine Verbindung zu einem nicht vorhandenen Wert erzeugt. Hier scheint die HSQLDB nicht mit letzter Konsequenz durchdacht. Es ist also möglich, die Integrität der Relationen zu zerstören.

**Vorsicht**

Ist der Defaultwert im Fremdschlüsselfeld nicht durch einen Primärschlüssel der Ursprungstabelle belegt, so wird eine Verbindung zu einem nicht vorhandenen Wert erzeugt. Es ist also möglich, die referentielle Integrität der Datenbank zu zerstören.

Vor diesem Hintergrund scheint es sinnvoll, diese Einstellung nicht in Betracht zu ziehen.

Wird ein Datensatz aus der Tabelle Strasse gelöscht, so stehen die folgenden Optionen zur Verfügung:

**Keine Aktion:**

Es wird nichts unternommen. Ist ein Datensatz aus der Tabelle Adresse von der Löschung betroffen, so wird die Löschung abgelehnt. Es existiert schließlich weiterhin ein entsprechender Fremdschlüssel in der Tabelle Adresse.

Wie bei den Update-Optionen kann nur mit dieser Option eine Lösung des Datensatzes unterbunden werden. Die weiteren Optionen geben hingegen den Weg vor, den die Datenbank beschreiten soll, falls von dem Datensatz in der Tabelle Strasse ein Fremdschlüssel in der Tabelle Adresse betroffen ist.

**Kaskadiert Löschen:**

Wird ein Datensatz aus der Tabelle Strasse gelöscht und ist davon ein Datensatz aus der Tabelle Adresse betroffen, so wird auch dieser gelöscht.

Das mag in diesem Zusammenhang merkwürdig klingen, ergibt aber bei anderen Tabellenkonstruktionen sehr wohl einen Sinn. Angenommen es existiert eine Tabelle mit CDs und eine Tabelle, die die Titel, die auf diesen CDs enthalten sind, speichert. Wird nun ein Datensatz aus der CD-Tabelle gelöscht, so stehen lauter Titel in der anderen Tabelle, die gar keinen Bezug mehr haben, also gar nicht mehr vorhanden sind. Hier ist dann ein kaskadierendes Löschen sinnvoll. So muss der Nutzer nicht vor dem Entfernen der CD aus der Datenbank erst sämtliche Titel löschen.

**Null setzen:**

Dieses Verhalten entspricht der gleich lautenden Update-Option.

**Default setzen:**

Dieses Verhalten entspricht ebenfalls der gleich lautenden Update-Option. Die Bedenken bei dieser Option sind entsprechend die gleichen.

**Tipp**

Sollen möglichst Fehlermeldungen der Datenbank vermieden werden, da sie dem Datenbanknutzer vielleicht nicht deutbar sind, so sind auf jeden Fall die Einstellungen «Keine Aktion» zu vermeiden.

In Extras → Beziehungen können durch Ziehen mit der Maus nur Fremdschlüssel erstellt werden, die sich auf ein Feld einer Tabelle beziehen. Für die Verbindung mit einer Tabelle, die einen zusammengesetzten Primärschlüssel hat, muss in Extras → Beziehungen der Menüpunkt Einfügen → Neue Relation oder der entsprechende Button aufgesucht werden. Es erscheint dann der Dialog «Relationen» mit der freien Auswahl der beteiligten Tabellen.

Tabellen: Verknüpfung von Tabellen
Eingabe von Daten in Tabellen

Datenbanken, bestehend aus einer Tabelle, erfordern in der Regel keine Formulare zur Eingabe, es sei denn sie enthalten ein Feld für Bildmaterial. Sobald allerdings eine Tabelle den Fremdschlüssel einer anderen Tabelle enthält, muss der Nutzer entweder auswendig wissen, welche Schlüsselnummern er eintragen muss, oder er muss die andere Tabelle zum Nachschauen gleichzeitig offen halten. Dafür wird dann spätestens ein Formular sinnvoll.

Eingabe über die grafische Benutzeroberfläche der Tabelle


Abbildung 14: Eingabe in Tabellen - Spalten wieder einblenden.


Die Eingabe in die Tabellenzellen erfolgt in der Regel von links nach rechts über Tastatur und Tabulator. Natürlich ist auch die Nutzung der Maus möglich.


Fehlt eine Eingabe, die vorher im Tabellenentwurf als zwingend notwendig deklariert wurde (NOT NULL), so wird eine entsprechende Fehlermeldung erzeugt:

```sql
Attempt to insert null into a non-nullable column ...
```

Die entsprechende Spalte, die Tabelle dazu und der von der GUI abgesetzte SQL-Befehl werden angezeigt.
Die Änderung eines Datensatzes ist entsprechend einfach möglich: Feld aufsuchen, geänderten Wert eingeben und Datenzeile wieder verlassen.

Zum Löschen eines Datensatzes wird der Datensatz auf dem Zeilenkopf markiert, dann die rechte Maustaste gedrückt und Zeilen löschen gewählt.


Tabellen: Eingabe von Daten in Tabellen

Der Datensatz mit dem Primärschlüsselwert '1' wird als neuer Datensatz mit dem neuen Primär-
enschlüsselwert '9' eingefügt.

Wird nicht nur ein Datensatz markiert, sondern unter Zuhilfenahme der Shift- oder der Strg-Taste
gleich mehrere Datensätze, so können auch mehrere Datensätze direkt kopiert werden.

**Tipp**

Die Spaltenköpfe können auf die für die Eingabe erforderliche Breite gezogen wer-
den. Wird dies in der Tabelle gemacht, so wird die Speicherfunktion von Base akti-
viert. Base speichert diese Spaltenbreiten aus den Tabellen.

Die Spaltenbreiten der Tabellen beeinflussen die Spaltenbreiten in Abfragen. Sind
Spalten in Abfragen zu klein, so hilft dort nur vorübergehend eine Breitenänderung.
Die Einstellung der Abfrage wird nicht gespeichert. Hier hilft dann eine Breitenände-
 rung in der Tabelle, sofern es sich in der Abfrage nicht um Felder handelt, die durch
irgendwelche Funktionen beeinflusst wurden.

Hilfreich zum Aufsuchen des entsprechenden Datensatzes sind die Sortier-, Such- und Filterfunk-
 tionen.

**Sortieren von Tabellen**

![Abbildung 15: Schnelle Sortierung](image)

Die schnelle Sortiervariante verbirgt sich hinter den Buttons A → Z bzw. Z → A. Ein Feld innerhalb
einer Spalte wird aufgesucht, ein Mausklick auf den Button und nach der Spalte wird sortiert. Hier
wurde gerade die Spalte "Titel" absteigend sortiert.

Mit der schnellen Variante kann immer nur nach einer Spalte sortiert werden. Für eine Sortierung
nach mehreren Spalten ist eine weitere Sortierfunktion vorgesehen:
Abbildung 16: Sortierung nach mehreren Spalten

Der Feldname der Spalte sowie die jeweilige Sortierreihenfolge werden ausgesucht. Wurde vorher bereits eine schnelle Sortierung vorgenommen, so ist in der ersten Zeile der entsprechende Feldname und die Sortierreihenfolge bereits eingetragen.

**Suchen in Tabellen**


**Tipp**

Bevor die Suche aufgerufen wird sollte auf jeden Fall darauf geachtet werden, dass die zu durchsuchenden Spalten von der Breite her weit genug eingestellt sind, um den gefundenen Datensatz auch anzuzeigen. Die Suchfunktion bleibt nämlich im Vordergrund und lässt keine Korrektur der Einstellungen der Spaltenweite in der darunterliegenden Tabelle zu. Um an die Tabelle zu gelangen muss die Suche also abgebrochen werden.

Die weiteren Einstellungen der Datensatzsuche können die Suche nach bestimmten Kombinationen vereinfachen. Als Platzhalter-Ausdruck sind hier die in SQL üblichen Platzhalterbezeichnungen («_» für ein variables Zeichen, «%» für eine beliebige Anzahl variabler Zeichen und «\» als Escape-Zeichen, um auch nach den variablen Zeichen selbst suchen zu können).

Reguläre Ausdrücke werden in der Hilfe von LibreOffice unter diesem Begriff ausführlich aufgeführt. Ansonsten gibt sich die Hilfestellung zu diesem Modul recht sparsam. Ein Klick auf den Button Hilfe landet bei LO 3.3.4 und LO 3.5.2 auf einer leeren Seite.

**Abbildung 17: Eingabemaske zur Datensatzsuche**


Die weiteren Einstellungen der Datensatzsuche können die Suche nach bestimmten Kombinationen vereinfachen. Als Platzhalter-Ausdruck sind hier die in SQL üblichen Platzhalterbezeichnungen («_» für ein variables Zeichen, «%» für eine beliebige Anzahl variabler Zeichen und «\» als Escape-Zeichen, um auch nach den variablen Zeichen selbst suchen zu können).

Reguläre Ausdrücke werden in der Hilfe von LibreOffice unter diesem Begriff ausführlich aufgeführt. Ansonsten gibt sich die Hilfestellung zu diesem Modul recht sparsam. Ein Klick auf den Button Hilfe landet bei LO 3.3.4 und LO 3.5.2 auf einer leeren Seite.

Insgesamt ist dieses Suchmodul eher etwas für Leute, die durch häufige Anwendung genau wissen, an welchen Stellen sie zum Erreichen eines Ziels drehen müssen. Für den Normaluser dürfte die Möglichkeit, Datensätze durch Filter zu finden, schneller zum Ziel führen.

Für Formulare ist in einem der folgenden Kapitel beschrieben, wie mittels SQL, und erweitert mit Makros, eine Stichwortsuche schneller zum Ziel führt.

**Filtern von Tabellen**

Die schnelle Filterung läuft über den AutoFilter. Der Cursor wird in ein Feld gesetzt, der Filter übernimmt nach einem Klick auf den Button diesen Feldinhalt. Es werden nur noch die Datensätze angezeigt, die dem Inhalt des gewählten Feldes entsprechen. Die folgende Abbildung zeigt die Filterung nach einem Eintrag in der Spalte "E_Jahr".


Durch Betätigung des ganz rechts stehenden Filtersymbols lassen sich alle bereits veranlassten Filterungen und Sortierungen wieder entfernen. Der Filter wird wieder inaktiv und kann nicht mehr mit seinem alten Wert aufgerufen werden.
In eine Tabelle, die gefiltert oder durch Suche eingegrenzt wurde, können dennoch ganz normal Daten eingegeben werden. Sie bleiben so lange in der Tabellenansicht stehen, bis die Tabelle durch Betätigung des Buttons **Aktualisieren** aktualisiert wird.

Mit dem Standardfilter öffnet sich ein Fenster, in dem ähnlich der Sortierung eine Filterung über mehrere Zeilen ausgeführt werden kann. Ist bereits vorher ein AutoFilter eingestellt, so zeigt die erste Zeile des Standardfilters bereits diesen vorgefilterten Wert an.

Abbildung 19: Umfangreiche Datenfilterung im Standardfilter

Der Standardfilter bringt viele Funktionen einer SQL-Datenfilterung mit. Die folgenden SQL-Bedingungen stehen zur Verfügung:

<table>
<thead>
<tr>
<th>Bedingung GUI</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Vollständige Gleichheit, entspricht dem Begriff wie, wenn keine zusätzlichen Platzhalterbezeichnungen verwendet werden.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Ungleicher als</td>
</tr>
<tr>
<td>&lt;</td>
<td>Kleiner als</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Kleiner als und gleich</td>
</tr>
<tr>
<td>&gt;</td>
<td>Größer als</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Größer als und gleich</td>
</tr>
<tr>
<td>wie</td>
<td>Für Text, in Hochkommata geschrieben (' '): «_» für ein variables Zeichen, «%» für eine beliebige Anzahl variabler Zeichen (in der GUI auch «*»); wird intern umgewandelt). In SQL entspricht wie dem Begriff LIKE</td>
</tr>
<tr>
<td>nicht wie</td>
<td>Umkehrung von wie, in SQL NOT LIKE</td>
</tr>
<tr>
<td>leer</td>
<td>Kein Inhalt auch nicht eine Leertaste. In SQL entspricht dies dem Begriff NULL</td>
</tr>
<tr>
<td>nicht leer</td>
<td>Umkehrung von leer, in SQL NOT NULL</td>
</tr>
</tbody>
</table>

Bevor die Verknüpfung eines Filterkriteriums mit dem nächsten Filterkriterium erfolgen kann, muss in der Folgezeile zumindest schon einmal ein Feldname ausgewählt worden sein. In der obigen Abbildung steht dort statt eines Feldnamens «-keiner-», so dass die Verknüpfung inaktiv ist. Als Verknüpfung stehen hier **UND** und **ODER** zur Verfügung.
Als Feldname kann hier sowohl ein neuer Feldname als auch ein bereits ausgewählter Feldname erscheinen.

Selbst bei großen Datenbeständen dürfte sich bei geschickter Filterung die Anzahl der angezeigten Datensätze mit diesen 3 Bedingungsmöglichkeiten doch auf einen übersichtlichen Bestand ein grenzen lassen.

Auch für die Filterung werden für Formulare in einem der folgenden Kapitel einige weitere Möglichkeiten vorgestellt, die die GUI so nicht zur Verfügung stellt.

**Eingabemöglichkeiten über SQL direkt**

Die Eingabe direkt über SQL ist vor allem dann sinnvoll, wenn mehrere Datensätze mit einem Befehl eingefügt, geändert oder gelöscht werden sollen.

### Neue Datensätze einfügen

```
INSERT INTO "Tabellenname" [( "Feldname" [,....] )]
{ VALUES("Feldwert" [,....]) | <Select-Formulierung>};
```


```
INSERT INTO "Tabellenname" ("Feldname") VALUES ('Test');
CALL IDENTITY();
```

In die Tabelle wird in der Spalte "Name" der Wert 'Test' eingegeben. Das automatisch hochzählende Primärschlüsselfeld "ID" wird nicht angerührt. Der entsprechende Wert für die "ID" wird mit CALL IDENTITY() anschließend ausgelesen. Dies ist bei der Verwendung von Makros wichtig, damit entsprechend mit dem Wert dieses Schlüsselfeldes weiter gearbeitet werden kann.

```
INSERT INTO "Tabelle" ("Vorname","Nachname") VALUES ('Eva','Müller');
```

Hiermit werden die Werte für 2 Felder auf einmal eingegeben. Die Reihenfolge der Felder und die Reihenfolge der entsprechenden Werte muss dabei gleich sein. Text ist bei den Werten in einfache Anführungszeichen zu setzen: 'Text'.

```
INSERT INTO "Tabelle" ("Ware","Preis") VALUES ('Kaffee', 8.79);
```


```
INSERT INTO "Tabellenname" ("Feldname") SELECT "anderer_Feldname" FROM "Name_anderer_Tabelle";
```

In die erste Tabelle werden jetzt so viele neue Datensätze in "Feldname" eingefügt, wie in der Spalte "anderer_Feldname" der zweiten Tabelle enthalten sind. Die SELECT-Formulierung kann hier natürlich einschränkend wirken.

### Bestehende Datensätze ändern

```
UPDATE "Tabellenname" SET "Feldname" = <Expression> [, ...] [WHERE <Expression>];
```

Vor allem bei der Änderung vieler Datensätze bietet es sich an, doch einmal die SQL-Befehlseingabe aufzusuchen. Angenommen alle Schüler einer Klasse sollen zum neuen Schuljahr um eine Jahrgangsstufe heraufgesetzt werden:

```
UPDATE "Tabellenname" SET "Jahrgang" = "Jahrgang"+1
```

Schneller geht es nicht: Alle Datensätze werden mit einem Befehl geändert. Natürlich muss jetzt noch nachgesehen werden, welche Schüler denn davon nicht betroffen sein sollen. Einfacher wäre...
es, vorher in einem Ja/Nein-Feld die Wiederholungen anzukreuzen und dann nur diejenigen eine Stufe heraufzusetzen, die nicht angekreuzt wurden:

```
UPDATE "Tabellenname" SET "Jahrgang" = "Jahrgang"+1 WHERE "Wiederholung" = FALSE
```

Diese Bedingung funktioniert allerdings nur dann, wenn das Feld nur die Werte FALSE und TRUE annehmen kann, also nicht NULL. Sicherer wäre die Formulierung WHERE "Wiederholung" <> TRUE.

Soll nachträglich für ein Feld bei allen leeren Feldern ein Standardwert eingetragen werden, so geht dies mit

```
UPDATE "Tabellenname" SET "Feld" = 1 WHERE "Feld" IS NULL
```


```
UPDATE "Bücher" SET "Autor_Vorname" = 'Erich', "Autor_Nachname" = 'Kästner' WHERE "Autor_Vorname" = 'Eric' AND "Autor_Nachname" = 'Käschtner'
```

Auch andere Rechenschritte sind beim Update möglich. Wenn z.B. Waren ab 150,- € zu einem Sonderangebot herausgegeben werden sollen und der Preis um 10% herabgesetzt werden soll geschieht das mit dem folgenden Befehl:

```
UPDATE "Tabellenname" SET "Preis" = "Preis"*0.9 WHERE "Preis" >= 150
```

Mit der Wahl des Datentyps CHAR wird eine fixe Breite festgelegt. Gegebenenfalls wird Text mit Leerzeichen aufgefüllt. Bei einer Umstellung auf VARCHAR bleiben diese Leerzeichen erhalten. Sollen die Leerzeichen entfernt werden, so gelingt dies mittels

```
UPDATE "Tabellenname" SET "Feldname" = RTRIM("Feldname")
```

**Bestehende Datensätze löschen**

```
DELETE FROM "Tabellenname" [WHERE <Expression>];
```

Ohne einen eingrenzenden Bedingungsausdruck wird durch

```
DELETE FROM "Tabellenname"
```

der gesamte Inhalt der Tabelle gelöscht.

Da ist es dann doch besser, wenn der Befehl etwas eingegrenzt ist. Wird z.B. der Wert des Primärschlüssels angegeben, so wird nur genau ein Datensatz gelöscht:

```
DELETE FROM "Tabellenname" WHERE "ID" = 5;
```

Sollen bei einer Medienausleihe die Datensätze von Medien, die zurückgegeben wurden, gelöscht werden, so geht dies mit

```
DELETE FROM "Tabellenname" WHERE NOT "RueckgabeDatum" IS NULL;
```

oder alternativ mit

```
DELETE FROM "Tabellenname" WHERE "RueckgabeDatum" IS NOT NULL;
```

**Import von Daten aus anderen Datenquellen**

Manchmal existieren schon komplette Datensammlungen in einem anderen Programm, die über die Zwischenablage in Base importiert werden sollen. Hier gibt es die Möglichkeit, beim Import eine neue Tabelle erstellen zu lassen oder auch Datensätze an eine bereits bestehende Tabelle anzufügen.
Hinweis
Um einen Import über die Zwischenablage zu gewährleisten muss das Datenformat für Base lesbar sein. Dies ist bei allen Daten der Fall, die in LibreOffice geöffnet zur Verfügung stehen.

Sollen also z. B. Tabellen aus einer externen Datenbank in eine *.odb-Datei eingelesen werden, so muss zuerst zu der externen Datenbank über LibreOffice ein Kontakt hergestellt werden. Diese Datenbank muss außerdem geöffnet oder als Datenquelle in LibreOffice angemeldet sein. Siehe hierzu im Kapitel «Datenbank erstellen» das Unterkapitel «Zugriff auf externe Datenbanken».


In dem Tabellencontainer mit der rechten Maustaste das Kontextmenü über der Tabelle aufgerufen, an die die Daten angehängt werden sollen.
Importierte Daten an bestehende Daten einer Tabelle anfügen


Spalten zuorden

Tabellen: Eingabe von Daten in Tabellen
Die Spalten der Quelltabelle aus Calc und der Zieltabelle in Base müssen nicht von der Reihenfolge her, vom Namen her oder von der Anzahl her übereinstimmen. Es werden nur die auf der linken Seite ausgewählten Elemente übertragen. Die Zuordnung zwischen Quelltabelle und Zieltabelle muss mit den danebenliegenden Pfeiltasten vorgenommen werden.

Danach wird der Import vollzogen.

Der Import kann Probleme bereiten, wenn

- Felder der Zieltabelle eine Eingabe erfordern, in der Quelltabelle hierfür aber keine Daten vorliegen,
- Felddefinitionen der Zieltabelle mit den Daten der Quelltabelle nicht vereinbar sind (wenn z. B. der Vorname in ein Zahlenfeld geschrieben werden soll oder das Feld der Zieltabelle zu wenig Zeichen zulässt) oder
- die Quelltabelle Daten vorgibt, die mit der Zieltabelle nicht vereinbar sind, wie z. B. eindeutige Werte beim Primärschlüssel oder in anderen Feldern, die so definiert wurden.

Neue Tabelle beim Import erstellen


An dieser Stelle kann ein neues, zusätzliches Datenbankfeld für einen Primärschlüssel erzeugt werden. Der Name dieses Datenbankfeldes darf nicht schon als Spalte in der Calc-Tabelle existieren. Anderenfalls erscheint die Fehlermeldung:

'Es sind bereits folgende Felder als Primärschlüssel gesetzt : ID'.

Diese Meldung gibt den Sachverhalt leider nicht ganz korrekt wieder.
Soll ein bereits vorhandenes Feld als Schlüsselfeld genutzt werden, lassen Sie «Primärschlüssel erzeugen» abgewählt. Das Festlegen als Primärschlüssel erfolgt in diesem Fall auf der dritten Dialogseite des Assistenten.

Beim Import soll die Definition der Tabelle erfolgen und die Daten übertragen werden.

Alle vorhandenen Spalten werden übernommen.

---

Tabellen: Eingabe von Daten in Tabellen 134
Bei der Formatierung des Tabellentyps muss meist nachgebessert werden. In der Regel werden die Felder als Textfelder mit sehr hoher Länge vordefiniert. Zahlen- und Datumsfelder sollten also auf jeden Fall unter **Typformatierungen → Spalteninformation → Feldtyp** entsprechend eingestellt werden. Bei Dezimalzahlen mit Nachkommastellen ist außerdem auf die Anzahl der Nachkommastellen zu achten.

Die Auswahl des Primärschlüssels liegt etwas versteckt im Kontextmenü des Feldes, das den Primärschlüssel erhalten soll. Hier wird gerade das Feld "ID" formatiert, das als Primärschlüssel vorgesehen ist. Der Primärschlüssel muss hier noch einmal über das Kontextmenü des Feldnamens gesondert ausgewählt werden, wenn er nicht durch den Assistenten im Fenster «Tabelle kopieren» als zusätzliches Feld erstellt wurde.

Nach der Betätigung des Buttons **Fertigstellen** wird die Tabelle erstellt und mit dem kopierten Inhalt gefüllt.


**Daten Aufsplitten beim Import**


Einen möglichen Weg, die Relation direkt zu erstellen, stellt die folgende Fassung dar:

Abfrage hierfür:

```
SELECT DISTINCT "PLZ", "Ort" FROM "Adresse"
```

3. Der Tabelle "Adresse" wird ein Feld "PLZ_ID" hinzugefügt.

4. Über Extras → SQL wird ein Update für diese Tabelle ausgeführt:

```
UPDATE "Adresse" AS "a" SET "a"."PLZ_ID" = (SELECT "ID" FROM "PLZ_Ort" WHERE "PLZ"||"Ort" = "a"."PLZ"||"a"."Ort")
```

5. Die Tabelle "Adresse" wird zum Bearbeiten geöffnet und die Felder "PLZ" und "Ort" gelöscht. Die Änderung wird gespeichert und anschließend die Tabelle wieder geschlossen.

Damit sind die Tabellen so getrennt, dass eine 1:n-Beziehung zwischen der Tabelle "PLZ_Ort" und der Tabelle "Adresse" erstellt werden kann. Diese Beziehung wird anschließend über Extras → Beziehungen definiert.

Für Details zum SQL-Code siehe auch das Kapitel «Abfragen».

### Mängel dieser Eingabemöglichkeiten

Eingaben mit einer Tabelle alleine berücksichtigt nicht die Verknüpfungen zu anderen Tabellen. Am Beispiel einer Medienausleihe sei das hier verdeutlicht:

Die Ausleihtabelle besteht aus Fremdschlüsseln für das auszuleihende Medium Medien_ID und den entsprechenden Nutzer Leser_ID sowie einem Ausleihdatum Leih_Datum. In die Tabelle werden also bei der Ausleihe zwei Zahlenwerte (Mediennummer und Benutzernummer) und ein Datum eingetragen. Der Primärschlüssel wird im Feld ID automatisch erstellt. Ob der Benutzer zu der Nummer passt bleibt unsichtbar, es sei denn, eine zweite Tabelle mit den Benutzern wird gleichzeitig offen gehalten. Ob das Medium mit der korrekten Nummer ausgeliehen wird ist genauso wenig einsehbar. Hier muss sich die Ausleihe auf das Etikett auf dem Medium oder auf eine weitere geöffnete Tabelle verlassen.

All dies lässt sich mit Formularen wesentlich besser zusammenfassen. Hier können die Nutzer und die Medien durch Listenfelder nachgeschlagen werden. Im Formular stehen dann sichtbar die Nutzer und die Medien, nicht die versteckten Nummern. Auch kann das Formular so aufgebaut werden, dass zuerst ein Nutzer ausgewählt wird, dann das Ausleihdatum eingestellt wird und jede Menge Medien diesem einen Datum durch Nummer zugeordnet werden. An anderer Stelle werden dann diese Nummern wieder mit entsprechender genauer Medienbezeichnung sichtbar gemacht.

Die Eingabe in Tabellen ist in Datenbanken daher nur bei einfachen Tabellen sinnvoll. Sobald Tabellen in Relation zueinander gesetzt werden, bietet sich besser ein entsprechendes Formular an. In Formularen können diese Relationen durch Unterformulare oder Listenfelder gut bedienbar gemacht werden.
Formulare
Formulare als Eingabeerleichterung

Formulare werden dann genutzt, wenn die Eingabe direkt über eine Tabelle zu unübersichtlich wird, eventuelle Fehleingaben rechtzeitig abgefangen werden sollen oder zu viele Tabellen eine direkte Verwaltung der Daten unmöglich machen.

Hinweis

Ein *Formular* ist in Base ein für den Nutzer *nicht sichtbares*Konstrukt. Es dient innerhalb von Base dazu, den Kontakt zur Datenbank zu ermöglichen.

Sichtbar sind für den Nutzer die *Kontrollfelder*, die dazu dienen, Text, Zahlen usw. einzugeben oder anzuzeigen. Diese Kontrollfelder werden über die GUI in verschiedene Feldarten unterteilt.


Erstellung von Formularen


Einfaches Formular

Als Start dient uns aus dem Formularbereich die Aufgabe *Formular in Entwurfsansicht erstellen*.

Rufen wir damit den Formulareditor auf, so zeigt sich erst einmal das Fenster *Formular in der Entwurfsansicht*.
Abbildung 20: Formular in der Entwurfsansicht


Symbolleisten des Formularentwurfs

Auf der leeren Fläche soll nun ein Formular entstehen. Dies kann auf verschiedene Arten geschehen:

- Aufruf des Formular-Navigators, von dort Gründung eines Formulars sowie
- Erstellung von Formularfeldern und Gründung des Formulars über das dortige Kontextmenü.

**Formulargründung über den Navigator**

Mit einem Rechtsklick auf das Verzeichnis Formulare öffnet sich ein Kontextmenü, in dem über den Menüpunkt **Neu** ein neues **Formular** erstellt werden kann. Die weiteren Befehle des Kontextmenüs entsprechen denen der Buttons in der Abbildung 21.

---

**Hinweis**

Soll ein Formular automatisch mit dem Cursor im ersten Formulareelement starten, so kann dies durch die Auswahl **Automatischer Kontrollelement-Fokus** bewirkt werden. Das erste Formularelement wird über die **Aktivierungsreihenfolge** geregelt.

Leider hat diese Funktion zur Zeit einen Bug. Befindet sich in einem Formular ein Tabellenkontrollfeld, so wird der Fokus automatisch auf das erste Feld des Tabellenkontrollfeldes gesetzt. Dieser Bug ist merkwürdigerweise zu beheben, wenn nach der Wahl des automatischen Kontrollelement-Fokus die Sprache der Benutzeroberfläche umgestellt wird: **Bug 87290**.

Das Formular erhält standardmäßig den Namen **Formular**. Diese Bezeichnung kann direkt oder später geändert werden. Sie hat allerdings für die spätere Funktion nur dann eine Bedeutung, wenn über Makros auf Teile des Formulars zugegriffen werden soll. Spätestens dann sollten nicht zwei Elemente mit gleicher Bezeichnung in der gleichen Ebene des Verzeichnisbaums auftauchen.

Über das Kontextmenü des Formulars geht schließlich der Weg zu den Formulareigenschaften.

---

**Formulargründung über ein Formularfeld**

Über die Symbolleiste für die Formularsteuerelemente (Abbildung 22) stehen direkt einige Formularfelder zur Verfügung. Während die ersten vier Elemente identisch zu denen des Formular-Entwurfs sind, folgen anschließend häufiger verwendete Formularfelder.

Über die Symbolleiste «Formular-Steuerelemente» kann die zusätzliche Symbolleiste «Weitere Steuerelemente» als frei schwebende Symbolleiste angezeigt werden.
Über den Aufruf eines Formularfeldes wird automatisch ein Formular mit gegründet:

Formulare: Erstellung von Formularen


**Externe Formulare**


### Formular-Eigenschaften

![Formular-Eigenschaften](image)

Werden die Formulareigenschaften über das Kontextmenü im Formular-Navigator oder das Kontextmenü eines Formularfeldes aufgerufen, so erscheint ein Fenster mit drei Registerreitern: Allgemein, Daten und Ereignisse

#### Allgemein

Für die Erstellung interner Formulare von Base ist dies der wichtigste Registerreiter. Hier wird zuerst der Inhalt des Formulars festgelegt.

- **Art des Inhalts**: Hier besteht die Wahl zwischen *Tabelle*, *Abfrage* und *SQL-Befehl*. Während Tabellen in der Regel immer für Eingaben in einem Formular genutzt werden können, so ist dies bei Abfragen eventuell nicht der Fall. Näheres dazu im Kapitel 'Abfragen'. Gleiches wie für Abfragen gilt auch für die direkten Eingabe eines SQL-Befehls. Hier handelt es sich dann lediglich um eine Abfrage, die nicht im Abfragecontainer von Base sichtbar ist, aber vom Prinzip her die gleiche Struktur aufweist.

- **Inhalt**: Wird unter Art des Inhaltes Tabelle oder Abfrage gewählt, so werden hier alle verfügbaren Tabellen und Abfragen gelistet. Soll ein SQL-Befehl erstellt werden, so besteht die Möglichkeit, den Abfrageeditor dazu über den Button mit den drei Punkten rechts von dem Inhaltsfeld aufzurufen.

- **SQL-Befehl analysieren**: Wird die Analyse des SQL-Befehls nicht zugelassen (weil z. B. mit Code gearbeitet wird, den die GUI eventuell nicht richtig deuten kann), so ist hier 'Nein' zu wählen. Allerdings schließt dies aus, dass das Formular weiterhin noch mit der Filterung oder mit der Sortierung auf die zugrundeliegenden Daten zugreifen kann.

- **Filter**: Hier kann ein Filter gesetzt werden. Hilfe dazu bietet ein Klick auf den Button rechts von dem Eingabefeld. Siehe auch das Kapitel 'Tabellen'.

- **Sortierung**: Hier kann eine Sortierung der Daten festgelegt werden. Hilfe dazu bietet ein Klick auf den Button rechts von dem Eingabefeld. Siehe auch hierzu das Kapitel 'Tabellen'.

- **Daten hinzufügen**: Sollen neue Daten erstellt werden können? Standardmäßig ist dies auf Ja eingestellt.

- **Daten ändern**: Sollen Daten geändert werden können? Ebenfalls Standard Ja.

- **Daten löschen**: Auch das Löschen von Daten wird standardmäßig ermöglicht.

- **Nur Daten hinzufügen**: Ist dies gewählt, so erscheint immer ein leeres Formular. Auf die alten Datensätze besteht kein Zugriff, sie können nicht bearbeitet oder auch nur angesehen werden.

- **Navigationsleiste**: Das Erscheinen der Navigationsleiste am unteren Bildschirmrand kann angeschaltet oder ausgeschaltet werden. Außerdem besteht die Möglichkeit, bei einem
Unterformular immer die Navigationsleiste des darüberliegenden Hauptformulars anzeigen zu lassen, so dass eine Betätigung der Navigationsleiste direkte Auswirkung auf das Hauptformular hat. Diese Einstellung zur Navigationsleiste betrifft nicht die Leiste, die gegebenenfalls als Formularfeld eingefügt werden kann.

- **Zyklus**: Standard bedeutet hier für Base-Datenbanken, dass nach der Eingabe im letzten Feld innerhalb eines Formulars mit dem Tabulator zum ersten Feld des nächsten Datensatzes, also gegebenenfalls eines neuen Datensatzes, gesprungen wird. Dies ist für die Datenbanken gleichbedeutend mit «Alle Datensätze». Wird hingegen bei Datenbanken «Aktueller Datensatz» gewählt, so bewegt sich der Cursor nur innerhalb des einen Datensatzes, beim Erreichen des letzten Feldes also zum ersten Feld des gleichen Datensatzes. Aktuelle Seite bezieht sich wieder besonders auf HTML-Formulare. Hier springt dann der Cursor vom Ende eines Formulars gegebenenfalls zum nächsten Formular auf der Seite, das weiter unten liegt.

**Ereignisse**

<table>
<thead>
<tr>
<th>Ereignis</th>
<th>Makrodivision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vor dem Zurücksetzen</td>
<td>org.openoffice.comp.svx.FormController</td>
</tr>
<tr>
<td>Nach dem Zurücksetzen</td>
<td>com.sun.star.comp.forms.ODatabaseForm</td>
</tr>
<tr>
<td>Vor dem Submit</td>
<td>oForm.ImplementationName</td>
</tr>
<tr>
<td>Beim Laden</td>
<td></td>
</tr>
<tr>
<td>Vor dem erneuten Laden</td>
<td></td>
</tr>
<tr>
<td>Beim erneuten Laden</td>
<td></td>
</tr>
<tr>
<td>Vor dem Entladen</td>
<td></td>
</tr>
<tr>
<td>Beim Entladen</td>
<td></td>
</tr>
<tr>
<td>Löschen bestätigen</td>
<td></td>
</tr>
<tr>
<td>Vor der Datensatzaktion</td>
<td></td>
</tr>
<tr>
<td>Nach der Datensatzaktion</td>
<td></td>
</tr>
<tr>
<td>Vor dem Datensatzwechsel</td>
<td></td>
</tr>
<tr>
<td>Nach dem Datensatzwechsel</td>
<td></td>
</tr>
<tr>
<td>Paramter füllen</td>
<td></td>
</tr>
<tr>
<td>Fehler aufgetreten</td>
<td></td>
</tr>
</tbody>
</table>


Fehler aufgetreten: Dieses Ereignis lässt sich nicht nachvollziehen.

Eigenschaften der Kontrollfelder

Ist ein Formular erstellt, so kann es mit den sichtbaren Kontrollfeldern bestückt werden. Die Kontrollfelder sind für verschiedene Aufgaben gedacht:

- Die meisten zeigen den Inhalt aus der Datenbank an oder nehmen die Daten entgegen, die in die Datenbank eingefügt werden.
- Andere Kontrollfelder dienen zur Navigation, zum Suchen und zur Ausführung von Befehlen (Interaktion).
- Weitere Kontrollfelder sorgen für eine zusätzliche grafische Aufarbeitung des Formulars.

<table>
<thead>
<tr>
<th>Dateneingabe und Datenanzeige</th>
<th>Anwendungsgebiet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kontrollfeld</strong></td>
<td><strong>Anwendungsgebiet</strong></td>
</tr>
<tr>
<td>Textfeld</td>
<td>Texteingaben</td>
</tr>
<tr>
<td>Numerisches Feld</td>
<td>Zahleneingabe</td>
</tr>
<tr>
<td>Datumsfeld</td>
<td>Datumseingabe</td>
</tr>
<tr>
<td>Zeitfeld</td>
<td>Zeiteingabe</td>
</tr>
<tr>
<td>Währungsfeld</td>
<td>Zahleneingabe, vorformatiert für Währungen</td>
</tr>
<tr>
<td>Formatiertes Feld</td>
<td>Anzeige und Eingabe mit zusätzlicher Formatierung wie z. B. Maßeinheiten</td>
</tr>
<tr>
<td>Listenfeld</td>
<td>Auswahl zwischen vielen verschiedenen Möglichkeiten, Weitergabe eines anderen als des angezeigten Wertes an die Datenbank.</td>
</tr>
<tr>
<td>Kombinationsfeld</td>
<td>Wie Listenfeld, nur Weitergabe des angezeigten Wertes und dazu noch die Möglichkeit, auch neue Werte einzugeben.</td>
</tr>
<tr>
<td>Markierfeld</td>
<td>Ja/Nein-Felder</td>
</tr>
<tr>
<td>Optionsfeld</td>
<td>Auswahl zwischen verschiedenen, stark begrenzten Möglichkeiten.</td>
</tr>
<tr>
<td>Grafisches Kontrollfeld</td>
<td>Anzeige von Bildern aus einer Datenbank und Neueingabe von Bildern in eine Datenbank über eine Pfadangabe</td>
</tr>
<tr>
<td>Maskiertes Feld</td>
<td>Eingabe in eine vorgefertigte Maske; grenzt die Eingabemöglichkeiten auf bestimmte Zeichenkombinationen ein.</td>
</tr>
<tr>
<td>Tabellen-Kontrollfeld</td>
<td>Universelles Eingabemodul, das ganze Tabelle bedienen kann. Integriert in dieses Kontrollfeld sind wieder viele der obigen Kontrollfelder</td>
</tr>
</tbody>
</table>
### Gestaltung

<table>
<thead>
<tr>
<th>Kontrollfeld</th>
<th>Anwendungsgebiet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftungsfeld</td>
<td>Überschrift über das Formular, Beschriftung anderer Kontrollfelder</td>
</tr>
<tr>
<td>Gruppierungsrahmen</td>
<td>Linienzug um z. B. verschiedene Optionsfelder</td>
</tr>
</tbody>
</table>

### Interaktion

<table>
<thead>
<tr>
<th>Kontrollfeld</th>
<th>Anwendungsgebiet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schaltfläche</td>
<td>Button mit Beschriftung</td>
</tr>
<tr>
<td>Grafische Schaltfläche</td>
<td>Wie der Button, nur mit einer zusätzlich auf dem Button erscheinenden Grafik</td>
</tr>
<tr>
<td>Navigationsleiste</td>
<td>Leiste mit geringen Abweichungen zu der, die am unteren Bildschirmrand angezeigt wird.</td>
</tr>
<tr>
<td>Dateiauswahl</td>
<td>Auswahl von Dateien z. B. zum Hochladen in HTML-Formularen. - nicht weiter beschrieben</td>
</tr>
<tr>
<td>Drehfeld</td>
<td>Nur über Makroprogrammierung verwendbar – nicht weiter beschrieben</td>
</tr>
<tr>
<td>Bildlaufleiste</td>
<td>Nur über Makroprogrammierung verwendbar – nicht weiter beschrieben</td>
</tr>
<tr>
<td>Verstecktes Steuerelement</td>
<td>Hier kann ein Wert über Makros eingespeichert und wieder ausgelesen werden.</td>
</tr>
</tbody>
</table>

### Standardeinstellungen vieler Kontrollfelder

Die Eigenschaften werden wie beim Formular in drei Kategorien unterteilt: Allgemein, Daten und Ereignisse. Unter Allgemein wird all das eingestellt, was für den Nutzer sichtbar ist. In der Kategorie Daten wird die Verbindung zu einem Feld der Datenbank hergestellt. Die Kategorie Ereignisse schließlich regelt Auslösemomente, die mit irgendwelchen Makros verbunden werden können. Für eine Datenbank ohne Makros spielt diese Kategorie keine Rolle.
In den folgenden Übersichten stehen die deutschen Begriffe vor allem der Eigenschaften – genau wie im Formular-Entwurf. In Makros werden zur Beeinflussung der Felder die englischen Begriffe benötigt; diese sind in [Klammern] angegeben.

Beispiel:
```python
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("Formular")
oFeld = oForm.getByName("Formularfeld")
oFeld.Printable = True  # Das Feld wird auf "Druckbar"→"Ja"
gesetzt.
```

Weitere Hinweise stehen im Kapitel «Makros».
Allgemein

Name: Textfeld 1

Beschriftungsfeld: 


Aktiviert: Ja

Nicht aktivierte Felder sind nicht verwendbar und werden grau hinterlegt. Sinnvoll bei Steuerung über Makros. (Entscheidung: Wenn in Feld 1 ein Wert eingegeben wird, darf in Feld 2 kein Wert eingegeben werden – Feld 2 wird deaktiviert)

Sichtbar: Ja

In der Regel 'Ja'; nicht sichtbare Felder können Werte zwischen speichern. Anwendung z. B. bei der Erstellung von Kombinationsfeldern mit Makros.

Nur lesen: Nein

'Ja' würde eine Veränderung des Wertes ausschließen; z. B. für die Anzeige eines automatisch erstellten Primärschlüssels sinnvoll.

Druckbar: Ja

Manchmal sind Seitenausdrucke aus einem Formular sinnvoller als ein separater Bericht. Hier sollen dann eventuell nicht alle Felder erscheinen.

Tabstop: Ja

Durch ein Formular wird in der Regel mit dem Tabulator navigiert. Ein Feld, das z. B. nur gelesen wird, braucht keinen Stop des Tabulators, würde also übersprungen.

Aktivierungsreihenfolge: 0

Hat das Feld einen Tabstop? Hier wird die Reihenfolge innerhalb des Formulars eingestellt.

Formulare: Formular-Eigenschaften
Verankerung der Grafik, die das Textfeld darstellt.

Position links oben vom linken Rand aus. [PosSize.X]

Position von oben aus. [PosSize.Y]

Breite des Feldes [PosSize.Width]

Höhe des Feldes [PosSize.Height]

Schriftart, Schriftschnitt, Schriftgrad und Schrifteffekt sind hier einstellbar. [Fontxxx]

Der Eintrag wird linksbündig dargestellt. [Align]

Standard | Oben | Mitte | Unten [VerticalAlign]

Hintergrundfarbe des angezeigten Textfeldes [BackgroundColor]

Rahmenform: Ohne Rahmen | 3D-Look | Flach [Border]

Wenn ein Rahmen, dann kann hier die Umrandungsfarbe eingestellt werden. [BorderColor]

Markierter Text wird so nicht mehr als markiert angezeigt, wenn das Textfeld den Fokus verliert. [HideInactiveSelection]

Gut nutzbar für Informationen, die mittels Makros ausgelesen werden sollen, siehe hierzu das Kapitel 'Makros'. [Tag]

Erscheint als sogenannter Tooltip, wenn mit der Maus über das Textfeld gefahren wird. [HelpText]
Verweist auf eine Hilfedatei – eher für HTML-Zwecke geeignet. Durch F1 abrufbar, wenn der Fokus auf dem Feld liegt.

[HelpURL]

Zusätzlich sind bei Zahlenfeldern, Datumsfeldern u.ä. üblich:

Formatüberprüfung .......... Ja

Mit eingeschalteter Überprüfung ist nur die Eingabe von Ziffern und Komma möglich.
[EnforceFormat]

Mausradverhalten .......... Nie

Nie erlaubt keine Änderung mit dem Mausrad; Wenn ausgewählt lässt eine Änderung zu, wenn das Feld ausgewählt ist und die Maus sich über dem Feld befindet; Immer bedeutet, dass sich die Maus über dem Feld befinden muss.
[MouseWheelBehavior]

Drehfeld .................. Nein

Ein Drehsymbol wird an der rechten Seite des Feldes eingeblendet.
[Spin]

Wiederholung .............. Nein

Wenn ein Drehfeldpfeil länger gedrückt wird lässt sich hier einstellen, ob nicht nur zum nächsten Wert gedreht werden soll.
[Repeat]

Verzögerung .............. 50 ms

Stellt die Verzögerungszeit ein, nach der der Druck auf die Maustaste beim Drehfeld Wiederholung interpretiert wird.
[RepeatDelay]

Daten

[Formular-Eigenschaften]

Datenfeld .................. Nachname
Leere Zeichenfolge ist NULL .......... Ja
Eingabe erforderlich .......... Ja
Filtervorschlag .............. Nein

Formulare: Formular-Eigenschaften
**Datenfeld:** Hier wird die Verbindung zur Tabelle hergestellt, die dem Formular zugrunde liegt.

*[Model.DataField]*

**Leere Zeichenfolge ist NULL:** Soll ein leeres Feld geleert werden (NULL) oder nur der Inhalt gelöscht werden?

**Eingabe erforderlich:** Dieser Eintrag sollte sich mit dem in der Tabelle decken. Dann fragt die GUI gegebenenfalls nach einer Eingabe, wenn vom Nutzer kein Wert eingegeben wurde.

*[Model.InputRequired]*

**Filtervorschlag:** Bei einer Filterung der Daten werden die Inhalte dieses Feldes als Vorschläge zwischengespeichert. Achtung – dies ist bei umfangreichen Inhalten recht speicherintensiv.

*[Model.UserValueFilterProposal]*

---

**Ereignisse**

*Modifiziert:* Dieses Ereignis tritt ein, wenn das Kontrollfeld geändert wurde und anschließend den Fokus verloren hatte. Das Ereignis geht verloren, wenn direkt zum nächsten Datensatz gewechselt wird. Unter den Umständen ist also eine Änderung gespeichert ohne vorher wahr-genommen zu werden. [com.sun.star.lang.EventObject]

*Text modifiziert:* Direkt auf den Inhalt bezogen; kann Text, Zahl o.a. sein, tritt also nach jeder Tastatureingabe auf. [com.sun.star.awt.TextEvent]

*Fokuserhalt:* Der Cursor kommt in das Feld hinein. Hier sollte auf keinen Fall über das Makro eine Messagebox auf dem Bildschirm erzeugt werden. Durch das Anklicken dieser Box verliert das Formularfeld den Fokus und erhält ihn direkt danach zurück – eine Schleife wird dadurch erzeugt. Sie kann nur durch Tastatureingaben unterbrochen werden.
Fokusverlust: Der Cursor bewegt sich aus dem Feld heraus. Auch dies kann zu einem Wechselspiel führen, wenn eine zu bestätigende Bildschirmausgabe erfolgt.


Maus: selbsterklärend; Ereignisse treten nur ein, wenn vorher die Maus innerhalb des Feldes ist oder war («außerhalb» entspricht javascript onMouseOut). [com.sun.star.awt.MouseEvent]

Zurücksetzen: Das Formular wird von allen Daten geleert (Anlegen eines neuen Datensatzes) oder auf den alten Datenstand zurück gesetzt (Änderung eines bestehenden Datensatzes). Bei einem Formularfeld wird das Ereignis nur ausgelöst, wenn über den Button in der Navigationsleiste die Dateneingabe rückgängig gemacht wird. [com.sun.star.lang.EventObject]

Wenn ein Formular aufgerufen wird, wird nacheinander das Ereignis Vor dem Zurücksetzen und Nach dem Zurücksetzen abgearbeitet, bevor das Formular für eine Eingabe verfügbar ist.


Textfeld
Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

Allgemein

Max. Textlänge

Solange der Wert 0 ist wird die Einstellung nicht berücksichtigt. In der Praxis wird hier die Zeichenlänge des Feldes aus der Datenbank übernommen, auf das sich das Textfeld bezieht. [MaxTextLen]

Standardtext

Soll in einem leeren Feld ein Standardtext erscheinen? Dieser Text muss gelöscht werden, wenn ein anderer Eintrag erfolgen soll. [DefaultText]

Text-Typ

Möglichere Typen: Einzeilig | Mehrzeilig | Mehrzeilig mit Formatierungen (wobei sich die beiden letzten nur im Tabulator unterscheiden – und ein Feld mit Formatierungen nicht an eine Datenbank angebunden werden kann). Bei mehrzeiligen Feldern ist die vertikale Ausrichtung nicht aktiv. [MultiLine]

Nur bei mehrzeiligen Feldern: Horizontal | Vertikal | Beide [HScroll], [VScroll]

Aktiv nur bei einzeiligen Feldern. [EchoChar]

**Daten**

keine weiteren Besonderheiten

**Ereignisse**

keine weiteren Besonderheiten

**Numerisches Feld**

Neben den bereits unter *Standardeinstellungen vieler Kontrollfelder* erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

Mindestwert, den dieses Feld einnehmen kann. Sollte mit dem Mindestwert übereinstimmen, der in der Tabelle erwartet wird. [ValueMin]

Maximalwert [ValueMax]

Intervall-Wert für die Funktion als Scrollelement per Mausrad bzw. Drehfeld [ValueStep]

Wert, der beim Erstellen eines neuen Datensatzes angezeigt wird. [DefaultValue]

Anzahl Nachkommastellen, bei Integer-Feldern auf 0 zu stellen [DecimalAccuracy]

Trennzeichen für Tausenderstellen, in der Regel der Punkt [ShowThousandsSeparator]
Daten
Es erfolgt keine Nachfrage, ob das Feld NULL sein soll. Eine fehlende Eingabe belässt dennoch das Feld auf NULL, nicht auf 0. Ein Filtervorschlag wird ebenfalls nicht erstellt.

Ereignisse
Es fehlt das Feld «Modifiziert». Änderungen werden über «Text modifiziert» (hier wohl nicht wörtlich zu nehmen) angesprochen.

**Datumsfeld**
Neben den bereits unter **Standardeinstellungen vieler Kontrollfelder** erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. Datum</td>
<td>01.01.1800, einstellbar über ein Aufklappfeld, der einen Kalender bereitstellt.</td>
</tr>
<tr>
<td>Max. Datum</td>
<td>31.12.2200</td>
</tr>
<tr>
<td>Datumsformat</td>
<td>Standard (kurz)</td>
</tr>
<tr>
<td>Standarddatum</td>
<td>Hier kann ein festes Datum vorgegeben werden. Das aktuelle Datum (Heute) beim Aufruf des Formulars muss leider (noch) durch ein Makro eingetragen werden.</td>
</tr>
<tr>
<td>Aufklappbar</td>
<td>Nein</td>
</tr>
</tbody>
</table>

**Daten**
Es erfolgt keine Nachfrage, ob das Feld NULL sein soll. Eine fehlende Eingabe belässt dennoch das Feld auf NULL, nicht auf 0. Ein Filtervorschlag wird ebenfalls nicht erstellt.

**Ereignisse**
Es fehlt das Feld «Modifiziert». Änderungen werden über «Text modifiziert» (hier wohl nicht wörtlich zu nehmen) angesprochen.

**Zeitfeld**
Neben den bereits unter **Standardeinstellungen vieler Kontrollfelder** erklärten Eigenschaften gibt es die folgenden Besonderheiten:
Allgemein

Min. Zeit ................... 00:00:00 
Mindestwert für das Feld, standardmäßig auf 0 gesetzt. [TimeMin]

Max. Zeit ................... 23:59:59 
Maximalwert, standardmäßig auf 1 Sekunde unter 24 Uhr gesetzt. [TimeMax]

Zeitformat ................... 13:45 
Kurzform ohne Sekunden, Langform mit Sekunden sowie Benennungen mit PM (post meridiem / Nachmittag) [TimeFormat]

Eine feste Zeit ist voreinstellbar, die aktuelle Zeit beim Abspeichern des Formulars leider (bisher) nur mit Makro. [DefaultTime]

Daten

Es erfolgt keine Nachfrage, ob das Feld NULL sein soll. Eine fehlende Eingabe belässt dennoch das Feld auf NULL, nicht auf 0. Ein Filtervorschlag wird ebenfalls nicht erstellt.

Ereignisse

Es fehlt das Feld «Modifiziert». Änderungen werden über «Text modifiziert» (hier wohl nicht wörtlich zu nehmen) angesprochen.

Währungsfeld

Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

Allgemein

Min. Wert, Max. Wert, Intervall, Standardwert, Nachkommastellen und Tausender-Trennzeichen entsprechen den allgemeinen Eigenschaften Numerisches Feld. Daneben gibt es lediglich:

Währungssymbol ........... € 
Das Symbol wird angezeigt, aber nicht in der dem Formular zugrunde liegenden Tabelle mit abgespeichert. [CurrencySymbol]

Symbol voranstellen ....... Nein 
Soll das Symbol vor oder hinter der Zahl erscheinen? [PrependCurrencySymbol]

Daten

Es erfolgt keine Nachfrage, ob das Feld NULL sein soll. Eine fehlende Eingabe belässt dennoch das Feld auf NULL, nicht auf 0. Ein Filtervorschlag wird ebenfalls nicht erstellt.
Ereignisse
Es fehlt das Feld «Modifiziert». Änderungen werden über «Text modifiziert» (hier wohl nicht wörtlich zu nehmen) angesprochen.

Formatiertes Feld
Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

Allgemein

Über den rechten Button mit den drei Punkten erscheint die Auswahl zum Zahlenformat, die auch in LibreOffice-Calc üblich ist.

Abbildung 24: Formatiertes Feld mit allgemeinen Zahleneinstellungen

In dem Zahlenformat sind dann neben Datum, Zeit, Währung oder normalen Zahlenformaten auch Möglichkeiten gegeben, die Felder mit einer Maßeinheit wie z. B. kg zu bestücken. Siehe dazu die allgemeine Hilfe zu Zahlenformat-Codes.

Mit dem formatierten Feld ist es auch möglich, Timestamp-Felder aus Tabellen mit nur einem Feld darzustellen und zu beschreiben. Der Formularassistent nutzt dazu sonst ein Datumsfeld und ein Zeitfeld.

**Daten**
keine weiteren Besonderheiten.

**Ereignisse**
Es fehlt das Feld «Modifiziert». Änderungen werden über «Text modifiziert» (hier wohl nicht wörtlich zu nehmen) angesprochen.

**Listenfeld**
Sobald ein Listenfeld erstellt wird, erscheint standardmäßig der Listenfeldassistent. Diese Automatik lässt sich gegebenenfalls über Assistenten an/aus (Abbildung 22) abschalten.

**Assistent**


In der Tabelle "Ausleihe" soll verzeichnet werden, welcher "Leser" welche Medien entliehen hat. Allerdings wird in dieser Tabelle nicht der Name des Lesers gespeichert sondern der Primärschlüssel aus der Tabelle "Leser". Die Tabelle "Leser" bildet also die Grundlage für das Listenfeld.
Das Feld "Nachname" aus der Tabelle "Leser" soll in dem Listenfeld sichtbar sein. Es ist also das Anzeigefeld.

Das Feld "Leser_ID" befindet sich in der dem Formular zugrundeliegenden Tabelle "Ausleihe". Diese Tabelle wird hier als Wertetabelle bezeichnet. Der Primärschlüssel "ID" aus der Tabelle...
"Leser" soll mit diesem Feld verbunden werden. Die Tabelle "Leser" wird hier als Listentabelle bezeichnet.

Das Listenfeld ist jetzt komplett mit Daten und Standardeinstellungen erstellt und funktionsfähig.

Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

Die Listeneinträge wurden bereits durch den Assistenten erstellt. Hier könnten sonst auch Einträge stehen, die in keiner Tabelle der Datenbank verzeichnet sind. Mit Listeneinträgen sind hier die sichtbaren Einträge gemeint; nicht die Einträge, die über das Formular an die Tabelle weitergegeben werden.

- **Wird das Feld als nicht aufklappbar festgelegt, so erscheinen bei Aufruf des Formulars am rechten Rand des Listenfeldes Scrollpfeile. Das Listenfeld wird dann automatisch zu einem mehrzeiligen Feld, bei dem die aktuelle Auswahl markiert ist.** [Dropdown]
- **Falls aufklappbar wird hier die maximal sichtbare Anzahl der Zeilen eingegangen. Geht der Inhalt über mehr Zeilen, so erscheint beim Aufklappen ein Scrollbalken.** [LineCount]
Schon der deaktivierte Button macht deutlich, dass auch eine Standardselection bei einer Verbindung mit einer Datenbanktabelle, wie über den Listenfeldassistenten erzeugt, wenig Sinn macht. Schließlich kann es passieren, dass der der Standardselection entsprechende Datensatz in der Beispieltabelle "Leser" gar nicht mehr vorhanden ist.

Daten

Neben den üblichen Daten-Eigenschaften Datenfeld und Eingabe erforderlich sind hier Eigenschaften von Bedeutung, die die Verbindung von anzuzeigenden Daten und in die dem Formular zugrundeliegende Tabelle einzutragenden Daten herstellen.

Art des Listeninhaltes: Werteliste | Tabelle | Abfrage | SQL | SQL (Native) | Tabellenfelder [ListSourceType]

Listeninhalt Werteliste: Sind unter 'Allgemein' Listeneinträge gemacht worden, so werden hier die entsprechenden Werte eingegeben, die an mit dem Formular abgespeichert werden sollen. Der Listeninhalt wird bestückt, indem bei der Eingabe die Inhalte über Shift – Enter aneinandergehängt werden. Es erscheint dann in dem Listeninhalt "Wert1";"Wert2";"Wert3" ... Die Eigenschaft 'Gebundenes Feld' ist inaktiv.

Listeninhalt Tabelle: Hier wird eine der Datenbanktabellen ausgesucht. Dabei muss aber berücksichtigt werden, dass dann immer das erste Feld der Tabelle in dem Listenfeld angezeigt wird.

Üblicherweise sind Tabellen so aufgebaut, dass zuerst der Primärschlüssel erscheint und anschließend die Inhalte. Solche Tabellen eignen sich nur dann für ein Listenfeld, wenn tatsächlich der Primärschlüssel angezeigt werden soll.

Listeninhalt Abfrage: Hier kann extern eine Abfrage vorformuliert werden, die dann auch als Abfrage sichtbar abgespeichert wird. Die Gestaltung solcher Abfragen wird im Kapitel 'Abfragen' erklärt. Durch die Abfrage ist es dann möglich, das Feld "ID" von der ersten Position in der ursprünglichen Tabelle an die zweite Position zu bewegen, was hier dem gebundenen Feld 1 entspricht.

Listeninhalt SQL: Hiermit bestückt der Listenfeldassistent das Listenfeld. Die von dem Assistenten konstruierte Abfrage sieht dann so aus:
Die Abfrage ist die einfachste Möglichkeit, die sich bietet. Das Feld "Nachname" erscheint an Position 0, das Feld "ID" an Position 1. Beide werden aus der Tabelle "Leser" ausgelesen. Da das gebundene Feld das Feld 1 ist reicht diese SQL-Formulierung aus. Schon vom Standard her sollte aber hier ergänzt werden ORDER BY "Nachname" ASC, denn ohne diese Formulierung werden die Nachnamen so wiedergegeben, dass ihre Reihenfolge der Eingabe in die Tabelle Leser entspricht. Zusätzliches Problem dürfte sein, dass "Nachname" bei einigen Lesern gleich ist. Hier muss dann "Vorname" hinzugezogen werden können und, als letztes Mittel, wohl auch der Primärschlüssel "ID". Wie so etwas genauer formuliert wird, kann im Kapitel 'Abfragen' genauer nachgelesen werden.


Listeninhalt Tabellenfelder: Hier werden die Feldnamen einer Tabelle aufgelistet, nicht die Inhalte. Für die Tabelle "Leser" erscheint dort also die Liste "ID", "Vorname", "Nachname", "Sperre", "Geschlecht_ID".

Hinweis
Für Zeitfelder, die auch Zeiten im Millisekunden-Bereich darstellen sollen, sind, wie unter «Zeitfelder in Tabellen» beschrieben, Timestamp-Felder erforderlich. Die Darstellung der Millisekunden funktioniert bei der Zusammenfügung von Zeichen in einem Listenfeld nicht. Hier muss der Timestamp in Text umgewandelt werden:

```
SELECT REPLACE(LEFT(RIGHT(CONVERT("Leistung_erforderlich"."Zeit", VARCHAR),15),8),'.','.','') AS "Listinhalt", "ID"
FROM "Leistung_erforderlich"
```

Dies erzeugt die Anzeige in Minuten:Sekunden,Hundertstelsekunden.

Gebundenes Feld: Im Listenfeld wird ein Inhalt angezeigt, der nicht mit dem Inhalt identisch sein muss, der auch im Formular abgespeichert werden soll. Üblicherweise wird z. B. Ein Name angezeigt und der Primärschlüssel zu diesem Namen als abzuspeicherndes Feld bestimmt.

```
SELECT "Name", "ID" FROM "Tabelle" ORDER BY "Name"
```

Das Feld "ID" wird als Fremdschlüssel in der Tabelle gespeichert, die dem Formular zugrunde liegt. Die Zählung der Felder in Datenbanken beginnt mit 0, das an das Formular gebundene Feld hat also die Position 1.

Wird stattdessen «0» gewählt, dann wird in dem Formular der Inhalt des Feldes "Name" abgespeichert. Das Feld "ID" könnte in der Abfrage entfallen.

Es ist sogar möglich, die Position «-1» zu wählen. Dann wird nicht der Inhalt der Abfrage sondern die Position des Eintrages in der Liste gespeichert. Der erste Datensatz hat dabei die Position 1.

Die oben aufgeführte Abfrage ergäbe z. B. die folgenden Daten:
Name | ID
---|---
Anneliese | 2
Dorothea | 3
Sieglinde | 1

Im Listenfeld könnte nur der Name ausgewählt werden. Das Listenfeld wird auf den Namen «Dorothea» eingestellt. Folgende Abspeicherungen würden über das Listenfeld möglich: Gebundenes Feld = 1 → «3» wird abgespeichert, da das Feld "ID" gespeichert wird. Gebundenes Feld = 0 → «Dorothea» wird abgespeichert, da das Feld "Name" gespeichert wird. Gebundenes Feld = -1 → «2» wird abgespeichert, da «Dorothea» als zweiter Datensatz in der Liste aufgeführt wird.

**Hinweis**

Die Änderung des gebundenen Feldes auf «0» oder «-1» ist erst mit der Version LO 4.1 eingeführt worden. Vorher war nur die Auswahl von Werten >= 1 möglich.

**Ereignisse**

Neben den Standardereignissen werden folgende Ereignisse hinzugefügt:

Aktion ausführen: Wird durch Tastatureingabe ein Listenfeld dazu veranlasst, einen entsprechenden Inhalt anzuzeigen, so ist dies eine Aktion, die das Listenfeld ausführt. Auch die Auswahl des Wertes aus einem Listenfeld entspricht dieser Aktion.

Status geändert: Dies kann die Änderung des angezeigten Inhaltes eines Listenfeldes durch Betätigen des Aufklappbuttons sein. Es kann auch lediglich ein Klicken auf den Aufklappbutton des Feldes sein.

Fehler aufgetreten: Das Ereignis lässt sich leider beim Listenfeld nicht nachvollziehen.

**Kombinationsfeld**

Sobald ein Kombinationsfeld erstellt wird erscheint wie beim Listenfeld standardmäßig ein Assistent. Diese Automatik lässt sich gegebenenfalls über Assistenten an/aus (Abbildung 22) abschalten.

Kombinationsfelder schreiben direkt den ausgewählten Wert in die dem Formular zugrundeliegende Tabelle. Deshalb ist im folgenden Beispiel die dem Formular zugrundeliegende Tabelle die Tabelle "Leser" und die für das Kontrollfeld gewählte Tabelle ebenfalls die Tabelle "Leser".
Das Formular ist wieder vordefiniert, diesmal als Beispiel mit der Tabelle "Leser". Da die Daten, die in dem Listenfeld angezeigt werden, auch in dieser Tabelle abgespeichert werden sollen, wird als Quelle für die Daten des Listenfeldes ebenfalls die Tabelle "Leser" ausgewählt.

**Kombinationsfeld-Assistent - Tabellenauswahl**

*Formular*

Art des Inhaltes: Tabelle
Inhalt: Leser

*Kontrollfeld*

Auf der rechten Seite sehen Sie alle Tabellen aus der Datenquelle des Formulars. Wählen Sie bitte die Tabelle, deren Daten die Grundlage für den Listeninhalt bilden sollen.

Das Formular ist wieder vordefiniert, diesmal als Beispiel mit der Tabelle "Leser". Da die Daten, die in dem Listenfeld angezeigt werden, auch in dieser Tabelle abgespeichert werden sollen, wird als Quelle für die Daten des Listenfeldes ebenfalls die Tabelle "Leser" ausgewählt.

**Kombinationsfeld-Assistent - Feldauswahl**

**Vorhandene Felder**

ID
Vorname
Nachname
Spree
Geschlecht_ID

**Anzeigefeld**

Vorname

Der Inhalt des ausgewählten Feldes wird in der Liste des Kombinationsfeldes angezeigt.
In der Tabelle "Leser" befindet sich das Feld "Vorname". Es soll im Kombinationsfeld angezeigt werden.

Innerhalb einer Datenbank scheint es wenig Sinn zu machen, den Wert des Kombinationsfeldes nicht in einem Feld zu speichern. Aus der Tabelle "Leser" sollen die Vornamen ausgelesen werden und für neue Leser als Vornamen zur Verfügung stehen, damit bei gleichen Vornamen eben nicht immer wieder eine neue Eingabe erfolgen muss. Das Kombinationsfeld zeigt dann hier den Vornamen an und die Texteingabe braucht nicht weiter durchgeführt zu werden.

Soll ein neuer Wert eingegeben werden, so ist dies im Kombinationsfeld ohne weiteres möglich, denn das Kombinationsfeld zeigt genau das an, was es selbst an die zugrundeliegende Tabelle des Formulars weitergibt.

Neben den bereits unter Standardeinstellungen vieler Kontrollfelder und beim Listenfeld erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

Automatisch füllen............ Ja

Während der Eingabe von neuen Werten wird eine Liste der eventuell zutreffenden Werte angezeigt, die noch zur Auswahl stehen. [AutoComplete]
Daten

Die Datenfeldern entsprechen den bereits vorgestellten Standardeinstellungen und den Einstellungen bei einem Listenfeld. Der SQL-Befehl weist hier allerdings eine Besonderheit auf:

```
SELECT DISTINCT "Vorname" FROM "Leser"
```

Durch den Zusatz des Begriffes DISTINCT werden gleiche Vornamen nur einmal angezeigt. Wieder fehlt allerdings bei der Erstellung durch den Assistenten eine Sortierung der Inhalte.

Ereignisse

Die Ereignisse entsprechen denen beim Listenfeld.

**Markierfeld**

Das Markierfeld erscheint direkt als eine Kombination von Markierbox und Beschriftung dieser Box. Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

<table>
<thead>
<tr>
<th>Eigenschaft</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titel</td>
<td>Markierfeld</td>
</tr>
<tr>
<td>Standardstatus</td>
<td>Nicht ausgewählt</td>
</tr>
<tr>
<td>Wortumbruch</td>
<td>Nein</td>
</tr>
</tbody>
</table>

Die Beschriftung dieser Box, einschließlich der Beschriftung der Box selbst, erscheint standardmäßig rechts von der Box. Zusätzlich ist noch eine Verbindung zu einem Beschriftungsfeld möglich.


Standardmäßig wird der Titel nicht umgebrochen. Ein zu langer Titel wird abgeschnitten, wenn das Feld nicht groß genug ist.
Hier kann eine Grafik statt bzw. zusätzlich zum Titel eingefügt werden. Dabei sollte beachtet werden, dass die Grafiken standardmäßig erst einmal nicht in das *.odb-Dokument eingebunden werden. Sinnvoll bei kleinen Grafiken ist es, die Grafiken einzubetten und nicht zu verknüpfen.

Ist eine Grafik ausgewählt, so kann sie im Verhältnis zum Titel ausge richtet werden.

Standardmäßig gibt es für Markierfelder nur Ausgewählt (Wert: 1) und Nicht ausgewählt (Wert: 0). Beim dreifachen Status kommt die Definition als leeres Feld (NULL) hinzu.

Daten

Dem Markierfeld kann ein Referenzwert zugewiesen werden. Allerdings wird an das darunter liegende Datenfeld nur der Wert 1 für (ein) und 0 für (aus) weitergegeben. Schließlich werden Markierfelder als Felder für die Auswahl von ‘Ja’ und ‘Nein’ genutzt.

Wird ein Markierfeld an ein Textfeld gebunden, so wird an das Textfeld ab der Version LO 4.2 standardmäßig «true» und «false» weiter gegeben. Allerdings liest das Markierfeld auch die Werte 1 und 0 korrekt aus. Eine klare Wahl des Referenzwertes als 1 und 0 dient dann dazu, auch 1 und 0 in die Felder zu schreiben.

Ereignisse

Es fehlen die Felder «Modifiziert», «Text modifiziert», «Vor der Aktualisieren» und «Nach dem Aktualisieren».

Zusätzlich erscheinen «Aktion ausführen» (siehe Listenfeld) und «Status geändert» (entspricht «Modifiziert»).
Optionsfeld
Das Optionsfeld entspricht bis auf eine Ausnahme in den allgemeinen Eigenschaften und die äußerlich andere Form (rund) dem eben beschriebenen Markierfeld. Werden mehrere Optionsfelder mit einem Tabellenfeld über das Formular verbunden, so kann von den Optionen immer nur eine Option gewählt werden.

Allgemein

[GroupName]

Daten
Siehe Markierfeld, hier werden aber die Referenzwerte, die eingetragen wurden, auch tatsächlich an das Datenfeld weitergegeben.

Ereignisse
Siehe Markierfeld

Grafisches Kontrollfeld

Vorsicht
Bilder sollten auf jeden Fall verkleinert werden, wenn sie schon in der Datenbank gespeichert werden. Hat ein Kamerabild z.B. 3 MB, so passiert es sehr schnell bei Nutzung der internen HSQLDB, dass mit Angabe einer Java-Fehlermeldung (NullPointerException) die Abspeicherung der Daten nicht mehr möglich ist. Dies kann zum Verlust sämtlicher Daten führen, die seit dem Öffnen der Base-Datenbank in die Tabelle eingegeben wurden.

Die Eingabe in das grafische Kontrollfeld erfolgt entweder über einen Doppelklick mit der Maus; dann erscheint ein Dateiauswahlfenster. Oder sie erfolgt über einen Klick der rechten Maustaste; dann kann ausgewählt werden, ob die Grafik nur gelöscht oder eine andere geladen werden soll.

Ein Grafisches Kontrollfeld erhält standardmäßig erst einmal keinen Tabstop.

Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

Allgemein
Die hier ausgesuchte Grafik wird nur im Kontrollfeld angezeigt, wenn das Formular bearbeitet wird. Für die spätere Eingabe ist sie ohne Bedeutung.

[Graphic]
Nein: Das Bild wird nicht an das Kontrollfeld angepasst. Ist das Bild zu groß, so wird ein Ausschnitt des Bildes gezeigt. Das Bild wird nicht verzerrt.

Seitenverhältnis beibehalten: Das Bild wird in das Kontrollfeld einge passt, aber nicht verzerrt dargestellt.


Daten
keine weiteren Besonderheiten

Ereignisse
Es fehlen die Ereignisse «Modifiziert», «Text modifiziert», «Vor dem Aktualisieren» und «Nach dem Aktualisieren».

**Maskiertes Feld**

Neben den bereits unter **Standardeinstellungen vieler Kontrollfelder** erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

<table>
<thead>
<tr>
<th>Eingabemaske……………</th>
<th>Hier wird festgelegt, welche Inhalte eingegeben werden können.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[EditMask]</td>
<td>[LiteralMask]</td>
</tr>
<tr>
<td>Zeichenmaske……………</td>
<td>Diese Ansicht wird dem Formularnutzer präsentiert.</td>
</tr>
</tbody>
</table>

Der folgende Inhalt entstammt der LibreOffice-Hilfe:

Die Länge der Eingabemaske bestimmt, wie viele Stellen die mögliche Eingabe haben darf. Sollten die vom Benutzer eingegebenen Zeichen der Eingabemaske nicht entsprechen, so wird die Eingabe beim Verlassen des Kontrollfelds verworfen. Zur Definition der Eingabemaske stehen die folgenden Zeichen zur Verfügung:
<table>
<thead>
<tr>
<th>Zeichen</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Es können die Buchstaben a-z und A-Z eingegeben werden. Großbuchstaben werden nicht in Kleinbuchstaben konvertiert.</td>
</tr>
<tr>
<td>A</td>
<td>Es können die Buchstaben A-Z eingegeben werden. Eingegebene Kleinbuchstaben werden automatisch in Großbuchstaben konvertiert.</td>
</tr>
<tr>
<td>c</td>
<td>Es können die Buchstaben a-z und A-Z sowie die Ziffern 0-9 eingegeben werden. Großbuchstaben werden nicht in Kleinbuchstaben konvertiert.</td>
</tr>
<tr>
<td>C</td>
<td>An dieser Stelle können die Zeichen A-Z und 0-9 angegeben werden. Wird ein Kleinbuchstabe angegeben, wird automatisch in Großschrift umgewandelt.</td>
</tr>
<tr>
<td>N</td>
<td>Es können nur die Zeichen 0-9 angegeben werden.</td>
</tr>
<tr>
<td>x</td>
<td>Es können alle druckbaren Zeichen angegeben werden.</td>
</tr>
<tr>
<td>X</td>
<td>Es können alle druckbaren Zeichen angegeben werden. Wird dabei ein Kleinbuchstabe verwendet, wird automatisch in Großschrift umgewandelt.</td>
</tr>
</tbody>
</table>

Definieren Sie z. B. für die Zeichenmaske «___.__.2000» die Eingabemaske «NNLNNLLLLL», um dem Benutzer nur noch die Eingabe von 4 Ziffern zur Datumsangabe zu ermöglichen.

**Daten**
keine weiteren Besonderheiten

**Ereignisse**
Es fehlt das Ereignis «Modifiziert»

**Tabellen-Kontrollfeld**
Dieses Kontrollfeld ist das umfassendste Feld. Es stellt eine Tabelle bereit, die wiederum durch Kontrollfelder für die einzelnen Spalten beschickt werden kann. Damit ist bei der Eingabe nicht nur der Blick auf die aktuell einzugebenden Daten, sondern auch auf vorher eingegebene Daten möglich, ohne mit der Navigationsleiste durch die Datensätze zu scrollen.

Nicht alle Felder, die auch im Formular möglich sind, können im Tabellen-Kontrollfeld wieder ausgewählt werden. So fehlen z. B. Schaltflächen, das grafische Kontrollfeld und das Optionsfeld.

Ein Assistent für das Tabellen-Kontrollfeld stellt in einem Fenster die Felder zusammen, die anschließend in der Tabelle zur Verfügung stehen sollen.
In dem Kontrollfeld soll die Tabelle "Ausleihe" zur Bearbeitung zur Verfügung stehen. Bis auf die Felder "ID" (Primärschlüssel) und das Feld "Medien_ID_BC" (Eingabe der Medien über einen Barcodescanner) sollen alle Felder in dem Kontrolelement benutzt werden.


Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:
Allgemein

In welcher Höhe sollen die einzelnen Zeilen angezeigt werden. Ohne einen Wert wird die Höhe automatisch der Schriftgröße angepasst. Mehrzeilige Textfelder würden so nur einzeilig zum Scrollen angezeigt. [RowHeight]

Wie bei Tabellen werden am unteren Rand des Kontrollfeldes die Datensatznummer und Navigationshilfen angezeigt. [HasNavigationBar]

Am linken Rand des Kontrollfeldes befindet sich standardmäßig der Datensatzmarkierer. Er zeigt den aktuellen Datensatz an. Über den Datensatzmarkierer wird die Löschfunktion für den gesamten Datensatz zur Verfügung gestellt. [HasRecordMarker]

Daten
Da es sich um ein Feld handelt, das selbst keine Daten anzeigt oder abspeichert, sondern stattdessen andere Felder verwaltet, die dies tun, gibt es keine Eigenschaft Daten.

Ereignisse
Es fehlt das Ereignis «Modifiziert» und «Text modifiziert». Das Ereignis «Fehler aufgetreten» kommt hinzu.

Beschriftungsfeld
Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

Allgemein
Der Titel dient als Beschriftung für ein anderes Kontrollfeld. Wird der Titel mit einem Kontrollfeld verbunden, dann kann er auch als Sprungziel für dieses Kontrollfeld festgelegt werden. Mit einem eingefügten «~» kann ein bestimmter Buchstabe des Titels als Sprungziel angegeben werden. «Na~me» definiert das «m» als Sprungziel. Befindet sich der Cursor in dem Formular, so wird die Schaltfläche über Alt + m erreicht.
Standardmäßig wird die Beschriftung nicht umgebrochen. Ein zu langer Titel wird abgeschnitten, wenn das Feld nicht groß genug ist. Der Wortumbruch hat allerdings keine Trennfunktion, so dass bei zu kleinen Feldern innerhalb eines Wortes ein Umbruch erfolgt.

**Daten**
keine

**Ereignisse**
Selbst das Beschriftungsfeld reagiert noch auf Ereignisse, die mit der Maus einer Taste oder dem Fokuserhalt zusammenhängen.

**Gruppierungsrahmen**
Der Gruppierungsrahmen fasst lediglich grafisch mehrere Kontrollfelder zu einer Gruppe zusammen und versieht sie mit einem zusätzlich grafisch eingebundenen Titel.

Wird ein Gruppierungsrahmen mit eingeschaltetem Assistenten aufgezogen, so geht der Assistent davon aus, dass mehrere Optionsfelder zusammen in diesem Rahmen erscheinen sollen.

Dem Formular liegt die Tabelle Leser zugrunde. Hier wird gerade die Auswahl des Geschlechtes zusammengestellt. Die Einträge sind anschließend die Beschriftungen der Optionsfelder.
Standardmäßig wird hier vorgewählt, dass das Geschlecht «weiblich» angewählt wird. Wird kein Feld vorgewählt, so ist der Standardeintrag in die zugrundeliegende Tabelle NULL.

Den Optionsfeldern wird bereits standardmäßig durch den Assistenten ein jeweils separater Wert zugeteilt. Da dieser Wert in unserem Fall nicht dem des Primärschlüssels entspricht, wird er geändert, und zwar für «männlich» auf den Wert 'm', für «weiblich» auf den Wert 'w'. Diese Werte entsprechen in dem Beispiel den Primärschlüsselfeldern der Tabelle “Geschlecht”
Der jeweils in den Optionsfeldern angeklickte Wert wird an das Feld "Geschlecht_ID" der dem Formular zugrundeliegenden Tabelle "Leser" übertragen. Die Tabelle "Leser" wird also über ein Optionsfeld mit den entsprechenden Fremdschlüsseln der Tabelle "Geschlecht" versorgt.

Die Gruppe der Optionsfelder erhält einen Rahmen, in den die Beschriftung 'Geschlecht' eingefügt wird.

Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**
Der Titel kann neben den Standardeinstellungen verändert werden. Eine Veränderung des Rahmens (Liniendicke, Linienfarbe) ist zur Zeit nicht vorgesehen, wohl aber eine Formatierung der Schrift.

**Daten**
Keine, fasst schließlich Eingabefelder nur optisch zusammen.

**Ereignisse**
Auch der Gruppierungsrahmen reagiert noch auf Ereignisse, die mit der Maus einer Taste oder dem Fokuserhalt zusammenhängen.

**Schaltfläche**
Neben den bereits unter Standardeinstellungen vieler Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**


Formulare: Formular-Eigenschaften 176
Wenn ja, so kann die Schaltfläche auch eingerückt dargestellt werden. Der Schaltzustand wird wie bei einem Schalter angezeigt – im Gegensatz zu einem Taster wie sonst bei Buttons. [Toggle]

Aktiv, wenn Umschalten auf Ja gesetzt ist. Ausgewählt ist dann der eingerückte Zustand. [DefaultState]

Wortumbruch, wenn die Schaltfläche zu schmal ist. [MultiLine]

Verschiedene Aktionen ähnlich denen der Navigationsleiste stehen zur Verfügung. Es kann hier auch ein Dokument oder eine Webseite geöffnet werden. Hierzu ist die URL in dem folgenden Feld einzugeben.


Nur für HTML-Formulare: Der Ziel-Frame (Rahmeneinteilung für verschiedene HTML-Seiten), in dem die Datei geöffnet werden soll. [TargetFrame]
Schaltfläche wird bei Ja zusätzlich umrandet. Bei mehreren alternativen Schaltflächen auf einem Formular soll dies die übliche Schaltfläche kennzeichnen. Sie wird durch Betätigung der Eingabetaste ausgelöst, wenn keine andere Aktion gerade auf die Eingabetaste reagieren muss. Nur eine Schaltfläche sollte im Formular die Standardschaltfläche sein. 

[DefaultButton]

Soll eine Grafik auf der Schaltfläche erscheinen?

[Graphic] [ImageURL]

Nur aktiv, wenn eine Grafik ausgewählt wurde. Die Grafik wird im Verhältnis zum Text angeordnet.

[ImagePosition] [ImageAlign]

Daten

Keine, es werden nur Aktionen ausgeführt.

Ereignisse

«Aktion bestätigen», «Aktion ausführen» und «Statusänderung»

**Grafische Schaltfläche**

Neben den bereits unter Eigenschaften der Kontrollfelder erklärten Eigenschaften gibt es die folgenden Besonderheiten:

**Allgemein**

Entspricht der normalen Schaltfläche. Allerdings wird diese Schaltfläche ohne Text und ohne einen sichtbaren Button dargestellt. Lediglich eine Rahmen um die Grafik kann eingestellt werden.

Eine grafische Schaltfläche erhält standardmäßig erst einmal keinen Tabstop.

Achtung: zur Zeitpunkt der Handbucherstellung funktioniert nahezu keine Aktion mit dieser Schaltfläche. Sie ist fast ausschließlich mit Makros nutzbar.

Daten

Keine, es werden nur Aktionen ausgeführt.

Ereignisse

«Aktion bestätigen» sowie alle Ereignisse, die die Maus, eine Taste oder den Fokus betreffen.

**Navigationsleiste**

Datum von 1 von 10

**Abbildung 26: Formularkontrolelement Navigationsleiste**

Eine separat zu den jeweiligen Elementen eines Formulars angeordnete Navigationsleiste macht dagegen klar, durch welche Elemente mit der Leiste navigiert wird. Diese Navigationsleiste bietet die folgenden Elemente:

<table>
<thead>
<tr>
<th>Datensatz suchen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datensatz</td>
</tr>
<tr>
<td>Datensatznummer</td>
</tr>
<tr>
<td>Zahl der Datensätze</td>
</tr>
<tr>
<td>Erster Datensatz</td>
</tr>
<tr>
<td>Vorheriger Datensatz</td>
</tr>
<tr>
<td>Nächster Datensatz</td>
</tr>
<tr>
<td>Letzter Datensatz</td>
</tr>
<tr>
<td>Neuer Datensatz</td>
</tr>
<tr>
<td>Datensatz speichern</td>
</tr>
<tr>
<td>Rückgängig: Dateneingabe</td>
</tr>
<tr>
<td>Datensatz löschen</td>
</tr>
<tr>
<td>Aktualisieren</td>
</tr>
<tr>
<td>Steuerelement aktualisieren</td>
</tr>
<tr>
<td>Sortieren</td>
</tr>
<tr>
<td>Aufsteigend sortieren</td>
</tr>
<tr>
<td>Absteigend sortieren</td>
</tr>
<tr>
<td>AutoFilter</td>
</tr>
<tr>
<td>Filter anwenden</td>
</tr>
<tr>
<td>Formularbasiertes Filter</td>
</tr>
<tr>
<td>Filter/Sortierung entfernen</td>
</tr>
<tr>
<td>Datensquelle als Tabelle</td>
</tr>
</tbody>
</table>

Die Such- und Filterfunktionen werden zum Schluss des Kapitels erklärt.

Die Navigation durch die Datensatznummern dürfte von der Bedienung von Multimedia-Geräten bekannt sein. Wird bei der Datensatznummer eine konkrete Nummer eingegeben, so bedeutet dies einen Sprung zum entsprechenden Datensatz.

Beim Aktualisieren gibt es auch den Button «Steuerelement aktualisieren». Werden Inhalte von Listenfeldern durch ein anderes Formular beeinflusst, so muss der Cursor in das Listenfeld gesetzt werden und der Inhalt kann über diesen Button neu eingelesen werden.

Neben den bereits unter **Standardeinstellungen vieler Kontrollfelder** erklärten Eigenschaften gibt es die folgenden Besonderheiten:
Allgemein

Daten
Keine, es werden nur Aktionen ausgeführt.

Ereignisse
Alle Ereignisse, die die Maus, eine Taste oder den Fokus betreffen.

Unabhängig von diesem Formularkontrollelement existiert natürlich auch die unten einblendbare Navigationsleiste mit den Elementen der obigen Abbildung.

Diese einblendbare Navigationsleiste bietet neben den Elementen des entsprechenden Formularkontrollementes noch die allgemeine Datensatzsuche, den formularbasierten Filter und die Darstellung der dem Formular zugrundeliegenden Datenquelle in der Tabellenansicht oberhalb des Formulars.

Wird nicht nur mit einem Formular, sondern mit Unterformularen und Nebenformularen gearbeitet, so ist darauf zu achten, dass diese einblendbare Navigationsleiste nicht beim Formularwechsel je nach Einstellung verschwindet. Das erzeugt nur Unruhe auf dem Bildschirm.

Drehfeld und Bildlaufleiste
Beide Felder haben keine direkte Verbindung zu Daten des Formulars. Sie lassen sich nur über Makros entsprechend nutzen, wobei das Drehfeld von der Anwendung her bereits z.B. in das Zah- lenfeld integriert ist.

Die Bildlaufleiste unterscheidet sich von dem Drehfeld dadurch, dass sie neben den Buttons für steigende und fallende Werte innerhalb einer vorher festgelegten Grenze auch einen Schieberegler haben. Der Name «Bildlaufleiste» ist dabei etwas irritierend, da über dieses Schieberegler nicht irgendwelche Bilder verschoben werden, sondern lediglich Werte in einem Feld verändert werden können.

Beim Start eines Formulars muss der Bildlaufleiste zuerst mitgeteilt werden, welchen Wert das Feld hat, das mit ihr verbunden sein soll. Dieses Makro ist an die Formular-Eigenschaften → Ereignisse → Nach dem Datensatzwechsel gebunden.

```
SUB Bildlaufleiste_Formular(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    oForm = oEvent.Source
    oFeld = oForm.getByName("Bildlaufleiste")
    oFeld.ScrollValue = oForm.getInt(oForm.findColumn("Zahlenwert"))
END SUB
```

Wird der Wert der Bildlaufleiste geändert, so muss entsprechend der Wert in dem verbundenen Feld geändert werden. Dieses Makro wird an Eigenschaften: Bildlaufleiste → Ereignisse → Beim Justieren gebunden

```vbnet
SUB Bildlaufleiste_Aenderung(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM inValue AS INTEGER
    oFeld = oEvent.Source.Model
    inValue = oEvent.Value
    oForm = oFeld.Parent
    oForm.updateInt(oForm.findColumn("Zahlenwert"), inValue)
END SUB
```

Zuerst werden die Variablen deklariert. Aus dem Feld wird der gewählte Wert als Ganzzahlenwert ausgelesen. Das dem Formular zugrundeliegende Tabellenfeld "Zahlenwert" wird auf den entsprechenden Wert eingestellt.

Details zu einem Code wie diesem sind in dem Kapitel «Makros» zu finden.

**Verstecktes Steuerelement**

Ein Kontrollfeld, das nicht über die Symbolleisten einfügbar ist, ist das versteckte Steuerelement. Es wird über den Formularnavigator im Kontextmenü eines Formulars neu erstellt.

Das versteckte Steuerelement erscheint wie jedes andere Kontrollfeld als Teil des Formulars. Es ist auf der Benutzeroberfläche nicht sichtbar.

---

*Formulare: Formular-Eigenschaften* 181
Die kompletten Eigenschaften des versteckten Steuerelementes sind recht spärlich. Der Name und die Zusatzinformationen sind von den anderen Steuerelementen bekannt. Zusätzlich kann hier ein Wert angegeben werden. [HiddenValue]

Für die Nutzung ohne Makros ist dies sicher nicht weiter von Bedeutung. Bei der Arbeit mit Makros kann es aber sinnvoll sein, an einer Stelle des Formular Werte zwischenspeichern zu können, um auf sie bei einer weiteren Aktion wieder zugreifen zu können. Ein entsprechendes Anwendungsbeispiel findet sich bei den Makros im Kapitel «Hierarchische Listenfelder».

**Mehrfachselektion**

Wird mit dem Auswahlpfeil (siehe Abbildung 22) ein größerer Bereich oder sogar sämtliche Elemente eines Formulars ausgewählt, so sind über diese Mehrfachselektion die Änderungen nach Abbildung 27 möglich.

Eher bietet sich eine Mehrfachselektion an, wenn insgesamt die Schriftart, die Höhe oder die Hintergrundfarbe gewechselt werden soll. Allerdings wirkt sich natürlich die Hintergrundfarbe auch auf die Beschriftungsfelder aus.

Sollen z. B. nur die Beschriftungsfelder geändert werden, so bietet es sich an, bei gedrückter Strg-Taste mit der linken Maustaste diese Felder direkt oder im Navigator anzuklicken, mit der rechten Maustaste über eins der Felder zu gehen und die Kontrollfeldeigenschaften aufzurufen. Jetzt wird die Auswahl der änderbaren Eigenschaften größer, da ja nur gleiche Felder ausgewählt wurden. Entsprechend kann hier auch all das geändert werden, was sonst in einem Beschriftungsfeld zur Verfügung steht.

Die Möglichkeiten der Mehrfachselektion richten sich also nach der Auswahl der Felder. Kontrollfelder der gleichen Art können in allen allgemeinen Eigenschaften, die das einzelne Kontrollfeld bietet, auf diese Art gemeinsam angepasst werden.

**Einfaches Formular komplett erstellt**

Ein einfaches Formular stellt Formularkontrollfelder für das Schreiben oder Lesen von Datensätzen aus einer einzigen Tabelle oder Abfrage zur Verfügung. Seiner Konstruktion entspricht dem folgenden Schaubild:

![Schaubild Arbeitsfläche mit Formular, Kontrollfeldern 1 bis 3 und Kontrollfeld...](image)

Am Beispiel eines einfachen Formulars für die Bibliotheksausleihe sollen hier verschiedenen Varianten eines Formulars vorgestellt werden.

![Schaubild Medienausleihe](image)

**Felder als Gruppe hinzufügen**

Eine schnelle Variante, direkt Felder mit Beschriftungen einzufügen, bietet die Funktion *Feld hinzufügen*.

Über diese auf der Symbolleiste Formular-Entwurf (siehe *Abbildung 21*) erreichbaren Funktion lassen sich alle Felder der dem Formular zugrundeliegenden Tabelle auswählen.

![Feld hinzufügen: Tabelle Ausleihen](image)

Über einen Doppelklick auf die Felder werden sie als Gruppierung zusammen mit einer Beschriftung (leider alle an die gleiche Stelle) in das Formular eingegliedert. Die Gruppen müssen also auf jeden Fall noch verschoben werden, damit anschließend das Formular im Einsatz wie das folgende Bild aussehen kann. Für die bessere Übersicht wurden alle unnötigen Leisten des Fensters entfernt und das Fenster entsprechend schmal zusammengeschoben, so dass nicht mehr alle Elemente der Navigationsleiste sichtbar sind.

Es wurden alle Felder bis auf "Medien_ID_BC" ausgewählt, da dies speziell für die Bedienung mit einem Barcodescanner gedacht ist.

![Medien ohne Makros.odb: Formular1 - LibreOffice Base: Database Form](image)

*Abbildung 28: Einfaches Formular über «Feld hinzufügen»*

**Felder anpassen**

Gestalterisch kann jetzt einiges getan werden, indem die Felder von der Länge her angepasst werden und die Datumsfelder aufklappbar gemacht werden. Wichtiger ist aber, dass die Felder für die "Medien_ID" und die "Leser_ID" für den Nutzer lösbar werden – es sei denn, jeder Bibliotheksbesucher muss einen Ausweis mit der "ID" mitbringen und jedes Medium wird bei der Aufnahme mit der "ID" versehen. Hiervon wurde im Folgenden nicht ausgegangen.

Um einzelne Felder anzupassen, muss zuerst einmal die Gruppe betreten werden. Dies kann über einen rechten Mausklick auf die Gruppe und das entsprechende Kontextmenü erfolgen. Übersichtlicher für die späteren Verfahren ist allerdings die Arbeit mit dem Formularnavigator.

![Abbildung 29: Formularkontrollelemente: Gruppe betreten](image-url)

**Formulare: Formular-Eigenschaften** 185
Abbildung 30: Formularkontrollelemente über den Formularnavigator direkt anwählen

Der Formularnavigator stellt alle Elemente des Formulars mit ihren Bezeichnungen dar. Für die Kontrollfelder wurden als Bezeichnungen direkt die Namen der Felder aus der dem Formular zugrundeliegenden Tabelle genommen. Die Beschriftungselemente haben den Zusatz «Beschriftung».

Durch einen Klick auf "Medien_ID" ist dieses Feld ausgewählt. Mit einem Rechtsklick wird es über das Kontextmenü möglich, das ausgewählte Feld durch eine andere Feldart zu ersetzen:
Diese Ersetzung wird in dem Formular für "Medien_ID" und "Leser_ID" vorgenommen.

Die Änderung ist im Formularnavigator über die Symbole sichtbar.
Die SQL-Abfragen für die Listenfelder können jetzt über den rechten Button mit der grafischen Benutzeroberfläche erstellt werden. Der Listenfeldassistent steht hier nicht zur Verfügung. Er springt nur automatisch ein, wenn ein Listenfeld direkt neu gegründet, nicht aber durch Umwandlung von einem anderen Feld aus erstellt wird. Zum SQL-Befehl siehe das Kapitel "Abfragen für die Erstellung von Listenfeldern".

Nachdem die Listenfelder als aufklappbar angepasst wurden, können noch die folgenden Mängel behoben werden:

- Das Kontrollfeld «ID» sollte als nicht beschreibbar erklärt werden.

**Felder einzeln hinzufügen**


Lediglich beim Aufziehen der Listenfelder kommt der Assistent zum Einsatz, der die Schritte zur Erstellung eines korrekten Feldes für Ungeübte vereinfacht. Nach kurzer Anwendungsphase wird der Assistent allerdings den Ansprüchen nicht mehr gerecht, da er

- Die Einträge nicht automatisch sortiert.
- Eine Zusammenfassung von mehreren Feldern im aufzulistenden Inhalt nicht ermöglicht.

Hier muss dann immer wieder nachgebessert werden, so dass ganz schnell der SQL-Code direkt über den eingeblendeten Abfrageeditor erstellt wird.

Formulare: Formular-Eigenschaften 188
Beim Hinzufügen einzelner Felder müsste eine Gruppierung von Feld und Beschriftung gesondert vorgenommen werden (siehe Standardeinstellungen vieler Kontrollfelder). In der Praxis kann sich aber die fehlende Verknüpfung auch positiv bemerkbar machen, da der Zugang zu den Eigenschaften eines Kontrollfeldes über das Kontextmenü sofort möglich ist und nicht erst ein Betreten der Gruppe erfordert.

**Tabellenkontrollfeld**

Unter der Beschreibung zum Tabellen-Kontrollfeld wurde bereits über den Tabellenassistenten das entsprechende Tabellenkontrollfeld erstellt. Es hat allerdings noch einige Nachteile, die zu verbessern sind:

- Aus den Feldern «Medien_ID» und «Leser_ID» muss ein Listenfeld erstellt werden.
- Numerische Felder müssen gegebenenfalls auf Felder ohne Nachkommastellen umgestellt werden, da der Assistent hier grundsätzlich 2 Nachkommastellen belässt.


Die Kontrollfelder innerhalb des Tabellenkontrollfeldes werden als Spalten bezeichnet. Über das Kontextmenü ist es jetzt möglich, Felder durch andere Felder zu ersetzen. Allerdings steht nicht die ganze Palette an Feldern zur Verfügung. So fehlen Buttons, Optionsfelder oder das grafische Kontrollfeld.
Die Eigenschaften der Felder sind über das Kontextmenü unter dem Begriff Spalte verborgen. Hier kann dann z. B. das Numerische Feld *Verlängerung* geändert werden, so dass keine Nachkommastellen mehr angezeigt werden. Auch der dort standardmäßig eingetragene minimale Wert von –1.000.000,00 macht wohl für Verlängerungen wenig Sinn. Die Anzahl dürfte wohl im positiven einstelligen Bereich bleiben.

Sobald die Eigenschaften einer Spalte aufgerufen sind, lässt sich ohne das Eigenschaftsfeld zu schließen auch eine andere Spalte aufrufen. Ohne eine separate Speicherung ist es so möglich, alle Felder nacheinander abzuarbeiten.

Die Speicherung erfolgt schließlich im gesamten Formular und letztlich mit dem Datenbankdokument selbst.

Die Eigenschaften dieser in das Tabellenkontrollfeld eingebauten Felder sind übrigens nicht ganz so umfangreich wie die der Felder außerhalb. Die Schrift z. B. lässt sich nur für das ganze Tabellenkontrollfeld einstellen. Außerdem gibt es hier nicht die Möglichkeit, einzelne Spalten ohne Tabulatorstop zu überspringen.

**Tipp**

Durch ein Formular erfolgt die Bewegung entweder mit der Maus oder mit dem Tabulator.

Gerät ein Tabulator einmal in ein Tabellenkontrollfeld, so bewegt sich der Cursor dort mit jeder Tabulatorbewegung ein Feld weiter nach rechts und bei Zeilenende schließlich zum ersten Feld des nächsten Datensatzes im Tabellenkontrollfeld. Aus dem Tabellenkontrollfeld heraus geht es mittels *Strg + Tab*.

Die Anordnung der Spalten lässt sich über das Verschieben der Spaltenköpfe ändern:

Wird diese Verschiebung im Formularentwurf gemacht, so ist sie dauerhaft wirksam. Vorübergehend lässt sich so eine Verschiebung auch im Formular erledigen, wenn es zur Eingabe von Daten geöffnet ist.

**Hinweis**

Leider enthält diese Funktion zur Zeit einen Bug. Verschiebungen sind seit der Version LO 3.6 nicht mehr möglich. *(Bug 54021)*

Zur Zeit können also nur Felder neu eingefügt und auf diese Weise auch positioniert werden.

Sollen nur bestimmte Felder zum Bearbeiten offen stehen, so bietet es sich an, das Formular gleich mit mehreren Tabellenkontrollfeldern zu bestücken, denn der Tabulator wird standardmäßig von einem Tabellenkontrollfeld gefangen.
In dem in Abbildung 32 aufgezeigten Formular wird oben die Ausgabe von Medien geregelt. Hier sind nur die direkt notwendigen Felder.

Im unteren Tabellen-Kontrollfeld erscheinen noch einmal alle Felder, damit auch ersichtlich ist, für welche Person und welches Medium denn die Rückgabe erfolgt.

<table>
<thead>
<tr>
<th>Leser</th>
<th>Medium</th>
<th>Ausleihdatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Das sogenannte Böse</td>
<td>02.11.11</td>
</tr>
<tr>
<td>Gerd, Lisa - Nr. 2</td>
<td>Eine kurze Geschichte der Zeit</td>
<td>15.10.11</td>
</tr>
<tr>
<td>Mirinda, Monika - Nr. 3</td>
<td>Der kleine Hobbit</td>
<td>02.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Traditionelle und kritische Theorie</td>
<td>04.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>I hear you knocking</td>
<td>28.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Die neue deutsche Rechtschreibu</td>
<td>28.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Die neue deutsche Rechtschreibu</td>
<td>09.11.11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Leser</th>
<th>Medium</th>
<th>Ausleihdatum</th>
<th>Verlängerung</th>
<th>Rückgabedatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Das sogenannte Böse</td>
<td>02.11.11</td>
<td>04.11.11</td>
<td></td>
</tr>
<tr>
<td>Gerd, Lisa - Nr. 2</td>
<td>Eine kurze Geschichte der Zeit</td>
<td>15.10.11</td>
<td>25.02.12</td>
<td>04.11.11</td>
</tr>
<tr>
<td>Mirinda, Monika - Nr. 3</td>
<td>Der kleine Hobbit</td>
<td>02.11.11</td>
<td>1</td>
<td>04.04.12</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Traditionelle und kritische Theorie</td>
<td>04.11.11</td>
<td>2</td>
<td>28.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>I hear you knocking</td>
<td>28.11.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Die neue deutsche Rechtschreibu</td>
<td>28.11.11</td>
<td></td>
<td>04.04.12</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Die neue deutsche Rechtschreibu</td>
<td>09.11.11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Diese Filterung ist entweder über eine Abfrage oder direkt in den Formulareigenschaften möglich. Wird in den Formulareigenschaften eine Filterung erstellt, so kann diese auch bei der Eingabe in das Formular vorübergehend ausgeschaltet werden. Die Filterung mittels Abfrage wird in dem entsprechenden Kapitel «Abfragen» aufgegriffen. Hier wird jetzt der Weg über die Formulareigenschaften gezeigt:
Die Filterung wird über den Button mit den drei Punkten gestartet. Sie könnte allerdings auch direkt in das Textfeld eingegeben werden, wenn die SQL-Formulierung bekannt ist.

In der grafischen Benutzeroberfläche lässt sich jetzt das Feld mit dem Namen "Rueck_Datum" auswählen. Angezeigt werden sollen nur die Datensätze, bei denen dieses Feld «leer» ist, wobei «leer» für die SQL-Bezeichnung `NULL` steht.


Das auf diese Art bereinigte Formular sieht dann schon etwas übersichtlicher aus:
Sicher ist dies noch verbesserbar, bietet aber neben der Funktionalität der anderen Formulare den unbestreitbaren Vorteil, dass alle Medien auf einen Blick sichtbar sind.


Wird eine Zeile verlassen, so wird der Datensatz automatisch abgespeichert.

Noch ist das Formular zur Medienausleihe in vielen Bereichen zu verbessern.

- Es wäre wünschenswert, wenn an einer Stelle der Leser ausgewählt würde und an anderer Stelle zu diesem Leser die entliehenen Medien erscheinen.
- In der oberen Tabelle sind lauter Datensätze zu sehen, die dort eigentlich gar nicht notwendig sind. Die Medien sind ja schon entliehen. Das obere Tabellenblatt ist aber erstellt worden, um die Ausleihe zu ermöglichen. Besser wäre es, wenn hier nur ein leeres Blatt erscheint, das dann mit den neuen Ausleihen ausgefüllt wird.

Solche Lösungen sind mit Hilfe von weiteren Formularen möglich, die hierarchisch aufeinander aufgebaut sind und eine getrennte Sicht auf die Daten ermöglichen.

---

**Tipp**

Das Tabellenkontrollfeld kann mit anderen Feldern im Formular kombiniert werden. In dem Tabellenkontrollfeld werden z. B. nur die Medientitel angezeigt, in den darunter stehenden Formularfeldern dann an diesem Datensatz Änderungen vorgenommen.


---

**Medienausleihe**

<table>
<thead>
<tr>
<th>Leser</th>
<th>Medium</th>
<th>Ausleihdatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>I hear you knocking</td>
<td>28.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Die neue deutsche Rechtschreibung</td>
<td>09.11.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Traditionelle und kritische Theorie</td>
<td>09.12.11</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Das Postfix-Buch</td>
<td>25.02.12</td>
</tr>
<tr>
<td>Nobody, Terence - Nr. 9</td>
<td>Der kleine Hobbit</td>
<td>04.04.12</td>
</tr>
<tr>
<td>Müller, Heinrich - Nr. 1</td>
<td>Eine kurze Geschichte der Zeit</td>
<td>04.04.12</td>
</tr>
<tr>
<td>Lederstrumpf, Bert - Nr. 0</td>
<td>Das sogennante Büße</td>
<td>04.04.12</td>
</tr>
</tbody>
</table>

**Formulare: Formular-Eigenschaften**
**Hauptformular und Unterformular**


Typische Tabellenkonstruktionen, die sich für Unterformulare eignen, sind Tabellen mit einer 


Wir nutzen jetzt erst einmal die Beziehung der Tabelle "Leser" zur Tabelle "Ausleihe" (siehe 

«Tabellen Ausleihe»). Die Tabelle "Leser" wird Grundlage für das Hauptformular, die Tabelle "Ausleihe" wird in dem Unterformular wiedergegeben.

Formulare: Hauptformular und Unterformular 194
Das Hauptformular wird auf die Tabelle "Leser" eingestellt. Damit die Leser auch schnell genug gefunden werden, wird die Tabelle sortiert wiedergegeben. Auf eine Navigationsleiste wird verzichtet, weil zwischen Hauptformular und Navigationsleiste der Inhalt des Unterformulars angezeigt wird. Stattdessen soll das Formularkontrollelement Navigationsleiste eingebaut werden.

Durch Rechtsklick auf das Hauptformular im Formular-Navigator wird über das Kontextmenü ein neues Formular gegründet. Das Formular hat standardmäßig wieder den Namen Formular, ist jetzt aber ein Element im Unterverzeichnis des Hauptformulars.

Die Eigenschaften des Unterformulars müssen jetzt entsprechend eingestellt werden, damit es die richtige Datenquelle und die zu dem entsprechenden Leser gehörigen Daten wiedergibt.

Über Verknüpfen von und Verknüpfen nach wird eine Verbindung zum Hauptformular hergestellt, in dem das Unterformular liegt. Der Button mit den drei Punkten zeigt wieder an, dass es hier ein helfendes Fenster für die Auswahl der Einstellungen gibt.

Die ausgewählte Verknüpfung von der Tabelle des Unterformulars nach der Tabelle des Hauptformulars wird jetzt mit den entsprechenden Feldern der Tabellen angegeben.

Um jetzt ein Tabellenkontrollfeld für das Hauptformular zu erstellen, muss das Hauptformular im Formular-Navigator markiert sein. Dann zeigt bei eingeschaltetem Tabellenkontrollfeld-Assistenten der Assistent die Felder an, die im Hauptformular zur Verfügung stehen. Entsprechend wird mit dem Unterformular verfahren.

Nachdem so die Tabellenkontrollfelder aufgezogen wurden werden die entsprechenden Änderungen durchgeführt, die schon beim einfachen Formular erklärt wurden:

• Ersetzen des numerischen Feldes «Medien_ID» im Unterformular durch ein Listenfeld.
• Umbenennung des Feldes «Medien_ID» in Medien.
• Anpassung der numerischen Felder an ein Format ohne Nachkommastellen.
• Eingrenzung der minimalen und maximalen Werte.
• Umbenennung anderer Felder, um Platz zu sparen oder Umlaute darzustellen, die bei der Feldbenennung in den Datenbanktabellen vermieden wurden.

Sortier- und Filterfunktion werden für das Hauptformular ergänzt, indem eine Navigationsleiste hinzugefügt wird. Die anderen Felder der Navigationsleiste werden nicht benötigt, da sie vom Tabel-
lenkontrollfeld weitgehend zur Verfügung gestellt werden. (Datensatzanzeige, Datensatznaviga-

tion) bzw. durch die Bewegung im Tabellenkontrollfeld erledigt werden (Speicherung von Daten).

Das erstellte Formular könnte schließlich wie in der folgenden Abbildung aussehen:

**Medienausleihe**

<table>
<thead>
<tr>
<th>ID</th>
<th>Nachname</th>
<th>Vorname</th>
<th>Sperre</th>
<th>Geschlecht</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Keindurchblick</td>
<td>Hein</td>
<td>männlich</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Lederstrumpf</td>
<td>Bert</td>
<td>männlich</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Miirinda</td>
<td>Monika</td>
<td>weiblich</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Müller</td>
<td>Heinrich</td>
<td>männlich</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MüZiggang</td>
<td>Kerstin</td>
<td>weiblich</td>
<td></td>
</tr>
</tbody>
</table>

Datensatz 7 von 10 (1)

**Ausgeliehene Medien des ausgewählten Lesers**

<table>
<thead>
<tr>
<th>Medien</th>
<th>Ausleihdatum</th>
<th>Rückgabedatum</th>
<th>Verlängerung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eine kurze Geschichte der Zeit</td>
<td>04.04.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Im Augenblick - Nr. 8</td>
<td>22.04.12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Datensatz 1 von 2

**Abbildung 33: Formular, bestehend aus Hauptformular (oben) und Unterformular (unten).**

Wird jetzt im Hauptformular ein Leser ausgewählt, so werden im Unterformular nur die Medien auf-
gezeigt, die der Leser zur Zeit entliehen hat. Wird ein Medium zurückgegeben, so erscheint dies
noch so lange im Formular, bis das Formular selbst aktualisiert wird. Dies geschieht automatisch,
 wenn im Hauptformular ein anderer Datensatz gewählt wurde. Bei der erneuten Anwahl des
ursprünglichen Lesers sind also die zurückgegebenen Medien nicht mehr in der Anzeige.

Diese verzögerte Aktualisierung ist in diesem Fall wohl auch erwünscht, da so direkt eingesehen
werden kann, welche Medien denn jetzt gerade auf der Theke der Mediothek liegen und ob diese
schon registriert wurden.

Diese Formularkonstruktion bietet schon deutlich mehr Komfort als die vorherige mit nur einem
einzeligen Formular. Allerdings gibt es noch Details, die verbesserungswürdig erscheinen:

- Medien und Ausleihdaten können geändert werden, wenn die Medien schon länger entlie-
  hen sind.
  Eine Änderung der Medien-Daten führt dazu, dass nicht mehr nachvollzogen werden kann,
  welches Medium denn nun noch in der Mediothek vorhanden ist und welches entliehen
  wurde.
  Eine Änderung des Ausleihdatums kann zu fehlerhaften bzw. nicht beweisbaren Anmahn-
  nungen führen.
- Wird ein Leser nicht durch Klick auf den Datensatzmarkierer markiert, so zeigt nur der
  kleine grüne Pfeil auf dem Markierer an, welcher Datensatz gerade aktiv ist. Es ist auch
  möglich, den aktiven Datensatz komplett aus dem Tabellenkontrollfenster zu scrollen.
  Statt des Textes «Ausgeliehene Medien des ausgewählten Lesers» würde hier besser auch
  der Name erwähnt.
- Es ist möglich, mehrmals das gleiche Medium auszuleihen, ohne dass es zurückgegeben
  wurde.
- Es ist möglich, die Datensätze für ausgeliehene Medien einfach zu löschen.

Zuerst wird einmal die Auswahl der Leser verbessert. Dies soll vor Änderungen in der Ausleihe schützen. Eine einfache Lösung wäre, keine Änderungen zuzulassen, aber neue Datensätze eingeben zu können. Dazu wird immer noch eine Suchfunktion benötigt, wenn ein Leser ein Medium entleihen will. Besser wäre, in einem Listenfeld die Leser auszusuchen und in voneinander getrennten Tabellenkontrollfeldern die Ausgabe und die Rückgabe zu erledigen.

Für das Hauptformular benötigen wir eine Tabelle, in die das Listenfeld seinen mit dieser Tabelle verbundenen Wert schreiben kann. Die Tabelle muss also ein Integer-Feld und einen Primärschlüssel haben. Sie wird beständig nur einen Datensatz enthalten; daher kann das Feld ID als Primärschlüssel ruhig 'Tiny Integer' sein. Die folgende Tabelle mit der Bezeichnung "Filter" soll also gebildet werden:

<table>
<thead>
<tr>
<th>Tabellename: Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feldname</strong></td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>Integer</td>
</tr>
</tbody>
</table>

Die Tabelle wird mit einem Primärschlüsselwert gefüllt, und zwar dem Wert 0. Diesen Datensatz wird das Hauptformular beständig lesen und neu schreiben.

Das Hauptformular beruht auf der Tabelle "Filter". Es wird nur der Wert aus der Tabelle gelesen, bei dem der Primärschlüssel "ID" '0' ist. Es sollen keine Daten hinzugefügt werden, sondern nur der aktuelle Datensatz beständig neu geschrieben werden. Daher ist hier nur das Ändern erlaubt, eine Navigationsleiste sowieso überflüssig.
Das Hauptformular wird mit dem Unterformular so verknüpft, dass der Wert aus dem Feld "Integer" der Tabelle "Filter" gleich dem Wert aus dem Feld "Leser_ID" aus der Tabelle "Ausleihe" ist. Das Unterformular bleibt in seinen Eigenschaften ansonsten gegenüber der Vorversion unberührt.

Bevor jetzt im Hauptformular ein Listenfeld aufgezogen wird, wird erst einmal der Assistent ausgeschaltet. Mit dem Assistenten könnte nur ein Feld erzeugt werden, das lediglich einen Feldinhalt anzeigt; es wäre unmöglich, Nachname und Vorname und zusätzlich noch eine Nummer in dem Anzeigefeld des Listenfeldes zu positionieren. Wie bereits bei dem einfachen Formular wird jetzt das Listenfeld mit Nachname, Vorname – Nr. ID bestückt. Das Listenfeld gibt außerdem die ID an die darunterliegende Tabelle weiter.


Der Button kann einfach mit 'OK' als Aufschrift versehen werden. Als Aktion wird Formular aktualisieren zugewiesen.

Abbildung 34: Hauptformular als Filter für ein Unterformular

Das Formular funktioniert jetzt schon insofern besser, als

- keine Leser mehr bearbeitet, vor allem verändert und gelöscht werden können und
- Leser schneller über das Eintippen in das Kontrollfeld gefunden werden als über Filter.

Für eine weitreichendere Funktionalität (Rückgabe ohne Änderung der vorherigen Daten) muss ein zweites Unterformular gegründet werden, das sich auf die gleiche Tabelle "Ausleihe“ bezieht. Damit dennoch die Funktionalität des Listenfeldes aus Abbildung 34 gewährleistet wird, müssen beide Unterformulare allerdings noch eine Ebene tiefer gelegt werden, also Unterformulare eines Unterformulars werden. Eine Aktualisierung von Daten verläuft in der Hierarchie nur vom Hauptformular zum Unterformular abwärts. Der Button im letzten vorgestellten Formular würde nur ein Unterformular aktualisieren, nicht aber das zweite, daneben liegende Unterformular.


<table>
<thead>
<tr>
<th>Hinweis</th>
</tr>
</thead>
</table>

Formulare: Hauptformular und Unterformular
Das Hauptformular und das Unterformular nutzen einfach die gleiche Tabelle. Im Unterformular werden keine Daten eingegeben. Deshalb stehen alle diesbezüglichen Felder auf *Nein*. Verknüpft werden Hauptformular und Unterformular durch das Feld, dessen Wert auch an die Unter-Unterformulare weitergegeben werden soll: das Feld "Integer" der Tabelle "Filter".

Im ersten Unter-Unterformular werden keine alten Daten angezeigt sondern nur neue Daten verarbeitet. Hierzu reicht der Filter, der gesetzt wurde. Es werden nur Daten angezeigt, die zu der "Leser_ID" passen und deren Leihdatum leer ist ("Leih_Datum" IS NULL). Das bedeutet beim Aufruf ein leeres Tabellen-Kontrollfeld. Da das Tabellen-Kontrollfeld zwischendurch nicht laufend aktualisiert wird, bleiben die gerade neu ausgeliehenen Medien so lange in dem Tabellen-Kontrollfeld stehen, bis über den Aktualisierungslbutton 'OK' entweder ein neuer Name ausgewählt oder auch nur die Übernahme der Daten in das zweite Unter-Unterformular veranlasst wird.
Das zweite Unter-Unterformular erfordert mehr Einstellungen. Auch dieses Formular enthält die Tabelle "Ausleihe". Hier werden aber die Daten gefiltert, bei denen das Rückgabedatum leer ist ("Rueck_Dat" IS NULL). Die Daten wird wie im vorhergehenden Formular so sortiert, dass die am längsten entliehenen Medien direkt sichtbar sind.


Im oberen Tabellen-Kontrollfeld werden alle Felder entfernt, die nichts mit der Ausleihe zu tun haben. Es beleibt lediglich das Medium als Auswahlfeld sowie das Ausleihdatum "Leih_Dat" stehen.

Wird schließlich noch die Abfrage für das Listenfeld im oberen Tabellen-Kontrollfeld entsprechend gewählt, so werden dort nur Medien angezeigt, die noch entliehen werden können. Mehr dazu im Kapitel «Abfragen».

Formulare: Hauptformular und Unterformular
Abbildung 35: Das Auswahlfeld im oberen Unterformular zeigt nur Medien an, die nicht ausgeliehen sind.

Das Formular zur Medienausleihe ist jetzt schon wesentlich besser zu bedienen. Kommt ein Leser an die Ausleihtheke, so wird der Name herausgesucht. Die zu entleihenden Medien können aus dem Listenfeld gewählt und das Entleihdatum eingestellt werden. Mit dem Tabulator geht es dann zum nächsten Datensatz.

Eine letzte Verbesserung wäre noch wünschenswert: Das Entleihdatum muss jedes Mal gewählt werden. Stellen wir uns einen Tag in der Mediothek mit vielleicht 200 Entleihvorgängen vor, vielleicht auch nur eine Person, die gleich 10 Medien auf einmal entleiht. Das wären mehrmals hintereinander die gleichen Eingabevorgänge für ein Feld. Hier muss eine Einsparmöglichkeit her.

Unser Hauptformular beruht auf einer Tabelle "Filter". Das Hauptformular arbeitet dabei immer nur mit dem Datensatz, der als Primärschlüssel die "ID" '0' hat. In die Tabelle "Filter" können ohne weiteres noch mehr Felder eingebaut werden. Da noch kein Feld enthalten ist, das ein Datum speichern kann, gründen wir einfach ein neues Feld mit dem **Feldname → Datum** und dem **Feldtyp → Datum**. In der Tabelle "Filter" wird jetzt nicht nur die "Leser_ID" ("Filter"."Integer") sondern auch das "Leih_Datum" ("Filter"."Datum") gespeichert.
Im Hauptformular erscheint jetzt zusätzlich ein Datumsfeld, außerdem noch ein Beschriftungsfeld, das auf den Inhalt des Datumsfeldes hinweist. Der Wert aus dem Datumsfeld wird in der Tabelle "Filter" gespeichert und über die Verbindung vom Unterformular zum Unter-Unterformular weitergegeben.


In der Beispieldatenbank ist dies entsprechend aufgezeigt. Zur Vorstellung des ersten Formularentwurfs sollte das obige Beispiel aber erst einmal ausreichend sein.

Da allerdings das letztlich in der Beispieldatenbank «Medien_ohne_Makros.odb» vorgesehene Formular noch weiter entwickelt wurde, sollen die Erweiterungen hier noch kurz vorgestellt werden.
In die Ausleihe wurden die folgenden Eigenschaften aufgenommen:

- In einem Tabellenkontrollfeld werden die Leser und Leserinnen angezeigt. Hier können auch neue Leser und Leserinnen eingefügt werden.


- Im Untertitel wird noch einmal der Name der Person aufgezeigt, für die die Ausleihe erfolgen soll. Ist die Ausleihe für die Person gesperrt, so wird dies angezeigt.

- Das Entleihdatum ist auf das aktuelle Datum eingestellt. Dazu wurde die Filtertabelle über SQL so eingestellt, dass bei einem nicht eingegebenen Datum der Default-Wert das aktuelle Datum abspeichert.

- Die noch entleihbaren Medien werden in einem Listenfeld ausgewählt. Über den Button 'Aktualisieren' wird die Ausleihe in das darunter stehende Tabellenkontrollfeld übertragen.
• Das mittlere Tabellenkontrollfeld dient lediglich der Anzeige der zu dem angegebenen aktuellen Datum ausgeliehenen Medien. Hier kann auch eine irrtümliche Ausleihe durch Löschen der Zeile rückgängig gemacht werden.

• Im unteren Tabellenkontrollfeld ist wie im vorher gezeigten Beispiel die Änderung von Medien aus Ausleihdatum nicht möglich. Auch eine Löschung ist nicht möglich.

• Neben der Eingabe des Rückgabedatums oder gegebenenfalls einer Verlängerung wird angezeigt, für wie viele Tage das Medium entliehen werden darf und wie viele Tage die restliche Entleihzeit beträgt.

• Geht die Restzeit in den negativen Bereich, so muss das Medium sofort zurückgegeben werden. Die Ausgabe ist deswegen gesperrt. Sie wird dadurch wieder ermöglicht, dass die Medien zurückgegeben werden. Nach der Medienrückgabe muss lediglich einmal auf 'Aktualisieren' gedrückt werden.

Dieses Formular ist mit Hilfe von Abfragen wesentlich komplexer strukturiert als die vorher vorgestellte Variante. Mehr zu den Grundlagen ist deshalb im Kapitel 'Abfragen' zu erfahren.

Eine Ansicht – viele Formulare

Während das Beispiel für die Ausleihe nur Eingaben in eine Tabelle erforderte (Tabelle "Ausleihe") und zusätzlich die Eingabe in der einfacheren Form auch für neue Leser ermöglichte, ist die Eingabe für die Medien wesentlich umfassender. Schließlich spielen rund um die Medien-Tabelle insgesamt 8 zusätzliche Tabellen mit (siehe «Tabellen Medienaufnahme»).

Durch die Zuordnung im Verhältnis n:1 bieten sich die Tabellen "Untertitel" und "rel_Medien_Verfasser" als Unterformulare zum Formular 'Medien' an. Tabellen, die hingegen ein Verhältnis von 1:n zur Tabelle "Medien" haben, müssten eigentlich ein Formular bilden, das über dem der Tabelle Medien liegt. Da es sich aber um mehrere entsprechende Tabellen handelt, werden deren Werte über Listenfelder in das Hauptformular eingetragen.

Die Tabelle eines Hauptformulars steht zur Tabelle eines Unterformulars grundsätzlich im Verhältnis 1:n, in seltenen Ausnahmen im Verhältnis 1:1. Das Hauptformular beherbergt in der Regel nach längerem Gebrauch der Datenbank also eine Tabelle, die deutlich weniger Datensätze hat als die Tabelle des Unterformulars.

Mehrere Hauptformulare können nicht auf ein Unterformular zugreifen. Es ist also nicht möglich, viele 1:n-Beziehungen gleichzeitig über die Formularanordnung zu lösen, bei denen das Unterformular den gleichen Inhalt hat. Gibt es eine 1:n-Beziehung für die Tabelle eines Formulars, so lässt sich dies über ein Listenfeld regeln. Hier stehen wenige Begriffe aus einer anderen Tabelle zur Auswahl, deren Fremdschlüssel auf diese Art in die Tabelle des Hauptformulars eingetragen werden.
Über Listenfelder werden dem Hauptformular, das auf der Tabelle "Medien" basiert, z. B. die Inhalte der Tabellen "Kategorie", "Ort" oder "Verlag" zugewiesen. Über Unterformulare sind die Tabelle "rel_Medien_Verfasser" und "Untertitel" mit dem Hauptformular und damit mit der Tabelle "Medien" verbunden.

Das Unterformular für die Tabelle "rel_Medien_Verfasser" besteht außerdem wieder aus zwei Listenfeldern, damit nicht die Fremdschlüssel der Tabelle "Verfasser" und "Verf_Zusatz" (Zusätze wie Hrsg., Gesang usw.) direkt als Ziffern eingegeben werden müssen.

Bei dem Formular für die Medieneingabe müssen die Listenfelder meist während der Eingabe nach und nach aufgefüllt werden. Hierzu werden neben dem Hauptformular weitere Formulare eingebaut. Sie existieren unabhängig vom Hauptformular:

Das gesamte Formular zur Medieneingabe sieht so aus:

Die folgenden Tabellenkontrollfelder sind alle in einzelnen Nebenformularen zum Hauptformular, dem Eingabeformular, untergebracht:
Hier werden jeweils die kompletten Daten für eine Tabelle eingegeben. Am Anfang ist es häufig erforderlich, auf diese Nebenformulare auszuweichen, da z. B. nicht viele Verfasser in der entsprechenden Tabelle bereits abgespeichert wurden.

Wurde in einem der Tabellenkontrollfelder ein neuer Datensatz abgespeichert, so ist in dem Hauptformular das entsprechende Listenfeld aufzusuchen und über **Kontrollfeld aktualisieren** (siehe «Navigationsleiste») neu einzulesen.

Der Formularnavigator zeigt entsprechend viele Formulare an:
Die Formulare sind einzeln benannt, so dass sie erkennbar bleiben. Lediglich das Hauptformular hat vom Formularassistenten noch die Bezeichnung «MainForm» behalten. Insgesamt existieren also 8 Formulare parallel. Das Formular «Filter» beherbergt eine Suchfunktion, das Formular «MainForm» die Haupteingabefläche. Alle anderen Formulare stehen für je eins der oben abgebildeten Tabellenkontrollfelder.

Ohne die Tabellenkontrollfelder erscheint das Hauptformular schon etwas übersichtlicher:

Das Feld für den Suchbegriff liegt in dem Nebenformular «Filter», die beiden Tabellenkontrollfelder (für die Verfasser und für die Untertitel) liegen in Unterformularen zum Hauptformular der Medien-eingabe.

Im Formularnavigator sieht das Formular selbst dann schon wesentlich unübersichtlicher aus, da natürlich alle Kontrollfelder, auch die Beschriftungen, dort auftauchen. In den vorhergehenden Formularen waren ja die meisten Felder als Spalten von Tabellen-Kontrollfeldern im Formularnavigator nicht zu sehen.


Die Reihenfolge, in der die Elemente im Navigator auftauchen, hat aber nichts damit zu tun, in welcher Reihenfolge die Elemente durch einen Tabulatorsprung erreicht werden. Dies wird durch die Aktivierungsreihenfolge bestimmt.

Über die Aktivierungsreihenfolge werden alle Elemente, die Daten an die dem Formular zugrunde liegende Tabelle weitergeben oder Aktionen hervorrufen können, in ihrer Reihenfolge zueinander festgelegt. Dies entspricht der Einstellung in den Eigenschaften in Standardeinstellungen vieler Kontrollfelder zur Aktivierungsreihenfolge.

In der Aktivierungsreihenfolge tauchen allerdings auch die Elemente auf, bei denen der Tabstop eigentlich abgeschaltet ist. Sie werden zwar in die Nummerierung aufgenommen, aber tatsächlich bei der Arbeit mit dem Formular über die Tastatur nicht angesprungen.


Existiert ein Unterformular, so wird bei der Auto Sortierung das Ganze so eingestellt, dass nach dem Hauptformular direkt in das Unterformular gesprungen wird. Bei einem Tabellenkontrollfeld führt dies über die Tastatureingabe dann dazu, dass der Cursor in diesem Unterformular gefangen ist und nur mit der Maus oder über Strg + Tab aus dem Subformular wieder herausbewegt werden kann.

Die Auto Sortierung funktioniert allerdings, bezogen auf das Tabellenkontrollfeld, nur einmal. Ein weiteres Unterformular mit Tabellenkontrollfeld wird nicht mit einbezogen. Parallel liegende Formulare werden also nicht berücksichtigt. Eine Auto-Sortierung kann in Bezug auf ein Unterfor-
mular mit Tabellenkontrollfeld auch nicht rückgängig gemacht werden. Hierzu müsste das Unterformular komplett entfernt (oder vorübergehend in ein anderes Formular verschoben) werden.


**Fehlermeldungen bei der Eingabe in Formulare**

Einige Fehlermeldungen treten gerade bei den ersten Formularerstellungen gehäuft auf, so dass sie hier kurz erklärt werden sollen:

- **Attempt to insert null into a non-nullable column: column: ID table: Tabellenname in statement ...**

  Es gibt Felder, die nicht leer sein dürfen. Wird dies nur in der Tabelle definiert (NOT NULL), so erscheint eine englischsprachige Fehlermeldung. Erfolgt die Definition auch in den Kontrollfeldern des Formulars, dann erscheint eine deutschsprachige Meldung mit dem genauen Hinweis, welches Feld denn nun ausgefüllt werden muss.

  Die obige Meldung passiert besonders häufig, wenn das Primärschlüsselfeld, hier 'ID', nicht in das Formular mit aufgenommen wurde, da es ja als automatisch hoch zählendes AutoWert-Feld gedacht war. Leider wurde aber die entsprechende Definition als AutoWert-Feld schlicht vergessen. Also muss dies entweder nachgeholt werden oder das Feld muss im Formular mit erscheinen, damit ein Wert eingegeben werden kann.

  Die nachträgliche Definition eines Feldes als AutoWert-Feld ist manchmal etwas schwierig, wenn die Tabelle mit anderen in der Beziehungsdefinition bereits verknüpft wurde oder wenn bereits mit einer Ansicht auf die Tabelle zugegriffen wird. Hier müssen alle Verbindungen gelöst werden damit die Tabelle gerade im Bereich des Primärschlüssels wieder editierbar wird. Der Inhalt des SQL-Befehls, mit dem z. B. eine Ansicht erstellt wurde, kann ja gegebenenfalls als Abfrage zwischen durch abgespeichert werden.

- **Integrity constraint violation - no parent SYS_FK_95 table: Tabellenname in statement ...**


  **Fehler beim Einfügen des neuen Datensatzes.**

  **Fehler in der Funktionsfolge**

vor dem Abspeichern dieses Feld abgehakt und der Datensatz kann problemlos gespeichert wer-
den.

Die über die anderen Formulare weitergegebenen Werte werden anscheinend erst dann in die ent-
sprechenden Felder eingefügt, wenn wenigstens eine zusätzliche Aktion im Formular stattfindet.

Suchen und Filtern in Formularen über die Navigationsleiste

Die Navigationsleiste der Formulare bietet verschiedene Such- und Filtermöglichkeiten an. Die ein-
zelnen Elemente der Navigationsleiste wurden bereits weiter oben aufgezählt.

Datensatzsuche mit Parametern

Die Datensatzsuche im Formular bietet die Möglichkeit, das entsprechende Formular sowie die Unterformulare zu durchsuchen. Es ist allerdings nicht möglich, alle Formulare gleichzeitig zu durchsuchen.


Leider kennzeichnen diese Suche aber die folgenden Nachteile:

- Die Parametersuche ist aber für den Normalgebrauch viel zu umständlich zu handhaben.
- Die Suchfunktion selbst ist äußerst langsam, da nicht die Abfragefunktionen der Datenbank genutzt werden.
- Die Suche funktioniert nur innerhalb eines Formulars, dort dann aber auch gegebenenfalls für alle Felder. Unterformulare müssen separat angewählt werden.
- Es wird nur in dem Unterformular gesucht, das zu dem aktuellen Hauptformular passt. Einträge anderer Unterformulare lassen sich nicht ermitteln. So ist es z.B. nicht möglich, einen Untertitel eines anderen Mediums als des gerade angezeigten Mediums zu finden.
- Die Suche funktioniert bei Listenfeldern auf der Basis des Schlüsselfeldes (Fremdschlüssel), das in der Tabelle abgespeichert ist. Es lassen sich nicht die angezeigten Begriffe verwenden.

Filterung mit dem Autofilter

Der Autofilter kann direkt in der Navigationsleiste angewählt werden. Wird auf ein Feld im Hauptformular geklickt, so wird treffsicher nach diesem Feld gefiltert. Der Autofilter funktioniert auch sicher mit Listenfeldern, hat hier also einen klaren Vorteil gegenüber der händischen Eingabe von Fremdschlüsselwerten zu verbuchen.
In dem obigen Beispiel wurde einfach ein Datensatz gesucht, bei dem im Feld «Medienart» 'CD' steht. Anschließend wurde der Autofilter betätigt. Von den ursprünglich 9 Datensätzen der Medien-Tabelle werden jetzt noch 3 Datensätze in der Datensatzzählung angezeigt. Obwohl nur in zwei Datensätzen im Unterformular eine «Nr.» '3' wie oben existiert werden alle anderen Datensätze des Hauptformulars weiter angezeigt. Im Unterformular erscheinen hingegen nur die Felder mit «Nr.» '3'.

Der Autofilter kann die folgenden Probleme allerdings nicht lösen:

- Die Suche in Unterformularen führt dazu, dass zwar in den Unterformularen nur der entsprechende Wert angezeigt wird, die Formulare selbst aber weiter komplett angezeigt werden, auch wenn sie den entsprechenden Wert im Unterformular gar nicht verzeichnen können. Wird z.B. nach einem Medium mit einem zweiten Verfasser gesucht, so werden trotzdem auch alle Medien angezeigt, für die nur ein Verfasser verzeichnet ist. Hier bleibt das Feld für die Verfasser schlicht leer.

- Die Wahl des Filterinhaltes muss eindeutig sein. Es gibt nicht die Möglichkeit, über Ähnlichkeitsuche zu einem entsprechenden Wert zu kommen, da eben nur der Originalwert aus dem Formular verwandt wird.
Filterung mit dem formularbasierten Filter

Der formularbasierte Filter bietet statt eines Filtersymbols das Formular zur Eingabe von entsprechenden Filterwerten an. Hier kann gleichzeitig nach mehreren Werten gefiltert werden. Diese Werte können entweder zusammen als gemeinsame Bedingung formuliert werden (UND) oder als unterschiedliche Bedingungen gesetzt werden (ODER).

Die Eingabe der Filterwerte muss nicht eindeutig sein. Hier hilft dann die Formulierung von Bedingungen, wie sie bei der Filterung von Tabellen geschildert wurde. So führt ein «WIE '%Auge%'», eingetragen in das Eingabefeld für den Titel des Mediums, direkt zur Auffindung aller Titel, in denen 'Auge' enthalten ist – in der Beispieldatenbank «Medien ohne Makros» z.B. zur Anzeige des Titels «Im Augenblick».

Filter für die Listenfelder werden direkt über die Listenfelder ausgewählt. Eine Eingabe von Fremdschlüsselwerten ist also nicht nötig.

Hier werden alle Datensätze gesucht, die die Medienart 'CD' haben (entspricht dem Fremdschlüsselwert '1') und gleichzeitig im Titel irgendwo die Buchstabenfolge 'Auge' haben. Bei der Eingabe «WIE '%Auge%'» ersetzt die grafische Benutzeroberfläche das in SQL übliche Allroundzeichen für mehrere Werte «%» durch das Zeichen «*», das für viele Nutzer eher gebräuchliche Allroundzeichen.


Aus dem Unterformular wird noch der Verfasser 'Edmunds, Dave' aus dem Listenfeld ausgewählt. Der entsprechende Fremdschlüsselwert ist hier die '8'.

Formulare: Suchen und Filtern in Formularen über die Navigationsleiste
Der formularbasierte Filter zeigt bei der Eingabe nur die Felder an, die vor dem ODER mit einem Haken versehen sind. So lassen sich auch mehrere Bedingungen für ein Feld angeben. Leider unterschlägt der Filter dabei die Anzeige des Listenfeldes «Medienart» mit dem entsprechenden Wert im Hauptformular, während dies im Tabellenkontrollfeld des Unterformulars gelingt.

Statt der Navigationsleiste werden während des Erstellen des Filters unter dem Formular lediglich drei Auswahlmöglichkeiten für den formularbasierten Filter dargestellt: Filtern nach den Vorgaben, Filterwerte hinzufügen und die Anzeige des formularbasierten Filters einfach schließen.

Der Filter wird angewandt und das Formular nach den vorgegebenen Werten gefiltert.

Ist die Filterung aktiv, so wird die Zahl der Datensätze auf die Datensätze mit einem Treffer eingeschränkt. Hier gibt es insgesamt 5 Datensätze mit Treffern. In dem obigen Bild stimmt zwar nicht der Titel und die Medienart, dafür aber die Bedingung, dass das Erscheinungsjahr vor 1980 liegen soll.


Diese schon sehr gelungene Filterfunktion leidet allerdings an den folgenden Nachteilen:

- Die Filterung funktioniert wieder nur im Hauptformular. Im Unterformular werden wie bei Autofilter zwar die fehlenden Werte unterdrückt. Dies wirkt sich aber nicht auf die Anzeige
im Hauptformular aus. Es werden also im Hauptformular auch Inhalte angezeigt, die den gefilterten Inhalt aus dem Unterformular nicht vorweisen können.

- Es sind umfangreiche Kenntnisse der Bedingungsformulierung notwendig, wenn mehr als nur die kompletten Begriffe gesucht werden sollen. Hier müssen Kenntnisse über Abfragetechniken vorhanden sein, die aber den meisten suchmaschinenverwöhnten Nutzern fehlen.

Filterung über den Standardfilter

Die Filterung über den Standardfilter kann durch die Anzeige der Datenquelle als Tabelle erreicht werden. Dort gelten dann die gleichen Regeln wie bei der Filterung mit dem Standardfilter in Tabellen.

Filterung über den Standardfilter

Die Filterung über den Standardfilter kann durch die Anzeige der Datenquelle als Tabelle erreicht werden. Dort gelten dann die gleichen Regeln wie bei der Filterung mit dem Standardfilter in Tabellen.
Tabelle "Medien" werden hier die Fremdschlüsselfelder nicht mit ihren Schlüsselwerten sondern mit den Inhalten dargestellt. Ein Klick auf das Feld «Medienart» zeigt, dass hier aus ein Listenfeld nach den Vorgaben des Formulars erstellt wurde.

<table>
<thead>
<tr>
<th>Hinweis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dieser Bug tritt auch in Tabellenkontrollfeldern auf. Die Darstellung ist korrekt, wenn statt eines Zahlfeldes im Formular ein formatierbares Feld genommen wird.</td>
</tr>
</tbody>
</table>

Die Tabellenansicht des Formulars hat die folgenden Nachteile für diese Recherchemethode:

- Die Suche funktioniert nur innerhalb der einem Formular zugrundeliegenden Tabelle oder Abfrage, nicht in einem dazugehörigen Unterformular.

**Fazit**

Sämtliche Such- und Filterfunktionen eignen sich für Formulare nur eingeschränkt. Die eingebaute Suche ist viel zu langsam und schränkt die Zahl der Datensätze nicht auf die mit entsprechenden Treffern ein. Die Filter funktionieren grundsätzlich nur innerhalb eines Formulars. Soll nach Werten im Unterformular gefiltert werden, so ist nur das aktuelle Unterformular betroffen. Es werden einfach nicht alle Datensätze korrekt ermittelt.

Aus diesem Grund wurde weiter oben in die Formulare der Beispieldatenbank eine entsprechend einfacher zu bedienende Suchfunktion integriert, die allerdings etwas Handarbeit und SQL-Kenntnis verlangt.

**Dateneingabe und Navigation**

Wird ein Formular so eingestellt, dass beim Öffnen gleich der Kontrollelemente-Fokus aktiviert wird, so steht der Cursor direkt im ersten Eingabefeld. Statt mit der Maus von einem Feld zum anderen zu springen ist bei Formularen die Bewegung mit der Tabulatortaste von einem Feld zum nächsten üblich.

Manchmal erfordern Formulare auch Sprünge zu einem anderen Formularfeld. Diese können natürlich mit der Maus geregelt werden, sind aber auch über Kurzbefehle anspringbar.

![Name](Name)


Leider funktioniert das Anspringen von Feldern nur innerhalb eines Formulars, nicht aber bei einer Konstruktion mit Unterformularen. Ein Sprung aus einem Unterformular zu einem Hauptformular ist so also zur Zeit nicht möglich.

Aus einem Tabellenkontrollfeld sollte der Cursor mit **Strg + Tab** herausspringen. Manchmal kann es passieren, dass die Tastenkombination nicht den gewünschten Erfolg erzielt. Daher hier eine völlig andere Möglichkeit:  

In dem Unterformular wird eine Schaltfläche angelegt. Der Titel der Schaltfläche heißt z.B. «~neu». Das «~» sorgt, wie oben erwähnt, dafür, dass der Button mit dem Kürzel **Alt + N** ausgelöst werden kann. In den Zusatzinformationen des Buttons wird außerdem noch der Name des Feldes im Hauptformular benannt, zu dem der Cursor springen soll. Die Schaltfläche wird über **Eigenschaften → Ereignisse → Aktion ausführen** mit dem folgenden Makro verbunden:

```
SUB JumpToMainform(oEvent AS OBJECT)
    DIM oField AS OBJECT
    DIM oForm AS OBJECT
    DIM oDoc AS OBJECT
    DIM oController AS OBJECT
    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 "ä"
            IF oForm.isLast() THEN
                oForm.MoveToInsertRow()
            ELSE
                oForm.Next()
            END IF
    END SELECT
    oDoc = thisComponent
    oController = oDoc.getCurrentController()
    oView = oController.getControl(oForm.getByname(oField.Tag))
    oView.setFocus
END SUB
```

Das Makro kann dazu genutzt werden, nur ins Hauptformular zurück zu springen. Es kann bei der obigen Einstellung auch dazu dienen, direkt einen neuen Datensatz anzulegen oder auch über einen weiteren Button mit dem Kürzel **Alt + ä** (für «nächster Datensatz») zum nächsten Datensatz zu springen. Befindet sich der Cursor des Hauptformulars allerdings schon auf dem letzten Datensatz, so muss der Sprung zu einem neuen Datensatz erfolgen. Weiteres zu Makros siehe im entsprechenden Kapitel.

Der Sprung auf die Schaltfläche gelingt nur mit einer sichtbaren Schaltfläche. Die Schaltfläche kann aber ruhig eine ganz geringe Ausdehnung oder eine Breite von 0 cm und eine Höhe von 0 cm haben. Bei der Definition einer nicht vorhandenen Breite und einer nicht vorhandenen Höhe kann es allerdings vorkommen, dass der Button als Strich im Formular über die gesamte Bildschirmbreite sichtbar wird.

---

7 Siehe hierzu auch die Beispieltdatenbank «Beispiel_Cursorsprung_Subform_Mainform.odb»

---

Formulare: Dateneingabe und Navigation  222
Drucken aus Formularen


Details zu der Erstellung von Serienbriefen sind in dem Kapitel «Datenbank-Anbindung» enthalten.
Abfragen
Allgemeines zu Abfragen

Abfragen an eine Datenbank sind das mächtigste Werkzeug, was uns zur Verfügung steht, um Datenbanken sinnvoll zu nutzen. Sie fassen Daten aus unterschiedlichen Tabellen zusammen, berechnen gegebenenfalls irgendwelche Ergebnisse, filtern einen ganz bestimmten Datensatz aus einer Unmenge an Daten mit hoher Geschwindigkeit heraus. Die großen Internetdatenbanken, die viele täglich nutzen, haben ihren Hauptsinn darin, dass aus der Unmenge an Informationen durch geschickte Wahl der Schlüsselwörter schnell ein brauchbares Ergebnis für den Nutzer geliefert wird – einschließlich natürlich der zu den Suchbegriffen gehörenden Anzeigen, die zum Kauf animieren sollen.

Eingabemöglichkeiten für Abfragen

Die Eingabe von Abfragen kann sowohl in der GUI als auch direkt per SQL erfolgen. In beiden Fällen öffnet sich ein Fenster, das es ermöglicht, die Abfragen auszuführen und gegebenenfalls zu korrigieren.

Abfrageerstellung mit der grafischen Benutzeroberfläche

Die Erstellung von Abfragen mit dem Assistenten wird hier nicht dargestellt, da der Assistent zwar bei wenig Tabellen schnell zu einem Ergebnis führt, für eine größere Datenbank aber wenig ziel führend ist. Hier wird stattdessen die direkte Erstellung über Abfrage in der Entwurfsansicht erstellen erklärt.

Nach einem Aufruf der Funktion erscheinen zwei Fenster. Ein Fenster stellt die Grundlagen für den Design-Entwurf der Abfrage zur Verfügung, das andere dient dazu, Tabellen, Ansichten oder Abfragen der Abfrage hinzuzufügen.

Da unser einfaches Formular auf der Tabelle "Ausleihe" beruhte, wird zuerst einmal an dieser Tabelle die Grundkonstruktion von Abfragen mit dem Abfrageeditor erklärt.
Aus den zur Verfügung stehenden Tabellen wird die Tabelle Ausleihe ausgewählt. Dieses Fenster bietet die Möglichkeit, gleich mehrere Tabellen (und unter diesen auch Ansichten) sowie Abfragen miteinander zu kombinieren. Jede gewünschte Tabelle wird markiert (linke Maustaste) und dann dem grafischen Bereich des Abfrageeditors hinzugefügt.

Sind alle erforderlichen Tabellen ausgewählt, so wird dieses Fenster geschlossen. Gegebenenfalls können später noch mehr Tabellen und Abfragen hinzugefügt werden. Ohne eine einzige Tabelle lässt sich aber keine Abfrage erstellen, so dass eine Auswahl zu Beginn schon sein muss.

_Abbildung 37_ zeigt die grundsätzliche Aufteilung des grafischen Abfrageeditors: Im grafischen Bereich werden die Tabellen angezeigt, die mit der Abfrage zusammen hängen sollen. Hier kann auch ihre Beziehung zueinander in Bezug auf die Abfrage angegeben werden. Im tabellarischen Bereich erfolgt die Auswahl der Felder, die angezeigt werden sollen sowie Bedingungen, die mit diesen Feldern verbunden sind.

Ein Klick mit der Maustaste auf das Feld der ersten Spalte im tabellarischen Bereich öffnet die Feldauswahl:
Hier stehen jetzt alle Felder der Tabelle “Ausleihe” zur Verfügung. Die Schreibweise: **Tabellenname.Feldname** – deshalb beginnen alle Feldbezeichnungen hier mit dem Begriff "Ausleihe.".

Eine besondere Bedeutung hat die markierte Feldbezeichnung **Ausleihe.***. Hier wird mit einem Klick jeder Feldname der zugrundeliegenden Tabelle in der Abfrage wiedergegeben. Wenn allein diese Feldbezeichnung mit dem Jokerzeichen «*» für alle Felder gewählt wird unterscheidet sich die Abfrage nicht von der Tabelle.

<table>
<thead>
<tr>
<th>Feld</th>
<th>Alias</th>
<th>Tabelle</th>
<th>Sortier</th>
<th>Sichtbar</th>
<th>Funktion</th>
</tr>
</thead>
</table>

**Tipp**

Sollen schnell alle möglichen Felder aus Tabellen in die Abfrage übertragen werden, so kann auch direkt im grafischen Bereich auf die Tabellenübersicht geklickt werden. Mit einem Doppelklick auf ein Feld wird das Feld im tabellarischen Bereich an der nächsten freien Position eingefügt.

Abfragen: Eingabemöglichkeiten für Abfragen
Grundsätzlich lässt sich eine Abfrage, die eine Fehlerrückmeldung bei der zugrundeliegenden Datenbank produziert, gar nicht erst abspeichern.

Abbildung 38: Abfrage bearbeitbar

<table>
<thead>
<tr>
<th>ID</th>
<th>Medien_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
</tr>
</tbody>
</table>

Abbildung 39: Abfrage nicht bearbeitbar

<table>
<thead>
<tr>
<th>Medien_ID</th>
<th>Lease</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

Besonderes Augenmerk sollte in dem obigen Test einmal auf die erste Spalte des dargestellten Abfrageergebnisses geworfen werden. Auf der linken Seite der Tabelle erscheint immer der Datensatzmarkierer, der hier auf den ersten Datensatz als aktivem Datensatz hinweist. Während in Abbildung 38 aber das erste Feld des ersten Datensatzes grün markiert ist zeigt das erste Feld in Abbildung 39 nur eine gestrichelte Umrandung an. Die grüne Markierung deutet bereits an, dass hier im Feld selbst etwas geändert werden kann. Die Datensätze sind also änderbar. In Abbildung 38 ist außerdem eine zusätzliche Zeile zur Eingabe neuer Daten vorhanden, in der für das Feld "ID" schon <AutoWert> vorgemerkt ist. Auch hier also sichtbar, dass Neueingaben möglich sind.

Grundsätzlich sind dann keine Neueingaben möglich, wenn der Primärschlüssel der abgefragten Tabelle nicht in der Abfrage enthalten ist.

<table>
<thead>
<tr>
<th>Feld</th>
<th>ID</th>
<th>Medien_ID</th>
<th>Leser_ID</th>
<th>Leih_Datum</th>
<th>Rueck_Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td></td>
<td></td>
<td></td>
<td>Ausleihdatum</td>
<td>Rückgabedatum</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>ID</th>
<th>Medien_ID</th>
<th>Leser_ID</th>
<th>Ausleihdatum</th>
<th>Rueckgabedatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>3</td>
<td>0</td>
<td>09.12.11</td>
<td></td>
</tr>
</tbody>
</table>

Entsprechend ist in der Tabellenansicht der Alias statt der eigentlichen Feldbezeichnung zu sehen.

Dem Feld "Rueck_Datum" wurde nicht nur ein Alias sondern auch ein Kriterium zugewiesen, nach dem nur die Datensätze angezeigt werden sollen, bei denen das Feld "Rueck_Datum" leer ist. Die
Angabe erfolgt hier in deutscher Sprache, wird dann aber in der eigentlichen Abfrage in SQL über-
setzt.
Durch dieses Ausschlusskriterium werden nur die Datensätze von Medien angezeigt, die ein
Medium enthalten, das noch nicht zurückgegeben wurde.

Um die SQL-Sprache besser kennen zu lernen, empfiehlt es sich immer wieder einmal vom De-
sign-Modus aus in den SQL-Darstellungsmodus zu wechseln.

<table>
<thead>
<tr>
<th>ID</th>
<th>Medien_ID</th>
<th>Leser_ID</th>
<th>Ausleihdatum</th>
<th>Rückgabedatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>3</td>
<td>0</td>
<td>09.12.11</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>0</td>
<td>25.02.12</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>04.04.12</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>1</td>
<td>04.04.12</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>1</td>
<td>22.04.12</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>9</td>
<td>04.04.12</td>
<td></td>
</tr>
</tbody>
</table>

SELECT "ID", 'Medien_ID', 'Leser_ID', "Leih_Datum" AS "Ausleihdatum", "Rueck_Datum" AS "Rueckgabedatum"
FROM "Ausleihe"
WHERE "Rueck_Datum" IS NULL
ORDER BY "Leser_ID" ASC, "Ausleihdatum" ASC

Hier wurde die durch die Auswahl erstellte SQL-Formulierung sichtbar gemacht. Für die bessere
Übersicht ist die Ansicht mit Zeilenumbrüchen versehen worden. Leider speichert der Editor diese
Zeilenumbrüche nicht mit ab, so dass beim nächsten Aufruf die Abfrage wieder komplett als eine
durchgängige Zeile mit Umbruch am Fensterrand wiedergegeben wird.

Über SELECT wird die Auswahl gestartet. Mit AS werden die Aliasbezeichnungen eingeführt. FROM
zeigt auf die Tabellenquelle der Abfrage. WHERE gibt die Bedingung für die Abfrage wieder, hier
also, dass der Inhalt des Feldes "Rueck_Datum" leer ist (IS NULL). Mit ORDER BY wird die Sortie-
 rung definiert, und zwar als aufsteigend (ASC – ascending) für die beiden Felder "Leser_ID" und
"Ausleihdatum". Diese Sortierung zeigt auch, dass die Zuweisung eines Alias das Feld
"Leih_Datum" auch in der Abfrage selbst mit dem Alias ansprechbar macht.

Bisher sind die Felder "Medien_ID" und "Leser_ID" nur als Zahlenfelder sichtbar. Welchen Namen
der Leser hat bleibt unklar. Um dies in einer Abfrage anzuzeigen muss die Tabelle Leser eingebun-
den werden. Um die folgende Ansicht zu erhalten muss in den Design-Modus zurückgeschaltet
werden. Danach kann dann eine neue Tabelle in der Design-Ansicht hinzugefügt werden.
Hier können im Nachhinein weitere Tabellen oder Abfragen in der grafischen Benutzeroberfläche sichtbar gemacht werden. Sind bei der Erstellung der Tabellen Beziehungen geklärt worden (siehe Kapitel «Beziehungen zwischen Tabellen allgemein»), dann werden die Tabellen entsprechend direkt miteinander verbunden angezeigt:

Fehlt die Verbindung, so kann hier durch ein Ziehen mit der Maus von "Ausleihe"."Leser_ID" zu "Leser"."ID" eine direkte Verknüpfung erstellt werden.

**Hinweis**

Eine Verbindung von Tabellen geht zur Zeit nur, wenn es sich um die interne Datenbank oder ein relationales Datenbanksystem handelt. Tabellen einer Tabellenkalkulation z. B. lassen sich so nicht miteinander verbinden. Sie müssen dazu in eine interne Datenbank importiert werden.

Um die Verbindung der Tabellen herzustellen reicht ein Import ohne zusätzliche Erstellung eines Primärschlüssels aus.

Jetzt können im tabellarischen Bereich auch die Felder der Tabelle "Leser" ausgewählt werden. Die Felder werden dabei erst einmal am Schluss der Abfrage angeordnet.

Mit der Maus kann in dem tabellarischen Bereich des Editors die Lage der Felder korrigiert werden. Hier wird z. B. gerade das Feld "Vorname" direkt vor das Feld "Leih_Datum" verlegt.

<table>
<thead>
<tr>
<th>ID</th>
<th>Medien_ID</th>
<th>Leser_ID</th>
<th>Vorname</th>
<th>Nachname</th>
<th>Ausleiheidatum</th>
<th>Rückgabedatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>3</td>
<td>0</td>
<td>Bert</td>
<td>Lederstrumpf</td>
<td>09.12.11</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>0</td>
<td>Bert</td>
<td>Lederstrumpf</td>
<td>25.02.12</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>Bert</td>
<td>Lederstrumpf</td>
<td>04.04.12</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>1</td>
<td>Heinrich</td>
<td>Müller</td>
<td>04.04.12</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>1</td>
<td>Heinrich</td>
<td>Müller</td>
<td>22.04.12</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>9</td>
<td>Terence</td>
<td>Nobody</td>
<td>04.04.12</td>
<td></td>
</tr>
</tbody>
</table>

**Abfragen: Eingabemöglichkeiten für Abfragen**
Die Namen wurden jetzt sichtbar gemacht. Die "Leser_ID" ist eigentlich überflüssig. Auch ist die Sortierung nach "Nachname" und "Vorname" eigentlich sinnvoller als nach der "Leser_ID".

Diese Abfrage eignet sich nicht mehr für Base als Abfrage mit Eingabemöglichkeit, da zu der neu hinzugekommenen Tabelle Leser der Primärschlüssel fehlt. Erst wenn auch dieser Primärschlüssel eingebaut wird ist die Abfrage wieder editierbar – allerdings komplett editierbar, so dass auch die Namen der Leser geändert werden können. Die Möglichkeit der Editierbarkeit ist also sehr vorsichtig zu nutzen, gegebenenfalls über ein Formular einzuschränken.

Selbst wenn die Abfrage weiter editierbar ist, lässt sie sich nicht so komfortabel nutzen wie ein Formular mit Listenfeldern, die zwar die Lesernamen anzeigen, aber die "Leser_ID" an die Tabelle weitergeben. Listenfelder lassen sich in eine Abfrage nicht einfügen. Sie sind den Formularen vorbehalten.


Folgende Abfrage funktioniert auch ohne Tabellenname vor den Feldnamen:

```
SELECT "Ausleihe"."ID", "Ausleihe"."Medien_ID", "Ausleihe"."Leser_ID",
"Leser"."Vorname", "Leser"."Nachname", "Ausleihe"."Leih_Datum" AS
"Ausleihtdatum", "Ausleihe"."Rueck_Datum" AS "Rueckgabedatum"
FROM "Ausleihe", "Leser"
WHERE "Ausleihe"."Leser_ID" = "Leser"."ID" AND
"Ausleihe"."Rueck_Datum" IS NULL
ORDER BY "Ausleihe"."Leser_ID" ASC, "Ausleihtdatum" ASC
```

Wird einem Feld in der Abfrage ein Alias zugewiesen, so kann sie z. B. in der Sortierung mit diesem Alias ohne einen Tabellennamen angesprochen werden. Sortierungen werden in der grafischen Benutzeroberfläche nach der Reihenfolge der Felder in der Tabellenansicht vorgenommen. Sollte stattdessen zuerst nach "Ausleihtdatum" und dann nach "Ausleihe"."Leser_ID" sortiert werden, so kann dies erzeugt werden, indem

Abfragen: Eingabemöglichkeiten für Abfragen 232
• die Reihenfolge der Felder im tabellarischen Bereich der grafischen Benutzeroberfläche geändert wird,
• ein zweites Feld eingefügt wird, das auf unsichtbar geschaltet ist und nur die Sortierung gewährleisten soll (wird allerdings beim Editor nur vorübergehend angenommen, wenn kein Alias definiert wurde) oder
• der Text für die ORDER BY – Anweisung im SQL-Editor entsprechend umgestellt wird.

Die Beeinflussung der Sortierreihenfolge arbeitet je nach Version nicht ganz fehlerfrei. Wird in LO 3.3.4 die Reihenfolge anders gewählt als durch die GUI vorgegeben, so funktioniert die Abfrage korrekt. Ein erneutes Aufrufen der Abfrage zur Bearbeitung zeigt aber die Einstellung der Sortierung nach Reihenfolge der Felder in der GUI. Die Kontrolle ergibt dann auch eine entsprechend geänderte Sortierreihenfolge wie angezeigt wieder. Sobald also die Abfrage nach einer zusätzlichen Änderung an anderer Stelle gespeichert wird, ist die Sortierung leider unbeabsichtigt mit geändert worden. In LO 3.5.3 RC2 wird die Sortierung aus der SQL-Ansicht korrekt übernommen und entsprechend mit nicht sichtbaren Feldern in der grafischen Benutzeroberfläche angezeigt.

Funktionen in der Abfrage

Mittels Funktionen lässt sich aus Abfragen auch mehr ersehen als nur ein gefilterter Blick auf die Daten einer oder mehrerer Tabellen. In der folgenden Abfrage wird, abhängig von der "Leser_ID", gezählt, wie viele Medien ausgeliehen wurden.

<table>
<thead>
<tr>
<th>Feld</th>
<th>ID</th>
<th>Leser_ID</th>
<th>Rueck_Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>Anzahl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tabelle</td>
<td>Ausleihe</td>
<td>Ausleihe</td>
<td>Ausleihe</td>
</tr>
<tr>
<td>Sortierung</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sichtbar</td>
<td>✓</td>
<td>✓</td>
<td>□</td>
</tr>
<tr>
<td>Funktion</td>
<td>Anzahl</td>
<td>Gruppierung</td>
<td></td>
</tr>
<tr>
<td>Kriterium</td>
<td></td>
<td>IST LEER</td>
<td></td>
</tr>
</tbody>
</table>


Für die "Leser_ID", die ja Rückschlüsse auf den Leser zulässt, wird als Funktion die Gruppierung gewählt. Dadurch werden die Datensätze nach der "Leser_ID" zusammengefasst. So zählt denn die Anweisung die Datensätze, die zu jeder "Leser_ID" passen.

Als Kriterium ist wie in den vorhergehenden Beispielen das "Rueck_Datum" auf IST LEER gesetzt.
Die Abfrage zeigt im Ergebnis, dass z. B. "Leser_ID" '0' insgesamt noch 3 Medien entliehen hat. Wäre die Funktion Anzahl statt der "ID" dem "Rueck_Datum" zugewiesen worden, so würden für alle "Leser_ID" jeweils '0' entliehene Medien dargestellt, da ja "Rueck_Datum" als LEER vordefiniert ist.

Die entsprechende Formulierung für den SQL-Code ist oben wieder abgebildet. Aus dem Begriff Anzahl der deutschen GUI wird COUNT(). Aus dem Begriff Gruppierung wird der Zusatz GROUP BY.

Insgesamt stehen über die grafische Benutzeroberfläche folgende Funktionen zur Verfügung, die ihre Entsprechung zu Funktionen in der zugrundeliegenden HSQLDB haben:

Eine Erläuterung zu den Funktionen ist in dem folgenden Kapitel Abfrageerweiterungen im SQL-Modus nachzulesen.

Wird einem Feld in einer Abfrage eine Sammelfunktion hinzugefügt, so müssen alle anderen Felder der Abfrage auch mit Funktionen versehen sein, sofern die Felder sichtbar sein sollen. Dies liegt daran, dass in einem Datensatz nicht plötzlich zwischendurch Felder mehrere Datensätze abbilden können. Wird dies nicht beachtet, so erscheint die folgende Fehlermeldung:

Abfragen: Eingabemöglichkeiten für Abfragen
Etwas frei übersetzt: Der folgenden Ausdruck enthält ein Feld ohne eine der Sammelfunktionen oder eine Gruppierung.

Danach wird die gesamte Abfrage aufgelistet, leider ohne das Feld konkret zu benennen. Hier wurde einfach das Feld "Rueck_Datum" als sichtbar hinzugefügt. Dieses Feld hat keine Funktion zugewiesen bekommen und ist auch nicht in der Gruppierung enthalten.

Die über den Button Mehr erreichbaren Informationen sind für den Normalnutzer einer Datenbank nicht aufhellender. Hier wird lediglich zusätzlich noch der SQL-Fehlercode aufgeführt.

Innerhalb der GUI können auch die Grundrechenarten sowie weitere Funktionen angewandt werden.


Vorsicht

Werden in der grafischen Benutzeroberfläche Zahlen mit Nachkommastellen eingegeben, so ist auf jeden Fall darauf zu achten, dass statt eines Kommas ein Punkt der Trenner für Dezimalzahlen in SQL ist. Kommata sind hingegen die Trenner der Felder. Ein Komma in der GUI-Eingabe bringt LO 3.3.4 direkt zum Totalabsturz. Seit der Version 3.5.3 ist dies nicht mehr der Fall. Stattdessen werden neue Abfragefelder gegründet, die die Nachkommastellen ausgeben.

Eine Eingabe mit Komma in der SQL-Ansicht führt hingegen dazu, dass ein weiteres Feld allein mit dem Zahlenwert der Nachkommastelle angezeigt wird. Dies entspricht dem Verhalten der grafischen Benutzeroberfläche in LO 3.5.3.

Abfragen: Eingabemöglichkeiten für Abfragen
Die Abfrage ermittelt jetzt für jedes noch entliehene Medium anhand der herausgegebenen Mahnungen und der zusätzlich eingefügten Multiplikation die Mahngebühren. Die folgende Abfragekonstruktion hilft weiter, wenn die Gebühren für jeden Leser berechnet werden sollen:

```
FROM "Mahnung", "Ausleihe"
WHERE "Mahnung"."Ausleihe_ID" = "Ausleihe"."ID"
AND "Ausleihe"."Rueck_Datum" IS NULL
GROUP BY "Ausleihe"."ID", "Ausleihe"."Medien_ID", "Ausleihe"."Leser_ID"
```

Die Felder "Ausleihe"."ID" und "Ausleihe"."Medien_ID" wurden entfernt. Sie erzeugten in der vorherigen Abfrage über die Gruppierung für jedes Medium einen separaten Datensatz. Jetzt wird nur noch nach den Lesern gruppiert. Das Abfrageergebnis sieht dann so aus:

<table>
<thead>
<tr>
<th>Feld</th>
<th>Leser_ID</th>
<th>Datum</th>
<th>Anzahl( &quot;Mahnung&quot;.&quot;Datum&quot; ) * 2</th>
<th>Rueck_Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td></td>
<td></td>
<td>Mahnzahl</td>
<td>Mahnbetrag</td>
</tr>
<tr>
<td>Tabelle</td>
<td>Ausleihe</td>
<td>Mahnung</td>
<td>Ausleihe</td>
<td></td>
</tr>
<tr>
<td>Sortierung</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sichtbar</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Funktion</td>
<td>Gruppierung</td>
<td>Anzahl</td>
<td>IST LEER</td>
<td></td>
</tr>
</tbody>
</table>

Die Felder "Ausleihe"."ID" und "Ausleihe"."Medien_ID" wurden entfernt. Sie erzeugten in der vorherigen Abfrage über die Gruppierung für jedes Medium einen separaten Datensatz. Jetzt wird nur noch nach den Lesern gruppiert. Das Abfrageergebnis sieht dann so aus:

<table>
<thead>
<tr>
<th>Leser_ID</th>
<th>Mahnzahl</th>
<th>Mahnbetrag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Statt die Medien für "Leser_ID" '0' separat aufzulisten werden alle Felder aus "Mahnung"."Datum" zusammengezählt und die Summe von 8,- € als Mahngebühr ermittelt.

**Beziehungsdefinition in der Abfrage**

Werden Daten in Tabellen oder einem Formular gesucht, so beschränkt sich die Suche in der Regel auf eine Tabelle bzw. auf ein Formular. Selbst der Weg von einem Hauptformular zu einem Unterformular ist für die eingebauten Suchfunktionen nicht gangbar. Da bietet es sich dann an, zu durchsuchende Daten mit einer Abfrage zusammenzufassen.

Die Verknüpfungsbedingung muss zum Bearbeiten geöffnet werden, damit alle gewünschten Datensätze angezeigt werden. Es handelt sich hier nicht um die Verknüpfung von Tabellen im Relationenentwurf sondern um die Verknüpfung in einer Abfrage.
Standardmäßig steht die Verknüpfung als *Innerer Verbund* zur Verfügung. Das Fenster gibt darüber Aufschluss, wie diese Form der Verknüpfung sich auswirkt.

Als beteiligte Tabellen werden die beiden vorher ausgewählten Tabellen gelistet. Sie sind hier nicht wählbar. Die beteiligten Felder der beiden Tabellen werden aus der Tabellendefinition ausgelesen. Ist eine Beziehung in der Tabellendefinition nicht vorgegeben, so kann sie hier für die Abfrage erstellt werden. Eine saubere Datenbankplanung mit der HSQLDB sieht aber so aus, dass auch an diesen Feldern nichts zu verstellen ist.

Wichtigste Einstellung ist die Option des *Verbundes*. Hier können Verknüpfungen so gewählt werden, dass alle Datensätze von der Tabelle "Untertitel" und nur die Datensätze aus "Medien" gewählt werden, die in der Tabelle "Untertitel" auch "Untertitel" verzeichnet haben.

Umgekehrt kann gewählt werden, dass auf jeden Fall alle Datensätze aus der Tabelle "Medien" angezeigt werden – unabhängig davon, ob für sie auch "Untertitel" existieren.

Die Option *Natürlich* setzt voraus, dass die zu verknüpfenden Felder in den Tabellen gleich lauten. Auch von dieser Einstellung ist Abstand zu nehmen, wenn bereits zu Beginn bei der Datenbankplanung die Beziehungen definiert wurden.
Für den Typ → Rechter Verbund zeigt die Beschreibung an, dass aus der Tabelle "Medien" auf jeden Fall alle Datensätze angezeigt werden. Da es keine "Untertitel" gibt, die nicht in "Medien" mit einem "Titel" verzeichnet sind, sehr wohl aber "Titel" in "Medien", die nicht mit einem "Untertitel" versehen sind, ist dies also die richtige Wahl.

<table>
<thead>
<tr>
<th>Titel</th>
<th>Untertitel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Der kleine Hobbit</td>
<td></td>
</tr>
<tr>
<td>Das sogenannte Böse</td>
<td></td>
</tr>
<tr>
<td>Eine kurze Geschichte der Zeit</td>
<td></td>
</tr>
<tr>
<td>Traditionelle und kritische Theorie</td>
<td></td>
</tr>
<tr>
<td>Die neue deutsche Rechtschreibung</td>
<td></td>
</tr>
<tr>
<td>I hear you knocking You can't catch me</td>
<td></td>
</tr>
<tr>
<td>I hear you knocking The stumble</td>
<td></td>
</tr>
<tr>
<td>I hear you knocking Sabre dance (Single version)</td>
<td></td>
</tr>
<tr>
<td>Datenbanken mit OpenOffice.org 3</td>
<td></td>
</tr>
<tr>
<td>Das Postfix-Buch</td>
<td></td>
</tr>
<tr>
<td>m August 25 Amsterdam</td>
<td></td>
</tr>
<tr>
<td>m August 25 Hier unten am Deich</td>
<td></td>
</tr>
<tr>
<td>m August 25 Köln-Ehrenfeld</td>
<td></td>
</tr>
<tr>
<td>m August 25 Bei Mir</td>
<td></td>
</tr>
<tr>
<td>m August 25 Gott sei Dank</td>
<td></td>
</tr>
</tbody>
</table>

Nach Bestätigung des rechten Verbundes sieht das Abfrageergebnis aus wie gewünscht. "Titel" und "Untertitel" werden komplett zusammen in einer Abfrage angezeigt. Natürlich kommen jetzt "Titel" wie in der vorhergehenden Verknüpfung mehrmals vor. Solange allerdings Suchtreffer nicht gezählt werden, könnte diese Abfrage im weiteren Verlauf als Grundlage für eine Suchfunktion die-

Abfragen: Eingabemöglichkeiten für Abfragen

239
nen. Siehe hierzu die Codeschnipsel in diesem Kapitel, im Kapitel «Makros» («420») und im Kapitel «DB-Aufgaben komplett» («357»).

**Vorsicht**

Werden mehrere Tabellen über einen rechten oder linken Verbund bearbeitet, so verarbeitet die grafische Benutzeroberfläche unter LO 3.3.4 den Verbundbefehl nicht korrekt. Dies führt dazu, dass die Abfrage mit einem SQL-Fehler abgebrochen wird. Seit der Version 3.5.3 ist dies nicht mehr der Fall!
Die Eingabe im SQL-Modus ist hiervon nicht berührt.

---

**Abfrageeigenschaften definieren**

Mit der Version 4.1 von LibreOffice ist es möglich, in dem Abfrageeditor zusätzliche Abfrageeigenschaften zu definieren.

Abbildung 40: Aufruf der Abfrageeigenschaften im Abfrageeditor (ab LO 4.1)

Neben dem Button zum Aufruf der Abfrageeigenschaften befindet sich noch ein Kombinationsfeld, mit dem die Anzahl der anzuzeigenden Datensätze reguliert werden kann, sowie ein Button «Eindeutige Werte». Diese Funktionen sind zusätzlich noch einmal in dem folgenden Dialog unterbrach:

Mit der Einstellung «Eindeutige Werte» wird beeinflusst, ob gleichlautende Datensätze in den Abfragen unterdrückt werden sollen.

<table>
<thead>
<tr>
<th>Vorname</th>
<th>Nachname</th>
<th>RückDatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa</td>
<td>Gerd</td>
<td></td>
</tr>
<tr>
<td>Lisa</td>
<td>Gerd</td>
<td></td>
</tr>
<tr>
<td>Lisa</td>
<td>Gerdl</td>
<td></td>
</tr>
<tr>
<td>Bert</td>
<td>Lederstumpf</td>
<td></td>
</tr>
<tr>
<td>Bert</td>
<td>Lederstumpf</td>
<td></td>
</tr>
<tr>
<td>Bert</td>
<td>Lederstumpf</td>
<td></td>
</tr>
<tr>
<td>Bert</td>
<td>Lederstumpf</td>
<td></td>
</tr>
<tr>
<td>Heinrich</td>
<td>Müller</td>
<td></td>
</tr>
<tr>
<td>Heinrich</td>
<td>Müller</td>
<td></td>
</tr>
<tr>
<td>Terence</td>
<td>Nobody</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vorname</th>
<th>Nachname</th>
<th>RückgDatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa</td>
<td>Gerd</td>
<td></td>
</tr>
<tr>
<td>Bert</td>
<td>Lederstumpf</td>
<td></td>
</tr>
<tr>
<td>Heinrich</td>
<td>Müller</td>
<td></td>
</tr>
<tr>
<td>Terence</td>
<td>Nobody</td>
<td></td>
</tr>
</tbody>
</table>

Wird «Eindeutige Werte» ausgewählt, so verschwinden die Datensätze mit gleichem Inhalt.

Die Abfrage sind dann so aus:

```sql
SELECT DISTINCT "Leser"."Vorname", "Leser"."Nachname", "Ausleihe"."Rueck_Datum"
FROM "Ausleihe", "Leser"
WHERE "Ausleihe"."Leser_ID" = "Leser"."ID" AND "Ausleihe"."Rueck_Datum" IS NULL
ORDER BY "Leser"."Nachname" ASC
```

Der ursprünglichen Abfrage

```
SELECT "Leser"."Vorname", "Leser"."Nachname" ...
```

wird ein «DISTINCT» hinzugefügt:

```
SELECT DISTINCT "Leser"."Vorname", "Leser"."Nachname" ...
```

Damit werden alle gleichlautenden Zeilen unterdrückt.


Mit der Einstellung «Limit» wird beeinflusst, wie viele Datensätze in der Abfrage angezeigt werden sollen. Es wird also nur eine begrenzte Zahl an Datensätzen wieder gegeben.

<table>
<thead>
<tr>
<th>ID</th>
<th>Titel</th>
<th>E_Jahr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Der kleine Hobbit</td>
<td>1972</td>
</tr>
<tr>
<td>1</td>
<td>Das sogenannte Böse</td>
<td>1972</td>
</tr>
<tr>
<td>2</td>
<td>Eine kurze Geschichte der Zeit</td>
<td>1983</td>
</tr>
<tr>
<td>3</td>
<td>Traditionelle und kritische Theorie</td>
<td>1970</td>
</tr>
<tr>
<td>4</td>
<td>Die neue deutsche Rechtschreibung</td>
<td>1996</td>
</tr>
<tr>
<td>5</td>
<td>I hear you knocking</td>
<td>1972</td>
</tr>
<tr>
<td>6</td>
<td>Einenbanken mit OpenOffice.org 3</td>
<td>2009</td>
</tr>
<tr>
<td>7</td>
<td>Das Postfix-Buch</td>
<td>2008</td>
</tr>
<tr>
<td>8</td>
<td>Im Augenblick</td>
<td>2009</td>
</tr>
</tbody>
</table>


Abfragen: Eingabemöglichkeiten für Abfragen 241
Nur die ersten fünf Datensätze werden angezeigt (ID 0 bis 4). Eine Sortierung wurde nicht vorge- 
wählt. Die Standardsortierung ist hier die nach dem Primärschlüssel, sofern nichts anderes festge-
legt wurde. Die Abfrage ist trotz der Begrenzung weiterhin editierbar. Dies unterscheidet die Ein-
gabe im grafischen Modus von der, die in früheren Versionen nur mit dem direkten SQL-Modus 
verfügbar ist.

Der ursprünglichen Abfrage wurde lediglich «LIMIT 5» hinzugefügt. Die entsprechende Größe des 
Limits kann beliebig festgelegt werden.

Vorsicht

Die Einstellung des Limits durch die grafische Benutzeroberfläche ist nicht abwärts-
kompatibel. In allen LO-Versionen vor der Version 4.1 konnte ein Limit nur im direk-
ten SQL-Modus eingegeben werden. Dort erforderte das Limit eine Sortierung 
(ORDER BY ...) oder eine Bedingung (WHERE ...).

Abfrageerweiterungen im SQL-Modus

Wird von der grafischen Eingabe über Ansicht → Design-Ansicht an-, ausschalten die Design-
Ansicht ausgeschaltet, so erscheint der SQL-Befehl, der bisher in der Design-Ansicht erstellt 
wurde. Für Neueinsteiger ist dies der beste Weg, die Standardabfragesprache für Datenbanken 
kennen zu lernen. Manchmal ist es auch der einzige Weg, eine Abfrage an die Datenbank abzuset-
zen, da die GUI die Abfrage nicht in den für die Datenbank notwendigen SQL-Befehl umwandeln 
kann.

SELECT *
FROM "Tabellenname"

Dies zeigt wirklich alles an, was in der Tabelle "Tabellenname" steht. Das «*» berücksichtigt sämtli-
che Felder der Tabelle.

SELECT *
FROM "Tabellenname"
WHERE "Feldname" = 'Karl'

Eine deutliche Einschränkung wurde gemacht. Jetzt werden nur noch die Datensätze angezeigt, 
die in dem Feld "Feldname" den Begriff 'Karl' stehen haben – aber wirklich nur den Begriff, nicht 
z. B. 'Karl Egon'.

Manchmal sind Abfragen in Base nicht über die GUI ausführbar, da bestimmte Kommandos nicht 
bekannt sind. Hier hilft es dann die Design-Ansicht zu verlassen und über Bearbeiten → SQL-
Komando direkt ausführen den direkten Weg zur Datenbank zu wählen. Diese Methode hat 
allerdings den Nachteil, dass in dem angezeigten Abfrageergebnis keine Eingaben mehr möglich 
sind. Siehe hierzu Eingabemöglichkeit in Abfragen.
Die direkte Ausführung ist auch über die grafische Benutzeroberfläche erreichbar. Wie in der Abbildung zu sehen muss aber auch hier die Designansicht ausgeschaltet sein. Entsprechende Abfrageanweisungen sind teilweise mit SQL gekennzeichnet.

Hier jetzt also die recht umfangreichen Möglichkeiten, an die Datenbank Fragen zu stellen und auf entsprechendes Ergebnis zu hoffen:

```sql
SELECT {{LIMIT <offset> <limit> | TOP <limit>}}[ALL | DISTINCT] 
{ <Select-Formulierung> | "Tabellenname".* | * } [, ...]
[INTO [CACHED | TEMP | TEXT] "neueTabelle"]
FROM "Tabellenliste"
[WHERE SQL-Expression]
[GROUP BY SQL-Expression [, ...]]
[HAVING SQL-Expression]
[ORDER BY Ordnungs-Expression [, ...]]
[LIMIT <limit> [OFFSET <offset>]]
```


Alle Eingaben in der Limitierung können nur direkt als Ganzzahl erfolgen. Es ist nicht möglich, die Eingaben durch eine Unterabfrage zu ersetzen, so dass z. B. fortlaufend die 5 letzten Datensätze einer Datenreihe angezeigt werden können.

ALL | DISTINCT

SELECT ALL ist die Standardeinstellung. Es werden alle Ergebnisse angezeigt, auf die die Bedingungen zutreffen. Beispiel:

```sql
SELECT ALL "Name" FROM "Tabellenname" gibt alle Namen an; kommt «Peter» dreifach und «Egon» vierfach in der Tabelle vor, so werden die Datensätze eben dreifach oder vierfach angezeigt. SELECT DISTINCT "Name" FROM "Tabellenname" sorgt hingegen dafür, dass alle Abfrageergebnisse mit gleichem Inhalt unterdrückt werden. Hier würden also «Peter» und «Egon» nur einmal erscheinen. DISTINCT bezieht sich dabei auf den ganzen Datensatz, der in der Abfrage erfasst wird. Wird z. B. auch der Nachname erfasst, so unterscheiden sich die
```
Datensätze mit «Peter Müller» und «Peter Maier». Sie werden also auch bei der Bedingung `DISTINCT` auf jeden Fall angezeigt.

**Select-Formulierung**

```sql
{ Expression | COUNT(*) |
{ COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP }
([ALL | DISTINCT]] Expression) } [[AS] "anzuzeigende Bezeichnung"]
```

Feldnamen, Berechnungen, Zählen der gesamten Datensätze – alles mögliche Eingaben.

**Hinweis**


Außerdem stehen in der Felddarstellung auch verschiedene Funktionen zur Verfügung. Mit Ausnahme von `COUNT(*)` (zählt alle Datensätze) berücksichtigen die verschiedenen Funktionen keine Felder, die `NULL` sind.

COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP

`COUNT("Name")` zählt alle Felder, die einen Namen enthalten.
`MIN("Name")` zeigt den ersten Namen im Alphabet. Das Ergebnis dieser Funktion ist so formatiert, wie es dem Feldinhalt entspricht. Text wird im Ergebnis Text, Ganzzahl zu Ganzzahl, Dezimalzahl zu Dezimalzahl usw.
`MAX("Name")` zeigt entsprechend den letzten Namen im Alphabet.
`SUM("Zahl")` kann nur Werte aus Zahlenfeldern addieren.

**Tipp**

Auch bei Zeitfeldern versagt die Funktion. Hier kann folgendes hilfreich sein:

```sql
SELECT (SUM( HOUR("Zeit") )*3600 +
        SUM( MINUTE("Zeit") ) )*60 + SUM( SECOND("Zeit") ) AS "Sekunden" FROM "Tabelle"
```

Die Summe wird von den Stunden, Minuten und gegebenenfalls Sekunden gebildet. Anschließend wird so erweitert, dass alles zusammen in einer Maßeinheit, hier in Sekunden, wiedergegeben wird. Es werden also weiter nur Zahlen mit der Summenformel addiert. Mit

```sql
SELECT ((SUM( HOUR("Zeit") )*3600 +
        SUM( MINUTE("Zeit") ) ) )*60 + SUM( SECOND("Zeit") ) ) /
        3600.0000 AS "Stunden" FROM "Tabelle"
```

AVG("Zahl") zeigt den Mittelwert der Inhalte einer Spalte. Auch diese Funktion beschränkt sich auf Zahlenfelder.

SOME("Ja Nein"), EVERY("Ja Nein"): SOME zeigt bei Ja/Nein Feldern (boolschen Feldern) die Version an, die nur einige Felder erfüllen. Da ein boolsches Feld die Werte 0 und 1 wiedergibt erfüllt nur einige (SOME) die Bedingung 1, aber jeder (EVERY) mindestens die Bedingung 0. Über eine gesamte Tabelle abgefragt wird bei SOME also immer «Ja» erscheinen, wenn mindestens 1 Datensatz mit «Ja» angekreuzt ist. EVERY wird so lange «Nein» ergeben, bis alle Datensätze mit «Ja» angekreuzt sind. Beispiel:

SELECT
"Klasse",
EVERY("Schwimmer")
FROM "Tabelle1"
GROUP BY "Klasse";

Die Klassen werden alle angezeigt. Erscheint irgendwo kein Kreuz für «Ja», so muss auf jeden Fall eine Betreuung für das Nichtschwimmerbecken im Schwimmunterricht dabei sein, denn es gibt mindestens eine Person in der Klasse, die nicht schwimmen kann.

VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP sind statistische Funktionen und greifen nur bei Ganzzahl- und Dezimalzahlfeldern. Alle Funktionen ergeben 0, wenn die Werte einer Gruppe alle gleich sind. Die statistischen Funktionen erlauben nicht die Einschränkung von DISTINCT. Sie rechnen also grundsätzlich über alle Werte, die die Abfrage beinhaltet. DISTINCT hingegen würde Datensätze mit gleichen Werten von der Anzeige ausschließen.


"Tabellenname".* | *[, ...]
Jedes anzeigende Feld kann mit seinem Feldnamen, getrennt durch Komma, angegeben werden. Werden Felder aus mehreren Tabellen in der Abfrage aufgeführt, so ist zusätzlich eine Kombination mit dem Tabellennamen notwendig: "Tabellenname"."Feldname".


[INTO [CACHED | TEMP | TEXT] "neueTabelle"]
Das Ergebnis dieser Abfrage soll direkt in eine neue Tabelle geschrieben werden. Die neue Tabelle wird hier benannt. Die Definition der Feldeigenschaften der neuen Tabelle wird dabei aus der Definition der Felder, die in der Abfrage erhalten sind, erstellt. Das Schreiben in eine Tabelle funktioniert nicht vom Abfrageeditor aus, da dieser nur anzeigbare Ergebnisse liefert. Hier muss die Eingabe über Extras → SQL erfolgen. Die Tabelle, die entsteht, ist anschließend erst einmal nicht editierbar, da ein Primärschlüsselfeld fehlt.


Um Sonderzeichen anzeigen zu können muss der Zeichensatz geändert werden. Dies kann entweder durch einen SQL-Befehl geschehen, der den Zeichensatz für die aktuelle Sitzung einstellt, oder aber durch eine dauerhafte Einstellung in der Datenbankdatei.
Die vorübergehende Änderung ist durch die Eingabe von
\texttt{SET PROPERTY "textdb.encoding" 'UTF-8';}
unter Extras → SQL möglich. Hier kann natürlich auch entsprechend 'ansi' gewählt werden.

Zur dauerhaften Änderung des Zeichensatzes muss die Datenbankdatei entpackt werden. Das in dieser Datei liegende Verzeichnis \texttt{database} enthält eine Datei \texttt{properties}. Diese muss um einen Eintrag erweitert werden: \texttt{textdb.encoding=ansi} für Windows-Systeme oder \texttt{textdb.encoding=UTF-8} für Linux-Systeme. Wie der Zugriff auf diese Einstellungen möglich ist wird im Anhang im Kapitel zur Datenbankreparatur beschrieben.

FROM <Tabellenliste>
"Tabellenname 1" [{CROSS | INNER | LEFT OUTER | RIGHT OUTER} JOIN
"Tabellenname 2" ON Expression] [, ...]

Die Tabellen, aus denen die Daten zusammengesucht werden sollen, werden in der Regel durch Komma getrennt aufgeführt. Die Beziehung der Tabellen zueinander wird anschließend mit dem Schlüsselwort \texttt{WHERE} definiert.

Werden die Tabellen durch einen JOIN statt durch ein Komma miteinander verbunden, so wird die Beziehung der Tabellen zueinander direkt nach der jeweils folgenden Tabellen mit dem Begriff \texttt{ON} beginnend definiert.

Ein einfacher JOIN bewirkt, dass nur die Datensätze angezeigt werden, auf die die Bedingung in beiden Tabellen zutrifft. Beispiel:

\begin{verbatim}
SELECT
  "Tabelle1"."Name",
  "Tabelle2"."Klasse"
FROM "Tabelle1",
  "Tabelle2"
WHERE "Tabelle1"."KlasseID" = "Tabelle2"."ID"
\end{verbatim}

entspricht von der Wirkung her

\begin{verbatim}
SELECT
  "Tabelle1"."Name",
  "Tabelle2"."Klasse"
FROM "Tabelle1"
JOIN "Tabelle2"
ON "Tabelle1"."KlasseID" = "Tabelle2"."ID"
\end{verbatim}

Es werden hier die Namen und die dazugehörigen Klassen aufgelistet. Fehlt zu einem Namen eine Klasse, so wird der Name nicht aufgelistet. Fehlen zu einer Klasse Namen, so werden diese ebenfalls nicht aufgelistet. Der Zusatz \texttt{INNER} bewirkt hierbei keine Änderung.

\begin{verbatim}
SELECT
  "Tabelle1"."Name",
  "Tabelle2"."Klasse"
FROM "Tabelle1"
LEFT JOIN "Tabelle2"
ON "Tabelle1"."KlasseID" = "Tabelle2"."ID"
\end{verbatim}

Bei dem Zusatz \texttt{LEFT} würden auf jeden Fall alle Inhalte von "Name" aus "Tabelle1" angezeigt – auch die, zu denen keine "Klasse" existiert. Beim Zusatz \texttt{RIGHT} hingegen würden alle Klassen angezeigt – auch die, zu denen kein Name existiert. Der Zusatz \texttt{OUTER} muss hier nicht unbedingt mit angegeben werden.

\begin{verbatim}
SELECT
  "Tabelle1"."Spieler1",
  "Tabelle2"."Spieler2"
FROM "Tabelle1" AS "Tabelle1"
\end{verbatim}
Beim **CROSS JOIN** müssen auf jeden Fall die Tabellen mit einem Aliasnamen versehen werden, wobei das Hinzufügen des Begriffes **AS** nicht unbedingt notwendig ist. Es werden einfach alle Datensätze aus der ersten Tabelle mit allen Datensätzen der zweiten Tabelle gekoppelt. So ergibt die obige Abfrage alle möglichen Paarungen aus der ersten Tabelle mit denen aus der zweiten Tabelle mit Ausnahme der Paarungen, bei denen es sich um gleiche Spieler handelt. Die Bedingung darf beim **CROSS JOIN** allerdings keine Verknüpfung der Tabellen mit **ON** enthalten. Stattdessen können unter **WHERE** Bedingungen eingegeben werden. Würde hier die Bedingung genauso formuliert wie beim einfachen **JOIN**, so wäre das Ergebnis gleich:

```
SELECT
    "Tabelle1"."Name",
    "Tabelle2"."Klasse"
FROM "Tabelle1"
JOIN "Tabelle2"
ON "Tabelle1"."KlasseID" = "Tabelle2"."ID"
liefert das gleiche Ergebnis wie
```

```
SELECT
    "Tabelle1"."Name",
    "Tabelle2"."Klasse"
FROM "Tabelle1" AS "Tabelle1"
CROSS JOIN "Tabelle2" AS "Tabelle2"
WHERE "Tabelle1"."KlasseID" = "Tabelle2"."ID"
```

**[WHERE SQL-Expression]**

Die Standardeinleitung, um Bedingungen für eine genauere Filterung der Daten zu formulieren. Hier werden in der Regel auch die Beziehungen der Tabellen zueinander definiert, sofern die Tabellen nicht mit **JOIN** verbunden sind.

**[GROUP BY SQL-Expression [ , ...]]**

Wenn Felder mit einer bestimmten Funktion bearbeitet werden (z. B. **COUNT**, **SUM** ...), so sind alle Felder, die nicht mit einer Funktion bearbeitet werden aber angezeigt werden sollen, mit **GROUP BY** zu einer Gruppe zusammen zu fassen.
Beispiel:

```
SELECT
    "Name",
    SUM("Einnahme" - "Ausgabe") AS "Saldo"
FROM "Tabelle1"
GROUP BY "Name";
```

Datensätze mit gleichen Namen werden jetzt aufsummiert. Im Ergebnis wird jeweils **Einnahme** – **Ausgabe** ermittelt und darüber die Summe, die jede Person erzielt hat, aufgelistet. Das Feld wird unter dem Namen **Saldo** dargestellt.

**[HAVING SQL-Expression]**


Beispiel:

```
SELECT
    "Name",
    "Laufzeit"
```
Es werden alle Namen und Laufzeiten aufgelistet, bei denen die Laufzeit weniger als 40 Minuten beträgt.

**[SQL Expression]**

SQL-Ausdrücke werden nach dem folgenden Schema miteinander verbunden:

[NOT] Bedingung [{ OR | AND } Bedingung]

Beispiel:

```sql
SELECT *
FROM "Tabellenname"
WHERE NOT "Rückgabedatum" IS NULL
AND "LeserID" = 2;
```

Aus der Tabelle werden die Datensätze ausgelesen, bei denen ein "Rückgabedatum" eingetragen wurde und die "LeserID" 2 lautet. Das würde in der Praxis bedeuten, dass alle Medien, die eine bestimmte Person ausgeliehen und wieder zurückgegeben hat, damit ermittelt werden könnten. Die Bedingungen sind nur durch **AND** miteinander verbunden. Das **NOT** bezieht sich rein auf die erste Bedingung.

```sql
SELECT *
FROM "Tabellenname"
WHERE NOT ("Rückgabedatum" IS NULL AND "LeserID" = 2);
```

Wird eine Klammer um die Bedingung gesetzt und **NOT** steht außerhalb der Klammer, so werden genau die Datensätze angezeigt, die die in den Klammern stehenden Bedingungen komplett nicht erfüllen. Das wären alle Datensätze mit Ausnahme derer, die "LeserID" mit der Nummer 2 noch nicht zurückgegeben hat.

**[SQL Expression]: Bedingungen**

{ Wert || Wert]

Ein Wert kann einzeln oder mit mehreren Werten zusammen über zwei senkrechte Striche || kombiniert werden. Dies gilt dann natürlich auch für Feldinhalte.

```sql
SELECT "Nachname" || ', ' || "Vorname" AS "Name"
FROM "Tabellenname"
```

Die Inhalte aus den Feldern "Nachname" und "Vorname" werden in einem Feld "Name" gemeinsam angezeigt. Dabei wird ein Komma und eine Leertaste zwischen "Nachname" und "Vorname" eingefügt.

| Wert { = | < | <= | > | >= | <> | != } Wert

Die Zeichen entsprechend den aus der Mathematik bekannten Operatoren:

{ Gleich | Kleiner als | Kleiner oder gleich | Größer als | Größer oder gleich | nicht gleich | nicht gleich }

| Wert IS [NOT] NULL

Das entsprechende Feld hat keinen Inhalt, ist auch nicht beschrieben worden. Dies kann in der GUI nicht unbedingt beurteilt werden, denn ein leeres Textfeld bedeutet noch nicht, dass das Feld völlig ohne Inhalt ist. Die Standardeinstellung von Base ist aber so, dass leere Felder in der Datenbank auf **NULL** gesetzt werden.
| EXISTS(Abfraageaussage)
Beispiel:
SELECT
  "Name"
FROM "Tabelle1"
WHERE EXISTS
  (SELECT
    "Vorname"
  FROM "Tabelle2"
    WHERE "Tabelle2"."Vorname" = "Tabelle1"."Name")
Es werden die Namen aus Tabelle1 aufgeführt, die als Vornamen in Tabelle2 verzeichnet sind.
| Wert BETWEEN Wert AND Wert
BETWEEN Wert1 AND Wert2 gibt alle Werte ab Wert1 bis einschließlich Wert2 wieder. Wer-
den hier Buchstaben als Werte eingesetzt, so wird die alphabetische Sortierung angenommen,
wobei Kleinbuchstaben und Großbuchstaben die gleichen Werte haben.
SELECT
  "Name"
FROM "Tabellenname"
WHERE "Name" BETWEEN 'A' AND 'E';
Diese Abfrage gibt alle Namen wieder, die mit A, B, C und D beginnen (ggf. auch mit entspre-
chendem Kleinbuchstaben). Da als unterer Begrenzung E gesetzt wurde sind alle Namen mit E
nicht mehr in der Auswahl enthalten. Der Buchstabe E würde in einer Sortierung ganz am
Anfang der Namen mit E stehen.
| Wert [NOT] IN ( {Wert [, ...] | Abfrageaussage } )
Hier wird entweder eine Liste von Werten oder eine Abfrage eingesetzt. Die Bedingung ist
erfüllt, wenn der Wert in der Werteliste bzw. im Abfrageergebnis enthalten ist.
| Wert [NOT] LIKE Wert [ESCAPE] Wert }
Der LIKE-Operator ist derjenige, der in vielen einfachen Suchfunktionen benötigt wird. Die
Angabe der Werte erfolgt hier nach folgendem Muster:
'%' steht für beliebig viele, ggf. auch 0 Zeichen,
'_' ersetzt genau ein Zeichen.
Um nach '%' oder '_' selbst zu suchen müssen die Zeichen direkt nach einem zweiten Zeichen
auftauchen, das nach ESCAPE definiert wird.
SELECT
  "Name"
FROM "Tabellenname"
WHERE "Name" LIKE '\_\%' ESCAPE '\'
Diese Abfrage zeigt alle Namen auf, die mit einem Unterstrich beginnen. Als ESCAPE-Zeichen
ist hier '\' definiert worden.

[SQL Expression]: Werte
[+ | -] { Ausdruck [{ + | - | * | / | || }] Ausdruck]
Vorzeichen vor den Werten sind möglich. Die Addition, Subtraktion, Multiplikation, Division und
Verkettung von Ausdrücken ist erlaubt. Beispiel für eine Verkettung:
SELECT
  "Nachname"||', '||"Vorname"
FROM "Tabelle"
Auf diese Art und Weise werden Datensätze in der Abfrage als ein Feld ausgegeben, in der "Nachname, Vorname" steht. Die Verkettung erlaubt also jeden der weiter unten genannten Ausdrücke.

| ( Bedingung )
Siehe hierzu den vorhergehenden Abschnitt
| Funktion ( [Parameter] [, ...] )
Siehe hierzu im Anhang das Kapitel "Eingebaute Funktionen und abgespeicherte Prozeduren".

Die folgenden Abfragen werden auch als Unterabfragen (Subselects) bezeichnet.

| Abfrageaussage, die nur genau einen Wert ergibt
Da ein Datensatz für jedes Feld nur einen Wert darstellen kann, kann auch nur die Abfrage komplett angezeigt werden, die genau einen Wert ergibt.

| {ANY|ALL} (Abfrageaussage, die den Inhalt einer ganzen Spalte wiedergibt)
Manchmal gibt es Bedingungen, bei denen ein Ausdruck mit einer ganzen Gruppe von Werten verglichen wird. Zusammen mit ANY bedeutet das, dass der Ausdruck mindestens einmal in der Gruppe vorkommen muss. Dies ließe sich auch mit der IN-Bedingung beschreiben. = ANY ergibt das gleiche Ergebnis wie IN.

Zusammen mit ALL bedeutet dies, dass alle Werte der Gruppe dem einen Ausdruck entsprechen müssen.

**[SQL Expression]: Ausdrücke**

```
{ 'Text' | Ganzzahl | Fließkommazahl
| ["Tabelle".]."Feld" | TRUE | FALSE | NULL }
```


**UNION [ALL | DISTINCT] Abfrageaussage SQL**

```sql
SELECT "Vorname"
FROM "Tabelle1"
UNION DISTINCT
SELECT
```

Abfragen: Eingabemöglichkeiten für Abfragen
"Vorname"
FROM "Tabelle2";


Mit Hilfe dieser Abfragetechnik ist es auch möglich, Werte einer Datenzeile z.B. für eine Liste untereinander in einer Spalte anzuordnen. Angenommen es existiert eine Tabelle "Ware", in der der "Verkaufspreis" sowie ein "Rabattpreis_1" und ein "Rabattpreis_2" enthalten sind. Daraus soll der Inhalt für ein Kombinationsfeld erstellt werden, das genau diese Preise untereinander auflistet:

SELECT
"Verkaufspreis"
FROM "Ware" WHERE "Ware_ID" = 1
UNION
"Rabattpreis_1"
FROM "Ware" WHERE "Ware_ID" = 1
UNION
"Rabattpreis_2"
FROM "Ware" WHERE "Ware_ID" = 1;

Der Primärschlüsselwert für die Ware müsste hier natürlich entsprechend über ein Makro gesetzt werden, das dem Kombinationsfeld je nach Ware einen entsprechenden Inhalt zuweist.

MINUS [DISTINCT] | EXCEPT [DISTINCT] Abfrageaussage
SELECT
"Vorname"
FROM "Tabelle1"
EXCEPT
SELECT
"Vorname"
FROM "Tabelle2";


INTERSECT [DISTINCT] Abfrageaussage
SELECT
"Vorname"
FROM "Tabelle1"
INTERSECT
SELECT
"Vorname"
FROM "Tabelle2";

Hier werden nur die Vornamen angezeigt, die in beiden Tabellen vorhanden sind. Die Sortierung ist wieder alphabetisch. Dies funktioniert zur Zeit nur, wenn das SQL-Kommando direkt ausgeführt wird.

[ORDER BY Ordnungs-Expression [, ...]]
Hier können Feldnamen, die Nummer der Spalte (beginnend mit 1 von links), ein Alias (formuliert z.B. mit AS) oder eine Wert zusammensetzung für Abfragen (siehe [SQL Expression]: Werte) angege-

Abfragen: Eingabemöglichkeiten für Abfragen 251
ben werden. Die Sortierung erfolgt in der Regel aufsteigend (ASC). Nur wenn die Sortierung absteigend erfolgen soll muss DESC angegeben werden.

SELECT
  "Vorname",
  "Nachname" AS "Name"
FROM "Tabelle1"
ORDER BY "Nachname";

ist identisch mit

SELECT
  "Vorname",
  "Nachname" AS "Name"
FROM "Tabelle1"
ORDER BY "Nachname" ASC;

ist identisch mit

SELECT
  "Vorname",
  "Nachname" AS "Name"
FROM "Tabelle1"
ORDER BY "Name";

Verwendung eines Alias in Abfragen

Durch Abfragen können auch Felder in einer anderen Bezeichnung wiedergegeben werden.

SELECT
  "Vorname",
  "Nachname" AS "Name"
FROM "Tabelle1"

Dem Feld Nachname wird in der Anzeige die Bezeichnung Name zugeordnet.

Wird eine Abfrage aus zwei Tabellen erstellt, so steht vor den jeweiligen Feldbezeichnungen der Name der Tabelle:

SELECT
  "Tabelle1"."Vorname",
  "Tabelle1"."Nachname" AS "Name",
  "Tabelle2"."Klasse"
FROM "Tabelle1",
  "Tabelle2"
WHERE "Tabelle1"."Klasse_ID" = "Tabelle2"."ID"

Auch den Tabellennamen kann ein Aliasname zugeordnet werden, der allerdings in der Tabellenansicht nicht weiter erscheint. Wird so ein Alias zugeordnet, so müssen sämtliche Tabellenbezeichnungen in der Abfrage entsprechend ausgetauscht werden:

SELECT
  "a"."Vorname",
  "a"."Nachname" AS "Name",
  "b"."Klasse"
FROM "Tabelle1" AS "a",
  "Tabelle2" AS "b"
WHERE "a"."Klasse_ID" = "b"."ID"

Die Zuweisung eines Aliasnamens kann auch verkürzt ohne den Zuweisungsbegriff AS erfolgen:
SELECT "a"."Vorname", 
"a"."Nachname" "Name", 
"b"."Klasse"
FROM "Tabelle1" "a",
"Tabelle2" "b"
WHERE "a"."Klasse_ID" = "b"."ID"

Dies erschwert allerdings die eindeutige Lesbarkeit des Codes. Daher sollte nur in Ausnahmefällen
auf die Verkürzung zurückgegriffen werden.

Hinweis

Leider ist in der grafischen Benutzeroberfläche des Abfrageeditors seit einiger Zeit
der Code so geändert worden, dass Alias-Bezeichnungen grundsätzlich ohne den
Zusatz AS erstellt werden. Das liegt daran, dass bei der Verwendung von externen
Datenbanken, bei denen diese Art der Alias-Zuweisung nicht vorgesehen war, der
Zusatz zu Fehlermeldungen führte.

Wenn der Code, der zum Bearbeiten geöffnet wird, nicht direkt in der SQL-Ansicht,
sondern über die GUI geöffnet wird, verlieren bereits erstellte Alias-Bezeichnungen
hierbei die Zuweisung AS. Wer Wert auf diese Bezeichnung legt sollte also Abfragen
immer über In SQL-Ansicht bearbeiten öffnen.

Über den Aliasnamen kann auch eine Tabelle mit entsprechenden Filterungen mehrmals innerhalb
einer Abfrage genutzt werden:

SELECT "Kasse"."Betrag", "Kasse"."Datum",
"a"."Betrag" AS "Haben",
"b"."Betrag" AS "Soll"
FROM "Kasse"
LEFT JOIN "Kasse" AS "a"
ON "Kasse"."ID" = "a"."ID" AND "a"."Betrag" >= 0
LEFT JOIN "Kasse" AS "b"
ON "Kasse"."ID" = "b"."ID" AND "b"."Betrag" < 0

Abfragen für die Erstellung von Listenfeldern

In Listenfeldern wird ein Wert angezeigt, der nicht an die den Formularen zugrundeliegenden
Tabellen weitergegeben wird. Sie werden schließlich eingesetzt, um statt eines Fremdschlüssels
zu sehen, welchen Wert der Nutzer damit verbindet. Der Wert, der schließlich in Formularen abgespeichert wird, darf bei den Listenfeldern nicht an der ersten Position stehen.

SELECT "Vorname",
"ID"
FROM "Tabelle1";

Diese Abfrage würde alle Vornamen anzeigen und den Primärschlüssel "ID" an die dem Formular
zugrundeliegende Tabelle weitergeben. So ist das natürlich noch nicht optimal. Die Vornamen erscheinen unsortiert; bei gleichen Vornamen ist außerdem unbekannt, um welche Person es sich denn handelt.

SELECT "Vorname"||' '||"Nachname",
"ID"
FROM "Tabelle1"
ORDER BY "Vorname"||' '||"Nachname";

Jetzt erscheint der Vorname, getrennt von dem Nachnamen, durch ein Leerzeichen. Die Namen
werden unterscheidbarer und sind sortiert. Die Sortierung erfolgt der Logik, dass natürlich nach
Den vorne stehenden Buchstaben zuerst sortiert wird, also Vornamen zuerst, dann Nachnamen. Andere Sortierreihenfolgen als nach der Anzeige des Feldes würden erst einmal verwirren.

```sql
SELECT
    "Nachname"||', '||"Vorname",
    "ID"
FROM "Tabelle1"
ORDER BY "Nachname"||', '||"Vorname";
```

Dies würde jetzt zu der entsprechenden Sortierung führen, die eher unserer Gewohnheit entspricht. Familienmitglieder erscheinen zusammen untereinander, bei gleichen Nachnamen und unterschiedlichen Familien wird allerdings noch durcheinander gewürfelt. Um dies zu unterscheiden müsste eine Gruppenzuordnung in der Tabelle gemacht werden.

Letztes Problem ist noch, dass eventuell auftauchende Personen mit gleichem Nachnamen und gleichem Vornamen nicht unterscheidbar sind. Variante 1 wäre, einfach einen Namenszusatz zu entwerfen. Aber wie sieht das denn aus, wenn ein Anschreiben mit Herr «Müller II» erstellt wird?

```sql
SELECT
    "Nachname"||', '||"Vorname"||' - ID:'||"ID",
    "ID"
FROM "Tabelle1"
ORDER BY "Nachname"||', '||"Vorname"||"ID";
```

Hier wird auf jeden Fall jeder Datensatz unterscheidbar. Angezeigt wird «Nachname, Vorname – ID:IDWert».

Falls es einmal passieren sollte, dass Felder für die Vornamen leer (NULL) sind, so wird so ein Datensatz in den Listenfeldern natürlich erst einmal nicht angezeigt. Leere Felder, verbunden mit anderen Feldern, sind ebenfalls NULL. Hier hilft dann eine entsprechend eingefügte Bedingung:

```sql
SELECT
    "Nachname"||IFNULL('', '||"Vorname",""),
    "ID"
FROM "Tabelle1"
ORDER BY "Nachname"||IFNULL('', '||"Vorname","");
```

Damit wird das Komma und der anschließende Vorname durch einen leeren Text ersetzt, wenn das Feld "Vorname" NULL ist.

Im Formular Ausleihe wurde ein Listenfeld erstellt, das lediglich die Medien darstellen sollte, die noch nicht entliehen wurden. Dies wird über die folgende SQL-Formulierung möglich:

```sql
SELECT
    "Titel" || ' - Nr. ' || "ID",
    "ID"
FROM "Medien"
WHERE "ID"
    NOT IN
        (SELECT
            "Medien_ID"
        FROM "Ausleihe"
        WHERE "Rueck_Datum" IS NULL)
ORDER BY "Titel" || ' - Nr. ' || "ID" ASC
```

Dieses Listenfeld muss allerdings immer dann aktualisiert werden, wenn eine Ausleihe eines darin verzeichneten Mediums erfolgt ist.

Das folgende Listenfeld stellt die Inhalte mehrerer Felder in Tabellenform dar, so dass zusammengehörige Elemente direkt untereinander erscheinen:

---

Abfragen: Abfragen für die Erstellung von Listenfeldern 254
Damit so eine Darstellung gelingt muss zuerst einmal eine entsprechende nicht proportionale Schrift ausgewählt werden. Courier oder alle Mono-Schriftarten wie z.B. Liberation Mono können hier genutzt werden. Die Darstellung in tabellarischer Form gelingt über den SQL-Code:

```sql
SELECT 
  LEFT("Ware"||SPACE(25),25) || ' - ' || 
  RIGHT(SPACE(8)||"Preis",8) || ' €', 
  "ID"
FROM "Waren"
ORDER BY ("Ware" || ' - ' || "Preis" || ' €') ASC
```

An den Inhalt des Feldes "Ware" werden entsprechend viele Leerzeichen angehängt, so dass "Ware" zusammen mit den Leerzeichen eine Mindestlänge von 25 Zeichen hat. Anschließend werden die ersten 25 Buchstaben dargestellt, die überflüssigen Leerzeichen also abgeschnitten.

Komplizierter wird es, wenn in dem Inhalt des Listenfeldes auch nicht druckbare Zeichen wie z.B. Zeilenumbrüche enthalten sind. Dann muss der Code entsprechend angepasst werden:

```sql
SELECT 
  LEFT(REPLACE("Ware",CHAR(10),' ')||SPACE(25), 25) || ' - ' || ...
```

Damit wird ein Zeilenvorschub in Linux durch ein Leerzeichen ersetzt. In Windows muss zusätzlich noch der Wagenrücklauf (CHAR(13)) entfernt werden.

Die Anzahl der notwendigen Leerzeichen kann übrigens auch per Abfrage ermittelt werden. So wird vermieden, dass doch einmal ein Wert aus "Ware" in seiner Länge beschnitten wird.

```sql
SELECT 
  LEFT("Ware"||SPACE((SELECT MAX(LENGTH("Ware")) FROM "Waren")),(SELECT MAX(LENGTH("Ware")) FROM "Waren"))|| ' - ' || 
  RIGHT('    '||"Preis",8) || ' €', 
  "ID"
FROM "Waren"
ORDER BY ("Ware" || ' - ' || "Preis" || ' €') ASC
```

Da der Preis rechtsbündig dargestellt werden soll wird hier vor dem Preis mit Leerzeichen aufgefüllt und entsprechend maximal 8 Zeichen von rechts aus dargestellt. Die angestrebte Darstellung reicht also für alle Preise bis 99999,99 € aus.

Soll auch noch der Punkt aus der SQL-Darstellung durch ein Komma ersetzt werden, so ist der SQL-Code entsprechend zu ergänzen. Dies wäre über

```sql
REPLACE(RIGHT('    '||"Preis",8),'.','',')
```

möglich.
Abfragen als Grundlage von Zusatzinformationen in Formularen

Will man zusätzliche Informationen im Formular im Auge haben, die so gar nicht sichtbar wären, so bieten sich verschiedene Abfragemöglichkeiten an. Die einfachste ist, mit gesonderten Abfragen diese Informationen zu ermitteln und in andere Formulare einzustellen. Der Nachteil dieser Variante kann sein, dass sich Datenänderungen auf das Abfrageergebnis auswirken würden, aber leider die Änderungen nicht automatisch angezeigt werden.

Hier ein Beispiel aus dem Bereich der Warenwirtschaft für eine einfache Kasse:

Die Tabelle für eine Kasse enthält Anzahl und Fremdschlüssel von Waren, außerdem eine Rechnungsnummer. Besonders wenig Informationen erhält der Nutzer an der Supermarktkasse, wenn keine zusätzlichen Abfrageergebnisse auf den Kassenzettel gedruckt werden. Schließlich werden die Waren nur über den Barcode eingelesen. Ohne Abfrage ist im Formular nur zusehen:

<table>
<thead>
<tr>
<th>Anzahl</th>
<th>Barcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
</tr>
</tbody>
</table>

usw.

Was sich hinter den Nummern versteckt kann nicht mit einem Listenfeld sichtbar gemacht werden, da ja der Fremdschlüssel über den Barcode direkt eingegeben wird. So lässt sich auch nicht über das Listenfeld neben der Ware zumindest der Einzelpreis ersehen.

Hier hilft eine Abfrage:

```
SELECT
  "Kasse"."RechnungID",
  "Kasse"."Anzahl",
  "Ware"."Ware",
  "Ware"."Einzelpreis",
  "Kasse"."Anzahl"*"Ware"."Einzelpreis" AS "Gesamtpreis"
FROM "Kasse", "Ware"
WHERE "Ware"."ID" = "Kasse"."WareID";
```

Jetzt ist zumindest schon einmal nach der Eingabe die Information da, wie viel denn für 3 * Ware'17' zu bezahlen ist. Außerdem werden nur die Informationen durch das Formular zu filtern sein, die mit der entsprechenden "RechnungID" zusammenhängen. Was auf jeden Fall noch fehlt ist das, was denn der Kunde nun bezahlen muss:

```
SELECT
  "Kasse"."RechnungID",
  SUM("Kasse"."Anzahl"*"Ware"."Einzelpreis") AS "Summe"
FROM "Kasse", "Ware"
WHERE "Ware"."ID" = "Kasse"."WareID"
GROUP BY "Kasse"."RechnungID";
```

Für das Formular liefert diese Abfrage nur eine Zahl, da über das Formular natürlich wieder die "RechnungID" gefiltert wird, so dass ein Kunde nicht gleichzeitig sieht, was andere vor ihm eventuell gezahlt haben.
Sollen in einem Formular Datumswerte dargestellt werden, die in Abhängigkeit von einem anderen Datum definiert sind (z.B.: Eine Entleihzeit für ein Medium beträgt 21 Tage – wann muss das Medium zurückgegeben werden?), dann ist dies mit Standardfunktionen der HSQLDB nicht zu machen. Es fehlt eine Funktion wie `DATEADD`.

Die Abfrage

```sql
SELECT "Datum", DATEDIFF('dd','1899-12-30','Datum')+21 AS "RueckDatum" FROM "Tabelle"
```


Allerdings handelt es sich bei dem ermittelten Wert um eine Zahl, nicht um ein Datum, das anschließend in z.B. einer Abfrage weiter genutzt werden kann.

In einer Abfrage lässt sich die ermittelte Zahl schlecht nutzen, da die Formatierung von Abfragen nicht gespeichert wird. Dafür müsste eine Ansicht erstellt werden.

---

**Eingabemöglichkeit in Abfragen**

Um Eingaben in Abfragen zu tätigen muss der Primärschlüssel für die jeweils zugrundeliegende Tabelle in der Abfrage enthalten sein.

Abfrage-Eingabemöglichkeit:

- **Editierbar:** Der Button "Daten bearbeiten" ist aktiv.
- **Editierbar:** Ein neuer Datensatz kann eingefügt werden.
- **Editierbar:** Der Primärschlüssel der Tabelle in der Abfrage enthalten ist.
- **Editierbar:** nur eine Tabelle (Ausleihe) mit allen Feldern (Ausleihe.*) abgefragt wird.

Bei der Ausleihe von Medien ist es z. B. nicht sinnvoll, für einen Leser auch die Medien noch anzeigen zu lassen, die längst zurückgegeben wurden.

---

*Abfragen: Eingabemöglichkeit in Abfragen* 257
SELECT 
  "ID",
  "LeserID",
  "MedienID",
  "Ausleihdatum"
FROM "Ausleihe"
WHERE "Rückgabedatum" IS NULL;

So lässt sich im Formular innerhalb eines Tabellenkontrollfeldes all das anzeigen, was ein bestimmter Leser zur Zeit entliehen hat. Auch hier ist die Abfrage über entsprechende Formularkonstruktion (Leser im Hauptformular, Abfrage im Unterformular) zu filtern, so dass nur die tatsächlich entliehenen Medien angezeigt werden. Die Abfrage ist zur Eingabe geeignet, da der Primärschlüssel in der Abfrage enthalten ist.

Die Abfrage wird dann nicht mehr editierbar, wenn sie aus mehreren Tabellen besteht und die Tabellen über einen Alias-Namen angesprochen werden. Dabei ist es unerheblich, ob die Primärschlüssel in der Abfrage enthalten sind.

SELECT 
  "Medien"."ID",
  "Medien"."Titel",
  "Medien"."Kategorie_ID",
  "Kategorie"."ID" AS "katID"
FROM "Medien",
Abfragen

Abfragen: Eingabemöglichkeit in Abfragen 258
Diese Abfrage bleibt editierbar, da **beide Primärschlüssel enthalten** sind und auf die Tabellen nicht mit einem Alias zugegriffen wird. Auch bei einer Abfrage, die mehrere Tabellen miteinander verknüpft, müssen lediglich alle Primärschlüssel vorhanden sein.

Es ist allerdings nicht möglich, in einer Abfrage, die auf mehreren Tabellen beruht, das Fremd­schlüsselfeld der einen Tabelle, das sich auf die andere Tabelle bezieht, zu ändern. Im angezeig­ten Datensatz wurde versucht, die Kategorie für den Titel «Der kleine Hobbit» umzustellen. Das Feld "Kategorie_ID" wurde von '0' auf '2' geändert. Die Änderung schien vollzogen, die neue Katego­rie erschien auch. Das Speichern war dann aber nicht möglich.

Allerdings ist es möglich, den Inhalt von "Kategorie" selbst zu bearbeiten, also z.B. 'Fantasy' zum Begriff 'Fantasie' zu ändern. Das ändert dann allerdings diesen Begriff für die Kategorie, betrifft also alle damit verbundenen Datensätze.

```sql
SELECT
  "a"."ID",
  "a"."Titel",
  "Kategorie"."Kategorie",
  "Kategorie"."ID" AS "katID"
FROM "Medien" AS "a",
"Kategorie"
WHERE "a"."Kategorie_ID" = "Kategorie"."ID";
```

In dieser Abfrage wird auf die Tabelle "Medien" mit einem **Alias** zugegriffen. Sie wird jetzt **nicht editierbar** sein.
In dem obigen Beispiel ist das leicht vermeidbar. Wenn aber eine **Korrelierte Unterabfrage** gestellt wird, so muss mit einem Tabellenalias gearbeitet werden. So eine **Abfrage mit Alias** kann nur **dann editierbar** bleiben, wenn sie **lediglich eine Tabelle in der Hauptabfrage** enthält.

In der Design-Ansicht erscheint hier lediglich eine Tabelle. Die Tabelle "Medien" ist mit einem Alias versehen, damit auf die Inhalte des Feldes "Kategorie_ID" mit der korrelierenden Unterabfrage zugegriffen werden kann.

In so einer Abfrage ist es jetzt möglich, das Fremdschlüsselfeld "Kategorie_ID" auf eine andere Kategorie umzustellen. In dem obigen Beispiel wurde das Feld "Kategorie_ID" von '0' auf '2' geändert. Dem "Titel" 'Der kleine Hobbit' wurde so die "Kategorie" 'Liedermacher' zugeordnet.

Abfragen: Eingabemöglichkeit in Abfragen

260

Verwendung von Parametern in Abfragen

Wird viel mit gleichen Abfragen, aber eventuell unterschiedlichen Werten als Voraussetzung zu diesen Abfragen gearbeitet, so sind Parameterabfragen möglich. Vom Prinzip her funktionieren Parameterabfragen erst einmal genauso wie Abfragen, die in Unterformularen abgelegt werden:

```
SELECT
    "ID",
    "LeserID",
    "MedienID",
    "Ausleihdatum"
FROM "Ausleihe"
WHERE "Rückgabedatum" IS NULL
AND "LeserID"=2;
```

Diese Abfrage zeigt nur die entliehenen Medien des Lesers mit der Nummer 2 an.

```
SELECT
    "ID",
    "LeserID",
    "MedienID",
    "Ausleihdatum"
FROM "Ausleihe"
WHERE "Rückgabedatum" IS NULL
AND "LeserID" = :Lesernummer;
```

Jetzt erscheint beim Aufrufen der Abfrage ein Eingabefeld. Es fordert zur Eingabe einer Lesernummer auf. Wird hier ein Wert eingegeben, so werden die zur Zeit entliehenen Medien dieses Lesers angezeigt.

```
SELECT
    "Ausleihe"."ID",
    "Leser"."Nachname"||', '||"Leser"."Vorname",
    "Ausleihe"."MedienID",
    "Ausleihe"."Ausleihdatum"
FROM "Ausleihe", "Leser"
WHERE "Ausleihe"."Rückgabedatum" IS NULL
AND "Leser"."ID" = "Ausleihe"."LeserID"
AND "Leser"."Nachname" LIKE '%' || :Lesernummer || '%'
ORDER BY "Leser"."Nachname"||', '||"Leser"."Vorname" ASC;
```


Wird

```
"Leser"."Nachname" LIKE '%' || :Lesernummer || '%'
```

durch

```
LOWER("Leser"."Nachname") LIKE '%' || LOWER( :Lesernummer ) || '%'
```
ersetzt, so ist es auch noch egal, ob die Namen groß oder klein geschrieben sind.
Wird der **Parameter** in der obigen Abfrage **leer** gelassen, so werden bis LO 4.4 alle **Leser** angezeigt, da ein leeres Parameterfeld erst ab der Version LO 4.4 auch tatsächlich nicht als leerer Text, sondern als **NULL** weiter gegeben wird. Soll dies vermieden werden, so muss etwas in die Trickkiste gegriffen werden:

```
LOWER ("Leser"."Nachname") LIKE '%' || IFNULL( NULLIF ( LOWER (:Lesername), '' ), '§§' ) || '%'
```

Das leere Parameterfeld gibt an die Abfrage einen leeren String, aber nicht NULL weiter. Deshalb muss erst einmal dem leeren Parameterfeld die Eigenschaft NULL mit **NULLIF** zugewiesen werden. Anschließend wird für den Fall, dass eben die Parametereingabe jetzt NULL ergibt, dieser Eingabe ein Wert zugewiesen, der in der Regel in keinem der Datensätze vorkommt. In dem obigen Beispiel ist das '§§'. Dieser Wert wird mit der Abfrage entsprechend auch nicht gefunden.

Ab der Version **LO 4.4** muss diese Abfragetechnik etwas angepasst werden:

```
LOWER ("Leser"."Nachname") LIKE '%' || LOWER (:Lesername) || '%'
```

ergibt zwangsläufig bei einer fehlenden Eingabe für die gesamte Kombination '% || LOWER (:Lesername) || %' den Wert NULL. Dagegen hilft das Hinzufügen einer weiteren Bedingung, dass bei einem leeren Feld tatsächlich alle Werte aufgezeigt werden:

```
(LOWER ("Leser"."Nachname") LIKE '%' || LOWER (:Lesername) || '%' OR :Lesername IS NULL)
```

Dies sollte in Klammern gesetzt werden. So wird entweder ein Name ausgesucht oder, bei learem Feld, also **NULL** ab LO 4.4, die zweite Bedingung erfüllt.

Ein Parameter kann bei Formularen auch von einem Hauptformular an ein Unterformular weitergegeben werden. Es kann allerdings passieren, dass Parameterabfragen in Unterformularen nicht aktualisiert werden, wenn Daten geändert oder neu eingegeben werden.


### Unterabfragen

Unterabfragen, die in Felder eingebaut werden, dürfen immer nur genau einen Datensatz wiedergeben. Das Feld kann schließlich auch nur einen Wert wiedergeben.

```
SELECT
    "ID",
    "Einnahme",
    "Ausgabe",
    ( SELECT
        SUM( "Einnahme" ) - SUM( "Ausgabe" )
    FROM "Kasse"
    ) AS "Saldo"
FROM "Kasse";
```


Abfragen: Unterabfragen
SELECT
   "ID",
   "Einnahme",
   "Ausgabe",
   ( SELECT
       SUM( "Einnahme" ) - SUM( "Ausgabe" )
   FROM "Kasse"
   WHERE "RechnungID" = :Rechnungsnummer)
AS "Saldo"
FROM "Kasse"
WHERE "RechnungID" = :Rechnungsnummer;

Bei einer Parameterabfrage muss der Parameter in beiden Abfrageanweisungen gleich sein, wenn er als ein Parameter verstanden werden soll.


Hinweis

Unterformulare, die auf Abfragen beruhen, werden nicht in Bezug auf die Parameter automatisch aktualisiert. Es bietet sich also eher an, den Parameter direkt aus dem darüber liegenden Formular weiter zu geben.

Korrellierte Unterabfrage

Mittels einer noch verfeinerten Abfrage lässt sich innerhalb einer bearbeitbaren Abfrage sogar der laufende Kontostand mitführen:

SELECT
   "ID",
   "Einnahme",
   "Ausgabe",
   ( SELECT
       SUM( "Einnahme" ) - SUM( "Ausgabe" )
   FROM "Kasse"
   WHERE "ID" <= "a"."ID"
   AS "Saldo"
   FROM "Kasse" AS "a"
ORDER BY "ID" ASC


Hinweis

Soll nach den Inhalten der Unterabfrage mit Hilfe der Filterfunktionen des Abfrageeditors gefiltert werden, so funktioniert dies zur Zeit nur, wenn statt der einfachen Klammern zu Beginn und Ende der Unterabfrage die Klammern doppelt gesetzt werden: «((SELECT ....)) AS "Saldo" »
Abfragen als Bezugstabellen von Abfragen

In einer Abfrage soll für alle Leser, bei denen eine 3. Mahnung zu einem Medium vorliegt, ein Sperrvermerk ausgegeben werden.

```sql
SELECT "Ausleihe"."Leser_ID",
     '3 Mahnungen - der Leser ist gesperrt' AS "Sperre"
FROM (SELECT COUNT( "Datum" ) AS "Anzahl",
     "Ausleihe_ID"
FROM "Mahnung"
GROUP BY "Ausleihe_ID")
AS "a",
"Ausleihe"
WHERE "a"."Ausleihe_ID" = "Ausleihe"."ID"
AND "a"."Anzahl" > 2
```


Die äußere Abfrage bezieht sich in diesem Fall nur in der Bedingungsformulierung auf die innere Abfrage. Sie zeigt nur dann eine "Leser_ID" und den Text zur "Sperre" an, wenn "Ausleihe"."ID" und "a"."Ausleihe_ID" gleich sind sowie "a"."Anzahl" > 2 ist.

Prinzipiell stehen der äußeren Abfrage alle Felder der inneren Abfrage zur Verfügung. So ließe sich beispielsweise die Anzahl mit "a"."Anzahl" in der äußeren Abfrage einblenden, damit auch die tatsächliche Mahnzahl erscheint.

Es kann allerdings sein, dass im Abfrageeditor die grafische Benutzeroberfläche nach so einer Konstruktion nicht mehr verfügbar ist. Soll die Abfrage anschließend wieder zum Bearbeiten geöffnet werden, so erscheint die folgende Meldung:

Ist die Abfrage dann in der SQL-Ansicht zum Bearbeiten geöffnet und wird versucht von dort in die grafische Ansicht zu wechseln, so erscheint die Fehlermeldung:

```
LibreOffice Base

Die Abfrage basiert auf einem SQL-Kommando, das nicht analysiert werden konnte.
Die Abfrage wird in der SQL-Ansicht geöffnet.

OK | Mehr
```

```
LibreOffice Base

Die Spalte 'Ausleihe_ID' ist unbekannt.
Die Spalte konnte nicht gefunden werden. Bitte beachten Sie, dass die Datenbank Groß- und Kleinschreibung unterscheidet.

OK | Mehr
```

Abfragen: Abfragen als Bezugstabellen von Abfragen 264
Die grafische Benutzeroberfläche findet also nicht das in der inneren Abfrage enthaltene Feld "Ausleihe_ID", mit dem die Beziehung von innerer und äußerer Abfrage geregelt wird.

Wird die Abfrage allerdings aufgerufen, so wird der entsprechende Inhalt anstandslos wiedergegeben. Es muss also nicht der direkte SQL-Mode bemüht werden. Damit stehen auch die Sortier- und Filterfunktionen der grafischen Benutzeroberfläche weiter zur Verfügung.

Die folgenden Screenshots zeigen, wie der unterschiedliche Weg zu einem Abfrageergebnis mit Unterabfragen auch verlaufen kann. Hier soll in der Abfrage einer Rechnungsdatenbank ermittelt werden, was der Kunde an der Kasse letztlich zahlen muss. Die Preise werden multipliziert mit der Anzahl der entsprechenden Ware zum "Teilbetrag". Außerdem soll auch noch die Summe der Teilbeträge ausgegeben werden. Und all das soll editierbar bleiben, damit die Abfrage als Grundlage für ein Formular dienen kann.


## Hinweis
Aufgrund des Bugs 61871 aktualisiert Base leider den Teilbetrag nicht automatisch.
Abbildung 42: Die Tabelle "Ware" wurde in eine Unterabfrage verschoben. Diese Unterabfrage wird im Tabellenbereich (nach dem Begriff «FROM») erstellt und mit einem Alias versehen. Jetzt ist der Primärschlüssel aus der Tabelle "Ware" in der Abfrage nicht mehr zwingend erforderlich. Die Abfrage bleibt auch so editierbar.

Abbildung 43: Jetzt soll die Rechnungssumme noch in der Abfrage erscheinen. Bereits die einfache Abfrage der Rechnungssumme ist nicht editierbar, da hier gruppiert und summiert wird.
Abbildung 44: Mit der zweiten Unterabfrage wird das scheinbar Unmögliche möglich. Die vorhergehenden Abfrage wird als Unterabfrage in der Tabellendefinition dieser Abfrage (nach «FROM») eingefügt. Im Ergebnis bleibt die gesamte Abfrage so editierbar. Eingaben sind in diesem Fall nur in den Spalten "Anzahl" und "warID" möglich. Dies wird im Formular anschließend berücksichtigt.

### Zusammenfassung von Daten mit Abfragen

Sollen Daten in der gesamten Datenbank gesucht werden, so ist dies über die einfachen Formularfunktionen meist nur mit Problemen zu bewältigen. Ein Formular greift schließlich nur auf eine Tabelle zu, und die Suchfunktion bewegt sich nur durch die Datensätze dieses Formulars.

Einfacher wird der Zugriff auf alle Daten mittels Abfragen, die wirklich alle Datensätze abbilden. Im Kapitel *Beziehungsdefinition in der Abfrage* wurde so eine Abfragekonstruktion bereits angedeutet. Diese wird im Folgenden entsprechend der Beispieldatenbank ausgebaut.

```
SELECT
  "Medien"."Titel",
  "Untertitel"."Untertitel",
  "Verfasser"."Verfasser"
FROM "Medien"
LEFT JOIN "Untertitel"
```

Abfragen: Zusammenfassung von Daten mit Abfragen
Hier werden alle "Titel", "Untertitel" und "Verfasser" zusammen angezeigt.


Zu 2 Titeln der Tabelle "Medien" existiert noch keine Verfassereingabe und kein "Untertitel". Gleichzeitig existieren bei einem "Titel" insgesamt 3 "Verfasser". Würde jetzt die Tabelle Verfasser einfach ohne LEFT JOIN verbunden, so würden die beiden "Medien" ohne "Verfasser" nicht angezeigt. Da aber ein Medium statt einem Verfasser drei Verfasser hat würde die angezeigte Zahl an Datensätzen bei 15 Datensätzen bleiben.

Erst über die Kette von LEFT JOIN wird die Abfrage angewiesen, weiterhin die Tabelle "Medien" als den Maßstab für alles anzuzeigende zu nehmen. Jetzt erscheinen auch wieder die Datensätze, zu denen weder "Untertitel" noch "Verfasser" existieren, also insgesamt 17 Datensätze.


**Schnellerer Zugriff auf Abfragen durch Tabellenansichten**


Bezieht sich eine neue Abfrage auf eine andere Abfrage, so sieht das in Base in der SQL-Ansicht so aus, als ob die andere Abfrage eine Tabelle wäre. Wird daraus ein 'View' erstellt, so zeigt sich, dass eigentlich mit Unterabfragen (Subselects) gearbeitet wird. Eine Abfrage 2, die sich auf eine andere Abfrage 1 bezieht, kann daher nicht über SQL-Kommando direkt ausführen ausgeführt werden.
werden, da nur die grafische Benutzeroberfläche, nicht aber die Datenbank selbst die Abfrage 1 kennt.


<table>
<thead>
<tr>
<th>Tipp</th>
<th>Eine Abfrage, die über <strong>SQL-Kommando direkt ausführen</strong> gestartet wird, hat den Nachteil, dass sie für die GUI nicht mehr sortiert und gefiltert werden kann. Sie kann also nur beginőt genutzt werden. Ein View hingegen ist für Base handhabbar wie eine normale Tabelle – mit der Ausnahme, dass keine Änderung der Daten möglich ist. Hier stehen also trotz direktem SQL-Befehle alle Möglichkeiten zur Sortierung und zur Filterung zur Verfügung. Auch bleibt die Formatierung von Spalten in der Ansicht im Gegensatz zu Spalten in der Abfrage beim Schließen der Datenbank erhalten.</th>
</tr>
</thead>
</table>

Ansichten sind bei manchen Abfragekonstruktionen eine Lösung, um überhaupt ein Ergebnis zu erzielen. Wenn z. B. auf das Ergebnis eines Subselects zugegriffen werden soll, so lässt sich dies nur über den Umweg eines Views erledigen. Entsprechende Beispiele im Kapitel «Datenbank-Aufgaben komplett»: **Zeilennummerierung** und **Gruppieren und Zusammenfassen**

| Hinweis | Seit der LibreOffice-Version 4.3 ist ein Bearbeiten der Ansichten in Base nicht mehr möglich. Wird über einen rechten Mausklick die Funktion zum Bearbeiten aufgerufen, so wird ein Abfrageeditor ohne jeglichen Abfrageinhalt angezeigt (Bug 80938) Gegebenenfalls also nachsehen, ob dieser Bug inzwischen behoben ist. |

**Rechenfehler in Abfragen**

Abfragen dienen auch zur Berechnung von Werten. Manchmal produziert die interne HSQLDB-Datenbank dabei scheinbar Fehler, die bei genauerer Kenntnis des Zusammenhanges allerdings als logisch korrekter Umgang mit der Sache verstanden werden können. Dann wiederum gibt es Rundungsprobleme, die leicht zu Verwirrung beitragen können.

Zeiten werden innerhalb der HSQLDB nur bis zu einer Differenz von 23:59:59 Stunden korrekt als Zeit formatiert angezeigt. Sollen mehrere Zeitdifferenzen z.B. bei Arbeitszeiten addiert werden, so muss ein anderer Weg beschritten werden. Insgesamt gibt es hier unterschiedlich komplizierte Ansätze:


Wird stattdessen der Zeitunterschied in Minuten ermittelt und anschließend durch 60 dividiert, so erscheint der Zeitunterschied in Stunden als 0. Es wird jetzt scheinbar der korrekte Wert angezeigt – nur wo ist die Minute geblieben?


Statt mit lauter angehängten Nullen zu arbeiten kann auch der Datentyp von DATEDIFF direkt beeinflusst werden. Mit 
\[
(CONVERT(DATEDIFF('mi', "Datum_Zeit_Start", "Datum_Zeit_Ende"), DECIMAL(50, 49)))/60
\]

wird eine Genauigkeit von 49 Nachkommastellen erreicht.

Bei der Nutzung der Rechenfunktionen sollte immer klar sein, dass die Datentypen der HSQLDB nur eine festgesetzte Genauigkeit haben. Auch noch so viele Nachkommastellen können nicht darüber hinwegtäuschen, dass ein Zwischenergebnis mit einer periodischen Zahl nur bedingt für weitere Berechnungen geeignet ist.

Soll eine Zeit hinterher in einem Formular oder einem Bericht als formatierbare Zeit vorliegen, so ist zu berücksichtigen, dass als Grundlage des Zeitformates der Tag gilt.
Hier wird die Differenz in Minuten berechnet. Die Angabe wird als Bruchteil des Tages berechnet. Ein Tag hat 60 * 24 Minuten. Würde einfach durch 1440 geteilt, so wäre das Ergebnis 0, die entsprechend formatierte Zeitangabe auch 0. Hier muss also wieder die Berechnung auf die Nachkommastellen ausgedehnt werden. Erst dann entsteht in der formatierbaren Anzeige die Zeitangabe von 0 Stunden und 1 Minute.

Der Formatcode für eine Zeit, die über einen Tag hinaus geht, lautet \[HH]:MM. Bei falscher Formatierung könnte also ein Zeitunterschied von 1 Tag und 1 Minute lediglich als 1 Minute angezeigt werden.

Schon wieder hat der Fehlerteufel zugeschlagen. Eine Zeitdifferenz von 10 Minuten darf nicht über formatiert 9 Minuten betragen. Wo das Problem steckt zeigt sich bei genauer Betrachtung der Rechnung:

\[10 : 1440 = 0,0069\] → Ergebnis wird gekürzt auf 0,0069, weil nur 4 Nachkommastellen definiert sind und damit bei 4 abgerundet wird → 0,0069 * 1440 = 9,936 → 9 Minuten sowie 56,16 Sekunden, die hier nicht aufgezeigt werden.

Bereits eine Erweiterung um eine Stelle bei dem Divisor (von 1440,0000 auf 1440,00000) behebt diesen Fehler. Jetzt wird gerundet auf 0,00694 → Ergebnis wird gekürzt auf 0,0069, weil nur 4 Nachkommastellen definiert sind und damit bei 4 abgerundet wird → 0,00694 * 1440 = 9,9936 → 0,9936 Minuten sind 59,616 Sekunden. Damit wird die Sekundenzahl auf 60 Sekunden aufgerundet und zu den 9 Minuten eine weitere Minute hinzugerechnet.

So schön es ist, hier eine entsprechend formatierbare Anzeige in einem Formular zu erhalten sollte immer klar sein, dass es sich um gerundete Werte handelt, mit denen tunlichst nicht maschinell weiter gerechnet wird. Die entsprechend erstelle Dezimalzahl würde dazu verleiten, aber leider die entsprechenden Rundungsfehler weitertragen. Für weitere Berechnungen eignet sich daher nur die Darstellung der Zeitdifferenz in der kleinsten erforderlichen Maßeinheit, hier also z.B. in Minuten.
Berichte
Berichte mit dem Report-Designer

Mit Hilfe von Berichten werden Daten so dargestellt, dass sie auch für Personen ohne Datenbankkenntnisse gut lesbar sind. Berichte können

- Daten tabellarisch gut lesbar darstellen,
- zu Daten Diagramme erstellen,
- mit Hilfe von Daten Etikettendruck ermöglichen,
- Serienbriefe wie z. B. Rechnungen, Mahnungen oder auch nur Bestätigungen über einen Vereinsbeitritt oder -austritt erstellen

Um einen Bericht zu erstellen muss die Datenbankgrundlage des Berichtes gut vorbereitet sein. Ein Bericht kann nicht, wie ein Formular, Unterberichte und damit zusätzliche Datenquellen aufnehmen. Ein Bericht kann auch nicht, wie im Formular, über Listenfelder andere Daten darstellen als in der zugrundeliegenden Datenquelle vorhanden sind.


Vorsicht


Die Funktionsweise fertiger Berichte ist davon nicht betroffen – auch wenn diese Berichte unter einer anderen Version erstellt wurden, wo eben z. B. das oben genannte Verhalten nicht auftaucht.

Hinweis

Der Report-Designer ist seit LO 4.1 bereits so in LO integriert, dass er nicht mehr sichtbar in den Erweiterungen («Extensions») erscheint. Es sollte auf jeden Fall vermieden werden, neben dem integrierten Report-Designer noch eine nicht zu der LO-Version passende Report-Designer-Erweiterung zu installieren.


Da mit dem Report-Designer Berichte weiter ausgestaltet werden können behandelt dieses Kapitel nur diese Berichtsvariante.

Die Benutzeroberfläche des Report-Designers

Über Berichte → Bericht in der Entwurfsansicht erstellen ... wird der Report-Designer aufgerufen.

Berichte: Die Benutzeroberfläche des Report-Designers 274
Der Report-Designer startet in einer dreiteiligen Ansicht. Links ist die vorläufige Einteilung des Berichts in Seitenkopf, Detail und Seitenfuß zu sehen, in der Mitte befinden sich die entsprechenden Bereiche, die Inhalte des Berichtes aufnehmen und rechts werden die Eigenschaften des Berichtes angezeigt.


Zwei Ausgabeformate für Berichte sind wählbar: «ODF Textdokument», also ein Writer-Dokument oder «ODF Tabellendokument», also ein Calc-Dokument. Soll einfach nur eine tabellarische Übersicht ausgegeben werden, so ist das Calc-Dokument für die Berichtserstellung eindeutig vorzuziehen. Es ist wesentlich schneller erstellt und kann anschließend auch besser nachformatiert werden, da weniger Formatvorgaben berücksichtigt werden und Spalten einfach nach der erforderlichen Breite anschließend entsprechend gezogen werden können.

Standardmäßig sucht der Report-Designer als Datenquelle die erste Tabelle der Datenbank aus. So ist auf jeden Fall gewährleistet, dass zumindest ein Ausprobieren der Funktionen möglich ist. Erst nach Auswahl der Datenquelle kann der Bericht mit Feldern beschickt werden.

<table>
<thead>
<tr>
<th>Schaltflächen inhaltliche Bearbeitung</th>
<th>Schaltflächen Elementausrichtung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speichern</td>
<td>Links im Bereich ausrichten</td>
</tr>
<tr>
<td>Datei bearbeiten</td>
<td>Rechts im Bereich ausrichten</td>
</tr>
<tr>
<td>Ausschneiden</td>
<td>Oben im Bereich ausrichten</td>
</tr>
<tr>
<td>Kopieren</td>
<td>Unten im Bereich ausrichten</td>
</tr>
<tr>
<td>Einfügen</td>
<td>Verkleinern</td>
</tr>
<tr>
<td>Eigenschaft 'Command' ändern</td>
<td>Von oben verkleinern</td>
</tr>
<tr>
<td>Wiederherstellen</td>
<td>Von unten verkleinern</td>
</tr>
<tr>
<td>Feld hinzufügen</td>
<td>An Element mit geringster Breite anpassen</td>
</tr>
<tr>
<td>Bericht-Navigator</td>
<td>An breitestes Element anpassen</td>
</tr>
<tr>
<td>Sortierung und Gruppierung</td>
<td>An Element mit geringster Höhe anpassen</td>
</tr>
<tr>
<td>Bericht ausführen</td>
<td>An höchstes Element anpassen</td>
</tr>
<tr>
<td>LibreOffice Hilfe</td>
<td>Links</td>
</tr>
<tr>
<td>Direkthilfe</td>
<td>Zentriert</td>
</tr>
<tr>
<td></td>
<td>Rechts</td>
</tr>
<tr>
<td></td>
<td>Oben</td>
</tr>
<tr>
<td></td>
<td>Mitte</td>
</tr>
<tr>
<td></td>
<td>Unten</td>
</tr>
</tbody>
</table>

Der Report-Designer stellt einige zusätzliche Schaltflächen zur Verfügung, so dass in der vorstehenden Tabelle noch einmal die Schaltflächen mit einer entsprechenden Beschriftung abgebildet sind. Die Schaltflächen zur Elementausrichtung werden in diesem Kapitel nicht weiter beschrieben.

Wie schon bei den Formularen ist es hilfreich, den entsprechenden Navigator bei Problemen aufzurufen. So kann es zum Beispiel sein, dass durch einen unvorsichtigen Klick beim Start des Report-Designers die Eigenschaften zu den Daten des Berichts verzweifelt gesucht werden. Diese Daten können nur über den Bericht-Navigator erreicht werden:

Ein Klick mit der linken Maustaste auf «Bericht» und die Eigenschaften des Berichtes sind wieder erreichbar.


Um bei dem oben geöffneten Beispiel sinnvolle Ausgaben zu erhalten, muss der Inhalt der Ansicht gruppiert wiedergegeben werden. Ein Leser soll gebündelt die Anmahnungen für alle seine entliehenen und überzogenen Medien erhalten.


Zuerst erfolgt die Sortierung nach dem obersten eingetragenen Feld. Gegebenenfalls müssen also die Einträge entsprechend verschoben werden, wenn die gewünschte Sortierung nicht eintritt.
Hier wurde nach dem Feld "Leser_Name" gruppiert und sortiert. In die obere Tabelle können untereinander mehrere Felder eingetragen werden. Soll z. B. zusätzlich nach dem "Leih_Datum" gruppiert und sortiert werden, so ist dies als zweite Zeile anzuwählen.


Standardmäßig wird nach jedem Wert gruppiert. Ändert sich also "Leser_Name", so entsteht eine neue Gruppe. Alternativ kann hier nach dem Anfangsbuchstaben gruppiert werden. Das würde aber bei einem Mahnverfahren alle Leser-Nachnamen mit gleichem Anfangsbuchstaben zusammenfassen in einer Gruppe. 'Schmidt', 'Schulze' und 'Schulte' erhielten so eine gemeinschaftliche Mahnung. Eine wohl recht sinnlose Aktion an dieser Stelle für dieses Beispiel.

Abbildung 45: Sortierung und Gruppierung

Hier wurde nach dem Feld "Leser_Name" gruppiert und sortiert. In die obere Tabelle können untereinander mehrere Felder eingetragen werden. Soll z. B. zusätzlich nach dem "Leih_Datum" gruppiert und sortiert werden, so ist dies als zweite Zeile anzuwählen.


Standardmäßig wird nach jedem Wert gruppiert. Ändert sich also "Leser_Name", so entsteht eine neue Gruppe. Alternativ kann hier nach dem Anfangsbuchstaben gruppiert werden. Das würde aber bei einem Mahnverfahren alle Leser-Nachnamen mit gleichem Anfangsbuchstaben zusammenfassen in einer Gruppe. 'Schmidt', 'Schulze' und 'Schulte' erhielten so eine gemeinschaftliche Mahnung. Eine wohl recht sinnlose Aktion an dieser Stelle für dieses Beispiel.

Je nach Einstellung kann eine Gruppe entweder mit dem ersten Detail zusammen gehalten werden, oder, sofern möglich, als ganze Gruppe. Standardmäßig ist das Zusammenhalten auf «Nein» eingestellt. Für ein Mahnverfahren wird vermutlich sowieso die Gruppe so angeordnet, dass für jede Person, die eine Mahnung erhalten soll, eine Seite ausgedruckt wird. Daher ist stattdessen an anderer Stelle zu wählen, dass nach der Gruppe jeweils ein Seitenumbruch erfolgt, bevor der nächste Wert abzuzeichnen ist.

Sind Gruppenkopf und gegebenenfalls Gruppenfuß ausgewählt, so erscheinen diese Elemente als Teile des Bericht navigators unter dem entsprechenden Feldnamen "Leser_Name". Zusätzlich wird auch da wieder die Möglichkeit für Funktionen geboten, die sich nur auf diese Gruppe beschränken.

Das obige Bild zeigt den Berichtsentwurf für die Mahnung an. Im Seitenkopf ist fest die Überschrift Libre-Office Bibliothek als Beschriftungsfeld eingesetzt. Hier könnte auch ein Briefkopf mit Logo stehen, da auch die Einbindung von Grafiken möglich ist. Wenn die Ebene «Seitenkopf» heißt, so

«Leser_Name Kopf» ist die Gruppierung und Sortierung, nach der die Daten zusammengefasst werden sollen. In den Feldern, die später Daten aufnehmen, steht hellgrau die Bezeichnung der Datenfelder, die hier ausgelesen werden. So hat die dem Bericht zugrundeliegende Ansicht z. B. ein Feld mit der Bezeichnung Adresse, in dem die komplette Adresse mit Straße und Ort für die anzumahnende Person steht. Um dies in ein Feld zu setzen, sind Absatzumbrüchen in der Abfrage notwendig. Mittels \texttt{CHAR(13) \textbar | CHAR(10)} wird in einer Abfrage ein Absatz erzeugt. Beispiel:

\begin{verbatim}
SELECT "Anrede" \textbar | CHAR(13) \textbar | CHAR(10) \textbar | "Vorname" \textbar | ' ' \textbar | "Nachname" \textbar | CHAR(13) \textbar | CHAR(10) \textbar | "Strasse" \textbar | ' ' \textbar | "Nr" \textbar | CHAR(13) \textbar | CHAR(10) \textbar | "Postleitzahl" \textbar | ' ' \textbar | "Ort" AS "Adresse" FROM "Leser"
\end{verbatim}

Bei dem Feld ≈\texttt{TODAY()}≈ handelt es sich um eine eingebaute Funktion, die das aktuelle Datum an dieser Stelle einliest.

In «Leser_Name Kopf» sind außer der Anrede und weiteren Informationen auch die Spaltenköpfe für die anschließende Tabellenansicht untergebracht. Sie sollen ja nur einmal auftauchen, auch wenn mehrere Medien aufgelistet werden.

In den Hintergrund dieser Spaltenköpfe wurde je ein graues Rechteck gelegt. Dieses Rechteck sorgt gleichzeitig für eine entsprechende Umrandung.

Der Detailbereich wird so oft wiederholt, wie unterschiedliche Datensätze mit den gleichen Daten existieren, die in «Leser_Name» stehen. Hier werden also alle Medien aufgelistet, die nicht rechtzeitig zurückgegeben wurden. Auch hier liegt im Hintergrund ein Rechteck, um die Inhalte zu umranden. Das Rechteck selbst hat die Füllfarbe «weiß».

\begin{table}
\begin{tabular}{|l|l|}
\hline
\textbf{Hinweis} & Grundsätzlich gibt es in LO auch die Möglichkeit, horizontale und vertikale Linien hinzu zu fügen. Im Design-Modus werden diese auch angezeigt. Beim Ausführen des Berichtes erscheinen sie erst ab der Version LO 4.0.5 bzw. LO 4.1.1. Diese Linien haben den Nachteil, dass sie nur als Haarlinien ausgelegt sind. Sie lassen sich besser nachbilden, indem Rechtecke benutzt werden. Der Hintergrund der Rechtecke wird auf die Farbe Schwarz eingestellt, die Größe wird z. B. mit einer Breite von 17 cm und einer Höhe von 0,03 cm festgelegt. Dann erscheint eine horizontale Linie mit einer Dicke von 0,03 cm mit einer Länge von 17 cm. Leider hat auch diese Variante einen Nachteil: grafische Elemente lassen sich nicht richtig positionieren, wenn der Bereich über eine Seite hinweg geht.
\hline
\end{tabular}
\end{table}

\begin{table}
\begin{tabular}{|l|}
\hline
\textbf{Allgemein} \\
\hline
Name……………………… | Gruppenfuß \\
Seitenumbruch erzwingen…… | Nach Bereich \\
Zusammenhalten……………… | Ja \\
Bereich wiederholen…………… | Nein \\
Sichtbar…………………… | Ja \\
Höhe………………………… | 4,87 cm \\
Ausdruck für bedingte Anzeige………………………… | … \\
Hintergrund transparent……………… | Ja \\
Hintergrundsfarbe……… | \\
\hline
\end{tabular}
\end{table}
Der «Leser_Name Fuß» schließt schließlich das Briefformat mit einer Grußformel und einem Unterschriftsbereich ab. Der Fuß ist so definiert, dass anschließend ein Seitenumbruch nach dem Bereich erfolgt. Außerdem ist gegenüber den Standardeinstellungen verändert so eingestellt, dass der Bereich auf jeden Fall zusammen gehalten werden soll. Schließlich würde es reichlich merkwürdig aussehen, wenn bei vielen Mahnungen allein z. B. das Unterschriftsfeld auf die nächste Seite verschoben würde.

Zusammenhalten bezieht sich hier immer auf den Seitenumbruch. Soll der Inhalt eines Datensatzes unabhängig vom Seitenumbruch zusammengehalten werden, so ist dies zur Zeit nur möglich, indem der Datensatz nicht als «Detail» eingelesen wird, sondern als Grundlage für eine Gruppierung genommen wird. Der Bereich «Detail» wird leider auch dann aufgetrennt, wenn «Zusammenhalten – Ja» ausgewählt wird.

Für die Aufsummierung der Gebühren wurde eine interne Funktion benutzt.

### Tabelle der Mahnrufe

<table>
<thead>
<tr>
<th>Mahndatum</th>
<th>Mahn-Nr.</th>
<th>Medium</th>
<th>Liefdatum</th>
<th>Überzogen</th>
<th>Gebühr</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.04.12</td>
<td>2</td>
<td>3 - Traditoriale und kritische Theorie - von Hermann, Max</td>
<td>09.12.11</td>
<td>128 Tage</td>
<td>4,50 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>2</td>
<td>5 - I hear you moaning - von Edmunds, Dave</td>
<td>28.11.11</td>
<td>139 Tage</td>
<td>4,75 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>3</td>
<td>4 - Die neue deutsche Rechtschreibung - von Hermann, Ursula</td>
<td>09.11.11</td>
<td>151 Tage</td>
<td>5,25 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>1</td>
<td>1 - Das sogenannte Böse - von Lorenz, Konrad</td>
<td>06.04.12</td>
<td>4 Tage</td>
<td>0,00 €</td>
</tr>
<tr>
<td>22.04.12</td>
<td>1</td>
<td>7 - Das Postfix Buch - von ?</td>
<td>25.02.12</td>
<td>43 Tage</td>
<td>1,50 €</td>
</tr>
</tbody>
</table>

18,00 €

Mit freundlichen Grüßen

(Bibliotheksverwaltung)
So könnte dann eine entsprechende Mahnung aussehen. Im Detailbereich sind hier 5 Medien angegeben, die der Leser entliehen hat. Im Gruppenfuß wird noch einmal die Summe für die Anmahnung ausgegeben.

Allgemeine Eigenschaften von Feldern


Felder werden wie bei den Formularen mit Namen bezeichnet. Standardmäßig ist hier der Name gleich der Feldbezeichnung der zugrundeliegenden Datenbank.

Ein Feld kann unsichtbar geschaltet werden. Bei Feldern macht dies vielleicht wenig Sinn, bei Gruppenkopf oder Gruppenfuß hingegen schon eher, da hier auch andere Funktionen der Gruppie-
rung erfüllt sein sollen, der Gruppenkopf oder Gruppenfuß aber nicht unbedingt mit Inhalt versehen ist.


In dem Report-Designer kann die Ansicht bestimmter Inhalte durch einen «Ausdruck für bedingte Anzeige» unterdrückt werden oder der Wert des Feldes als Grundlage für eine Formatierung von Schrift und Hintergrund genommen werden. Mehr zu diesen Ausdrücken unter «Bedingte Anzeige».

Die Einstellung zum Mausradverhalten spielt keine Rolle, da ja die Felder in dem Bericht nicht editierbar sind. Sie scheint ein Überbleibsel aus dem Bereich der Formulare zu sein.

Die Funktion «Bei Gruppenwechsel anzeigen» konnte im Bericht ebenfalls nicht nachvollzogen werden.

Ist der Hintergrund nicht als transparent definiert, so kann für das jeweilige Feld eine Hintergrundfarbe definiert werden.

Die weiteren Einträge beziehen sich auf den Inhalt innerhalb der gezogenen Felder. Dies sind im Einzelnen die Schriftart (mit Schriftfarbe, Schriftdicke etc., siehe Abbildung 46), die Ausrichtung des Schriftzugs in dem Feld und die Formatierung mit dem entsprechenden Dialog «Zahlenformat» (siehe Abbildung 47).

Abbildung 46: Schrift - Zeicheneinstellungen

Berichte: Die Benutzeroberfläche des Report-Designers 283
Das grafische Kontrollfeld kann sowohl Grafiken von außerhalb der Datenbank darstellen als auch Grafiken aus der Datenbank auslesen. Leider ist es zur Zeit nicht möglich, eine Grafik dauerhaft in Base zu speichern, um z. B. ein Brieflogo einzulesen. Hierzu muss zwingend die Grafik in dem
gesuchten Pfad vorhanden sein, auch wenn die Auswahl anbietet, Bilder ohne Verknüpfung aufzunehmen und das erste Feld mit «Als Verknüpfung vorbereiten» auf eine entsprechende geplante Funktionalität schließen lässt.

Alternativ dazu kann natürlich eine Grafik innerhalb der Datenbank selbst gespeichert werden und so auch intern verfügbar bleiben. Sie muss dann aber über die dem Bericht zugrundeliegende Abfrage in einem der Felder zugänglich sein.


Die Einstellung der Vertikalen Ausrichtung scheint im Entwurf nichts zu bewirken. Wird allerdings der Bericht aufgerufen, so erscheint die Grafik entsprechend positioniert.


- 'Nein': Das Bild wird nicht an das Kontrollfeld angepasst. Ist das Bild zu groß, so wird ein Ausschnitt des Bildes gezeigt. Das Bild wird nicht verzerrt.
- 'Seitenverhältnis beibehalten': Das Bild wird in das Kontrollfeld eingepasst, aber nicht verzerrt dargestellt.
- 'Automatische Größe': Das Bild wird der Größe des Kontrollfeldes angepasst und gegebenenfalls verzerrt dargestellt.

Wenn die Datenbankdatei in einem Packprogramm geöffnet wird, ist dieser Ordner mit Inhalt im Verzeichnis «reports» im Unterverzeichnis zu dem jeweiligen Bericht sichtbar. Dieses Verzeichnis kann über das Packprogramm gefahrlos gelöscht werden.

**Hinweis**
Der Bug ist hier gemeldet: https://bugs.freedesktop.org/show_bug.cgi?id=80320

**Diagramme im Bericht einbinden**
Über das entsprechende Kontrollfeld oder **Einfügen → Bericht-Steuerelemente → Diagramm** lässt sich ein Diagramm dem Bericht hinzufügen. Ein Diagramm wäre hier die einzige Möglichkeit, in einem Bericht Daten wiederzugeben, die nicht aus der Quelle stammen, die im Bericht als Datenquelle angegeben ist. Ein Diagramm ist in sofern ein Unterbericht des Berichtes, kann aber auch als eigenständiger Berichtsteil gesehen werden.

Das Diagramm wird aufgezogen. Bei den allgemeinen Eigenschaften zeigt sich neben bekannten Feldern die Möglichkeit, einen *Diagrammtypen* auszuwählen (siehe entsprechende Typen in Calc). Außerdem wird eine maximal für die Vorschau benutzte Anzahl an Datensätzen (*Vorschau für Zeile(n)*) eingestellt, die schon einmal eine Vorstellung davon geben kann, wie das Diagramm letztlich aussieht.

Diagramme können, ebenfalls wie in Calc, entsprechend formatiert werden (Doppelklick auf das Diagramm. Hierzu siehe die Beschreibungen im Handbuch Calc.

beschriftet werden sollen, die zweite Spalte liefert dann die Anzahl der Entleihvorgänge, die sich in den Höhen der Säulen widerspiegelt.

In dem obigen Beispiel zeigt das Diagramm erst einmal nur sehr wenig an, da die Test-Entleihvorgänge sich zum Zeitpunkt der SQL-Eingabe in Grenzen hielten.


Eine Filterung und Sortierung mit den internen Werkzeugen des Report-Designers ist nicht notwendig, weil dies eben schon so weit wie möglich über die Abfrage erledigt wurde.


Eine Filterung und Sortierung mit den internen Werkzeugen des Report-Designers ist nicht notwendig, weil dies eben schon so weit wie möglich über die Abfrage erledigt wurde.


**Tipp**

Grundsätzlich sollte die Berichtserstellung von so viel Aufgaben wie möglich entlastet werden. Was vorher durch Abfragen bewerkstelligt werden kann, das muss nachher nicht mehr die im Verhältnis zur Datenbank doch recht träge Ausführung des Berichtes belasten.


**Tipp**

Grundsätzlich sollte die Berichtserstellung von so viel Aufgaben wie möglich entlastet werden. Was vorher durch Abfragen bewerkstelligt werden kann, das muss nachher nicht mehr die im Verhältnis zur Datenbank doch recht träge Ausführung des Berichtes belasten.


---

8 Die Datenbank «Beispiel_Sport.odb» ist den Beispieldatenbanken für dieses Handbuch beigefügt.

Der Reiter «Datenreihen» hingegen verbirgt ein paar Einstellungsmöglichkeiten, die die Standarddarstellung deutlich verändern können. Es werden dort alle Datenreihen angeboten, die neben der ersten Spalte der Abfrage noch verfügbar sind. Datenreihen, die nicht angezeigt werden sollen, können hier entfernt werden.

Nachdem bis auf die Datenreihe «Anzahl» alle Datenreihen entfernt wurden sieht die Vorschau des Diagramms so aus:

Die Vorschau zeigt hier 10 Säulen auf – die ersten 10 Säulen aus der Abfrage. In der Ausführung werden allerdings anschließend nur die Säulen dargestellt, die zu der entsprechenden Gruppe passen: «m» für «männlich» und «w» für «weiblich».

Alle weiteren Einstellungen sind ähnlich den Möglichkeiten, die LO-Calc für die Einstellung von Diagrammen bietet.

### Hinweis

Die Darstellung von Diagrammen gelingt in LibreOffice zur Zeit nur den Versionen 3.3.*, 3.4.* sowie 4.*. Die Versionen 3.5.* und 3.6.* von LO sind nicht in der Lage, diese Berichte zu öffnen. Bei der Vorversion LO 5.0.0.3 tritt leider erneut ein Fehlverhalten auf, das mit einer langen Fehlerliste das Öffnen des Berichtes aufgibt.

Nicht nur LibreOffice hat hiermit Probleme. Auch in den Versionen ab 3.3 von OpenOffice klappt die Darstellung nicht. OpenOffice öffnet zwar die Berichte – aber eben einfach ohne die Diagramme.

Sollen also Diagramme in einem Bericht auf jeden Fall genutzt werden, so empfiehlt sich zur Zeit immer eine Version installiert zu lassen – zur Not auch als portable Version – die sich beim Öffnen der Diagramme bewährt hat.

### Dateneigenschaften von Feldern

<table>
<thead>
<tr>
<th>Allgemein</th>
<th>Daten</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datenfeld-Typ:</td>
<td>Feld oder Formel</td>
</tr>
<tr>
<td>Datenfeld:</td>
<td>Gebühr</td>
</tr>
<tr>
<td>Funktion:</td>
<td></td>
</tr>
<tr>
<td>Geltungsbereich:</td>
<td></td>
</tr>
</tbody>
</table>


Vorwählbar sind die Funktionen «Summe», «Minimum» und «Maximum». Ihr Geltungsbereich bezieht sich entweder auf die aktuelle Gruppe oder auf den gesamten Bericht. Bereits diese Funktionen können zu Problemen führen, wenn ein Feld leer, also NULL, ist. In solchen Feldern, sofern sie als Zahl formatiert sind, erscheint «N,aN», was wohl so viel wie «kein Zahlenwert» bedeuten soll. Mit leeren Feldern wird keine Rechnung durchgeführt; das Ergebnis ist 0.

Solche Felder können zwar in der Ansicht durch die folgende Formel im Bereich «Daten» zur Anzeige von 0-Werten umformatiert werden.

\[
\text{IF}([\text{Zahlenfeld}];[\text{Zahlenfeld}];0)
\]

Die Funktion rechnet aber mit dem tatsächlichen Wert, also mit dem Feld, das eben keinen Wert besitzt. Da erscheint es dann einfacher, wenn gewünscht, die dem Bericht zugrundeliegende Abfrage so zu formulieren, dass statt NULL in Zahlenfeldern '0' wiedergegeben wird.

Der «Zähler» zählt einfach nur die Datensätze, die entweder in einer Gruppe oder im ganzen Bericht enthalten sind. Wird der Zähler in den Bereich Detail eingesetzt, so wird durch ihn jeder Datensatz mit einer fortlaufenden Nummer versehen. Dabei kann die Nummerierung auch hier nur die Gruppe oder sämtliche Datensätze des Berichts fortlaufend nummerieren.
Schließlich steht noch die detaillierten *Benutzerdefinierte Funktionen* zur Verfügung. Es kann passieren, dass der Report-Designer diese Variante selbst wählt, wenn er zwar den Rechenschritt vorgegeben bekommt, aber die Datenquelle aus irgendeinem Grunde nicht richtig interpretieren kann. Dann können allerdings dort auch weitere Einstellungen vorgenommen werden.


**Funktionen im Report-Designer**

Der Report-Designer bietet sowohl bei den Daten als auch bei den Bedingungen für eine Anzeige die Nutzung verschiedener Funktionen an. Reichen die Funktionen nicht, so lassen sich mit einfachen Rechenschritten auch benutzerdefinierte Funktionen erstellen, die vor allem in Gruppenfüßen als Zusammenfassungen Sinn machen.

**Formeleingaben**


Teilweise ist auch ein einfacher Blick in Richtung Calc sinnvoll, da dort eine deutsche Beschreibung zumindest einiger Formeln aus dem Report-Designer enthalten ist.
Grundlegend gilt:

| Formeleingaben starten mit einem Gleichheitszeichen | = |
| Bezüge zu Datenfeldern werden in eckige Klammer gesetzt | [Feldbezeichnung] |
| Enthalten die Datenfelder Sonderzeichen, zu denen auch Leertasten gehören, so ist die Feldbezeichnung außerdem noch in Anführungsstriche zu setzen | ["Diese Feldbezeichnung wäre in Anführungsstriche zu setzen"] |
| Texteingaben haben immer in doppelten Anführungszeichen zu stehen | "Texteingabe" |
| Folgende Operatoren sind möglich | +, -, *, / (Division), % (dividiert vorherstehende Zahl durch 100), ^ (Potenzierung mit der nachfolgenden Zahl), & (verbindet Texte miteinander), |
| Folgende Beziehungsdefinitionen sind möglich | = , <> , < , <= , > , >= |
| Runde Klammerungen sind ebenfalls möglich | ( ) |
| Standardfehlermeldung | NA (vermutlich not available, in Calc in NV übersetzt – nicht verfügbar) |
| Fehlermeldung beim Bericht, wenn ein leeres Feld auftaucht und als Zahl definiert ist. | N,aN (erscheint, wenn kein Wert enthalten ist – vielleicht not a number?) |

Sämtliche Formeleingaben wirken sich nur auf den aktuell eingelesenen Datensatz aus. Beziehungen zu vorhergehenden oder nachfolgenden Datensätzen sind also nicht möglich.

Im Gegensatz zu den Funktionen in Calc sind hier die Funktionen nicht ins Deutsche übertragen. Die Funktionen sind allerdings längst nicht so umfangreich wie die, die in Calc üblich sind. Viele Funktionen haben allerdings ihre Entsprechung in Calc. Dort gibt der Assistent dann auch direkt das Ergebnis der Funktion wieder.


Die folgenden Funktionen stehen zur Verfügung: [Funktion im Report-Designer / (Funktion in Calc)]

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE (DATUM)</td>
<td>Erzeugt aus einer Zahlenangabe für das Jahr, den Monat und den Tag ein gültiges Datum.</td>
</tr>
<tr>
<td>DATEDIF (TAGE</td>
<td>MONATE</td>
</tr>
<tr>
<td>DATEVALUE (DATWERT)</td>
<td>Wandelt eine amerikanische Datumseingabe in Textform (Anführungsstriche) in eine Datumsbeschreibung um. Die erzeugte amerikanische Variante kann durch Formatierungseinstellung umgewandelt werden.</td>
</tr>
<tr>
<td>DAY (TAG)</td>
<td>Gibt den Tag des Monats eines angegebenen Datums wieder. DAY([Datumsfeld])</td>
</tr>
</tbody>
</table>

Berichte: Funktionen im Report-Designer 293
<table>
<thead>
<tr>
<th>Funktion</th>
<th>Erklärung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAYS (TAGE)</td>
<td>Gibt die Zahl der Tage zwischen zwei Datumseingaben wieder.</td>
</tr>
<tr>
<td>HOUR (STUNDE)</td>
<td>Gibt die Stunde einer angegebenen Zeit im Rahmen von 0 bis 23 wieder.</td>
</tr>
<tr>
<td></td>
<td>HOUR([DatumZeitFeld]) gibt die Stunde des Feldes wieder.</td>
</tr>
<tr>
<td>MINUTE (MINUTE)</td>
<td>Gibt die Minuten einer internen Zahl wieder.</td>
</tr>
<tr>
<td></td>
<td>MINUTE([Zeitfeld]) gibt den Minutenanteil der Zeit wieder.</td>
</tr>
<tr>
<td>MONTH (MONAT)</td>
<td>Gibt den Monat einer Datumseingabe als Zahl wieder.</td>
</tr>
<tr>
<td></td>
<td>MONTH([Datumsfeld])</td>
</tr>
<tr>
<td>NOW (JETZT)</td>
<td>Gibt das aktuelle Datum und die aktuelle Zeit wieder.</td>
</tr>
<tr>
<td>SECOND (SEKUNDE)</td>
<td>Gibt die Sekunden einer internen Zahl wieder.</td>
</tr>
<tr>
<td></td>
<td>SECOND(NOW()) gibt den Sekundenanteil der Zeit beim Ausführen des Berichts</td>
</tr>
<tr>
<td>TIME (ZEIT)</td>
<td>Zeigt die aktuelle Zeit an.</td>
</tr>
<tr>
<td>TIMEVALUE (ZEITWERT)</td>
<td>Wandelt die Texteingabe einer Zeit in eine Zeit für Berechnungen um</td>
</tr>
<tr>
<td>TODAY (HEUTE)</td>
<td>Gibt das aktuelle Datum wieder</td>
</tr>
<tr>
<td>WEEKDAY (WOCHENTAG)</td>
<td>Gibt den Wochentag einer Datumseingabe als Zahl wieder. Der Tag mit der</td>
</tr>
<tr>
<td></td>
<td>Nummer 1 ist der Sonntag.</td>
</tr>
<tr>
<td>YEAR (JAHR)</td>
<td>Gibt das Jahr einer Datumseingabe wieder.</td>
</tr>
</tbody>
</table>

**Logische Funktionen**

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Erklärung</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND (UND)</td>
<td>Gibt TRUE wieder, wenn alle Argumente TRUE sind</td>
</tr>
<tr>
<td>FALSE (FALSCH)</td>
<td>Definiert den logischen Wert als FALSE</td>
</tr>
<tr>
<td>IF (WENN)</td>
<td>Wenn eine Bedingung TRUE, dann diesen Wert, ansonsten einen anderen Wert.</td>
</tr>
<tr>
<td>IFNA</td>
<td>Wenn der Wert nicht verfügbar ist, dann setze diesen Wert. Ist ähnlich</td>
</tr>
<tr>
<td></td>
<td>der Anweisung <strong>IFNULL</strong> in SQL. Vor allem bei Funktionen sinnvoll, die</td>
</tr>
<tr>
<td></td>
<td>Feldwerte addieren sollen, bei denen aber nicht klar ist, ob jedes Feld</td>
</tr>
<tr>
<td></td>
<td>einen Wert hat. <strong>IFNA([Feldwert];0)</strong> rechnet dann trotzdem z.B. die</td>
</tr>
<tr>
<td></td>
<td>Summe aus. (ab LO 3.5)</td>
</tr>
<tr>
<td>NOT (NICHT)</td>
<td>Kehrt den logischen Wert eines Argumentes um</td>
</tr>
<tr>
<td>OR (ODER)</td>
<td>Gibt TRUE zurück, wenn eine der Bedingungen TRUE ist.</td>
</tr>
<tr>
<td>TRUE (WAHR)</td>
<td>Definiert den logischen Wert als TRUE</td>
</tr>
<tr>
<td>XOR</td>
<td>Nur wenn ein einziger Wert der Verknüpfung TRUE ist ist auch der</td>
</tr>
<tr>
<td></td>
<td>logische Wert TRUE.</td>
</tr>
</tbody>
</table>
### Rundungsfunktionen

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Rundet zum nächsten Ganzzahlwert ab</td>
</tr>
</tbody>
</table>

### Mathematische Funktionen

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS (ABS)</td>
<td>Gibt den absoluten, nicht negativen Wert einer Zahl wieder.</td>
</tr>
<tr>
<td>ACOS (ARCCOS)</td>
<td>Berechnet den Arcuscosinus einer Zahl. - Werteingabe zwischen -1 und 1 (ab LO 3.5)</td>
</tr>
<tr>
<td>ACOSH (ARCCOSHYP)</td>
<td>Berechnet den Areacosinus (inverser hyperbolischer Cosinus) – Werteingabe &gt;= 1 (ab LO 3.5)</td>
</tr>
<tr>
<td>ASIN (ARCSIN)</td>
<td>Berechnet den Arcussinus einer Zahl. - Werteingabe zwischen -1 und 1 (ab LO 3.5)</td>
</tr>
<tr>
<td>ATAN (ARCTAN)</td>
<td>Berechnet den Arcustangens einer Zahl. (ab LO 3.5)</td>
</tr>
<tr>
<td>ATAN2 (ARCTAN2)</td>
<td>Berechnet den Arcustangens einer x-Koordinaten und einer y-Koordinaten. (ab LO 3.5)</td>
</tr>
<tr>
<td>AVERAGE (MITTELWERT)</td>
<td>Ermittelt den Mittelwert der angegebenen Werte. (taucht im Formularassistenten LO 3.3.4 doppelt auf)</td>
</tr>
<tr>
<td>AVERAGEA (MITTELWERTA)</td>
<td>Ermittelt den Mittelwert der angegebenen Werte. Text wird als 0 gewertet. (ab LO 3.5)</td>
</tr>
<tr>
<td>COS (COS)</td>
<td>Winkel im Bogenmaß, dessen Cosinus berechnet werden soll. (ab LO 3.5)</td>
</tr>
<tr>
<td>EVEN (GERADE)</td>
<td>Rundet eine positive Zahl zum nächsten geraden Integer auf, eine negative Zahl zum nächsten geraden Integer ab.</td>
</tr>
<tr>
<td>EXP (EXP)</td>
<td>Berechnet die Exponentialfunktion zur Basis ‘e’ (ab LO 3.5)</td>
</tr>
<tr>
<td>LN (LN)</td>
<td>Berechnet den natürlichen Logarithmus einer Zahl. (ab LO 3.5)</td>
</tr>
<tr>
<td>LOG10 (LOG10)</td>
<td>Berechnet den Logarithmus einer Zahl zur Basis ‘10’. (ab LO 3.5)</td>
</tr>
<tr>
<td>MAX (MAX)</td>
<td>Gibt den Maximalwert aus einer Reihe von Werten wieder.</td>
</tr>
<tr>
<td>MAXA (MAXA)</td>
<td>Gibt den Maximalwert aus einer Reihe wieder. Eventuell vorhandener Text wird als ‘0’ gesetzt.</td>
</tr>
<tr>
<td>MIN (MIN)</td>
<td>Gibt den Minimalwert aus einer Reihe von Werten wieder.</td>
</tr>
<tr>
<td>MINA (MINA)</td>
<td>Gibt den Minimalwert aus einer Reihe wieder. Eventuell vorhandener Text wird als ‘0’ gesetzt.</td>
</tr>
<tr>
<td>Funktion</td>
<td>Beschreibung</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>MOD</strong> (REST)</td>
<td>Gibt den Rest einer Division nach Eingabe von Dividend und Divisor wieder</td>
</tr>
<tr>
<td><strong>ODD</strong> (UNGERADE)</td>
<td>Rundet eine positive Zahl zum nächsten ungeraden Integer auf, eine negative Zahl zum nächsten ungeraden Integer ab.</td>
</tr>
<tr>
<td><strong>PI</strong> (PI)</td>
<td>Liefert den Wert der Zahl 'π' (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>POWER</strong> (POTENZ)</td>
<td>Liefert aus Basis und Exponent die Potenz. (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>SIN</strong> (SIN)</td>
<td>Berechnet den Sinus einer Zahl. (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>SQRT</strong> (WURZEL)</td>
<td>Berechnet die Quadratwurzel einer Zahl. (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>SUM</strong> (SUMME)</td>
<td>Summiert eine Liste von Zahlenwerten</td>
</tr>
<tr>
<td><strong>SUMA</strong></td>
<td>Summiert eine Liste von Zahlenwerten. Text und Ja/Nein-Felder sind erlaubt. Leider endet diese Funktion (noch) mit Fehlermeldung. (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>VAR</strong> (VARIANZ)</td>
<td>Berechnet die Varianz, ausgehend von einer Stichprobe. (ab LO 3.5)</td>
</tr>
</tbody>
</table>

### Textfunktionen

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXACT</strong> (IDENTISCH)</td>
<td>Zeigt an, ob zwei Texte völlig gleich sind.</td>
</tr>
<tr>
<td><strong>FIND</strong> (FINDEN)</td>
<td>Gibt die Position eines zu suchenden Textes in einem anderen Text an.</td>
</tr>
<tr>
<td><strong>LEFT</strong> (LINKS)</td>
<td>Der Text wird von links aus in der angegebenen Zeichenzahl wieder-gegeben.</td>
</tr>
<tr>
<td><strong>LEN</strong> (LÄNGE)</td>
<td>Gibt die Anzahl an Zeichen wieder, die ein Text hat.</td>
</tr>
<tr>
<td><strong>LOWER</strong> (KLEIN)</td>
<td>Gibt den Text in Kleinbuchstaben wieder.</td>
</tr>
<tr>
<td><strong>MESSAGE</strong></td>
<td>Formatiert die Werte in dem angegebenen Ausgabeformat. (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>MID</strong> (TEIL)</td>
<td>Gibt Text ab einer Anfangsposition mit einer gegebenen Zeichen-länge wider.</td>
</tr>
<tr>
<td><strong>REPLACE</strong> (ERSETZEN)</td>
<td>In dem Text wird ein Teil durch einen anderen Text ersetzt. Die Start-position und die Länge des zu ersetzenden Textes wird eingegeben.</td>
</tr>
<tr>
<td><strong>REPT</strong> (WIEDERHOLEN)</td>
<td>Wiederholt einen Text die angegebenen Male.</td>
</tr>
<tr>
<td><strong>RIGHT</strong> (RECHTS)</td>
<td>Der Text wird von rechts aus in der angegebenen Zeichenzahl widergegeben.</td>
</tr>
<tr>
<td><strong>SUBSTITUTE</strong> (WECHSELN)</td>
<td>Ersetzt in einem vorgegebenen Text bestimmte Textteile durch andere Textteile. Zusätzlich kann angegeben werden, der wievielte Textteil ersetzt werden soll.</td>
</tr>
<tr>
<td>Funktion</td>
<td>Beschreibung</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>T</strong></td>
<td>Gibt den Text wieder oder einen leeren Textwert, wenn es sich z. B. um eine Zahl handelt.</td>
</tr>
<tr>
<td><strong>TEXT</strong></td>
<td>Konvertierung von Zahlen oder Uhrzeiten in Text.</td>
</tr>
<tr>
<td><strong>TRIM</strong></td>
<td>Entfernt führende Leerzeichen, Leerzeichen am Textende und reduziert mehrere Leerzeichen hintereinander im Text auf ein Leerzeichen.</td>
</tr>
<tr>
<td><strong>UNICHAR</strong></td>
<td>Wandelt eine Unicode-Nummer in Dezimalschreibweise in ein Unicode-Zeichen um. 196 wird zu 'Ä' ('Ä' hat den Hexadezimalwert 00C4, daraus wird bei der Dezimalschreibweise 196 ohne führende Nullen)</td>
</tr>
<tr>
<td><strong>UNICODE</strong></td>
<td>Wandelt eine Unicode-Zeichen in eine Unicode-Nummer in Dezimalschreibweise um. 'Ä' wird zu 196</td>
</tr>
<tr>
<td><strong>UPPER</strong></td>
<td>Gibt den Text in Großbuchstaben wieder.</td>
</tr>
</tbody>
</table>

**Informationsfunktionen**

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHOOSE</strong></td>
<td>Zuerst wird ein Index angegeben, danach eine Liste von Werten, aus der über den Index der entsprechende Wert ausgesucht wird. <strong>CHOOSE(2;&quot;Apfel&quot;;&quot;Birne&quot;;&quot;Banane&quot;)</strong> gibt Birne wieder. <strong>CHOOSE([Altersstufenfeld];&quot;Milch&quot;;&quot;Cola&quot;;&quot;Bier&quot;)</strong> gibt je nach «Altersstufenfeld» wieder, welches Getränk möglich ist.</td>
</tr>
<tr>
<td><strong>COUNT</strong></td>
<td>Nur die Felder werden gezählt, die eine Zahl oder einen Datums-bzw. Zeitwert enthalten. <strong>COUNT([Zeit];[Nummer])</strong> gibt 2 aus, wenn beide Felder einen Wert enthalten, also nicht NULL sind, ansonsten 1 oder schließlich 0.</td>
</tr>
<tr>
<td><strong>COUNTA</strong></td>
<td>Berücksichtigt auch die Felder, die einen Text beinhalten. Auch NULL wird gezählt, ebenso boolesche Felder.</td>
</tr>
<tr>
<td><strong>COUNTBLANK</strong></td>
<td>Zählt die leeren Felder in einem Bereich.</td>
</tr>
<tr>
<td><strong>HASCHANGED</strong></td>
<td>Überprüft, ob mit ihrem Namen angegebene Spalte sich geändert hat. Allerdings werden keine Spaltenangaben übernommen.</td>
</tr>
<tr>
<td><strong>INDEX</strong></td>
<td>Arbeitet mit Bereichen (ab LO 3.5)</td>
</tr>
<tr>
<td><strong>ISBLANK</strong></td>
<td>Prüft, ob das Feld NULL (leer) ist.</td>
</tr>
<tr>
<td><strong>ISERR</strong></td>
<td>Gibt TRUE zurück, wenn die Eingabe fehlerhaft ist, aber nicht vom Typ NA <strong>ISERR(1/0)</strong> ergibt TRUE</td>
</tr>
<tr>
<td><strong>ISERROR</strong></td>
<td>Wie ISERR, nur dass auch NA als Meldung TRUE ergibt.</td>
</tr>
<tr>
<td>Funktion</td>
<td>Beschreibung</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ISEVEN (ISTGERADE)</td>
<td>Prüft, ob es sich um einen gerade Zahl handelt.</td>
</tr>
</tbody>
</table>
| ISLOGICAL (ISTLOG)   | Prüft, ob es sich um einen Ja/Nein-Wert handelt.  
|                   | ISLOGICAL(TRUE()) oder ISLOGICAL(FALSE()) ergeben TRUE, Textwerte wie ISLOGICAL("TRUE") ergeben FALSE. |
| ISNA (ISTNV)         | Prüft, ob der Ausdruck vom Fehlertyp NA ist.                               |
| ISNONTEXT (ISTKTEXT) | Prüft, ob es sich bei dem Wert nicht um einen Text handelt.                |
| ISNUMBER (ISTZAHL)   | Prüft, ob es sich um eine Zahl handelt.  
|                   | ISNUMBER(1) ergibt TRUE, ISNUMBER("1") ergibt FALSE                        |
| ISODD (ISTUNGERADE)  | Prüft, ob es sich um eine ungerade Zahl handelt.                           |
| ISREF (ISTBEZUG)     | Prüft, ob es sich um einen Bezug handelt.  
|                   | ISREF([Feldname]) ergibt TRUE, ISREF(1) ergibt FALSE                       |
| ISTEXT (ISTTEXT)     | Prüft, ob es sich bei dem Wert um einen Text handelt.                      |
| NA (NV)              | Gibt den Fehlercode NA wieder.                                              |
| VALUE               | (ab LO 3.5)                                                                  |

**Benutzerdefiniert**

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSVARRAY</td>
<td>Wandelt einen CSV-Text in ein Array um. (ab LO 3.5)</td>
</tr>
<tr>
<td>CSVTEXT</td>
<td>Wandelt ein Array in einen CSV-Text um. (ab LO 3.5)</td>
</tr>
<tr>
<td>NORMALIZEARRAY</td>
<td>(ab LO 3.5)</td>
</tr>
<tr>
<td>NULL</td>
<td>Gibt NULL wieder</td>
</tr>
</tbody>
</table>

**Dokumenteninformationen**

<table>
<thead>
<tr>
<th>Eigenschaft</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOR</td>
<td>Autor, wird ausgelesen aus Extras → Optionen → LibreOffice → Benutzerdaten. Der eigentliche Autor wird hier also nicht wiedergegeben, sondern der aktuelle Nutzer der Datenbank.</td>
</tr>
<tr>
<td>TITLE</td>
<td>Gibt den Titel des Berichts wieder.</td>
</tr>
</tbody>
</table>

**Benutzerdefinierte Funktionen**

Benutzerdefinierte Funktionen können z. B. dazu dienen, bestimmte Zwischenergebnisse nach einer Gruppe von Datensätzen wieder zu geben. Im obigen Beispiel ist so etwas bei der Berechnung der Gebühr im Bereich «Leser_Name_Fuß» erstellt worden.
Im Bericht-Navigator ist für die Gruppe «Leser_Name» eine Funktion verzeichnet. Durch Rechtsklick auf Funktionen können zusätzliche Funktionen hier mit Namen definiert werden.


«Unterberichte einbinden» scheint momentan keine Funktion zu haben, es sei denn, dass Diagramme als Unterberichte verstanden werden.

Ist für die Funktion «Vorausberechnung» aktiviert, so kann das Ergebnis auch im Gruppenkopf stehen. Ohne die Aktivierung steht im Gruppenkopf nur der entsprechende Wert des abgefragten ersten Feldes der Gruppe.

Selbstdefinierte Funktionen können auch auf selbstdefinierte Funktionen zurückgreifen. Dabei ist dann aber zu beachten, dass die genutzten Funktionen vorher ausgeführt sein müssen. Die Vorausberechnung in dieser Funktion, die auf andere Funktionen zurückgreift, muss ausgeschaltet sein.

\[
\frac{\text{SummeZensurKlasse}}{\left(\text{[ZählerKlasse]}+1\right)}
\]

bezieht sich auf die Gruppe «Klasse». Der Inhalt des Feldes «Zensur» wird aufaddiert, die Anzahl der Datensätze wird ermittelt. Die Summe aus Zensur wird durch die Anzahl der Datensätze divi-

**Formeleingabe für ein Feld**

Über den Weg Daten → Datenfeld können Formeln eingegeben werden, die nur ein einziges Feld im Bereich «Detail» betreffen.

\[
\text{IF([boolschesFeld];"ja";"nein")}
\]
schreibt, dort eingegeben, statt WAHR und FALSCH einfach "ja" und "nein".

Es kann passieren, dass in einem Feld mit einer Formeleingabe grundsätzlich eine Zahl erscheint. Bei Text ist das dann eine «0». Hier muss nachgebessert werden, indem für das Textfeld vom Standardformat «Zahl» zum Format «Text» gewechselt wird.

**Bedingte Anzeige**


\[
\text{[Feldbezeichnung]="true"}
\]
sorgt dafür, dass der Inhalt nur dann angezeigt wird, wenn der Inhalt des Feldes «Feldbezeichnung» wahr ist. Dies funktioniert auch bei grafischen Steuerelementen.

Grafische Formen reagieren zur Zeit nicht auf Einträge in der bedingten Anzeige. Soll z. B. eine farbige Trennlinie nach dem 10. Platz einer Wettkampfliste eingezogen werden, so geht dies nicht, indem der grafischen Form über die bedingte Anzeige mitgegeben wird

\[
[\text{Platz}]=10
\]

Dieser Befehl wirkt nicht auf die Grafik. Sie erscheint in dem Abschnitt Detail dann weiter nach jedem Datensatz.

Dieser Bug ist als Bug 73707 gemeldet.


Hier lassen sich dann auch neben den Formen aus Einfügen → Formen → Standardformen z. B. die horizontalen Linien mit dem Gruppenfuß ausblenden.

9 Siehe hierzu die Datenbank «Beispiel_Bericht_bedingte_Einblendung_von_Grafiken.odb», die diesem Handbuch beiliegt.

---

*Berichte: Funktionen im Report-Designer*
Bedingte Formatierung
Bei der bedingten Formatierung kann z. B. ein Kalender so formatiert werden, dass die Wochenenden besonders gekennzeichnet werden. Unter **Format → bedingte Formatierung** ist dann einzutragen

\[ \text{WEEKDAY([Datumsfeld])}=1 \]

sowie die entsprechende Formatierung für den Sonntag.


**Beispiele für Berichte mit dem Report-Designer**

Der Report-Designer birgt einige Tücken in der Anwendung, da bestimmte Funktionen zwar grundsätzlich vorgesehen sind, aber zur Zeit nicht richtig funktionieren. Auch gibt es eine nur sehr

Rechnungserstellung

Folgende Vorgaben sollen für die Erstellung der Rechnung\textsuperscript{10} erfüllt werden:

- Die einzelnen Rechnungspositionen sollen durchnummeriert werden.
- Rechnungen, die mehr als eine Seite umfassen, sollen mit einer Seitennummerierung versehen werden.
- Rechnungen, die mehr als eine Seite umfassen, sollen auf jeder Seite eine Zwischensumme bilden und auf der darauffolgenden Seite diese Zwischensumme als Übertrag darstellen.

Mehrere aktuelle existierende Bugs scheinen das Verfahren unmöglich zu machen:

- \textit{Bug 51452}: Wird eine Gruppierung auf «Bereich wiederholen» eingestellt, so wird automatisch vor und hinter der Gruppierung ein Seitenumbruch eingefügt.
- \textit{Bug 51453}: Gruppierungen mit neu startender Seitenzählung sind zwar eigentlich vorgesehen, funktionieren aber nicht.
- \textit{Bug 51959}: Ein Gruppenfuß lässt sich nicht wiederholen. Er kann nur am Ende einer Gruppe auftauchen, nicht z. B. am Ende einer jeden Seite. Wird er auf «Bereich wiederholen» gesetzt, so verschwindet er komplett.

Außerdem gibt es noch Schwierigkeiten, Linien in den Bericht einzufügen. Die vorgesehenen horizontalen und vertikalen Linien werden erst ab den Versionen LO 4.0.5 bzw. 4.1.1 angezeigt. Als Ersatz dafür können Rechtecke genutzt werden. Die lassen sich aber nicht richtig positionieren, wenn in einem Bereich ein Seitenumbruch stattfindet.

Der Bericht in einer einfachen Form sollte also so aussehen:

\footnotesize
\begin{itemize}
\item Die Datenbank «Beispiel_Bericht_Rechnung.odb» ist den Beispieldatenbanken für dieses Handbuch beigefügt. Eine komplette Beschreibung der Datenbank ist «Base_Beispiele.pdf» verfügbar.
\end{itemize}

\textit{Berichte: Beispiele für Berichte mit dem Report-Designer}
Um trotz der oben genannten Einschränkungen eine den Anforderungen entsprechende Rechnungserstellung durchführen zu können muss genau mit den Seitenmaßen des auszudruckenden Dokumentes gearbeitet werden. In diesem Beispiel wird von einem DIN-A4-Ausgabeformat ausgegangen. Die Gesamthöhe eines Blattes beträgt also 29,7 cm.

Der Bericht muss dazu in mehrere Gruppen unterteilt werden. Zwei Gruppen beziehen sich auf das gleiche Tabellenfeld und enthalten jeweils nur einen entsprechenden Tabellenwert.
Die folgende Tabelle zeigt die Einteilung der Seite in unterschiedliche Bereiche des Berichtes:

<table>
<thead>
<tr>
<th>Bereich</th>
<th>Beschreibung</th>
<th>Abstand (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Seitenrand oben</td>
<td>2,00</td>
</tr>
<tr>
<td>B</td>
<td>Seitenkopf (erscheint auf jeder Seite, enthält keine Eingaben der Datenbank, sondern z. B. nur das Firmenlogo sowie die Adresse des Absenders)</td>
<td>3,00</td>
</tr>
<tr>
<td>C</td>
<td>Gruppenkopf für die Rechnungsnummer (Nur die Beträge, die zu einer Rechnungsnummer gehören, sollen später auch addiiert werden. Der Gruppenkopf erscheint nur zum Beginn der Rechnung.)</td>
<td>2,50</td>
</tr>
<tr>
<td>D</td>
<td>Gruppenkopf für die Rechnungsposten (Der Bereich «Detail» wird für andere Inhalte benötigt. Deshalb erfolgt hier eine Gruppierung, die gleichzeitig die Sortierung der Rechnungs- Inhalte z. B. nach der Eingabe vornimmt. Diese Gruppe besteht immer nur aus einem Wert.)</td>
<td>0,70</td>
</tr>
<tr>
<td>E</td>
<td>Gruppenkopf, ebenfalls an Rechnungsposten gebunden (Dieser Bereich wird nur dann angezeigt, wenn so viele Rechnungsposten vorkommen, dass auf jeden Fall eine weitere Rechnungsseite erforderlich ist. Er enthält die Zwischensumme und die Seitennummerierung unten auf der Seite. Nach diesem Bereich erfolgt ein Seitenumbruch.)</td>
<td>2,00</td>
</tr>
<tr>
<td>F</td>
<td>Detail-Bereich (Dieser Bereich wird nur dann angezeigt, wenn so viele Rechnungsposten vorkommen, dass auf jeden Fall eine weitere Rechnungsseite erforderlich ist. Er enthält den Übertrag, die Seitennummerierung oben auf der Seite)</td>
<td>2,50</td>
</tr>
<tr>
<td>G</td>
<td>Gruppenfuß für die Rechnungsnummer (Hier erfolgt die Ausgabe des Rechnungsbetrages, ggf. auch mit Angabe der Mehrwertsteuer. Der Gruppenfuß erscheint nur zum Schluss der Rechnung.)</td>
<td>1,60</td>
</tr>
<tr>
<td>H</td>
<td>Seitenfuß (z. B. Angabe der Kontoverbindung)</td>
<td>1,00</td>
</tr>
<tr>
<td>I</td>
<td>Seitenrand</td>
<td>1,00</td>
</tr>
</tbody>
</table>
Ein Seitenumbruch soll nur dann erfolgen, wenn zu viele Rechnungsposten vorhanden sind. Für die Rechnungsposten ergibt sich der folgende freie Platz:

<table>
<thead>
<tr>
<th>Platz</th>
<th>Breite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos. A</td>
<td>2,00 cm</td>
</tr>
<tr>
<td>Pos. B</td>
<td>3,00 cm</td>
</tr>
<tr>
<td>Pos. C</td>
<td>2,50 cm</td>
</tr>
<tr>
<td>Pos. G</td>
<td>1,60 cm</td>
</tr>
<tr>
<td>Pos. H</td>
<td>1,00 cm</td>
</tr>
<tr>
<td>Pos. I</td>
<td>1,00 cm</td>
</tr>
</tbody>
</table>

\[
29,70 \text{ cm (DIN A 4)} - 2,00 \text{ cm (Pos. A)} - 3,00 \text{ cm (Pos. B)} - 2,50 \text{ cm (Pos. C)} - 1,60 \text{ cm (Pos. G)} - 1,00 \text{ cm (Pos. H)} - 1,00 \text{ cm (Pos. I)} = 18,60 \text{ cm}
\]

Der verbleibende freie Platz kann also maximal 18,60 cm / 0,70 cm = 26,57..., abgerundet also 26 Rechnungspositionen erfassen.

Sobald die 27. Rechnungsposition vorkommt muss auf jeden Fall ein Seitenumbruch erfolgen. Das bedeutet, dass dann auf jeden Fall Gruppenkopf E und der Detail-Bereich angezeigt werden müssen. Es muss also ein Zähler für die Rechnungsnummer (Bereich C) eingerichtet werden. Erreicht dieser Zähler die Nummer 27, so wird der Detail-Bereich (F) angezeigt.

Der Zähler der Rechnungsnummer wird wie folgt definiert:

\[
\text{AND(}[\text{ZählerRechnungsnummer}]=26;[\text{ZählerKomplett}]>26)
\]


Der Gruppenkopf E und der Detailbereich F werden angezeigt, wenn insgesamt mehr als 26 Positionen in der Rechnung stehen und die momentane Rechnungsposition 26 erreicht hat. Der Ausdruck für die bedingte Anzeige ist also in beiden Bereichen

\[
\text{AND(}[\text{ZählerRechnungsnummer}]=26;[\text{ZählerKomplett}]>26)
\]

Jetzt ist zu berechnen, wie groß der Anteil des Bereiches ist, der noch auf der ersten Seite angezeigt wird:

| 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 |

Der Gruppenfuß fällt aus der ersten Seite raus, insgesamt 26 Rechnungsposten sind enthalten. Der Gruppenkopf E kann also maximal 2 cm auf der ersten Seite anzeigen. In diesem 2 cm muss auf jeden Fall die Zwischensumme und die Seitenzahl untergebracht werden. Um auf jeden Fall einen korrekten Seitenumbruch zu erzeugen, sollte der Bereich also etwas kleiner sein. In dem Beispiel wurde er auf 1,90 cm eingestellt.

Der Bereich «Detail» wird auf der Folgeseite oben angezeigt. Da auf der Folgeseite der Gruppenkopf der Rechnungsnummer (C) nicht mehr erscheint, kann hier der Bereich «Detail» so viel Platz einnehmen wie der Gruppenkopf, nämlich 2,50 cm. Dann starten die darauffolgenden Rechnungsposten wieder mit der gleichen Einstellung wie auf der Vorseite.


Die Seitennummerierung wird ähnlich abgefragt wie die Anzeige des Gruppenkopfes E des Bereiches «Detail».

IF([ZählerRechnungsnummer]=26;"Seite 1";"")

Hiermit wird die Seitennummer auf der ersten Seite erstellt. Weitere IF-Abfragen können dann für die weiteren Seitennummern eingebunden werden.

Die Seitennummer für die Folgeseite wird bei der gleichen Bedingung einfach auf "Seite 2" gesetzt. Werden die Formeln entsprechend weitgehender formuliert, so können beliebig viele Seiten des Berichtes abgedeckt werden.

Der Ausdruck für die bedingte Anzeige wechselt also von

AND([ZählerRechnungsnummer]=26;[ZählerKomplett]>26)

zu

AND(MOD([ZählerRechnungsnummer];26)=0;[ZählerKomplett]>[ZählerRechnungsnummer])

Berichte: Beispiele für Berichte mit dem Report-Designer

306
Der Gruppenkopf E und der Detailbereich F erscheinen also nur, wenn sich aus der Division des Zählers der Rechnungsnummer mit 26 kein Rest ergibt und die Komplettzahl der Rechnungsposten größer ist als der Zähler der Rechnungsnummer.

Der Ausdruck für die Seitenzahl wechselt von

IF([ZählerRechnungsnummer]=26;"Seite 1";"")

zu

"Seite "&[ZählerRechnungsnummer]/26 für die aktuelle Seite bzw.

"Seite "&([ZählerRechnungsnummer]/26)+1 als Anzeige für die Folgeseite.

Der folgende Berichtsausdruck ist mit diesen Einstellungen noch nicht zu bewerkstelligen:

Durch das Adressfeld sind auf der ersten Seite der Rechnung weniger Rechnungsposten enthalten als auf der zweiten Seite. Der Bereich «Detail», der auf der zweiten Seite ganz oben abgebildet ist, ist also deutlich kleiner als der Gruppenkopf für die Rechnungsnummer (C).

Um unterschiedliche Mengen an Rechnungsposten auf der ersten Seite und den Folgeseiten zuzulassen müssen also die Formeln noch einmal angepasst werden.

Die folgende Berechnung sichert ab, dass die entsprechenden Bereiche korrekt angezeigt werden.

\[ \text{AND} \left( \text{MOD}([\text{ZählerRechnungsnummer}]-20;24)=0; \right. \]
\[ \left. [\text{ZählerRechnungKomplett}]> [\text{ZählerRechnungsnummer}] \right) \]

**Berichte: Beispiele für Berichte mit dem Report-Designer** 307
Vom Zähler für die Rechnungsnummer wird die Anzahl der Rechnungsposten auf der ersten Seite subtrahiert. Diese Differenz wird durch die mögliche Anzahl der Rechnungsposten auf der zweiten Seite dividiert. Wenn die Division glatt auf geht (kein Rest, MOD = 0), dann ist die erste Bedingung für die Anzeige von Gruppenkopf E und der Detailbereich F gegeben. Außerdem muss, wie bereits vorher definiert, der Zähler für die Rechnungsnummer kleiner sein als die insgesamt zu erwartenden Rechnungsposten. Sonst würde ja die Rechnungssumme noch genug Platz auf der entsprechenden Seite haben.

Die mögliche Anzahl der Rechnungsposten auf der zweiten Seite wurde dadurch geringer, dass jetzt auf der zweiten Seite zusätzlich die Rechnungsnummer und das Datum enthalten sind.

Die Seitenzahl wird jetzt etwas einfacher berechnet:

```
"Seite " & INT([ZählerRechnungsnummer]/24)+1
```

INT rundet auf die nächste Ganzzahl ab. Die erste Seite enthält maximal 20 Rechnungsposten. Die Division ergibt für die erste Seite also ein Ergebnis < 1. Abgerundet wird das zu 0. Also muss zu der ermittelten Seitenzahl 1 addiert werden, damit auf der ersten Seite auch 1 erscheint. Entsprechend muss für die zweite Seite 2 addiert werden.

Der Bericht weist in der obigen Form noch einen Schönheitsfehler auf. Ein genauer Blick auf die Rechnungsposten zeigt, dass sich die unteren drei Rechnungsposten gleichen. Dies wurde hier durch einfaches Kopieren der Datensätze erzeugt. Es handelt sich also nicht um die gleichen Datensätze, sondern unterschiedliche Rechnungspositionen, die nacheinander über die Software abgearbeitet wurden. Besser wäre hier, die Waren entsprechend gruppiert anzuzeigen, damit die gleiche Ware nicht mehrmals, sondern nur einmal mit entsprechender Anzahl aufgelistet wird.


Dann ergibt sich die folgende Rechnungsübersicht:

**Ausdruck von Berichten zum aktuellen Datensatz des Formulars**

Gerade bei der Rechnungserstellung wie in dem vorhergehenden Beispiel kann es sinnvoll sein, nach jeder Eingabe der Rechnungsposten einen entsprechenden Ausdruck anzufertigen. In einem Formular soll der Inhalt für die Rechnung bestimmt werden und schließlich der Ausdruck des Einzeldokumentes erfolgen.

Berichte lassen sich nicht über Makros mit einem Filter starten. Allerdings kann die Abfrage, die als Grundlage für den Bericht genutzt wird, vorher gefiltert werden. Dies geht entweder über die Form einer Parameterabfrage

```sql
SELECT * FROM "Rechnung" WHERE "ID" = :ID
```

oder über eine Abfrage, die über eine einzeilige Filtertabelle mit Daten versorgt wird:

```sql
SELECT * FROM "Rechnung" WHERE "ID" = (SELECT "Integer" FROM "Filter" WHERE "ID" = TRUE)
```

Bei der Parameterabfrage muss der Inhalt in ein entsprechendes Dialogfeld nach dem Start des Berichtes eingegeben werden.

Bei der Steuerung über eine Filtertabelle wird der Inhalt der Filtertabelle per Makro geschrieben. Eine separate Eingabe ist also nicht mehr nötig. Diese Möglichkeit ist also für den Nutzer einfacher zu handhaben und soll deshalb im Folgenden beschrieben werden.
**Aufbau der Filtertabelle**


<table>
<thead>
<tr>
<th>Feldname</th>
<th>Feldtyp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Ja/Nein [BOOLEAN]</td>
</tr>
<tr>
<td>Integer</td>
<td>Integer [INTEGER]</td>
</tr>
</tbody>
</table>

Diese Tabelle wird zum Start mit einem Datensatz gefüllt. Hierzu muss lediglich das Feld "ID" einmal markiert werden, so dass es den Wert «Ja» (oder in SQL «TRUE») hat.

**Aufbau des Makros zum Start des gefilterten Berichtes**

Das Formular muss für das Aufrufen eines einzelnen Berichtes an irgendeiner Stelle den Primärschlüssel der Tabelle "Rechnung" enthalten. Dieser Primärschlüsselwert wird ausgelesen und über ein Makro in die Tabelle "Filter" übertragen. Anschließend wird in dem Makro noch der gewünschte Bericht gestartet.

```
SUB Filtern_und_Drucken
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oDatenquelle AS OBJECT
    DIM oVerbindung AS OBJECT
    DIM oSQL_Anweisung AS OBJECT
    DIM strSQL AS STRING
    oDoc = thisComponent
    oDrawpage = oDoc.Drawpage
    oForm = oDrawpage.Forms.getByName("MainForm")
    oFeld = oForm.getByName("fmtID")
    oDatenquelle = ThisComponent.Parent.CurrentController
    If NOT (oDatenquelle.isConnected()) THEN
        oDatenquelle.connect()
    END IF
    oVerbindung = oDatenquelle.ActiveConnection()
    oSQL_Anweisung = oVerbindung.createStatement()
    strSQL = "UPDATE ""Filter"" SET ""Integer"" = ""+oFeld.GetCurrentValue()+"" WHERE ""ID"" = TRUE"
    oSQL_Anweisung.executeUpdate(strSql)
    ThisDatabaseDocument.ReportDocuments.getByName("Rechnung").open
END SUB
```


Berichte: Beispiele für Berichte mit dem Report-Designer 310
... WHERE "Rechnung_ID" = IFNULL((SELECT "Integer" FROM "Filter" WHERE "ID" = TRUE),"Rechnung_ID") ...

Es wird das Feld "Integer" ausgelesen. Falls dieses Feld keinen Wert enthält wird stattdessen "Rechnung_ID" = "Rechnung_ID" gesetzt. Das bedeutet, dass alle Datensätze angezeigt werden – nicht nur der Filterdatensatz. So können also gegebenenfalls mit der gleichen Ansicht auch alle gespeicherten Rechnungen ausgedruckt werden.

**Wechselnde Einfärbung von Zeilen**

Tabellenzeilen in einem Bericht können manchmal beim Lesen dazu führen, dass der Blick um eine Zeile nach unten oder oben rutscht. Dagegen kann eine Einfärbung von mindestens einer Zeile helfen. Im folgenden Beispiel werden die Zeilen einfach wechselseitig mit unterschiedlichen Farben eingefärbt. Der Bericht sieht dann wie folgt aus:

<table>
<thead>
<tr>
<th>Geburtsdatenliste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honne 13.04.70</td>
</tr>
<tr>
<td>Meike 13.04.71</td>
</tr>
<tr>
<td>Egon 01.07.67</td>
</tr>
<tr>
<td>Karl 01.07.67</td>
</tr>
<tr>
<td>Susanne 02.08.65</td>
</tr>
<tr>
<td>Erkan 17.12.75</td>
</tr>
</tbody>
</table>

Grundlage des Berichtes ist eine Abfrage mit Namen und Datumseinträgen. Es wurde aus der ursprünglichen Tabelle eine Abfrage erstellt, damit die Daten nach Monaten und Tagen sortiert sind, also die Geburtsdatenfolge im Jahr anzeigen. Dies geht über ...

... ORDER BY MONTH("Geburtstag") ASC, DAY("Geburtstag") ASC

Um eine wechselseitige Einfärbung zu erreichen muss irgendeine Funktion erstellt werden, aus der heraus über eine Wert später die Bedingung für die Einfärbung abgeleitet werden kann. Hier wird ein Textfeld im Bericht aufgezogen und über die Eigenschaften → Daten → Datenfeld-Typ ein Zähler definiert.

Der Name der Funktion wird für die bedingte Formatierung benötigt. Der Zähler selbst braucht beim Ausdruck nicht zu erscheinen. Der Name kann direkt aus dem aufgezogenen Feld abgelesen werden. Ist das Feld wieder gelöscht, so ist der Funktionsname weiterhin über Ansicht → Berichts-Navigator nachschlagbar.

Die Datenbank «Beispiel_Bericht_Zeilen_Farbwechsel_Spalten.odb» zu diesem Bericht ist den Beispiel-datenbanken für dieses Handbuch beigefügt.

Berichte: Beispiele für Berichte mit dem Report-Designer 311
Jetzt muss jedem der Textfelder über den Zähler ein entsprechendes Format zugewiesen werden. Die Bedingung ist jedes Mal ein Ausdruck, der nicht direkt mit dem Feld zusammen hängt. Deshalb wird als **Bedingung 1** → *Ausdruck ist* → \( \text{MOD([ZählerBericht];2)}>0 \) eingestellt. \( \text{MOD} \) berechnet den Rest einer Division. Bei allen ungeraden Zahlen ist der Rest größer als 0, bei allen Geraden Zahlen ist der Rest 0. Zeile 1, 3, 5 usw. wird also mit dem entsprechenden Format versehen.


Da die bedingte Formatierung alle Formate der Standardformatierung überschreibt muss z.B. eine Ausrichtung der Schrift in den Zeicheneinstellungen der bedingten Formatierung erfolgen:
Hier wird die Schrift vertikal mittig in dem eingefärbten Textfeld dargestellt. Horizontal lassen sich die üblichen Ausrichtungen einstellen, nicht aber eine Einrückung, so dass die Buchstaben nicht am linken Rand des Textfeldes stehen bleiben.


Zielführender ist es dagegen, einfach ein Textfeld vor dem eigentlichen Text zu positionieren, das aber nicht mit irgendeinem Datenfeld-Typ verbunden wird. Das Textfeld wird genau wie die anderen enthaltenen Felder mit der entsprechenden bedingten Formatierung versehen, so dass eine einheitlich breite (scheinbare) Einrückung beim Ausdruck erfolgt.

**Zweispaltige Berichte**

Mit geschickter Abfragetechnik ist es möglich, einen Bericht mit mehreren Spalten zu erstellen, bei dem die folgenden Spalten auch die folgenden Datensätze darstellen:

<table>
<thead>
<tr>
<th>Name</th>
<th>Geburtsdatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kathen</td>
<td>17.01.84</td>
</tr>
<tr>
<td>Mick</td>
<td>03.03.53</td>
</tr>
<tr>
<td>Meike</td>
<td>13.04.71</td>
</tr>
<tr>
<td>Monika</td>
<td>05.06.85</td>
</tr>
<tr>
<td>Paul</td>
<td>11.07.69</td>
</tr>
<tr>
<td>Susanne</td>
<td>02.08.65</td>
</tr>
<tr>
<td>Ysabelle</td>
<td>17.09.89</td>
</tr>
<tr>
<td>John</td>
<td>19.11.07</td>
</tr>
<tr>
<td>Johann</td>
<td>23.12.91</td>
</tr>
<tr>
<td>Sally</td>
<td>15.02.01</td>
</tr>
<tr>
<td>Hanne</td>
<td>13.04.70</td>
</tr>
<tr>
<td>Lara</td>
<td>23.04.85</td>
</tr>
<tr>
<td>Karl</td>
<td>01.07.67</td>
</tr>
<tr>
<td>Egon</td>
<td>23.07.67</td>
</tr>
<tr>
<td>Georg</td>
<td>28.08.95</td>
</tr>
<tr>
<td>Maik</td>
<td>28.10.93</td>
</tr>
<tr>
<td>Erkan</td>
<td>17.12.75</td>
</tr>
</tbody>
</table>

Der erste Datensatz wird in der linken Spalte aufgeführt, der zweite Datensatz in der rechten Spalte. Die Daten sind nach der Lage des Geburtstages im Jahr sortiert.


Grundlage für diesen Bericht ist die folgende Abfrage:
SELECT "T1"."Name" AS "Name1", "T1"."Geburtsdatum" AS "Geburtsdatum1", "T2"."Name" AS "Name2", "T2"."Geburtsdatum" AS "Geburtsdatum2" FROM (SELECT "Name", "Geburtsdatum", "ZeilenNr" AS "Zeile" FROM (SELECT "a".*,
(SELECT COUNT( "ID" ) FROM "Geburtstage" WHERE
RIGHT( '0' || MONTH( "Geburtsdatum" ), 2 ) ||
RIGHT( '0' || DAY( "Geburtsdatum" ), 2 ) || "ID"
<= RIGHT( '0' || MONTH( "a"."Geburtsdatum" ), 2 ) ||
RIGHT( '0' || DAY( "a"."Geburtsdatum" ), 2 ) || "a"."ID" )
AS "ZeilenNr"
FROM "Geburtstage" AS "a")
WHERE MOD( "ZeilenNr", 2 ) > 0 )
AS "T1"
LEFT JOIN (SELECT "Name", "Geburtsdatum", "ZeilenNr"-1 AS "Zeile" FROM (SELECT "a".*,
(SELECT COUNT( "ID" ) FROM "Geburtstage" WHERE
RIGHT( '0' || MONTH( "Geburtsdatum" ), 2 ) ||
RIGHT( '0' || DAY( "Geburtsdatum" ), 2 ) || "ID"
<= RIGHT( '0' || MONTH( "a"."Geburtsdatum" ), 2 ) ||
RIGHT( '0' || DAY( "a"."Geburtsdatum" ), 2 ) || "a"."ID" )
AS "ZeilenNr"
FROM "Geburtstage" AS "a")
WHERE MOD( "ZeilenNr", 2 ) = 0 )
AS "T2"
ON "T1"."Zeile" = "T2"."Zeile"
ORDER BY "T1"."Zeile"

Es werden zwei identische Unterabfragen in der Abfrage dargestellt. Die ersten beiden Spalten beziehen sich auf die Unterabfrage mit dem Alias "T1", die letzten beiden Spalten auf die Unterabfrage mit dem Alias "T2".

Die Unterabfragen stellen neben den Feldern der Tabelle "Geburtstage" noch Felder zur Verfügung, die eine Unterscheidung in Zeilen und damit eine Sortierung ermöglichen. Dazu wird im Kern die Abfragemöglichkeit zur Zeilennummerierung (siehe Kapitel «Abfragen») genutzt.

RIGHT( '0' || MONTH( "Geburtsdatum" ), 2 ) || RIGHT( '0' ||
DAY( "Geburtsdatum" ), 2 ) || "ID"
Diese Formulierung dient dazu, eine eindeutige Reihenfolge der Datensätze zu gewährleisten. Da die Beispieldatensätze nach dem Datum sortiert werden sollen, ließe sich ja leicht sagen, dass lediglich das Datum zum Vergleich herangezogen werden sollte. Etwas schwieriger ist es allerdings schon dadurch, dass nicht das Geburtsdatum, sondern die Lage des Datums im Jahr maßgebend sein soll. Zusätzliche Probleme bereiten schließlich gleiche Datumswege, die eine eindeutige Reihenfolge verhindern. Deshalb wird zur Sortierung neben dem Monat und dem Tag auch noch der Primärschlüssel der Tabelle herangezogen, der schließlich eindeutig ist. Damit nicht der Monat '10' vor den Monat '2' gesetzt wird, wird bei der Zusammenführung des Sortierkriteriums in einen Text über || vor jede Monatszahl eine führende '0' positioniert, die anschließend bei zweistelligen Monatszahlen über RIGHT( ... , 2 ) wieder entfernt wird.

Mit SELECT COUNT( "ID" ) wird die Anzahl der Datensätze ermittelt, deren Kombination aus Monat, Tag und Primärschlüssel kleiner oder gleich der entsprechenden Kombination des aktuellen Datensatzes der Tabelle "Geburtstage" ist. Hier handelt es sich um eine korrelierende Unterabfrage (siehe Kapitel «Abfragen»).

Über MOD( "ZeilenNr", 2 ) wird aus den so ermittelten Zeilennummerierungen ermittelt, ob es sich um eine gerade oder ungerade Zahl handelt. MOD ermittelt den Rest der Division, in dem
genannten Beispiel der Division durch 2. Dadurch gibt die "ZeilenNr" wechselweise die Nummern '1' und '0' aus. Hierdurch werden die Abfragen für "T1" und "T2" unterschieden.

In der nächsthöheren Abfrageebene von "T2" wird als "Zeile" "ZeilenNr"-1 definiert. Dadurch werden "T1" und "T2" direkt vergleichbar.

"T1" wird mit "T2" über LEFT JOIN verbunden, damit auch bei ungerader Datensatzzahl der Tabelle "Geburtstage" alle Daten dargestellt werden. In der Zusammenführung von "T1" und "T2" kann jetzt auf die Spalten direkt Bezug genommen werden: "T1"."Zeile" = "T2"."Zeile".

Zum Schluss wird der ganze Inhalt noch nach dem Wert aus "Zeile" sortiert, der für "T1" und "T2" gleich ist. Dies könnte auch der Bericht über die Gruppierung direkt übernehmen.

### Geburtstagsliste

<table>
<thead>
<tr>
<th>Januar</th>
<th>Februar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kathrin 17.01.84</td>
<td>Sally 15.02.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>März</th>
<th>April</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mick 03.03.53</td>
<td>Hanne 13.04.70</td>
</tr>
<tr>
<td>Meike 13.04.71</td>
<td>Lara 23.04.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mai</th>
<th>Juni</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Monika 05.06.86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Juli</th>
<th>August</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karl 01.07.67</td>
<td>Susanne 02.08.65</td>
</tr>
<tr>
<td>Paul 11.07.89</td>
<td>Georg 28.08.90</td>
</tr>
<tr>
<td>Egon 23.07.67</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>September</th>
<th>Oktober</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ysabelle 17.09.89</td>
<td>Maik 28.10.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>November</th>
<th>Dezember</th>
</tr>
</thead>
<tbody>
<tr>
<td>John 16.11.07</td>
<td>Ercan 17.12.75</td>
</tr>
<tr>
<td></td>
<td>Johann 23.12.91</td>
</tr>
</tbody>
</table>


Die folgende Ansicht "MonatZ" wird zuerst erstellt:

```
SELECT "a"."ID", "a"."Name", "a"."Geburtsdatum", MONTH( "a"."Geburtsdatum" ) AS "MonatZahl",
( SELECT COUNT( "ID" ) FROM "Geburtstage"
WHERE MONTH( "Geburtsdatum" ) = MONTH( "a"."Geburtsdatum" )
AND RIGHT( '0' || DAY( "Geburtsdatum" ), 2 ) || "ID" <=
RIGHT( '0' || DAY( "a"."Geburtsdatum" ), 2 ) || "a"."ID" )
AS "MonatZaehler"
FROM "Geburtstage" AS "a"
```

Alle Felder der Tabelle "Geburtstage" werden übernommen. Zusätzlich wird der Monat als Zahl dargestellt. Der Tabelle "Geburtstage" wird ein Alias "a" zugewiesen, damit auf die Tabelle mit einer korrelierenden Unterabfrage zugreifen werden kann.

Auf die Ansicht "MonatZ" greift die Ansicht "Bericht_Monat_zweispaltig" zu. Diese Ansicht wird hier nur in Ausschnitten wieder gegeben:

SELECT
  "Tab1"."Name" AS "Name1",
  "Tab1"."Geburtsdatum" AS "Geburtsdatum1",
  1 AS "MonatZahl1",
  IFNULL("Tab1"."MonatZaehler",999) AS "MonatZaehler1",
  'Januar' AS "Monat1",
  "Tab2"."Name" AS "Name2",
  "Tab2"."Geburtsdatum" AS "Geburtsdatum2",
  2 AS "MonatZahl2",
  IFNULL("Tab2"."MonatZaehler",999) AS "MonatZaehler2",
  'Februar' AS "Monat2"
FROM
  (SELECT * FROM "MonatZ" WHERE "MonatZahl" = 1) AS "Tab1"
RIGHT JOIN (SELECT * FROM "MonatZ" WHERE "MonatZahl" = 2) AS "Tab2"
on "Tab1"."MonatZaehler" = "Tab2"."MonatZaehler"
UNION
SELECT
  "Tab1"."Name" AS "Name1",
  "Tab1"."Geburtsdatum" AS "Geburtsdatum1",
  1 AS "MonatZahl1",
  IFNULL("Tab1"."MonatZaehler",999) AS "MonatZaehler1",
  'Januar' AS "Monat1",
  "Tab2"."Name" AS "Name2",
  "Tab2"."Geburtsdatum" AS "Geburtsdatum2",
  2 AS "MonatZahl2",
  IFNULL("Tab2"."MonatZaehler",999) AS "MonatZaehler2",
  'Februar' AS "Monat2"
FROM
  (SELECT * FROM "MonatZ" WHERE "MonatZahl" = 1) AS "Tab1"
LEFT JOIN (SELECT * FROM "MonatZ" WHERE "MonatZahl" = 2) AS "Tab2"
on "Tab1"."MonatZaehler" = "Tab2"."MonatZaehler"
UNION
...
ORDER BY "MonatZahl1", "MonatZahl2", "MonatZaehler1", "MonatZaehler2"

Zuerst werden in der Unterabfrage aus "MonatZ" alle Daten ausgelesen, bei denen die "MonatZahl" 1 ist. Diese Auswahl wird mit dem Alias "Tab1" versehen. Gleichzeitig werden mit dem Alias "Tab2" alle Daten ausgelesen, bei denen die "MonatZahl" 2 ist. Beide Tabellen werden durch einen RIGHT JOIN verbunden, so dass alle Datensätze aus "Tab2" und nur die Datensätze aus "Tab1" angezeigt werden, bei denen der "MonatZaehler" mit dem von "Tab2" übereinstimmt.

Die Spalten der Ansicht müssen unterschiedliche Bezeichnungen ausweisen, so dass jede Spalte mit einem Alias versehen ist. Außerdem wird direkt als Spaltenwert für "Tab1" eine 1 als "MonatZahl" sowie 'Januar' als "MonatZahl" eingegeben. Diese Einträge erscheinen auch, wenn es keinen Datensatz aus "Tab1", wohl aber noch Datensätze aus "Tab2" gibt. Ist kein "MonatZaehler" vorhanden, so soll dort der Wert 999 eingetragen werden. Da "Tab1" mit "Tab2" über einen RIGHT JOIN verbunden ist kann es ja vorkommen, dass bei weniger Datensätzen in "Tab1" stattdessen leere Felder angezeigt werden. Leere Felder würden aber bei einer späteren Sortierung vor allen Feldern mit Inhalt sortiert, so dass stattdessen ein besonders hoher Wert gewählt wurde.

Bei der Darstellung der Spalten für "Tab2" wird entsprechend vorgegangen. Hier könnte allerdings IFNULL("Tab2"."MonatZaehler",999) entfallen, da bei einem RIGHT JOIN zugunsten von "Tab2" zwar alle Zeilen aus "Tab2", nicht aber mehr Zeilen aus "Tab1" als aus "Tab2" dargestellt werden.
Genau dieses Problem wird mit der Verbindung zweier Abfragen gelöst. Mit **UNION** werden alle Datensätze aus der ersten Abfrage und alle Datensätze aus der zweiten Abfrage dargestellt. Die Datensätze aus der zweiten Abfrage erscheinen aber nur dann, wenn sie nicht identisch mit einem vorhergehenden Datensatz sind. **UNION** wirkt also wie **DISTINCT**.

Über **UNION** wird die gleiche Abfrage noch einmal gestellt, nur sind "Tab1" und "Tab2" jetzt mit einem **LEFT JOIN** verbunden. Damit erscheinen auf jeden Fall alle Datensätze aus "Tab1" auch wenn in "Tab2" weniger Datensätze vorhanden sind als in "Tab1".

Für die Monate 3 und 4, 5 und 6 usw. werden entsprechend angepasste Abfragen verwendet und wiederum mit **UNION** an die vorhergehenden Abfragen angehängt.


**Fehlerquellen bei Berichten**

Der Report-Builder birgt manchmal Fehler, die nicht so ohne weiteres nachvollzogen werden können. Hier einige Fehlerquellen und eventuell nützliche Gegenmaßnahmen.

**Der Inhalt eines Feldes aus einer Abfrage erscheint nicht**

Eine Datenbank soll den Verkauf von Waren simulieren. In einer Abfrage wird aus der Anzahl einer gekauften Ware und dem Einzelpreis der Gesamtpreis ermittelt.

```
SELECT "Verkauf"."Anzahl", "Ware"."Ware", "Ware"."Preis",
"Verkauf"."Anzahl"*"Ware"."Preis" FROM "Verkauf", "Ware"
WHERE "Verkauf"."Ware_ID" = "Ware"."ID"
```

Diese Abfrage dient als Grundlage des Berichtes. Wird aber das Feld "Verkauf"."Anzahl"*"Ware"."Preis" in dem Bericht aufgerufen, so bleibt es ohne Inhalt.

Wird dem Feld in der Abfrage hingegen ein Alias zugewiesen, so kann der Report-Builder darauf einwandfrei zugreifen:

```
SELECT "Verkauf"."Anzahl", "Ware"."Ware", "Ware"."Preis",
"Verkauf"."Anzahl"*"Ware"."Preis" AS "GPreis"
FROM "Verkauf", "Ware" WHERE "Verkauf"."Ware_ID" = "Ware"."ID"
```

Das Feld greift jetzt auf "Gpreis" zu und stellt den entsprechenden Wert dar.

**Ein Bericht lässt sich nicht ausführen**


Manchmal kommt es aber vor, dass sich ein Bericht zwar erstellen, aber anschließend nicht ausführen oder nicht einmal speichern lässt. Es erscheint eine Fehlermeldung, die erst einmal wenig aussagekräftig ist:

«Bericht konnte nicht ausgeführt werden. Eine Ausnahme vom Typ com.sun.star.lang.WrappedTargetException wurde entdeckt.»

Hier hilft es vielleicht, über den Berichtsnavigator die Eigenschaften des Berichtes aufzusuchen: **Daten → SQL-Befehl analysieren → Nein.** Leider hat diese Problemlösung zur Folge, dass eine einmal eingestellte Gruppierung nicht mehr funktioniert.

Datenbank-Anbindung
Allgemeines zur Datenbank-Anbindung


Anmeldung der Datenbank

Für viele Funktionen wie z.B. die Nutzung beim Etikettendruck, die Nutzung von Daten für Serienbriefe usw. ist die Anmeldung einer Datenbank in den Einstellungen von LibreOffice notwendig.

Über Extras → Optionen → LibreOffice Base → Datenbanken → Neu lässt sich eine Datenbank für den weitergehenden Gebrauch in anderen LibreOffice-Komponenten anmelden.

Die Datenbank wird über den Dateibrowser gesucht und auf ähnliche Weise mit LibreOffice verknüpft wie sonst bei einem einfachen Formular. Der Datenbank selbst wird allerdings ein eingängiger Name gegeben, der ohne weiteres anders lauten kann als der Name der eigentlichen Datenbankdatei. Die Namensbezeichnung ist so etwas wie ein Alias-Name, was ja auch bei Abfragen in Datenbanken vergeben werden kann.

Datenquellenbrowser

Auf der linken Seite im Datenquellen-Browser sind die angemeldeten Datenquellen zu sehen. Die Datenquelle "Bibliography" ist standardmäßig bei LibreOffice mit dabei. Die weiteren Datenquellen unterscheiden sich je nach angemeldetem Namen.

Durch einen Klick auf das + - Zeichen vor den Datenbankbezeichnungen öffnet sich die Datenbank und zeigt an, dass sich darin ein Unterordner für Abfragen und ein Unterordner für Tabellen befindet. Die weiteren Unterordner der Datenbank werden nicht zur Verfügung gestellt. Interne Formulare sowie Berichte sind also nur über die Datenbank selbst verfügbar.

Erst beim Klick auf den Ordner Tabellen erfolgt der tatsächliche Datenbankzugriff. Bei passwortgeschützten Datenbanken erfolgt zu diesem Zeitpunkt die Passwortabfrage.

Rechts von dem Verzeichnisbaum zeigt sich die angewählte Tabelle. Sie kann wie in Base selbst bearbeitet werden. Bei stärker ausgeprägten relationalen Datenbanken ist allerdings die Eingabe in die Tabelle mit Vorsicht zu bewerkstelligen, da die Tabellen ja über Fremdschlüssel miteinander verknüpft sind. Allein die unten abgebildete Datenbank zeigt eine separate Tabelle für die Straßennamen, eine für Postleitzahlen und noch eine für den Ort.

Für den richtigen Blick auf die Daten ohne entsprechende Editiermöglichkeit eignen sich daher Abfragen oder entsprechende Ansichten (Views) besser.

**Daten in Text**

Die Funktion «*Daten in Text*» steht zur Verfügung, sobald ein Datensatz markiert wurde.

Abbildung 49: Markierung eines Datensatzes

Wird jetzt «*Daten in Text*» angewählt, so erscheint ein Assistent, der die erforderlichen Formatierungen vornimmt:
Abbildung 50: Daten Einfügen als Tabelle

Für das Einfügen von Daten in den Text stehen die Möglichkeiten der Darstellung in Tabellenform, als einzelne Felder oder als Gesamttext zur Verfügung.


Sobald Tabellenspalten gewählt sind, wird der Button Eigenschaften für die Tabellen aktiviert. Hier können die üblichen Tabelleneigenschaften des Writer eingestellt werden (Breite der Tabelle, Spaltenbreite …).

Das Markierfeld gibt darüber Auskunft, ob eine Tabellenüberschrift gewünscht ist. Ist dieses Markierfeld nicht angewählt, so wird für die Überschrift keine separate Zeile eingefügt.

Die so eventuell gewählte Zeile für die Tabellenüberschrift kann entweder die Spaltennamen übernehmen oder nur die Zeile erstellen und den Platz für die Überschrift zum späteren Editieren schaffen.
Abbildung 51: Über «Autoformat» stehen Tabellenformate zur Auswahl


Um dem Autoformat-Vorschlag für Tabellen eine Tabelle hinzuzufügen muss eine Tabelle erstellt worden sein. Diese wird dann markiert und kann über den Button Hinzufügen in die Tabellenliste aufgenommen werden.

Die Tabelle wird schließlich mit der Anzahl der markierten Zeilen erstellt.

Abbildung 52: «Daten einfügen als Felder» – entspricht auch dem Fenster für «Daten einfügen als Text»

Die auf diese Weise in den Text eingefügten Felder können nachher auch einzeln gelöscht oder als Serienbrieffelder weiter genutzt werden.

Wird Daten einfügen als Text gewählt, so unterscheidet sich das gegenüber den Feldern nur insofern, als bei den Feldern weiterhin die Datenbankverknüpfung bestehen bleibt. Bei der Einfügung als Text wird lediglich der direkte Inhalt der jeweiligen Felder übernommen, nicht aber die Verknüpfung zur Datenbank selbst. Daher unterscheidet sich auch das Fenster für diese Funktion nicht von dem Fenster der vorhergehenden.

Das Ergebnis der beiden Funktionen sieht im Vergleich so aus:

<table>
<thead>
<tr>
<th>Daten einfügen als Felder</th>
<th>Daten einfügen als Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rob van Delft</td>
<td>Rob van Delft</td>
</tr>
<tr>
<td>Tel.: 09871/1946703</td>
<td>Tel.: 09871/1946703</td>
</tr>
<tr>
<td>geboren am 27.07.17</td>
<td>geboren am 27.07.77</td>
</tr>
<tr>
<td>Adressen Person.GebDat</td>
<td></td>
</tr>
</tbody>
</table>

Abbildung 53: Vergleich: «Daten als Felder» - Daten als «Text»

Die Felder sind grau hinterlegt. Wird mit einer Maus über die Felder gefahren, so zeigt sich, dass die Felder weiterhin eine Verbindung zur Datenbank "Adressen", zur dortigen Tabelle "Person" und zum, in dieser Tabelle befindlichen, Feld "GebDat" haben.
Abbildung 54: Doppelklick auf ein eingefügtes Feld zeigt die Eigenschaften des Seriendruckfeldes


Einfacher lässt sich so ein Feld übrigens erzeugen, wenn im Datenquellenbrowser der Spaltenkopf der Tabelle markiert und mit der Maus in das Dokument gezogen wird. So kann direkt ein Serienbrief erstellt werden.

Daten in Felder

Über «Daten einfügen als Felder» im vorherigen Abschnitt wurden in einem Writer-Dokument Seriendruckfelder erzeugt. Wird jetzt im Datenquellen-Browser ein anderer Datensatz markiert und dann «Daten in Felder» ausgewählt, so werden die vorherigen dargestellten Daten durch die neuen Daten ersetzt:

```
Daten einfügen als Felder                          Daten einfügen als Text
Mirna  Milinda                                      Rob  van Delft
Tel.:  467531                                        Tel.:  09871/1946703
geboren am 08.07.09                                   geboren am 27.07.77
```

Hier wurde ein anderer Datensatz markiert. Während bei Betätigung von «Daten in Felder» die vorher eingefügten Felder auf den neuen Datensatz umgeschrieben werden, bleibt der über «Daten einfügen als Text» erstellte Text bestehen.

Seriendruck

Mit diesem Button wird der Seriendruck-Assistent gestartet. Da für einen Serienbrief im obigen Beispiel die Daten aus verschiedenen Tabelle zusammengefasst werden müssen, muss zuerst einmal
die Datenbank gestartet werden. In der Datenbank ist dann eine neue Abfrage zu erstellen, die die Daten entsprechend zur Verfügung stellt.

Erfolgt der Start der Datenbank durch einen rechten Mausklick auf die Datenbank oder eine der Tabellen oder Abfragen der Datenbank, so wird die Ansicht im Datenquellenbrowser anschließend sofort aktualisiert. Anschließend kann dann der Serienbrief-Assistent über den entsprechenden Button aufgerufen werden.

**Aktuelle Dokument-Datenquelle**

Ein Klick auf diesen Button öffnet direkt die Ansicht auf die Tabelle, die die Grundlage für die Daten bildet, die in das Dokument eingefügt wurden. Im obigen Beispiel zeigt sich also sofort die Tabelle "Person" aus der Datenbank "Adressen".

**Explorer ein/aus**

Mit Betätigung dieses Buttons wird die Ansicht auf den Verzeichnisbaum links von der Tabellenansicht ein- und ausgeschaltet. Dies dient dazu, gegebenenfalls mehr Platz für eine Sicht auf die Daten zu erhalten. Um auf eine andere Tabelle zugreifen zu können muss der Explorer eingeschaltet werden.

**Serienbrieferstellung**

Das Ausgangsdokument des Serienbriefs ist das Dokument, mit dem die Datenbankfelder verknüpft werden.

Das Serienbriefdokument hingegen ist das, in dem nachher die Daten der verschiedenen Personen stehen, die den Serienbrief erhalten haben. Im Serienbriefdokument ist keine Verknüpfung zur Datenquelle mehr vorhanden. Dies entspricht der Einstellung aus «Daten einfügen als Text».

Mit dem Serienbrief-Assistenten können entweder Briefe oder E-Mails entsprechend mit Daten aus der Datenbank versorgt werden.
Das Einfügen des Adressblocks erfordert die umfangreichsten Einstellungen. Als Adressenliste wird erst einmal die aktuell gewählte Abfrage oder Tabelle der aktuell gewählten Datenbank vorgeschlagen.


Punkt 3 dient dazu, den entsprechend angedeuteten Feldern aus dem Adressblock die richtigen Felder aus der Datenbank zuzuweisen. Der Assistent erkennt zunächst einmal nur die Felder aus der Datenbank, die genau so geschrieben wurden wie die Feldbenennungen, mit denen der Assistent zuerst einmal arbeitet.

Unter Punkt 4 werden die Adressen angezeigt. Mit den Pfeiltasten können verschiedene Adressen aus der Datenbank abgerufen werden. Bei der abgebildeten Adresse fallen 2 Elemente auf, die einer Nachbearbeitung bedürfen:

- Eine Anrede fehlt
- Bis auf den Vornamen sind alle anderen Felder &lt; noch nicht zugewiesen &gt;, da die Benennung in der Datenbank anders ist als die Feldbenennungen, mit denen der Assistent zuerst einmal arbeitet.

Um diese Fehler zu bearbeiten, muss zuerst der Adressblock aus Punkt 2 nachbearbeitet werden.
Im Hintergrund ist zu erkennen, dass bei der Nachbearbeitung zuerst einmal noch wieder eine erweiterte Auswahl von Adressblöcken präsentiert wird. Der passendste Adressblock wurde hier ausgewählt und entsprechend bearbeitet.

Mit den Pfeiltasten werden Felder in den Adressblock übernommen bzw. aus dem Adressblock entfernt. Der Adressblock kann nicht direkt editiert werden. Stattdessen wird die Anordnung der Felder über die auf der rechten Seite sichtbaren Pfeiltasten hergestellt.

Für die Adressanrede wurde einfach das Element «Anrede» eingefügt. Bis auf den Vornamen müssen jetzt noch alle anderen Felder entsprechend zugewiesen werden.

Hier wurden die Adresselemente aus dem Serienbriefassistenten erfolgreich entsprechenden Elementen aus der Abfrage der Datenbank zugeordnet. Als Vorschau dient weiterhin der erste Datensatz der Abfrage.

Drei verschiedene Briefanreden werden so erzeugt. Alle mit einem 'w' versehenen Datensätze starten mit 'Sehr geehrte Frau …', alle mit 'm' versehenen Datensätze mit 'Sehr geehrter Herr …'. Ist kein Geschlecht angegeben, so wird schließlich 'Sehr geehrte Damen und Herren' gewählt.
Das **Ausgangsdokument** ist das Dokument, in dem die **Feldeigenschaften** und die **Verbindung zur Datenbank** gespeichert werden. Im Hintergrund ist währenddessen das Originaldokument des ersten Datensatzes sichtbar, das in der Form verschickt würde. Dies wird als Serienbriefdokument bezeichnet.

Erst wenn eine der Optionen durchgeführt wurde, in dem obigen Fall also das Ausgangsfeld gespeichert wurde, kann der Serienbriefassistent beendet werden.
Erstellung von Etiketten


In dem Editor kann mit der Tastatur gearbeitet werden. So ist es z. B. möglich, zu Beginn einen Zeilenumbruch einzufügen, damit die Etiketten nicht direkt an der oberen Kante beschriftet werden sondern der Inhalt möglichst komplet und gut sichtbar gedruckt werden kann.

Das Format kann im Reiter «Etiketten» vorgewählt werden. Hier sind viele Etikettenmarken einge-arbeitet, so dass meist keine weiteren Einstellungen im Reiter «Format» notwendig sind.
Hinweis

In den Versionen 3.4.x bis 3.5.2 kann aufgrund einer Änderung der Grundeinstellungen im Etikettenassistenten ein Darstellungsfehler vorliegen, wenn die Etiketten zusammen genau die gleiche Breite haben wie die Seitenbreite. Unter diesen Bedingungen rutscht das letzte Etikett einfach eine Zeile tiefer.

In der Version 3.3.x wurde hier standardmäßig die Seitenbreite für DIN-A4-Etikettenbögen geringfügig von 21,00 cm auf 21,04 cm erweitert. Ein nachträgliches Erweitern der Seitenbreite bringt bei den aktuellen Etikettenbögen auch hier die Lösung.

In der Version 3.5.3 wurden deshalb im Reiter «Format» Seiteneinstellungen hinzugefügt.

Unter dem Reiter «Format» lässt sich eine genaue Einstellung der Etikettengröße vornehmen. Die Einstellungen haben nur eine Bedeutung, wenn die Marke und der Typ nicht bekannt sind. Wichtig ist hier bei Etiketten der Breite 7,00 cm, die Seitenbreite (hier: «Page Width») geringfügig größer als 3*7,00 cm = 21,00 cm zu stellen. Erst dann werden 3 Etiketten nebeneinander auf einer Seite ausgegeben.
Unter dem Reiter «Zusätze» kann schließlich noch eingestellt werden, ob nur ein einzelnes Etikett oder eine ganze fortlaufende Seite erstellt werden soll. Die fortlaufende Seite wird dann, beginnend mit dem ersten Datensatz, mit Daten aus der Datenbank bestückt. Sind mehr Datensätze vorhanden als auf eine Seite passen, so wird automatisch das nächste Blatt mit den fortlaufenden Daten versehen.

Mit dem Markierfeld «Inhalte synchronisieren» werden alle Etiketten miteinander verbunden, so dass die anschließenden Änderungen im Layout einer Etikette auf die anderen Etiketten übernommen werden. Um den editierten Inhalt zu übertragen, muss lediglich die eingeblendete Schaltfläche mit dem Aufdruck Synchronisieren betätigt werden.

Über den Button Neues Dokument wird schließlich ein Dokument mit Serienbrieffeldern erzeugt.

Mit dem Aufruf des Druckvorganges erscheint die folgende Nachfrage:

Erst wenn diese Nachfrage mit «Ja» beantwortet wird, werden die Adressdatenbank-Felder auch mit einem entsprechenden Inhalt gefüllt.
Die Vorlage für den Druck wird nicht automatisch gefunden. Es wird lediglich die Datenbank vorgewählt. Die entsprechende Abfrage muss vermutlich selbst gesucht werden, weil es sich in diesem Fall eben nicht um eine Tabelle handelt.


**Serienbriefe und Etiketten direkt erstellen**

Neben den Assistenten steht natürlich der Weg offen, durch möglichst direkte Interaktion Serienbriefe und Etiketten selbst zu erstellen.

**Serienbrieffelder mit der Maus erstellen**

Serienbrief-Felder können mit der Maus aus dem Datenbankbrowser heraus erstellt werden:


**Serienbrieffelder über Feldbefehle erstellen**

Serienbrief-Felder können über Einfügen → Feldbefehl → Andere → Datenbank eingefügt werden.

Hier stehen in der gewählten Datenbank alle Tabellen und Abfragen zur Verfügung. Über den Button Einfügen können nacheinander die verschiedenen Felder direkt in den Text an der momentanen Cursorposition eingebunden werden.

Soll, wie im Serienbrief, eine Anrede erstellt werden, so kann dies mit Hilfe eines versteckten Absatzes oder versteckten Textes funktionieren: Einfügen → Feldbefehl → Andere → Funkti...
nen → Versteckter Absatz. Bei beiden Varianten ist zu beachten, dass die Bedingung, die formuliert wird, nicht erfüllt sein darf, damit der Absatz erscheint.

Damit die Formulierung 'Sehr geehrte Frau <Nachname>,' nur dann erscheint, wenn die Person weiblich ist, reicht als Bedingung

[Aadressen.Serienbriefabfrage.Geschlecht] != "w".

Nur kann es jetzt noch passieren, dass kein Nachname existiert. Unter diesen Umständen sollte dort stattdessen 'Sehr geehrte Damen und Herren,' erscheinen. Also ist diese Bedingung hinzuzufügen. Insgesamt ergibt das die Bedingung:


Damit ist schließlich ausgeschlossen, dass der Absatz erscheint, wenn die Person nicht weiblich ist oder kein Nachname eingetragen wurde.

Entsprechend kann mit den Eintragungen für das männliche Geschlecht und für fehlende Eintragungen für die beiden verbleibenden Anreden verfahren werden.

Auf die gleiche Art und Weise kann natürlich auch eine Anrede im Adressfeld erstellt werden, soweit eben das Geschlecht angegeben wurde.

Weitere Informationen hierzu in der Hilfestellung unter dem Titel «Versteckter Text» bzw. «Bedingter Text».

Noch einfacher wäre es natürlich für Personen mit Datenbankkenntnissen, die gesamte Anrede bereits in der Abfrage zu hinterlegen. Dies ist über eine Korrigierte Unterabfrage (Kapitel Abfragen) möglich.

Für Etiketten besonders interessant ist der Feldtyp «Nächster Datensatz». Wird dieser Feldtyp am Schluss einer Etikette gewählt, so wird die nächste Etikette mit dem darauffolgenden Datensatz gefüllt. Typische Etiketten für den fortlaufenden Etikettendruck sehen also wie im folgenden Bild aus, wenn über Ansicht → Feldnamen die entsprechenden Bezeichnungen sichtbar gemacht werden:

Abbildung 55: Feldbefehle für Etiketten mit fortlaufendem Inhalt

Bei der letzten Etikette auf der Seite ist allerdings darauf zu achten, dass hier der nächste Datensatz schon automatisch mit dem Seitenumbruch aufgerufen wird. Dort darf also der Feldtyp «Nächster Datensatz» nicht erscheinen. Sonst fehlt ein Datensatz, weil ein doppelter Datensatzsprung enthalten ist.

Tipp

Das Erstellen von Serienbriefen ist auch direkt im Formular einer Datenbank möglich. Voraussetzung ist lediglich, dass die Datenbank in LibreOffice registriert wird.

Beim Erstellen der Serienbriefe sollte allerdings darauf geachtet werden, dass Ansicht → Drucklayout gewählt wird. Dadurch ist sichergestellt, dass die Elemente anschließend auch an der korrekten Position auf dem Blatt erscheinen. Wird ein so erstelltes Formular gedruckt, so erscheint die übliche Serienbriefabfrage.

Serienbriefe dieser Art haben den Vorteil, dass neben der *.odb-Datei keine weiteren Dateien für den Druck existieren müssen.
**Externe Formulare**

Sollen einfache Formulareigenschaften unter LibreOffice in anderen Programmteilen wie Writer und Calc genutzt werden, so reicht es aus, über Ansicht → Symbolleisten → Formularentwurf die Formularentwurfsleiste anzeigen zu lassen, den «Formularnavigator» zu öffnen und ein Formular zu gründen oder, wie im Kapitel «Formulare» beschrieben, eine Formulargründung über ein Formularfeld. Es erscheint bei den Eigenschaften unter dem Reiter «Daten» ein etwas anderer Aufbau als bei Formularen direkt in der Datenbankdatei *.odb:

![Abbildung 56: Formular mit einer externen ...](image)

![Abbildung 57: ... und einer internen Datenquelle.](image)


Wird stattdessen nur im Listenfeld gesucht, so stehen dort die bereits in LibreOffice registrierten Datenbanken unter dem registrierten Namen zur Verfügung.

Die Formulare werden im Weiteren dann genau so erstellt wie unter Base selbst. Zur Formularerstellung gilt daher das Gleiche wie unter Base.

Die so erstellten Formulare werden standardmäßig bei jedem neuen Öffnen der Datei im Bearbeitungsmodus und nicht, wie in Base, schreibgeschützt geöffnet. Um eine versehentliche Änderung des Formulars zu vermeiden kann über Datei → Eigenschaften → Sicherheit die Datei schreibgeschützt geöffnet werden. Das Formular kann hier sogar mit einem Passwort gegen Veränderungen geschützt werden. Auf Betriebssystemebene lässt sich letztlich auch die ganze Datei als schreibgeschützt deklarieren. Dann sind immer noch die Eingaben in die Formularfelder möglich, aber nicht mehr das Verschieben der Felder oder eine Texteingabe zwischen den Feldern.

Für den Writer gilt: Mit der linken Maustaste markieren, Shift- und Strg-Taste gedrückt halten, so dass ein Verknüpfungssymbol als Mauscursor erscheint, in das Writerdokument ziehen.

Bei Calc-Dateien muss ohne Zuhilfenahme zusätzlicher Tasten in das Calc-Dokument gezogen werden. Dort erscheint das Kopiersymbol als Mauscursor.


### Vorteil externer Formulare

Base muss nicht erst geöffnet werden, um mit der Datenbank zu arbeiten. Im Hintergrund ist also nicht immer ein weiteres Fenster geöffnet.

Bei einer fertigen Datenbank können anderen Nutzern der Datenbank anschließend verbesserte Formulare problemlos zugesandt werden. So können sie während der Entwicklung weiterer Formulare die Datenbank weiter nutzen und müssen nicht, für Außenstehende kompliziert, Formulare aus einer Datenbank in eine andere kopieren.

Formulare zu einer Datenbank können je nach Nutzer der Datenbank unterschiedlich sein. Nutzer, die keine Datenkorrekturen und Neueingaben tätigen, können von anderen Nutzern den jeweils aktuellen Datenbestand zugespickt bekommen und einfach die *.odb-Datei austauschen, um einen aktuellen Bestand zu haben. Dies könnte z.B. bei einer Datenbank für Vereine sinnvoll sein, wo alle Vorstandsmitglieder die Datenbank erhalten, aber nur eine Person Daten letztlich bearbeitet, die anderen aber einen Blick auf die Adressen ihrer jeweiligen Abteilung haben.

Über externe Formulare ist der Zugriff und die Bearbeitung von Inhalten mehrerer Datenbanken möglich. Die unterschiedlichen Datenbanken brauchen nur im jeweiligen Neben- oder Unterformular selbst angegeben zu werden. Auch der Zugriff auf mehrere externe Datenbanken wie z.B. eine Datenbanksammlung unter MySQL/MariaDB ist damit möglich.

### Nachteil externer Formulare

Andere Nutzer müssen immer Formulare und Base in der gleichen Verzeichnisstruktur installieren. Nur so kann der einwandfreie Kontakt zur Datenbank hergestellt werden. Da die Links relativ zum Formular gespeichert werden reicht es allerdings aus, Datenbank und Formular in einem gemeinsamen Verzeichnis zu lagern. Sind die Datenbanken angemeldet, so muss allerdings lediglich der Anmeldename übereinstimmen.


### Datenbanknutzung in Calc

Daten können in Calc zu Berechnungszwecken genutzt werden. Dazu ist es notwendig, die Daten zuerst in einem Tabellenblatt von Calc verfügbar zu machen.
Daten in Calc einfügen

Daten lassen sich aus der Datenbank auf verschiedene Weisen in Calc einfügen:


Die so nach Calc hereingezogenen Daten weisen die folgenden Eigenschaften auf:

Nicht nur die Daten werden importiert, sondern auch die Eigenschaften der Felder werden ausgelesen und beim Import beachtet. Felder wie z.B. die Hausnummern, die als Textfelder deklariert wurden, werden auch als Text formatiert in Calc eingefügt.

Der Import wird direkt als Bereich eingefügt. Ihm wird standardmäßig der Name Import1 zugewiesen. Über Daten → Bereich aktualisieren wird der Bereich gegebenenfalls einem neuen Datenstand aus der Datenbank heraus angepasst.
Die hier eingelesenen Daten sind nicht weiter formatiert als es den Eigenschaften der Datenbankfelder entspricht.

Die Unterschiede zeigen sich besonders bei Feldern, die in der Datenbank als Text formatiert sind. Beim Import macht Calc daraus Textfelder und setzt Zahlen, die es sonst auch als Zahlen interpretieren würde, ein Hochkomma ('137) voran. Mit diesen Zahlen kann also nicht direkt weiter gerechnet werden.

Bei einem eventuellen Export wird das Hochkomma allerdings wieder entfernt, so dass die Daten letztlich in der gleichen Art bestehen bleiben.

<table>
<thead>
<tr>
<th>Import</th>
<th>Kopie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vorname</td>
<td>Vorname</td>
</tr>
<tr>
<td>Hausnummer</td>
<td>Hausnummer</td>
</tr>
<tr>
<td>Rob</td>
<td>137</td>
</tr>
<tr>
<td>Mirina</td>
<td>27 b</td>
</tr>
<tr>
<td>Heinz</td>
<td>159 b</td>
</tr>
<tr>
<td>Moni</td>
<td>37</td>
</tr>
<tr>
<td>Kerstin</td>
<td>45</td>
</tr>
</tbody>
</table>

Die Unterschiede zeigen sich besonders bei Feldern, die in der Datenbank als Text formatiert sind. Beim Import macht Calc daraus Textfelder und setzt Zahlen, die es sonst auch als Zahlen interpretieren würde, ein Hochkomma ('137) voran. Mit diesen Zahlen kann also nicht direkt weiter gerechnet werden.

Bei einem eventuellen Export wird das Hochkomma allerdings wieder entfernt, so dass die Daten letztlich in der gleichen Art bestehen bleiben.

<table>
<thead>
<tr>
<th>Tippe</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Tippe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daten können auch direkt aus der Datenbank über die Zwischenablage oder über Ziehen und Ablegen mit der Maus kopiert werden. Wird eine Tabelle oder Abfrage in ein Calc-Blatt gezogen, so wird der gesamte Inhalt eingefügt. Wird die Tabelle oder Abfrage geöffnet und ein oder mehrere Datensätze markiert, so werden nur diese Datensätze zusammen mit den Feldnamen beim Ziehen kopiert.</td>
</tr>
</tbody>
</table>

**Daten aus Calc in eine Datenbank exportieren**

Die Daten werden in dem Tabellenblatt von Calc markiert. Die linke Maustaste wird gedrückt und auf den Tabellenbereich der gewünschten Datenbank im Datenbankbrowser gezogen.

*Datenbank-Anbindung: Datenbanknutzung in Calc* 345
Der Cursor verändert sein Symbol und deutet an, dass etwas hinzugefügt werden kann.


**Daten von einer Datenbank zu einer anderen konvertieren**

Im Explorer des Datenquellenbrowsers können Tabellen von einer Datenbank zur anderen kopiert werden, indem die Quelltabelle mit der linken Maustaste markiert wird, die Taste dann gedrückt gehalten wird und über dem Tabellencontainer der Zieldatenbank losgelassen wird. Es erscheint dann der Dialog zum Kopieren von Tabellen.

Auf diese Weise können beispielsweise Datenbanken, die sonst nur gelesen werden können (Datenquelle z.B. ein Adressbuch aus einem Mailprogramm oder eine Tabellenkalkulationstabelle), als Grundlage für eine Datenbank genutzt werden, die anschließend auch schreibend auf die Daten zugreift. Auch können beim Wechsel eines Datenbankprogramms (z.B. von PostgreSQL zu MySQL) die Daten direkt kopiert werden.

Wird erwünscht, dass die neue Datenbank andere Relationen aufweisen soll als die alte Datenbank, so kann dies durch entsprechende Abfragen in der Datenbank realisiert werden. Wer darin
nicht so firm ist, kann stattdessen Calc nutzen. Hier werden die Daten in ein Tabellenblatt gezogen und anschließend mit Hilfe von Calc für den Import in die Zieldatenbank vorbereitet.

Für einen möglich sauberen Import in eine neue Datenbank sollten die Tabellen der neuen Datenbank vorher erstellt werden. So können Formatierungsprobleme und Probleme bei der Erstellung von Primärschlüsseln rechtzeitig erkannt werden.

**Daten über die Zwischenablage in eine Tabelle einfügen**

Der unter beschriebene Assistent zum Einfügen von Daten in eine Tabelle ist auch über Base direkt verfügbar. Sind Daten in Tabellenform vorhanden, so können diese über die Zwischenablage und den Assistenten in Base eingefügt werden.


Wird stattdessen nur in den Tabellencontainer mit der rechten Maustaste geklickt, so erscheint der Importassistent mit der entsprechenden Wahl einer neuen Tabelle.

**Datenimport aus PDF-Formularen**

Wer Daten aus verschiedenen Quellen von außen her importieren will sollte am besten eine Formularstruktur wählen, die bei der Eingabe der Daten nicht weiter verändert werden kann. Mit Hilfe des Writers lassen sich solche PDF-Formulare erstellen, im Internet verteilen und entsprechend z.B. als E-Mail-Anhang ausgefüllt zurückschicken. Fehlt schließlich nur noch der möglichst einfache Import in die Base-Datenbank. Das Beispiel12 soll so eine Importmöglichkeit aufzeigen.

**Erstellen eines PDF-Formulars**

Ein PDF-Formular wird als externes Formular ohne Datenbankanbindung erstellt. Über Ansicht → Symbolleisten → Formular-Steuerelemente werden die notwendigen Elemente für das Formular angezeigt und können entsprechend eingefügt werden.


Grundsätzlich ermöglichen PDF-Formulare nur die folgenden Formularfelder:

- Schaltfläche
- Textfeld
- Checkbox
- Combobox und
- Listbox

---

12 Die Datenbank «Beispiel_PDFFormular_Import.odb» zu diesem Bericht ist den Beispieldatenbanken für dieses Handbuch beigefügt.
Das Testformular enthält insgesamt 4 Textfelder. In Eigenschaften: Textfeld → Allgemein → Name soll für den folgenden Importmodus die jeweilige Bezeichnung gewählt werden, die auch in der Tabelle der Datenbank als Feldname vorgesehen ist. Dadurch kann einwandfrei Feldname und Feldinhalt zugeordnet werden.

Hilfetexte werden zwar beim Auslesen der Daten mit angezeigt, erscheinen allerdings wohl nicht in jedem *pdf-Viewer.


Auslesen der Daten aus dem PDF-Formular

Das Formular der Base-Datenbank sieht recht einfach aus. Es ist mit der Tabelle verbunden und zeigt die gerade eingelesenen Daten an. Dabei sind die jüngsten Einträge in dem Tabellenkontrollfeld oben zu sehen.
Das Makro zum Einlesen der Daten wird unter **Eigenschaften : Schaltfläche → Ereignisse → Aktion ausführen** eingetragen.


Die mit «pdftk» ausgelesenen Daten werden in eine Textdatei geschrieben, die wie folgt aussieht:

```plaintext
1 ---
2 Fieldtype: Text
3 FieldName: Vorname
4 FieldFlags: 0
5 Fieldvalue: Karl
6 FieldJustification: Left
7 ---
8 Fieldtype: Text
9 FieldName: Nachname
10 FieldFlags: 0
11 Fieldvalue: Käfer
12 FieldJustification: Left
13 ---
14 Fieldtype: Text
15 FieldName: Geburtsdatum
16 FieldNameAlt: Datum mit mindestens zweistelliger Jahreszahl
17 FieldFlags: 0
18 Fieldvalue: 1.3.12
19 FieldJustification: Left
20 ---
21 Fieldtype: Text
22 FieldName: Punktzahl
23 FieldNameAlt: Dezimalzahl, 2 Nachkommastellen
24 FieldFlags: 0
25 Fieldvalue: 3,71
26 FieldJustification: Left
27---
```

Für jedes Feld sind in der Textdatei 5 Zeilen enthalten. Für die Auswertung im Makro sind die Zeilen «FieldName» (Fieldname auch in der Zieltabelle), «FieldValue» (Inhalt des Feldes nach Abspeicherung der *.pdf-Datei) sowie «FieldJustification» (letzte Zeile, die einen Feldeintrag beendet) von Bedeutung.

Der gesamte Import wird über Makros geregelt. Hier muss das PDF-Formular im gleichen Pfad wie die Datenbank abgelegt werden. Die Daten werden in die Textdatei ausgelesen und aus dieser Textdatei wieder eingelesen. Dies geschieht so oft, wie *.pdf-Dateien mit Formulardaten in dem Verzeichnis liegen. Alte Dateien sollten also tunlichst aus dem Verzeichnis entfernt werden, denn die Funktion überprüft nicht auf Duplikate.

```plaintext
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 infFields 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
```

**Datenbank-Anbindung: Datenimport aus PDF-Formularen** 349
Nach der Deklaration der Variablen wird die Anzahl der Felder angegeben, die das PDF-Formular enthält. Die Zählung beginnt hier mit 0. Beim Wert '3' sind also insgesamt 4 Felder in dem Formular vorhanden. Mit Hilfe dieser Zählung wird ermittelt, wann alle Daten für einen Datensatz ausgelesen wurden, so dass die Daten in die Tabelle der Datenbank übergeben werden können.

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


DB = ThisComponent.Parent  
stDir = Left(DB.Location,Len(DB.Location)-Len(DB.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))  
 END IF  


Im Anschluss daran gibt die Befehl Shell() die Startinformationen für pdftk an die Konsole weiter. Über die Angabe von True wird dabei geregelt, dass LibreOffice wartet, bis der Shell-Prozess beendet wird.

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


Datenbank-Anbindung: Datenimport aus PDF-Formularen
OPEN stFile FOR INPUT AS inNumber
DO WHILE NOT Eof(inNumber)
    LINE INPUT #inNumber, stRow
Zeil für Zeile wird die PDF-Datendatei jetzt ausgelesen. Taucht der Begriff «FieldName: » auf, so entspricht der folgende Inhalt der Zeile der Bezeichnung des Feldes im PDF-Formular und durch die Definition des Formulars auch der Bezeichnung des Feldes in der Datenbanktabelle, in die der Inhalt geschrieben werden soll.
Alle Feldnamen werden direkt für den späteren SQL-Befehl zusammengefasst. Das bedeutet, dass die Feldnamen mit doppelten Anführungszeichen versehen und durch Kommas voneinander getrennt werden.
Zu jedem Feldnamen wird außerdem durch eine Abfrage ermittelt, um welchen Feldtyp der Tabelle es sich hierbei handelt. Datumswerte und Dezimalzahlen müssen anders weiter gegeben werden als Texte.

IF instr(stRow, "FieldName: ") THEN
    IF stFieldName = "" THEN
        stFieldName = "" + mid(stRow, 12) + ""
    ELSE
        stFieldName = stFieldName & "," + mid(stRow, 12) + ""
    END IF
    strSql = "SELECT TYPE_NAME FROM INFORMATION_SCHEMA.SYSTEM_COLUMNS WHERE TABLE_NAME = "" + stTable + "' AND COLUMN_NAME = "" + mid(stRow, 12) + "'"
    oResult = oSQL_Command.executeQuery(strSql)
    WHILE oResult.next
        stFieldType = oResult.getString(1)
    WEND
END IF

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
Taucht der Begriff «FieldJustification: » auf, so ist das Ende einer Kombination von Feldbezeichnung und Feldwert erreicht. Der Zähler i, der anschließend mit der vorher angegebenen Anzahl der Felder inFelder verglichen werden soll, wird um '1' heraufgesetzt.
Wenn schließlich i und inFields gleich ist wird der SQL-Befehl zusammengestellt. Allerdings soll vermieden werden, dass leere Datensätze bei leeren Formularen entstehen. Deshalb wird vorher nachgesehen, ob die Werte aller Felder NULL sind. Ist dies der Fall, so wird kein SQL-Befehl ausgelöst. Ansonsten wird der Datensatz in die Tabelle "Name" eingefügt. Anschließend werden die Variablen wieder auf ihre Standardwerte zurückgesetzt und das nächste PDF-Formular kann ausgelesen werden.

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
Datenbank-Anbindung: Datenimport aus PDF-Formularen 351
Datenbank-Anbindung: Datenimport aus PDF-Formularen

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

Zum Schluss der Prozedur bleibt schließlich eine Datei «PDF_Form_Data.txt» übrig. Diese Datei wird gelöscht. Anschließend wird das Formular neu geladen, damit die eingelesenen Daten direkt angezeigt werden können.

Kill(stFile)
END IF
NEXT

Enthält ein Text eine «'», so wird die beim Einfügen in SQL als Ende des Textes angesehen. Der SQL-Code für den Insert-Befehl scheitert, wenn anschließend noch Text folgt, der nicht in einfachen Anführungszeichen steht. Um dies zu vermeiden muss jedes einfache Anführungszeichen innerhalb des Textes mit einem weiteren einfachen Anführungszeichen maskiert werde. Das erledigt die Funktion 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

Die Datumsangabe in einem PDF-Formular wird als Text ausgelesen. Sie kann nicht vorher auf korrekte Eingabe kontrolliert werden.

In der deutschen Schreibweise werden Tag, Monat und Jahr voneinander durch einen Punkt getrennt. Dabei kann die Tagesangabe und die Monatsangabe einstellig oder zweistellig, die Jahresangabe zweistellig oder vierstellig sein.

Für den SQL-Code muss die Datumsangabe beginnend mit der vierstelligen Jahreszahl in dem folgenden Format geschrieben werden: YYYY-MM-DD. Es muss also auf jeden Fall eine Umwandlung des eingegebenen Datums erfolgen.

Die Datumsangabe wird in den Tagesanteil, den Monatsanteil und den Jahresanteil aufgesplittet. Die Tagesangabe und die Monatsangabe wird mit einer führenden «0» versehen und anschließend von rechts aus auf zwei Zeichen begrenzt. Damit ist die Angabe auf jeden Fall zweistellig.

Ist die Jahresangabe bereits vierstellig (größer als 1000), so wird der Wert nicht geändert. Ansonsten wird bei einer Jahresangabe größer als 30 davon ausgegangen, dass es sich um eine Angabe handelt, die im letzten Jahrhundert liegt und um 1900 erhöht werden muss. Alle anderen Jahresangaben werden als Angaben für das aktuelle Jahrhundert angesehen.

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
  stDate = stDay & "." & stMonth & "." & inYear
  Date_to_SQLDate = stDate
END FUNCTION
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

Die Funktion SQL_Value fasst die vorhergehenden Funktionen sowie die NULL-Werte zusammen und gibt entsprechend vorformatierte Werte für die Eingabe in die Datenbank an die aufrufende Prozedur zurück.

Für leere Felder wird NULL zurück gegeben. Damit bleibt das Feld auch in der Tabelle anschließend leer.

FUNCTION SQL_Value(st AS STRING, stType AS STRING) AS STRING
  DIM stValue AS STRING
  IF st = "" THEN
    SQL_Value = "NULL"
  END IF
  IF isDate(st) THEN
    SQL_Value = ";" & Date_to_SQLDate(st) & ";"
  ELSE
    SQL_Value = "NULL"
  END IF
  HANDelt es sich bei dem Feldtyp um ein Datumsfeld und ist der Inhalt als Datum zu erkennen, so soll der Inhalt in ein SQL-Datumsformat umgewandelt werden. Ist der Inhalt nicht als Datum erkennbar, so soll das Feld leer bleiben.

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


ELSEIF stType = "DECIMAL" THEN
  stValue = Str(Val(Join(Split(st,"","","")))
  SQL_Value = Join(Split(stValue,"","",""))
  Alle anderen Inhalte werden als Text behandelt. Einfache Anführungszeichen werden mit einem weiteren einfachen Anführungszeichen maskiert und der Gesamtbegriff wiederum in einfachen Anführungszeichen gefasst weitergegeben.

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

 Zu weiteren Details beim Aufbau von Makros sollte das separate Kapitel dieses Handbuches zu Rate gezogen werden. Dieses Beispiel sollte lediglich aufzeigen, dass es auch möglich ist, Daten aus PDF-Formularen nach Base zu übertragen, ohne die Werte über die Zwischenablage aus jedem Feld in die Datenbank kopieren zu müssen. Der Aufbau der obigen Prozedur ist dabei recht allgemein gehalten und müsste sicher den jeweiligen Bedürfnissen angepasst werden.
Datenbank-Aufgaben
Allgemeines zu Datenbankaufgaben


Datenfilterung

Die Datenfilterung mittels der GUI ist bereits bei der Dateneingabe in Tabellen beschrieben. Hier soll eine Lösung aufgezeigt werden, die bei vielen Nutzern gefragt ist: Mittels Listenfeldern werden Inhalte von Tabellenfeldern ausgesucht, die dann im darunterliegenden Formularteil herausgefiltert erscheinen und bearbeitet werden können.

Grundlage für diese Filterung ist neben einer bearbeitbaren Abfrage (siehe das Kapitel «Eingabemöglichkeit in Abfragen») eine weitere Tabelle, in der die zu filternden Daten abgespeichert werden. Die Abfrage zeigt aus der ihr zugrundeliegenden Tabelle nur die Datensätze an, die dem eingeggebenen Filterwert entsprechen. Ist kein Filterwert angegeben, so zeigt die Abfrage alle Datensätze an.

Für das folgende Beispiel wird von einer Tabelle "Medien" ausgegangen, die unter anderem die folgenden Felder beinhaltet: "ID" (Primärschlüssel), "Titel", "Kategorie".

Zuerst wird eine Tabelle "Filter" benötigt. Diese Tabelle erhält einen Primärschlüssel und 2 Filterfelder (das kann natürlich beliebig erweitert werden): "ID" (Primärschlüssel), "Filter_1", "Filter_2". Da die Felder der Tabelle "Medien", die gefiltert werden sollen, vom Typ VARCHAR sind, haben auch die Felder "Filter_1" und "Filter_2" diesen Typ. "ID" kann dem kleinsten Zahlen­typ, TINYINT, entsprechen. Die Tabelle "Filter" wird sowieso nur einen Datensatz abspeichern.

Natürlich kann auch nach Feldern gefiltert werden, die in der Tabelle "Medien" nur über einen Fremdschlüssel vertreten sind. Dann müssen die entsprechenden Felder in der Tabelle "Filter" natürlich dem Typ des Fremdschlüssels entsprechen, in der Regel also «Integer» sein.

Folgende Abfrageform bleibt sicher editierbar:

```
SELECT * FROM "Medien"
```

Alle Datensätze der Tabelle "Medien" werden angezeigt, auch der Primärschlüssel.

```
SELECT * FROM "Medien" WHERE "Titel" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), "Titel" )
```

Ist das Feld "Filter_1" nicht NULL, so werden die Datensätze angezeigt, bei denen der "Titel" gleich dem "Filter_1" ist. Wenn das Feld "Filter_1" NULL ist wird stattdessen der Wert des Feldes "Titel" genommen. Da "Titel" gleich "Titel" ist werden alle Datensätze angezeigt – sollte angenommen werden, trifft aber nicht zu, wenn im Feld "Titel" irgendwann ein leeres Feld 'NULL' enthalten ist. Das bedeutet, dass die Datensätze nie angezeigt werden, die keinen Titleleintrag haben. Hier muss in der Abfrage nachgebessert werden.

```
SELECT *, IFNULL( "Titel", '' ) AS "T" FROM "Medien" WHERE "T" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), "T" )
```

Leider spielt hier die GUI nicht mit. Der Befehl ist nur direkt über SQL absetzbar. Um ihn mit der GUI editierbar zu machen ist weitere Handarbeit erforderlich:

```
SELECT "Medien".* , IFNULL( "Medien"."Titel", '' ) AS "T" FROM "Medien" WHERE "T" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), "T" )
```

Wenn jetzt der Tabellenbezug zu den Feldern hergestellt ist, ist die Abfrage auch in der GUI editierbar.

Zum Testen kann jetzt einfach ein Titel in "Filter"."Filter_1" eingegeben werden. Als "Filter"."ID" wird der Wert '0' gesetzt. Der Datensatz wird abgespeichert und die Filterung kann nachvollzogen werden. Wird "Filter"."Filter_1" wieder geleert, so macht die GUI daraus NULL. Ein erneuter Test ergibt, dass jetzt wieder alle Medien angezeigt werden. Bevor ein Formular erstellt und getestet wird, sollte auf jeden Fall ein Datensatz, aber wirklich nur einer, mit einem Primärschlüssel in der Tabelle "Filter" stehen. Nur ein Datensatz darf es sein, da Unterabfragen wie oben gezeigt nur einen Wert wiedergeben dürfen.

Die Abfrage wird jetzt erweitert, um auch ein 2. Feld zu filtern:

```
SELECT "Medien".*, IFNULL( "Medien"."Titel", '' ) AS "T", IFNULL( "Medien"."Kategorie", '' ) AS "K" FROM "Medien" WHERE "T" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), "T" ) AND "K" = IFNULL( ( SELECT "Filter_2" FROM "Filter" ), "K" )
```

Damit ist die Erstellung der editierbaren Abfrage abgeschlossen. Jetzt wird noch die Grundlage für die beiden Listenfelder als Abfrage zusammengestellt:

```
SELECT DISTINCT "Titel", "Titel" FROM "Medien" ORDER BY "Titel" ASC
```

Das Listenfeld soll sowohl die "Titel" anzeige als auch die "Titel" an die dem Formular zugrundeliegende Tabelle "Filter" in das Feld "Filter_1" weitergeben. Dabei sollen keine doppelten Werte angezeigt werden (Anordnung «DISTINCT»). Und das Ganze soll natürlich richtig sortiert erscheinen. Dabei ist die Abfrage an die Standardeinstellung der Listenfelder angepasst, die dem gebundenen Feld eine '1' zuweist. Wird stattdessen unter Eigenschaften Listenfeld → Daten → Gebundenes Feld eine '0' zugewiesen, so braucht nur einmal das Feld "Titel" abgefragt werden.

Eine entsprechende Abfrage wird dann auch für das Feld "Kategorie" erstellt, die ihre Daten in der Tabelle "Filter" in das Feld "Filter_2" schreiben soll.

Handelt es sich bei einem der Felder um ein Fremdschlüssel Feld, so ist die Abfrage entsprechend so anzupassen, dass der Fremdschlüssel an die zugrundeliegende Tabelle "Filter" weitergegeben wird.


Formular 1 enthält 2 Listenfelder mit entsprechenden Überschriften. Listenfeld 1 soll Werte für "Filter_1" liefern und wird mit der Abfrage für das Feld "Titel" versorgt. Listenfeld 2 soll Werte für "Filter_2" weitergeben und beruht auf der Abfrage für das Feld "Kategorie".

Wenn das Formular fertiggestellt ist, geht es zur Testphase. Wird ein Listenfeld geändert, so reicht die Betätigung des Buttons aus dem Formular 2 aus, um zuerst diesen Wert zu speichern und dann das Formular 2 zu aktualisieren. Das Formular 2 bezieht sich jetzt auf den Wert, den das Listenfeld angibt. Die Filterung kann über die Wahl des im Listenfeld enthaltenen leeren Feldes rückgängig gemacht werden.

**Datensuche**

Der Hauptunterschied zwischen der Suche von Daten und der Filterung von Daten liegt in der Abfragetechnik. Schließlich soll zu frei eingegebenen Begriffen ein Ergebnis geliefert werden, das diese Begriffe auch nur teilweise beinhaltet.

**Suche mit LIKE**

Die Tabelle für die Suchinhalte kann die gleiche sein, in die bereits die Filterwerte eingetragen werden. Die Tabelle "Filter" wird einfach ergänzt um ein Feld mit der Bezeichnung "Suchbegriff". So kann gegebenenfalls auf die gleiche Tabelle zugreifen werden und in Formularen gleichzeitig gefiltert und gesucht werden. "Suchbegriff" hat die Feldeigenschaft «VARCHAR».


```
SELECT * FROM "Medien" WHERE "Titel" = ( SELECT "Suchbegriff" FROM "Filter" )
```

Diese Abfrage würde in der Regel ins Leere führen. Das hat mehrere Gründe:

- Selten weiß jemand bei der Eingabe des Suchbegriffs den kompletten Titel fehlerfrei auswendig. Damit würde der Titel nicht angezeigt. Um das Buch «Per Anhalter durch die Galaxis» zu finden müsste es ausreichen, in das Suchfeld 'Anhalter' einzugeben, vielleicht auch nur 'Anh'.
- Ist das Feld "Suchbegriff" leer, so würde überhaupt kein Datensatz angezeigt. Die Abfrage gäbe `NULL` zurück und `NULL` kann in einer Bedingung nur mittels `IS NULL` erscheinen.
- Selbst wenn dies ignoriert würde, so würde die Abfrage dazu führen, dass alle die Datensätze angezeigt würden, die keine Eingabe im Feld "Titel" haben.

Die letzten beiden Bedingungen könnten erfüllt werden, indem wie bei der Filterung vorgegangen würde:

```
SELECT * FROM "Medien" WHERE "Titel" = IFNULL( ( SELECT "Suchbegriff" FROM "Filter" ), "Titel" )
```

Mit den entsprechenden Verfeinerungen aus der Filterung (was ist mit Titeln, die `NULL` sind?) würde das zum entsprechenden Ergebnis führen. Nur würde die erste Bedingung nicht erfüllt. Die Suche lebt ja schließlich davon, dass nur Bruchstücke geliefert werden. Die Abfragetechnik der Wahl müsste daher über den Begriff «LIKE» gehen:

```
SELECT * FROM "Medien" WHERE "Titel" LIKE ( SELECT '%' || "Suchbegriff" || '%' FROM "Filter" )
```

oder besser:
SELECT * FROM "Medien" WHERE "Titel" LIKE IFNULL( ( SELECT '%' || "Suchbegriff" || '%' FROM "Filter" ), "Titel" )

**LIKE**, gekoppelt mit '%', bedeutet ja, dass alle Datensätze gezeigt werden, die an irgendeiner Stelle den gesuchten Begriff stehen haben. '%' steht als Joker für beliebig viele Zeichen vor und hinter dem Suchbegriff. Verschiedene Baustellen bleiben nach dieser Abfrageversion:

- Besonders beliebt ist ja, in Suchformularen alles klein zu schreiben. Wie bekomme ich mit 'anhalter' statt 'Anhalter' auch noch ein Ergebnis?
- Welche anderen Schreibgewohnheiten gibt es noch, die vielleicht zu berücksichtigen wären?
- Wie sieht es mit Feldern aus, die nicht als Textfelder formatiert sind? Lassen sich auch Datumsanzeigen oder Zahlen mit dem gleichen Feld suchen?
- Und was ist, wenn, wie bei dem Filter, ausgeschlossen werden muss, dass NULL-Werte in dem Feld verhindern, dass alle Datensätze angezeigt werden?

Die folgende Variante deckt ein paar mehr an Möglichkeiten ab:

SELECT * FROM "Medien" WHERE
LOWER("Titel") LIKE IFNULL( ( SELECT '%' || LOWER("Suchbegriff") || '%' FROM "Filter" ), LOWER("Titel") )


SELECT * FROM "Medien" WHERE
LOWER("Titel") LIKE IFNULL( ( SELECT '%' || LOWER("Suchbegriff") || '%' FROM "Filter" ), LOWER("Titel") ) OR
LOWER("Kategorie") LIKE ( SELECT '%' || LOWER("Suchbegriff") || '%' FROM "Filter" )


Tipp

Die Abfragen, die zur Filterung und zum Durchsuchen von Daten genutzt werden, lassen sich auch direkt in ein Formular einbauen. Die gesamten obigen Bedingungen sind bei den Formular-Eigenschaften in der Zeile Filter eintragbar. Aus

```
SELECT * FROM "Medien" WHERE "Titel" = IFNULL( ( SELECT "Suchbegriff" FROM "Filter" ), "Titel" )
```

wird dann ein Formular, das als Inhalt die Tabelle "Medien" nutzt. Unter «Filter» steht dann

```
("Medien"."Titel" = IFNULL( ( SELECT "Suchbegriff" FROM "Filter" ), "Medien"."Titel" ))
```

In den Filtereingaben ist darauf zu achten, dass die Bedingung in Klammern gesetzt wird und jeweils mit der Angabe "Tabelle"."Feld" arbeitet.

Vorteil dieser Variante ist, dass der Filter bei geöffnetem Formular ein- und wieder ausgeschaltet werden kann.

**Suche mit LOCATE**

Die Suche mit LIKE ist in der Regel völlig ausreichend für Datenbanken mit Feldern, die Text in überschaubarem Maße enthalten. Was aber, wenn der Inhalt über Memo-Felder eingegeben wird, also ohne weiteres auch einmal mehrere Seiten Text enthalten kann? Dann geht die Suche erst einmal los, wo denn nun der Text zu finden ist.

Um Text genau zu finden gibt es in der HSQLDB die Funktion LOCATE. LOCATE erwartet einen Suchbegriff sowie den Text, der durchsucht werden soll, als Parameter. Zusätzlich kann noch angegeben, ab welcher Position gesucht werden soll. Kurz also: LOCATE(Suchbegriff, Textfeld aus der Datenbank, Position).


![Parameter-Eingabe](image)

Die Beispielabfragen sind als Parameterabfragen angelegt. Der einzugebende Suchtext ist jeweils 'office'.

---

13 Die Screenshots zu diesem Kapitel entstammen der Datenbank «Beispiel_Autotext_Suchmarkierung_Rechtschreibung.odb», die dem Handbuch beiliegt.

LOCATE gibt genauer wieder, an welcher Stelle sich der Suchbegriff befindet. In Datensatz 1 und 2 ist der Suchbegriff nicht vorhanden. LOCATE gibt als Position hier '0' aus. Leicht nachzählen lässt sich das Ergebnis am Datensatz 5: Mit dem 6. Buchstaben beginnt hier die Textfolge 'Office'.

Natürlich wäre es auch möglich, das entsprechende Ergebnis wie bei LIKE auch über LOCATE zu erhalten:

```
SELECT "ID", "Memo"
FROM "Tabelle"
WHERE LOCATE(LOWER(:Suchtext),LOWER("Memo")) > 0
```


In der Spalte «Treffer» wird das Suchergebnis genauer dargestellt. Die vorherige Abfrage ist einfach als Basis für diese Abfrage genommen worden. Dies bewirkt, dass in der äußeren Abfrage

```
SELECT "ID", "Memo", "Position",
CASE
  WHEN "Position" = 0 THEN "Keine Fundstelle"
  WHEN "Position" < 10 THEN SUBSTRING ("Memo", 1 )
  ELSE SUBSTRING ("Memo", LOCATE( "", "Memo", "Position" - 10 ) + 1 )
END AS "Treffer"
FROM
( SELECT "ID", "Memo", LOCATE( LOWER ( :Suchtext ), LOWER ( "Memo" ) ) AS "Position"
FROM "Tabelle"
) AS "Tabelle"
```
nicht jedes Mal \texttt{LOCATE(LOWER(:Suchtext),LOWER("Memo"))} sondern einfach "Position" eingegangen werden kann. Vom Prinzip her ist dieses Verfahren nicht anders, als wenn die vorhergehenden Abfrage gespeichert würde und diese Abfrage als Quellenangabe auf die vorherige Abfrage zugreift.

"\textit{Position}" = 0 bedeutet, dass kein Suchergebnis vorhanden ist. In dem Fall also die Ausgabe '**keine Fundstelle**'.

"\textit{Position}" < 10 bedeutet, dass sich der Suchbegriff direkt am Anfang des Textes befindet. 10 Zeichen können leicht überblickt werden. Es wird also der gesamte Text wiedergegeben. Hier könnte also auch statt \texttt{SUBSTRING("Memo",1)} direkt "Memo" stehen.

Für alle anderen Treffer wird ab 10 Zeichen vor der Position des Treffers nach einer Leerstelle ' ' gesucht. Der Text soll nicht mitten in einem Wort starten, sondern nach so einer Leerstelle beginnen. Über \texttt{SUBSTRING("Memo",LOCATE(' ','"Memo","Position"-10)+1)} wird erreicht, dass der Text mit dem Beginn eines Wortes startet, das maximal 10 Zeichen vor dem Begriff 'office' erscheint.


\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
ID & \textit{Memo} & \textit{Position} & \textit{Treffer} \\
\hline
1 & Dieses Dokument unterliegt dem & 0 & **keine Fundstelle** \\
2 & Alles, was an eine Mailingliste & 0 & **keine Fundstelle** \\
3 & Jeder, der sich in das Modul Base von LibreOffice einarbeiten & 44 & LibreOffice-einarbeit \\
4 & Dieses Buch, wie auch die anderen & 40 & LibreOffice-Handbücher; \\
5 & LibreOffice besitzt ein umfangreiches & 5 & LibreOffice besitzt ein \\
\hline
\end{tabular}
\end{table}


Die Staffelung der Abfragen ist hier beliebig weit möglich. Allerdings werden sie mit jeder weiteren äußeren Abfrage mehr für das System belastender. Hier sind entsprechende Test notwendig, was denn nun sinnvoll und was realistisch machbar ist. Wie mit Hilfe von Makros über ein Formular sogar eine Abfragetechnik machbar ist, die gleichzeitig alle Fundstellen im Text markiert, ist im Kapitel «Makros» erklärt.

### Bilder und Dokumente mit Base verarbeiten

Base-Formulare bieten für die Verarbeitung von Bildern grafische Kontrollfelder an. Nur über diese grafischen Kontrollfelder ist ohne Einsatz von Makros bei der internen HSQLDB möglich, Bilder in
die Datenbank ein zu lesen. Das grafische Kontrollfeld kann aber auch dazu genutzt werden, nur die Verknüpfung zu Bildern außerhalb der Datenbankdatei zu ermöglichen.14

**Bilder in die Datenbank einlesen**

Die Datenbank benötigt eine Tabelle, die mindestens die folgenden Voraussetzungen erfüllt:

<table>
<thead>
<tr>
<th>Feldname</th>
<th>Feldtyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>Die ID ist Primärschlüssel dieser Tabelle.</td>
</tr>
<tr>
<td>Bild</td>
<td>Text</td>
<td>Nimmt das Bild als Binärdatenstrom auf.</td>
</tr>
</tbody>
</table>

Beim Primärschlüssel ist natürlich nicht unbedingt der Integer-Feldtyp bestimmend. Ein Primärschlüssel ist aber unabdingbar. Andere Felder, die zumindest Informationen zu dem Bild enthalten, sollten noch hinzugefügt werden.

Daten, die irgendwann in das Bild-Feld eingetragen werden, sind in den Tabellen nicht lesbar. Dort erscheint als Anzeige nur `<OBJECT>`. Entsprechend sind Bilder auch nicht direkt in die Tabelle eingebbar. Sie müssen über ein Formular und dort mit Hilfe des grafischen Kontrollfeldes eingegeben werden. Das grafische Kontrollfeld öffnet beim Mausklick auf das Feld eine Dateiauswahl. Es zeigt hinterher das Bild an, das über die Dateiauswahl in die Datenbank eingelesen wurde.

Bilder, die direkt in die Datenbank eingefügt werden, sollten möglichst klein sein. Da Base ohne den Einsatz von Makros auch keine Möglichkeit bietet, die Bilder in Originalgröße wieder aus der Datenbank heraus zu befördern macht es aus dieser Warte erst einmal Sinn, als Maßstab für die Größe z.B. einen möglichen Ausdruck im Bericht anzusehen. Originalbilder im Megapixelbereich sind hier völlig unnötig und blähen die Datenbank stark auf. Bereits nach wenigen Bildern meldet bei der internen HSQLDB Base eine `JavaNullPointerException` und kann den Datensatz nicht mehr speichern. Auch wenn die Bilder nicht ganz so groß sind kann es irgendwann passieren, dass die Datenbankdatei unbedienbar wird.


**Bilder und Dokumente verknüpfen**

Mit einer entsprechend durchdachten Ordnerstruktur ist es günstiger, direkt auf die Dateien von außerhalb zuzugreifen. Dateien außerhalb der Datenbank können beliebig groß sein, ohne dass die Funktionen der Datenbank selbst beeinflusst werden. Leider bedeutet dies aber auch, dass eine Umbenennung von Ordnern auf dem eigenen Rechner oder im Internet dazu führt, dass der Zugriff auf die entsprechenden Dateien verloren geht.

Um Bilder nicht in eine Datenbankdatei einzulesen, sondern nur zu verknüpfen, bedarf es nur einer kleinen Änderung gegenüber der vorhergehenden Tabelle:

<table>
<thead>
<tr>
<th>Feldname</th>
<th>Feldtyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>Die ID ist Primärschlüssel dieser Tabelle.</td>
</tr>
<tr>
<td>Bild</td>
<td>Text</td>
<td>Nimmt den Pfad zu dem Bild auf.</td>
</tr>
</tbody>
</table>

Wird einfach statt des Feldtyps Bild der Feldtyp Text gewählt, so wird über das grafische Kontrollfeld der Pfad zu dem Bild eingetragen. Das Bild kann über das Kontrollfeld genauso betrachtet werden wie ein Bild, das in die Datenbank eingefügt wurde. Leider lässt sich aber nicht auf die glei-

---

14 Die Datenbank «Beispiel_Dokumente_einlesen_auslesen.odb» liegt diesem Handbuch bei.

*Datenbank-Aufgaben: Bilder und Dokumente mit Base verarbeiten* 364
che Art und Weise auch die Verbindung zu einem Dokument herstellen. Schon das Einlesen des Pfades ist nicht möglich, da das grafische Kontrollfeld von der Konstruktion her erst einmal nur für Bilder gedacht ist und auch entsprechend nur Bildtypen zur Auswahl anbietet, die es selbst auch anzeigen kann.


<table>
<thead>
<tr>
<th>Feldname</th>
<th>Feldtyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>Die ID ist Primärschlüssel dieser Tabelle.</td>
</tr>
<tr>
<td>Beschreibung</td>
<td>Text</td>
<td>Beschreibung des Dokumentes, Suchbegriffe ...</td>
</tr>
<tr>
<td>Datei</td>
<td>Text</td>
<td>Nimmt den Pfad zu dem Bild auf.</td>
</tr>
</tbody>
</table>

Damit der Pfad zur Datei sichtbar wird, muss in dem Formular ein Dateiauswahlfeld mit eingebaut werden.

Ein Dateiauswahlfeld hat in seinen Eigenschaften keinen Reiter für Daten, ist also auch nicht mit irgendeinem Feld der dem Formular zugrundeliegenden Tabelle verbunden.

**Dokumente mit absoluter Pfadangabe verknüpfen**

Über das Dateiauswahlfeld kann zwar der Pfad angezeigt aber nicht gespeichert werden. Hierfür ist eine gesonderte Prozedur erforderlich, die über Ereignisse → Text modifiziert ausgelöst wird:

```vba
SUB Pfad_Einlesen(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oFeld2 AS OBJECT
    DIM stUrl AS STRING
    oFeld = oEvent.Source.Model
    oForm = oFeld.Parent
    oFeld2 = oForm.getByName("GraphischesFeld")
    IF oFeld.Text <> "" THEN
        stUrl = ConvertToUrl(oFeld.Text)
        oFeld2.BoundField.updateString(stUrl)
    END IF
END SUB
```

Das auslösende Ereignis wird beim Aufruf der Prozedur mitgeliefert und hilft dabei, das Formular und auch das Feld zu finden, in dem der Pfad gespeichert werden soll. Mit **oEvent AS OBJECT** ist der Zugriff vor allem dann einfacher, wenn ein anderer User ein gleichlautendes Makro in irgendeinem Unterformular verwenden will. Das Dateiauswahlfeld ist von dort aus über **oEvent.Source.Model** zu erreichen. Das Formular ist **Parent** (Elternteil) zu dem Dateiauswahlfeld. Der Name des Formulars spielt also keine Rolle. Vom Formular aus geht dann der Zugriff auf das Feld mit dem Namen «GraphischesFeld». Über dieses Feld werden die Dateipfade der Bil-
der abgespeichert. Jetzt wird in das Feld die URL der ausgesuchten Datei geschrieben. Damit die
URL betriebssystemunabhängig verwendet werden kann, wird der in dem Dateiauswahlfeld ste-
hende Text vorher über `ConvertToUrl` in eine allgemeingültige URL-Schreibweise überführt.
In der Tabelle der Datenbank steht jetzt ein Pfad mit absoluter Schreibweise: `file:///...`
Wenden allerdings über ein graphisches Kontrollfeld Pfadeingaben eingelesen, so erfolgt die Auf-
nahme in relater Pfadangabe. Um dies zu bewerkstelligen müsste nachgebessert werden. Die Prozedur für diesen Zweck sieht wesentlich umfangreicher aus, da ein Vergleich der Pfadeingabe
der aktuell.

**Dokumente mit relativer Pfadangabe verknüpfen**
Das folgende Makro ist an die Eigenschaft «Text modifiziert» des Dateiauswahlfeldes gebunden.

```vba
SUB PfadAuslesen
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oFeld2 AS OBJECT
    DIM arUrl_Start() AS STRING
    DIM ar() AS STRING
    DIM ar1() AS STRING
    DIM ar2() AS STRING
    DIM stText AS STRING
    DIM stUrl_gesamt AS STRING
    DIM stUrl_Text AS STRING
    DIM stUrl AS STRING
    DIM stUrl_weg AS STRING
    DIM ink AS INTEGER
    DIM i AS INTEGER
    oDoc = thisComponent
    oDrawpage = oDoc.Drawpage
    oForm = oDrawpage.Forms.getByName("Formular")
    oFeld = oForm.getByName("GraphischesFeld")
    oFeld2 = oForm.getByName("Dateiauswahl")

    Zuerst werden, wie bei allen Prozeduren, die Variablen deklariert. Anschließend werden die ent-
sprechenden Felder aufgesucht, die für die Aufnahme des Pfades wichtig sind. Der gesamte
anschließende Code wird nur dann ausgeführt, wenn in der Dateiauswahl auch Inhalt steht, das
Feld also z.B. nach einem Datensatzwechsel nicht einfach geleert wird.

    IF oFeld2.Text <> "" THEN
        arUrl_Start = split(oDoc.Parent.Url,oDoc.Parent.Title)
        ar = split(ConvertToUrl(oFeld2.Text),"/"
        stText = ""
        FOR i = LBound(ar()) TO UBound(ar()) - 1
            stText = stText & ar(i) & "/"
        NEXT
        stText = Left(stText,Len(stText)-1)
    END IF

    Der Pfad zur Datenbankdatei wird ausgelesen. Dies erfolgt, wie vorher schon erklärt, indem zuerst
die gesamte Url gelesen wird und anschließend der Dateiname als Trenner eines Arrays dafür
sorgt, dass im ersten Teil des Arrays nur der direkte Pfad steht.

    Anschließend werden alle Elemente des Pfades, der über die Dateiauswahl ausgesucht wurde, in
das Array `ar` eingelesen. Als Trenner dient `/`. Das gelingt unter Linux direkt. Unter Windows muss
der Inhalt von oFeld2.Text auf jeden Fall noch in eine URL umgewandelt werden, die mit dem
Slash und nicht mit dem Backslash \ arbeitet.

    Die Trennung dient dazu, für die Ermittlung des Pfades zu der Datei einfach nur den Dateinamen
hinten abzuschneiden. Daher wird im nächsten Schritt der Pfad zu der aufzurufenden Datei in der
Variablen `stText` wieder zusammengesetzt. Die Schleife endet allerdings nicht mit dem höchsten
Wert des vorher erstellten Arrays `ar`, sondern schon beim erreichen der vorhergehenden Wertes.

    FOR i = LBound(ar()) TO UBound(ar()) - 1
        stText = stText & ar(i) & "/"
    NEXT
    stText = Left(stText,Len(stText)-1)
```

*Datenbank-Aufgaben: Bilder und Dokumente mit Base verarbeiten* 366

```
arUrl_Start(0) = Left(arUrl_Start(0),Len(arUrl_Start(0))-1)
```

In ar1 werden alle Elemente gespeichert, die den Pfad zu der aufzurufenden Datei wieder geben. In ar2 sind alle Elemente gespeichert, die den Pfad zur Datenbankdatei wieder geben. Anschließen wird das gesamte Array ar2 schrittweise für einen Vergleich in einer Schleife durchlaufen.

```
FOR i = LBound(ar2()) TO UBound(ar2())
    IF i <= UBound(ar1()) THEN
        IF ar2(i) = ar1(i) AND ink = 0 THEN
            stUrl_weg = stUrl_weg & ar1(i) & "/"
        ELSE
            stUrl = stUrl & ".//"
            ink = 1
        END IF
    ELSE
        stUrl = stUrl & ",/"
    END IF
NEXT
```

Ist irgendwann ein Unterschied zwischen den beiden Arrays aufgetaucht, so wird für jeden unterschiedlichen Wert die Bezeichnung für «setze ein Verzeichnis höher an» in der Variablen stUrl gespeichert.

```
stUrl_gesamt = ConvertToUrl(oFeld2.Text)
oFeld.boundField.UpdateString(stUrl & Right(stUrl_gesamt,Len(stUrl_gesamt) - Len(stUrl_weg)))
```

Verknüpfte Bilder und Dokumente anzeigen

Verknüpfte Bilder können direkt in dem kleinen graphischen Kontrollfeld angezeigt werden. Besser wäre aber ein größere Anzeige, um auch Details erkennen zu können.

Datenbank-Aufgaben: Bilder und Dokumente mit Base verarbeiten 367
Dokumente lassen sich standardmäßig in Base überhaupt nicht anzeigen.

Um dennoch eine Anzeige zu ermöglichen bedarf es wieder einer Makrolösung. Dieses Makro wird über einen Button in dem Formular gestartet, in dem das graphische Kontrollfeld liegt.

```
SUB Betrachten(oEvent AS OBJECT)
  DIM oDoc AS OBJECT
  DIM oForm AS OBJECT
  DIM oFeld AS OBJECT
  DIM oShell AS OBJECT
  DIM stUrl AS STRING
  DIM stFeld AS STRING
  DIM arUrl_Start() =
  oDoc = thisComponent
  oForm = oEvent.Source.Model.Parent
  oFeld = oForm.getByName("GraphischesFeld")
  stUrl = oFeld.BoundField.getString

  Das graphische Kontrollfeld im Formular wird aufgesucht. Da in der Tabelle nicht das Bild selbst, sondern nur der Pfad als Text gespeichert wird, wird hier über getString dieser Text ausgelesen.


  arUrl_Start = split(oDoc.Parent.Url,oDoc.Parent.Title)
  oShell = createUnoService("com.sun.star.system.SystemShellExecute")
  stFeld = convertToUpper(arUrl_Start(0) + stUrl)
  oShell.execute(stFeld,,0)
END SUB
```


### Dokumente in die Datenbank einlesen

Beim Einlesen der Dokumente sollten folgende Bedingungen immer im Auge behalten werden:

- Je größer die Dokumente sind, desto schwerfälliger wird die Datenbank. Bei entsprechend großen Dokumenten, vor allem Bildern, ist deshalb eine externe Datenbank der internen Datenbank vorzuziehen.
- In Dokumenten kann ebenso wenig wie in Bildern recherchiert werden. Sie werden als Binärdaten gespeichert und können in einem Feld gespeichert werden, das einem Bildfeld entspricht.
- Dokumente, die in die HSQLDB-Datenbank eingelesen werden, können nur über den Makroweg auch wieder ausgelesen werden. Ein SQL-Weg ist für die interne Datenbank nicht erreichbar.

Datenbank-Aufgaben: Bilder und Dokumente mit Base verarbeiten
Die folgenden Makros zum Ein- und Auslesen bauen dabei auf eine Tabelle auf, die neben der Datei im Binärformat eine Beschreibung der Datei und den ursprünglichen Dateinamen enthalten soll. Schließlich wird der Dateiname ja nicht mit abgespeichert und sollte beim Auslesen der Datei aber darauf Schlüsse zulassen, um welchen Dateityp es sich denn handelt. Nur dann kann die Datei auch zum Lesen durch andere Programme einwandfrei erkannt werden.

Die Tabelle enthält die folgenden Felder:

<table>
<thead>
<tr>
<th>Feldname</th>
<th>Feldtyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Integer</td>
<td>Die ID ist Primärschlüssel dieser Tabelle.</td>
</tr>
<tr>
<td>Beschreibung</td>
<td>Text</td>
<td>Beschreibung des Dokumentes, Suchbegriffe ...</td>
</tr>
<tr>
<td>Datei</td>
<td>Bild</td>
<td>Nimmt den binären Inhalt des Bildes oder der Datei auf.</td>
</tr>
<tr>
<td>Dateiname</td>
<td>Text</td>
<td>Soll die Bezeichnung der Datei mit Dateiendung abspeichern. Wichtig für das spätere Auslesen.</td>
</tr>
</tbody>
</table>

Das Formular zum Einlesen und wieder Ausgeben der Dateien sieht so aus:

ID
Beschreibung: Geld, Scheine und Münzen

Bild oder Datei

Dateiname: Geld.png

Datenbank-Aufgaben: Bilder und Dokumente mit Base verarbeiten


Das folgende Makro für das Einlesen der Dateien wird über Eigenschaften: Dateiauswahl → Ereignisse → Text modifiziert ausgelöst.

```vba
SUB DateiEinlesen_mitName(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oFeld2 AS OBJECT
    DIM oFeld3 AS OBJECT
    DIM oStream AS OBJECT
    DIM oSimpleFileAccess AS OBJECT
    DIM stUrl AS STRING
    DIM stName AS STRING
    oFeld = oEvent.Source.Model
    oForm = oFeld.Parent
    oFeld2 = oForm.getByName("Dateiname")
    oFeld3 = oForm.getByName("GraphischesFeld")
    IF oFeld.Text <> "" THEN
        stUrl = ConvertToUrl(oFeld.Text)
        ar = split(stUrl,"/"
        stName = ar(UBound(ar))
    END IF
END SUB
```
Da das Makro über das auslösende Ereignis die Position der anderen Formularfelder ermittelt, muss nicht besonders überprüft werden, ob die Felder nun in einem Formular oder Unterformular liegen. Alle Felder müssen lediglich im gleichen Formular positioniert sein.


Das Feld «GraphischesFeld» speichert die Daten in dem gemeinsamen Feld für Bilder und Dateien ab.


Die Daten sind jetzt wie eine normale Eingabe in die Formularfelder eingefügt. Wird das Formular einfach geschlossen, so sind die Daten noch nicht abgespeichert. Die Speicherung erfolgt erst bei Betätigung des Speicherbuttons in der Navigationsleiste oder automatisch bei der Navigation zum nächsten Datensatz.

### Bildnamen ermitteln

Bei dem obigen Verfahren wurde kurz erwähnt, dass der Name der Datei bei der Eingabe über das graphische Kontrollfeld so nicht ermittelt werden kann. Hier jetzt kurz ein Makro zur Ermittlung des Dateinamens, das zum obigen Formular passt. Der Dateiname lässt sich nicht sicher durch ein Ereignis ermitteln, das mit dem grafischen Kontrollfeld direkt verbunden ist. Deswegen wird das Makro über **Formular-Eigenschaften → Ereignisse → Vor der Datensatzaktion** gestartet.

```vba
SUB BildnamenAuslesen(oEvent AS OBJECT)
or = oEvent.Source
IF InStr(or.ImplementationName, "ODatabaseForm") THEN
oFeld = or.getByName("GraphischesFeld")
oFeld2 = or.getByName("Dateiname")
IF oFeld.ImageUrl <> "" THEN
stUrl = ConvertToUrl(oFeld.ImageUrl)
ar = split(stUrl,"/"
stName = ar(UBound(ar))
oFeld2.BoundField.updateString(stName)
END IF
END IF
END SUB
```

Vor der Datensatzaktion werden zwei Implementationen mit unterschiedlichem Implementationsnamen ausgeführt. Das Formular ist am einfachsten über die Implementation erreichbar, das in seinem Namen als `ODatabaseForm` bezeichnet wird.

---

**Datenbank-Aufgaben: Bilder und Dokumente mit Base verarbeiten**
In dem graphischen Kontrollfeld ist die URL der Datenquelle über die ImageUrl erreichbar. Diese URL wird ausgelesen, der Dateinamen wie in der vorhergehenden Prozedur «DateiEinlesen mit Name» ausgelesen und in das Feld «DateiName» übertragen.

**Bildnamen aus dem Speicher entfernen**

Wird nach dem Ablauf des obigen Makros zum nächsten Datensatz gewechselt, so ist der Pfad zu dem ursprünglichen Bild weiter vorhanden. Würde jetzt eine allgemeine Datei über das Dateiauswahlfeld eingesehen, so würde der Name der Datei ohne das folgende Makro einfach durch den Namen der zuletzt eingesehenen Bilddatei überschrieben.


Das Makro wird über Formular-Eigenschaften → Ereignisse → Nach der Datensatzaktion gestartet.

```vba
SUB BildnamenZuruecksetzen(oEvent AS OBJECT)
    oForm = oEvent.Source
    IF InStr(oForm.ImplementationName, "ODatabaseForm") THEN
        oFeld = oForm.getByName("GraphischesFeld")
        IF oFeld.ImageUrl <> "" THEN
            oFeld.ImageUrl = ""
        END IF
    END IF
END SUB
```


**Bilder und Dokumente auslesen und anzeigen**

Für Dateien wie für die Bilder in Originalgröße gilt, dass der Button Datei mit externem Programm betrachten ausgelöst werden muss. Dann werden die Dateien in das temporäre Verzeichnis ausgelesen und anschließend über das mit der Endung verknüpfte Programm des Betriebssystems angezeigt.

Das Makro wird über Eigenschaften: Schaltfläche → Ereignisse → Aktion ausführen gestartet.

```vba
SUB DateiAuslesen_mitName(oEvent AS OBJECT)
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oFeld2 AS OBJECT
    DIM oStream AS OBJECT
    DIM oShell AS OBJECT
    DIM oPath AS OBJECT
    DIM oSimpleFileAccess AS OBJECT
    DIM stName AS STRING
    DIM stPfad AS STRING
    DIM stFeld AS STRING
    oForm = oEvent.Source.Model.Parent
    oFeld = oForm.getByName("GraphischesFeld")
    oFeld2 = oForm.getByName("Dateiname")
    stName = oFeld2.Text
    IF stName = "" THEN
        stName = "Datei"
    END IF
    oStream = oFeld.BoundField.getBinaryStream
    oPath = createUnoService("com.sun.star.util.PathSettings")
    stPfad = oPath.Temp & "/" & stName
    oSimpleFileAccess.writeFile(stPfad, oStream)
```
oShell = createUnoService("com.sun.star.system.SystemShellExecute")
stFeld = convertToUrl(stPfad)
oShell.execute(stFeld,,0)

END SUB

Die Lage der anderen betroffenen Felder im Formular wird abhängig von dem auslösenden Button ermittelt. Fehlt ein Dateiname, so wird der ausgelesenen Datei einfach der Name «Datei» zugeordnet.


**Codeschnipsel**

Die Codeschnipsel erwachsen aus Anfragen innerhalb von Mailinglisten. Bestimmte Problemstellungen tauchen dort auf, die vielleicht gut als Lösungen innerhalb der eigenen Datenbankentwürfe genutzt werden können.

**Aktuelles Alter ermitteln**

Aus einem Datum soll mittels Abfrage das aktuelle Alter ermittelt werden. Siehe hierzu die Funktionen im Anhang zu diesem Base-Handbuch.

```
SELECT DATEDIFF('yy',"Geburtsdatum",CURDATE()) AS "Alter" FROM "Person"
```


```
SELECT CASEWHEN
( DAYOFYEAR("Geburtsdatum") > DAYOFYEAR(CURDATE())) ,
DATEDIFF ('yy',"Geburtsdatum",CURDATE())-1,
DATEDIFF ('yy',"Geburtsdatum",CURDATE()))
AS "Alter" FROM "Person"
```

Jetzt wird das aktuelle Alter in Jahren ausgegeben.

Über 'CASEWHEN' könnte dann auch in einem weiteren Feld der Text 'Heute Geburtstag' ausgegeben werden, wenn DAYOFYEAR("Geburtsdatum") = DAYOFYEAR(CURDATE()).


Mit

```
CASEWHEN ( (MONTH("Geburtsdatum") > MONTH(CURDATE())) OR ((MONTH("Geburtsdatum") = MONTH(CURDATE())) AND (DAY("Geburtsdatum") > DAY(CURDATE()))),
DATEDIFF('yy','Geburtsdatum',CURDATE())-1,
DATEDIFF('yy','Geburtsdatum',CURDATE())
```

*Datenbank-Aufgaben: Codeschnipsel* 372
wird das Ziel erreicht. Solange der Monat des Geburtsdatum größer ist als der aktuelle Monat wird
auf jeden Fall von der Jahresdifferenz 1 Jahr abgezogen. Ebenfalls 1 Jahr abgezogen wird, wenn
zwar der Monat gleich ist, der Tag im Monat des Geburtsdatums aber größer ist als der Tag im
aktuellen Monat. Leider ist diese Eingabe für die GUI nicht verständlich. Erst **SQL-Kommando
direkt ausführen** läßt die Abfrage erfolgreich absetzen. So ist also unsere Abfrage nicht mehr
editierbar. Die Abfrage soll aber weiter editierbar sein; also gilt es die GUI zu überlisten:

```sql
CASE
    WHEN MONTH("Geburtsdatum") > MONTH(CURDATE())
    THEN DATEDIFF('yy',"Geburtsdatum",CURDATE())-1
    WHEN (MONTH("Geburtsdatum") = MONTH(CURDATE()) AND DAY("Geburtsdatum")
    > DAY(CURDATE())
    THEN DATEDIFF('yy',"Geburtsdatum",CURDATE())-1
    ELSE DATEDIFF('yy',"Geburtsdatum",CURDATE())
END
```

Auf diese Formulierung reagiert die GUI nicht mit einer Fehlermeldung. Das Alter wird jetzt auch in
Schaltjahren genau ausgegeben und die Abfrage bleibt editierbar.

**Geburtstage in den nächsten Tagen anzeigen**

Mit Hilfe von einigen kleinen Berechnungsgriffen läßt sich auch aus einer Tabelle z.B. auslesen,
er in den nächsten 8 Tagen Geburtstag hat.

```sql
SELECT *
FROM "Tabelle"
WHERE
    DAYOFYEAR("Datum") BETWEEN DAYOFYEAR(CURDATE()) AND
    DAYOFYEAR(CURDATE()) + 7
    OR DAYOFYEAR("Datum") < 7 -
    DAYOFYEAR(CAST(YEAR(CURDATE()) || '-12-31' AS DATE)) +
    DAYOFYEAR(CURDATE())
```

Die Abfrage zeigt alle Datensätze an, deren Datumseintrag zwischen dem momentanen Jahrestag
und den folgenden 7 Tagen liegt.

Damit auch am Jahresende entsprechend 8 Tage angezeigt werden müssen die Anfangstage des
Jahres grundsätzlich auch überprüft werden. Diese Überprüfung findet aber nur für die Jahrestage
statt, die höchstens den Wert 7 abzüglich des Jahrestages des letzten Datumswertes des aktuel-
en Jahres (meist 365) zuzüglich des Jahrestages des aktuellen Datumswertes. Liegt das aktuelle
Datum mehr als 7 Tage vom Jahresende entfernt, so ist der Gesamtwert also < 1. Kein Eintrag in
der Tabelle hat so ein Datum. Diese Teilbedingung wird dann nicht erfüllt.

In der oben stehenden Formulierung werden bei Schaltjahren noch Unstimmigkeiten hervortreten,
da sich dann ab dem 29.2. die Jahrestagzählung verschiebt. Der Code, um dieses zu berücksichti-
gen, gestaltet sich um einiges umfangreicher:

```sql
SELECT *
FROM "Tabelle"
WHERE
    CASE
        WHEN DAYOFYEAR(CAST(YEAR("Datum") || '-12-31' AS DATE)) = 366
            AND DAYOFYEAR("Datum") > 60 THEN DAYOFYEAR("Datum") - 1
        ELSE
            DAYOFYEAR("Datum")
    END
```

Datenbank-Aufgaben: Codeschnipsel
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())

Schaltjahre sind daran zu erkennen, dass das gesamte Jahr 366 und nicht 365 Tage hat. Dies wird für die entsprechenden Unterscheidungen genutzt.

Zum einen muss jeder Datumswert darauf überprüft werden, ob er in einem Schaltjahr liegt und außerdem für die korrekte Zählung der 60. Tag im Jahr ist (31 Tage im Januar und 29 Tage im Februar). Für diesen Fall werden alle darauffolgende DAYOFYEAR – Werte für das Datum um 1 herabgestuft. Dann trifft ein 1.3. eines Schaltjahres wieder exakt auf einen 1.3. eines normalen Jahres.

Zum anderen muss auch beim aktuellen Jahr (CURDATE()) überprüft werden, ob es sich um ein Schaltjahr handelt. Auch hier wird entsprechend die Tageszahl um 1 herabgestuft.

Auch eine Anzeige von Terminen für die nächsten 8 Tage ist so noch nicht einwandfrei möglich, da das Jahr in der Abfrage noch ausgeklammert ist. Dies wäre aber über eine einfache zusätzliche Bedingung YEAR("Datum") = YEAR(CURDATE()) für das aktuelle bzw. YEAR("Datum") = YEAR(CURDATE()) + 1 für das zukünftige Jahr zu erfüllen.

Tage zu Datumswerten addieren

Bei der Ausleihe von Medien möchte vielleicht der Entleiher genau wissen, an welchem Tag denn die Rückgabe des Mediums erfolgen soll. Leider bietet die interne HSQLDB dazu nicht die Funktion DATEADD() an, wie dies in vielen der externen Datenbanken und auch bei Firebird der Fall ist. Hier lässt sich für einen begrenzten Zeitraum der folgende Umweg nutzen.


Auch über Abfragetechniken ist die Erstellung so einer Übersicht möglich. Diese Übersicht kann, wenn eine Filtertabelle genutzt wird, in Startdatum und ihrem Umfang an Datumswerten sogar gesteuert werden:

```
SELECT DISTINCT CAST
  ("Y"."Nr" + (SELECT "Jahr" FROM "Filter" WHERE "ID" = True) - 1 || '-' ||
CASEWHEN("M"."Nr" < 10, '0' || "M"."Nr", '' || "M"."N\r") || '-' ||
CASEWHEN("D"."Nr" < 10, '0' || "D"."Nr", '' || "D"."N\r")
AS DATE ) AS "Datum"
FROM "NrBis31" AS "D", "NrBis31" AS "M", "NrBis31" AS "Y"
WHERE "Y"."Nr" <= (SELECT "Jahre" FROM "Filter" WHERE "ID" = True)
AND "M"."Nr" <= 12 AND "D"."Nr" <= 31
```


Hierauf greift die folgende Ansicht zu:

```
SELECT "a"."Datum",
(SELECT COUNT(*) FROM "Ansicht_Datum" WHERE "Datum" <=
 "a"."Datum")
AS "lfdNr"
FROM "Ansicht_Datum" AS "a"
```

Über eine Zeilennummerierung wird den Datumswerten eine Nummer hinzugefügt. Da aus Ansichten sowieso keine Daten gelöscht werden können muss hier natürlich nicht extra ein Schreibschutz bemüht werden.

Mit einer Abfrage kann jetzt zu einem bestimmten Datum ermittelt werden, wie z.B. das Datum in 14 Tagen lauten wird:

```
SELECT "a"."Leih_Datum",
(SELECT COUNT(*) FROM "Datum" WHERE "ID" =
 (SELECT "ID" FROM "Datum" WHERE "Datum" = "a"."Leih_Datum")+14)
AS "Rueckgabedatum"
FROM "Ausleihe" AS "a"
```


Leider wird bei der Anzeige in der Abfrage der Datumstyp nicht automatisch erkannt, so dass hier entsprechend formatiert werden muss. In einem Formular ist die entsprechende Anzeige aber dauerhaft speicherbar, so dass auch bei jeder Abfrage entsprechend ein Datumswert ausgegeben wird.

Eine Direktvariante zur Ermittlung des Datumswertes ist sogar auf kürzerem Wege möglich. Hierbei wird der Startwert genutzt, an dem die interne Tageszählung von Base beginnt:

```
SELECT "Leih_Datum",
DATEDIFF( 'dd', '1899-12-30', "Leih_Datum" ) + 14
AS "Rueckgabedatum"
FROM "Tabelle"
```
Der ermittelte Zahlenwert kann im Formular als Datum über ein formatiertes Feld dargestellt werden. Es ist allerdings nur mit großem Aufwand möglich, ihn einfach für weiteren Abfragecode in ein SQL-Datum zu übertragen.

Zeiten zu Zeitstempeln addieren


Im Gegensatz zur Addition von Tagen zu einem Datum tritt bei den Zeiten allerdings ein Problem auf, das anfangs vielleicht gar nicht auffällt:

```sql
SELECT "DatumZeit"
  DATEDIFF( 'ss', '1899-12-30', "DatumZeit" ) / 86400.0000000000 +
  36/24 AS "DatumZeit+36Stunden"
FROM "Tabelle"
```


**Hinweis**
Das Nulldatum am 30.12.1899 ist vermutlich deshalb entstanden, weil das Jahr 1900 im Gegensatz zur üblichen 4-Jahres-Rechnung kein Schaltjahr war. So ist der Tag '1' der internen Rechnung auf den 31.12.1899 vorverlegt worden und eben nicht der 1.1.1900.


Zu diesem Ergebnis wird die Stundenzahl als Bruchteil eines Tages hinzugezählt. Die berechnete Ziffer lässt sich bei entsprechender Formatierung in der Anfrage darstellen. Dort wird die Formatierung aber leider nicht gespeichert. Innerhalb eines Formulars mit formatierbarem Feld oder innerhalb eines Berichts kann sie aber dauerhaft mit entsprechender Formatierung übernommen werden.

Sollen Minuten oder Sekunden addiert werden, so ist hier auch immer darauf zu achten, dass die Angaben der Minuten und Sekunden als Tagesbruchteile angegeben werden.

Lieg das Datum in den Monaten November, Dezember, Januar usw., dann fällt bei der Rechnung erst einmal nichts auf. Die Darstellung ist stimmig, zum Zeitstempel von 20.01.2015 13:00:00 eine Zeit von 36 Stunden addiert ergibt den neuen (nur dargestellten) Stempel 22.01.2015 01:00:00. Anders verhält es sich bei 20.04.2015 13:00:00. Da wird anschließend der 22.04.2015 00:00:00 ausgegeben. Das liegt an der Sommerzeit, die der Berechnung hier in die Quere kommt. Die gerade bei der Zeitumstellung «verlorene» oder «gewonnene» Stunde lässt sich bei einer Übergang nicht berücksichtigen. Innerhalb einer Zeitzone kann allerdings auf verschiedene Weise eine entsprechende Berechnung mit «korrektem» Ergebnis durchgeführt werden. Hier die dafür einfachere Variante:

```sql
SELECT "DatumZeit"
  DATEDIFF( 'dd', '1899-12-30', "DatumZeit" ) +
```
Statt die Stunden, Minuten und Sekunden aus dem Ursprungsdatum zu beziehen werden sie hier im Verhältnis zum aktuellen Datum dargestellt. Am 20.05.2015 steht die Uhr auf 13:00 Uhr, würde aber ohne die Sommerzeit 12:00 Uhr anzeigen. Die Funktion HOUR berücksichtigt die Sommerzeit und gibt 13 Stunden als Stundenanteil aus. Damit kann dann der Stundenanteil korrekt zum Tagesanteil addiert werden. Auf gleiche Art und Weise geschieht dies mit Hilfe des Minutenanteils und des Sekundenanteils. Anschließend werden die zu addierenden Stunden wieder als Tagesbruchteil addiert und das Ganze mit Hilfe der Zellenformatierung als berechneter Zeitstempel ausgegeben.

Zwei Dinge sollten aber bei diesen Berechnungen nie aus den Augen verloren werden:


Laufenden Kontostand nach Kategorien ermitteln


Mit dieser Abfrage wird bei jedem neuen Datensatz direkt ausgerechnet, welcher Kontostand jetzt erreicht wurde. Dabei bleibt die Abfrage editierbar, da das Feld "Saldo" durch eine korrelierende Unterabfrage erstellt wurde. Die Abfrage gibt den Kontostand in Abhängigkeit von dem automatisch erzeugten Primärschlüssel "ID" an. Kontostände werden aber eigentlich täglich ermittelt. Es muss also eine Datumsabfrage her.


Datenbank-Aufgaben: Codeschnipsel
Hiermit wird eine editierbare Abfrage erzeugt, in der neben den Eingabefeldern (Datum, Betrag, Konto_ID) der Kontoname, der jeweilige Kontostand und der Saldo insgesamt erscheint. Da sich die korrelierenden Unterabfragen teilweise auch auf vorhergehende Eingaben stützen ("Datum" <= "a"."Datum") werden nur Neueingaben reibungslos dargestellt. Änderungen eines vorhergehenden Datensatzes machen sich zuerst einmal nur in diesem Datensatz bemerkbar. Die Abfrage muss aktualisiert werden, damit auch die davon abhängigen späteren Berechnungen neu durchgeführt werden.

**Zeilenummerierung**


```sql
SELECT "ID", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "ID" <= "a"."ID" ) AS "lfdNr." FROM "Tabelle" AS "a"
```

Das Feld "ID" wird ausgelesen, im zweiten Feld wird durch eine korrelierende Unterabfrage festgestellt, wie viele Feldinhalte von "ID" kleiner oder gleich dem aktuellen Feldinhalt sind. Daraus wird dann die laufende Zeilennummer erstellt.

Wird einer Abfrage eine Bedingung hinzugefügt oder beruht eine Abfrage aus einer Verbindung von mehreren Tabellen, so ist diese Bedingung bzw. die Verbindung von mehreren Tabellen auch in der korrelierenden Unterabfrage hinzuzufügen. Andernfalls stimmt die Zählung nicht:

```sql
SELECT "Name", ( SELECT COUNT( "ID" ) FROM "Name" WHERE "GebDat" > '1995-12-31' AND "ID" <= "a"."ID" ) AS "lfdNr." FROM "Name" AS "a"
```

Die gesamte Bedingung der äußeren Abfrage muss in der korrelierenden Unterabfrage wiederholt werden. Hinzu kommt in der korrelierenden Unterabfrage noch die Bedingung, mit der sich die korrelierende Unterabfrage auf den aktuellen Datensatz bezieht.

Auch eine Nummerierung für entsprechende Gruppierungen ist möglich:

```sql
SELECT "ID", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "ID" <= "a"."ID" AND "RechnungsID" = "a"."RechnungsID" ) AS "lfdNr." FROM "Tabelle" AS "a"
```

Hier gibt es in einer Tabelle verschiedene Rechnungsnummern ("RechnungsID"). Für jede Rechnungsnummer wird separat die Anzahl der Felder "ID" aufsteigend nach der Sortierung des Feldes "ID" wiedergegeben. Das erzeugt für jede Rechnung die Nummerierung von 1 an aufwärts.

Soll die aktuelle Sortierung der Abfrage mit der Zeilennummer übereinstimmen, so ist die Art der Sortierung entsprechend abzubilden. Dabei muss die Sortierung allerdings einen eindeutigen Wert ergeben. Sonst liegen 2 Nummern auf dem gleichen Wert.

```sql
SELECT "Name", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "Name" <= "a"."Name" ) AS "lfdNr." FROM "Tabelle" AS "a"
```

ORDER BY "Name"

Nur wenn ein eindeutiger Index für das Feld "Name" definiert wurde ist hier eine eindeutige Sortierung und auch eine klare Nummerierung zu erwarten. Tauchen aber zwei gleiche Namen auf, so erhalten beide die gleiche Nummer.

```sql
SELECT "Name", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "Name"||"ID" <= "a"."Name"||"a"."ID" ) AS "lfdNr." FROM "Tabelle" AS "a"
```

Datenbank-Aufgaben: Codeschnipsel
ORDER BY "Name"||"ID"

Hier wird der Inhalt des Feldes "Name" mit dem Inhalt des Feldes "ID" zusammengefügt. Die Sortierung ist jetzt eindeutig, die laufende Nummer entsprechend auch. Bei genauerer Betrachtung fällt allerdings auf, dass die Sortierung der "ID" Natürlich jetzt nicht der Zah lensortierung folgt. Stattdessen wird z.B. die Zahl '9' nach der Zahl '10' einsortiert, da 9 größer als 1 ist. Wer dem noch abhelfen will muss entsprechend mit Leerstellen oder Nullen auffüllen:

```
SELECT "Name", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "Name"||RIGHT('000000000'||"ID",10) <= "a"."Name"||RIGHT('000000000'||"a"."ID",10) ) AS "lfdNr."
FROM "Tabelle" AS "a"
ORDER BY "Name"||RIGHT('000000000'||"ID",10)
```


Sollen Zeilen für eine Abfrage nummeriert werden, die sich nicht nur auf eine Tabelle bezieht oder mit einer zusätzlichen Bedingung versehen ist, so muss in der korrelierenden Unterabfrage die komplette Bedingung mit allen betroffenen Tabellen enthalten sein. Nur dann kann die Unterabfrage die gleichen Daten verarbeiten wie die äußere Abfrage.

Gleiche Werte können natürlich genutzt werden, wenn z.B. die Platzierung bei einem Wettkampf wiedergegeben werden soll, da hier gleiche Wettkampfergebnisse auch zu einer gleichen Platzierung führen. Damit die Platzierung allerdings so wiedergegeben wird, dass bei einer gleichen Platzierung der nachfolgende Wert ausgelassen wird, ist die Abfrage etwas anders zu konstruieren:

```
SELECT "ID", ( SELECT COUNT( "ID" ) + 1 FROM "Tabelle" WHERE "Zeit" < "a"."Zeit" ) AS "Platz" FROM "Tabelle" AS "a"
```


Schwieriger wird es, wenn neben der Platzierung auch eine Zeilennummerierung erfolgen soll. Dies kann z.B. sinnvoll sein, um mehrere Datensätze in einer Zeile zusammen zu fassen.

```
SELECT "ID", ( SELECT COUNT( "ID" ) + 1 FROM "Tabelle" WHERE "Zeit" < "a"."Zeit" ) AS "Platz",
CASE WHEN
( SELECT COUNT( "ID" ) + 1 FROM "Tabelle" WHERE "Zeit" = "a"."Zeit" ) = 1
THEN ( SELECT COUNT( "ID" ) + 1 FROM "Tabelle" WHERE "Zeit" < "a"."Zeit" )
ELSE ( SELECT COUNT( "ID" ) + 1 FROM "Tabelle" WHERE "Zeit" < "a"."Zeit" ) + COUNT( "ID" ) FROM "Tabelle" WHERE "Zeit" = "a"."Zeit" "ID" < "a"."ID"
END
AS "Zeilennummer" FROM "Tabelle" AS "a"
```

Diese Abfrage gibt also solange identische Werte zur Platzierung heraus, wie keine zweite Person mit der gleichen Zeit existiert. Existiert eine zweite Person mit der gleichen Zeit, so wird nach der ID entschieden, welche Person die geringere Zeilennummer enthält.

Diese Zeilensortierung entspricht übrigens der, die die Datenbanken anwenden. Wird z.B. eine Reihe Datensätze nach dem Namen sortiert, so erfolgt die Sortierung bei gleichen Datensätzen nicht nach dem Zufallsprinzip sondern aufsteigend nach dem Primärschlüssel, der ja schließlich eindeutig ist. Es lässt sich auf diese Weise also über die Nummerierung eine Sortierung der Datensätze abbilden.

Die Zeilennummerierung ist auch eine gute Voraussetzung, um einzelne Datensätze als einen Datensatz zusammen zu fassen. Wird eine Abfrage zur Zeilennummerierung als Ansicht erstellt, so kann darauf mit einer weiteren Abfrage problemlos zugegriffen werden. Als einfaches Beispiel hier noch einmal die erste Abfrage zur Nummerierung, nur um ein Feld ergänzt:

```
SELECT "ID", "Name", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "ID" <= "a"."ID" ) AS "lfdNr." FROM "Tabelle" AS "a"
```

Aus dieser Abfrage wird jetzt die Ansicht 'Ansicht1' erstellt. Die Abfrage, mit der z.B. die ersten 3 Namen zusammen in einer Zeile erscheinen können, lautet:

```
SELECT "Name" AS "Name_1", ( SELECT "Name" FROM "Ansicht1" WHERE "lfdNr." = 2 ) AS "Name_2", ( SELECT "Name" FROM "Ansicht1" WHERE "lfdNr." = 3 ) AS "Name_3" FROM "Ansicht1" WHERE "lfdNr." = 1
```

Auf diese Art und Weise können mehrere Datensätze nebeneinander als Felder dargestellt werden. Allerdings läuft diese Nummerierung einfach vom ersten bis zum letzten Datensatz durch.

Sollen alle Personen zu einem Nachnamen zugeordnet werden, so ließe sich das folgendermaßen realisieren:

```
SELECT "ID", "Name", "Nachname", ( SELECT COUNT( "ID" ) FROM "Tabelle" WHERE "ID" <= "a"."ID" AND "Nachname" = "a"."Nachname") AS "lfdNr." FROM "Tabelle" AS "a"
```

Jetzt kann über die erstellte Ansicht eine entsprechende Familienzusammenstellung erfolgen:

```
SELECT "Nachname", "Name" AS "Name_1", ( SELECT "Name" FROM "Ansicht1" WHERE "lfdNr." = 2 AND "Nachname" = "a"."Nachname") AS "Name_2", ( SELECT "Name" FROM "Ansicht1" WHERE "lfdNr." = 3 AND "Nachname" = "a"."Nachname") AS "Name_3" FROM "Ansicht1" AS "a" WHERE "lfdNr." = 1
```

In einem Adressbuch ließen sich so alle Personen einer Familie ("Nachnamen") zusammenfassen, damit jede Adresse nur einmal für ein Anschreiben berücksichtigt würde, aber alle Personen, an die das Anschreiben gehen soll, aufgeführt würden.

Da es sich um keine fortwährende Schleifenfunktion handelt ist hier allerdings Vorsicht geboten. Schließlich wird die Grenze der parallel als Felder angezeigten Datensätze durch die Abfrage im obigen Beispiel z.B. auf 3 begrenzt. Diese Grenze wurde willkürlich gesetzt. Weitere Namen tauchen nicht auf, auch wenn die Nummerierung der "lfdNr." größer als 3 ist.

In seltenen Fällen ist so eine Grenze aber auch klar nachvollziehbar. Soll z.B. ein Kalender erstellt werden, so können mit diesem Verfahren die Zeilen die Wochen des Jahres darstellen, die Spalten die Wochentage. Da im ursprünglichen Kalender nur das Datum über den Inhalt entscheidet werden durch die Zeilennummerierung immer die Tage einer Woche durchnummeriert und nach Wochen im Jahr als Datensatz später ausgegeben. Spalte 1 gibt dann Montag wieder, Spalte 2 Dienstag usw. Die Unterabfrage endet also jeweils bei der "lfdNr." = 7. Damit lassen sich dann im Bericht alle sieben Wochentage nebeneinander anzeigen und eine entsprechende Kalenderübersicht erstellen.

**Datenbank-Aufgaben: Codeschnipsel**
Zeilenumbruch durch eine Abfrage erreichen


Der Zeilenumbruch innerhalb einer Abfrage erfolgt durch 'Char(13)'. Beispiel:

```
SELECT "Vorname" || ' ' || "Nachname" || Char(13) || "Straße" || Char(13) || "Ort"
FROM "Tabelle"
```

Dies erzeugt nachher:

```
Vorname Nachname
Straße
Ort
```


Je nach Betriebssystem kann es auch notwendig sein, zusätzlich zu Char(13) auch Char(10) in den Code mit aufzunehmen:

```
SELECT "Vorname" || ' ' || "Nachname" || Char(13) || Char(10) || "Straße" || Char(13) || Char(10) || "Ort"
FROM "Tabelle"
```

Gruppieren und Zusammenfassen


Dieses Beispiel entspricht in vielen Teilen dem der Zeilennummerierung. Die Gruppierung zu einem gemeinsamen Feld stellt hier eine Ergänzung dar.

<table>
<thead>
<tr>
<th>Nachname</th>
<th>Vornamen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Müller</td>
<td>Karin</td>
</tr>
<tr>
<td>Schneider</td>
<td>Gerd</td>
</tr>
<tr>
<td>Müller</td>
<td>Egon</td>
</tr>
<tr>
<td>Schneider</td>
<td>Volker</td>
</tr>
<tr>
<td>Müller</td>
<td>Monika</td>
</tr>
<tr>
<td>Müller</td>
<td>Rita</td>
</tr>
</tbody>
</table>

Wird nach der Abfrage zu:

<table>
<thead>
<tr>
<th>Nachname</th>
<th>Vornamen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Müller</td>
<td>Karin, Egon, Monika, Rita</td>
</tr>
<tr>
<td>Schneider</td>
<td>Gerd, Volker</td>
</tr>
</tbody>
</table>

Dieses Verfahren kann in Grenzen auch in der HSQLDB nachgestellt werden. Das folgende Beispiel bezieht sich auf eine Tabelle "Name" mit den Feldern "ID", "Vorname" und "Nachname". Die folgende Abfrage wird zuerst an die Tabelle gestellt und als Ansicht "Ansicht_Gruppe" gespeichert:

SELECT DISTINCT "Nachname",
( SELECT "Vorname" FROM "Ansicht_Gruppe" WHERE "Nachname" = "a"."Nachname" AND "GruppenNr" = 1 ) ||
IFNULL( ( SELECT ', ' || "Vorname" FROM "Ansicht_Gruppe" WHERE "Nachname" = "a"."Nachname" AND "GruppenNr" = 2 ), '' ) ||
IFNULL( ( SELECT ', ' || "Vorname" FROM "Ansicht_Gruppe" WHERE "Nachname" = "a"."Nachname" AND "GruppenNr" = 3 ), '' ) ||
IFNULL( ( SELECT ', ' || "Vorname" FROM "Ansicht_Gruppe" WHERE "Nachname" = "a"."Nachname" AND "GruppenNr" = 4 ), '' ) ||
IFNULL( ( SELECT ', ' || "Vorname" FROM "Ansicht_Gruppe" WHERE "Nachname" = "a"."Nachname" AND "GruppenNr" = 5 ), '' )
AS "Vornamen"
FROM "Ansicht_Gruppe" AS "a"

Durch Unterabfragen werden nacheinander die Vornamen zu Gruppenmitglied 1, 2 usw. abgefragt und zusammengefasst. Ab der 2. Unterabfrage muss abgesichert werden, dass 'NULL'-Werte nicht die Zusammenfassung auf 'NULL' setzen. Deshalb wird bei einem Ergebnis von 'NULL' stattdessen '' angezeigt.
Makros
Allgemeines zu Makros

Prinzipiell kommt eine Datenbank unter Base ohne Makros aus. Irgendwann kann aber das Bedürfnis kommen,

- bestimmte Handlungsschritte zu vereinfachen (Wechsel von einem Formular zum anderen, Aktualisierung von Daten nach Eingabe in einem Formular ...),
- Fehleingaben besser abzusichern,
- häufigere Aufgaben zu automatisieren oder auch
- bestimmte SQL-Anweisungen einfacher aufzurufen als mit dem separaten SQL-Editor.

Es ist natürlich jedem selbst überlassen, wie intensiv er/sie Makros in Base nutzen will. Makros können zwar die Bedienbarkeit verbessern, sind aber auch immer mit geringen, bei ungünstiger Programmierung auch stärkeren Geschwindigkeitseinbußen des Programms verbunden. Es ist immer besser, zuerst einmal die Möglichkeiten der Datenbank und die vorgesehenen Einstellmöglichkeiten in Formularen auszuprobieren, bevor mit Makros zusätzliche Funktionen bereitgestellt werden. Makros sollten deshalb auch immer wieder mit größeren Datenbanken getestet werden, um ihren Einfluss auf die Verarbeitungsgeschwindigkeit abschätzen zu können.

Makros werden über den Weg Extras → Makros → Makros verwalten → LibreOffice Basic... erstellt. Es erscheint ein Fenster, das den Zugriff auf alle Makros ermöglicht. Makros für Base werden meistens in dem Bereich gespeichert, der dem Dateinamen der Base-Datei entspricht.

Über den Button Neu im Fenster «LibreOffice Basic Makros» wird ein zweites Fenster geöffnet. Hier wird lediglich nach der Bezeichnung für das Modul (Ordner, in dem das Makro abgelegt wird) gefragt. Der Name kann gegebenenfalls auch noch später geändert werden.

Sobald dies bestätigt wird, erscheint der Makro-Editor und auf seiner Eingabefläche wird bereits der Start und das Ende für eine Prozedur angegeben:

```
REM ***** BASIC *****
Sub Main
End Sub
```

Um Makros, die dort eingegeben wurden, nutzen zu können, sind folgende Schritte notwendig:

Die Datenbankdatei muss nach der Erstellung des ersten Makro-Moduls einmal geschlossen und anschließend wieder geöffnet werden.

Einige Grundprinzipien zur Nutzung des Basic-Codes in LibreOffice:

- Eigentlich wird zwischen Prozeduren (beginnend mit SUB) und Funktionen (beginnend mit FUNCTION) unterschieden. Prozeduren sind ursprünglich Programmabschnitte ohne Rückgabewert, Funktionen können Werte zurückgeben, die anschließend weiter ausgewertet werden können. Inzwischen ist diese Unterscheidung weitgehend irrelevant; man spricht allgemein von Methoden oder Routinen – mit oder ohne Rückgabewert. Auch eine Prozedur kann einen Rückgabewert (außer «Variant») erhalten; der wird einfach in der Definition zusätzlich festgelegt:

```vbnet
SUB myProcedure AS INTEGER
END SUB
```

Zu weiteren Details siehe auch das Handbuch 'Erste Schritte Makros mit LibreOffice'.

<table>
<thead>
<tr>
<th>Hinweis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makros in diesem Kapitel sind entsprechend den Vorgaben aus dem Makro-Editor von LibreOffice eingefärbt:</td>
</tr>
<tr>
<td>Makro-Bezeichner</td>
</tr>
<tr>
<td>Makro-Kommentar</td>
</tr>
<tr>
<td>Makro-Operator</td>
</tr>
<tr>
<td>Makro-Reservierter-Ausdruck</td>
</tr>
<tr>
<td>Makro-Zahl</td>
</tr>
<tr>
<td>Makro-Zeichenkette</td>
</tr>
</tbody>
</table>

**Makros in Base**

**Makros benutzen**

Der «direkte Weg» über Extras → Makros → Makros ausführen ist zwar auch möglich, aber bei Base-Makros nicht üblich. Ein Makro wird in der Regel einem Ereignis zugeordnet und durch dieses Ereignis gestartet.

- Ereignisse eines Formular
- Bearbeitung einer Datenquelle innerhalb des Formulars
- Wechsel zwischen verschiedenen Kontrollfeldern
- Reaktionen auf Maßnahmen innerhalb eines Kontrollfelds

Der «direkte Weg» ist vor allem dann nicht möglich – auch nicht zu Testzwecken –, wenn eines der Objekte thisComponent (siehe den Abschnitt «Zugriff auf das Formular») oder oEvent (siehe den Abschnitt «Zugriff auf Elemente eines Formulars») benutzt wird.
Makros zuweisen


Ereignisse eines Formulars beim Öffnen oder Schließen des Fensters

Maßnahmen, die beim Öffnen oder Schließen eines Formulars erledigt werden sollen, werden so registriert:

- Rufen Sie im Formularentwurf über Extras → Anpassen das Register Ereignisse auf.
- Wählen Sie das passende Ereignis aus. Bestimmte Makros lassen sich nur starten, wenn die Ansicht wurde erzeugt gewählt ist. Andere Makros wie z.B. das Erzeugen eines Vollbild-Formulars kann über Dokument öffnen gestartet werden.
- Suchen Sie über die Schaltfläche Makro das dafür definierte Makro und bestätigen Sie diese Auswahl.
- Unter Speichern in ist das Formular anzugeben.

Dann kann diese Zuweisung mit OK bestätigt werden.
Ereignisse eines Formulars bei geöffnetem Fenster

Nachdem das Fenster für die gesamten Inhalte des Formulars geöffnet wurde kann auf die einzelnen Elemente des Formulars zugegriffen werden. Hierzu gehören auch die dem Formular zugeordneten Formularelemente.


Hinweis

Der Gebrauch des Begriffes «Formular» ist bei Base leider nicht eindeutig. Der Begriff wird zum einen für das Fenster benutzt, das zur Eingabe von Daten geöffnet wird. Zum anderen wird der Begriff für das Element genutzt, das mit diesem Fenster eine bestimmte Datenquelle (Tabelle oder Abfrage) verbindet.

Es können auf einem Formularfenster sehr wohl mehrere Formulare mit unterschiedlichen Datenquelle untergebracht sein. Im Formularnavigator steht zuerst immer der Begriff «Formulare», dem dann in einem einfachen Formular lediglich ein Formular untergeordnet wird.
Ereignisse innerhalb eines Formulars
Alle anderen Makros werden bei den Eigenschaften von Teilformularen und Kontrollfeldern über das Register Ereignisse registriert.

- Öffnen Sie (sofern noch nicht geschehen) das Fenster mit den Eigenschaften des Kontrollfelds.
- Wählen Sie im Register Ereignisse das passende Ereignis aus.
  - Um die Datenquelle zu bearbeiten, gibt es vor allem die Ereignisse, die sich auf Datensatz oder Aktualisieren oder Zurücksetzen beziehen.
  - Zu Schaltflächen oder einer Auswahl bei Listen- oder Optionsfeldern gehört in erster Linie das Ereignis Aktion ausführen.
  - Alle anderen Ereignisse hängen vom Kontrollfeld und der gewünschten Maßnahme ab.
- Durch einen Klick auf den rechts stehenden Button ... wird das Fenster «Aktion zuweisen» geöffnet.
- Über die Schaltfläche Makro wird das dafür definierte Makro ausgewählt.

Über mehrfaches OK wird diese Zuweisung bestätigt.
Bestandteile von Makros

In diesem Abschnitt sollen einige Teile der Makro-Sprache erläutert werden, die in Base – vor allem bei Formularen – immer wieder benutzt werden. (Soweit möglich und sinnvoll, werden dabei die Beispiele der folgenden Abschnitte benutzt.)

Der «Rahmen» eines Makros

Die Definition eines Makros beginnt mit dem Typ des Makros – SUB oder FUNCTION – und endet mit END SUB bzw. END FUNCTION. Einem Makro, das einem Ereignis zugewiesen wird, können Argumente (Werte) übergeben werden; sinnvoll ist aber nur das Argument oEvent. Alle anderen Routinen, die von einem solchen Makro aufgerufen werden, können abhängig vom Zweck mit oder ohne Rückgabewert definiert werden und beliebig mit Argumenten versehen werden.

```vba
SUB Ausleihe_actuallisieren
END SUB

SUB Zu_Formular_von_Formular(oEvent AS OBJECT)
END SUB

FUNCTION Loeschen_bestaetigen(oEvent AS OBJECT) AS BOOLEAN
Loeschen_bestaetigen = FALSE
END FUNCTION
```


Variablen definieren


```vba
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
```

Für die Namensvergabe stehen nur Buchstaben (A–Z oder a–z), Ziffern und der Unterstrich '_' zur Verfügung, aber keine Umlaute oder Sonderzeichen. (Unter Umständen ist das Leerzeichen zulässig. Sie sollten aber besser darauf verzichten.) Das erste Zeichen muss ein Buchstabe sein.


Arrays definieren
Gerade für Datenbanken ist die Sammlung von mehreren Variablen in einem Datensatz von Bedeutung. Werden mehrere Variablen zusammen in einer gemeinsamen Variablen gespeichert, so wird dies als ein Array bezeichnet. Ein Array muss definiert werden, bevor Daten in das Array geschrieben werden können.

\[ \text{DIM arDaten}() \]
erzeugt eine leeres Array.

\[ \text{arDaten} = \text{array} ("Lisa","Schmidt") \]
So wird ein Array auf eine bestimmte Größe von 2 Elementen festgelegt und gleichzeitig mit Daten versehen.

Über
\[ \text{print arDaten}(0), \text{arDaten}(1) \]
Wenden die beiden definierten Inhalte auf dem Bildschirm ausgegeben. Die Zählung für die Felder beginnt hier mit 0.

\[ \text{DIM arDaten}(2) \]
\[ \text{arDaten}(0) = "Lisa" \]
\[ \text{arDaten}(1) = "Schmidt" \]
\[ \text{arDaten}(2) = "Köln" \]
Dies erstellt ein Array, in dem 3 Elemente beliebigen Typs gespeichert werden können, also z.B. ein Datensatz mit den Variablen «Lisa» «Schmidt» «Köln». Mehr passt leider in dieses Array nicht hinein. Sollen mehr Elemente gespeichert werden, so muss das Array vergrößert werden. Wird während der Laufzeit eines Makros allerdings die Größe eines Arrays einfach nur neu definiert, so ist das Array anschließend leer wie eben ein neues Array.

\[ \text{ReDIM Preserve arDaten}(3) \]
\[ \text{arDaten}(3) = "18.07.2003" \]
Durch den Zusatz \text{Preserve} werden die vorherigen Daten beibehalten, das Array also tatsächlich zusätzlich um die Datumseingabe, hier in Form eines Textes, erweitert.

Das oben aufgeführte Array kann leider nur einen einzelnen Satz an Daten speichern. Sollen stattdessen, wie in einer Tabelle einer Datenbank, mehrere Datensätze gespeichert werden, so muss dem Array zu Beginn eine zusätzliche Dimension hinzugefügt werden.

\[ \text{DIM arDaten}(2,1) \]
\[ \text{arDaten}(0,0) = "Lisa" \]
\[ \text{arDaten}(1,0) = "Schmidt" \]
\[ \text{arDaten}(2,0) = "Köln" \]
\[ \text{arDaten}(0,1) = "Egon" \]
\[ \text{arDaten}(1,1) = "Müller" \]
\[ \text{arDaten}(2,1) = "Hamburg" \]
Auch hier gilt bei einer Erweiterung über das vorher definierte Maß hinaus, dass der Zusatz \text{Preserve} die vorher eingeggebenen Daten mit übernimmt.

Zugriff auf das Formular
Das Formular liegt in dem momentan aktiven Dokument. Der Bereich, der dargestellt wird, wird als \text{drawpage} bezeichnet. Der Behälter, in dem alle Formulare aufbewahrt werden, heißt \text{forms} – im Formularnavigator ist dies sozusagen der oberste Begriff, an den dann sämtliche Formulare angehängt werden. Die o.g. Variablen erhalten auf diesem Weg ihre Werte:

\[ \text{oDoc} = \text{thisComponent} \]
\[ \text{oDrawpage} = \text{oDoc.drawpage} \]
\[ \text{oForm} = \text{oDrawpage.forms.getByName("Filter")} \]
Das Formular, auf das zugegriﬀen werden soll, ist hier mit dem Namen "Filter" versehen. Dies ist der Name, der auch im Formularnavigator in der obersten Ebene sichtbar ist. (Standardmäßig
erhält das erste Formular den Namen "MainForm".) Unterformulare liegen – hierarchisch angeordnet – innerhalb eines Formulars und können Schritt für Schritt erreicht werden:

```vba
DIM oSubForm AS OBJECT
DIM oSubSubForm AS OBJECT
oSubForm = oForm.getByName("Leserauswahl")
oSubSubForm = oSubForm.getByName("Leseranzeige")
```

Anstelle der Variablen in den «Zwischenstufen» kann man auch direkt zu einem bestimmten Formular gelangen. Ein Objekt der Zwischenstufen, das mehr als einmal verwendet wird, sollte selbstständig deklariert und zugewiesen werden. (Im folgenden Beispiel wird oSubForm nicht mehr benutzt.)

```vba
oForm = thisComponent.drawpage.forms.getByName("Filter")
oSubSubForm = oForm.getByName("Leserauswahl").getByName("Leseranzeige")
```

### Hinweis

Sofern ein Name ausschließlich aus Buchstaben und Ziffern besteht (keine Umlaute, keine Leer- oder Sonderzeichen), kann der Name in der Zuweisung auch direkt verwendet werden:

```vba
oForm = thisComponent.drawpage.forms.Filter
oSubSubForm = oForm.Leserauswahl.Leseranzeige
```

 Anders als bei Basic sonst üblich, ist bei solchen Namen auf Groß- und Kleinschreibung genau zu achten.

Einen anderen Zugang zum Formular ermöglicht das auslösende Ereignis für das Makro.
Startet ein Makro über ein Ereignis des Formulars, wie z. B. **Formular-Eigenschaften → Vor der Datensatzaktion**, so wird das Formular selbst folgendermaßen erreicht:

```vba
SUB MakrobeispielBerechne(oEvent AS OBJECT)
oForm = oEvent.Source
...
END SUB
```

Startet ein Makro über ein Ereignis eines Formularfeldes, wie z. B. Eigenschaften: **Textfeld → Bei Fokusverlust**, so kann sowohl das Formularfeld als auch das Formular ermittelt werden:

```vba
SUB MakrobeispielBerechne(oEvent AS OBJECT)
oFeld = oEvent.Source.Model
oForm = oFeld.Parent
...
END SUB
```

Die Zugriffe über das Ereignis haben den Vorteil, dass kein Gedanke darüber verschwendet werden muss, ob es sich bei dem Formular um ein Hauptformular oder Unterformular handelt. Auch interessiert der Name des Formulars für die Funktionsweise des Makros nicht.

### Zugriff auf Elemente eines Formulars

In gleicher Weise kann man auf die Elemente eines Formulars zugreifen: Deklarieren Sie eine entsprechende Variable als **object** und suchen Sie das betreffende Kontrollfeld innerhalb des Formulars:

```vba
DIM btnOK AS OBJECT ' Button »OK»
btnOK = oSubSubForm.getByName("Schaltfläche 1") ' aus dem Formular Leseranzeige
```

Dieser Weg funktioniert immer dann, wenn bekannt ist, mit welchem Element das Makro arbeiten soll. Wenn aber im ersten Schritt zu prüfen ist, welches Ereignis das Makro gestartet hat, ist der o.g. Weg über **oEvent** sinnvoll. Dann wird die Variable innerhalb des Makro-"Rahmens" deklariert und beim Start des Makros zugewiesen. Die Eigenschaft **Source** liefert immer dasjenige Element, das das Makro gestartet hat; die Eigenschaft **Model** beschreibt das Kontrollfeld im Einzelnen:

```vba
SUB Auswahl_bestaetigen(oEvent AS OBJECT)
DIM btnOK AS OBJECT
btnOK = oEvent.Source.Model
```

---

*Makros: Makros in Base* 391
Mit dem Objekt, das man auf diesem Weg erhält, werden die weiteren angestrebten Maßnahmen ausgeführt.

Bitte beachten Sie, dass auch Unterformulare als Bestandteile eines Formulars gelten.

**Zugriff auf die Datenbank**

Normalerweise wird der Zugriff auf die Datenbank über Formulare, Abfragen, Berichte oder die Serienbrief-Funktion geregelt, wie es in allen vorhergehenden Kapiteln beschrieben wurde. Wenn diese Möglichkeiten nicht genügen, kann ein Makro auch gezielt die Datenbank ansprechen, wofür es mehrere Wege gibt.

**Die Verbindung zur Datenbank**

Das einfachste Verfahren benutzt dieselbe Verbindung wie das Formular, wobei `oForm` wie oben bestimmt wird:

```vba
DIM oConnection AS OBJECT
oConnection = oForm.activeConnection()
```

Oder man holt die Datenquelle, also die Datenbank, durch das Dokument und benutzt die vorhandene Verbindung auch für das Makro:

```vba
DIM oDatasource AS OBJECT
DIM oConnection AS OBJECT
oDatasource = thisComponent.Parent.dataSource
oConnection = oDatasource.getConnection("",""")
```

Ein weiterer Weg stellt sicher, dass bei Bedarf die Verbindung zur Datenbank hergestellt wird:

```vba
DIM oDatasource AS OBJECT
DIM oConnection AS OBJECT
oDatasource = thisComponent.Parent.CurrentController
IF NOT (oDatasource.isConnected()) THEN oDatasource.connect()
oConnection = oDatasource.ActiveConnection()
```

Die IF-Bedingung bezieht sich hier nur auf eine Zeile. Deshalb ist **END IF** nicht erforderlich.

Wenn das Makro durch die Benutzeroberfläche – nicht aus einem Formulardokument heraus – gestartet werden soll, ist folgende Variante geeignet:

```vba
DIM oDatasource AS OBJECT
DIM oConnection AS OBJECT
oDatasource = thisDatabaseDocument.CurrentController
IF NOT (oDatasource.isConnected()) THEN oDatasource.connect()
oConnection = oDatasource.ActiveConnection()
```

Der Zugriff auf Datenbanken außerhalb der aktuellen Datenbank ist folgendermaßen möglich:

```vba
DIM oDatabaseContext AS OBJECT
DIM oDatasource AS OBJECT
DIM oConnection AS OBJECT
oDatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
oDatasource = oDatabaseContext.getName("angemeldeter Name der Datenbank in LO")
oConnection = oDatasource.GetConnection("",""")
```

Auch die Verbindung zu nicht in LO angemeldete Datenbanken ist möglich. Hier muss dann lediglich statt des angemeldeten Namens der Pfad zur Datenbank mit «file:///..../Datenbank.odb» angegeben werden.

Ergänzende Hinweise zur Datenbankverbindung stehen im Abschnitt «Verbindung mit Datenbanken erzeugen».

---

*Makros: Makros in Base*
SQL-Befehle

Die Arbeit mit der Datenbank erfolgt über SQL-Befehle. Ein solcher muss also erstellt und an die Datenbank geschickt werden; je nach Art des Befehls wird das Ergebnis ausgewertet und weiterverarbeitet. Mit der Anweisung `createStatement` wird das Objekt dafür erzeugt:

```sql
DIM oSQL_Statement AS OBJECT  ' das Objekt, das den SQL-Befehl ausführt
DIM stSql AS STRING           ' Text des eigentlichen SQL-Befehls
DIM oResult AS OBJECT         ' Ergebnis für executeQuery
DIM iResult AS INTEGER        ' Ergebnis für executeUpdate
oSQL_Statement = oConnection.createStatement()
```

Um **Daten abzufragen**, wird mit dem Befehl die Methode `executeQuery` aufgerufen und ausgeführt; das Ergebnis wird anschließend ausgewertet. Tabellennamen und Feldnamen werden üblicherweise in doppelte Anführungszeichen gesetzt. Diese müssen im Makro durch weitere doppelte Anführungszeichen maskiert werden, damit sie im Befehl erscheinen.

```sql
stSql = "SELECT * FROM ""Tabelle1"
```

```sql
oResult = oSQL_Statement.executeQuery(stSql)
```

Um **Daten zu ändern** – also für **INSERT**, **UPDATE** oder **DELETE** – oder um die **Struktur der Datenbank** zu beeinflussen, wird mit dem Befehl die Methode `executeUpdate` aufgerufen und ausgeführt. Je nach Art des Befehls und der Datenbank erhält man kein nutzbares Ergebnis (ausgedrückt durch die Zahl 0) oder die Anzahl der bearbeiteten Datensätze.

```sql
stSql = "DROP TABLE ""Suchtmp"" IF EXISTS"
```

```sql
iResult = oSQL_Statement.executeUpdate(stSql)
```

Der Vollständigkeit halber sei noch ein Spezialfall erwähnt: Wenn `oSQL_Statement` unterschiedlich für **SELECT** oder für andere Zwecke benutzt wird, steht die Methode `execute` zur Verfügung. Diese benutzen wir nicht; wir verweisen dazu auf die API-Referenz.

**Vorbereitete SQL-Befehle mit Parametern**

In allen Fällen, in denen manuelle Eingaben der Benutzerin in einen SQL-Befehl übernommen werden, ist es einfacher und sicherer, den Befehl nicht als lange Zeichenkette zu erstellen, sondern ihn vorzubereiten und mit Parametern zu benutzen. Das vereinfacht die Formatierung von Zahlen, Datumsangaben und auch Zeichenketten (die ständigen doppelten Anführungszeichen entfallen) und verhindert Datenverlust durch böswillige Eingaben.

Bei diesem Verfahren wird zunächst das Objekt für einen bestimmten SQL-Befehl erstellt und vorbereitet:

```sql
DIM oSQL_Statement AS OBJECT  ' das Objekt, das den SQL-Befehl ausführt
DIM stSql AS STRING           ' Text des eigentlichen SQL-Befehls
stSql = "UPDATE Verfasser 
(" & "SET Nachname = ?, Vorname = ?" 
& "WHERE ID = ?"
```

```sql
oSQL_Statement = oConnection.prepareStatement(stSql)
```


Anschließend werden mit passenden Anweisungen die Werte übergeben und danach der SQL-Befehl ausgeführt. Die Werte werden hier aus Kontrollfeldern des Formulars übernommen, können aber auch aus anderen Makro-Elementen stammen oder im Klartext angegeben werden:

```sql
oSQL_Statement.setString(1, oTextfeld1.Text)   ' Text für den Nachnamen
oSQL_Statement.setString(2, oTextfeld2.Text)   ' Text für den Vornamen
oSQL_Statement.setLong(3, oZahlenfeld1.Value)  ' Wert für die betreffende ID
iResult = oSQL_Statement.executeUpdate
```

Die vollständige Liste der Zuweisungen findet sich im Abschnitt «**Parameter für vorbereitete SQL-Befehle**». 
Wer sich weiter über die Vorteile dieses Verfahrens informieren möchte, findet hier Erläuterungen:

- **SQL-Injection (Wikipedia)**
- **Why use PreparedStatement (Java JDBC)**
- **SQL-Befehle (Einführung in SQL)**

**Datensätze lesen und benutzen**

Es gibt – abhängig vom Zweck – mehrere Wege, um Informationen aus einer Datenbank in ein Makro zu übernehmen und weiter zu verarbeiten.

Bitte beachten Sie: Wenn hier von einem «Formular» gesprochen wird, kann es sich auch um ein Unterformular handeln. Es geht dann immer über dasjenige (Teil-) Formular, das mit einer bestimmten Datenmenge verbunden ist.

**Mithilfe des Formulars**

Der aktuelle Datensatz und seine Daten stehen immer über das Formular zur Verfügung, das die betreffende Datenmenge (Tabelle, Abfrage, SELECT) anzeigt. Dafür gibt es mehrere Methoden, die mit `get` und dem Datentyp bezeichnet sind, beispielsweise diese:

```basic
DIM ID AS LONG
DIM sName AS STRING
DIM dValue AS CURRENCY
DIM dEintritt AS NEW com.sun.star.util.Date
ID = oForm.getLong(1)
sName = oForm.getString(2)
dValue = oForm.getDouble(4)
dEintritt = oForm.getDate(7)
```

Bei allen diesen Methoden ist jeweils die Nummer der Spalte in der Datenmenge anzugeben – gezählt ab 1.

**Hinweis**

Bei allen Methoden, die mit Datenbanken arbeiten, wird ab 1 gezählt. Das gilt sowohl für Spalten als auch für Zeilen.

Möchte man anstelle der Spaltennummern mit den Spaltennamen der zugrundeliegenden Datenmenge (Tabelle, Abfrage, View) arbeiten, kann man die Spaltennummer über die Methode `findColumn` ermitteln – hier ein Beispiel zum Auffinden der Spalte "Name":

```basic
DIM sName AS STRING
nName = oForm.findColumn("Name")
sName = oForm.getString(nName)
```

Man erhält immer einen Wert des Typs der Methode, wobei die folgenden Sonderfälle zu beachten sind.

- Es gibt keine Methode für Daten des Typs `Decimal`, `Currency` o.ä., also für kaufmännisch exakte Berechnungen. Da Basic automatisch die passende Konvertierung vornimmt, kann ersatzweise `getDouble` verwendet werden.
- Datumsangaben können nicht nur mit dem Datentyp `DATE` definiert werden, sondern auch (wie oben) als `util.Date`. Das erleichtert u.a. Lesen und Ändern von Jahr, Monat, Tag.
- Bei ganzen Zahlen sind Unterschiede der Datentypen zu beachten. Im obigen Beispiel wird `getLong` verwendet; auch die Basic-Variable `ID` muss den Datentyp `Long` erhalten, da dieser vom Umfang her mit `Integer` aus der Datenbank übereinstimmt.

Die vollständige Liste dieser Methoden findet sich im Abschnitt «Datenzeilen bearbeiten».
Sollen Werte aus einem Formular für direkte Weiterverarbeitung in SQL genutzt werden (z.B. für die Eingabe der Daten in eine andere Tabelle), so ist es wesentlich einfacher, nicht nach dem Typ der Felder zu fragen.

Das folgende Makro, an Eigenschaften: Schaltfläche → Ereignisse → Aktion ausführen gekoppelt, liest das erste Feld des Formulars aus – unabhängig von dem für die Weiterverarbeitung in Basic erforderlichen Typ.

```
SUB WerteAuslesen(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM stFeld1 AS STRING
    oForm = oEvent.Source.Model.Parent
    stFeld1 = oForm.getString(1)
END SUB
```

Werden die Felder über `getString()` ausgelesen, so werden die Formatierung beibehalten, die für eine Weiterverarbeitung in SQL notwendig sind. Ein Datum, das in einem deutschsprachigen Formular als '08.03.15' dargestellt wird, wird so im Format '2015-03-08' ausgelesen und kann direkt in SQL weiter verarbeitet werden.

Die Auslesung in dem dem Typ entsprechenden Format ist nur erforderlich, wenn im Makro mit Werten weiter verarbeitet, z.B. mit ihnen gerechnet werden soll.

**Ergebnis einer Abfrage**

In gleicher Weise kann die Ergebnismenge einer Abfrage benutzt werden. Im Abschnitt «SQL-Befehle» steht die Variable `oResult` für diese Ergebnismenge, die üblicherweise so oder ähnlich ausgelesen wird:

```
WHILE oResult.next
    rem übernimm die benötigten Werte in Variablen
    stVar = oResult.getString(1)
    inVar = oResult.getLong(2)
    boVar = oResult.getBoolean(3)
    rem mach etwas mit diesen Werten
WEND
```


Soll nur der erste Datensatz ausgewertet werden, so wird mit

```
oResult.next
```

zuerst die Zeile auf diesen Datensatz bewegt und dann mit

```
stVar = oResult.getString(1)
```

z.B. der Inhalt des ersten Datenfeldes gelesen. Die Schleife entfällt hier.

Die Abfrage zu dem obigen Beispiel hat in der ersten Spalte einen Text, in der zweiten Spalte einen Integer-Zahlenwert (Integer aus der Datenbank entspricht `Long` in Basic) und in der dritten Spalte ein Ja/Nein-Feld. Die Felder werden durch den entsprechenden Indexwert angesprochen. Der Index für die Felder beginnt hier, im Gegensatz zu der sonstigen Zählung bei Arrays, mit dem Wert '1'.

In dem so erstellten Ergebnis ist allerdings keine Navigation möglich. Nur einzelne Schritte zum nächsten Datensatz sind erlaubt. Um innerhalb der Datensätze navigieren zu können muss der `ResultSetType` bei der Erstellung der Abfrage bekannt sein. Hierauf wird über

```
oSQL_Anweisung.ResultSetType = 1004
```

oder

```
oSQL_Anweisung.ResultSetType = 1005
```

zugegriffen. Der Typ `1004` - SCROLL_INTENSIVE erlaubt eine beliebige Navigation. Allerdings bleibt eine Änderung an den Originaldaten während des Auslesens unbemerkt. Der Typ `1005` -
SCROLL_SENSITIVE berücksichtigt zusätzlich gegebenenfalls Änderungen an den Originaldaten, die das Abfrageergebnis beeinflussen könnten.

Die Anzahl der Zeilen, die die Ergebnismenge enthält, kann nur nach Wahl der entsprechenden Typen so bestimmt werden:

```vba
DIM iResult AS LONG
IF oResult.Last = ' gehe zum letzten Datensatz, sofern möglich
   iResult = oResult.getRow ' die laufende Nummer ist die Anzahl
ELSE
   iResult = oResult.getRow
END IF
```

Mithilfe eines Kontrollfelds

Wenn ein Kontrollfeld mit einer Datenmenge verbunden ist, kann der Wert auch direkt ausgelesen werden, wie es im nächsten Abschnitt beschrieben wird. Das ist aber teilweise mit Problemen verbunden. Sicherer ist – neben dem Verfahren «Mithilfe des Formulars» – der folgende Weg, der für verschiedene Kontrollfelder gezeigt wird:

```vba
sValue = oTextField.BoundField.Text ' Beispiel für ein Textfeld
nValue = oNumericField.BoundField.Value ' Beispiel für ein numerisches Feld
dValue = oDateField.BoundField.Date ' Beispiel für ein Datumsfeld
```

Diese Methode nutzt die Verbindung zwischen dem (sichtbaren) Kontrollfeld und dem eigentlichen Inhalt der Datenmenge.

In einer Datenmenge navigieren

Im vorletzten Beispiel wurde mit der Methode Next von einer Zeile der Ergebnismenge zur nächsten gegangen. In gleicher Weise gibt es weitere Maßnahmen und Prüfungen, und zwar sowohl für die Daten eines Formulars – angedeutet durch die Variable oForm – als auch für eine Ergebnismenge. Beispielsweise kann man beim Verfahren «Automatisches Aktualisieren von Formularen» den vorher aktuellen Datensatz wieder markieren:

```vba
DIM loRow AS LONG
loRow = oForm.getRow ' notiere die aktuelle Zeilennummer
oForm.reload() ' lade die Datenmenge neu
oForm.absolute(loRow) ' gehe wieder zu der notierten Zeilennummer
```

Im Abschnitt «In einer Datenmenge navigieren» stehen alle dazu passenden Methoden.

---

**Hinweis**

Leider existiert seit den Anfängen von LO noch aus der OpenOffice-Zeit ein Bug in Formularen, der die aktuelle Zeilennummer bei Änderung von Daten in einem Formular auf '0' setzt: [https://bugs.documentfoundation.org/show_bug.cgi?id=82591](https://bugs.documentfoundation.org/show_bug.cgi?id=82591). Hier kann folgende Ermittlung der aktuellen Zeilennummer an Formular → Eigenschaften → Ereignisse → Nach dem Datensatzwechsel gebunden werden:

```vba
GLOBAL loRow AS LONG

SUB RowCounter(oEvent AS OBJECT)
   loRow = oEvent.Source.Row
END SUB
```

Die neue Zeilennummer wird ausgelesen und an die globale Variable loRow weitergegeben. Diese Variable ist am Anfang aller Module zu positionieren und bleibt so lange mit ihrem Inhalt gefüllt, bis entweder Base beendet wird oder eben die Position durch erneutes Auslösen von RowCounter überschrieben wird.

**Datensätze bearbeiten – neu anlegen, ändern, löschen**

Um Datensätze zu bearbeiten, müssen mehrere Teile zusammenpassen: Eine Information muss vom Anwender in das Kontrollfeld gebracht werden; das geschieht durch die Tastatureingabe. Anschließend muss die Datenmenge «dahinter» diese Änderung zur Kenntnis nehmen; das...
geschieht durch das Verlassen eines Feldes und den Wechsel zum nächsten Feld. Und schließlich muss die Datenbank selbst die Änderung erfahren; das erfolgt durch den Wechsel von einem Datensatz zu einem anderen.


Bitte beachten Sie, dass nur Datenmengen vom Typ «Tabelle» problemlos geändert werden können. Bei anderen Datenmengen ist dies nur unter besonderen Bedingungen möglich.

**Inhalt eines Kontrollfelds ändern**

Wenn es um die Änderung eines einzelnen Wertes geht, wird das über die Eigenschaft `BoundField` des Kontrollfelds mit einer passenden Methode erledigt. Anschließend muss nur noch die Änderung an die Datenbank weitergegeben werden. Beispiel für ein Datumsfeld, in das das aktuelle Datum eingetragen werden soll:

```vbnet
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() ' Weitergabe der Änderung an die Datenbank
```

Für `BoundField` wird diejenige der `updateXxx`-Methoden aufgerufen, die zum Datentyp des Feldes passt – hier geht es um einen `Date`-Wert. Als Argument wird der gewünschte Wert übergeben – hier das aktuelle Datum, konvertiert in die vom Makro benötigte Schreibweise.

**Zeile einer Datenmenge ändern**

Wenn mehrere Werte in einer Zeile geändert werden sollen, ist der vorstehende Weg ungeeignet. Zum einen müsste für jeden Wert ein Kontrollfeld existieren, was oft nicht gewünscht oder sinnvoll ist. Zum anderen muss man sich für jedes dieser Felder ein Objekt «holen». Der einfache und direkte Weg geht über das Formular, beispielsweise so:

```vbnet
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, "ein Text")
oForm.updateDouble(6, 3.14)
oForm.updateInt(7, 16)
oForm.updateRow() 
```

Für jede Spalte der Datenmenge wird die zum Datentyp passende `updateXxx`-Methode aufgerufen. Als Argumente werden die Nummer der Spalte (ab 1 gezählt) und der jeweils gewünschte Wert übergeben. Anschließend muss nur noch die Änderung an die Datenbank weitergegeben werden.

**Zeilen anlegen, ändern, löschen**

Die genannten Änderungen beziehen sich immer auf die aktuelle Zeile der Datenmenge des Formulars. Unter Umständen muss vorher eine der Methoden aus «In einer Datenmenge navigieren» aufgerufen werden. Es werden also folgende Maßnahmen benötigt:

1. Wählen den aktuellen Datensatz.
2. Ändere die gewünschten Werte, wie im vorigen Abschnitt beschrieben.
3. Bestätige die Änderungen mit folgendem Befehl:

```vbnet
oForm.updateRow() 
```
Als Sonderfall ist es auch möglich, die Änderungen zu verwerfen und den vorherigen Zustand wiederherzustellen:

```javascript
oForm.cancelRowUpdates()
```

Für einen **neuen Datensatz** gibt es eine spezielle Methode (vergleichbar mit dem Wechsel in eine neue Zeile im Tabellenkontrollfeld). Es werden also folgende Maßnahmen benötigt:

1. Bereite einen neuen Datensatz vor:
   ```javascript
   oForm.moveToInsertRow()
   ```
3. Bestätige die Neuaufnahme mit folgendem Befehl:
   ```javascript
   oForm.insertRow()
   ```

Für das **Löschen** eines Datensatzes gibt es einen einfachen Befehl; es sind also folgende Maßnahmen nötig:

1. Wähle – wie für eine Änderung – den gewünschten Datensatz und mache ihn zum aktuellen.
2. Bestätige die Löschung mit folgendem Befehl:
   ```javascript
   oForm.deleteRow()
   ```

**Tipp**

Damit eine Änderung in die Datenbank übernommen wird, ist sie durch `updateRow` bzw. `insertRow` ausdrücklich zu bestätigen. Während beim Betätigen des Speicher-Buttons die passende Funktion automatisch ermittelt wird, muss vor dem Abspeichern ermittelt werden, ob der Datensatz neu ist (`Insert`) oder ein bestehender Datensatz bearbeitet wurde (`Update`).

```javascript
IF oForm.isNew THEN
    oForm.insertRow()
ELSE
    oForm.updateRow()
END IF
```

**Kontrollfelder prüfen und ändern**

Neben dem Inhalt, der aus der Datenmenge kommt, können viele weitere Informationen zu einem Kontrollfeld gelesen, verarbeitet und geändert werden. Das betrifft vor allem die Eigenschaften, die im Kapitel «Formulare» aufgeführt werden. Eine Übersicht steht im Abschnitt «Eigenschaften bei Formularen und Kontrollfeldern».

In mehreren Beispielen des Abschnitts «Bedienbarkeit verbessern» wird die Zusatzinformation eines Feldes benutzt:

```vbnet
DIM stTag AS STRING
stTag = oEvent.Source.Model.Tag
```

Die Eigenschaft `Text` kann – wie im vorigen Abschnitt erläutert – nur dann sinnvoll geändert werden, wenn das Feld nicht mit einer Datenmenge verbunden ist. Aber andere Eigenschaften, die «eigentlich» bei der Formulardefinition festgelegt werden, können zur Laufzeit angepasst werden. Beispielsweise kann in einem Beschriftungsfeld die Textfarbe gewechselt werden, wenn statt einer Meldung ein Hinweis oder eine Warnung angezeigt werden soll:

```vbnet
SUB showWarning(oField AS OBJECT, iType AS INTEGER)
    SELECT CASE iType
    CASE 1
        oField.TextColor = RGB(0,0,255) ' 1 = blau
    CASE 2
        oField.TextColor = RGB(255,0,0) ' 2 = rot
    END CASE
END SUB
```

*Makros: Makros in Base* 398
CASE ELSE
  oField.TextColor = RGB(0,255,0) ' 0 = grün (weder 1 noch 2)
END SELECT
END SUB

Englische Bezeichner in Makros


Eigenschaften, die üblicherweise nur in der Formular-Definition festgelegt werden, stehen nicht in den Übersichten. Gleiches gilt für Methoden (Funktionen und Prozeduren), die nur selten verwendet werden oder für die kompliziertere Erklärungen nötig wären.

Die Übersichten nennen folgende Angaben:

• **Name** Bezeichnung der Eigenschaft oder Methode im Makro-Code
• **Datentyp** Einer der Datentypen von Basic
  Bei Funktionen ist der Typ des Rückgabewerts angegeben; bei Prozeduren entfällt diese Angabe.
• **L/S** Hinweis darauf, wie der Wert der Eigenschaft verwendet wird:
  - **L** nur Lesen
  - **S** nur Schreiben (Ändern)
  - **(L)** Lesen möglich, aber für weitere Verarbeitung ungeeignet
  - **(S)** Schreiben möglich, aber nicht sinnvoll
  - **L+S** geeignet für Lesen und Schreiben

Weitere Informationen finden sich vor allem in der *API-Referenz* mit Suche nach der englischen Bezeichnung des Kontrollfelds. Gut geeignet, um herauszufinden, welche Eigenschaften und Methoden denn eigentlich bei einem Element zur Verfügung stehen, ist auch das Tool *Xray*.

```basic
SUB Main(oEvent)
  Xray(oEvent)
END SUB
```

Hiermit wird die Erweiterung Xray aus dem Aufruf heraus gestartet.

**Eigenschaften bei Formularen und Kontrollfeldern**


**Schrift**

In jedem Kontrollfeld, das Text anzeigt, können die Eigenschaften der Schrift angepasst werden.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>FontName</td>
<td>string</td>
<td>L+S</td>
<td>Schriftart.</td>
</tr>
<tr>
<td>FontHeight</td>
<td>single</td>
<td>L+S</td>
<td>Schriftgröße.</td>
</tr>
<tr>
<td>FontWeight</td>
<td>single</td>
<td>L+S</td>
<td>Schriftstärke.</td>
</tr>
<tr>
<td>FontSlant</td>
<td>integer</td>
<td>L+S</td>
<td>Art der Schrägestellung.</td>
</tr>
<tr>
<td>FontUnderline</td>
<td>integer</td>
<td>L+S</td>
<td>Art der Unterstreichung.</td>
</tr>
<tr>
<td>FontStrikeout</td>
<td>integer</td>
<td>L+S</td>
<td>Art des Durchstreichens.</td>
</tr>
</tbody>
</table>
**Formular**  
Englische Bezeichnung: *Form*

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyFilter</td>
<td>boolean</td>
<td>L+S</td>
<td>Filter aktiviert.</td>
</tr>
<tr>
<td>Filter</td>
<td>string</td>
<td>L+S</td>
<td>Aktueller Filter für die Datensätze.</td>
</tr>
<tr>
<td>FetchSize</td>
<td>long</td>
<td>L+S</td>
<td>Anzahl der Datensätze, die «am Stück» geladen werden.</td>
</tr>
<tr>
<td>Row</td>
<td>long</td>
<td>L</td>
<td>Nummer der aktuellen Zeile.</td>
</tr>
<tr>
<td>RowCount</td>
<td>long</td>
<td>L</td>
<td>Anzahl der Datensätze.</td>
</tr>
</tbody>
</table>

**Einheitlich für alle Arten von Kontrollfeld**  
Englische Bezeichnung: *Control* – siehe auch *FormComponent*

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>string</td>
<td>L+(S)</td>
<td>Bezeichnung für das Feld.</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>L+S</td>
<td>Aktiviert: Feld kann ausgewählt werden.</td>
</tr>
<tr>
<td>EnableVisible</td>
<td>boolean</td>
<td>L+S</td>
<td>Sichtbar: Feld wird dargestellt.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>boolean</td>
<td>L+S</td>
<td>Nur lesen: Inhalt kann nicht geändert werden.</td>
</tr>
<tr>
<td>TabStop</td>
<td>boolean</td>
<td>L+S</td>
<td>Feld ist in der Tabulator-Reihenfolge erreichbar.</td>
</tr>
<tr>
<td>Align</td>
<td>integer</td>
<td>L+S</td>
<td>Horizontale Ausrichtung: 0 = links, 1 = zentriert, 2 = rechts</td>
</tr>
<tr>
<td>BackgroundColor</td>
<td>long</td>
<td>L+S</td>
<td>Hintergrundfarbe.</td>
</tr>
<tr>
<td>Tag</td>
<td>string</td>
<td>L+S</td>
<td>Zusatzinformation.</td>
</tr>
<tr>
<td>HelpText</td>
<td>string</td>
<td>L+S</td>
<td>Hilfetext als «Tooltip».</td>
</tr>
</tbody>
</table>

**Einheitlich für viele Arten von Kontrollfeld**

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>string</td>
<td>(L+S)</td>
<td>Inhalt des Feldes aus der Anzeige. Bei Textfeldern nach dem Lesen auch zur weiteren Verarbeitung geeignet, andernfalls nur in Ausnahmefällen.</td>
</tr>
<tr>
<td>Spin</td>
<td>boolean</td>
<td>L+S</td>
<td>Drehfeld eingeblendet (bei formatierten Feldern).</td>
</tr>
<tr>
<td>TextColor</td>
<td>long</td>
<td>L+S</td>
<td>Textfarbe.</td>
</tr>
<tr>
<td>DataField</td>
<td>string</td>
<td>L</td>
<td>Name des Feldes aus der Datenmenge</td>
</tr>
<tr>
<td>BoundField</td>
<td>object</td>
<td>L</td>
<td>Objekt, das die Verbindung zur Datenmenge herstellt und vor allem dem Zugriff auf den Feldinhalt dient.</td>
</tr>
</tbody>
</table>

**Textfeld – weitere Angaben**  
Englische Bezeichnung: *TextField*

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>string</td>
<td>L+S</td>
<td>Inhalt des Feldes aus der Anzeige.</td>
</tr>
<tr>
<td>MaxTextLen</td>
<td>integer</td>
<td>L+S</td>
<td>Maximale Textlänge.</td>
</tr>
<tr>
<td>Name</td>
<td>Datentyp</td>
<td>L/S</td>
<td>Eigenschaft</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-----</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>DefaultText</td>
<td>string</td>
<td>L+S</td>
<td>Standardtext.</td>
</tr>
<tr>
<td>MultiLine</td>
<td>boolean</td>
<td>L+S</td>
<td>Mehrzeilig oder einzeilig.</td>
</tr>
<tr>
<td>EchoChar</td>
<td>(integer)</td>
<td>L+S</td>
<td>Zeichen für Kennwörter (Passwort-Eingabe verstecken).</td>
</tr>
</tbody>
</table>

**Numerisches Feld**
Englische Bezeichnung: *NumericField*

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueMin</td>
<td>double</td>
<td>L+S</td>
<td>Minimalwert zur Eingabe.</td>
</tr>
<tr>
<td>ValueMax</td>
<td>double</td>
<td>L+S</td>
<td>Maximalwert zur Eingabe.</td>
</tr>
<tr>
<td>Value</td>
<td>double</td>
<td>L+(S)</td>
<td>Aktueller Wert für Werte aus der Datenmenge verwenden.</td>
</tr>
<tr>
<td>ValueStep</td>
<td>double</td>
<td>L+S</td>
<td>Intervall bei Verwendung mit Mausrad oder Drehfeld.</td>
</tr>
<tr>
<td>DefaultValue</td>
<td>double</td>
<td>L+S</td>
<td>Standardwert.</td>
</tr>
<tr>
<td>DecimalAccuracy</td>
<td>integer</td>
<td>L+S</td>
<td>Nachkommastellen.</td>
</tr>
<tr>
<td>ShowThousandsSeparator</td>
<td>boolean</td>
<td>L+S</td>
<td>Tausender-Trennzeichen anzeigen.</td>
</tr>
</tbody>
</table>

**Datumsfeld**
Englische Bezeichnung: *DateField*

Datumswerte werden als Datentyp *long* definiert und im ISO-Format YYYYMMDD angezeigt, also 20120304 für den 04.03.2012. Zur Verwendung dieses Typs zusammen mit `getDateTime` und `updateDateTime` sowie dem Typ `com.sun.star.util.Date` verweisen wir auf die Beispiele.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Datentyp ab LO 4.1.1</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateMin</td>
<td>long</td>
<td>com.sun.star.util.Date</td>
<td>L+S</td>
<td>Minimalwert zur Eingabe.</td>
</tr>
<tr>
<td>DateMax</td>
<td>long</td>
<td>com.sun.star.util.Date</td>
<td>L+S</td>
<td>Maximalwert zur Eingabe.</td>
</tr>
<tr>
<td>Date</td>
<td>long</td>
<td>com.sun.star.util.Date</td>
<td>L+(S)</td>
<td>Aktueller Wert für Werte aus der Datenmenge verwenden.</td>
</tr>
<tr>
<td>DateFormat</td>
<td>integer</td>
<td>L+S</td>
<td></td>
<td>Datumsformat nach Festlegung des Betriebssystems: 0 = kurze Datumsangabe (einfach) 1 = kurze Datumsangabe tt.mm.jj (Jahr zweistellig) 2 = kurze Datumsangabe tt.mm.jjjj (Jahr vierstellig) 3 = lange Datumsangabe (mit Wochentag und Monatsnamen) Weitere Möglichkeiten sind der Formulardefinition oder der API-Referenz zu entnehmen.</td>
</tr>
</tbody>
</table>

*Makros: Makros in Base*

401
<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Datentyp ab LO 4.1.1</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultDate</td>
<td>long</td>
<td>com.sun.-star.util.Date</td>
<td>L+S</td>
<td>Standardwert.</td>
</tr>
<tr>
<td>DropDown</td>
<td>boolean</td>
<td></td>
<td>L+S</td>
<td>Aufklappbaren Monatskalender anzeigen.</td>
</tr>
</tbody>
</table>

**Zeitfeld**  
Englische Bezeichnung: *TimeField*  
Auch Zeitwerte werden als Datentyp `long` definiert.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Datentyp ab LO 4.1.1</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeMin</td>
<td>long</td>
<td>com.sun.-star.util.Time</td>
<td>L+S</td>
<td>Minimalwert zur Eingabe.</td>
</tr>
<tr>
<td>TimeMax</td>
<td>long</td>
<td>com.sun.-star.util.Time</td>
<td>L+S</td>
<td>Maximalwert zur Eingabe.</td>
</tr>
<tr>
<td>Time</td>
<td>long</td>
<td>com.sun.-star.util.Time</td>
<td>L+(S)</td>
<td>Aktueller Wert nicht für Werte aus der Datenmenge verwenden.</td>
</tr>
<tr>
<td>TimeFormat</td>
<td>integer</td>
<td></td>
<td>L+S</td>
<td>Zeitformat:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = kurz als hh:mm (Stunde, Minute, 24 Stunden)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = lang als hh:mm:ss (dazu Sekunden, 24 Stunden)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 = kurz als hh:mm (12 Stunden AM/PM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3 = lang als hh:mm:ss (12 Stunden AM/PM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 = als kurze Angabe einer Dauer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5 = als lange Angabe einer Dauer</td>
</tr>
<tr>
<td>DefaultTime</td>
<td>long</td>
<td>com.sun.-star.util.Time</td>
<td>L+S</td>
<td>Standardwert.</td>
</tr>
</tbody>
</table>

**Währungsfeld**  
Englische Bezeichnung: *CurrencyField*  
Ein Währungsfeld ist ein numerisches Feld mit den folgenden zusätzlichen Möglichkeiten.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencySymbol</td>
<td>string</td>
<td>L+S</td>
<td>Währungssymbol (nur zur Anzeige).</td>
</tr>
<tr>
<td>PrependCurrencySymbol</td>
<td>boolean</td>
<td>L+S</td>
<td>Anzeige des Symbols vor der Zahl.</td>
</tr>
</tbody>
</table>

**Formatiertes Feld**  
Englische Bezeichnung: *FormattedControl*  
Ein formatiertes Feld wird wahlweise für Zahlen, Währungen oder Datum/Zeit verwendet. Sehr viele der bisher genannten Eigenschaften gibt es auch hier, aber mit anderer Bezeichnung.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentValue</td>
<td>variant</td>
<td>L</td>
<td>Aktueller Wert des Inhalts; der konkrete Datentyp hängt vom Inhalt des Feldes und dem Format ab.</td>
</tr>
<tr>
<td>EffectiveValue</td>
<td></td>
<td>L+(S)</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Datentyp</td>
<td>L/S</td>
<td>Eigenschaft</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-----</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>EffectiveMin</td>
<td>double</td>
<td>L+S</td>
<td>Minimalwert zur Eingabe.</td>
</tr>
<tr>
<td>EffectiveMax</td>
<td>double</td>
<td>L+S</td>
<td>Maximalwert zur Eingabe.</td>
</tr>
<tr>
<td>EffectiveDefault</td>
<td>variant</td>
<td>L+S</td>
<td>Standardwert.</td>
</tr>
<tr>
<td>FormatKey</td>
<td>long</td>
<td>L+(S)</td>
<td>Format für Anzeige und Eingabe. Es gibt kein einfaches Verfahren, das Format durch ein Makro zu ändern.</td>
</tr>
<tr>
<td>EnforceFormat</td>
<td>boolean</td>
<td>L+S</td>
<td>Formatüberprüfung: Bereits während der Eingabe sind nur zulässige Zeichen und Kombinationen möglich.</td>
</tr>
</tbody>
</table>

**Listenfeld**

Englische Bezeichnung: *ListBox*

Der Lese- und Schreibzugriff auf den Wert, der hinter der ausgewählten Zeile steht, ist etwas umständlich, aber möglich.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListSource</td>
<td>array of string</td>
<td>L+S</td>
<td>Datenquelle: Herkunft der Listeneinträge oder Name der Datenmenge, die die Einträge liefert.</td>
</tr>
<tr>
<td>ListSourceType</td>
<td>integer</td>
<td>L+S</td>
<td>Art der Datenquelle:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = Werteliste</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = Tabelle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 = Abfrage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 = Ergebnismenge eines SQL-Befehls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 = Ergebnis eines Datenbank-Befehls</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 = Feldnamen einer Datenbank-Tabelle</td>
</tr>
<tr>
<td>StringItemList</td>
<td>array of string</td>
<td>L</td>
<td>Listeneinträge, die zur Auswahl zur Verfügung stehen.</td>
</tr>
<tr>
<td>ItemCount</td>
<td>integer</td>
<td>L</td>
<td>Anzahl der vorhandenen Listeneinträge.</td>
</tr>
<tr>
<td>ValueItemList</td>
<td>array of string</td>
<td>L</td>
<td>Liste der Werte, die über das Formular an die Tabelle weitergegeben werden.</td>
</tr>
<tr>
<td>DropDownList</td>
<td>boolean</td>
<td>L+S</td>
<td>Aufklappbar.</td>
</tr>
<tr>
<td>LineCount</td>
<td>integer</td>
<td>L+S</td>
<td>Anzahl der angezeigten Zeilen im aufgeklappten Zustand.</td>
</tr>
<tr>
<td>MultiSelection</td>
<td>boolean</td>
<td>L+S</td>
<td>Mehrfachselektion vorgesehen.</td>
</tr>
<tr>
<td>SelectedItems</td>
<td>array of integer</td>
<td>L+S</td>
<td>Liste der ausgewählten Einträge, und zwar als Liste der Positionen in der Liste aller Einträge.</td>
</tr>
</tbody>
</table>

Das (erste) ausgewählte Element aus dem Listenfeld erhält man auf diesem Weg:

```plaintext```
oControl = oForm.getByName("Name des Listenfelds")
sEintrag = oControl.ValueItemList( oControl.SelectedItems(0) )
```

**Makros: Makros in Base**
Seit LO 4.1 wird direkt der Wert ermittelt, der bei einem Listenfeld an die Datenbank weitergegeben wird.

\[
oControl = oForm.getName("Name des Listenfelds") \]
\[
ID = oControl.getCurrentValue() \]

Mit `getCurrentValue()` wird also immer der Wert ausgegeben, der auch tatsächlich in der Tabelle der Datenbank abgespeichert wird. Dies ist beim Listenfeld von dem hiermit verknüpften gebundenen Feld (BoundField) abhängig.

Bis einschließlich LO 4.0 wurde hier immer der angezeigte Inhalt, nicht aber der an die darunterliegende Tabelle weitergegebene Wert wiedergegeben.

Soll für die Einschränkung einer Auswahlmöglichkeit die Abfrage für ein Listenfeld ausgetauscht werden, so ist dabei zu beachten, dass es sich bei dem Eintrag um ein «array of string» handelt:

```
SUB Listenfeldfilter
  DIM strSql(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.getName("MainForm")
  oFeld = oForm.getName("Listenfeld")
  strSql(0) = "SELECT ""Name"", ""ID"" FROM ""Filter_Name"" ORDER BY ""Name"
  oFeld.ListSource = strSql
  oFeld.refresh
END SUB
```

**Kombinationsfeld**

Englische Bezeichnung: *ComboBox*

Trotz ähnlicher Funktionalität wie beim Listenfeld weichen die Eigenschaften teilweise ab.

Hier verweisen wir ergänzend auf das Beispiel «*Kombinationsfelder als Listenfelder mit Eingabemöglichkeit*».

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autocomplete</td>
<td>boolean</td>
<td>L+S</td>
<td>Automatisch füllen.</td>
</tr>
<tr>
<td>StringItemList</td>
<td>array of string</td>
<td>L+S</td>
<td>Listeneinträge, die zur Auswahl zur Verfügung stehen.</td>
</tr>
<tr>
<td>ItemCount</td>
<td>integer</td>
<td>L</td>
<td>Anzahl der vorhandenen Listeneinträge.</td>
</tr>
<tr>
<td>DropDownList</td>
<td>boolean</td>
<td>L+S</td>
<td>Aufklappbar.</td>
</tr>
<tr>
<td>LineCount</td>
<td>integer</td>
<td>L+S</td>
<td>Anzahl der angezeigten Zeilen im aufgeklappten Zustand.</td>
</tr>
<tr>
<td>Text</td>
<td>string</td>
<td>L+S</td>
<td>Aktuell angezeigter Text.</td>
</tr>
<tr>
<td>DefaultText</td>
<td>string</td>
<td>L+S</td>
<td>Standardeintrag.</td>
</tr>
<tr>
<td>ListSource</td>
<td>string</td>
<td>L+S</td>
<td>Name der Datenquelle, die die Listeneinträge liefert.</td>
</tr>
<tr>
<td>ListSourceType</td>
<td>integer</td>
<td>L+S</td>
<td>Art der Datenquelle; gleiche Möglichkeiten wie beim Listenfeld (nur die Auswahl «Werteliste» wird ignoriert).</td>
</tr>
</tbody>
</table>

**Markierfeld, Optionsfeld**

Englische Bezeichnungen: *CheckBox* (Markierfeld) bzw. *RadioButton* (Optionsfeld; auch «Option Button» möglich)
<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>string</td>
<td>L+S</td>
<td>Titel (Beschriftung)</td>
</tr>
<tr>
<td>State</td>
<td>short</td>
<td>L+S</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = nicht ausgewählt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = ausgewählt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 = unbestimmt</td>
</tr>
<tr>
<td>MultiLine</td>
<td>boolean</td>
<td>L+S</td>
<td>Wortumbruch (bei zu langem Text).</td>
</tr>
</tbody>
</table>

**Maskiertes Feld**
Englische Bezeichnung: *PatternField*

Neben den Eigenschaften für «einfache» Textfelder sind folgende interessant.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditMask</td>
<td>string</td>
<td>L+S</td>
<td>Eingabemaske.</td>
</tr>
<tr>
<td>LiteralMask</td>
<td>string</td>
<td>L+S</td>
<td>Zeichenmaske.</td>
</tr>
<tr>
<td>StrictFormat</td>
<td>boolean</td>
<td>L+S</td>
<td>Formatüberprüfung bereits während der Eingabe.</td>
</tr>
</tbody>
</table>

**Tabellenkontrollfeld**
Englische Bezeichnung: *GridControl*

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>long</td>
<td>L</td>
<td>Anzahl der Spalten.</td>
</tr>
<tr>
<td>ElementNames</td>
<td>array of string</td>
<td>L</td>
<td>Liste der Spaltennamen.</td>
</tr>
<tr>
<td>HasNavigationBar</td>
<td>boolean</td>
<td>L+S</td>
<td>Navigationsleiste vorhanden.</td>
</tr>
<tr>
<td>RowHeight</td>
<td>long</td>
<td>L+S</td>
<td>Zeilenhöhe.</td>
</tr>
</tbody>
</table>

**Beschriftungsfeld**
Englische Bezeichnung: *FixedText* – auch *Label* ist üblich

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>string</td>
<td>L+S</td>
<td>Der angezeigte Text.</td>
</tr>
<tr>
<td>MultiLine</td>
<td>boolean</td>
<td>L+S</td>
<td>Wortumbruch (bei zu langem Text).</td>
</tr>
</tbody>
</table>

**Gruppierungsrahmen**
Englische Bezeichnung: *GroupBox*

Keine Eigenschaft dieses Kontrollfelds wird üblicherweise durch Makros bearbeitet. Wichtig ist der Status der einzelnen Optionsfelder.

**Schaltfläche**
Englische Bezeichnungen: *CommandButton* – für die grafische Schaltfläche *ImageButton*

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>string</td>
<td>L+S</td>
<td>Titel – Text der Beschriftung.</td>
</tr>
<tr>
<td>State</td>
<td>short</td>
<td>L+S</td>
<td>Standardstatus «ausgewählt» bei «Umschalten».</td>
</tr>
<tr>
<td>MultiLine</td>
<td>boolean</td>
<td>L+S</td>
<td>Wortumbruch (bei zu langem Text).</td>
</tr>
</tbody>
</table>
**Navigationsleiste**

Englische Bezeichnung: *NavigationBar*


<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>L/S</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>IconSize</td>
<td>short</td>
<td>L+S</td>
<td>Symbolgröße.</td>
</tr>
<tr>
<td>ShowPosition</td>
<td>boolean</td>
<td>L+S</td>
<td>Positionierung anzeigen und eingeben.</td>
</tr>
<tr>
<td>ShowNavigation</td>
<td>boolean</td>
<td>L+S</td>
<td>Navigation ermöglichen.</td>
</tr>
<tr>
<td>ShowRecordActions</td>
<td>boolean</td>
<td>L+S</td>
<td>Datensatzaktionen ermöglichen.</td>
</tr>
<tr>
<td>ShowFilterSort</td>
<td>boolean</td>
<td>L+S</td>
<td>Filter und Sortierung ermöglichen.</td>
</tr>
</tbody>
</table>

**Methoden bei Formularen und Kontrollfeldern**

Die Datentypen der Parameter werden durch Kürzel angedeutet:

- c Nummer der Spalte des gewünschten Feldes in der Datenmenge – ab 1 gezählt
- n numerischer Wert – je nach Situation als ganze Zahl oder als Dezimalzahl
- s Zeichenkette (String); die maximale Länge ergibt sich aus der Tabellendefinition
- b *boolean* (Wahrheitswert) – *true* (wahr) oder *false* (falsch)
- d Datumswert

**In einer Datenmenge navigieren**

Diese Methoden gelten sowohl für ein Formular als auch für die Ergebnismenge einer Abfrage.

Mit «Cursor» ist in den Beschreibungen der Datensatzzeiger gemeint.

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prüfungen für die Position des Cursors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isBeforeFirst</td>
<td>boolean</td>
<td>Der Cursor steht vor der ersten Zeile, wenn der Cursor nach dem Einlesen noch nicht gesetzt wurde.</td>
</tr>
<tr>
<td>isFirst</td>
<td>boolean</td>
<td>Gibt an, ob der Cursor auf der ersten Zeile steht.</td>
</tr>
<tr>
<td>isLast</td>
<td>boolean</td>
<td>Gibt an, ob der Cursor auf der letzten Zeile steht.</td>
</tr>
<tr>
<td>isAfterLast</td>
<td>boolean</td>
<td>Der Cursor steht hinter der letzten Zeile, wenn er von der letzten Zeile aus mit next weiter gesetzt wurde.</td>
</tr>
<tr>
<td>getRow</td>
<td>long</td>
<td>Nummer der aktuellen Zeile</td>
</tr>
</tbody>
</table>

**Setzen des Cursors**

Beim Datentyp *boolean* steht das Ergebnis «true» dafür, dass das Navigieren erfolgreich war.

<table>
<thead>
<tr>
<th>Name</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>beforeFirst</td>
<td>Wechselt vor die erste Zeile.</td>
</tr>
<tr>
<td>first</td>
<td>Wechselt zur ersten Zeile.</td>
</tr>
<tr>
<td>previous</td>
<td>Geht um eine Zeile zurück.</td>
</tr>
<tr>
<td>next</td>
<td>Geht um eine Zeile vorwärts.</td>
</tr>
</tbody>
</table>

*Makros: Makros in Base*
<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>last</td>
<td>boolean</td>
<td>Wechselt zur letzten Zeile.</td>
</tr>
<tr>
<td>afterLast</td>
<td>–</td>
<td>Wechselt hinter die letzte Zeile.</td>
</tr>
<tr>
<td>absolute(n)</td>
<td>boolean</td>
<td>Geht zu der Zeile mit der angegebenen Nummer.</td>
</tr>
<tr>
<td>relative(n)</td>
<td>boolean</td>
<td>Geht um eine bestimmte Anzahl von Zeilen weiter: bei positivem Wert von n vorwärts, andernfalls zurück.</td>
</tr>
</tbody>
</table>

**Maßnahmen zum Status der aktuellen Zeile**

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>refreshRow</td>
<td>–</td>
<td>Liest die ursprünglichen Werte der aktuellen Zeile neu ein.</td>
</tr>
<tr>
<td>rowInserted</td>
<td>boolean</td>
<td>Gibt an, ob es sich um eine neue Zeile handelt.</td>
</tr>
<tr>
<td>rowUpdated</td>
<td>boolean</td>
<td>Gibt an, ob die aktuelle Zeile geändert wurde.</td>
</tr>
<tr>
<td>rowDeleted</td>
<td>boolean</td>
<td>Gibt an, ob die aktuelle Zeile gelöscht wurde.</td>
</tr>
</tbody>
</table>

**Datenzeilen bearbeiten**

Die Methoden zum Lesen stehen bei jedem Formular und bei einer Ergebnismenge zur Verfügung. Die Methoden zum Ändern und Speichern gibt es nur bei einer Datenmenge, die geändert werden kann (in der Regel also nur bei Tabellen, nicht bei Abfragen).

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>insertRow</td>
<td>–</td>
<td>Speichert eine neue Zeile.</td>
</tr>
<tr>
<td>updateRow</td>
<td>–</td>
<td>Bestätigt Änderungen der aktuellen Zeile.</td>
</tr>
<tr>
<td>deleteRow</td>
<td>–</td>
<td>Löscht die aktuelle Zeile.</td>
</tr>
<tr>
<td>cancelRowUpdates</td>
<td>–</td>
<td>Macht Änderungen der aktuellen Zeile rückgängig.</td>
</tr>
<tr>
<td>moveToInsertRow</td>
<td>–</td>
<td>Wechselt den Cursor in die Zeile für einen neuen Datensatz.</td>
</tr>
<tr>
<td>moveToCurrentRow</td>
<td>–</td>
<td>Kehrt nach der Eingabe eines neuen Datensatzes zurück zur vorherigen Zeile.</td>
</tr>
</tbody>
</table>

**Werte lesen**

<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>getString(c)</td>
<td>string</td>
<td>Liefert den Inhalt der Spalte als Zeichenkette.</td>
</tr>
<tr>
<td>getBoolean(c)</td>
<td>boolean</td>
<td>Liefert den Inhalt der Spalte als Wahrheitswert.</td>
</tr>
<tr>
<td>getByte(c)</td>
<td>byte</td>
<td>Liefert den Inhalt der Spalte als einzelnes Byte.</td>
</tr>
<tr>
<td>getShort(c)</td>
<td>short</td>
<td>Liefert den Inhalt der Spalte als ganze Zahl.</td>
</tr>
<tr>
<td>getInt(c)</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>getLong(c)</td>
<td>long</td>
<td></td>
</tr>
<tr>
<td>getFloat(c)</td>
<td>float</td>
<td>Liefert den Inhalt der Spalte als Dezimalzahl von einfacher Genauigkeit.</td>
</tr>
<tr>
<td>getDouble(c)</td>
<td>double</td>
<td>Liefert den Inhalt der Spalte als Dezimalzahl von doppelter Genauigkeit. – Wegen der automatischen Konvertierung durch Basic ist dies auch für decimal- und currency-Werte geeignet.</td>
</tr>
<tr>
<td>getBytes(c)</td>
<td>array of bytes</td>
<td>Liefert den Inhalt der Spalte als Folge einzelner Bytes.</td>
</tr>
<tr>
<td>Name</td>
<td>Datentyp</td>
<td>Beschreibung</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getDate(c)</td>
<td>Date</td>
<td>Liefert den Inhalt der Spalte als Datumswert.</td>
</tr>
<tr>
<td>getTime(c)</td>
<td>Time</td>
<td>Liefert den Inhalt der Spalte als Zeitwert.</td>
</tr>
<tr>
<td>getTimestamp(c)</td>
<td>DateTime</td>
<td>Liefert den Inhalt der Spalte als Zeitstempel (Datum und Zeit).</td>
</tr>
</tbody>
</table>


| wasNull         | boolean  | Gibt an, ob der Wert der zuletzt gelesenen Spalte NULL war.                  |

**Werte speichern**

<table>
<thead>
<tr>
<th>Method</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateNull(c)</td>
<td>–</td>
<td>Setzt den Inhalt der Spalte c auf NULL.</td>
</tr>
<tr>
<td>updateBoolean(c,b)</td>
<td>–</td>
<td>Setzt den Inhalt der Spalte c auf den Wahrheitswert b.</td>
</tr>
<tr>
<td>updateByte(c,x)</td>
<td>–</td>
<td>Speichert in Spalte das angegebene Byte x.</td>
</tr>
<tr>
<td>updateShort(c,n)</td>
<td>–</td>
<td>Speichert in Spalte die angegebene ganze Zahl n.</td>
</tr>
<tr>
<td>updateInt(c,n)</td>
<td>–</td>
<td>Speichert in Spalte die angegebene Dezimalzahl n.</td>
</tr>
<tr>
<td>updateLong(c,n)</td>
<td>–</td>
<td>Speichert in Spalte die angegebene Zeichenkette s.</td>
</tr>
<tr>
<td>updateFloat(c,n)</td>
<td>–</td>
<td>Speichert in Spalte das angegebene Byte-Array x.</td>
</tr>
<tr>
<td>updateDate(c,d)</td>
<td>–</td>
<td>Speichert in Spalte das angegebene Datum d.</td>
</tr>
<tr>
<td>updateTime(c,d)</td>
<td>–</td>
<td>Speichert in Spalte den angegebenen Zeitwert d.</td>
</tr>
<tr>
<td>updateTimestamp(c,d)</td>
<td>–</td>
<td>Speichert in Spalte den angegebenen Zeitstempel d.</td>
</tr>
</tbody>
</table>

**Einzelne Werte bearbeiten**

Mit diesen Methoden wird über BoundField aus einem Kontrollfeld der Inhalt der betreffenden Spalte gelesen oder geändert. Diese Methoden entsprechen fast vollständig denen im vorigen Abschnitt; die Angabe der Spalte entfällt.
<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werte lesen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getString</td>
<td>string</td>
<td>Liefert den Inhalt der Spalte als Zeichenkette.</td>
</tr>
<tr>
<td>getBoolean</td>
<td>boolean</td>
<td>Liefert den Inhalt der Spalte als Wahrheitswert.</td>
</tr>
<tr>
<td>getByte</td>
<td>byte</td>
<td>Liefert den Inhalt der Spalte als einzelnes Byte.</td>
</tr>
<tr>
<td>getShort</td>
<td>short</td>
<td>Liefert den Inhalt der Spalte als ganze Zahl.</td>
</tr>
<tr>
<td>getInt</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>getLong</td>
<td>long</td>
<td></td>
</tr>
<tr>
<td>getFloat</td>
<td>float</td>
<td>Liefert den Inhalt der Spalte als Dezimalzahl von einfacher Genauigkeit.</td>
</tr>
<tr>
<td>getDouble</td>
<td>double</td>
<td>Liefert den Inhalt der Spalte als Dezimalzahl von doppelter Genauigkeit. –</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wegen der automatischen Konvertierung durch Basic ist dies auch für decimal-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>und currency- Werte geeignet.</td>
</tr>
<tr>
<td>getBytes</td>
<td>array of bytes</td>
<td>Liefert den Inhalt der Spalte als Folge einzelner Bytes.</td>
</tr>
<tr>
<td>getDate</td>
<td>Date</td>
<td>Liefert den Inhalt der Spalte als Datumswert.</td>
</tr>
<tr>
<td>getTime</td>
<td>Time</td>
<td>Liefert den Inhalt der Spalte als Zeitwert.</td>
</tr>
<tr>
<td>getTimestamp</td>
<td>DateTime</td>
<td>Liefert den Inhalt der Spalte als Zeitstempel (Datum und Zeit).</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>wasNull</td>
<td>boolean</td>
<td>Gibt an, ob der Wert der zuletzt gelesenen Spalte NULL war.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Werte speichern</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>updateNull</td>
<td>–</td>
<td>Setzt den Inhalt der Spalte auf NULL.</td>
</tr>
<tr>
<td>updateBoolean(b)</td>
<td>–</td>
<td>Setzt den Inhalt der Spalte auf den Wahrheitswert b.</td>
</tr>
<tr>
<td>updateByte(x)</td>
<td>–</td>
<td>Speichert in der Spalte das angegebene Byte x.</td>
</tr>
<tr>
<td>updateShort(n)</td>
<td>–</td>
<td>Speichert in der Spalte die angegebene ganze Zahl n.</td>
</tr>
<tr>
<td>updateInt(n)</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>updateLong(n)</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>updateFloat(n)</td>
<td>–</td>
<td>Speichert in der Spalte die angegebene Dezimalzahl n.</td>
</tr>
<tr>
<td>updateDouble(n)</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>updateString(s)</td>
<td>–</td>
<td>Speichert in der Spalte die angegebene Zeichenkette s.</td>
</tr>
<tr>
<td>updateBytes(x)</td>
<td>–</td>
<td>Speichert in der Spalte das angegebene Byte-Array x.</td>
</tr>
<tr>
<td>updateDate(d)</td>
<td>–</td>
<td>Speichert in der Spalte das angegebene Datum d.</td>
</tr>
<tr>
<td>updateTime(d)</td>
<td>–</td>
<td>Speichert in der Spalte den angegebenen Zeitwert d.</td>
</tr>
<tr>
<td>updateTimestamp(d)</td>
<td>–</td>
<td>Speichert in der Spalte den angegebenen Zeitstempel d.</td>
</tr>
</tbody>
</table>

**Makros: Makros in Base**
Parameter für vorbereitete SQL-Befehle


<table>
<thead>
<tr>
<th>Name</th>
<th>Datentyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>setNull(i, n)</td>
<td>–</td>
<td>Setzt den Inhalt der Spalte auf NULL n bezeichnet den SQL-Datentyp gemäß API-Referenz.</td>
</tr>
<tr>
<td>setBoolean(i, b)</td>
<td>–</td>
<td>Fügt den angegebenen Wahrheitswert b in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>setByte(i, x)</td>
<td>–</td>
<td>Fügt das angegebene Byte x in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>setShort(i, n)</td>
<td>–</td>
<td>Fügt die angegebene ganze Zahl n in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>setInt(i, n)</td>
<td>–</td>
<td>Fügt die angegebene Dezimalzahl n in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>setLong(i, n)</td>
<td>–</td>
<td>Fügt die angegebene Zeichenkette s in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>setFloat(i, n)</td>
<td>–</td>
<td>Fügt das angegebene Datum d in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>setDouble(i, n)</td>
<td>–</td>
<td>Fügt den angegebenen Zeitwert d in den SQL-Befehl ein.</td>
</tr>
<tr>
<td>clearParameters</td>
<td>–</td>
<td>Entfernt die bisherigen Werte aller Parameter eines SQL-Befehls.</td>
</tr>
</tbody>
</table>

Bedienbarkeit verbessern

Als erste Kategorie werden verschiedene Möglichkeiten vorgestellt, die zur Verbesserung der Bedienbarkeit von Base-Formularen dienen. Sofern nicht anders erwähnt sind diese Makros Bestandteil der Beispieldatenbank «Medien_mit_Makros.odb».

Automatisches Aktualisieren von Formularen

Oft wird in einem Formular etwas geändert und in einem zweiten, auf der gleichen Seite liegenden Formular, soll die Änderung anschließend erscheinen. Hier hilft bereits ein kleiner Codeschnipsel, um das betreffende Anzeigeformular zu aktualisieren.

SUB Aktualisieren

Zuerst wird einmal das Makro benannt. Die Standardbezeichnung für ein Makro ist SUB. Dies kann groß oder klein geschrieben sein. Mit SUB wird eine Prozedur ablaufen gelassen, die nach außen keinen Wert weitergibt. Weiter unten wird im Gegensatz dazu einmal eine Funktion beschrieben, die im Unterschied dazu Rückgabewerte erzeugt.

Das Makro hat jetzt den Namen «Aktualisieren». Um sicher zu gehen, dass keine Variablen von außen eingeschleust werden gehen viele Programmierer so weit, dass sie Basic über Option Explicit gleich zu Beginn mitteilen: Erzeuge nicht automatisch irgendwelche Variablen sondern nutze nur die, die ich auch vorher definiert habe.
Deshalb werden jetzt standardgemäß erst einmal die Variablen deklariert. Bei allen hier deklarierten Variablen handelt es sich um Objekte (nicht z.B. Zahlen oder Texte), so dass der Zusatz **OBJECT** hinter der Deklaration steht. Um später noch zu erkennen, welchen Typ eine Variable hat, ist vor die Variablenbezeichnung ein «o» gesetzt worden. Prinziell ist aber die Variablenbezeichnung nahezu völlig frei wählbar.

```
DIM oDoc AS OBJECT
DIM oDrawpage AS OBJECT
DIM oForm AS OBJECT
```

Das Formular liegt in dem momentan aktiven Dokument. Der Behälter, in dem alle Formulare aufbewahrt werden, wird als **Drawpage** bezeichnet. Im Formularnavigator ist dies sozusagen der oberste Begriff, an den dann sämtliche Formulare angehängt werden.

Das Formular, auf das zugegriffen werden soll, ist hier mit den Namen "Anzeige" versehen. Dies ist der Name, der auch im Formularnavigator sichtbar ist. So hat z.B. das erste Formular standardmäßig erst einmal den Namen "MainForm".

```vbnet
oDoc = thisComponent
oDrawpage = oDoc.Drawpage
oForm = oDrawpage.forms.getByName("Anzeige")
```

Nachdem das Formular jetzt ansprechbar gemacht wurde und der Punkt, an dem es angesprochen wurde, in der Variablen **oForm** gespeichert wurde, wird es jetzt mit dem Befehl **reload()** neu geladen.

```vbnet
oForm.reload()
```
END SUB

Die Prozedur hat mit **SUB** begonnen. Sie wird mit **END SUB** beendet.

Dieses Makro kann jetzt z.B. ausgelöst werden, wenn die Abspeicherung in einem anderen Formular erfolgt. Wird z.B. in einem Kassenformular an einer Stelle die Anzahl der Gegenstände und (über Barcodescanner) die Nummer eingegeben, so kann in einem anderen Formular im gleichen geöffneten Fenster hierdurch der Kassenstand, die Bezeichnung der Ware usw. nach dem Abspeichern sichtbar gemacht werden.

### Filtern von Datensätzen

Der Filter selbst funktioniert ja schon ganz ordentlich in einer weiter oben beschriebenen Variante im Kapitel «Datenfilterung». Die untenstehende Variante ersetzt den Abspeicherungsbutton und liest die Listenfelder neu ein, so dass ein gewählter Filter aus einem Listenfeld die Auswahl in dem anderen Listenfeld einschränken kann.16

```
SUB Filter
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm1 AS OBJECT
    DIM oForm2 AS OBJECT
    DIM oFeldList1 AS OBJECT
    DIM oFeldList2 AS OBJECT
    oDoc = thisComponent
    oDrawpage = oDoc.drawpage
```

Zuerst werden die Variablen definiert und auf das Gesamtformular zugegriffen. Das Gesamtformular besteht aus den Formularen "Filter" und "Anzeige". Die Listenfelder befinden sich in dem Formular "Filter" und sind mit dem Namen "Liste_1" und "Liste_2" versehen.

```vbnet
oForm1 = oDrawpage.forms.getByName("Filter")
oForm2 = oDrawpage.forms.getByName("Anzeige")
oFeldList1 = oForm1.getByNamed("Liste_1")
oFeldList2 = oForm1.getByNamed("Liste_2")
```


*Makros: Bedienbarkeit verbessern* 411

```
oFeldList1.commit()
oFeldList2.commit()
OForm1.updateRow()
```


Anschließend wird das Formular2, das den gefilterten Inhalt anzeigen soll, neu geladen.

```
oFeldList1.refresh()
oFeldList2.refresh()
OForm2.reload()
```

Soll mit diesem Verfahren ein Listenfeld von der Anzeige her beeinflusst werden, so kann das Listenfeld mit Hilfe verschiedener Abfragen bestückt werden.

Die einfachste Variante ist, dass sich die Listenfelder mit ihrem Inhalt aus dem Filterergebnis versorgen. Dann bestimmt der eine Filter, aus welchen Datenbestand anschließend weiter gefiltert werden kann.

```
SELECT
 "Feld_1" || ' - ' || "Anzahl" AS "Anzeige",
"Feld_1"
FROM
 ( SELECT COUNT( "ID" ) AS "Anzahl", "Feld_1" FROM
  "Tabelle_Filterergebnis" GROUP BY "Feld_1"
 )
ORDER BY "Feld_1"
```

Es wird der Feldinhalt und die Trefferzahl angezeigt. Um die Trefferzahl zu errechnen, wird eine Unterabfrage gestellt. Dies ist notwendig, da sonst nur die Trefferzahl ohne weitere Information aus dem Feld in der Listbox angezeigt würde.

Das Makro erzeugt durch dieses Vorgehen ganz schnell Listboxen, die nur noch mit einem Wert gefüllt sind. Steht eine Listbox nicht auf NULL, so wird sie schließlich bei der Filterung bereits berücksichtigt. Nach Betätigung der 2. Listbox stehen also bei beiden Listboxen nur noch die leeren Felder und jeweils 1 angezeigter Wert zur Verfügung. Dies mag für eine eingrenzende Suche erst einmal praktisch erscheinen. Was aber, wenn z.B. in einer Bibliothek die Zuordnung zur Systematik klar war, aber nicht eindeutig, ob es sich um ein Buch, eine CD oder eine DVD handelt? Wurde einmal die Systematik angewählt und dann die 2. Listbox auf CD gestellt so muss, um auch die Bücher zu sehen, die 2. Listbox erst einmal wieder auf NULL gestellt werden, um dann auch die Bücher anwählen zu können. Praktischer wäre, wenn die 2. Listbox direkt die verschiedenen Medienarten anzeigen würde, die zu der Systematik zur Verfügung stehen – natürlich mit den entsprechenden Trefferquoten.

Um dies zu erreichen, wurde die folgende Abfrage konstruiert, die jetzt nicht mehr direkt aus dem Filterergebnis gespeist wird. Die Zahlen für die Treffer müssen anders ermittelt werden.

```
SELECT
 IFNULL( "Feld_1" || ' - ' || "Anzahl", 'leer - ' || "Anzahl" ) AS
 "Anzeige",
```

Makros: Bedienbarkeit verbessern
'Feld_1'
FROM
  ( SELECT COUNT( "ID" ) AS "Anzahl", "Feld_1" FROM "Tabelle"
    WHERE "ID" IN
      ( SELECT "Tabelle"."ID" FROM "Filter", "Tabelle"
        WHERE "Tabelle"."Feld_2" = IFNULL( "Filter"."Filter_2",
          "Tabelle"."Feld_2" ) )
    GROUP BY "Feld_1"
  )
ORDER BY "Feld_1"

Diese doch sehr verschachtelte Abfrage kann auch unterteilt werden. In der Praxis bietet es sich häufig an, die Unterabfrage in einer Tabellenansicht ('VIEW') zu erstellen. Das Listenfeld bekommt seinen Inhalt über eine Abfrage, die sich auf diesen 'VIEW' bezieht.

Die Abfrage im Einzelnen:


Als Tabelle, aus der diese Informationen gezogen werden, liegt eine Abfrage vor. In dieser Abfrage werden die Primärschlüsselfelder gezählt (SELECT COUNT( "ID" ) AS "Anzahl"). Dies geschieht gruppiert nach der Bezeichnung, die in dem Feld steht (GROUP BY "Feld_1"). Als zweite Spalte stellt diese Abfrage das Feld selbst als Begriff zur Verfügung. Diese Abfrage wiederum basiert auf einer weiteren Unterabfrage:

```
  SELECT "Tabelle"."ID"
  FROM "Filter", "Tabelle"
  WHERE "Tabelle"."Feld_2" = IFNULL( "Filter"."Filter_2",
    "Tabelle"."Feld_2" )
```

Diese Unterabfrage bezieht sich jetzt auf das andere zu filternde Feld. Prinzipiell muss das andere zu filternde Feld auch zu den Primärschlüsselnummern passen. Sollten noch mehrere weitere Filter existieren so ist diese Unterabfrage zu erweitern:

```
  SELECT "Tabelle"."ID"
  FROM "Filter", "Tabelle"
  WHERE "Tabelle"."Feld_2" = IFNULL( "Filter"."Filter_2",
    "Tabelle"."Feld_2" )
  AND
    "Tabelle"."Feld_3" = IFNULL( "Filter"."Filter_3",
    "Tabelle"."Feld_3" )
```

Alle weiteren zu filternden Felder beeinflussen, was letztlich in dem Listenfeld des ersten Feldes, "Feld_1", angezeigt wird.

Zum Schluss wird die gesamte Abfrage nur noch nach dem zugrundeliegenden Feld sortiert.

Wie letztlich die Abfrage aussieht, die dem anzuzeigenden Formular zugrunde liegt, ist im Kapitel «Datenfilterung» nachzulesen.

Mit dem folgenden Makro kann über das Listenfeld gesteuert werden, welches Listenfeld abgespeichert werden muss und welches neu eingelesen werden muss.


```
SUB Filter_Zusatzinfo(oEvent AS OBJECT)
```
Ein Array (Ansammlung von Daten, die hier über Zahlenverbindungen abgerufen werden können) wird gegründet und mit den Feldnamen der Listenfelder gefüllt. Der erste Name ist der Name von dem Listenfeld, das mit der Aktion (Event) verbunden ist.

Das Array wird von seiner Untergrenze (LBound()) bis zu seiner Obergrenze (UBound()) in einer Schleife durchlaufen. Alle Werte, die in den Zusatzinformationen durch Komma getrennt erscheinen, werden jetzt nacheinander weitergegeben.

```vba
FOR i = LBound(aList()) TO UBound(aList())
    IF i = 0 THEN
        Das auslösende Listenfeld muss abgespeichert werden. Es hat die Variable aList(0). Zuerst wird die Information des Listenfeldes auf die zugrundeliegende Tabelle übertragen, dann wird der Datensatz gespeichert.
        oForm1.getByName(aList(i)).commit()
        oForm1.updateRow()
    ELSE
        Die anderen Listenfelder müssen neu eingelesen werden, da sie ja in Abhängigkeit vom ersten Listenfeld jetzt andere Werte abbilden.
        oForm1.getByName(aList(i)).refresh()
    END IF
NEXT
oForm2.reload()
```

Die Abfragen für dieses besser nutzbare Makro sind natürlich die gleichen wie in diesem Abschnitt zuvor bereits vorgestellt.

**Daten über den Formularfilter filtern**

Alternativ zu dieser Vorgehensweise ist es auch möglich, die Filterfunktion des Formulars direkt zu bearbeiten.

```vba
SUB FilterSetzen
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM stFilter As String
    oForm = thisComponent.Drawpage.Forms.getByName("MainForm")
    oFeld = oForm.getByName("Filter")
    stFilter = oFeld.Text
    oForm.filter = " UCASE(""Name"") LIKE '%'|| UCASE(stFilter) + ||'%'" oFeld = oForm.getByName("Filter")
    oForm.ApplyFilter = TRUE oForm.reload()
END Sub
```


```vba
SUB FilterEntfernen
    DIM oForm AS OBJECT
    oForm = thisComponent.Drawpage.Forms.getByName("MainForm")
```

---

Makros: Bedienbarkeit verbessern 414
Die Beendigung des Filters kann natürlich auch über die Navigationsleiste erfolgen. In diesem Fall wird einfach ein weiteres Makro genutzt.

Über diese Filterfunktion kann ein Formular auch direkt mit einem Filter z.B. für nur einen Datensatz gestartet werden. Aus dem startenden Formular wird ein Wert (z.B. ein Primärschlüssel für den aktuellen Datensatz) ausgelesen und an das Zielformular als Filterwert weitergegeben.

**Daten aus Textfeldern auf SQL-Tauglichkeit vorbereiten**


```vba
FUNCTION String_to_SQL(st AS STRING)
    IF InStr(st, '''') THEN
        st = Join(Split(st, '''), ''''')
    END IF
    String_to_SQL = st
END FUNCTION
```

Es handelt sich hier um eine Funktion. Eine Funktion nimmt einen Wert auf und liefert anschließend auch einen Gegenwert zurück.

Der übergebende Text wird zuerst einmal daraufhin untersucht, ob er ein Hochkomma enthält. Ist dies der Fall, so wird der Text an der Stelle aufgetrennt - der Trenner dafür ist das Hochkomma – und anschließend stattdessen mit zwei Hochkommata wieder zusammengefügt. Der SQL-Code wird so maskiert.

Die Funktion übergibt ihr Ergebnis durch den folgenden Aufruf:

```vba
stTextneu = String_to_SQL(stTextalt)
```

Es wird also einfach nur die Variable stTextalt überarbeitet und der entsprechende Wert wieder in der Variablen stTextneu gespeichert. Dabei müssen die Variablen gar nicht unterschiedlichen heißen. Der Aufruf geht praktischer direkt mit:

```vba
stText = String_to_SQL(stText)
```

Diese Funktion wird in den nachfolgenden Makros immer wieder benötigt, damit Hochkommata auch über SQL abgespeichert werden können.

**Werte in einem Formular vorausberechnen**

Werte, die über die Datenbankfunktionen berechnet werden können, werden in der Datenbank nicht extra gespeichert. Die Berechnung erfolgt allerdings nicht während der Eingabe im Formular, sondern erst nachdem der Datensatz abgespeichert ist. Solange das Formular aus einem Tabellekontrollfeld besteht mag das nicht so viel ausmachen. Schließlich kann direkt nach der Eingabe ein berechneter Wert ausgelesen werden. Bei Formularen mit einzelnen Feldern bleibt der vorherige Datensatz aber nicht unbedingt sichtbar. Hier bietet es sich an, die Werte, die sonst in der Datenbank berechnet werden, direkt in entsprechenden Feldern anzuzeigen.17


```vba
SUB Berechnung_ohne_MwSt(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
```

17 Siehe hierzu die Beispieldatenbank «Beispiel_Direktberechnung_im Formular.odb»

Ist bereits im Feld «Anzahl» ein Wert enthalten, so wird eine Folgerechnung auch für die damit verbundenen Felder durchgeführt.

```
SUB Berechnung_gesamt(oEvent AS OBJECT)
  oFeld = oEvent.Source.Model
  Berechnung_gesamt2(oFeld)
END SUB
```

Diese kurze Prozedur dient nur dazu, die Auslösung der Folgeprozedur vom Verlassen des Formularfeldes «Anzahl» weiter zu geben. Die Angabe könnte genauso gut mit Hilfe der Bestimmung des Feldes über die Drawpage in der Folgeprozedur integriert werden.

```
SUB Berechnung_gesamt2(oFeld AS OBJECT)
  oForm = oFeld.Parent
  DIM oFeld2 AS OBJECT
  oFeld2 = oForm.getByName("Preis")
  DIM oFeld3 AS OBJECT
  oFeld3 = oForm.getByName("Preis_gesamt_mit_MWSt")
  DIM oFeld4 AS OBJECT
  oFeld4 = oForm.getByName("MWSt_gesamt")
  oFeld2.BoundField.UpdateDouble(oFeld2.getCurrentValue * oFeld4.getCurrentValue - oFeld3.getCurrentValue * 1.19)
END SUB
```


**Die aktuelle Office-Version ermitteln**

Mit der Version 4.1 sind Änderungen bei Listenfeldern und Datumswerten vorgenommen worden, die es erforderlich machen, vor der Ausführung eines Makros für diesen Bereich zu erkunden, welche Office-Version denn nun verwendet wird. Dazu dient der folgende Code:

```
FUNCTION OfficeVersion()
  DIM aSettings, aConfigProvider
  DIM aParams2[0] AS NEW com.sun.star.beans.PropertyValue
  DIM sProvider$, sAccess$
  sProvider = "com.sun.star.configuration.ConfigurationProvider"
END FUNCTION
```
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)

Diese Funktion gibt eine Array wieder, das als ersten Wert z.B. "LibreOffice" und als zweiten Wert die detaillierte Version, z.B. "4.1.5.2" ausgibt.

Wert von Listenfeldern ermitteln


FUNCTION ID_Ermittlung(oFeld 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
    stInhalt = oFeld.currentValue
  ELSE
    stInhalt = oFeld.ValueItemList(oFeld.SelectedItems(0))
  END IF
  IF IsEmpty(stInhalt) THEN
    stInhalt = -1
  ELSE
    stInhalt = Cint(stInhalt)
  END IF
END FUNCTION

Vor LO 4.1 wird der Wert, der weiter gegeben wird, aus der Werteliste des Listenfeldes ausgelesen. Der sichtbar ausgewählte Datensatz ist SelectedItems(0). '0', weil auch mehrere Werte in einem Listenfeld ausgewählt werden könnten.

FUNCTION ID_Ermittlung(oFeld 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
    stInhalt = oFeld.currentValue
  ELSE
    stInhalt = oFeld.ValueItemList(oFeld.SelectedItems(0))
  END IF
  IF IsEmpty(stInhalt) THEN
    stInhalt = -1
  ELSE
    stInhalt = Cint(stInhalt)
  END IF
END FUNCTION

Die Funktion gibt den Wert als Integer wieder. Meist werden für Primärschlüssel ja automatisch hoch zählende Integer-Werte verwendet. Für eine Verwendung von Fremdschlüsseln, die diesem Kriterium nicht entsprechen, muss die Ausgabe der Variablen entsprechend angepasst werden.

Der angezeigte Wert eines Listenfeldes lässt sich weiterhin über die Ansicht des Feldes ermitteln:

SUB Listenfeldanzeige
  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("Listenfeld")
  oController = oDoc.getCurrentController()
  oView = oController.getControl(oListbox)
  print "Angezeigter Inhalt: " & oView.SelectedItem
END SUB

Es wird über den Controller auf die Ansicht es Formulars zugegrißen. Damit wird ermittelt, was auf der sichtbaren Oberfläche tatsächlich erscheint. Der ausgewählte Wert ist der SelectedItem.
Listenfelder durch Eingabe von Anfangsbuchstaben einschränken

Manchmal kann es vorkommen, dass der Inhalt für Listenfelder unübersichtlich groß wird. Damit eine Suche schneller zum Erfolg führt wäre es sinnvoll, hier den Inhalt des Listenfeldes nach Eingabe eines oder mehrerer Buchstaben einzuschränken. Das Listenfeld selbst wird erst einmal mit einem SQL-Befehl versehen, der nur als Platzhalter dient. Hier könnte z.B. stehen:

SELECT "Name", "ID" FROM "Tabelle" ORDER BY "Name" LIMIT 5

So wird beim Öffnen des Formulars vermieden, dass Base erst einmal die umfangreiche Liste einlesen muss.

Das folgende Makro ist dafür an Eigenschaften: Listenfeld → Ereignisse → Taste losgelassen gekoppelt.

GLOBAL stListStart AS STRING
GLOBAL lZeit AS LONG


In der globalen Variablen stListStart werden die Buchstaben in der eingegebenen Reihenfolge gespeichert.


SUB ListFilter(oEvent AS OBJECT)
    oFeld = oEvent.Source.Model
    IF oEvent.KeyCode < 538 THEN


Die Abfrage des KeyCode ist hier wichtig, da auch der Schritt mit der Tabulatortaste auf das Auswahlfeld natürlich das Makro auslöst. Der KeyCode für die Tabulatortaste liegt allerdings bei 1282, so dass der weitere Code der Prozedur hier nicht ausgeführt wird.

GLOBAL stSql(0) AS STRING

Der SQL-Code für das Listenfeld wird in einem Array gespeichert. Das Array hat im Falle des SQL-Codes aber nur ein Datenfeld. Deshalb ist das Array direkt auf stSql(0) begrenzt.


Makros: Bedienbarkeit verbessern 418
stFeld = "" & Rtrim(ar1(1))


stQuery = Left(oFeld.ListSource(0), inStr(1,oFeld.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

Enthält die ermittelte Abfrage den Begriff «LOWER», so wird davon ausgegangen, dass die Abfrage bereits über die Prozedur ListFilter erstellt wurde. Deswegen wird die neu zu konstruierende Abfrage nur bis zur Position dieses Begriffes übernommen.

Ist dies nicht der Fall und es existiert in der Abfrage bereits der Begriff «WHERE» in beliebiger Schreibweise, so müssen weitere Bedingungen an die Abfrage nur bis zur Position dieses Begriffes übernommen.

Sind beide Bedingungen nicht erfüllt, so wird ein WHERE an den bestehenden Code angehängt.

IF lZeit > 0 AND Timer() - lZeit < 5 THEN
stListStart = stListStart & oEvent.KeyChar
ELSE
stListStart = oEvent.KeyChar
END IF


stText = LCase( stListStart & "%")
stSql(0) = stQuery + "LOWER("+stFeld+") LIKE "+stText+" ORDER BY "+stFeld+"
end stSql(oFeld.ListSource = stSql
end refresh
END IF
END SUB


Datumswert aus einem Formularwert in eine Datumsvariable umwandeln

FUNCTION Datumswert(oFeld 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

Makros: Bedienbarkeit verbessern
Hier werden alle Versionen ab 4.1 abgefangen. Dazu wird die Version in ihre Bestandteile aufge- 
splittet. Die Hauptversion und die erste Unterversion werden abgefragt. Das funktioniert vorerst bis 
zur LO-Version 9 einwandfrei.

\[ \text{DIM stMonat AS STRING} \]
\[ \text{DIM stTag AS STRING} \]
\[ \text{stMonat} = \text{Right(Str(0) & Str(oFeld.CurrentValue.Month),2)} \]
\[ \text{stTag} = \text{Right(Str(0) & Str(oFeld.CurrentValue.Day),2)} \]
\[ \text{Datumswert} = \text{CDateFromIso(oFeld.CurrentValue.Year} \& \text{stMonat} \& \text{stTag}) \]
\[ \text{ELSE} \]
\[ \text{Datumswert} = \text{CDateFromIso(oFeld.CurrentValue)} \]
\[ \text{ENDIF} \]
\[ \text{END FUNCTION} \]

Das Datum wird seit LO 4.1.2 als Array im Formularfeld gespeichert. Mit dem aktuellen Wert kann 
also nicht auf das Datum selbst zugegriffen werden. Entsprechend ist es neu aus den Werten für 
Tag, Monat und Jahr zusammen zu setzen, damit anschließend in Makros damit weiter gearbeitet 
werden kann.

**Suchen von Datensätzen**

Ohne Makro funktioniert das Suchen von Datensätzen auch. Hier ist aber die entsprechende 
Abfrage äußerst unübersichtlich zu erstellen. Da könnte eine Schleife mittels Makro Abhilfe schaf-
fen.

Die folgende Variante liest die Felder einer Tabelle aus, gründet dann intern eine Abfrage und 
schreibt daraus schließlich eine Liste der Primärschlüsselnummern der durchsuchten Tabelle auf, 
auf der der Suchbegriff zutrifft. Für die folgende Beschreibung existiert eine Tabelle "Suchtmp", die 
aus einem per Autowert erstellten Primärschlüsselfeld "ID" und einem Feld "Nr." besteht, in das die 
aus der zu durchsuchenden Tabelle gefundenen Primärschlüssel eingetragen werden. Der Tabell-
lenname wird dabei der Prozedur am Anfang als Variable mitgegeben.

Um ein entsprechendes Ergebnis zu bekommen, muss die Tabelle natürlich nicht die Fremdschlüs-
sel, sondern entsprechende Feldinhalte in Textform enthalten. Dafür ist gegebenenfalls ein "VIEW" 
zur erstellen, auf den das Makro auch zugreifen kann.18

18 Siehe zu diesem Abschnitt auch die Datenbank «Beispiel_Suchen_und_Filtern.odb», die diesem Hand-

**Makros: Bedienbarkeit verbessern**

420
Zuerst wird einmal geklärt, ob überhaupt ein Suchbegriff eingegeben wurde. Ist das Feld leer, so wird davon ausgegangen, dass keine Suche vorgenommen wird. Alle Datensätze sollen angezeigt werden; eine weitere Abfrage erübrigt sich.

Ist ein Suchbegriff eingegeben worden, so werden die Spaltennamen der zu durchsuchenden Tabelle ausgelesen, um auf die Felder mit einer Abfrage zugreifen zu können.

```java
IF stInhalt <> ""
    stInhalt = String_to_SQL(stInhalt)
stSql = "SELECT "COLUMN_NAME" FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS"
    WHERE "TABLE_NAME" = '" + stTabelle + "' ORDER BY "ORDINAL_POSITION"
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
END IF
```


Aus `stSql = "SELECT "Name" FROM "Tabelle";"` wird, wenn es mit dem Befehl `msgbox stSql` auf dem Bildschirm angezeigt wird, `SELECT "Name" FROM "Tabelle"`

Der Zähler des Arrays, in das die Feldnamen geschrieben werden, wird zuerst auf 0 gesetzt. Dann wird begonnen die Abfrage auszulesen. Da die Größe des Arrays unbekannt ist, muss immer wieder nachjustiert werden. Deshalb beginnt die Schleife damit, über `Redim Preserve` `arInhalt(inI)` die Größe des Arrays festzulegen und den vorherigen Inhalt dabei zu sichern. Anschließend werden die Felder ausgelesen und der Zähler des Arrays um 1 heraufgesetzt. Damit kann dann das Array neu dimensioniert werden und wieder ein weiterer Wert abgespeichert werden.

```java
InI = 0
WHILE oAbfrageergebnis.next
    Redim Preserve arInhalt(inI)
    arInhalt(inI) = oAbfrageergebnis.getString(1)
    inI = inI + 1
WEND
stSql = "DROP TABLE "Suchtmp" IF EXISTS"
oSQL_Anweisung.executeUpdate(stSql)
```

Jetzt wird die Abfrage in einer Schleife zusammengestellt, die anschließend an die zu Beginn angegebene Tabelle gestellt wird. Dabei werden alle Schreibweisen untersucht, da auch der Inhalt des Feldes in der Abfrage auf Kleinbuchstaben umgewandelt wird.

Die Abfrage wird direkt so gestellt, dass die Ergebniswerte in der Tabelle "Suchtmp" landen. Dabei wird davon ausgegangen, dass der Primärschlüssel an der ersten Position der Tabelle steht (`arInhalt(0)`).

```java
stSql = "SELECT """+arInhalt(0)+""INTO ""Suchtmp"" FROM ""+stTabelle+"
    WHERE "
    FOR inK = 0 TO (inI - 1)
        stSql = stSql++++LCase(""+arInhalt(inK)+"" LIKE '%"+stInhalt+"%'"
    IF inK < (inI - 1) THEN
        stSql = stSql+++ OR 
    END IF
NEXT
    oSQL_Anweisung.executeUpdate(stSql)
ELSE
    stSql = "DELETE FROM ""Suchtmp"
    oSQL_Anweisung.executeUpdate(stSql)
END IF
```
Das Anzeigeformular muss neu geladen werden. Es hat als Datenquelle eine Abfrage, in diesem Beispiel "Suchabfrage"

    oForm2 = oDrawpage.forms.getByName("Anzeige")
    oForm2.reload()

End Sub

Damit wurde eine Tabelle erstellt, die nun in einer Abfrage ausgewertet werden soll. Die Abfrage ist dabei möglichst so zu fassen, dass sie anschließend noch editiert werden kann. Im Folgenden also ein Abfragecode:

    SELECT * FROM "Suchtabelle"
    WHERE "Nr." IN ( SELECT "Nr." FROM "Suchtmp" )
    OR "Nr." =
        CASE WHEN ( SELECT COUNT( "Nr." ) FROM "Suchtmp" ) > 0
        THEN '0' ELSE "Nr." END


Der Primärschlüssel ist in dieser Beispieltabelle unter dem Titel "Nr." abgespeichert. Durch das Makro wird genau dieses Feld ausgelesen. Es wird jetzt also zuerst nachgesehen, ob der Inhalt des Feldes "Nr." in der Tabelle "Suchtabelle" vorkommt. Bei der Verknüpfung mit 'IN' werden ohne weiteres auch mehrere Werte erwartet. Die Unterabfrage darf also auch mehrere Datensätze liefern.

Bei größeren Datenmengen wird der Abgleich von Werten über die Verknüpfung IN aber zusehends langsamer. Es bietet sich also nicht an, für eine leere Eingabe in das Suchfeld einfach alle Primärschlüsselfelder der "Suchtabelle" in die Tabelle "Suchtmp" zu übertragen und dann auf die gleiche Art die Daten anzusehen. Stattdessen erfolgt bei einer leeren Eingabe eine Leerung der Tabelle "Suchtmp", so dass gar keine Datensätze mehr vorhanden sind. Hierauf zielt der zweite Bedingungsteil:

    OR "Nr." = CASE WHEN ( SELECT COUNT( "Nr." ) FROM "Suchtmp" ) > 0 THEN
        '-1' ELSE "Nr." END

Wenn in der Tabelle "Suchtmp" ein Datensatz gefunden wird, so ist das Ergebnis der ersten Abfrage größer als 0. Für diesen Fall gilt: "Nr." = '-1' (hier steht am Besten ein Zahlenwert, der als Primärschlüssel nicht vorkommt, also z.B. '-1'). Ergibt die Abfrage genau 0 (Dies ist der Fall wenn keine Datensätze da sind), dann gilt "Nr." = "Nr.". Es wird also jeder Datensatz dargestellt, der eine "Nr." hat. Da "Nr." der Primärschlüssel ist, gilt dies also für alle Datensätze.

Suchen in Formularen und Ergebnisse farbig hervorheben

Bei größeren Inhalten eines Textfeldes ist oft unklar, an welcher Stelle denn nun die Suche den Treffer zu verzeichnen hat. Da wäre es doch gut, wenn das Formular den entsprechenden Treffer auch markieren könnte. So sollte das dann im Formular aussehen:
Um so ein Formular zum Laufen zu bekommen bedarf es ein paar zusätzlicher Eingriffe in die Trickkiste.\textsuperscript{19}

Die Funktionsweise so eines Suchfeldes wurde bereits bei den Abfragetechniken erklärt: Es wird eine Filtertabelle erstellt. Über ein Formular wird nur der aktuelle Wert des einzigen Datensatzes in dieser Tabelle neu geschrieben. Das Hauptformular wird über eine Abfrage mit dem entsprechenden Inhalt versorgt. Die Abfrage sieht im obigen Fall so aus:

\begin{verbatim}
SELECT "ID", "Memo"
FROM "Tabelle"
WHERE LOWER ( "Memo" ) LIKE '%%%%' || LOWER ( ( SELECT "Suchtext" FROM "Filter" WHERE "ID" = TRUE ) ) || '%%%%'
\end{verbatim}

Wird ein Suchtext eingetragen, so werden nur die Datensätze der Tabelle "Tabelle" angezeigt, bei denen der Text im Feld "Memo" vorkommt. Dabei ist Groß- und Kleinschreibung egal.

Wird kein Suchtext eingetragen, so werden alle Datensätze der Tabelle "Tabelle" angezeigt. Da der Primärschlüssel dieser Tabelle auch in der Abfrage enthalten ist, ist die Abfrage außerdem editierbar.

\textsuperscript{19} Siehe zu diesem Abschnitt auch die Datenbank «Beispiel_Autotext_Suchmarkierung_Rechtschreibung.odb», die dem Handbuch beiliegt.

\textit{LibreOffice} besitzt ein umfangreiches Hilfsystem. Um zu dem Hilfeystem zu gelangen, drücken Sie F1 oder wählen Sie \textit{LibreOffice} Hilfe aus dem Hilfemenü. Zusätzlich können Sie wählen, ob Sie Tips, Erweiterte Tips und den \textit{Office}-Assistenten einschalten (Extras \textemdash Optionen \textemdash \textit{LibreOffice} \textemdash Allgemein).

Wenn die Tips eingeschaltet sind, plazieren Sie den Mauszeiger über eines der Symbole um eine kleine Box (\textit{Tooltip}) angezeigt zu bekommen. Dann befindet sich eine kurze Erklärung der Funktion des Symbols. Um noch mehr Erklärungen zu erhalten, wählen Sie \textit{Hilfe} \textemdash Direkhilfe und halten den Mauszeiger über das Symbol.


Die Prozedur ist an das folgende Ereignis gebunden: Formular → Ereignisse → Nach dem Datensatzwechsel

```vbnet
Sub InhaltUebertragen(oEvent AS OBJECT)
    DIM inMemo AS INTEGER
    DIM oFeld AS OBJECT
    DIM stSuchtext AS STRING
    DIM oCursor AS OBJECT
    DIM inSuch AS INTEGER
    DIM inSuchAlt AS INTEGER
    DIM inLen AS INTEGER
    oForm = oEvent.Source
    inMemo = oForm.findColumn("Memo")
    oFeld = oForm.getByName("MemoFormat")
    oFeld.Text = oForm.getString(inMemo)
Zuerst werden die Variablen definiert. Anschließend wird über das Formular das Tabellenfeld "Memo" gesucht und aus diesem Feld über getString() der entsprechende Text des Feldes "Memo" der Tabelle "Tabelle" ausgelesen. Der entsprechende Feldinhalt wird in das Feld übertragen, das sich formatieren lässt, aber keine Verbindung zur Datenbank hat: «MemoFormat».
```

Makros: Bedienbarkeit verbessern

\[
\text{Wait 5}
\]
\[
stSuchtext = oForm.Parent.getByName("FormularFilter").getByName("Suche").Text
\]


\[
\text{oCursor = oFeld.createTextCursor()}
\text{oCursor.gotoStart(true)}
\text{oCursor.gotoEnd(true)}
\text{oCursor.CharHeight = 10}
\text{oCursor.CharFontName = "Arial, Helvetica, Tahoma"}
\text{oCursor.CharColor = RGB(0,0,0)}
\text{oCursor.CharWeight = 100.000000 'com::sun::star::awt::FontWeight}
\text{oCursor.gotoStart(false)}
\]


\[
\text{IF oFeld.Text <> "" AND stSuchtext <> "" THEN}
\text{IF inStr(oFeld.Text, stSuchtext) THEN}
\text{inSuch = 1}
\text{inSuchAlt = 0}
\text{inLen = Len(stSuchtext)}
\]


\[
\text{DO WHILE inStr(inSuch, oFeld.Text, stSuchtext) > 0}
\text{inSuch = inStr(inSuch, oFeld.Text, stSuchtext) - 1}
\text{oCursor.goRight(inSuch-inSuchAlt,false)}
\text{oCursor.goRight(inLen, true)}
\text{oCursor.CharColor = RGB(102,102,255)}
\text{oCursor.CharWeight = 110.000000}
\text{oCursor.goLeft(inLen,false)}
\text{inSuchAlt = inSuch}
\text{inSuch = inSuch + 2}
\text{LOOP}
\text{END IF}
\text{END IF}
\]

Makros: Bedienbarkeit verbessern

Die Prozedur ist an das folgende Ereignis gebunden: **Formular → Ereignisse → Vor dem Daten- satzwechsel**

```vba
Sub InhaltSchreiben(oEvent AS OBJECT)
    Dim oForm AS OBJECT
    Dim inMemo AS INTEGER
    Dim loID AS LONG
    Dim oFeld AS OBJECT
    Dim stMemo AS STRING
    oForm = oEvent.Source
    If InStr(oForm.ImplementationName, "ODatabaseForm") Then
        If Not oForm.isBeforeFirst() And Not oForm.isAfterLast() Then
            inMemo = oForm.findColumn("Memo")
            loID = oForm.findColumn("ID")
            oFeld = oForm.getByName("MemoFormat")
            stMemo = oFeld.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
```

Das auslösende Ereignis ist doppelt belegt. Nur der Implementationsname, der auf **ODatabaseForm** endet, gibt den richtigen Zugriff auf den Datensatz.

Beim Einlesen, auch beim Reload des Formulars, steht der Cursor vor dem ersten Datensatz. Würde jetzt ein Schreibversuch unternommen, dann erscheint die Meldung «ungültiger Cursorstatus».

```vba
inMemo = oForm.findColumn("Memo")
oFeld = oForm.getByName("MemoFormat")
stMemo = oFeld.Text
If stMemo <> "" Then
    oForm.updateString(inMemo, stMemo)
End If
```

Das Tabellenfeld "Memo" wird aus der Datenquelle des Formulars herausgesucht. Ebenso das Feld "ID". Befindet sich im Feld «MemoFormat» Text, so wird er mit **oForm.updateString** in das Feld "Memo" der Datenquelle übertragen. Nur wenn bereits ein Eintrag im Feld "ID" existiert, also der Primärschlüssel belegt ist, erfolgt ein Update. Ansonsten wird ja sowieso ein neuer Datensatz über die Formularfunktionen eingefügt, da das Formular die Änderung entsprechend bemerkt und eine Abspeicherung selbständig vornimmt.

**Rechtschreibkontrolle während der Eingabe**

Auf mehrzeilige Textfelder mit Formatierungen greift auch dieses Makro zu. Entsprechend muss auch, wie bei dem vorherigen Kapitel, der Inhalt bei jedem Datensatzwechsel zuerst geschrieben und danach der Inhalt des neuen Datensatzes in das Formularfeld geladen werden. Die Prozeduren «InhaltUebertragen» und «InhaltSchreiben» unterscheiden sich höchstens in dem Punkt, dass die Suchfunktion ausgeklammert werden kann.²⁰

²⁰ Siehe auch hierzu: «Beispiel_Autotext_Suchmarkierung_Rechtschreibung.odb»
Die Rechtschreibkontrolle wird in dem obigen Formular dadurch ausgelöst, dass in dem Formularfeld entweder eine Leertaste oder ein Return betätigt wird. Sie läuft also nach der Beendigung eines Wortes jedes Mal ab und könnte gegebenenfalls auch noch mit dem Fokusverlust des Formularfeldes gekoppelt werden, damit auch das letzte Wort sicher überprüft wird.

Die Prozedur ist an das folgende Ereignis gebunden: Formular → Ereignisse → Taste losgelassen

**SUB** MarkierungFehlerDirekt(oEvent AS OBJECT)

GlobalScope.BasicLibraries.LoadLibrary("Tools")

Es wird die Funktion **RTrimStr** zum Entfernen von Satzzeichen am Ende vor Worten benötigt. Sonst werden alle Worte, denen ein Komma, Punkt oder irgendein anderes Satzzeichen folgt, als falsch angesehen. Mit **LTrimChar** müssen außerdem Klammern zum Beginn des Wortes entfernt werden.

```
DIM aProp() AS NEW com.sun.star.beans.PropertyValue
DIM oLinguSvcMgr AS OBJECT
DIM oSpellChk AS OBJECT
DIM oFeld AS OBJECT
DIM arText() =
DIM stWort AS STRING
DIM inlenWort 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
```

Zuerst werden alle Variablen deklariert. Danach wird auf das Rechtschreibüberprüfungsmodul **SpellChecker** zugegriffen. Mit diesem Modul werden anschließend die einzelnen Worte auf ihre Richtigkeit hin überprüft.

```
oFeld = oEvent.Source.Model
ink = 0
IF oEvent.KeyCode = 1280 OR oEvent.KeyCode = 1284 THEN
```

Das Ereignis, das das Makro auslöst, ist ein Tastendruck. Zu dem Ereignis wird ein Code für jede Taste mitgeliefert, der **KeyCode**. Der **KeyCode** für die Returntaste ist 1280, der für die Leertaste ist 1284. Wie viele andere Informationen sind diese Informationen einfach durch das Tool «Xray» gewonnen worden. Wird also eine Leertaste oder die Returntaste betätigt, so wird die Rechtschreibung überprüft. Sie startet also zu jedem Wortende. Lediglich die Überprüfung für das letzte Wort ist so nicht automatisch möglich.

Bei jedem Durchlauf werden alle Worte des Textes überprüft. Die Überprüfung einzelner Worte könnte eventuell auch möglich sein, bedeutet aber erheblich mehr Aufwand.
Der Text wird also in Worte aufgesplittet. Trenner ist hier das Leerzeichen. Vorher müssen allerdings noch Trennungen an Zeilenumbrüchen erzeugt werden, die sonst später als ein Wort wahrgenommen werden.

\[
stText = \text{Join}(\text{Split}(\text{oField.Text}, \text{CHR}(10)), " ")
\]
\[
stText = \text{Join}(\text{Split}(\text{stText}, \text{CHR}(13)), " ")
\]
\[
arText = \text{Split}(\text{RTrim}(\text{stText}), " ")
\]
\[
\text{FOR} \ i = \text{LBound}(\text{arText}) \ \text{TO} \ \text{Ubound}(\text{arText})
\]
\[
\text{stWort} = \text{arText}(i)
\]
\[
inlenWort = \text{len}(\text{stWort})
\]
\[
\text{stWort} = \text{Trim}(\text{RtrimStr}(\text{RtrimStr}(\text{RtrimStr}(\text{RtrimStr}(\text{RtrimStr}(\text{RtrimStr}(\text{stWort}, ""), ","), ","), ",?"), ",!"), ",."), ","))
\]
\[
\text{stWort} = \text{LTrimChar}(\text{stWort}, "")
\]

Das einzelne Wort wird ausgelesen, seine ungekürzte Länge ist notwendig für die folgenden Bearbeitungsschritte. Nur so kann die Position des Wortes innerhalb des gesamten Textes bestimmt werden, die auch für die gezielte Markierung von Schreibfehlern gebraucht wird.

Mit \text{Trim} werden Leerzeichen entfernt, mit der Funktion \text{RTrimStr} Kommas und Satzzeichen am Ende des Textes, mit der Funktion \text{LTrimChar} Zeichen am Anfang des Textes.

\[
\text{IF} \ \text{stWort} <> "" \ \text{THEN}
\]
\[
\text{oCursor} = \text{oFeld.createTextCursor()}
\]
\[
\text{oCursor.goStart(false)}
\]
\[
\text{oCursor.goRight(ink,false)}
\]
\[
\text{oCursor.goRight(inLenWort, true)}
\]
\[
\text{If Not oSpellChk.isValid(stWort, \text{"de"}, aProp()) Then}
\]
\[
\text{oCursor.CharUnderline = 9}
\]
\[
\text{oCursor.CharUnderlineHasColor = True}
\]
\[
\text{oCursor.CharUnderlineColor = RGB(255,51,51)}
\]
\[
\text{ELSE}
\]
\[
\text{oCursor.CharUnderline = 0}
\]
\[
\text{END IF}
\]
\[
\text{END IF}
\]
\[
\text{ink = ink + inLenWort + 1}
\]
\[
\text{NEXT}
\]
\[
\text{END IF}
\]
\[
\text{END SUB}
\]

Hat das Wort einen Inhalt, so wird zuerst einmal ein Textcursor erstellt. Der Textcursor wird ohne Markierung an den Start des Textes in dem Eingabefeld gesetzt. Dann geht es, immer noch ohne Markierung, um den Betrag nach rechts im Text vorwärts, der in der Variablen \text{ink} gespeichert ist. Diese Variable ist am Anfang 0, nach Durchlaufen der ersten Schleife dann so groß wie das vorhergehende Wort lang war +1 für das angehängte Leerzeichen. Dann wird der Cursor mit Markierung um die Länge des aktuellen Wortes weiter gesetzt. Erfolgt jetzt eine Änderung der Buchstaben-Eigenschaften, so betrifft diese nur den markierten Bereich.


**Kombinationsfelder als Listenfelder mit Eingabemöglichkeit**

Aus Kombinationsfeldern und Tabellenfeldern aus dem Formular kann direkt eine Tabelle mit einem Datensatz versehen werden und der entsprechende Primärschlüssel in eine andere Tabelle eingetragen werden.\textsuperscript{21}

\textsuperscript{21} Die Beispieldatenbank «Beispiel_Combobox_Listfeld.odb» zum Einsatz von Kombinationsfeldern statt Listenfeldern ist den Beispieldatenbanken für dieses Handbuch beigefügt.

Werden statt der Tabellen entsprechend konstruierte eingabefähige Abfragen erstellt, so kann der Text, den die Kombinationsfelder darstellen sollen, direkt aus den Abfragen ermittelt werden. Ein Makro ist dann für diesen Arbeitsschritt nicht notwendig.

Voraussetzung für die Funktionsweise des Makros ist, dass alle Primärschlüssel der Tabellen, die in den Kombinationsfeldern als Datenquellen auftauchen, mit einem automatisch hochzählenden Autowert versehen sind. Außerdem ist als Bezeichnung hier vorausgesetzt, dass die Primärschlüssel den Namen "ID" tragen.

**Textanzeige im Kombinationsfeld**


```vba
SUB TextAnzeigen(oEvent AS OBJECT)
    Dieses Makro sollte an das folgende Ereignis des Formulars gebunden werden: 'Nach dem Datensatzwechsel'
    Das Makro wird direkt aus dem Formular angesprochen. Über das auslösende Ereignis werden die gesamten notwendigen Variablen für das Makro ermittelt.
    Die Variablen werden deklariert. Einige Variablen sind in einem separaten Modul bereits global deklariert und werden hier nicht noch einmal erwähnt.

    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oFeldList AS OBJECT
    DIM stAbfrage AS STRING
    DIM stFeldWert AS STRING
    DIM stFeldID AS STRING
    DIM inCom AS INTEGER
    oForm = oEvent.Source

    Das Formular startet das Ereignis. Es ist die Quelle für das das Makro auslösende Ereignis.
    In dem Formular befindet sich ein verstecktes Kontrollerelement, aus dem hervor geht, wie die verschiedenen Kombinationsfelder in diesem Formular heißen. Nacheinander werden dann in dem Makro die Kombinationsfelder abgearbeitet.

    aComboboxen() = Split(oForm.getByName("Comboboxen").Tag, ",")
    FOR inCom = LBound(aComboboxen) TO Ubound(aComboboxen)
        ...
        NEXT inCom

    Aus den Zusatzinformationen («Tag») des versteckten Kontrollerelementes wird die Bezeichnung der Kombinationsfelder ermittelt. Sie sind dort durch Kommas voneinander getrennt aufgeschrieben. Die Namen der Felder werden in ein Array geschrieben und nacheinander in einer Schleife abgearbeitet. Die Schleife endet mit der Bezeichnung NEXT ...

    Das Kombinationsfeld, das jetzt statt eines Listenfeldes existiert, wird anschließend als oFeldList bezeichnet. Der Fremdschlüssel wird über die Bezeichnung des Tabellenfeldes, die in den Zusatzinformationen («Tag») der Kombinationsfelder hinterlegt ist, bestimmt.

Makros: Bedienbarkeit verbessern  429
den Zusatzinformationen des Kombinationsfeldes steht, aus der Tabellenspalte des Formulars ermittelt.

```plaintext
oFeldList = oForm.getByName(Trim(aComboboxen(inCom)))
stFeldID = oForm.getString(oForm.findColumn(oFeldList.Tag))
oFeldList.Refresh()
```

Das Kombinationsfeld wird mit `Refresh()` neu eingelesen. Es kann ja sein, dass sich der Inhalt des Feldes durch Neueingaben geändert hat. Diese müssen schließlich verfügbar gemacht werden.


Existiert keine Beziehungsdefinition, so beruht die Abfrage nur auf einer Tabelle. Die Tabelleninformation kann entfallen, die Bedingung direkt mit dem Fremdschlüsselwert zusammen formuliert werden.

```plaintext
IF stFeldID <> "" THEN
    stAbfrage = oFeldList.ListSource
    IF InStr(stAbfrage,"order by") > 0 THEN
        stSql = Left(stAbfrage, InStr(stAbfrage,"order by")-1)
    ELSE
        stSql = stAbfrage
    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"" = "+stFeldID
    ELSE
        stSql = stSql + "WHERE "]ID"" = "+stFeldID
    END IF
```

Jedes Feld und jeder Tabellenname muss bereits in der SQL-Eingabe mit doppelten Anführungsstrichen oben versehen werden. Da bereits Anführungsstriche einfacher Art in Basic als die Einführung zu Text interpretiert werden, sind diese bei der Weitergabe des Codes nicht mehr sichtbar. Erst bei einer Doppellierung der Anführungsstriche wird ein Element mit einfachen Anführungsstrichen weitergegeben. """ID"" bedeutet also, dass in der Abfrage auf das Feld "ID" (mit einfachen Anführungsstrichen für die SQL-Verbindung) zugegriffen wird.

Die in der Variablen `stSql` abgespeicherte Abfrage wird jetzt ausgeführt und das Ergebnis dieser Abfrage in der Variablen `oAbfrageergebnis` gespeichert.

```plaintext
oDatenquelle = ThisComponent.Parent.CurrentController
IF NOT (oDatenquelle.isConnected()) Then
    oDatenquelle.connect()
End If
oVerbindung = oDatenquelle.ActiveConnection()
oSQL_Anweisung = oVerbindung.createStatement()
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
```

Das Abfrageergebnis wird über eine Schleife ausgelesen. Hier könnten, wie in einer Abfrage aus der GUI, mehrere Felder und Datensätze dargestellt werden. Von der Konstruktion der Abfrage her
wird aber nur ein Ergebnis erwartet. Dieses Ergebnis wird in der ersten Spalte (1) der Abfrage zu finden sein. Es ist der Datensatz, der den anzuzeigenden Inhalt des Kombinationsfeldes widerruft. Der Inhalt ist ein Textinhalt (getString()), deshalb hier oAbfrageergebnis.getString(1).

```vbnet
    WHILE oAbfrageergebnis.next
        stFeldWert = oAbfrageergebnis.getString(1)
    WEND
```

Das Kombinationsfeld muss jetzt auf den aus der Abfrage sich ergebenden Textwert eingestellt werden.

```vbnet
    oFeldList.Text = stFeldWert
ELSE
    oFeldList.Text = ""
END IF
```

Falls überhaupt kein Wert in dem Feld für den Fremdschlüssel oFeld vorhanden ist, ist auch die ganze Abfrage nicht gelaufen. Das Kombinationsfeld wird jetzt auf eine leere Anzeige eingestellt.


### Fremdschlüsselwert vom Kombinationsfeld zum numerischen Feld übertragen

Wird nun ein neuer Wert ausgewählt oder neu in das Kombinationsfeld eingegeben (nur wegen dieser Eigenschaft wurde ja das Makro konstruiert), so muss der entsprechende Primärschlüssel als Fremdschlüssel in die dem Formular zugrundeliegende Tabelle eingetragen werden.

```vbnet
    SUB TextAuswahlWertSpeichern(oEvent AS OBJECT)
    ... IN diese Schleife wird der gleiche Start wie bei der Prozedur TextAnzeigen eingebaut:
    a() = Split(Trim(aComboxoxen(inCom))),">
    IF Ubound(a) > 0 THEN
        oFeldList = oForm.getByName(a(0)).getByName(a(1))
    ELSE
```
Anschließend wird die Abfrage aus dem Kombinationsfeld ausgelesen und in ihre Einzelteile zerlegt. Bei einfachen Kombinationsfeldern wären die notwendigen Informationen lediglich der Feldname und der Tabellenname:

```
SELECT "Feld" FROM "Tabelle"
```

Dies könnten gegebenenfalls noch durch eine Sortierung erweitert sein. Sobald zwei Felder in dem Kombinationsfeld zusammen dargestellt werden, muss aber bereits bei den Feldern zur Trennung entsprechend mehr Aufwand getrieben werden:

```
SELECT "Feld1" || ' ' || "Feld2" FROM "Tabelle"
```


Noch komplizierter wird es, wenn die beiden enthaltenen Felder aus zwei verschiedenen Tabellen stammen:

```
SELECT "Tabelle1"."Feld1" || ' > ' || "Tabelle2"."Feld2"
FROM "Tabelle1", "Tabelle2"
WHERE "Tabelle1"."ID" = "Tabelle2"."FremdID"
ORDER BY "Tabelle1"."Feld1" || ' > ' || "Tabelle2"."Feld2" ASC
```

Hier müssen die Felder voneinander getrennt werden, die Tabellenzuordnungen zu den Feldern erfasst werden und die Fremdschlüsselzuweisung ermittelt werden.

```
ostAbfrage = oFeldList.ListSource
aFelder() = Split(stAbfrage, ""
stInhalt = ""
FOR i = LBound(aFelder) + 1 TO UBound(aFelder)
    IF Trim(UCASE(aFelder(i))) = "ORDER BY" THEN
        EXIT FOR
    ELSEIF Trim(UCASE(aFelder(i))) = "FROM" THEN
        stInhalt = stInhalt + " §§ \\
    ELSEIF Trim(UCASE(aFelder(i))) = "WHERE" THEN
        stInhalt = stInhalt + " §§ \\
    ELSE
        stInhalt = stInhalt + Trim(aFelder(i))
    END IF
NEXT i
aInhalt() = Split(stInhalt, " §§ \\
```

Die sichtbare Feldanzeige wird gegebenenfalls in Inhalte aus verschiedenen Feldern aufgeteilt:

```
aErster() = Split(aInhalt(0),"
```

Der erste Teil enthält mindestens 2 Felder. Die Felder haben zu Beginn eine Tabellenbezeichnung. Der zweite Teil enthält zwei Tabellenbezeichnungen, die aber schon aus dem ersten Teil ermittelt werden können. Der dritte Teil enthält eine Beziehung über einen Fremdschlüssel mit «=>» getrennt:

```
aTest() = Split(aErster(0),"
NameTabelle1 = aTest(0)
```

Makros: Bedienbarkeit verbessern
NameTabellenFeld1 = aErster(1)
Erase aTest
Feldtrenner = Join(Split(aErster(1),""),"")
aTest() = Split(aErster(2),"")
NameTabelle2 = aTest(0)
NameTabellenFeld2 = aTest(1)
Erase aTest
aTest() = Split(aInhalt(2),"")
aTest1() = Split(aTest(0),"")
IF aTest1(1) <> "ID" THEN
   NameTab12ID = aTest1(1)
   IF aTest1(0) = NameTabelle1 THEN
      Position = 2
   ELSE
      Position = 1
   END IF
ELSE
   Erase aTest1
   aTest1() = Split(aTest1(1),".")
   NameTab12ID = aTest1(1)
   IF aTest1(0) = NameTabelle1 THEN
      Position = 2
   ELSE
      Position = 1
   END IF
END IF
ELSE
Der erste Teil enthält zwei Feldbezeichnungen ohne Tabellenbezeichnung, eventuell mit Trenner, der zweite Teil enthält die Tabellenbezeichnung. Ein dritter Teil ist nicht vorhanden:

IF UBound(aErster) > 1 THEN
   NameTabellenFeld1 = aErster(0)
   Feldtrenner = Join(Split(aErster(1),""),"")
   NameTabellenFeld2 = aErster(2)
ELSE
   NameTabellenFeld1 = aErster(0)
   NameTabellenFeld2 = aErster(1)
END IF
END IF

Es existiert nur ein Feld, das aus einer Tabelle stammt:

NameTabellenFeld1 = aErster(0)
NameTabelle1 = aInhalt(1)
END IF

Die maximale Zeichenlänge, die eine Eingabe haben darf, wird im Folgenden mit der Funktion Spaltengroesse ermittelt. Das Kombinationsfeld kann hier mit seiner Beschränkung nicht sicher weiterhelfen, da ja ermöglicht werden soll, gleichzeitig 2 Felder in einem Kombinationsfeld einzutragen.

LaengeFeld1 = Spaltengroesse(NameTabelle1,NameTabellenFeld1)
IF NameTabellenFeld2 <> "" THEN
   IF NameTabellenFeld2 <> "" THEN
      LaengeFeld2 = Spaltengroesse(NameTabelle2,NameTabellenFeld2)
   ELSE
      LaengeFeld2 = Spaltengroesse(NameTabelle1,NameTabellenFeld2)
   END IF
ELSE
   LaengeFeld2 = 0
END IF

Der Inhalt des Kombinationsfeldes wird ausgelesen:

stInhalt = oFeldList.getCurrentValue()

Der angezeigte Inhalt des Kombinationsfeldes wird ausgelesen. Leertasten und nicht druckbare Zeichen am Anfang und Ende der Eingabe werden gegebenenfalls entfernt.
StInhalt = Trim(stInhalt)
IF stInhalt <> "" THEN
  IF NameTabellenFeld2 <> "" THEN
    Wenn ein zweites Tabellenfeld existiert muss der Inhalt des Kombinationsfeldes gesplittet werden. Um zu wissen, an welcher Stelle die Aufteilung vorgenommen werden soll, ist der Feldtrenner von Bedeutung, der der Funktion als Variable mitgegeben wird.
    a_stTeile = Split(stInhalt, Feldtrenner, 2)
    Der letzte Parameter weist darauf hin, dass maximal 2 Teile erzeugt werden.
    IF Position = 2 THEN
      stInhalt = Trim(a_stTeile(0))
      IF UBound(a_stTeile()) > 0 THEN
        stInhaltFeld2 = Trim(a_stTeile(1))
      ELSE
        stInhaltFeld2 = ""
      END IF
    END IF
    ELSE
      stInhaltFeld2 = Trim(a_stTeile(0))
      IF UBound(a_stTeile()) > 0 THEN
        stInhalt = Trim(a_stTeile(1))
      ELSE
        stInhalt = ""
      END IF
    END IF
    END IF
  ELSE
    stInhaltFeld2 = Trim(a_stTeile(1))
  END IF
ELSE
  stInhalt = Trim(a_stTeile(1))
END IF
Es kann passieren, dass bei zwei voneinander zu trennenden Inhalten die Größeneinstellung des Kombinationsfeldes (Textlänge) nicht zu einem der abzuspeichernden Tabellenfelder passt. Bei Kombinationsfeldern für nur ein Feld wird dies in der Regel durch Einstellungen im Formularkontrollfeld erledigt. Hier muss hingegen ein eventueller Fehler abgefangen werden. Es wird darauf hingewiesen, wie lang der Inhalt für das jeweilige Feld sein darf.

IF (LaengeFeld1 > 0 AND Len(stInhalt) > LaengeFeld1) OR (LaengeFeld2 > 0 AND Len(stInhaltFeld2) > LaengeFeld2) THEN
  Es fügt hier einen Zeilenumbruch hinzu.
  stmsgbox1 = "Das Feld " + NameTabellenFeld1 + " darf höchstens " + LaengeFeld1 + " Zeichen lang sein." + CHR(13)
  stmsgbox2 = "Das Feld " + NameTabellenFeld2 + " darf höchstens " + LaengeFeld2 + " Zeichen lang sein." + CHR(13)
  Sind beide Feldinhalte zu lang, so wird der Text mit beiden Feldinhalten ausgegeben.
  IF (LaengeFeld1 > 0 AND Len(stInhalt) > LaengeFeld1) AND (LaengeFeld2 > 0 AND Len(stInhaltFeld2) > LaengeFeld2) THEN
    Die Anzeige erfolgt mit der Funktion 'msgbox()'. Sie erwartet zuerst einen Text, dann optional einen Zahlenwert (der zu einer entsprechenden Darstellungsform gehört) und schließlich optional einen Text als Überschrift über dem Fenster. Das Fenster hat hier also die Überschrift "Fehlerhafte Eingabe", die '64' fügt das Informationssymbol hinzu.
    msgbox ("Der eingegebene Text ist zu lang." + CHR(13) + stmsgbox1 + stmsgbox2 + "Bitte den Text kürzen.", 64,"Fehlerhafte Eingabe")
    Die Anzeige erfolgt mit der Funktion 'msgbox()'. Sie erwartet zuerst einen Text, dann optional einen Zahlenwert (der zu einer entsprechenden Darstellungsform gehört) und schließlich optional einen Text als Überschrift über dem Fenster. Das Fenster hat hier also die Überschrift "Fehlerhafte Eingabe", die '64' fügt das Informationssymbol hinzu.
    Im Folgenden werden alle auftretenden weiteren Fälle zu großer Textlänge abgearbeitet.
    ELSEIF (LaengeFeld1 > 0 AND Len(stInhalt) > LaengeFeld1) THEN
      msgbox ("Der eingegebene Text ist zu lang." + CHR(13) + stmsgbox1 + "Bitte den Text kürzen.",64,"Fehlerhafte Eingabe")
    ELSE
      Makros: Bedienbarkeit verbessern

434
Liegst kein zu langer Text vor, so kann die Funktion weiter durchlaufen. Ansonsten endet sie hier.

Jetzt werden die Inhaltseingaben so maskiert, dass eventuell vorhandene Hochkommata keine Fehlermeldung erzeugen.

```vba
stInhalt = String_to_SQL(stInhalt)
IF stInhaltFeld2 <> "" THEN
    stInhaltFeld2 = String_to_SQL(stInhaltFeld2)
END IF
```

Zuerst werden Variablen vorbelegt, die anschließend per Abfrage geändert werden können. Die Variablen `inID1` und `inID2` sollen den Inhalt der Primärschlüsselfelder der beiden Tabellen speichern. Da bei einer Abfrage, die kein Ergebnis wiedergibt, durch Basic einer Integer-Variablen 0 zugewiesen wird, dies aber für das Abfrageergebnis auch bedeuten könnte, dass der ermittelte Primärschlüssel eben den Wert 0 hat, wird die Variable auf jeweils -1 voreingestellt. Diesen Wert nimmt ein Autowert-Feld bei der HSQLDB nicht automatisch an.

Anschließend wird die Datenbankverbindung erzeugt, soweit sie nicht schon besteht.

```vba
inID1 = -1
inID2 = -1
oDatenquelle = ThisComponent.Parent.CurrentController
IF NOT (oDatenquelle.isConnected()) Then
    oDatenquelle.connect()
End If
oVerbindung = oDatenquelle.ActiveConnection()
oSQL_Anweisung = oVerbindung.createStatement()
IF NameTabellenFeld2 <> "" AND NOT IsEmpty(stInhaltFeld2) AND NameTabelle2 <> "" THEN
    stSql = "SELECT "ID" FROM " + NameTabelle2 + "" WHERE "" + NameTabellenFeld2 + ""='" + stInhaltFeld2 + "'
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
    WHILE oAbfrageergebnis.next
        inID2 = oAbfrageergebnis.getInt(1)
    WEND
    IF inID2 = -1 THEN
        stSql = "INSERT INTO "" + NameTabelle2 + ""("" + NameTabellenFeld2 + "") VALUES ('" + stInhaltFeld2 + "') "
        oSQL_Anweisung.executeUpdate(stSql)
        stSql = "CALL IDENTITY()"
    END IF
```


```vba
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
    WHILE oAbfrageergebnis.next
        inID2 = oAbfrageergebnis.getInt(1)
    WEND
    END IF
```

Der Primärschlüssel aus dem zweiten Wert wird in der Variablen `inID2` zwischengespeichert. Jetzt wird überprüft, ob eventuell dieser Schlüsselwert bereits in der Tabelle 1 vorhanden ist.

**Beispiel:** Die Tabelle 2 ist die Tabelle "Ort". In ihr werden also Orte abgespeichert. Ist z.B. ein Eintrag für den Ort 'Rheine' vorhanden, so wird der entsprechende Primärschlüsselwert ausgelesen. Ist der Eintrag 'Rheine' nicht vorhanden, wird er eingefügt und anschließend der beim Einfügen erzeugte Primärschlüsselwert festgestellt.

```vba
stSql = "SELECT "ID" FROM " + NameTabelle2 + "" WHERE "" + NameTabellenFeld2 + ""='" + stInhaltFeld2 + "'
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
    WHILE oAbfrageergebnis.next
        inID2 = oAbfrageergebnis.getInt(1)
    WEND
    IF inID2 = -1 THEN
        stSql = "INSERT INTO "" + NameTabelle2 + ""("" + NameTabellenFeld2 + "") VALUES ('" + stInhaltFeld2 + "') "
        oSQL_Anweisung.executeUpdate(stSql)
        stSql = "CALL IDENTITY()"
    END IF
```

Eintrag aus dem ersten Feld vorhanden ist. Ist diese Kombination nicht vorhanden, so wird sie neu eingefügt.

Beispiel: Für den Ort 'Rheine' aus der Tabelle 2 können in der Tabelle 1 mehrere Postleitzahlen verfügbar sein. Ist die Kombination '48431' und 'Rheine' vorhanden, so wird nur der Primärschlüssel aus der Tabelle 1 ausgelesen, in der die Postleitzahlen und der Fremdschlüssel aus der Tabelle 2 gespeichert wurden.

```
stSql = "SELECT "ID" FROM " + NameTabelle1 + "" WHERE "" + NameTab1ID + "" = '" + inID2 + "" AND "" + NameTabellenFeld1 + "" = '" + stInhalt + ""
```

```
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
```

```
WHILE oAbfrageergebnis.next
  inID1 = oAbfrageergebnis.getInt(1)
WEND
```

War der Inhalt der ersten Tabelle noch nicht vorhanden war, so wird der Inhalt neu abgespeichert (INSERT).

Beispiel: Existiert bereits die Postleitzahl '48429' in Kombination mit dem Fremdschlüssel aus der Tabelle 2 "Ort", so wird auf jeden Fall ein neuer Datensatz erzeugt, wenn jetzt die Postleitzahl '48431' auftaucht. Der vorhergehende Datensatz wird also nicht auf die neue Postleitzahl geändert. Schließlich sind durch die n:1-Verknüpfung der Tabellen mehrere Postleitzahlen für einen Ort ermöglicht worden.

```
IF inID1 = -1 THEN
  stSql = "INSERT INTO ""” + NameTabelle1 + "” ("” + NameTabellenFeld1 + "” + NameTabellenFeld2 + "”)
  VALUES (’” + stInhalt + ‘,”’ + inID2 + ‘’)
```

```
oSQL_Anweisung.executeUpdate(stSql)
```

Der Primärschlüssel der ersten Tabelle muss schließlich wieder ausgelesen werden, damit er in die dem Formular zugrundeliegende Tabelle übertragen werden kann.

```
stSql = "CALL IDENTITY()"
```

```
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
```

```
WHILE oAbfrageergebnis.next
  inID1 = oAbfrageergebnis.getInt(1)
WEND
END IF
END IF
```

Für den Fall, dass beide in dem Kombinationsfeld zugrundeliegenden Felder in einer Tabelle gespeichert sind (z.B. Nachname, Vorname in der Tabelle Name) muss eine andere Abfrage erfolgen:

```
IF NameTabellenFeld2 <> ‘” AND NameTabelle2 = ‘” THEN
```

```
stSql = "SELECT "ID" FROM " + NameTabelle1 + "" WHERE "" + NameTabellenFeld1 + "” + stInhalt + "” AND "” + NameTabellenFeld2 + "” + stInhalt + "”
```

```
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
```

```
WHILE oAbfrageergebnis.next
  inID1 = oAbfrageergebnis.getInt(1)
WEND
END IF
```

... und eine zweite Tabelle nicht existiert:

```
stSql = "INSERT INTO "” + NameTabelle1 + "” (”” + NameTabellenFeld1 + "” + NameTabellenFeld2 + "”)
  VALUES (’” + stInhalt + ‘,”’ + inID2 + ‘’)
```

```
oSQL_Anweisung.executeUpdate(stSql)
```

Anschließend wird das Primärschlüsselfeld wieder ausgelesen.

```
stSql = "CALL IDENTITY()"
```

```
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
```

```
WHILE oAbfrageergebnis.next
  inID1 = oAbfrageergebnis.getInt(1)
WEND
```
Jetzt wird der Fall geklärt, der der einfachste ist: Das 2. Tabellenfeld existiert nicht und der Eintrag ist noch nicht in der Tabelle vorhanden. In das Kombinationsfeld ist also ein einzelner neuer Wert eingetragen worden.

```sql
stSql = "SELECT "ID" FROM " + NameTabelle1 + "," + NameTabellenFeld1 + "," + stInhalt + ""
```

Wenn ein zweites Tabellenfeld nicht existiert wird der Inhalt neu eingefügt ...

```sql
stSql = "CALL IDENTITY()"
```

Der Wert des Primärschlüsselfeldes muss ermittelt werden, damit er in die Haupttabelle des Formulars übertragen werden kann.

Anschließend wird der aus all diesen Schleifen ermittelte Primärschlüsselwert in das Feld der Haupttabelle und die darunterliegende Datenbank übertragen. Mit `findColumn` wird das mit dem Formularfeld verbundene Tabellenfeld erreicht. Mit `updateLong` wird eine Integer-Zahl (siehe Zahlendefinition) diesem Feld zugewiesen.

```vbnet
oForm.updateLong(oForm.findColumn(oFeldList.Tag), inID1)
```

Ist kein Primärschlüsselwert einzutragen, weil auch kein Eintrag in dem Kombinationsfeld erfolgte oder dieser Eintrag gelöscht wurde, so ist auch der Inhalt des Feldes zu löschen. Mit `updateNULL()` wird das Feld mit dem databankspezifischen Ausdruck für ein leeres Feld, `NULL`, versehen.

```vbnet
oForm.updateNULL(oForm.findColumn(oFeldList.Tag), NULL)
```

Kontrollfunktion für die Zeichenlänge der Kombinationsfelder

Die folgende Funktion soll die Zeichenlänge der jeweiligen Tabellenspalten ermitteln, damit zu lange Eingaben nicht einfach gekürzt werden. Der Typ `FUNCTION` wurde hier wegen der Rückgabewerte gewählt.

```vbnet
FUNCTION Spaltengroesse(Tabellenname AS STRING, Feldname AS STRING) AS INTEGER
    oDatenquelle = ThisComponent.Parent.CurrentController
    If NOT (oDatenquelle.isConnected()) Then
        oDatenquelle.connect()
    End If
    oVerbindung = oDatenquelle.ActiveConnection()
    oSQL_Anweisung = oVerbindung.createStatement()
    stSql = "SELECT "COLUMN_SIZE" FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS"
        WHERE "TABLE_NAME" = '" + Tabellenname + "' AND "COLUMN_NAME" = '" + Feldname + "'
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
    WEND
    inID1 = oAbfrageergebnis.getInt(1)
END IF
```

Makros: Bedienbarkeit verbessern
+ Feldname + "'"
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
WHILE oAbfrageergebnis.next
    i = oAbfrageergebnis.getInt(1)
WEND
Spaltengroesse = i
END FUNCTION

Datensatzaktion erzeugen

SUB Datensatzaktion_erzeugen(oEvent AS OBJECT)

Dieses Makro sollte an das folgende Ereignis des Listenfeldes gebunden werden: 'Bei Fokuserhalt'. Es ist notwendig, damit auf jeden Fall bei einer Änderung des Listenfeldinhaltes die Speicherung abläuft. Ohne dieses Makro wird keine Änderung in der Tabelle erzeugt, die für Base wahrnehmbar ist, da die Combobox mit dem Formular nicht verbunden ist.

Dieses Makro stellt direkt die Eigenschaft des Formulars um.

    DIM oForm AS OBJECT
    oForm = oEvent.Source.Model.Parent
    oForm.IsModified = TRUE

END SUB

Bei Formularen, die bereits ihren Inhalt auch für die Kombinationsfelder aus Abfragen erhalten, ist dieses Makro nicht notwendig. Änderungen in den Kombinationsfeldern werden direkt registriert.

Navigation von einem Formular zum anderen

Ein Formular soll über ein entsprechendes Ereignis geöffnet werden.

Im Formularkontrollfeld wird unter den Eigenschaften in der Zeile "Zusatzinformationen" (Tag) hier der Name des Formulars eingetragen. Hier können auch weitere Informationen eingetragen werden, die über den Befehl Split() anschließend voneinander getrennt werden.

SUB Zu_Formular_von_Formular(oEvent AS OBJECT)
    REM Das zu öffnende Formular wird als erstes angegeben.
    REM Liegt ein Formular in einem Ordner, so ist die Beziehung über "/" zu definieren.
    REM so dass der Unterordner zu finden ist.
    DIM stTag AS STRING
    stTag = oEvent.Source.Model.Tag
    aForms() = Split(stTag, ",")

Das Array wird gegründet und mit den Formularnamen gefüllt, in diesem Fall zuerst in dem zu öffnenden Formular und als zweites dem aktuellen Formular, dass nach dem Öffnen des anderen geschlossen werden soll.

    ThisDatabaseDocument.FormDocuments.getByName( Trim(aForm(0)) ).open
    ThisDatabaseDocument.FormDocuments.getByName( Trim(aForm(1)) ).close

END SUB

Soll stattdessen nur beim Schließen ein anderes Formular geöffnet werden, weil z.B. ein Hauptformular existiert und alle anderen Formulare von diesem aus über entsprechende Buttons angesteuert werden, so ist das folgende Makro einfach an das Formular unter Extras → Anpassen → Ereignisse → Dokument wird geschlossen anzubinden:

SUB Hauptformular_oeffnen
    ThisDatabaseDocument.FormDocuments.getByName( "Hauptformular" ).open
END SUB

Wenn die Formulardokumente innerhalb der *.odb-Datei in Verzeichnissen sortiert sind, so muss das Makro für den Formularwechsel etwas umfangreicher sein:

SUB Zu_Formular_von_Formular_mit_Ordner(oEvent AS OBJECT)
    REM Das zu öffnende Formular wird als erstes angegeben.
    REM Liegt ein Formular in einem Ordner, so ist die Beziehung über "/" zu definieren.
    REM so dass der Unterordner zu finden ist.
    DIM stTag AS STRING
    stTag = oEvent.Source.Model.Tag
    aForms() = Split(stTag, ",")

Makros: Bedienbarkeit verbessern 438
Formularedokumente, die in einem Verzeichnis liegen, werden in den Zusatzinformationen als Verzeichnis/Formular angegeben. Dies muss umgewandelt werden zu ...

Hierarchische Listenfelder


Unter normalen Umständen zeigt das Listenfeld für den Jahrgang alle 13 Jahrgänge an. Das Listenfeld für die Klasse alle Buchstaben und das Listenfeld für die Schüler und Schülerinnen alle Schüler und Schülerinnen der Schule.

Wird mit hierarchischen Listenfeldern gearbeitet, so wird nach Auswahl des Jahrganges das Listenfeld für die Klasse eingegrenzt. Es werden nur noch die Klassenbezeichnungen angezeigt, die es in dem Jahrgang tatsächlich gibt. So könnte eben bei steigender Schüler- und Schülerinnenzahl die Anzahl der Klassen im Jahrgang ebenfalls steigen. Das letzte Listenfeld, die Namen, ist jetzt bereits stark eingegrenzt. Statt alle vermutlich deutlich über 1000 Schüler und Schülerinnen anzuzeigen listet das letzte Feld nur noch ca. 30 Schüler und Schülerinnen auf.

Zum Beginn steht nur die Auswahl des Jahrganges zur Verfügung. Ist ein Jahrgang ausgewählt, so steht die (bereits eingeschränkte) Auswahl der Klasse zur Verfügung. Erst zum Schluss wird schließlich das Listenfeld für die Namen freigegeben.

Wird das Listenfeld des Jahrganges geändert, so muss der Durchlauf wieder wie vorher starten. Wird nur das Listenfeld der Klasse geändert, so muss der Wert des Jahrganges für das letzte Listenfeld der Namen weiter gelten.

Um solch eine Funktion bereit zu stellen muss innerhalb eines Formulars eine Variable zwischen- gespeichert werden. Dies erfolgt in einem versteckten Kontrollfeld.

Der Makrostart wird an die Veränderung des Inhaltes eines Listenfeldes gekoppelt: Eigenschaft Listenfeld → Ereignisse → Modifiziert. In den Zusatzinformationen des Listenfeldes werden die notwendige Variablen gespeichert.

Hier der Beispielhafte Inhalt der Zusatzinformationen:
MainForm,Jahrgang,verstecktes Kontrollfeld,Listenfeld 2


SUB Hierarchisches_Kontrollfeld(oEvent AS OBJECT)
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oFeldHidden AS OBJECT
    DIM oFeld AS OBJECT
    DIM oFeld1 AS OBJECT
    DIM stSql AS STRING
    DIM aInhalt() = Split(stTag, ","
    stTag = oFeld.Tag
    oForm = oFeld.Parent
    REM Tag wird unter den Zusatzinformationen eingegeben
    REM Hier steht:
    REM 0. Feldname des zu filterndes Feldes in der Tabelle,
    REM 1. Feldname des versteckten Kontrollfeldes, das den Filterwert speichern soll,
    REM 2. eventuell weiteres Listfeld
    REM Der Tag wird von dem auslösenden Element ausgelesen. Die Variable wird an die
    REM Prozedur weitergegeben, die gegebenenfalls alle weiteren Listenfelder
    REM einstellt.
    aFilter() = Split(stTag, ","
    stFilter = ""

Nachdem die Variablen deklariert wurden, wird der Inhalt des Tags in ein Array übertragen. So kann auf die einzelnen Elemente zugegriffen werden. Anschließend wird der Zugang zu den verschiedenen Feldern im Formular deklariert.

```
IF Trim(aFilter(1)) = "" THEN
  IF oFeld.getCurrentValue <> "" THEN
    stFilter = "" + Trim(aFilter(0)) + "=" + oFeld.getCurrentValue() + ""
  END IF
END IF
```

Existiert bereits vorher ein Filter (weil es sich z.B. um das Listenfeld 2 handelt, das jetzt betätigt wurde), so wird der neue Inhalt an den vorherigen angehängt, der in dem versteckten Feld zwischengespeichert wurde.

```
IF oForm.Filter <> ""
  stFilter = oForm.Filter + " AND " + stFilter
```

Existiert bereits ein Filter und kommt das Feld, nach dem gefiltert werden soll, bereits im Filter vor, so muss die vorherige Filterung ab diesem Feldnamen gelöscht und die neue Filterung eingefügt werden.

```
ELSEIF oForm.Filter <> ""
  stFilter = Left(oForm.Filter, InStr(oForm.Filter, "" + Trim(aFilter(0)) + "=") - 1) + stFilter
END IF
```

Anschließend wird der Filter in das Formular eingetragen. Dieser Filter kann auch leer sein, wenn direkt das erste Listenfeld ohne Inhalt gewählt wurde.

```
oForm.Filter = stFilter
oForm.reload()
```

Die gleiche Prozedur wird durchlaufen, wenn nicht ein Formular direkt gefiltert werden soll. In dem Fall wird der Filterwert in einem versteckten Kontrollfeld zwischengespeichert.

```
ELSE
  oFeldHidden = oForm.getName(Trim(aFilter(1)))
  IF oFeld.getCurrentValue <> "" THEN
    stFilter = "" + Trim(aFilter(0)) + "=" + oFeld.getCurrentValue() + ""
  IF oFeldHidden.HiddenValue <> ""
    stFilter = oFeldHidden.HiddenValue + " AND " + stFilter
    ELSEIF oFeldHidden.HiddenValue <> ""
      stFilter = Left(oFeldHidden.HiddenValue, InStr(oFeldHidden.HiddenValue, "" + Trim(aFilter(0)) + "=") - 1) + stFilter
    END IF
  END IF
  END IF
  oFeldHidden.HiddenValue = stFilter
END IF
```

Ist in den Zusatzinformationen ein 4. Eintrag (Arraynummerierung beginnt bei 0!) vorhanden, so muss das folgende Listenfeld jetzt auf den entsprechenden Eintrag des aufrufenden Listenfeldes eingestellt werden.

```
IF UBound(aFilter()) > 1 THEN
  oFeld1 = oForm.getName(Trim(aFilter(2)))
  aFilter1() = Split(oFeld1.Tag, ",")
```

Die notwendigen Daten für die Filterung werden aus den Zusatzinformationen («Tag») des entsprechenden Listenfeldes ausgelesen. Leider ist es nicht möglich, lediglich den SQL-Code in dem
Listenfeld neu zu schreiben und anschließend das Listenfeld ein zu lesen. Vielmehr müssen die entsprechenden Werte direkt in das Listenfeld geschrieben werden.

Bei der Erstellung des Codes wird davon ausgegangen, dass die Tabelle, auf der das Formular beruht, die gleiche ist, auf der auch die Listenfelder beruhen. Für eine Weitergabe von Fremdschlüsseln an die Tabelle ist ein Listenfeld also erst einmal nicht gedacht.

```vba
IF ofeld.GetCurrentValue <> "" THEN
    stSql = "SELECT DISTINCT " + Trim(aFilter1(0)) + " FROM " + oForm.Command + 
    " WHERE " + stFilter + " ORDER BY " + Trim(aFilter1(0)) + ""
    oDatenquelle = ThisComponent.Parent.CurrentController
    IF NOT (oDatenquelle.isConnected()) THEN
        oDatenquelle.connect()
    END IF
    oVerbindung = oDatenquelle.ActiveConnection()
    oSQL_Anweisung = oVerbindung.createStatement()
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
    Die Werte werden in ein Array eingelesen. Das Array wird anschließend direkt in das Listenfeld übertragen. Die entsprechenden Zähler für das Array werden durch die Schleife kontinuierlich erhöht.

    inZaehler = 0
    WHILE oAbfrageergebnis.next
        ReDim Preserve aInhalt(inZaehler)
        aInhalt(inZaehler) = oAbfrageergebnis.getString(1)
        inZaehler = inZaehler + 1
    WEND
    ELSE
        aInhalt(0) = ""
    END IF
    oFeld1.StringItemList = aInhalt()


    oFeld1.refresh()
    WHILE UBound(aFilter1()) > 1
        DIM aLeer()
        oFeld2 = oForm.getByName(Trim(aFilter1(2)))
        DIM aFilter1()
        aFilter1() = Split(oFeld2.Tag, ",")
        oFeld2.StringItemList = aLeer()
        oFeld2.refresh()
    WEND
END IF
END SUB
```

Die sichtbaren Inhalte des Listenfeldes werden in oFeld1.StringItemList gespeichert. Soll zusätzlich auch ein Wert gespeichert werden, der als Fremdschlüssel an die darunterliegende Tabelle weitergegeben wird, wie das bei Listenfeldern in Formularen üblich, so ist dieser Wert in der Abfrage zusätzlich zu ermitteln und anschließend mit oFeld1.ValueItemList abzuspeichern. Für so eine Erweiterung sind allerdings zusätzliche Variablen notwendig wie z.B. neben der Tabelle, in der die Werte des Formulars gespeichert werden, noch die Tabelle, aus der die Listenfeldinhale gelesen werden.

Besondere Aufmerksamkeit ist dabei der Formulierung des Filters zu widmen.

```vba
stFilter = "" + Trim(aFilter1(1)) + "" + oFeld1.GetCurrentValue() + ""
```

funktioniert dann nur noch, wenn es sich bei der zugrundeliegenden LO-Version um eine Version ab LO 4.1 handelt, da hier als CurrentValue() der Wert wiedergegeben wird, der auch abgespeichert wird – nicht der Wert, der lediglich angezeigt wird. Damit das einwandfrei über verschiedene
Versionen hinweg funktionierte sollte unter **Eigenschaften: Listenfeld → Daten → Gebundenes Feld → '0'** angegeben sein.

### Zeiteingaben mit Millisekunden

Um Zeiten im Millisekunden-Bereich zu speichern ist in der Tabelle ein Timestamp-Feld erforderlich, das zudem per SQL separat darauf eingestellt wird (siehe «Erstellung von Tabellen»). Ein solches Feld kann vom Formular aus mit einem formatierten Feld beschrieben werden, das auch das Format **MM:SS,00** anbietet. Allerdings scheitert der erste Schreibversuch daran, dass der Eingabe der entsprechende Datumszusatz fehlt. Dies kann mit dem folgenden Makro erreicht werden, das an die Eigenschaft «Vor der Datensatzaktion» des Formulars gebunden wird:

```vba
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() = Split(stZeit, ",")
    DIM loNano AS LONG
    DIM inSecond AS INTEGER
    DIM inMinute AS INTEGER
    oDoc = thisComponent
    oDrawpage = oDoc.Drawpage
    oForm = oDrawpage.Forms.getByName("MainForm")
    oFeld = oForm.getByName("Zeit")
    stZeit = oFeld.Text


    IF stZeit <> "" THEN
        ar() = Split(stZeit, ",")
        loNano = CLng(ar(1)&"00000000")
        arMandS() = Split(ar(0), ":")
        inSecond = CInt(arMandS(1))
        inMinute = CInt(arMandS(0))
    END IF

    Die Einträge aus dem Feld «Zeit» werden in ihre Bestandteile zerlegt.

    Zuerst werden die Hundertstelsekunden abgetrennt und mit so vielen Nullen rechts aufgefüllt, dass sich insgesamt eine neunstellige Zahl ergibt. Eine so hohe Zahl kann nur in einer Long-Variablen gespeichert werden.

    Anschließend werden aus dem verbleibenden Rest durch eine Trennung am Trennzeichen «:» die Minuten von den Sekunden getrennt und in Integer-Zahlen umgewandelt.

    WITH unoStmp
        .NanoSeconds = loNano
        .Seconds = inSecond
        .Minutes = inMinute
        .Hours = 0
        .Day = 30
        .Month = 12
        .Year = 1899
    END WITH

```

*Makros: Bedienbarkeit verbessern*
Aktuelles Datum ermitteln und speichern:

```
DIM Jetzt AS DATE
Jetzt = Now()
WITH unoStmp
  .NanoSeconds = loNano
  .Seconds = inSecond
  .Minutes = inMinute
  .Hours = Hour(Jetzt)
  .Day = Day(Jetzt)
  .Month = Month(Jetzt)
  .Year = Year(Jetzt)
END WITH
```

Anschließend wird der erzeugte Zeitstempel über `updateTimestamp` in das Feld übertragen und mit dem Formular abgespeichert.

In älteren Anleitungen wird hier statt `NanoSeconds` der Begriff `HundredthSeconds` verwendet. Dieser entspricht aber nicht der API von LibreOffice und erzeugt deshalb nur Fehlermeldungen.

Ein Ereignis – mehrere Implementationen

Bei Formularen kommt es vor, dass ein Makro, mit einem Ereignis verknüpft, gleich zweimal aus-geführt wird. Dies liegt daran, dass mehrere Prozesse gleichzeitig z.B. mit dem Abspeichern eines geänderten Datensatzes verbunden sind. Die unterschiedlichen Ursachen für so ein Ereignis lassen sich folgendermaßen ermitteln:

```
SUB Ereignisursache_ermitteln(oEvent AS OBJECT)
  DIM oForm AS OBJECT
  oForm = oEvent.Source
  MsgBox oForm.ImplementationName
END SUB
```

Beim Abspeichern eines geänderten Datensatzes ergeben sich so zwei Implementationsnamen: `org.openoffice.comp.svx.FormController` und `com.sun.star.comp.forms.ODatabaseForm`. Über diese Namen kann jetzt gesteuert werden, dass ein Makro letztlich nur einmal den ganzen Code durchläuft. Die doppelte Durchführung ist oft nur eine (kleine) Bremse im Programmablauf. Sie kann aber auch dazu führen, dass sich z.B. ein Cursor nicht nur einen, sondern gleich zwei Datensätze zurück bewegt. Die Implementatio-
nen lassen auch nur bestimmte Befehle zu, so dass eine Kenntnis des Namens der Implementatio-
von Bedienhandlung von Bedeutung sein kann.

Abspeichern mit Nachfrage

Bei komplexen Änderungen eines Datensatzes kann es sinnvoll erscheinen, vor der Ausführung noch einmal nach zu fragen, ob die Änderungen denn durchgeführt werden sollen. Wird der Dialog mit Nein beantwortet, so wird der Speicherprozess abgebrochen, die Änderungen verworfen und der Cursor bleibt im aktuellen Datensatz stehen.

```
SUB SpeichernNachfrage(oEvent AS OBJECT)
  DIM oFormFeature AS OBJECT
  DIM oFormOperations AS OBJECT
  DIM inAntwort AS INTEGER
  oFormFeature = com.sun.star.form.runtime.FormFeature
  SELECT CASE oEvent.Source.ImplementationName
    CASE "org.openoffice.comp.svx.FormController"
      inAntwort = MsgBox("Soll der Datensatz geändert werden?", 4,"Datensatzänderung")
    SELECT CASE inAntwort
      CASE 6 ' Ja gedrückt, kein weiterer Eingriff nötig
END IF
END SUB
```

Außer Ja und Nein kann auch das Schließkreuz gedrückt werden. Dies gibt aber den gleichen Wert wie Nein zurück: 7'.

Wird lediglich mit den Tabulatortasten durch die Formulare navigiert, so kommt nur die Nachfrage der Messagebox. Bei der Navigation über die Navigationsleiste hingegen erscheint noch die Meldung, dass der Datensatz nicht verändert wurde.

**Primärschlüssel aus Nummerierung und Jahreszahl**


<table>
<thead>
<tr>
<th>Jahr</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>1</td>
</tr>
<tr>
<td>2014</td>
<td>2</td>
</tr>
<tr>
<td>2014</td>
<td>3</td>
</tr>
<tr>
<td>2015</td>
<td>1</td>
</tr>
<tr>
<td>2015</td>
<td>2</td>
</tr>
</tbody>
</table>

Damit lässt sich eine auf das Jahr bezogene Übersicht auch in den Dokumenten besser erzeugen.

```sub Datum_aktuell_ID_einfuegen```

<table>
<thead>
<tr>
<th>Datum aktuell ID einfuegen</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM oDatenquelle AS OBJECT</td>
</tr>
<tr>
<td>DIM oVerbindung AS OBJECT</td>
</tr>
<tr>
<td>DIM oSQL_Anweisung AS OBJECT</td>
</tr>
<tr>
<td>DIM stSql AS STRING</td>
</tr>
<tr>
<td>DIM oAbfrageergebnis AS OBJECT</td>
</tr>
<tr>
<td>DIM oDoc AS OBJECT</td>
</tr>
<tr>
<td>DIM oDrawpage AS OBJECT</td>
</tr>
<tr>
<td>DIM oForm AS OBJECT</td>
</tr>
<tr>
<td>DIM ofeld1 AS OBJECT</td>
</tr>
<tr>
<td>DIM ofeld2 AS OBJECT</td>
</tr>
<tr>
<td>DIM ofeld3 AS OBJECT</td>
</tr>
<tr>
<td>DIM inIDneu AS INTEGER</td>
</tr>
<tr>
<td>DIM inYear AS INTEGER</td>
</tr>
<tr>
<td>DIM unoDate</td>
</tr>
<tr>
<td>oDoc = thisComponent</td>
</tr>
<tr>
<td>oDrawpage = oDoc.drawpage</td>
</tr>
<tr>
<td>oForm = oDrawpage.forms.getName(&quot;MainForm&quot;)</td>
</tr>
</tbody>
</table>

22 Dem Handbuch liegt die Datenbank «Beispiel_Fortlaufende_Nummer_Jahr.odb» bei.
Alle Variablen werden deklariert. Die Formularfelder in dem Hauptformular werden angesteuert. Der Rest des Codes läuft nur ab, wenn der Eintrag für das Feld «fmtID» noch leer ist. Dann wird, wenn nicht schon ein Datum eingegeben wurde, zuerst ein Datumsstruct erstellt, um das aktuelle Datum und das aktuelle Jahr in die entsprechenden Felder übertragen zu können. Anschließend wird der Kontakt zu der Datenbank aufgebaut, sofern noch kein Kontakt existiert. Es wird zu dem höchsten Eintrag des Feldes "ID", bezogen auf das Jahr des Datumsfeldes, der Wert '1' addiert. Bleibt die Abfrage leer, so existiert noch kein Eintrag in dem Feld "ID". Jetzt könnte genauso gut '0' direkt in das Feld «fmtID» eingetragen werden. Die Nummerierung für die Aufträge sollte aber mit '1' beginnen, so dass der Variablen «inIDneu» eine '1' zugewiesen wird.

Die ermittelten Werte für das Jahr, die ID und, sofern nicht bereits ein Datum eingetragen wurde, das aktuelle Datum, werden schließlich in das Formular übertragen.

Im Formular sind die Felder für die Primärschlüssel "ID" und "Jahr" schreibgeschützt. Die Zuweisung kann so nur durch das Makro erfolgen.

Datenbankaufgaben mit Makros erweitert

Verbindung mit Datenbanken erzeugen

Hier wäre es möglich, fest einen Benutzernamen und ein Passwort einzugeben, wenn eine Passworteingabe erforderlich wäre. In den Klammer steht dann ("Benutzername","Passwort"). Statt
einen Benutzernamen und ein Passwort in Reinschrift einzutragen, wird für diesen Fall der Dialog für den Passwortschutz aufgerufen:

```
ELSE
  oAuthentifizierung = createUnoService("com.sun.star.sdb.InteractionHandler")
  oVerbindung = oDatenquelle.ConnectWithCompletion(oAuthentifizierung)
END IF
```

Wird allerdings von einem Formular innerhalb der Base-Datei auf die Datenbank zugegriffen, so reicht bereits

```
oDatenquelle = ThisComponent.Parent.CurrentController
IF NOT (oDatenquelle.isConnected()) Then
  oDatenquelle.connect()
End IF
oVerbindung = oDatenquelle.ActiveConnection()
```

Die Datenbank ist hier bekannt, ein Nutzernname und ein Passwort sind nicht erforderlich, da diese bereits in den Grundeinstellungen der HSQLDB für die interne Version ausgeschaltet sind.
Für Formulare außerhalb von Base wird die Verbindung über das erste Formular hergestellt:

```
oDatenquelle = Thiscomponent.Drawpage.Forms(0)
oVerbindung = oDatenquelle.activeConnection
```

### Daten von einer Datenbank in eine andere kopieren


Manchmal werden aber auch *.odb-Dateien so eingesetzt, dass ein möglich automatischer Daten austausch von einer Datenbankdatei zu einer anderen erfolgen soll. Die folgende Prozedur kann da hilfreich sein.  


```
SUB Datenkopie
  DIM oDatabaseContext AS OBJECT
  DIM oDatenquelle AS OBJECT
  DIM oDatenquelleZiel AS OBJECT
  DIM oVerbindung AS OBJECT
  DIM oVerbindungZiel AS OBJECT
  DIM oDB AS OBJECT
  DIM oSQL_Anweisung AS OBJECT
  DIM oSQL_AnweisungZiel AS OBJECT
  DIM oAbfrageergebnis AS OBJECT
  DIM oAbfrageergebnisZiel AS OBJECT
  DIM stSql AS String
  DIM stSqlZiel AS String
  DIM inID AS INTEGER
  DIM inIDZiel AS INTEGER
  DIM stName AS STRING
  DIM stort AS STRING
  oDB = ThisComponent.Parent
  stDir = Left(oDB.Location, Len(oDB.Location) - Len(oDB.Title))
  stDir = ConverToUrl(stDir & "ZielDB.odb")
  oDatenquelle = ThisComponent.Parent.CurrentController
```

23 Das Beispiel "Datenkopie_Quelle_Ziel" ist als gepacktes Verzeichnis diesem Handbuch beigefügt.

---

**Makros: Datenbankaufgaben mit Makros erweitert** 447
If NOT (oDatenquelle.isConnected()) THEN
    oDatenquelle.connect()
END IF
oVerbindung = oDatenquelle.ActiveConnection()
oDatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
oDatenquelleZiel = oDatabaseContext.getName(stDir)
oVerbindungZiel = oDatenquelleZiel.getConnection("","")
oSQL_Anweisung = oVerbindung.createStatement()
sql = "SELECT * FROM ","Tabelle"
OAbfrageergebnis = oSQL_Anweisung.executeQuery(sql)
WHILE oAbfrageergebnis.next
    inID = oAbfrageergebnis.getInt(1)
    stName = oAbfrageergebnis.getString(2)
    stOrt = oAbfrageergebnis.getString(3)
    oSQL_AnweisungZiel = oVerbindungZiel.createStatement()
    sqlZiel = "SELECT ","ID"", FROM ","Tabelle" WHERE ","ID"" = ","+inID+,"
    oAbfrageergebnisZiel = oSQL_AnweisungZiel.executeQuery(sqlZiel)
    inIDZiel = - -1
    WHILE oAbfrageergebnisZiel.next
        inIDZiel = oAbfrageergebnisZiel.getInt(1)
        IF inIDZiel = - -1 THEN
            sqlZiel = "INSERT INTO ","Tabelle" (","ID",","Name",","Ort") VALUES ","("+inID+,","+stName+,","+stOrt+")"
            oSQL_AnweisungZiel.executeUpdate(sqlZiel)
        END IF
    WEND
END IF
WEND
END SUB

Die komplette Tabelle der Ausgangsdatenbank wird ausgelesen und Zeile für Zeile anschließend über den Kontakt zur Zieldatenbank in die Tabelle der Zieldatenbank eingefügt. Vor dem Einfügen wird allerdings getestet, ob der Wert für den Primärschlüssel bereits vorhanden ist. Ist der Schlüs-
selwert vorhanden, so wird der Datensatz nicht kopiert.

Hier könnte gegebenenfalls auch eingestellt werden, dass statt einer Kopie des Datensatzes ein Update des bereits existierenden Datensatzes erfolgen soll. Auf jeden Fall wird so sicher gestellt, dass die Zieldatenbank die Datensätze mit den entsprechenden Primärschlüsseln der Quelldaten-
bank enthält.

Zugriff auf Abfragen

Abfragen lassen sich in der grafischen Benutzeroberfläche einfacher zusammenstellen als der gesamten Text in Makros zu übertragen, zumal dann auch noch innerhalb des Makros alle Felder-
und Tabellenbezeichnungen in zweifach doppelte Anführungszeichen gesetzt werden müssen.

SUB Abfrageninhalt
    DIM oDatenDatei AS OBJECT
    DIM oAbfragen AS OBJECT
    DIM stQuery AS STRING
    oDatenDatei = ThisComponent.Parent.CurrentController.DataSource
    oAbfragen = oDatenDatei.getQueryDefinitions()
    stQuery = oAbfragen.getByName("Query").Command
    msgbox stQuery
END SUB

Aus einem Formular heraus wird auf den Inhalt der *.odb-Datei zugegriffen. Die Abfragen werden über getQueryDefinitions() ermittelt. Die SQL-Formulierung der Abfrage "Query" wird über den Zusatz Command bereit gestellt. Command kann schließlich dazu genutzt werden, eine entspre-
chende Abfrage auch innerhalb eines Makros weiter zu nutzen.

Allerdings muss bei der Nutzung des SQLCodes der Abfrage darauf geachtet werden, dass sich der Code nicht wiederum auf eine Abfrage bezieht. Das führt dann unweigerlich zu der Meldung, dass die (angebliche) Tabelle der Datenbank unbekannt ist. Einfacher ist es daher, aus Abfragen Ansichten zu erstellen und entsprechend auf die Ansichten in Makros zuzugreifen.

Makros: Datenbankaufgaben mit Makros erweitert 448
**Datenbanksicherungen erstellen**

Vor allem beim Erstellen von Datenbanken kann es hin und wieder vorkommen, dass die *.odb-Datei unvermittelt beendet wird. Vor allem beim Berichtsmodul ist ein häufigeres Abspeichern nach dem Editieren sinnvoll.

Ist die Datenbank erst einmal im Betrieb, so kann sie durch Betriebssystemabstürze beschädigt werden, wenn der Absturz gerade während der Beendigung der Base-Datei erfolgt. Schließlich wird in diesem Moment der Inhalt der Datenbank in die Datei zurückgeschrieben.


```vba
SUB Datenbankbackup(inMax AS INTEGER)
    DIM oPath AS OBJECT
    DIM oDoc AS OBJECT
    DIM sTitel AS STRING
    DIM sUrl_Ziel AS STRING
    DIM sUrl_Start AS STRING
    DIM i AS INTEGER
    DIM k AS INTEGER
    oDoc = ThisComponent
    sTitel = oDoc.Title
    sUrl_Start = oDoc.URL
    DO WHILE sUrl_Start = ""
        oDoc = oDoc.Parent
        sTitel = oDoc.Title
        sUrl_Start = oDoc.URL
    LOOP

    oPath = createUnoService("com.sun.star.util.PathSettings")
    FOR i = 1 TO inMax + 1
        IF NOT FileExists(oPath.Backup & "/" & i & "_" & sTitel) THEN
            IF i > inMax THEN
                FOR k = inMax - 1 TO 1 STEP -1
                    IF FileDateTime(oPath.Backup & "/" & k & "_" & sTitel) <= FileDateTime(oPath.Backup & "/" & i & "_" & sTitel) THEN
                        i = k
                        EXIT FOR
                    END IF
                NEXT
            END IF
        ELSE
            i = k + 1
            EXIT FOR
        END IF
    NEXT
    sUrl_Ziel = oPath.Backup & "/" & i & "_" & sTitel
    FileCopy(sUrl_Start,sUrl_Ziel)
END SUB
```

Wird das Makro direkt beim Start der *.odb-Datei ausgeführt, so stimmen sTitel und sUrl_Start. Wird es hingegen von einem Formular aus ausgeführt, so muss erst einmal ermittelt werden, ob überhaupt eine URL verfügbar ist. Ist die URL leer, so wird eine Ebene höher (oDoc.Parent) nach einem Wert nachgesehen.
Werden vor der Ausführung der Prozedur «Datenbankbackup» während der Nutzung von Base die Daten aus dem Cache in die Datei zurückgeschrieben, so kann ein entsprechendes Backup auch z.B. nach einer bestimmten Nutzerzeit oder durch Betätigung eines Buttons sinnvoll sein. Das Zurückschreiben regelt die folgende Prozedur:

```vba
SUB Daten_aus_Cache_schreiben
    DIM oDaten AS OBJECT
    DIM oDataSource AS OBJECT
    oDaten = ThisDatabaseDocument.CurrentController
    IF NOT (oDaten.isConnected) THEN oDaten.connect()
oDataSource = oDaten.DataSource
    oDataSource.flush
END SUB
```

Soll alles zusammen aus einem Formular heraus über einen Button gestartet werden, so müssen beide Prozeduren über eine weitere Prozedur angesprochen werden:

```vba
SUB Backup_sofort
    Daten_aus_Cache_schreiben
    Datenbankbackup(10)
END SUB
```


Der Befehl soll anschließend in der Base-Datei gespeichert werden, in diesem Fall also in der Datei «Medien_mit_Makros.odb».
Über **Hinzufügen** erscheint das Fenster für das Hinzufügen von Befehlen. Ganz unten in dem Fenster ist auch die Möglichkeit gegeben, Makros aus der Datenbankdatei hinzu zu fügen. Hier wurde die Prozedur «Backup_Sofort» ausgesucht.

Der Befehl ist jetzt in der Symbolleiste an erster Stelle verfügbar. Um jetzt die Prozedur auszuführen genügt ein Auslösen des Buttons in der Symbolleiste.

Jetzt bietet es sich noch an, dem Befehl ein Symbol zuzuweisen. Über **Ändern → Symbol austauschen** wird der folgende Dialog geöffnet.
Hier wird jetzt ein passendes Symbol gesucht. Es kann auch ein eigenes Symbol erstellt und eingebunden werden.

Das Symbol erscheint anschließend statt der Benennung der Prozedur. Die Benennung wird als Tooltip angezeigt.

**Datenbanken komprimieren**

Eigentlich nur ein SQL-Befehl (*SHUTDOWN COMPACT*), der hin- und wieder, vor allem nach der Löschung von vielen Daten, durchgeführt werden sollte. Die Datenbank speichert neue Daten ab, hält aber gleichzeitig den Platz für die eventuell gelöschten Daten vor. Bei starker Änderung des Datenbestandes muss deshalb der Datenbestand wieder komprimiert werden.

**Hinweis**

Für die interne HSQLDB wird dieser Befehl seit der LibreOffice-Version 3.6.* automatisch beim Schließen der Datenbank durchgeführt. Diese Makro ist dort also nicht mehr nötig.

Wird die Komprimierung durchgeführt, so kann anschließend nicht mehr auf die Tabellen zugegriffen werden. Die Datei muss neu geöffnet werden. Deshalb schließt das Makro auch das Formular, aus dem heraus es aufgerufen wird. Leider lässt sich das Dokument selbst nicht schließen, ohne dass beim Neuauf Ruf die Wiederherstellung anspringt. Deshalb ist diese Funktion auskommentiert.

```vba
SUB Datenbank_komprimieren
DIM stMessage AS STRING
oDatenquelle = ThisComponent.Parent.CurrentController ' Zugriffsmöglichkeit aus dem Formular heraus
IF NOT (oDatenquelle.isConnected()) THEN
    oDatenquelle.connect()
END IF
oVerbindung = oDatenquelle.ActiveConnection()
oSQL_Anweisung = oVerbindung.createStatement()
```

*Makros: Datenbankaufgaben mit Makros erweitert* 452
Die Datenbank wird komprimiert und geschlossen.

```
stSql = "SHUTDOWN COMPACT"
Die Datenbank wird komprimiert und geschlossen.
```

```
 strSQL_Anweisung.executeQuery(stSql)
```

```
stMessage = "Die Datenbank wurde komprimiert." + CHR(13) + "Das Formular wird jetzt geschlossen."
```

```
stMessage = stMessage + CHR(13) + "Anschließend sollte die Datenbankdatei geschlossen werden."
```

```
stMessage = stMessage + CHR(13) + "Auf die Datenbank kann erst nach dem erneuten Öffnen der Datenbankdatei zugegriffen werden."
```

```
msgbox stMessage
```

```
ThisDatabaseDocument.FormDocuments.getByName("Wartung").close
```

REM Das Schließen der Datenbankdatei führt beim Neustart zu einem Wiederherstellungsablauf.

```
REM Das Schließen der Datenbankdatei führt beim Neustart zu einem Wiederherstellungsablauf.
```

```
ThisDatabaseDocument.close(True)
```

```
END SUB
```

**Tabellenindex heruntersetzen bei Autowert-Feldern**

Werden viele Daten aus Tabellen gelöscht, so stören sich Nutzer häufig daran, dass die automatisch erstellten Primärschlüssel einfach weiter hochgezählt werden, statt direkt an den bisher höchsten Schlüsselwert anzuschließen. Die folgende Prozedur liest für eine Tabelle den bisherigen Höchstwert des Feldes "ID" aus und stellt den nächsten Schlüsselstartwert um 1 höher als das Maximum ein.

Heißt das Primärschlüsselfeld nicht "ID", so müsste das Makro entsprechend angepasst werden.

```
SUB Tabellenindex_runter(stTabelle AS STRING)
REM Mit dieser Prozedur wird das automatisch hochgeschriebene Primärschlüsselfeld mit der vorgegebenen Bezeichnung "ID" auf den niedrigst möglichen Wert eingestellt.
DIM inAnzahl AS INTEGER
DIM inSequence_Value AS INTEGER
oDatenquelle = ThisComponent.Parent.CurrentController' Zugriffsmöglichkeit aus dem Formular heraus
IF NOT (oDatenquelle.isConnected()) THEN
  oDatenquelle.connect()
END IF
oVerbindung = oDatenquelle.ActiveConnection()
oSQL_Anweisung = oVerbindung.createStatement()
stSql = "SELECT MAX(""ID""') FROM ""+stTabelle+""'
Der höchste in "ID" eingetragene Wert wird ermittelt
```
```
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
Abfrage starten und den Rückgabewert in einer Variablen oAbfrageergebnis speichern
WHILE oAbfrageergebnis.next
  inAnzahl = oAbfrageergebnis.getInt(1)
  ' Erstes Datenfeld wird ausgelesen
WEND ' nächster Datensatz, in diesem Fall nicht mehr erforderlich, da nur ein Datensatz existiert
IF inAnzahl = "" THEN ' Falls der höchste Wert gar kein Wert ist, also die Tabelle leer ist, wird sie als höchstes Wert als -1 angenommen
  inAnzahl = -1
END IF
inSequence_Value = inAnzahl+1 ' Der höchste Wert wird um 1 erhöht
REM Ein neuer Befehl an die Datenbank wird vorbereitet. Die ID wird als neu startend ab inAnzahl+1 deklariert.
REM Diese Anweisung hat keinen Rückgabewert, da ja kein Datensatz ausgelesen werden muss
```
```
oSQL_Anweisung1 = oVerbindung.createStatement()
oSQL_Anweisung1.executeUpdate("ALTER TABLE "" + stTabelle + ""
ALTER COLUMN ""ID"" RESTART WITH " + inSequence_Value + "")
```
```
END SUB
```

**Drucken aus Base heraus**

Der Standard, um aus Base heraus ein druckbares Dokument zu erzeugen, ist die Nutzung eines Berichtes. Daneben gibt es noch Möglichkeiten, Tabellen und Abfragen einfach nach Calc zu
kopieren und dort zum Druck aufzubereiten. Auch der direkte Druck eines Formularinhaltes vom Bildschirm ist natürlich möglich.

**Druck von Berichten aus einem internen Formular heraus**

Um einen Bericht zu starten muss normalerweise die Benutzeroberfläche von Base aufgesucht werden. Ein Mausklick auf den Berichtsnamen startet dann die Ausführung des Berichtes. Einfacher geht es natürlich, den Bericht direkt aus dem Formular heraus zu starten:

```
SUB Berichtsstart
   ThisDatabaseDocument.ReportDocuments.getByNamed("Bericht").open
END SUB
```

Sämtliche Berichte werden über `ReportDocuments` mit ihrem Namen angesprochen. Mit `open` werden sie geöffnet. Wird ein Bericht jetzt an eine Abfrage gebunden, die über das Formular gefiltert wird, so kann auf diese Art und Weise der zum aktuellen Datensatz anfallende Ausdruck erfolgen.

**Start, Formatierung, direkter Druck und Schließen des Berichts**


```
SUB BerichtStart(oEvent AS OBJECT)
   DIM oForm AS OBJECT
   DIM strSql AS STRING
   DIM oDatenquelle AS OBJECT
   DIM oVerbindung AS OBJECT
   DIM oSQL_Anweisung AS OBJECT
   DIM oReport AS OBJECT
   DIM oReportView AS OBJECT
   oForm = oEvent.Source.model.parent
   strSql = "UPDATE "Filter" SET ""Integer"" = " +
     oForm.getInt(oForm.findColumn("ID")) + ": WHERE ""ID"" = TRUE"
   oDatenquelle = ThisComponent.Parent.CurrentController
   IF NOT (oDatenquelle.isConnected()) THEN
     oDatenquelle.connect()
   END IF
   oVerbindung = oDatenquelle.ActiveConnection()
   oSQL_Anweisung = oVerbindung.createStatement()
   oSQL_Anweisung.executeUpdate(strSql)
   oReport = ThisDatabaseDocument.ReportDocuments.getByNamed("Berichtsname").open
   oReportView.Visible = False
   BerichtZeilenhoeheAuto(oReport)
END SUB
```

Ohne Bezug auf das Formular könnte auch nur der Bericht aufgerufen werden. Dabei ist der aufgerufene Bericht gleich als Objekt ansprechbar (oReport). Anschließend wird das Fenster auf unsichtbar eingestellt. Dies geht leider nicht direkt mit dem Aufruf, so dass ganz kurz das Fenster erscheint, dann aber ggf. in Ruhe im Hintergrund mit dem entsprechenden Inhalt gefüllt wird.

Anschließend wird die Prozedur «BerichtZeilenhoeheAuto» gestartet. Dieser Prozedur wird der Hinweis auf den geöffneten Bericht mitgegeben.

Die Zeilenhöhe kann zur Zeit nicht beim Berichtsentwurf automatisch angepasst werden. Ist zu viel Text für ein Feld vorgesehen, so wird der Text abgeschnitten und darauf mit Hilfe eines roten Dreiecks hingewiesen. Solange dies nicht funktioniert stellt die folgende Prozedur sicher, dass z.B. in allen Tabellen mit der Bezeichnung «Detail» die automatische Höhe eingeschaltet wird.

```
SUB BerichtZeilenhoeheAuto(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
        ENDIF
    NEXT inT
    BerichtDruckenUndSchliessen(oReport)
END SUB
```

Bei dem Entwurf des Berichtes muss darauf geachtet werden, dass alle in einer Zeile des Bereiches «Detail» befindlichen Felder tatsächlich die gleiche Höhe haben. Sonst kann es zusammen mit der Automatik passieren, dass ein Feld plötzlich auf die doppelte Zeilenhöhe gesetzt wird.

Nachdem in allen Tabellen mit der Bezeichnung «Detail» die automatische Höhe eingestellt wurde, wird anschließend der Bericht über die Prozedur «BerichtDruckenUndSchliessen» weiter an den Drucker geschickt.

Das Array Props enthält die verschiedenen Werte, die mit dem Drucker bei einem Dokument verbunden sind. Für den Druckbefehl ist hier lediglich der Name des Standarddruckers wichtig. Das Berichtsdokument soll so lange geöffnet bleiben, bis der Druck tatsächlich abgeschlossen ist. Dies geschieht, indem dem Druckbefehl der Name und der Befehl "Warte, bis ich fertig bin" (Wait) mitgegeben wird.

```
Sub BerichtDruckenUndSchliessen(oReport AS OBJECT)
    DIM Props
    DIM stDrucker AS STRING
    Props = oReport.getPrinter()
    stDrucker = Props(0).value
    DIM arg(1) AS NEW com.sun.star.beans.PropertyValue
    arg(0).name = "Name"
    arg(0).value = "<" & stDrucker & ">"
    arg(1).name = "Wait"
    arg(1).value = True
    oReport.print(arg())
    oReport.close(true)
End Sub
```

Erst wenn der Druck komplett an den Drucker abgeschickt wurde, wird das Dokument geschlossen.

Zu Einstellungen des Druckers siehe die Drucker und Druckeinstellungen aus dem AOO-Wiki.

**Makros: Datenbankaufgaben mit Makros erweitert** 455
Soll statt des Drucks oder zusätzlich zu dem Druck auch eine *.pdf-Datei des Dokumentes als Sicherungskopie abgelegt werden, so wird darauf mit der Methode `storeToURL()` zugegriffen:

```vba
Sub BerichtAlsPDFspeichern(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
```


**Druck von Berichten aus einem externen Formular heraus**


```vba
SUB Berichtsstart(oEvent AS OBJECT)
  DIM oFeld AS OBJECT
  DIM oForm AS OBJECT
  DIM oDocument AS OBJECT
  DIM oDocView AS OBJECT
  DIM Arg() = oEvent.Source.Model
  oFeld = oEvent.Source.Model
  oForm = oFeld.Parent
  sURL = oForm.DataSourceName
  oDocument = StarDesktop.loadComponentFromURL(sURL, "_blank", 0, Arg() )
  oDocView.Visible = False
  oDocument.getCurrentController().connect
  Wait(100)
  oDocument.ReportDocuments.getByName("Bericht").open
  oDocument.close(True)
END SUB
```


Mit einem entsprechenden Zugriff können auch die Formulare innerhalb des *.odb-Datei gestartet werden. Hier sollte dann aber das Schließen des Dokumentes unterbleiben.

Deutlich schneller als mit dem Report-Builder und trotzdem gut gestaltet geht der Druck aber über Makros mit Hilfe von Serienbrieffunktionen oder Textfeldern.
**Serienbriefdruck aus Base heraus**


```vbnet
SUB Serienbriefdruck
  DIM oMailMerge AS OBJECT
  DIM aProps()
  oMailMerge = createunoservice("com.sun.star.text.MailMerge")

  Als Name der Datenquelle wird der Name angegeben, unter dem die Datenbank in LO angemeldet ist. Dieser Name muss nicht identisch mit dem Dateinamen sein. Der Anmeldename in diesem Beispiel lautet "Adressen"
  oMailMerge.DataSourceName = "Adressen"

  Die Pfadbeschreibung mit der Serienbriefdatei erfolgt in der Art der jeweiligen Betriebssystemumgebung, hier ab dem Wurzelpfad eines Linux-Systems.
  oMailMerge.DocumentURL = ConvertToUrl("home/user/Dokumente/Serienbrief.odt")

  Der Typ des Kommandos wird festgelegt. '0' steht für eine Tabelle, '1' für eine Abfrage und '2' für ein direktes SQL-Kommando.
  oMailMerge.CommandType = 1

  Hier wurde eine Abfrage gewählt, die den Namen "Serienbriefabfrage" trägt.
  oMailMerge.Command = "Serienbriefabfrage"

  Über den Filter wird festgelegt, für welche Datensätze aus der Serienbriefabfrage ein Druck erfolgen soll. Dieser Filter könnte z.B. über ein Formularfeld aus Base heraus an das Makro weitergegeben werden. Mit dem Primärschlüssel eines Datensatzes könnte so der Ausdruck eines einzelnen Dokumentes erfolgen.

  In diesem Beispiel wird aus der "Serienbriefabfrage" das Feld "Geschlecht" aufgesucht und dort nach Datensätzen gesucht, die in diesem Feld mit einem 'm' versehen sind.
  oMailMerge.Filter = """Geschlecht""='m'

  Es gibt die Ausgabetypen Drucker (1), Datei (2) und Mail (3). Hier wurde zu Testzwecken die Ausgabe in eine Datei gewählt. Diese Datei wird in dem angegebenen Pfad abgespeichert. Für jeden Serienbriefdatensatz wird ein Druck erzeugt. Damit dieser Druck unterscheidbar ist, wird das Feld Nachname in den Dateinamen aufgenommen.
  oMailMerge.OutputType = 2
  oMailMerge.OutputUrl = ConvertToUrl("home/user/Dokumente")
  oMailMerge.FileNameFromColumn = True
  oMailMerge.Filenameprefix = "Nachname"
  oMailMerge.execute(aProps())
END SUB
```

Wird allein der Filter über ein Formular bestückt, so kann auf diese Art und Weise also ohne die Öffnung des Writer-Dokumentes ein Serienbriefdruck erfolgen.

**Drucken über Textfelder**

Über **Einfügen → Feldbefehl → Funktionen → Platzhalter** wird im Writer eine Vorlage für das zukünftig zu druckende Dokument erstellt. Die Platzhalter sollten dabei sinnvollerweise mit dem Namen versehen werden, den die Felder auch in der Datenbank bzw. der Tabelle/Abfrage für das Formular haben, aus dem heraus das Makro aufgerufen wird.

Für einfache Zwecke wird «Text» als Typ für den Platzhalter gewählt.


Hier Teile des Codes, der im Wesentlichen diesem Beitrag von DPunch zu verdanken ist: http://de.openoffice.info/viewtopic.php?f=8&t=45868#p194799

```vba
SUB Textfelder_Fuellen
    oForm = thisComponent.Drawpage.Forms.MainForm
    IF oForm.RowCount = 0 THEN
        MsgBox "Kein Datensatz zum Drucken vorhanden"
        EXIT SUB
    END IF

    Das Hauptformular wird angesteuert. Hier könnte auch die Lage des auslösenden Buttons das Formular selbst ermitteln. Anschließend wird geklärt, ob in dem Formular überhaupt Daten liegen, die einen Druck ermöglichen.

    oColumns = oForm.Columns
    oDB = thisComponent.Parent
    Der Zugriff auf die URL ist nicht vom Formular aus direkt möglich. Es muss auf den darüber liegenden Frame der Datenbank Bezug genommen werden.

    stDir = Left(oDB.Location, Len(oDB.Location) - Len(oDB.Title))
    Der Titel der Datenbank wird von der URL abgetrennt.

    stDir = stDir & "Beispiel_Textfelder.ott"
    Die Vorlage wird aufgesucht und geöffnet

    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)

    Die Textfelder werden eingelesen

    oTextfields = oNewDoc.Textfields.createEnumeration
    DO WHILE oTextfields.hasMoreElements
        oTextfield = oTextfields.nextElement
        IF oTextfield.supportsService("com.sun.star.text.TextField.JumpEdit") THEN
            stColumnname = oTextfield.Placeholder
            Placeholder ist die Benennung für das Textfeld

            IF oColumns.hasByName(stColumnname) THEN
                inIndex = oForm.findColumn(stColumnname)
                oTextfield.Anchor.String = oForm.getString(inIndex)
                END IF
        END IF
    LOOP
END SUB
```

**Aufruf von Anwendungen zum Öffnen von Dateien**

Mit dieser Prozedur kann durch einen Mausklick in ein Textfeld ein Programm aufgerufen werden, das im eigenen Betriebssystem mit der Dateinamensendung verbunden ist. Damit werden auch
Links ins Internet oder sogar das Öffnen des Mailprogramms mit einer bestimmten Mailadresse möglich, die gerade in der Datenbank gespeichert wurde.

Siehe zu diesem Abschnitt auch die Beispieldatenbank «Beispiel_Mailstart_Dateiaufruf.odb»

```vba
SUB Link_oeffnen
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oFeld AS OBJECT
    DIM oShell AS OBJECT
    DIM stFeld AS STRING
    oDoc = thisComponent
    oDrawpage = oDoc.Drawpage
    oForm = oDrawpage.Forms.getByName("Formular")
    oFeld = oForm.getByName("Adresse")

    stFeld = oFeld.Text
    IF stFeld = "" THEN
        EXIT SUB
    END IF

    Ist das Feld leer, so soll das Makro sofort enden. Bei der Eingabe passiert es ja sehr oft, dass Felder mit der Maus aufgesucht werden. Ein Mausklick in das Feld, um dort zu ersten Mal schreiben zu können, soll aber noch nicht das Makro ausführen.

    Jetzt wird gesucht, ob in dem Feld ein '@' enthalten ist. Dies deutet auf eine E-Mail-Adresse hin. Es soll also das Mailprogramm gestartet werden und eine Mail an diese Mailadresse gesandt werden.

    IF InStr(stFeld,"@") THEN
        stFeld = "mailto:"+stFeld
    ELSE
        stFeld = convertToUrl(stFeld)
    END IF


    oShell = createUnoService("com.sun.star.system.SystemShellExecute")
    oShell.execute(stFeld,,0)
END SUB
```

**Aufruf eines Mailprogramms mit Inhaltsvorgaben**

Eine Erweiterung des vorhergehenden Beispiels zum Programmaufruf stellt dieser Aufruf eines Mailprogramms mit Vorgaben in der Betreffzeile und inhaltlichen Vorgaben dar.

Siehe auch zu diesem Abschnitt die Beispieldatenbank «Mailstart_Dateiaufruf.odb»

**Hinweis**


```vbnet
DIM attaches(0)
oMailer = createUnoService("com.sun.star.system.SimpleSystemMail")
oMailProgramm = oMailer.querySimpleMailClient()
oNeueNachricht = oMailProgramm.createSimpleMailMessage()
oNeueNachricht.setRecipient(stFeld1)
oNeueNachricht.setSubject(stFeld2)
attachs(0) = "file:///...
attaches.setAttachment(attachs)
oMailprogramm.sendSimpleMailMessage(oNeuenachricht)
```

Zu den möglichen Parametern siehe: [http://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1system_1_1XSimpleMailMessage.html](http://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1system_1_1XSimpleMailMessage.html)
Mauszeiger beim Überfahren eines Links ändern

Im Internet üblich, bei Base nachgebaut: Der Mauszeiger fährt über ein Textfeld und verändert seine Form zu einer zeigenden Hand. Der enthaltene Text kann jetzt noch in den Eigenschaften des Feldes zu der Farbe Blau und unterstrichen geändert werden – schon ist der Eindruck eines Links aus dem Internet perfekt. Jeder Nutzer erwartet nach einem Klick, dass sich ein externes Programm öffnet.

Siehe auch zu diesem Abschnitt die Beispieldatenbank «Mailstart_Dateiaufruf.odb»

Diese kurze Prozedur sollte mit dem Ereignis 'Maus innerhalb' des Textfeldes verbunden werden.

```
SUB Mauszeiger(Event AS OBJECT)
    REM Siehe auch Standardbibliotheken: Tools → ModuleControls → SwitchMousePointer
    DIM oPointer AS OBJECT
    oPointer = createUnoService("com.sun.star.awt.Pointer")
    oPointer.settype(27)'Typen in com.sun.star.awt.SystemPointer
    Event.Source.Peer.SetPointer(oPointer)
END SUB
```

Formulare ohne Symbolleisten präsentieren

Neunutzer von Base sind häufig irritiert, dass z.B. eine Menüleiste existiert, diese aber im Formular so gar nicht verfügbar ist. Diese Menüleisten können auf verschiedene Arten ausgeblendet werden. Am erfolgreichsten unter allen LO-Versionen sind die beiden im Folgenden vorgestellten Vorgehensweisen.

Fenstergrößen und Symbolleisten werden in der Regel über ein Makro beeinflusst, das in einem Formulardokument unter Extras → Anpassen → Ereignisse → Dokument öffnen gestartet wird. Gemeint ist hier das Dokument, nicht ein einzelnes Haupt- oder Unterformular.

**Formulare ohne Symbolleisten in einem Fenster**


```
SUB Symbolleisten_Ausblenden
    DIM oFrame AS OBJECT
    DIM oWin AS OBJECT
    DIM oLayoutMng AS OBJECT
    DIM aElemente
    oFrame = StarDesktop.getCurrentFrame()
    oFrame.setTitle "Mein Formular"
    oWin = oFrame.getContainerWindow()
    oWin.IsMaximized = true

    oWin.setPosSize(0,0,600,400,15)

    x=1, y=2, Breite=4, Höhe=8. Da x, y, Breite und Höhe angegeben wurden, hat das 'Flag' die Größe 1 + 2 + 4 + 8 = 15.
```

Makros: Datenbankaufgaben mit Makros erweitert 461
oLayoutMng = oFrameLayoutManager
aElemente = oLayoutMng.getElements()
FOR i = LBound(aElemente) TO UBound(aElemente)
    IF aElemente(i).ResourceURL = "private:resource/toolbar/formsnavigationbar" THEN
ELSE
    oLayoutMng.hideElement(aElemente(i).ResourceURL)
END IF
NEXT
END SUB

Wenn es sich um die Navigationsleiste handelt soll nichts geschehen. Das Formular soll schließ¬
lich bedienbar bleiben, wenn nicht das Kontrollfeld für die Navigationsleiste eingebaut und die
Navigationsleiste sowieso ausgeblendet wurde. Nur wenn es sich nicht um die Navigationsleiste
handelt soll die entsprechende Leiste verborgen werden. Deswegen erfolgt zu dieser Bedingung
keine Aktion.

Werden die Symbolleisten nicht wieder direkt beim Beenden des Formulars eingeblendet, so blei¬
ben sie weiterhin verborgen. Sie können natürlich über Ansicht → Symbolleisten wieder aufgeru¬
fen werden. Etwas irritierend ist es jedoch, wenn gerade die Standardleiste (Ansicht → Sym¬
bolleisten → Standardleiste) oder die Statusleiste (Ansicht → Statusleiste) fehlt.

Mit dieser Prozedur werden die Symbolleisten aus dem Versteck ("hideElement") wieder hervor¬geholt ("showElement"). Der Kommentar enthält die Leisten, die oben als sonst fehlende Leisten am ehesten auffallen.

SUB Symbolleisten_Einblenden
    DIM oFrame AS OBJECT
    DIM oLayoutMng AS OBJECT
    DIM i AS INTEGER
    DIM aElemente(5) AS STRING
    oFrame = StarDesktop.getCurrentFrame()
    oLayoutMng = oFrameLayoutManager
    aElemente(0) = "private:resource/menubar/menubar"
    aElemente(1) = "private:resource/statusbar/statusbar"
    aElemente(2) = "private:resource/toolbar/formsnavigationbar"
    aElemente(3) = "private:resource/toolbar/standardbar"
    aElemente(4) = "private:resource/toolbar/formsdesign"
    aElemente(5) = "private:resource/toolbar/formcontrols"
    FOR EACH i IN aElemente()
        IF NOT(oLayoutMng.requestElement(i)) THEN
            oLayoutMng.createElement(i)
        END IF
    NEXT
    oLayoutMng.showElement(i)
END SUB

Die Makros werden an die Eigenschaften des Formularfensters gebunden: Extras → Anpassen
→ Ereignisse → Dokument öffnen → Symbolleisten_Ausblen¬den bzw. ... Dokument
geschlossen → Symbolleisten_Einblenden

Leider tauchen häufig Symbolleisten trotzdem nicht wieder auf. In hartnäckigen Fällen kann es
daher helfen, nicht die Elemente auszulesen, die der Layoutmanager bereits kennt, sondern defini¬
tiv bestimmte Symbolleisten erst zu erstellen und danach schließlich zu zeigen:

SUB Symbolleisten_Einblenden
    DIM oFrame AS OBJECT
    DIM oLayoutMng AS OBJECT
    DIM i AS INTEGER
    DIM aElemente(5) AS STRING
    oFrame = StarDesktop.getCurrentFrame()
    oLayoutMng = oFrameLayoutManager
    aElemente(0) = "private:resource/menubar/menubar"
    aElemente(1) = "private:resource/statusbar/statusbar"
    aElemente(2) = "private:resource/toolbar/formsnavigationbar"
    aElemente(3) = "private:resource/toolbar/standardbar"
    aElemente(4) = "private:resource/toolbar/formsdesign"
    aElemente(5) = "private:resource/toolbar/formcontrols"
    FOR EACH i IN aElemente()
        IF NOT(oLayoutMng.requestElement(i)) THEN
            oLayoutMng.createElement(i)
        END IF
    NEXT
    oLayoutMng.showElement(i)
END SUB

Makros: Datenbankaufgaben mit Makros erweitert 462
Die darzustellenden Symbolleisten werden explizit benannt. Ist eine der entsprechenden Symbolleisten nicht für den Layoutmanager vorhanden, so wird sie zuerst über `createElement` erstellt und danach über `showElement` gezeigt.

**Formulare im Vollbildmodus**

Beim Vollbildmodus wird der gesamte Bildschirm vom Formular bedeckt. Hier steht keine Kontrollleiste o.ä. mehr zur Verfügung, die gegebenenfalls anzeigt, ob noch irgendwelche anderen Programme laufen.

```vbnet
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
```

Diese Funktion wird durch die folgenden Prozeduren eingeschaltet. In den Prozeduren läuft gleichzeitig die vorhergehende Prozedur zum Ausblenden der Symbolleisten ab – sonst erscheint die Symbolleiste, mit der der Vollbildmodus wieder ausgeschaltet werden kann. Auch dies ist eine Symbolleiste, wenn auch nur mit einem Symbol.

```vbnet
SUB Vollbild_ein
    Fullscreen(true)
    Symbolleisten_Ausblenden
END SUB
```

Aus dem Vollbild-Modus geht es wieder heraus über die 'ESC'-Taste. Wenn stattdessen ein Button mit einem entsprechenden Befehl belegt werden soll, so reichen auch die folgenden Zeilen:

```vbnet
SUB Vollbild_aus
    Fullscreen(false)
    Symbolleisten_Einblenden
END SUB
```

**Formular direkt beim Öffnen der Datenbankdatei starten**

Wenn jetzt schon die Symbolleisten weg sind oder gar das Formular im Vollbildmodus erscheint, dann müsste nur noch die Datenbankdatei beim Öffnen direkt in dieses Formular hinein starten. Der einfache Befehl zum Öffnen von Formularen reicht dabei leider nicht aus, da die Datenbankverbindung beim Öffnen des Base-Dokumentes noch nicht besteht.

Das folgende Makro wird über Extras → Anpassen → Ereignisse → Dokument öffnen gestartet. Dabei ist Speichern in → Datenbankdatei.odb zu wählen.

```vbnet
SUB Formular_Direktstart
    DIM oDatenquelle AS OBJECT
    oDatenquelle = ThisDatabaseDocument.CurrentController
    IF NOT (oDatenquelle.isConnected()) THEN
        oDatenquelle.connect()
    END IF
    ThisDatabaseDocument.FormDocuments.getByName("Formularname").open
END SUB
```

Zuerst muss der Kontakt mit der Datenquelle hergestellt werden. Der Controller hängt ebenso mit `ThisDatabaseDocument` zusammen wie das Formular. Anschließend kann das Formular gestartet werden und liest auch die Datenbankinhalte aus.

---

Makros: Datenbankaufgaben mit Makros erweitert 463
MySQL-Datenbank mit Makros ansprechen

Sämtliche bisher vorgestellten Makros wurden mit der internen HSQLDB verbunden. Bei der Arbeit mit externen Datenbanken sind ein paar Änderungen und Erweiterungen notwendig.

MySQL-Code in Makros

Wird die interne Datenbank angesprochen, so werden die Tabellen und Felder mit doppelten Anführungszeichen gegenüber dem SQL-Code abgesetzt:

SELECT "Feld" FROM "Tabelle"

Da in Makros der SQL-Befehl Text darstellt müssen die doppelten Anführungszeichen zusätzlich maskiert werden:

\texttt{stSQL = "SELECT "Feld" FROM "Tabelle""}

MySQL-Abfragen hingegen werden anders maskiert:

SELECT `Feld` FROM `Datenbank`.`Tabelle`

Durch diese andere Form der Maskierung wird daraus im Makro-Code:

\texttt{stSql = "SELECT `Feld` FROM `Datenbank`.`Tabelle`"}

Temporäre Tabelle als individueller Zwischenspeicher

In den vorhergehenden Kapiteln wurde häufiger eine einzeilige Tabelle zum Suchen oder Filtern von Tabellen genutzt. In einem Mehrbenutzersystem kann darauf nicht zurückgegriffen werden, da sonst andere Nutzer von dem Filterwert eines anderen Nutzers abhängig würden. Temporäre Tabellen sind in MySQL nur für den Nutzer der gerade aktiven Verbindung zugänglich, so dass für die Such- und Filterfunktionen auf diese Tabellenform zugegriffen werden kann.

Diese Tabellen können natürlich nicht vorher erstellt worden sein. Sie müssen beim Öffnen der Base-Datei erstellt werden. Deshalb ist das folgende Makro mit dem Öffnen der *.odb-Datei zu verbinden:

\texttt{SUB CreateTempTable}
\texttt{oDatenquelle = thisDatabaseDocument.CurrentController}
\texttt{IF NOT (oDatenquelle.isConnected()) THEN oDatenquelle.connect()}
\texttt{oVerbindung = oDatenquelle.ActiveConnection()}
\texttt{oSQL_Anweisung = oVerbindung.createStatement()}
\texttt{stSql = "CREATE TEMPORARY TABLE IF NOT EXISTS `Suchtmp` (`ID` INT PRIMARY KEY, `Name` VARCHAR(50))"}
\texttt{oSQL_Anweisung.executeUpdate(stSql)}
\texttt{END SUB}


Dialoge

Statt Formularen können für Base auch Dialoge zur Eingabe von Daten, zum Bearbeiten von Daten oder auch zur Wartung der Datenbank genutzt werden. Dialoge lassen sich auf das jeweilige Anwendungsgebiet direkt zuschneiden, sind aber natürlich nicht so komfortabel vordefiniert wie Formulare. Hier eine kurze Einführung, die mit einem recht komplexen Beispiel zur Datenbankwartung endet.
Dialoge starten und beenden


Wird der Dialog angeklickt, so gibt es bei den allgemeinen Eigenschaften die Möglichkeit, die Größe und Position einzustellen. Außerdem kann dort der Inhalt des Titels «Dialoge starten» eingegeben werden.

![Dialog starten](image)


Die Lage der Deklaration der Variablen für den Dialog ist besonders zu beachten. Der Dialog wird als globale Variable gesetzt, damit auf ihn von unterschiedlichen Prozeduren aus zugegriffen werden kann. In diesem Falle ist der Dialog mit der Variablen oDialog0 versehen, weil es noch weitere Dialoge gibt, die einfach entsprechend durchnummeriert wurden.

\[
\text{DIM oDialog0 AS OBJECT}
\]


\[
\text{SUB Dialog0Start}
\]

DialogLibraries.LoadLibrary("Standard")

\[
oDialog0 = \text{createUnoDialog(DialogLibraries.Standard.Dialog0)}
\]

\[
oDialog0.Execute()
\]

\[
\text{END SUB}
\]

Prinzipiell kann ein Dialog durch Betätigung des Schließkreuzes geschlossen werden. Soll dafür aber ein entsprechender Button vorgesehen werden, so reicht hier einfach der Befehl `EndExecute()` innerhalb einer Prozedur.

\[
\text{SUB Dialog0Ende}
\]

\[
oDialog0.EndExecute()
\]

\[
\text{END SUB}
\]

Mit diesem Rahmen können beliebig gestaltete Dialoge gestartet und wieder geschlossen werden.

---

Einfacher Dialog zur Eingabe neuer Datensätze


Mit

```vba
DIM oDialog1 AS OBJECT
```

wird wieder direkt auf der untersten Ebene des Moduls vor allen Prozeduren die globale Variable für den Dialog erstellt.

Der Dialog wird genauso gestartet und beendet wie der vorhergehende Dialog. Lediglich die Bezeichnung ändert sich von Dialog0 auf Dialog1. Die Prozedur zum Beenden des Dialogs ist mit dem Button Beenden verbunden.

Über den Button Neu werden alle Kontrollfelder des Dialogs durch die Prozedur «DatenfelderLeeren» von vorhergehenden Eingaben befreit:

```vba
SUB DatenfelderLeeren
    oDialog1.getControl("NumericField1").Text ="
    oDialog1.getControl("TextField1").Text ="
    oDialog1.getControl("TextField2").Text ="
END SUB
```

Jedes Feld, das in einen Dialog eingefügt wird, ist über einen eigenen Namen ansprechbar. Im Gegensatz zu Feldern eines Formulars wird hier durch die Benutzeroberfläche darauf geachtet, dass keine Namen doppelt vergeben werden.


Der Button Speichern löst schließlich die Prozedur «Daten1Speichern» aus:

```vba
SUB Daten1Speichern
    DIM oDatenquelle AS OBJECT
    DIM oVerbindung AS OBJECT
    DIM oSQL_Anweisung AS OBJECT
    DIM loID AS LONG
```

Makros: Dialoge 466

Sind die beiden Felder entsprechend mit Inhalt versehen, so wird eine Verbindung zur Datenbank aufgenommen. Da sich die Kontrollfelder nicht in einem Formular befinden muss die Datenbankverbindung über thisDatabaseDocument.CurrentController hergestellt werden.

Zuerst wird eine Abfrage an die Datenbank geschickt, ob vielleicht ein Datensatz mit dem vorgegebenen Primärschlüssel schon existiert. Hat die Abfrage Erfolg, so wird eine Meldung über eine MessageBox ausgegeben, die mit einem Stoppsymbol versehen ist (Code: 16) und die Überschrift «Doppelte Dateneingabe» trägt. Danach wird durch Exit SUB die Prozedur beendet.

Hat die Abfrage keinen Datensatz gefunden, der den gleichen Primärschlüssel hat, so wird der neue Datensatz über den Insert-Befehl in die Datenbank eingefügt. Anschließend wird über die Prozedur «DatenfelderLeeren» wieder ein leeres Formular präsentiert.
Dialog zum Bearbeiten von Daten in einer Tabelle


Die Button *Beenden* ist mit der entsprechend auf den Dialog2 abgewandelten Prozedur des vorhergehenden Dialogs zur Eingabe neuer Datensätze verbunden. Hier werden nur die weiteren Buttons mit ihren entsprechenden Funktionen beschrieben.

Die Dateneingabe ist im Dialog so beschränkt, dass im Feld «ID» der Mindestwert auf '1' eingestellt wurde. Diese Einschränkung hat mit dem Umgang mit Variablen in Basic zu tun: Zahlenvariablen werden bei der Definition bereits mit '0' als Grundwert vorbelegt. Werden Zahlenwerte von leeren Feldern und Feldern mit '0' ausgelesen, so ist für Basic der anschließende Inhalt der Variablen gleich. Es müsste bei der Nutzung von '0' im Feld «ID» also zur Unterscheidung erst Text ausgelesen und vielleicht später in eine Zahl umgewandelt werden.

Der Dialog wird unter den gleichen Voraussetzungen geladen wie vorher auch. Hier wird allerdings die Ladenprozedur davon abhängig gemacht, ob die Variable, die der Prozedur «DatenLaden» mitgegeben wird, 0 ist.

```vbnet
SUB DatenLaden(loID AS Long)
  DIM oDatenquelle AS OBJECT
  DIM oVerbindung AS OBJECT
  DIM oSQL_Anweisung AS OBJECT
  DIM stVorname AS String
  DIM stNachname AS String
  DIM loRow AS Long
  DIM loRowMax AS Long
  DIM inStart AS Integer
  oDatenquelle = thisDatabaseDocument.CurrentController
  If NOT(oDatenquelle.isConnected()) THEN
    oDatenquelle.connect()
  END IF
  oVerbindung = oDatenquelle.ActiveConnection()
  oSQL_Anweisung = oVerbindung.createStatement()
  IF loID < 1 THEN
    strSql = "SELECT MIN(""ID"" ) FROM ""Name"")
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(strSql)
    WHILE oAbfrageergebnis.next
      loID = oAbfrageergebnis.getInt(1)
    WEND
    inStart = 1
  END IF
```

Makros: Dialoge
Die Variablen werden deklariert. Die Datenbankverbindung wird, wie weiter oben erklärt, für den Dialog hergestellt. Zum Start ist \texttt{loID 0}. Für diesen Fall wird per SQL der niedrigste Wert für den Primärschlüssel ermittelt. Der entsprechende Datensatz soll in dem Dialog später angezeigt werden. Gleichzeitig wird die Variable \texttt{inStart} auf 1 gestellt, damit der Dialog später gestartet wird.

Enthält die Tabelle keine Daten, so bleibt \texttt{loID 0}. Entsprechend muss auch nicht nach dem Inhalt und der Anzahl irgendwelcher Datensätze im Folgenden gesucht werden.

Nur wenn \texttt{loID} größer als 0 ist wird zuerst mit einer Abfrage überprüft, welche Daten in dem Datensatz enthalten sind. Anschließend werden in einer zweiten Abfrage alle Datensätze für die Datensatzanzeige gezählt. Mit der dritten Abfrage wird die Position des aktuellen Datensatzes ermittelt, indem alle Datensätze, die einen kleineren oder eben den aktuellen Primärschlüssel haben, zusammengezählt werden.

\begin{verbatim}
IF loID > 0 THEN
  stSql = "SELECT * FROM ""Name"" WHERE ""ID"" = ''+loID++"" 
  oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
  WHILE oAbfrageergebnis.next
    loID = oAbfrageergebnis.getInt(1)
    stVorname = oAbfrageergebnis.getString(2)
    stNachname = oAbfrageergebnis.getString(3)
  WEND
  stSql = "SELECT COUNT(""ID"") FROM ""Name"
  oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
  WHILE oAbfrageergebnis.next
    loRowMax = oAbfrageergebnis.getInt(1)
  WEND
  stSql = "SELECT COUNT(""ID"") FROM ""Name"" WHERE ""ID"" <= ''+loID++""
  oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
  WHILE oAbfrageergebnis.next
    loRow = oAbfrageergebnis.getInt(1)
  WEND
  oDialog2.getControl("NumericField1").Value = loID
  oDialog2.getControl("TextField1").Text = stVorname
  oDialog2.getControl("TextField2").Text = stNachname
END IF

IF loRow = 1 THEN
  ' Vorheriger Datensatz
  oDialog2.getControl("CommandButton4").Model.enabled = False
ELSE
  oDialog2.getControl("CommandButton4").Model.enabled = True
END IF

IF loRow <= loRowMax THEN
  ' Nächster Datensatz | Neuer Datensatz | Löschen
  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 IF
END SUB
\end{verbatim}

Die ermittelten Werte für die Formularfelder werden übertragen. Die Einträge für die Nummer des aktuellen Datensatzes sowie die Anzahl aller Datensätze wird auf jeden Fall mit einer Zahl versorgt. Ist kein Datensatz vorhanden, so wird hier über den Default-Wert für eine numerische Variable 0 eingefügt.

Der Dialog soll möglichst nur dann gestartet werden, wenn er direkt aus einer Startdatei über DatenLaden(0) erstellt werden soll. Deshalb wurde die gesonderte Variable inStart mit dem Wert 1 zu Beginn der Prozedur versehen.


Hier gilt es zwei Fälle zu unterscheiden:

1. Es wurde vorher vorwärts zu einer Neueingabe navigiert, so dass das entsprechende Feld keinen Wert enthält. loID gibt dann den Standardwert wieder, der durch die Definition als Zahlenvariable vorgegeben ist: 0.

2. Ansonsten enthält loID einen Wert, der größer als 0 ist. Entsprechend kann über eine Abfrage die nächstkleinere «ID» ermittelt werden.

```
SUB vorherigerDatensatz
    DIM loID AS LONG
    DIM loIDneu AS LONG
    loID = oDialog2.getControl("NumericField1").Value
    oDatenquelle = thisDatabaseDocument.CurrentController
    IF NOT (oDatenquelle.isConnected()) THEN
        oDatenquelle.connect()
    END IF
    oVerbindung = oDatenquelle.ActiveConnection()
    oSQL_Anweisung = oVerbindung.createStatement()
    IF loID < 1 THEN
        stSql = "SELECT MAX("ID") FROM "Name"
    ELSE
        stSql = "SELECT MAX("ID") FROM "Name" WHERE "ID" < ""+loID+""
    END IF
    oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
    WHILE oAbfrageergebnis.next
        loIDneu = oAbfrageergebnis.getInt(1)
    WEND
    IF loIDneu > 0 THEN
        DatenLaden(loIDneu)
    END IF
END SUB
```

Bei einem leeren «ID»-Feld soll auf den Datensatz mit dem höchsten Wert in der Primärschlüsselnummer gewechselt werden. Können hingegen aus dem «ID»-Feld Daten entnommen werden, so wird der entsprechend nachrangige Wert für die "ID" ermittelt.

Das Ergebnis dieser Abfrage dient dazu, die Prozedur «DatenLaden» mit dem entsprechenden Schlüsselwert erneut durchlaufen zu lassen.

Über den Button > wird zum nächsten Datensatz navigiert. Diese Navigationsmöglichkeit steht nur zur Verfügung, wenn nicht bereits der Dialog für die Eingabe eines neuen Datensatzes geleert wurde. Dies ist natürlich auch beim Start und leerer Tabelle der Fall.

Zwangsläufig ist in dem Feld «NumericField1» ein Wert vorhanden. Von diesem Wert ausgehend kann also per SQL nachgesehen werden, welcher Primärschlüsselwert der nächsthöhere in der Tabelle ist. Bleibt die Abfrage leer, weil es keinen entsprechenden Datensatz gibt, so ist der Wert für loIDneu = 0. Ansonsten kann über die Prozedur «DatenLaden» der Inhalt des nächsten Datensatzes geladen werden.

Makros: Dialoge
SUB naechsterDatensatz
DIM loID AS LONG
DIM loIDneu AS LONG
loID = oDialog2.getControl("NumericField1").Value
oDatenquelle = thisDatabaseDocument.CurrentController
IF NOT (oDatenquelle.isConnected()) THEN
 oDatenquelle.connect()
END IF
oVerbindung = oDatenquelle.ActiveConnection()
oSQL_Anweisung = oVerbindung.createStatement()
stSql = "SELECT MIN("ID") FROM "Name" WHERE "ID" > '" + loID + "'
 oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
WHILE oAbfrageergebnis.next
  loIDneu = oAbfrageergebnis.getInt(1)
END WHILE
IF loIDneu > 0 THEN
  DatenLaden(loIDneu)
ELSE
  Datenfelder2Leeren
END IF
END SUB

Existiert beim Navigieren zum nächsten Datensatz kein weiterer Datensatz, so löst die Navigation die folgende Prozedur «Datenfelder2Leeren» aus, die zur Eingabe neuer Daten dient.

Mit der Prozedur «Datenfelder2Leeren» werden nicht nur die Datenfelder selbst geleert. Die Position des aktuellen Datensatzes wird um einen Datensatz höher als die maximale Datensatzzahl eingestellt. Das soll verdeutlichen, dass der aktuell bearbeitete Datensatz noch nicht in der Datenbank enthalten ist.

Sobald «Datenfelder2Leeren» ausgelöst wird, wird außerdem die Möglichkeit des Sprungs zum vorhergehenden Datensatz aktiviert. Sprünge zu einem nachfolgenden Datensatz, das erneute Aufrufen der Prozedur über Neu oder das Löschen sind deaktiviert.

SUB Datenfelder2Leeren
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 ' Vorheriger Datensatz
oDialog2.getControl("CommandButton5").Model.enabled = False ' Nächster Datensatz
oDialog2.getControl("CommandButton2").Model.enabled = False ' Neuer Datensatz
oDialog2.getControl("CommandButton6").Model.enabled = False ' Löschen
END SUB

Das Speichern der Daten soll nur möglich sein, wenn in den Feldern für «ID» und «Nachname» ein Eintrag erfolgt ist. Ist diese Bedingung erfüllt, so wird überprüft, ob der Datensatz ein neuer Datensatz ist. Das funktioniert über den Datensatzanzeiger, der bei neuen Datensätzen so eingestellt wurde, dass er für den aktuellen Datensatz einen um 1 höheren Wert als den maximalen Datensatzwert ausgibt.

Im Falle eines neuen Datensatzes gibt es weitere Überprüfungsschritte, damit eine Speicherung einwandfrei funktionieren kann. Kommt die Ziffer für den Primärschlüssel bereits einmal vor, so erfolgt eine Warnung. Wird die entsprechende Frage mit ja bestätigt, so wird der alte Datensatz mit der gleichen Schlüsselnummer überschrieben, Ansonsten erfolgt keine Speicherung. Solange noch gar kein Datensatz in der Datenbank enthalten ist (loRowMax = 0) braucht diese Überprüfung nicht zu erfolgen. In dem Falle kann der Datensatz direkt als neuer Datensatz abgespeichert werden. Bei einem neuen Datensatz wird schließlich noch die Zahl der Datensätze um 1 erhöht und die Eingabe für den nächsten Datensatz freigegeben.

Bei bestehenden Datensätzen wird einfach der alte Datensatz durch ein Update mit dem neuen Datensatz überschrieben.

SUB Daten2Speichern(oEvent AS OBJECT)
DIM oDatenquelle AS OBJECT
Die Löschprozedur ist mit einer Nachfrage versehen, die versehentliches Löschen verhindern soll. Dadurch, dass der Button deaktiviert wird, wenn die Eingabefelder leer sind, dürfte es nicht vor-
kommen, dass das Feld «NumericField1» leer ist. Deshalb könnte die Überprüfung der Bedingung
IF loID > 0 auch entfallen.

Beim Löschen wird die Zahl der Datensätze um einen Datensatz herabgesetzt. Dies muss entspre-
chend mit \[loRowMax - 1\] korrigiert werden. Anschließend wird der dem aktuellen Datensatz fol-
gende Datensatz angezeigt.

SUB DatenLoeschen(oEvent AS OBJECT)
DIM oDatenquelle AS OBJECT
DIM oVerbindung AS OBJECT
END SUB

**Fehleinträge von Tabellen mit Hilfe eines Dialogs bereinigen**

Fehleingaben in Feldern fallen häufig erst später auf. Manchmal müssen auch gleich mehrere Datensätze mit der gleichen Eingabe auf einmal geändert werden. Dies ist in der normalen Tabelle ansicht umständlich, je mehr Änderungen vorgenommen werden müssen, da für jeden Datensatz einzeln eine Eingabe erforderlich ist.

Formulare könnten hier mit Makros greifen. Wird aber für viele Tabellen ein identisch aufgebautes Formular benötigt, so bietet sich an, dies mit Dialogen zu erledigen. Ein Dialog wird zu Beginn mit den notwendigen Daten zu der jeweiligen Tabelle versehen und kann so statt mehrerer Formulare genutzt werden.

Wird ein Dialog ausgeführt, so erscheinen die Kontrollfelder entsprechend der Einstellung der grafischen Benutzeroberfläche.

Der oben abgebildete Dialog der Beispieldatenbank soll dazu dienen, die Tabellen zu bearbeiten, die nicht direkt in einem der Formulare als Grundlage vorhanden sind. So ist z.B. die Medienart über ein Listenfeld zugänglich, in der Makro-Version bereits durch ein Kombinationsfeld. In der Makro-Version können die Inhalte der Felder zwar durch neue Inhalte ergänzt werden, eine Änderung alter Inhalte ist aber nicht möglich. In der Version ohne Makros erfolgt die Änderung über ein separates Tabellenkontrollfeld.

Während die Änderung noch ohne Makros recht einfach in den Griff zu bekommen ist, so ist es doch recht umständlich, die Medienart vieler Medien auf eine andere Medienart zu ändern. Ange nommen, es gäbe die Medienarten 'Buch, gebunden', 'Buch, kartoniert', 'Taschenbuch' und 'Ring buch'. Jetzt stellt sich nach längerem Betrieb der Datenbank heraus, dass muntere Zeitgenossen noch weitere ähnliche Medienarten für gedruckte Werke vorgesehen haben. Nur ist uns die Differenzierung viel zu weitgehend. Es soll also reduziert werden, am liebsten auf nur einen Begriff.


Sollen mit dem Dialog mehrere Änderungen durchgeführt werden, so ist dies über das Markierfeld «Mehrere Datensätze bearbeiten» anzugeben. Dann endet der Dialog nicht mit der Betätigung des Buttons «OK».

Der Makrocode für diesen Dialog ist aus der Beispieldatenbank ersichtlich. Im Folgenden werden nur Ausschnitte daraus erläutert.

```vba
SUB Tabellenbereinigung(oEvent AS OBJECT)
Das Makro soll über Einträge im Bereich «Zusatzinformationen» des jeweiligen Buttons gestartet werden.

0: Formular, 1: Unterformular, 2: UnterUnterformular, 3: Kombinationsfeld oder Tabellenkontrollfeld, 4: Fremdschlüsselfeld im Formular, bei Tabellenkontrollfeld leer, 5: Tabellenname Nebentabelle, 6: Tabellenfeld1 Nebentabelle, 7: Tabellenfeld2 Nebentabelle, ggf. 8: Tabellenname Nebentabelle für Tabellenfeld 2

Die Einträge in diesem Bereich werden zu Beginn des Makros als Kommentar aufgelistet. Die damit verbundenen Ziffern geben die Ziffern wieder, unter denen der jeweilige Eintrag aus dem Array ausgelesen wird. Das Makro kann Listenfelder verarbeiten, die zwei Einträge, getrennt durch «;», enthalten. Diese beiden Einträge können auch aus unterschiedlichen Tabellen stammen und über eine Abfrage zusammengeführt sein, wie z.B. bei der Tabelle "Postleitzahl", die für die Orte lediglich das Fremdschlüsselfeld "Ort_ID" enthält, zur Darstellung des Ortes also die Tabelle "Ort" benötigt.

```vba
DIM aFremdTabellen(0, 0 to 1)
DIM aFremdTabellen2(0, 0 to 1)

Unter den zu Beginn definierten Variablen fallen zwei Arrays auf. Während normale Arrays auch durch den Befehl 'Split()' während der Laufzeit der Prozedur erstellt werden können, müssen zweidimensionale Arrays vorher definiert werden. Zweidimensionale Arrays werden z.B. benötigt, um aus einer Abfrage mehrere Datensätze zu speichern, bei denen die Abfrage selbst über mehr als ein Feld geht. Die beiden obigen Arrays müssen Abfragen auswerten, die sich jeweils auf zwei Tabellenfelder beziehen. Deshalb werden sie in der zweiten Dimension mit '0 to 1' auf zwei unterschiedliche Inhalte festgelegt.

```vba
stTag = oEvent.Source.Model.Tag
aTabelle() = Split(stTag, ";")
FOR i = LBound(aTabelle()) TO UBound(aTabelle())
aTabelle(i) = trim(aTabelle(i))
NEXT


Wenn Werte aus einer Tabelle entfernt werden, so muss zuerst einmal berücksichtigt werden, ob sie nicht noch als Fremdschlüssel in anderen Tabellen existieren. In einfachen Tabellenkonstruktionen gibt es von einer Tabelle aus lediglich eine Fremdschlüsselverbindung zu einer anderen
Tabelle. In der vorliegenden Beispieldatenbank aber wird z.B. die Tabelle "Ort" genutzt, um die Erscheinungsorte der Medien und die Orte für die Adressen zu speichern. Es wird also zweimal der Primärschlüssel der Tabelle "Ort" in unterschiedlichen Tabellen eingetragen. Diese Tabellen und Fremdschlüsselbezeichnungen könnten natürlich auch über die «Zusatzinformationen» eingegangen werden. Schöner wäre es aber, wenn sie universell für alle Fälle ermittelt werden. Dies geschieht durch die folgende Abfrage.

```
stSql = "SELECT "FKTABLE_NAME", "FKCOLUMN_NAME" FROM "INFORMATION_SCHEMA"."SYSTEM_CROSSREFERENCE" WHERE "PKTABLE_NAME" = "" + aTabelle(5) + ""
```

In der Datenbank sind im Bereich "INFORMATION_SCHEMA" alle Informationen zu den Tabellen der Datenbank abgespeichert, so auch die Informationen zu den Fremdschlüsseln. Die entsprechende Tabelle, die diese Informationen enthält, ist über "INFORMATION_SCHEMA"."SYSTEM_CROSSREFERENCE" erreichbar. Mit "PKTABLE_NAME" wird die Tabelle erreicht, die ihren Primärschlüssel ("Primary Key") in die Beziehung mit einbringt. Mit "FKTABLE_NAME" wird die Tabelle erreicht, die diesen Primärschlüssel als Fremdschlüssel ("Foreign Key") nutzt. Über "FKCOLUMN_NAME" wird schließlich die Bezeichnung des Fremdschlüsselfeldes ermittelt.


```
inZaehler = 0
stFremdIDTab1Tab2 = "ID"
stFremdIDTab2Tab1 = "ID"
stNebentabelle = aTabelle(5)
```

Bevor die Auslesung des Arrays gestartet wird, müssen einige Standardwerte gesetzt werden. Dies sind der Zähler für das Array, in das die Werte der Nebentabelle geschrieben werden, der Standardprimärschlüssel, wenn nicht der Fremdschlüssel für eine zweite Tabelle benötigt wird und die Standardnebentabelle, die sich auf die Haupttabelle bezieht, bei Postleitzahl und Ort z.B. die Tabelle für die Postleitzahl.

Bei der Verknüpfung von zwei Feldern zur Anzeige in den Listenfeldern kann es ja, wie oben erwähnt, zu einer Verknüpfung über zwei Tabellen kommen. Für die Darstellung von Postleitzahl und Ort lautet hier die Abfrage

```
SELECT "Postleitzahl"."Postleitzahl" || ' > ' || "Ort"."Ort"
FROM "Postleitzahl", "Ort"
WHERE "Postleitzahl"."Ort_ID" = "Ort"."ID"
```

Die Tabelle, die sich auf das erste Feld bezieht (Postleitzahl), ist mit der zweiten Tabelle über einen Fremdschlüssel verbunden. Lediglich die Information der beiden Tabellen und der Felder "Postleitzahl" und "Ort" wurde dem Makro mitgegeben. Die Primärschlüssel sind standardmäßig in dem Beispiel mit der Bezeichnung "ID" versehen. Der Fremdschlüssel von "Ort" in "Postleitzahl" muss also über das Makro ermittelt werden.

Genauso muss über das Makro jede andere Tabelle ermittelt werden, mit der die Inhalte des Listenfeldes über Fremdschlüssel in Verbindung stehen.

```
oAbfrageergebnis = oSQL_Anweisung.executeQuery(stSql)
WHILE oAbfrageergebnis.next
    ReDim Preserve aFremdTabellen(inZaehler,0 to 1)
```

Das Array muss jedes Mal neu Dimensioniert werden. Damit die alten Inhalte erhalten bleiben erfolgt über (Preserve) eine Sicherung des vorherigen Inhaltes.

```
aFremdTabellen(inZaehler,0) = oAbfrageergebnis.getString(1)
aFremdTabellen(inZaehler,1) = oAbfrageergebnis.getString(2)
```

---

*Makros: Dialoge*
Auslesen des zweiten Feldes mit der Bezeichnung des Fremdschlüsselfeldes. Ergebnis für die Tabelle "Postleitzahl" ist hier das Feld "Postleitzahl_ID" in der Tabelle "Adresse".

Für den Fall, dass dem Aufruf der Prozedur auch der Name einer zweiten Tabelle mitgegeben wurde erfolgt die folgende Schleife. Nur wenn der Name der zweiten Tabelle als Fremdschlüssel der erste Tabelle auftaucht erfolgt hier eine Änderung der Standardeinträge. In unserem Fall kommt dies nicht vor, da die Tabelle "Ort" keinen Fremdschlüssel der Tabelle "Postleitzahl" enthält. Der Standardeinträuf für die Nebentabelle bleibt also bei "Postleitzahl"; schließlich ist die Kombination von Postleitzahl und Ort eine Grundlage für die Adressentabelle, die einen Fremdschlüssel zu der Tabelle "Postleitzahl" enthält.

IF UBound(aTabelle()) = 8 THEN
  IF aTabelle(8) = aFremdTabellen(inZaehler, 0) THEN
    stFremdIDTab2Tab1 = aFremdTabellen(inZaehler, 1)
    stNebentabelle = aTabelle(8)
  END IF
END IF
inZaehler = inZaehler + 1

Da eventuell noch weitere Werte auszulesen sind, erfolgt eine Erweiterung des Zählers zur Neudimensionierung des Arrays. Anschließend wird die Schleife beendet.

WEND

Existiert im Aufruf der Prozedur ein zweiter Tabellenname, so wird die gleiche Abfrage jetzt mit dem zweiten Tabellennamen gestartet:

IF UBound(aTabelle()) = 8 THEN
  Der Ablauf ist identisch. Nur wird in der Schleife jetzt gesucht, ob vielleicht der erste Tabellenname als Fremdschlüssel-Tabellenname auftaucht. Das ist hier der Fall: Die Tabelle "Postleitzahl" enthält den Fremdschlüssel "Ort_ID" aus der Tabelle "Ort". Dieser Fremdschlüssel wird also jetzt der Variablen «stFremdIDTab1Tab2» zugewiesen, so dass die Beziehung der Tabellen untereinander definiert werden kann.

IF aTabelle(5) = aFremdTabellen2(inZaehler, 0) THEN
  stFremdIDTab1Tab2 = aFremdTabellen2(inZaehler, 1)
END IF

Nach einigen weiteren Einstellungen zur korrekten Rückkehr nach Aufruf des Dialogs in die entsprechenden Formulare (Ermittlung der Zeilenummer des Formulars, damit nach einem Neueinlesen auf die Zeilenummer wieder gesprungen werden kann) startet die Schleife, die den Dialog gegebenenfalls wieder neu erstellt, wenn die erste Aktion erfolgt ist, der Dialog aber für weitere Aktionen offen gehalten werden soll. Die Einstellung zur Wiederholung erfolgt über das entsprechende Markierfeld.

DO

Bevor der Dialog gestartet wird, wird erst einmal der Inhalt der Listenfelder ermittelt. Dabei muss berücksichtigt werden, ob die Listenfelder zwei Tabellenfelder darstellen und eventuell sogar einen Bezug zu zwei Tabellen haben.

IF UBound(aTabelle()) = 6 THEN
  Das Listenfeld bezieht sich nur auf eine Tabelle und ein Feld, da das Array bei dem Tabellenfeld1 der Nebentabelle endet.
  stSql = "SELECT " + aTabelle(6) + " FROM " + aTabelle(5) + " ORDER BY " + aTabelle(6) + ""
ELSEIF UBound(aTabelle()) = 7 THEN
  Das Listenfeld bezieht sich auf zwei Tabellenfelder, aber nur auf eine Tabelle, da das Array bei dem Tabellenfeld2 der Nebentabelle endet.
  stSql = "SELECT " + aTabelle(6) + "|" + aTabelle(5) + " ORDER BY " + aTabelle(6) + ""
ELSE
  Das Listenfeld hat zwei Tabellenfelder und zwei Tabellen als Grundlage. Diese Abfrage trifft also auf das Beispiel mit der Postleitzahl und den Ort zu.

Makros: Dialoge 477
stSql = "SELECT "" + aTabelle(5) + "","" + aTabelle(6) + "",""||' > '|"" + aTabelle(8) + "","" + aTabelle(7) + "","" FROM "" + aTabelle(5) + "","" + aTabelle(8) + "","" WHERE "" + aTabelle(8) + "","" + stFremdIDTab2Tab1 + "","" = "","" + aTabelle(5) + "",""||' > '||"" + aTabelle(8) + "","" + aTabelle(7) + "","" FROM "" + aTabelle(5) + "","" + aTabelle(8) + "","" WHERE "" + aTabelle(8) + "","" + stFremdIDTab1Tab2 + "","" ORDER BY "","" + aTabelle(6) + "","

END IF

Hier erfolgt die erste Auswertung zur Ermittlung von Fremdschlüsseln. Die Variablen «stFremdIDTab2Tab1» und «stFremdIDTab1Tab2» starten mit dem Wert "ID". Für «stFremdIDTab1Tab2» wurde in der Auswertung der vorhergehenden Abfrage ein anderer Wert ermittelt, nämlich der Wert "Ort_ID". Damit ergibt die vorherige Abfragekonstruktion genau den Inhalt, der weiter oben bereits für Postleitzahl und Ort formuliert wurde – lediglich erweitert um die Sortierung.


DialogLibraries.LoadLibrary("Standard")
oDlg = CreateUnoDialog(DialogLibraries.Standard.Dialog_Tabellenbereinigung)

Anschließend werden Einstellungen für die Felder, die der Dialog enthält, ausgeführt. Hier als Beispiel das Auswahllistenfeld, das mit dem Ergebnis der obigen Abfrage bestückt wird:

oCtlList1 = oDlg.GetControl("ListBox1")
oCtlList1.addItems(aInhalt(),0)

Der Zugriff auf die Felder des Dialogs erfolgt über GetControl sowie die entsprechende Bezeichnung. Bei Dialogen ist es nicht möglich, für zwei Felder die gleichen Bezeichnungen zu verwenden, da sonst eine Auswertung des Dialoges problematisch wäre.

Das Listenfeld wird mit den Inhalten aus der Abfrage, die in dem Array «aInhalt()» gespeichert wurden, ausgestattet. Das Listenfeld enthält nur die darzustellenden Inhalte als ein Feld, wird also nur in der Position '0' bestückt.

Nachdem alle Felder mit den gewünschten Inhalten versorgt wurden wird der Dialog gestartet.

Select Case oDlg.Execute()
Case 1 'Case 1 bedeutet die Betätigung des Buttons "OK"
Case 0 'Wenn Button "Abbrechen"
inWiederholung = 0
End Select
LOOP WHILE inWiederholung = 1

Der Dialog wird so lange durchgeführt, wie der Wert für «inWiederholung» auf 1 steht. Diese Setzung erfolgt mit dem entsprechenden Markierfeld.

Hier der Inhalt nach Betätigung des Buttons «OK» im Kurzüberblick:

Case 1
stInhalt1 = oCtlList1.getSelectedItem() 'Wert aus ListBox1 auslesen ...
REM ... und den dazugehörigen ID-Wert bestimmen.

Der ID-Wert des ersten Listenfeldes wird in der Variablen «inLB1» gespeichert.

stText = oCtlText.Text ' Den Wert des Feldes auslesen.

Ist das Textfeld nicht leer, so wird nur der Eintrag im Textfeld erledigt. Weder das Listenfeld für eine andere Zuweisung noch das Markierfeld für eine Lösung aller Daten ohne Bezug werden berücksichtigt. Dies wird auch dadurch verdeutlicht, dass bei Texteingabe die anderen Felder inaktiv geschaltet werden.

IF stText <> "" THEN


Makros: Dialoge
ELSEIF oCtlList2.getSelectedItem() <> "" THEN

Wenn das Textfeld leer ist und das Listenfeld 2 einen Wert aufweist muss der Wert des Listenfeldes 1 durch den Wert des Listenfeldes 2 ersetzt werden. Das bedeutet, dass alle Datensätze der Tabellen, in denen die Datensätze der Listenfelder Fremdschlüssel sind, überprüft und gegebenenfalls mit einem geänderten Fremdschlüssel beschrieben werden müssen.

stInhalt2 = oCtlList2.getSelectedItem()
REM Den Wert der Listbox auslesen.
REM ID für den Wert das Listenfeld ermitteln.

Der ID-Wert des zweiten Listenfeldes wird in der Variablen «inLB2» gespeichert. Auch dieses erfolgt wieder unterschiedlich, je nachdem, ob ein oder zwei Felder in dem Listenfeld enthalten sind, sowie eine oder zwei Tabellen Ursprungstabelle des Listenfeldinhaltes sind.

Der Ersetzungsprozess erfolgt danach, welche Tabelle als die Tabelle definiert wurde, die für die Haupttabelle den Fremdschlüssel darstellt. Für das oben erwähnte Beispiel ist dies die Tabelle "Postleitzahl", da die "Postleitzahl_ID" der Fremdschlüssel ist, der durch Listenfeld 1 und Listenfeld 2 wiedergegeben wird.

IF stNebentabelle = aTabelle(5) THEN
 FOR i = LBound(aFremdTabellen()) TO UBound(aFremdTabellen())

Sollte also der Fremdschlüssel die Eigenschaft 'UNIQUE' haben und bereits mit der gewünschten zukünftigen «inLB2» dort vertreten sein, so kann der Schlüssel nicht ersetzt werden.

stSql = "SELECT "COLUMN_NAME" FROM "INFORMATION_SCHEMA"."SYSTEM_INDEXINFO"
 WHERE "TABLE_NAME" = '" + aFremdTabellen(i,0) + '" AND "NON_UNIQUE" = False
 AND "INDEX_NAME" = (SELECT "INDEX_NAME" FROM "INFORMATION_SCHEMA"."SYSTEM_INDEXINFO" WHERE "TABLE_NAME" = '"
 + aFremdTabellen(i,0) + '" AND "COLUMN_NAME" = '"
 + aFremdTabellen(i,1) + '"

Mit "NON_UNIQUE" = False werden die Spaltennamen angegeben, die 'UNIQUE' sind. Allerdings werden nicht alle Spaltennamen benötigt sondern nur die, die gemeinsam mit dem Fremdschlüsselfeld einen Index bilden. Dies ermittelt der 'Subselect' mit dem gleichen Tabellennamen (der den Fremdschlüssel enthält) und dem Namen des Fremdschlüsselfeldes.

Wenn jetzt der Fremdschlüssel in der Ergebnismenge vorhanden ist, dann darf der Schlüsselwert nur dann ersetzt werden, wenn gleichzeitig andere Felder dazu benutzt werden, den entsprechenden Index als 'UNIQUE' zu definieren. Hierzu muss beim Ersetzen darauf geachtet werden, dass die Einzigartigkeit der Indexkombination nicht verletzt wird.

IF aFremdTabellen(i,1) = stFeldbezeichnung THEN
 inUnique = 1
ELSE
 ReDim Preserve aSpalten(inZaehler)
 aSpalten(inZaehler) = oAbfrageergebnis.getString(1)
 inZaehler = inZaehler + 1
 END IF

Alle Spaltennamen, die neben dem bereits bekannten Spaltennamen des Fremdschlüsselfeldes als Index mit der Eigenschaft UNIQUE auftauchen, werden in einem Array abgespeichert. Da der Spaltenname des Fremdschlüsselfeldes auch zu der Gruppe gehört wird durch ihn gekennzeichnet, dass die Einzigartigkeit bei der Datenänderung zu berücksichtigen ist.

IF inUnique = 1 THEN
 stSql = "UPDATE " + aFremdTabellen(i,0) + " AS " + "a" SET "
 + aFremdTabellen(i,1) + "='" + inLB2 + "' WHERE " + aFremdTabellen(i,1)

Makros: Dialoge 479
Gibt es mehrere Felder, die neben dem Fremdschlüsselfeld gemeinsam einen 'UNIQUE'-Index bilden, so werden die hier für eine SQL-Gruppierung zusammengeführt. Ansonsten erscheint als «stFeldgruppe» nur «aSpalten(0)».

Die SQL-Teilstücke werden für eine korrellierte Unterabfrage zusammengefügt.

WENN die Feldbezeichnung des Fremdschlüssels nichts mit dem Primärschlüssel oder einem 'UNIQUE'-Index zu tun hat, dann kann ohne weiteres auch ein Inhalt doppelt erscheinen

ELSE

oSQL_Anweisung.executeUpdate(stSql)

END IF

Das Update wird so lange durchgeführt, wie unterschiedliche Verbindungen zu anderen Tabellen mit vorkommen, d.h. Die aktuelle Tabelle einen Fremdschlüssel in anderen Tabellen liegen hat. Dies ist z.B. Bei der Tabelle "Ort" zweimal der Fall: in der Tabelle "Medien" und in der Tabelle "Postleitzahl".

Anschließend kann der alte Wert aus dem Listenfeld 1 gelöscht werden, weil er keine Verbindung mehr zu anderen Tabellen hat.

Das gleiche Verfahren muss jetzt auch für eine eventuelle zweite Tabelle durchgeführt werden, aus der die Listenfelder gespeist werden. In unserem Beispiel ist die erste Tabelle die Tabelle "Postleitzahl", die zweite Tabelle die Tabelle "Ort". Wenn das Textfeld leer ist und das Listenfeld 2 ebenfalls nichts enthält wird nachgesehen, ob eventuell das Markierfeld darauf hindeutet, dass alle überflüssigen Einträge zu löschen sind. Dies ist für die Einträge der Fall, die nicht mit anderen Tabellen über einen Fremdschlüssel verbunden sind.

ELSEIF oCtlCheck1.State = 1 THEN

END IF

Makros: Dialoge 480
Das letzte «AND» muss abgeschnitten werden, da sonst die Löschanweisung mit einem «AND»
enden würde.

```vba
    stBedingung = Left(stBedingung, Len(stBedingung) - 4)'
    strSql = "DELETE FROM " + stNebentabelle + "" WHERE " + stBedingung + ""
    oSQL_Anweisung.executeQuery(strSql)
```

Da nun schon einmal die Tabelle bereinigt wurde kann auch gleich der Tabellenindex überprüft und
gegebenenfalls nach unten korrigiert werden. Siehe hierzu die in dem vorhergehenden Kapitel
`Tabellenindex heruntersetzen bei Autowert-Feldern` erwähnte Prozedur.

```vba
    Tabellenindex_runter(stNebentabelle)
```

Anschließend wird noch gegebenenfalls das Listenfeld des Formulars, aus dem der Tabellenberei-
nigungsdialog aufgerufen wurde, auf den neuesten Stand gebracht. Unter Umständen ist das
gesamte Formular neu einzulesen. Hierzu wurde zu Beginn der Prozedur der aktuelle Datensatz
ermittelt, so dass nach einem Auffrischen des Formulars der aktuelle Datensatz auch wieder eing-
estellt werden kann.

```vba
    oDlg.endExecute()  'Dialog beenden ...
    oDlg.Dispose()    '... und aus dem Speicher entfernen
END SUB
```

Dialoge werden mit `endExecute()` beendet und mit `Dispose()` komplett aus dem Speicher ent-
fernt.

---

**Makrozugriff mit Access2Base**

In LibreOffice ist seit der Version 4.2 die Erweiterung Access2Base integriert. Der Zugriff auf diese Bibliothek erfolgt über

```vba
Sub DBOpen(Optional oEvent As Object)
    If GlobalScope.BasicLibraries.hasByName("Access2Base") then
        GlobalScope.BasicLibraries.loadLibrary("Access2Base")
    End If
    Call Application.OpenConnection(ThisDatabaseDocument)
End Sub
```

Eine englischsprachige Beschreibung mit Beispielen ist auf der Seite `http://www.access2base.com/access2base.html` zu finden.

Die Bibliothek stellt nicht zusätzliche Funktionen zur Verfügung, sondern versucht, dem Anwender
den Zugriff auf die Möglichkeiten der LibreOffice-API zu vereinfachen. Eine kurze Beschreibung ist
auch in der Hilfe zu LO zu finden.
Wartung
Allgemeines zur Wartung von Datenbanken

Werden in einer Datenbank die Datenbestände häufig geändert, vor allem auch gelöscht, so macht sich das an zwei Stellen besonders bemerkbar. Zum einen wird die Datenbank immer größer, obwohl sie ja eigentlich gar nicht mehr Daten enthält. Zum anderen wird der automatisch erstellte Primärschlüssel einfach weiter hoch geschrieben ohne Rücksicht auf die tatsächlich erforderliche nächste Schlüsselzahl.

Datenbank komprimieren

Die HSQLDB hat die Eigenart, auch für bereits gelöschte Daten weiterhin Speicherplatz bereitzustellen. Datenbanken, die zum Test einmal mit Daten, vor allem auch Bildern gefüllt waren, weisen auch dann noch die gleiche Größe auf, wenn all diese Daten gelöscht wurden.

Um den Speicherplatz wieder frei zu geben, müssen die Datenbankdateien neu geschrieben werden (Tabellen, Beschreibungen zu diesen Tabellen usw.).

Direkt auf der Oberfläche von Base kann über Extras → SQL ein einfaches Kommando direkt eingegeben werden, dass bei Serverdatenbanken nur dem Systemadministrator vorbehalten ist:

```
SHUTDOWN COMPACT
```

Die Datenbank wird heruntergefahren und dabei von allen Altlasten befreit. Anschließend muss allerdings Base neu gestartet werden, wenn wieder auf die Datenbank zugegriffen werden soll.

Seit der Version LO 3.6 wird die Komprimierung standardmäßig spätestens beim Schließen der Datenbank durchgeführt.

Autowerte neu einstellen

Eine Datenbank wird erstellt, alle möglichen Funktionen mit Beispieldaten ausprobiert, Korrekturen vorgenommen, bis alles klappt. Dann ist mittlerweile der Wert für manch einen Primärschlüssel über 100 gestiegen. Pech nur, wenn der Primärschlüssel, wie meistens üblich, als Autowertschlüssel angelegt wurde. Werden die Tabellen zum Normalbetrieb oder zur Weitergabe der Datenbank an andere Personen geleert, so zählt anschließend der Primärschlüssel trotzdem weiter und startet nicht neu ab 0.

Mit dem folgenden SQL-Kommando, wieder eingegeben über Extras → SQL, lässt sich der Startwert neu festlegen:

```
ALTER TABLE "Tabellenname" ALTER COLUMN "ID" RESTART WITH NeuerWert
```

Hier wird davon ausgegangen, dass das Primärschlüssel Feld die Bezeichnung "ID" hat und außerdem als Autowert definiert wird. Der neue Wert sollte der sein, der als nächster automatisch eingefügt wird. Existieren z.B. noch Einträge bis zum Wert 4, so ist als neuer Wert 5 (ohne einfache Anführungsstriche) anzugeben.

Datenbankeigenschaften abfragen

In einem gesonderten Bereich der HSQLDB sind alle Informationen zu den Tabellen der Datenbank in Tabellenform abgelagert. Dieser besondere Bereich ist über den Zusatz "INFORMATION_SCHEMA" zu erreichen.

Mit der folgenden Abfrage können Feldnamen, Feldtypen, Spaltengrößen und Standardwerte ermittelt werden, hier am Beispiel der Tabelle mit dem Namen 'Suchtabelle'.

Wartung: Datenbankeigenschaften abfragen
SELECT "COLUMN_NAME", "TYPE_NAME", "COLUMN_SIZE", "COLUMN_DEF" AS "Default Value" FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" WHERE "TABLE_NAME" = 'Suchtabelle' ORDER BY "ORDINAL_POSITION"

Im Anhang sind alle Spezialtabellen der HSQLDB aufgeführt. Informationen über den Inhalt der jeweiligen Tabelle sind am einfachsten über direkte Abfragen zu erreichen:

SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS"

Über das Sternchen werden alle verfügbaren Spalten der Tabelle angezeigt. Die obige Tabelle gibt dabei in der Hauptsache Auskunft über die Primärschlüssel der verschiedenen Tabellen.

Diese Informationen lassen sich besonders mittels Makros gut nutzen. So können statt detaillierter Informationen zu einer gerade erstellten Tabelle oder Datenbank gleich Prozeduren geschrieben werden, die diese Informationen direkt aus der Datenbank holen und somit universeller nutzbar sind. Die Beispieldatenbank zeigt dies unter anderem an der Ermittlung von Fremdschlüsseln bei einem entsprechenden Wartungsmodul.

### Daten exportieren

Neben der Möglichkeit, Daten durch öffnen der gepackten *.odb-Datei zu exportieren, existiert auch eine wesentlich einfachere Möglichkeit. Direkt auf der Oberfläche von Base kann über Extras → SQL ein einfaches Kommando direkt eingegeben werden, dass bei Serverdatenbanken nur dem Systemadministrator vorbehalten ist:

```
SCRIPT 'Datenbank'
```


Das Format der exportierter Datei lässt sich über

```
SET SCRIPTFORMAT {TEXT | BINARY | COMPRESSED};
```

ändern.

Die entsprechende Datei kann über Extras → SQL eingelesen werden und erzeugt so eine Datenbank mit den entsprechenden Daten. Dabei müssen für eine interne Datenbank allerdings die folgenden Zeilen entfernt werden:

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
```

Das Schema existiert schon. Es wird also ein ungültiger Schema-Name bemängelt.

```
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
SET SCHEMA PUBLIC
```

Tabellen auf unnötige Einträge überprüfen


Einträge durch Beziehungsdefinition kontrollieren

Über die Definition von Beziehungen kann die Integrität der Daten abgesichert werden. Das heißt, dass verhindert wird, dass die Löschung oder Änderung von Schlüsseln zu Fehlermeldungen der Datenbank führt.

Das folgende Fenster steht in Extras → Beziehungen nach einem Rechtsklick auf die Verbindungslinie zwischen zwei Tabellen zur Verfügung.

Hier sind die Tabelle "Adresse" und "Strasse" betroffen. **Alle aufgeführten Aktionen gelten für die Tabelle "Adresse"**, die den Fremdschlüssel "Strasse_ID" enthält. Update-Optionen beziehen sich auf ein Update des Feldes "ID" aus der Tabelle "Strasse". Wenn also in dem Feld "Strasse"."ID" die Schlüsselnummer geändert wird, so bedeutet «Keine Aktion», dass sich die Datenbank gegen diese Änderung wehrt, wenn "Strasse"."ID" mit der entsprechenden Schlüsselnummer als Fremdschlüssel in der Tabelle "Adresse" verwandt wird.

Wird «Null setzen» gewählt, so erzeugt die Änderung der "ID" in "Adresse"."Strasse_ID" ein leeres Feld.

Entsprechend werden die Löschoptionen gehandhabt.

Für beide Optionen steht über die GUI die Möglichkeit «Default setzen» zur Zeit nicht zur Verfügung, da die GUI-Standardeinstellungen sich von denen der Datenbank unterscheiden. Siehe hierzu den Abschnitt «Default setzen» aus dem Kapitel Tabellen.

Die Beziehungsdefinition hilft also nur die Beziehung selbst sauber zu halten, nicht aber unnötige Datensätze in der Tabelle zu entfernen, die ihren Primärschlüssel als Fremdschlüssel in der Beziehung zur Verfügung stellt. Es können also beliebig viele Straßen ohne Adresse existieren.

Einträge durch Formular und Unterformular bearbeiten


Die erste Abbildung zeigt, dass der Titel 'I hear you knocking' dem Verfasser 'Dave Edmunds' zugeordnet wurde. 'Dave Edmunds' darf also nicht gelöscht werden – sonst fehlt eine Information zu dem Medium 'I hear you knocking'. Es kann aber durch das Listenfeld statt 'Dave Edmunds' ein anderer Datensatz ausgewählt werden.

In dem Formular ist ein Filter eingebaut, so dass bei Betätigung des Filters zu erkennen ist, welche Kategorien denn überhaupt nicht in der Tabelle Medien benötigt werden. In dem gerade abgebildeten Fall sind fast alle Beispielverfasser in Benutzung. Es kann also nur der Datensatz 'Erich Kästner' gelöscht werden, ohne dass dies eine Konsequenz für einen anderen Datensatz in "Medien" hätte.

Der Filter ist in diesem Fall fest vorgegeben. Er befindet sich in dem Formular-Eigenschaften. Ein solcher Filter tritt direkt beim Start des Formulars automatisch in Kraft. Er kann ausgeschaltet und angewandt werden. Wurde er aber einmal gelöscht, so ist er nur dann wieder erreichbar, wenn das Formular komplett neu gestartet wurde. Das bedeutet, dass nicht nur die Daten zu aktualisieren sind, sondern das ganze Formulardokument erst geschlossen und dann wieder geöffnet werden muss.

Wartung: Tabellen auf unnötige Einträge überprüfen 486
Verwaiste Einträge durch Abfrage ermitteln

Der obige Filter ist Teil einer Abfrage, nach der die verwaisten Einträge ermittelt werden können.

```
SELECT "Nachname", "Vorname" FROM "Verfasser" WHERE "ID" NOT IN
(SELECT "Verfasser_ID" FROM "rel_Medien_Verfasser")
```

Bezieht sich eine Tabelle mit Fremdschlüsseln auf mehrere andere Tabellen, so ist die Abfrage entsprechend zu erweitern. Dies trifft z.B. auf die Tabelle "Ort" zu, die sowohl in der Tabelle "Medien" als auch in der Tabelle "Postleitzahl" Fremdschlüssel liegen hat. Daten aus der Tabelle "Ort", die gelöscht werden, sollten also möglichst in keiner dieser Tabellen noch benötigt werden. Dies zeigt die folgende Abfrage auf:

```
SELECT "Ort" FROM "Ort" WHERE "ID" NOT IN (SELECT "Ort_ID" FROM "Medien") AND "ID" NOT IN (SELECT "Ort_ID" FROM "Postleitzahl")
```

Verwaiste bzw. nicht benötigte Einträge können so durch die Markierung sämtlicher Einträge bei gesetztem Filter und Betätigung der rechten Maustaste über das Kontextmenü des Datensatzanzeigers gelöscht werden.

Datenbankgeschwindigkeit

Einfluss von Abfragen

Gerade Abfragen, die im vorherigen Abschnitt zur Filterung der Daten verwandt wurden, sollten allerdings im Hinblick auf die Geschwindigkeit einer Datenbank möglichst unterbleiben. Hier handelt es sich um Unterabfragen, die bei größeren Datenbanken eine entsprechend große Datenmenge für jeden einzelnen anzuzeigenden Datensatz zum Vergleich bereitstellen. Nur Vergleiche mit der Beziehung «IN» ermöglichen es überhaupt, einen Wert mit einer Menge von Werten zu vergleichen. In sofern kann

```
... WHERE "ID" NOT IN (SELECT "Verfasser_ID" FROM "rel_Medien_Verfasser")
```

eine große Menge an vorhandenen Fremdschlüsseln der Tabelle "rel_Medien_Verfasser" beinhalten, die erst mit dem Primärschlüssel der Tabelle "Verfasser" für jeden Datensatz der Tabelle "Ver-
...fassern" verglichen werden muss. So eine Abfrage sollte also nicht für den täglichen Gebrauch gedacht sein, sondern nur vorübergehend wie hier zum Beispiel der Wartung von Tabellen. Suchfunktionen sind anders aufzubauen, damit die Suche von Daten nicht endlos dauert und die Arbeit mit der Datenbank im Alltagsbetrieb verleidet.

Einfluss von Listenfeldern und Kombinationsfeldern

Je mehr Listenfelder in einem Formular eingebaut sind und je größer der Inhalt ist, den sie bereitstellen müssen, desto länger braucht das Formular zum Aufbau.

Je besser das Programm Base zuerst einmal die grafische Oberfläche bereitstellt und erst einmal Listenfelder nur teilweise einliest, desto weniger fällt eine entsprechende Last auf.

Listenfelder werden durch Abfragen erstellt, und diese Abfragen finden beim Formularstart für jedes Listenfeld statt.


Einfluss des verwendeten Datenbanksystems


Externe Datenbanken laufen hier deutlich schneller. Von der Geschwindigkeit sind die direkten Verbindungen zu MySQL oder PostgreSQL sowie die Verbindungen über ODBC nahezu gleichwertig. JDBC ist ebenfalls auf das Zusammenspiel mit Java angewiesen, funktioniert aber deutlich schneller als eine interne Verbindung zur HSQLDB.
Anhang
Barcode

Um die Barcode-Druckfunktion nutzen zu können, muss der Font «ean13.ttf» installiert sein. Dieser Font ist frei verfügbar.

EAN13-Barcodes können mittels «ean13.ttf» folgendermaßen erstellt werden:

<table>
<thead>
<tr>
<th>Zahl</th>
<th>Großbuchstaben, A=0 B=1 usw.</th>
<th>Kleinbuchstaben, a=0 b=1 usw.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>+</td>
</tr>
</tbody>
</table>

Siehe hierzu die Abfrage "Barcode_EAN13_ttf_Bericht" der Beispieldatenbank «Medien_ohne_Makros»

Datentypen des Tabelleneditors

### Ganzzahlen

<table>
<thead>
<tr>
<th>Typ</th>
<th>Zusatz</th>
<th>HSQLDB</th>
<th>Umfang</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny Integer</td>
<td>TINYINT</td>
<td>TINYINT</td>
<td>$2^8 = 256 \mid -128 \text{ bis } +127$</td>
<td>1 Byte</td>
</tr>
<tr>
<td>Small Integer</td>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>$2^{16} = 65536 \mid -32768 \text{ bis } +32767$</td>
<td>2 Byte</td>
</tr>
<tr>
<td>Integer</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>$2^{32} = 4294967296 \mid -2147483648 \text{ bis } +2147483647$</td>
<td>4 Byte</td>
</tr>
<tr>
<td>BigInt</td>
<td>BIGINT</td>
<td>BIGINT</td>
<td>$2^{64}$</td>
<td>8 Byte</td>
</tr>
</tbody>
</table>

### Fließkommazahlen

<table>
<thead>
<tr>
<th>Typ</th>
<th>Zusatz</th>
<th>HSQLDB</th>
<th>Umfang</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dezimal</td>
<td>DECIMAL</td>
<td>DECIMAL</td>
<td>Unbegrenzt, durch GUI auf 50 Stellen, einstellbar, feste Nachkommastellen, exakte Genauigkeit</td>
<td>variabel</td>
</tr>
<tr>
<td>Zahl</td>
<td>NUMERIC</td>
<td>NUMERIC</td>
<td>Unbegrenzt, durch GUI auf 50 Stellen, einstellbar, feste Nachkommastellen, exakte Genauigkeit</td>
<td>variabel</td>
</tr>
<tr>
<td>Float</td>
<td>FLOAT</td>
<td>(Double wird stattdessen genutzt)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double</td>
<td>DOUBLE</td>
<td>DOUBLE [PRECISION]</td>
<td>FLOAT</td>
<td>Einstellbar, nicht exakt, 15 Dezimalstellen maximal</td>
</tr>
</tbody>
</table>
### Text

<table>
<thead>
<tr>
<th>Typ</th>
<th>Zusatz</th>
<th>HSQLDB</th>
<th>Umfang</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>VARCHAR</td>
<td>VARCHAR</td>
<td>einstellbar</td>
<td>variabel</td>
</tr>
<tr>
<td>Text</td>
<td>VARCHAR _IG NORECASE</td>
<td>VARCHAR _IG NORECASE</td>
<td>Einstellbar, Auswirkung auf Sortierung, ignoriert Unterschiede zwischen Groß- und Klein-schreibung</td>
<td>variabel</td>
</tr>
<tr>
<td>Text (fix)</td>
<td>CHAR</td>
<td>CHAR</td>
<td>CHARACTER</td>
<td>Einstellbar, Rest zum tatsächlichen Text wird mit Leer-zeichen aufgefüllt</td>
</tr>
<tr>
<td>Memo</td>
<td>LONGVAR-CHAR</td>
<td>LONGVAR-CHAR</td>
<td>variabel</td>
<td></td>
</tr>
</tbody>
</table>

### Zeit

<table>
<thead>
<tr>
<th>Typ</th>
<th>Zusatz</th>
<th>HSQLDB</th>
<th>Umfang</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datum</td>
<td>DATE</td>
<td>DATE</td>
<td></td>
<td>4 Byte</td>
</tr>
<tr>
<td>Zeit</td>
<td>TIME</td>
<td>TIME</td>
<td></td>
<td>4 Byte</td>
</tr>
<tr>
<td>Datum/Zeit</td>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>DATETIME</td>
<td>Einstellbar (0, 6 – 6 bedeutet mit Millise-kunden)</td>
</tr>
</tbody>
</table>

### Sonstige

<table>
<thead>
<tr>
<th>Typ</th>
<th>Zusatz</th>
<th>HSQLDB</th>
<th>Umfang</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ja/Nein</td>
<td>BOOLEAN</td>
<td>BOOLEAN</td>
<td>BIT</td>
<td></td>
</tr>
<tr>
<td>Binärfeld (fix)</td>
<td>BINARY</td>
<td>BINARY</td>
<td>Wie Integer</td>
<td>fest</td>
</tr>
<tr>
<td>Binärfeld</td>
<td>VARBINARY</td>
<td>VARBINARY</td>
<td>Wie Integer</td>
<td>variabel</td>
</tr>
<tr>
<td>Bild</td>
<td>LONGVARBINARY</td>
<td>LONGVARBINARY</td>
<td>Wie Integer</td>
<td>variabel, für größere Bilder gedacht</td>
</tr>
<tr>
<td>OTHER</td>
<td>OTHER</td>
<td>OTHER</td>
<td>OBJECT</td>
<td></td>
</tr>
</tbody>
</table>

In den Tabellendefinitionen und bei der Änderung von Datentypen in Abfragen mit den Funktionen «CONVERT» oder «CAST» werden bei einigen Datentypen Informationen zur Anzahl an Zeichen (a), zur Genauigkeit (g, entspricht der Gesamtzahl an Ziffern) und zu den Dezimalstellen (d) erwartet: CHAR(a), VARCHAR(a), DOUBLE(g), NUMERIC(g,d), DECIMAL(g,d) und TIMESTAMP(g).

TIMESTAMP(g) kann nur die Werte ‘0’ oder ‘6’ annehmen. ‘0’ bedeutet, dass keine Sekunden im Nachkommabereich (Zehntel, Hundertstel …) gespeichert werden. Die Genauigkeit des Time-stamps kann nur direkt über den SQL-Befehl eingegeben werden. Sollen also Zeitangaben im Sportbereich eingetragen werden, so ist TIMESTAMP(6) über Extras → SQL voreinzustellen.
Datentypen in StarBasic

Zahlen

<table>
<thead>
<tr>
<th>Typ</th>
<th>Entspricht in HSQLDB</th>
<th>Startwert</th>
<th>Anmerkung</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>SMALLINT</td>
<td>0</td>
<td>$2^{16} = -32768$ bis $+32767$</td>
<td>2 Byte</td>
</tr>
<tr>
<td>Long</td>
<td>INTEGER</td>
<td>0</td>
<td>$2^{32} = -2147483648$ bis $+2147483647$</td>
<td>4 Byte</td>
</tr>
<tr>
<td>Single</td>
<td></td>
<td>0.0</td>
<td>Dezimaltrenner: «,»</td>
<td>4 Byte</td>
</tr>
<tr>
<td>Double</td>
<td>DOUBLE</td>
<td>0.0</td>
<td>Dezimaltrenner: «,»</td>
<td>8 Byte</td>
</tr>
<tr>
<td>Currency</td>
<td>Ähnlich DECIMAL, NUMERIC</td>
<td>0.0000</td>
<td>Währung, 4 Dezimalstellen fest</td>
<td>8 Byte</td>
</tr>
</tbody>
</table>

Sonstige

<table>
<thead>
<tr>
<th>Typ</th>
<th>Entspricht in HSQLDB</th>
<th>Startwert</th>
<th>Anmerkung</th>
<th>Speicherbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>BOOLEAN</td>
<td>False</td>
<td>1 = „ja“, alles andere: „nein“</td>
<td>1 Byte</td>
</tr>
<tr>
<td>Date</td>
<td>TIMESTAMP</td>
<td>00:00:00</td>
<td>Datum und Zeit</td>
<td>8 Byte</td>
</tr>
<tr>
<td>String</td>
<td>VARCHAR</td>
<td>Leerer String</td>
<td>bis 65536 Zeichen</td>
<td>variabel</td>
</tr>
<tr>
<td>Object</td>
<td>OTHER</td>
<td>Null</td>
<td>variabel</td>
<td>variabel</td>
</tr>
<tr>
<td>Variant</td>
<td></td>
<td>Leer</td>
<td>Kann jeden (anderen) Datentyp annehmen</td>
<td>variabel</td>
</tr>
</tbody>
</table>

Vor allem bei Zahlenwerten besteht große Verwechselungsgefahr. In der Datenbank steht z.B. häufig im Primärschlüssel der Datentyp «INTEGER». Wird jetzt per Makro ausgelesen, so muss dort die aufnehmende Variable den Typ «Long» haben, da diese vom Umfang her mit «INTEGER» aus der Datenbank übereinstimmt. Der entsprechende Auslesebefehl heißt dann auch «getLong».

Eingebaute Funktionen und abgespeicherte Prozeduren

In der eingebauten HSQLDB sind die folgenden Funktionen verfügbar. Ein paar Funktionen können leider nur dann genutzt werden, wenn in der Abfrage «SQL-Kommando direkt ausführen» gewählt wurde. Dies verhindert dann gleichzeitig, dass die Abfragen editierbar bleiben.

Funktionen, die mit der grafischen Benutzeroberfläche zusammenarbeiten, sind gekennzeichnet mit [funktioniert mit der GUI]. Funktionen, die nur über «SQL-Kommando direkt ausführen» ansprechbar sind, sind gekennzeichnet mit [SQL direkt – funktioniert nicht mit der GUI].

Anhang: Eingebaute Funktionen und abgespeicherte Prozeduren  492
**Numerisch**

Da hier mit Fließkommazahlen gerechnet wird empfiehlt es sich gegebenenfalls auf die Einstellung der Felder bei einer Abfrage zu achten. Meist ist hier die Anzeige der Nachkommazahlen begrenzt, so dass eventuell überraschende Ergebnisse zustande kommen. Dann wird z.B. in Spalte1 0,00, in Spalte2 1000 angezeigt. In Spalte3 soll dann Spalte1 * Spalte2 stehen – dort steht plötzlich 1.

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS(d)</td>
<td>Gibt den absoluten Wert einer Zahl wieder, entfernt also ggf. das Minus-Vorzeichen.</td>
</tr>
<tr>
<td>ACOS(d)</td>
<td>Gibt den Arcuscosinus wieder.</td>
</tr>
<tr>
<td>ASIN(d)</td>
<td>Gibt den Arcussinus wieder.</td>
</tr>
<tr>
<td>ATAN(d)</td>
<td>Gibt den Arcustangens wieder.</td>
</tr>
<tr>
<td>ATAN2(a,b)</td>
<td>Gibt den Arcustangens über Koordinaten wieder. 'a' ist der Wert der x-Achse, 'b' der Wert der y-Achse</td>
</tr>
<tr>
<td>BITAND(a,b)</td>
<td>Sowohl die binäre Schreibweise von 'a' als auch die binäre Schreibweise von 'b' müssen an der gleichen Stelle eine '1' stehen haben, damit die '1' in das Ergebnis übernommen wird.</td>
</tr>
<tr>
<td>BITOR(a,b)</td>
<td>Entweder die binäre Schreibweise von 'a' oder die binäre Schreibweise von 'b' müssen an der gleichen Stelle eine '1' stehen haben, damit die '1' in das Ergebnis übernommen wird.</td>
</tr>
<tr>
<td>CEILING(d)</td>
<td>Gibt die kleinste Ganzzahl an, die nicht kleiner als d ist.</td>
</tr>
<tr>
<td>COS(d)</td>
<td>Gibt den Cosinus wieder.</td>
</tr>
<tr>
<td>COT(d)</td>
<td>Gibt den Cotangens wieder.</td>
</tr>
<tr>
<td>DEGREES(d)</td>
<td>Gibt zu Bogenmaßen die Gradzahl wieder.</td>
</tr>
<tr>
<td>EXP(d)</td>
<td>Gibt ed ( e: (2.718...) ) wieder.</td>
</tr>
<tr>
<td>FLOOR(d)</td>
<td>Gibt die größte Ganzzahl an, die nicht größer als d ist.</td>
</tr>
<tr>
<td>LOG(d)</td>
<td>Gibt den natürlichen Logarithmus zur Basis 'e' wieder.</td>
</tr>
<tr>
<td>LOG10(d)</td>
<td>Gibt den Logarithmus zur Basis 10 wieder.</td>
</tr>
</tbody>
</table>

**Anhang: Eingebaute Funktionen und abgespeicherte Prozeduren** 493
<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
<th>Beispiel</th>
<th>Ergebnis/Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOD(a,b)</td>
<td>Gibt den Rest als Ganzzahl wieder, der bei der Division von 2 Ganzzahlen entsteht.</td>
<td></td>
<td><strong>MOD(11, 3)</strong> ergibt 2, weil 3*3+2=11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>PI()</td>
<td>Gibt π (3.1415...) wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>POWER(a,b)</td>
<td><strong>POWER(2, 3)</strong> = 8, weil 2³ = 8</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>RADIANS(d)</td>
<td>Gibt zu den Gradzahlen das Bogenmaß wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>RAND()</td>
<td>Gibt eine Zufallszahl x größer oder gleich 0.0 und kleiner als 1.0 wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>ROUND(a,b)</td>
<td>Rundet a auf b Stellen nach dem Dezimalzeichen.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>ROUNDMAGIC(d)</td>
<td>Löst Rundungsprobleme, die durch Fließkommazahlen entstehen.</td>
<td></td>
<td>3.11-3.1-0.01 ist eventuell nicht genau 0, wird aber als 0 in der GUI angezeigt. ROUNDMAGIC macht daraus einen tatsächlichen 0-Wert. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>SIGN(d)</td>
<td>Gibt -1 wieder, wenn 'd' kleiner als 0 ist, 0 wenn 'd'==0 und 1 wenn 'd' größer als 0 ist.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>SIN(A)</td>
<td>Gibt den Sinus eines Bogenmaßes wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>SQRT(d)</td>
<td>Gibt die Quadratwurzel wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>TAN(A)</td>
<td>Gibt den Tangens eines Bogenmaßes wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>TRUNCATE(a,b)</td>
<td>Schneidet 'a' auf 'b' Zeichen nach dem Dezimalpunkt ab.</td>
<td>TRUNCATE(2.37456,2) = 2.37</td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>ASCII(s)</td>
<td>Gibt den ASCII-Code des ersten Buchstaben des Strings wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>BIT_LENGTH(str)</td>
<td>Gibt die Länge des Textes str in Bits wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>CHAR(c)</td>
<td>Gibt den Buchstaben wieder, der zu dem ASCII-Code c gehört. Dabei geht es nicht nur um Buchstaben, sondern auch um Steuerzeichen. <strong>CHAR(13)</strong> erzeugt in einer Abfrage einen Zeilenumbruch, der in mehrzeiligen Feldern eines Formulars oder in Berichten sichtbar wird. [funktioniert in der GUI]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR_LENGTH(str)</td>
<td>Gibt die Länge des Textes in Buchstaben wieder.</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
<tr>
<td>CONCAT(str1,str2)</td>
<td>Verbindet str1 + str2</td>
<td></td>
<td>[funktioniert in der GUI]</td>
</tr>
</tbody>
</table>

**Text**

ASCII(s)          | Gibt den ASCII-Code des ersten Buchstaben des Strings wieder.                                       |          | [funktioniert in der GUI]              |

---

Anhang: Eingebaute Funktionen und abgespeicherte Prozeduren
<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbindet str1 + str2 + str3</td>
<td>einfachere Alternative zu CONCAT [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DIFFERENCE(s1,s2)</td>
<td>Gibt den ?Klang?unterschied zwischen s1 und s2 wieder. Hier wird lediglich eine Ganzzahl ausgegeben. 0 bedeutet dabei gleichen Klang. So erscheint 'for' und 'four' mit 0 gleich, Kürzen und Würzen wird auf 1 gesetzt, Mund und Mond wieder auf 0 [funktioniert in der GUI]</td>
</tr>
<tr>
<td>HEXTORAW(s1)</td>
<td>Übersetzt Hexadezimalcode in andere Zeichen [funktioniert in der GUI]</td>
</tr>
<tr>
<td>INSERT(s,start,len,s2)</td>
<td>Gibt einen Text wieder, bei dem Teile ersetzt werden. Beginnend mit «start» wird über eine Länge «len» aus dem Text s Text ausgeschnitten und durch den Text s2 ersetzt. <strong>INSERT( 'Bundesbahn', 3, 4, 'mmel')</strong> macht aus 'Bundesbahn' 'Bummelbahn', wobei die Länge des eingefügten Textes auch ohne weiteres größer als die des ausgeschnittenen Textes sein darf. So ergibt <strong>INSERT( 'Bundesbahn', 3, 5, 's und B')</strong> 'Bus und Bahn'. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>LCASE(s)</td>
<td>Wandelt den String in Kleinbuchstaben um. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>LEFT(s,count)</td>
<td>Gibt die mit count angegebene Zeichenanzahl vom Beginn des Textes s wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>LENGTH(s)</td>
<td>Gibt die Länge eines Textes in Anzahl der Buchstaben wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>LOCATE(search,s,[start])</td>
<td>Gibt den ersten Treffer für den Begriff aus search in dem Text s wieder. Der Treffer wird numerisch angegeben: (1=left, 0=not found) Die Angabe eines Startes innerhalb des Textes ist optional. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>LTRIM(s)</td>
<td>Entfernt führende Leerzeichen und nicht druckbare Zeichen von einem Text. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>OCTET_LENGTH(str)</td>
<td>Gibt die Länge eines Textes in Bytes an. Dies entspricht im Prinzip dem doppelten Wert der Zeichenanzahl. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>RAWTOHEX(s1)</td>
<td>Verwandelt in die Hexadezimalschreibweise, Umkehr von HEXTORAW() [funktioniert in der GUI]</td>
</tr>
<tr>
<td>REPEAT(s,count)</td>
<td>Wiederholt den Text s count Mal [funktioniert in der GUI]</td>
</tr>
<tr>
<td>REPLACE(s,replace,s2)</td>
<td>Ersetzt alle vorkommenden Textstücke mit dem Inhalt replace im Text s durch den Text s2 [funktioniert in der GUI]</td>
</tr>
<tr>
<td>RIGHT(s,count)</td>
<td>Umgekehrt zu LEFT; gibt die mit count angegebene Zeichenzahl vom Textende aus wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>Funktion</td>
<td>Beschreibung</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RTRIM(s)</td>
<td>Entfernt alle Leerzeichen und nicht druckbaren Zeichen am Textende. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>SOUNDEX(s)</td>
<td>Gibt einen Code von 4 Zeichen wieder, die dem Klang von s entsprechen sollen – passt zu der Funktion DIFFERENCE() [funktioniert in der GUI]</td>
</tr>
<tr>
<td>SPACE(count)</td>
<td>Gibt die in count angegebene Zahl an Leertasten wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>SUBSTR(s,start[,len])</td>
<td>Kürzel für SUBSTRING [funktioniert in der GUI]</td>
</tr>
<tr>
<td>SUBSTRING(s,start[,len])</td>
<td>Gibt den Text s ab der Startposition wieder. (1=links). Wird die Länge ausgelassen, so wird der gesamte Text wiedergegeben. Auch bei einer größeren Länge als der Textlänge wird der restliche Text wiedergegeben. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>UCASE(s)</td>
<td>Wandelt den String in Großbuchstaben um. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>LOWER(s)</td>
<td>Wie LCASE(s) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>UPPER(s)</td>
<td>Wie UCASE(s) [funktioniert in der GUI]</td>
</tr>
<tr>
<td><strong>Datum/Zeit</strong></td>
<td></td>
</tr>
<tr>
<td>CURDATE()</td>
<td>Gibt das aktuelle Datum wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>CURTIME()</td>
<td>Gibt die aktuelle Zeit wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DATEDIFF(string, datetime1, datetime2)</td>
<td>Datumsunterschied zwischen zwei Datums- bzw. Datumszeitangaben. Der Eintrag in string entscheidet darüber, in welcher Einheit der Unterschied wiedergegeben wird: 'ms'='millisecond', 'ss'='second', 'mi'='minute', 'hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'. Sowohl die Langfassung als auch die Kurzfassung ist für den einzusetzenden string möglich. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DAY(date)</td>
<td>Gibt den Tag im Monat wieder. (1-31) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DAYNAME(date)</td>
<td>Gibt den englischen Namen des Tages wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DAYOFMONTH(date)</td>
<td>Gibt den Tag im Monat wieder. (1-31), Synonym für DAY() [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DAYOFWEEK(date)</td>
<td>Gibt den Wochentag als Zahl wieder. (1 bedeutet Sonntag) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>DAYOFYEAR(date)</td>
<td>Gibt den Tag im Jahr wieder. (1-366) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>HOUR(time)</td>
<td>Gibt die Stunde wieder. (0-23) [funktioniert in der GUI]</td>
</tr>
</tbody>
</table>

Anhang: Eingebaute Funktionen und abgespeicherte Prozeduren
<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINUTE(time)</td>
<td>Gibt die Minute wieder. (0-59) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>MONTH(date)</td>
<td>Gibt den Monat wieder. (1-12) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>MONTHNAME(date)</td>
<td>Gibt den englischen Namen des Monats wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>NOW()</td>
<td>Gibt das aktuelle Datum und die aktuelle Zeit zusammen als Zeitstempel wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>QUARTER(date)</td>
<td>Gibt das Quartal im Jahr wieder. (1-4) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>SECOND(time)</td>
<td>Gibt die Sekunden einer Zeitangabe wieder. (0-59) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>WEEK(date)</td>
<td>Gibt die Woche des Jahres wieder. (1-53) [funktioniert in der GUI]</td>
</tr>
<tr>
<td>YEAR(date)</td>
<td>Gibt das Jahr aus einer Datumseingabe wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>CURRENT_DATE</td>
<td>Synonym für CURDATE(), SQL-Standard [funktioniert in der GUI]</td>
</tr>
<tr>
<td>CURRENT_TIME</td>
<td>Synonym für CURTIME(), SQL-Standard [funktioniert in der GUI]</td>
</tr>
<tr>
<td>CURRENT_TIMESTAMP</td>
<td>Synonym für NOW(), SQL-Standard [funktioniert in der GUI]</td>
</tr>
</tbody>
</table>

**Datenbankverbindung**

<table>
<thead>
<tr>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE()</td>
<td>Gibt den Pfad und Namen der Datenbank, die zu dieser Verbindung gehört, wieder. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>USER()</td>
<td>Gibt den Benutzernamen dieser Verbindung wieder. Der Nutzernamen ist dann von Bedeutung, wenn die Datenbank in eine externe Datenbank umgewandelt werden soll. [SQL direkt – funktioniert nicht mit der GUI]</td>
</tr>
<tr>
<td>CURRENT_USER</td>
<td>SQL Standardfunktion, Synonym für USER(). Zu beachten ist, dass hier keine Klammern zu setzen sind. [funktioniert in der GUI]</td>
</tr>
<tr>
<td>IDENTITY()</td>
<td>Gibt den letzten Wert für ein Autowertfeld wieder, das in der aktuellen Verbindung erzeugt wurde. Dies wird bei der Makroprogrammierung genutzt, um aus einem erstellten Primärschlüssel für eine Tabelle einen Fremdschlüssel für eine andere Tabelle zu erstellen. [funktioniert in der GUI]</td>
</tr>
</tbody>
</table>
### System

**IFNULL(exp,value)**


*[funktioniert in der GUI]*

**CASEWHEN(expr1,v1,v2)**

Wenn `expr1` wahr ist wird `v1` zurückgegeben, sonst `v2`. Stattdessen kann auch `CASE WHEN` genutzt werden. `CASEWHEN("a">10,'Ziel erreicht','noch üben')` gibt 'Ziel erreicht' aus, wenn der Inhalt des Feldes "a" größer als 10 ist.

*[funktioniert in der GUI]*

**CONVERT(term,type)**

Wandelt `term` in einen anderen Datentyp um. `CONVERT("a",DECIMAL(5,2))` macht aus dem Feld "a" ein Feld mit 5 Ziffern, davon 2 Nachkommastellen. Ist die Zahl zu groß, so wird ein Fehler ausgegeben.

*[funktioniert in der GUI]*

**CAST(term AS type)**

Synonym zu `CONVERT()`

*[funktioniert in der GUI]*

**COALESCE(expr1,expr2, expr3,...)**

Wenn `expr1` nicht NULL ist wird `expr1` wiedergegeben, ansonsten wird `expr2` überprüft, danach dann `expr3` usw. Sämtliche Ausdrücke müssen zumindest einen ähnlichen Datentyp haben. So geht die alternative Darstellung von Ganzzahlen und Fließkommazahlen, aber nicht auch noch des eines Datums- oder Zeitwertes.

*[funktioniert in der GUI]*

**NULLIF(v1,v2)**

Wenn `v1` gleich `v2` ist wird NULL wiedergegeben, ansonsten `v1`. Die Daten müssen vom Typ her vergleichbar sein.

*[funktioniert in der GUI]*

**CASE v1 WHEN v2 THEN v3 [ELSE v4] END**

Wenn `v1` gleich `v2` ist wird `v3` wiedergegeben. Sonst wird `v4` wiedergegeben oder NULL, wenn kein ELSE formuliert ist.

*[SQL direkt – funktioniert nicht mit der GUI]*

**CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END**

Wenn `expr1` wahr ist wird `v1` zurückgegeben. [Optional können weitere Fälle angegeben werden] Sonst wird `v4` wiedergegeben oder NULL, wenn kein ELSE formuliert ist.

`CASE WHEN DAYOFWEEK("Datum")=1 THEN 'Sonntag' WHEN DAYOFWEEK("Datum")=2 THEN 'Montag' ... END` könnte per SQL den Tagesnamen ausgeben, der sonst in der Funktion nur in Englisch verfügbar ist.

*[funktioniert in der GUI]*

**EXTRACT ({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM <Datums- oder Zeitwert>)**


*[funktioniert in der GUI]*

---

**Anhang: Eingebaute Funktionen und abgespeicherte Prozeduren** 498
Wenn der erste Text in dem zweiten enthalten ist wird die Position des ersten Textes wiedergeben, ansonsten 0. Dies könnte statt einer Suchmöglichkeit mit «LIKE» genutzt werden. [funktioniert in der GUI]

Liefert den Teil eines Textes ab der in FROM angegebenen Startposition, optional in der in FOR angegebenen Länge. Steht im Feld “Name” z.B. ‘Roberta’, so ergibt SUBSTRING("Name" FROM 3 FOR 3) den Teilstring ‘bert’. [funktioniert in der GUI]

Nicht druckbare Sonderzeichen und Leerzeichen werden entfernt. [funktioniert in der GUI]

### Steuerzeichen zur Nutzung in Abfragen

In Abfragen lassen sich Felder miteinander verknüpfen. Aus zwei Feldern in

```
SELECT "Vorname", "Nachname" FROM "Tabelle"
```

wird durch

```
SELECT "Vorname"||' '||"Nachname" FROM "Tabelle"
```


<table>
<thead>
<tr>
<th>CHAR( 9 )</th>
<th>Horizontaler Tabulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(10)</td>
<td>Zeilenvorschub</td>
</tr>
<tr>
<td>CHAR(13)</td>
<td>Wagenrücklauf</td>
</tr>
</tbody>
</table>

Erzeugt in Serienbriefen und im Report-Builder in einem Feld einen Zeilenumbruch (Linux, Unix, Mac).

Zeilenumbruch zusammen mit dem Zeilenvorschub in Windows. CHAR(13) || CHAR(10) ist auch für Linux, Mac usw. möglich, daher die universelle Variante.

### Einige uno-Befehle zur Nutzung mit einer Schaltfläche


<table>
<thead>
<tr>
<th>Uno-Befehl</th>
<th>Anwendung für ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>.uno:RecSearch</td>
<td>Öffnen der Suchfunktion im Formular</td>
</tr>
<tr>
<td>.uno:Paste</td>
<td>Zwischenablage einfügen, nur mit <strong>Fokussieren bei Klick</strong> → <strong>Nein</strong></td>
</tr>
<tr>
<td>.uno:Copy</td>
<td>Kopiert den markierten Inhalt in die Zwischenablage, nur mit <strong>Fokussieren bei Klick</strong> → <strong>Nein</strong></td>
</tr>
<tr>
<td>.uno:Print</td>
<td>Druckdialog für das Formular öffnen</td>
</tr>
<tr>
<td>.uno:PrintDefault</td>
<td>Drucken mit dem Defaultprinter ohne Dialog</td>
</tr>
</tbody>
</table>

**Informationstabellen der HSQLDB**


In einer Abfrage können die einzelnen Informationen sowie sämtliche dazugehörigen Felder auf die folgende Art ermittelt werden.

```sql
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_ALIASES"
```

Im Gegensatz zu einer normalen Tabelle ist es hier notwendig, dem jeweiligen folgenden Begriff "**INFORMATION_SCHEMA**" voranzustellen.

```
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
```

Anhang: Informationstabellen der HSQLDB

500
Die folgende Abfrage gibt z.B. eine komplette Übersicht über alle in der Datenbank genutzten Tabellen mit Feldtypen, Primärschlüsseln und Fremdschlüsseln:

```sql
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'
```

**Datenbankreparatur für *.odb-Dateien**


Formulare, Abfragen und Berichte können, sofern eine Vorversion der Datenbank gesichert wurde, über die Zwischenablage in eine neue Datenbank kopiert werden. Lässt sich allerdings, aus welchen Gründen auch immer, eine aktuelle Datenbankdatei nicht mehr öffnen, so ist das Hauptproblem: Wie komme ich (hoffentlich) an die Daten.

Bei plötzlichen Abstürzen des PC kann es passieren, dass geöffnete Datenbanken von LO (interne Datenbank HSQLDB) nicht mehr zu öffnen sind. Stattdessen wird beim Versuch, die Datenbank zu öffnen, nach einem entsprechenden Filter für das Format gefragt.

Das Ganze liegt daran, dass Teile der Daten der geöffneten Datenbank im Arbeitsspeicher liegen und lediglich temporär zwischenabgespeichert werden. Erst beim Schließen der Datei wird die gesamte Datenbank in die Datei zurückgeschrieben und gepackt.

**Wiederherstellung der Datenbank-Archivdatei**

Um eventuell doch noch an die Daten zu kommen, kann das folgende Verfahren hilfreich sein:

---

Anhang: Datenbankreparatur für *.odb-Dateien

501
1. Fertigen sie eine Kopie ihrer Datenbank für die weiteren Schritte an.
   Funktioniert das Öffnen nicht, so ist vermutlich von der Datenbank nichts mehr zu retten.
3. Folgende Verzeichnisse sehen sie nach dem Öffnen einer Datenbankdatei im Packprogramm auf jeden Fall:

Abbildung 59: Aufbau einer Datenbankdatei ohne Tabellen, Abfragen, Formulare und Berichte

4. Die Datenbankdatei muss ausgepackt werden. Die entscheidenden Informationen für die Daten liegen im Unterverzeichnis «database» in den Dateien «data» und «script».
5. Gegebenenfalls empfiehlt es sich, die Datei «script» einmal anzuschauen und auf Unge-reimtheiten zu überprüfen. Dieser Schritt kann aber auch erst einmal zum Testen über-sprungen werden. Die «script»-Datei enthält vor allem die Beschreibung der Tabellenstruk-
7. Ersetzen sie die Dateien «data» und «script» aus der neuen Datenbankdatei durch die unter «4.» entpackten Dateien.
8. Das Packprogramm muss nun geschlossen werden. War es, je nach Betriebssystem, not-
    wendig, die Dateien vor dem Öffnen durch das Packprogramm nach *.zip umzubenennen,
   so ist das jetzt wieder nach *.odb zu wandeln.
9. Öffnen sie die Datenbankdatei jetzt mit LO. Sie können hoffentlich wieder auf ihre Tabellen zugreifen.
10. Wie weit sich jetzt auch Abfragen, Formulare und Berichte auf ähnliche Weise wiederher-
    stellen lassen, bleibt dem weiteren Testen überlassen.

Weitere Informationen zur Datenbank-Archivdatei

Eine Datenbank-Archivdatei enthält im tatsächlichen Gebrauch neben dem grundlegenden Ver-
   zeichnis für die Datenbank und dem für das OpenDocument-Format vorgeschriebenen Verzeichnis
   «META-INF» noch weitere Verzeichnisse, um Formulare und Berichte abzuspeichern. Eine
   Beschreibung zum grundsätzliche Aufbau des OpenDocument-Formates ist u.a. unter http://de.wi-

Anhang: Datenbankreparatur für *.odb-Dateien

Abbildung 60: Datenbankdatei, die neben der Datenbank auch Informationen zu einem Formular und einem Bericht abgespeichert hat.

Hier einige der Dateien aus der Datenbank-Archivdatei im Überblick:

**mimetype**

application/vnd.oasis.opendocument.base

Diese kleine Textdatei enthält lediglich den Hinweis, dass es sich bei der Archivdatei um eine Datenbankdatei im OpenDocument-Format handelt.

**content.xml einer Datenbank ohne Inhalt**

```xml
<?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:draw:1.0"
    xmlns:xsl="urn:oasis:names:tc:opendocument:xmlns:xsl: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: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"
```

Anhang: Datenbankreparatur für *.odb-Dateien

503

Mit den Anfangsdefinitionen werden durch «xmlns» (XML-Namespace) die Namensräume umschrieben, auf die innerhalb dieser Datei zugegriffen werden kann. Anschließend werden etwas konkretere Angaben zum Inhalt gemacht. Hier ist dann z.B. ersichtlich, dass es sich um eine interne HSQLDB-Datenbank handelt und die Angabe eines Passwortes nicht erforderlich ist.

### content.xml einer Datenbank mit Inhalt
Der folgende Inhalt ist nur ein Auszug der content.xml-Datei und soll nur die Struktur klären.

```xml
<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:driver-settings>
      <db:application-connection-settings>
        <db:is-table-name-length-limited="false"/>
        <db:append-table-alias-name="false"/>
        <db:max-row-count="100"/>
      </db:application-connection-settings>
      <db:table-filter>
        <db:table-include-filter>
          <db:table-filter-pattern>%</db:table-filter-pattern>
        </db:table-filter>
      </db:application-connection-settings>
      <office:database/>
    </office:database>
  </office:body>
</office:document-content>
```

Anhang: Datenbankreparatur für *.odb-Dateien
Hier wird ein Feld als Währungsfeld festgelegt. Die Anzahl der Dezimalstellen werden genannt, der Abstand zwischen Zahlen und Währungssymbol sowie das Währungssymbol selbst.

Im zweiten Abschnitt erfolgt die Festlegung, dass bis zu einem bestimmten Wert die Währung in der Farbe Rot («#ff0000») erscheinen soll.

Anhang: Datenbankreparatur für *.odb-Dateien
Die Datenbank-Archivdatei enthält einen Unterordner, in dem die Details zu einem Bericht abgespeichert sind. Der Bericht ist auf der Benutzeroberfläche ebenfalls mit dem Namen «Quittung» verzeichnet.

Sämtliche Abfragen werden direkt in der content.xml gespeichert. «« steht dabei für ein doppeltes Anführungszeichen oben «»). Die oben stehende Abfrage ist in diesem Beispiel eigentlich recht umfangreich und besteht aus vielen korrelierenden Unterabfragen. Sie ist hier nur verkürzt wiedergegeben.

Wie sollen verschiedene Tabellen von der Ansicht her erscheinen. An dieser Stelle wird das Erscheinungsbild bestimmter Spalten gespeichert; in diesem Beispiel wurden Einstellungen der Tabelle "Ware" mit ihren einzelnen Feldern "ID", "MWSt" usw. abgespeichert. Anscheinend wurde hier direkt etwas eingegeben und dabei an den Spalten der Tabelle ein bisschen verändert.


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"
Bei einer Datenbank ohne weiteren Inhalt stehen hier nur die Grunddefinitionen. Mit Inhalt werden hier aber die unterschiedlichsten Einstellungen abgespeichert. Nach dem Start mit der obigen Definition sind folgende Einstellungen in der Beispieldatei abgespeichert:

```xml
<office:settings>
  <config:config-item-set config:name="ooo:view-settings">
    <config:config-item-set config:name="Queries">
      <config:config-item-set config:name="Verkauf_berechnet">
        <config:config-item config:name="WindowTitle" 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-set>
  <config:config-item-set config:name="ooo:configuration-settings">
    <config:config-item-set config:name="layout-settings">
      <config:config-item-set config:name="Table1">
        <config:config-item config:name="WindowTitle" 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">Verkauf</config:config-item>
      </config:config-item-set>
    </config:config-item-set>
    <config:config-item-set config:name="Table2">
      ... (identische config:type-Punkte wie "Table1"
    </config:config-item-set>
    <config:config-item-set config:name="Table3">
      ... (identische config:type-Punkte wie "Table1"
  </config:config-item-set>
</office:settings>
```

Anhang: Datenbankreparatur für *.odb-Dateien
Die gesamte Übersicht bezieht sich auf verschiedene Ansichten der Fenster für die (eine) Abfrage "Verkauf berechnet" und für die Tabellen "Verkauf", "Ware" und "Quittung". Die letzten beiden wurden hier nur verkürzt wiedergegeben. Würden diese Einstellungen bei einer defekten *.odb-Datei fehlen, so wäre das also nicht weiter von Bedeutung. Sie würden wieder erstellt, wenn die entsprechenden Fenster das nächste Mal geöffnet werden.

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>

Bei dieser Datei im Unterverzeichnis META-INF handelt es sich um ein Inhaltsverzeichnis der gesamten Datenbank-Archivdatei. Da es sich bei der oben gezeigten Datei um die Datei handelt, die in der leeren Datenbank (Abbildung 59) enthalten ist gibt es hier nur 5 Datei-Einträge («file-entry»). Bei der mit Formular und Bericht versehenen Datenbank-Archivdatei sind die Einträge in der META-INF entsprechend umfangreicher.

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

Die properties-Datei enthält die Grundeinstellungen für die interne HSQL Datenbank. Siehe dazu auch das folgende Kapitel.

database/script
   SET DATABASE COLLATION "German"

Anhang: Datenbankreparatur für *.odb-Dateien
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60

In der script-Datei finden sich die Standardeinstellungen für die Verbindung zur Datenbank, zur benutzten Sprache usw. Hier erscheint auch der später erwähnte Benutzer «SA».

In deiner mit Inhalt gefüllten Datenbank werden in dieser Datei die Grundlagen für die Tabellendefinitionen gespeichert:

SET DATABASE COLLATION "German"
CREATE SCHEMA PUBLIC AUTHORIZATION DBA

Die Tabellen werden definiert, bevor der Datenspeicher definiert wird. Zuerst werden die Tabellen mit ihren Feldern im Cache erstellt.

CREATE CACHED TABLE "Ware"
("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL PRIMARY KEY,"Ware" VARCHAR(50),"Preis" DECIMAL(8,2),"MwSt" TINYINT)
CREATE CACHED TABLE "Verkauf"
("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL PRIMARY KEY,"Anzahl" TINYINT,"Ware_ID" INTEGER,"Quittung_ID" INTEGER, CONSTRAINT SYS_FK_59 FOREIGN KEY("Ware_ID") REFERENCES "Ware"("ID")
CREATE CACHED TABLE "Quittung"
("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL PRIMARY KEY,"Datum" DATE)

Anschließend werden noch Änderungen an den Tabellen vorgenommen, damit die Beziehungen («REFERENCES») stimmig sind

ALTER TABLE "Verkauf" ADD CONSTRAINT SYS_FK_76 FOREIGN KEY("Quittung_ID") REFERENCES "Quittung"("ID")
SET TABLE "Ware" INDEX'608 20'
SET TABLE "Verkauf" INDEX'1872 1656 1872 12'
SET TABLE "Quittung" INDEX'2232 1'

Nach der Einstellung der Position des Indexes in der data-Datei (erscheint nur hier in der script-Datei, wird nie direkt in SQL eingegeben!) werden die automatisch hoch schreibenden Felder der Tabellen («AutoWert») so eingestellt, dass sie die nächsten Werte bei Neueingaben erstellen. So ist z.B. der letzte eingetragene Wert im Feld "ID" der Tabelle "Ware" die Nummer 19. Das automatische Hochschreiben beginnt also mit der Nummer 20.

ALTER TABLE "Ware" ALTER COLUMN "ID" RESTART WITH 20
ALTER TABLE "Verkauf" ALTER COLUMN "ID" RESTART WITH 12
ALTER TABLE "Quittung" ALTER COLUMN "ID" RESTART WITH 1
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60

Behebung von Versionsproblemen


Lassen sich interne Datenbanken nicht mehr öffnen so hilft pragmatisch erst einmal nur, LO ab der Version 3.5 zu nutzen. Ansonsten muss als externe Datenbank die mitgelieferte hsqldb.jar-Datei genutzt werden. Außerdem muss aus der *.odb-Datei das database-Verzeichnis extrahiert werden. Die Datei properties hat hier einen Eintrag, der in LO 3.3 zu dieser Fehlermeldung führt:

version=1.8.1

steht in Zeile 11.

Anhang: Datenbankreparatur für *.odb-Dateien
Diese Zeile ist zu ändern auf

version=1.8.0

Danach ist das database-Verzeichnis wieder in das *.odb-Päckchen einzulesen und die Datenbank lässt sich auch wieder unter LO 3.3 öffnen.

**Weitere Tipps**

Wenn aus irgendwelchen Gründen wohl die Datenbankdatei geöffnet wird, aber kein Zugang mehr zu den Tabellen existiert, kann direkt über **Extras → SQL** der Befehl **SHUTDOWN SCRIPT** eingegeben werden. Anschließend wird die Datenbank geschlossen und neu gestartet. Das Ganze funktionierte aber nicht, wenn bereits ein «Error im Script-file» gemeldet wird.

det wurde. Jetzt wird aus der komprimierten «backup»-Datei beim Neustart eine neue «data»-Datei erstellt.

**Datenbankverbindung zu einer externen HSQldb**

Die interne HSQldb unterscheidet sich erst einmal nicht von der externen Variante. Wenn, wie im Folgenden beschrieben, erst einmal nur der Zugriff auf die Datenbank nach außerhalb gelegt werden soll, dann ist keine Serverfunktion erforderlich. Hier reicht schlicht das Archiv, was in LO mitgelieft wurde. Im LO-Pfad liegt unter /program/classes/hsqldb.jar. Die Verwendung dieses Archivs ist die sicherste Variante, da dann keine Versionsprobleme auftauchen.


---

[Abbildung: Optionen - LibreOffice - Erweitert]

Anhang: Datenbankverbindung zu einer externen HSQldb
Der Datenbanktreiber muss, sofern er nicht in dem Pfad der Java-Runtime liegt, als ClassPath unter Extras – Optionen – Java hinzugefügt werden.


Wichtig, damit auch Daten in die Datenbank über die GUI geschrieben werden können, muss: ergänzend neben dem Datenbanknamen «;default_schema=true» stehen. Dies kann noch durch ein «;shutdown=true» ergänzt werden, damit die DB nach dem Schließen von LO heruntergefahren wird.

Also:

```java
jdbc:hsqldb:file:/home/PfadZurDatenbank/Datenbankname;default_schema=true;
shutdown=true
```

In dem Ordner befinden sich die Dateien

- Datenbankname.backup
- Datenbankname.data
- Datenbankname.properties
- Datenbankname.script
- Datenbankname.log

Weiter geht es mit der Angabe des Standardnutzers, sofern nichts an der HSQLDB-Konfiguration geändert wurde:

**Schritte**

1. Datenbank auswählen
2. JDBC-Verbindung einrichten
3. Benutzer-Authentifizierung einrichten
4. Fertigstellen und fortfahren

**Datenbankverbindung per JDBC einrichten**

Bitte geben Sie die benötigten Informationen ein, um per JDBC eine Verbindung zu einer Datenquelle herzustellen. Fragen Sie Ihren Systemadministrator, wenn Sie sich bei den folgenden Einstellungen unsicher sind.

**URL der Datenquelle**

```java
jdbc:hsqldb:file:/home/PfadZurDatenbank/Datenbankname;default_schema=true;shutdown=true
```

**JDBC Treiberklasse**

- org.hsqldb.jdbcDriver

Weiter geht es mit der Angabe des Standardnutzers, sofern nichts an der HSQLDB-Konfiguration geändert wurde:

Anhang: Datenbankverbindung zu einer externen HSQLDB
Damit ist die Verbindung erstellt und es kann auf die Datenbank zugegriffen werden.

Vorsicht


Ebenso wenig kann eine externe Datenbank, die einmal mit einer Version der 2er-Reihe bearbeitet wurde, anschließend wieder mit der externen Version 1.8.* bearbeitet werden, die kompatibel zu LibreOffice ist.

Auch kann bei der Nutzung einer anderen Version in der externen Datenbank die Version der internen Datenbank überschrieben werden. Der Aufruf als interne Datenbank ist dann unmöglich.

Parallelinstallation von interner und externer HSQLDB

Die Einbindung der externen Datei hsqldb.jar in den Class-Pfad kann bei unterschiedlichen Versionen dazu führen, dass interne Datenbanken anschließend nicht mehr zu öffnen sind. Base kommt mit den gleichlautenden Treibern nicht zurecht und will die externe Variante auch für die internen Datenbanken nutzen. Das geht beim ersten Öffnen noch gut. Beim zweiten Öffnen kommt dann aber die Meldung, dass die Datenbank nicht mehr zu öffnen ist, da sie mit einer neueren Version der HSQLDB geschrieben wurde.

Dem kann abgeholfen werden, indem für die externen Datenbanken die Datei hsqldb.jar nicht über den Class-Pfad in LO fest eingegeben wird, sondern stattdessen der Class-Pfad für die jeweilige Datenbankdatei über ein Makro geschrieben wird, wie auch hier zu lesen ist.
Hier wird die Datei «hsqldb.jar» unter Linux in dem o.g. Pfad abgelegt. Anschließend wird dieser Pfad der momentan geöffneten Datenbank als Treiberdatei zugewiesen.

Dieses Makro wird nach dem Öffnen der *.odb-Datei einmal aufgerufen. Es schreibt dann in die in der *.odb-Datei befindlichen Datei content.xml die entsprechende Verbindung zur Java-Klasse:

```
<db:connection-data>
  <db:connection-resource xlink:href="sdbc:embedded:hsqldb"/>
</db:connection-data>
```

Letztlich könnte also auch ohne das Makro ein entsprechender Pfad direkt in die content.xml der *.odb-Datei eingetragen werden. Nur ist dieser Weg für den Normalnutzer sicher nicht so komfortabel zu handhaben.

### Änderung der Datenbankverbindung zur externen HSQLDB


Aus einer Kopie der bestehenden Datenbank wird das Verzeichnis «database» extrahiert. Der Inhalt wird in das oben beschriebene frei wählbare Verzeichnis kopiert. Dabei sind die enthaltenen Dateien um den Datenbanknamen zu ergänzen:

- Datenbankname.backup
- Datenbankname.data
- Datenbankname.properties
- Datenbankname.script
- Datenbankname.log

Jetzt muss noch die «content.xml» aus dem *.odb-Päckchen extrahiert werden. Hier sind mit einem einfachen Texteditor die folgenden Inhalte zu suchen, die leider in einer durchlaufenden Zeile stehen:

```
<db:connection-data>
  <db:connection-resource xlink:href="sdbc:embedded:hsqldb"/>
</db:connection-data>
```
Diese Zeilen sind mit der Verbindung zur externen Datenbank zu ersetzen, hier der Verbindung zu einer Datenbank mit dem Namen "medien", die jetzt im Verzeichnis «hsqldb_data» liegt.

Falls, wie oben geschrieben, die Grundkonfiguration der HSQLDB nicht angetastet wurde stimmt auch der Nutzernamen und die nicht erforderliche Passworteinstellung.


Änderung der Datenbankverbindung für einen Mehrbenutzerbetrieb


Im Heimatverzeichnis des Servers, unter SuSE /var/lib/hsqldb, befinden sich unter anderem ein Verzeichnis «data», in dem die Datenbank abzulegen ist, und eine Datei «server.properties», die den Zugang zu den (eventuell also auch mehreren) Datenbanken in diesem Verzeichnis regelt.

Die folgenden Zeilen geben den kompletten Inhalt dieser Datei auf dem Rechner wieder. Es wird darin der Zugang zu 2 Datenbanken geregelt, nämlich der ursprünglichen Standard-Datenbank (die als neue Datenbank genutzt werden kann) als auch der Datenbank, die aus der *.odb-Datei extrahiert wurde.

```
# Hsqlldb Server cfg file.
# See the Advanced Topics chapter of the Hsqlldb User Guide.

server.database.0   file:data/db0
server.dbname.0     firstdb
server.urlid.0      db0-url
server.database.1   file:data/medien
server.dbname.1     medien
server.urlid.1      medien-url
server.silent       true
server.trace        false
server.port         9001
server.no_system_exit         true
```

Die Datenbank 0 wird mit dem Namen "firstdb" angesprochen, obwohl die einzelnen Dateien in dem Verzeichnis data mit "db0" beginnen. Die neue Datenbank wird als "Datenbank 1" hinzugefügt. Hier sind Datenbankname und Dateibeginn identisch.

Anhang: Änderung der Datenbankverbindung zur externen HSQLDB
Die beiden Datenbanken werden mit folgenden Zugängen angesprochen:

jdbc:hsqldb:hsql://localhost/firstdb;default_schema=true
username sa
password

jdbc:hsqldb:hsql://localhost/medien;default_schema=true
username sa
password

Die URL wurde hier bereits jeweils um den für den Schreibzugang über die grafische Benutzeroberfläche von LO erforderlichen Zusatz «;default_schema=true» ergänzt.

Wenn tatsächlich im Serverbetrieb gearbeitet werden soll ist natürlich aus Sicherheitsgründen zu überlegen, ob die Datenbank nicht mit einem Passwort geschützt werden soll.

Nun erfolgt die Serververbindung über LO. Im Hauptfenster von Base wird Bearbeiten → Datenbank → Eigenschaften aufgesucht.

![Datenbankeigenschaften](image)

Mit diesen Zugangsdaten wird auf den Server des eigenen Rechners zugegriffen. Im Netzwerk mit anderen Rechnern müsste dann entweder über Rechnernamen oder die IP-Adresse auf den Server, der ja auf dem aktuellen Rechner läuft, zugegriffen werden.

Beispiel: Der Rechner hat die IP 192.168.0.20 und ist im Netz bekannt mit dem Namen lin_serv. Jetzt ist an anderen Rechnern für die Verbindung zur Datenbank einzugeben:

jdbc:hsqldb:hsql://192.168.0.20/medien;default_schema=true
bzw.:

jdbc:hsqldb:hsql://lin_serv/medien;default_schema=true


Anhang: Änderung der Datenbankverbindung zur externen HSQLDB
Autoinkrementwerte mit der externen HSQLDB

Für die Nutzung der Auto-Werte müssen je nach Version von Base bei der Tabellenerstellung verschiedene Wege beschritten werden. Allen gleich ist erst einmal der folgende Eintrag unter Bearbeiten → Datenbank → Erweiterte Einstellungen erforderlich:

Mit dem Zusatz `GENERATED BY DEFAULT AS IDENTITY(START WITH 0)` soll die Funktion des automatisch hochzählenden Wertes für den Primärschlüssel erstellt werden. Die GUI von LO übernimmt zwar diesen Befehl (auch in der aktuellen LO-Version), schreibt davor aber leider die Anweisung `NOT NULL`, so dass die Reihenfolge der Befehlsfolge für die HSQLDB nicht lesbar ist. Hierbei ist zu berücksichtigen, dass die HSQLDB mit dem obigen Befehl ja bereits mitgeteilt bekommt, dass das entsprechende Feld den Primärschlüssel enthält.

Mit dem Auslesen des letzten Wertes und dem Hochlesen zum nächsten Wert klappt es in allen Versionen von LO über den Befehl `CALL IDENTITY()`. Dies trifft dann z.B. auf die Lösung zu, die Datenbank zuerst einmal als «*.odb-Päckchen» zu erstellen, gründlich zu testen und danach dann die Datenbanktabellen einfach auszulagern.
Sämtliche Abfragen, Formulare und Berichte lassen sich so weiter nutzen, da die Datenbank für die «*.odb-Datei» weiter auf die gleiche Weise angesprochen wird und eventuell spezifische SQL-Befehle mit der externen HSQLDB weiter gelesen werden können.

Umgang mit der internen Firebird-Datenbank

Die interne Firebird-Datenbank ist bisher nur als experimentelle Funktion verfügbar. Um solch eine Datenbank zu erstellen oder anschließend bearbeiten zu können, muss Extras → Optionen → LibreOffice → Erweitert → Optionale (instabile) Einstellungen → Experimentelle Funktionen aktivieren eingeschaltet sein. Schon dieser Weg zeigt auf, dass solch eine Datenbank nicht für den täglichen Gebrauch geeignet ist.

Über den folgenden Link können die wesentlichsten Bugs der internen Firebird-Datenbank zusammen mit LibreOffice eingesehen werden: Gemeldete Bugs für Firebird in Zusammenhang mit Base. Bitte nicht von der vermeintlich letzten Änderung am 3.01.2015 irritieren lassen. Zu dem Zeitpunkt hat sich nur ein QA-Mitglied ins CC gesetzt. Tatsächlich sind die Bugs leider meist so alt wie das Erscheinen der ersten internen Firebird-Version ...

Folgende Besonderheiten fallen dem Nutzer gegenüber der HSQLDB direkt auf:

2. Werden neue Daten eingegeben, so werden diese nicht automatisch in der Datenbankdatei abgespeichert. Der Speicherbutton wird bei jeder Dateneingabe erneut aktiviert. Bei der eingebauten HSQLDB ist die Abspeicherung nach Dateneingaben nicht notwendig.
4. Das Erstellen von Beziehungen ist nicht möglich, obwohl die externe Firebird-Datenbank Beziehungen unterstützt.

AutoWerte verfügbar machen:

Der folgende Code über Extras → SQL eingegeben kann darüber erst einmal hinweghelfen, dass AutoWerte bisher nicht vorgesehen sind:

```
CREATE TABLE "Tabelle1" ( "ID" INTEGER NOT NULL PRIMARY KEY, "Name" VARCHAR(20) NOT NULL );
CREATE GENERATOR GEN_T1_ID;
SET GENERATOR GEN_T1_ID TO 0;
```

Hiernach sollte zuerst einmal die SQL-Eingabeansicht ausgeschaltet werden. Es ist jetzt mindestens notwendig, Ansicht → Tabellen aktualisieren zu betätigen. Erst wenn die Tabelle dort aufgetaucht und vielleicht einfach schon einmal ein Eingabeversuch (erfolglos) gestartet wurde kann der folgende Trigger erstellt werden,

```
CREATE TRIGGER T1_BI FOR "Tabelle1" ACTIVE BEFORE INSERT POSITION 0 AS BEGIN IF (NEW.ID IS NULL) THEN NEW.ID = GEN_ID(GEN_T1_ID, 1); END;
```

Anschließend können zwar in die Tabelle lauter Einträge in "Name" getätigt werden, ohne einen Eintrag in "ID" zu vollziehen. Nur wird in dem Feld "ID" beständig '0' angezeigt. Erst wenn Aktuali-

Anhang: Umgang mit der internen Firebird-Datenbank
sieren betätigt wird werden die tatsächlich vergebenen Werte auch angezeigt. Der Trigger vergibt auf diese Art und Weise Werte beginnend mit 1.
Absolut speichern
In einigen Dialogen (z. B. Bearbeiten → AutoText) können Sie wählen, ob eine Datei relativ oder absolut gespeichert werden soll.

Wenn Sie absolut speichern wählen, werden alle Referenzen auf andere Dateien ebenfalls als absolut definiert, wobei sich diese Definition am jeweiligen Laufwerk, Volumen oder Quellverzeichnis orientiert. Der Vorteil dieser Methode besteht darin, dass das Dokument mit den Referenzen in ein anderes Verzeichnis oder einen anderen Ordner verschoben werden kann und die Referenzen weiterhin gültig bleiben.

Vergleichen Sie auch mit Relativ speichern.

Andocken
Einige Fenster in LibreOffice, zum Beispiel das Fenster Formatvorlagen und der Navigator, sind andockbare Fenster. Sie können diese Fenster verschieben, vergrößern oder sie an einer Kante des Arbeitsbereichs andocken. An jeder Kante können Sie auch mehrere Fenster über oder nebeneinander andocken. Sie können dann durch das Verschieben der Umrandungslinien die relativen Proportionen der Fenster verändern.


Andocken (AutoHide)
An jedem Fensterrand, an dem ein anderes Fenster angedockt ist, befindet sich eine Schaltfläche, die zum Einblenden oder Ausblenden des Fensters dient.

Wenn Sie zum Anzeigen des Fensters auf der Schaltfläche am Fensterrand klicken, bleibt das Fenster so lange eingeblendet, bis Sie es mit derselben Schaltfläche wieder ausblenden.

Wenn Sie das Fenster durch Klicken auf den Fensterrand einblenden, aktivieren Sie die Funktion AutoHide. AutoHide ermöglicht es, ein eigentlich ausgeblendetes Fenster durch Klicken auf dessen Rand kurzzeitig einzublenden. Sobald Sie mit der Maus in den Arbeitsbereich oder ein anderes Fenster klicken, wird das Fenster wieder verborgen.

ANSI
Das "American National Standards Institute" ist ein US-Normungsgremium, vergleichbar mit dem deutschen DIN.

ANSI-Zeichensatz

API

ASCII

Der erweiterte ASCII-Zeichensatz (8 bit) enthält 256 Zeichen. Jedem Zeichen ist eine eindeutige Nummer zugewiesen, die man auch als ASCII-Code bezeichnet.

In HTML-Seiten sollten nur die Zeichen des 7-Bit-ASCII-Zeichensatzes vorkommen. Andere Zeichen, wie etwa die deutschen Umlaute, werden durch Umschreibungen gekennzeichnet. So wird das kleine „ü“ etwa zu „uuml;“. Sie können in LibreOffice Zeichen im erweiterten ASCII-Code eingeben; der Exportfilter sorgt für die erforderliche Umwandlung.

Austauschformat
Ein Dateiformat, welches dem Austausch von Daten dient.

Das Austauschformat können Grafiken oder allgemeine Daten sein, z. B. von einer Tabellenkalkulation. Austauschformate dienen auch zum Import oder Export von einer Anwendung in eine andere.

Barcode

Bézierobjekt

Bildkompression
Bildkompression beruht wie jede Art der Datenkompression darauf, den ursprünglichen Datensatz entweder über eine Verlustfreie Kompression in einer vollständig rekonstruierbaren Form zu speichern oder Daten zu entfernen, deren Verlust kaum wahrnehmbar ist (Verlustbehaftete Kompression). Es gibt sehr viele Grafikformate, von denen aber viele veraltet sind und viele keine Kompression unterstützen, da sie Austauschformate für Grafikprogramme sind.

Complex Text Layout (CTL)
Sprachen mit komplexen Skripten können unter anderem folgende Eigenschaften aufweisen:

- Es werden Zeichen verwendet, die aus mehreren Teilen zusammengesetzt sind.
- Die Schreibrichtung des Texts läuft von rechts nach links.

LibreOffice unterstützt derzeit Hindi, Thai, Hebräisch und Arabisch als CTL-Sprachen. Aktivieren Sie die CTL-Unterstützung unter Extras → Optionen... → Spracheneinstellungen → Sprachen, um diese zu verwenden.

Dateiendung
Die Dateiendung gibt normalerweise das Format der Datei an und stellt die Buchstabenkombination hinter dem letzten Punkt des Dateinamen dar.

Datenbank
In einer Datenbank, meist Datenbanksystem (DBS) genannt, werden Daten gespeichert, verwaltet und miteinander in Beziehung gebracht. Zusätzlich können Informationen gesucht sowie gefiltert dargestellt werden. Stellt das System weitere Möglichkeiten, wie beispielsweise die
Verwaltung von Anwendern, Transaktionskontrolle usw. zur Verfügung, spricht man von einem Datenbankmanagementsystem (DBMS).

dBASE
dBASE war eines der ersten dateibasierten Datenbankverwaltungssysteme für PCs. Die strukturierten Daten wurden in „DataBaseFiles“ (daher die Dateiendung DBF) gespeichert. dBASE-Dateien und -Programme sind heute technisch veraltet, wobei das Dateiformat hin- und wieder als Austauschformat verwendet wird.

DDE
DDE steht für "Dynamic Data Exchange", also dem dynamischen Datenaustausch. Dies ist ein Vorgänger von OLE, dem "Object Linking and Embedding". Bei DDE wird ein Objekt in Form einer Verknüpfung zur Datei eingebunden, aber im Gegensatz zu OLE nicht selbst eingebettet.


Debian

DIN
Das "Deutsche Institut für Normung e. V." ist die bedeutendste nationale Normungsorganisation in der Bundesrepublik Deutschland.


Direkte Formatierung

Direkte Formatierungen können Sie aus Ihrem Dokument entfernen, indem Sie mit dem Text markieren und im Menü Format → Standardformatierung wählen.

Dem gegenüber steht die zu bevorzugende Indirekte Formatierung.

Drag&Drop
Sehen Sie unter Ziehen&Ablegen nach.

Drehfeld

Handelt es sich um ein Drehfeld für numerische Werte, dann können Sie auch eine Maßeinheit angeben, also z. B. „1 cm“, „5 mm“ „12 pt“ oder „2“.

**DTP-Programm**

Mittels eines DTP-Programms (Desktop-Publishing-Programm) werden hochwertige Dokumente, die aus Texten und Bildern bestehen, erstellt. Diese Dokumente können zum Beispiel für den Druck von Broschüren, Magazinen, Büchern oder Katalogen verwendet werden.

**Extension**


**Firebird**

Firebird ist ein freies Datenbankverwaltungsprogramm, welches auf Interbase basiert. Seit der LibreOffice-Version 4.2 ist Firebird in LibreOffice-Base integriert, vorerst aber nur als experimentelle Funktion neben der HSQLDB zugänglich.

**Formatierung**


**Gallery**

Die LibreOffice Gallery ist ein Fenster, in dem Sie Bilder und Klänge in thematischen Ordnern verwalten und zur Nutzung in LibreOffice bereit stellen können.

**GIF**

Das "Graphics Interchange Format" (deutsch: Grafik-Austauschformat) ist ein Grafikformat, das eine gute Verlustfreie Kompression für Bilder mit geringer Farbtiefe besitzt (bis zu 256 verschiedene Farben pro Einzelbild).

Darüber hinaus können mehrere Einzelbilder in einer Datei abgespeichert werden, die von geeigneten Betrachtungsprogrammen wie Webbrowsern als Animationen interpretiert werden.

**GPL**


**HSQLDB**

HSQLDB (Hyper Structured Query Language Database) ist ein freies, vollständig in Java programmiertes relationales Datenbankverwaltungssystem (RDBMS). Schon zu Zeiten von OpenOffice ist HSQLDB in Base integriert. Der Schwerpunkt von HSQLDB ist, dass es sehr leicht in andere Produkte integriert werden kann (die entsprechende Bibliothek hat nur etwa eine Größe von einem Megabyte).

**Hurenkinderregelung**

Schusterjungen und Hurenkinder sind historische Begriffe aus der Typographie, die seit langem benutzt werden.

Ein Hurenkind ist die letzte Zeile eines Absatzes, die alleine am oberen Rand der nächsten Seite steht. Mit einem Textdokument von LibreOffice können Sie diese unschöne Erscheinung automatisch für die gewünschte Absatzvorlage vermeiden. Dabei können Sie sogar wählen, wie viele Zeilen mindestens immer zusammen auf einer Seite gehalten werden sollen.

Dem gegenüber steht die **Schusterjungenregelung**.

**IEC**


**IME**

IME ist eine Abkürzung für "Input Method Editor", also ein Eingabemethoden-Editor. Dies ist ein Programm, das die Eingabe von komplexen Zeichen aus nichtwestlichen Zeichensätzen über eine Standardtastatur ermöglicht.

**Indirekte Formatierung**

Bei der indirekten oder "weichen" Formatierung nehmen Sie Formatierungen nicht direkt am Text vor, sondern durch Zuweisen von Formatvorlagen. Der große Vorteil besteht darin, dass Sie mit der Änderung einer Formatvorlage jedes Objekt (Absätze, Rahmen, usw.) ändern, dem Sie diese Formatvorlage zugewiesen haben.

Dem gegenüber steht die **Direkte Formatierung**.

**ISO**

Die Internationale Organisation für Normung (International Organization for Standardization) ist die internationale Vereinigung von Normungsorganisationen und erarbeitet internationale Normen in allen Bereichen mit Ausnahme der Elektrik, der Elektronik und der Telekommunikation, für welche die **IEC** zuständig ist.

**Java**

Java ist eine Programmiersprache, die von der Firma Oracle weiterentwickelt (ursprünglich von der Firma Sun Microsystems entwickelt) und bereitgestellt wird. Für die Ausführung von Java-Programmen in einer Anwendung ist die JRE (Java Runtime Environment) notwendig.

**JDBC**

Sie können die API der Java DataBase Connectivity (JDBC) zum Herstellen einer Verbindung mit einer Datenbank von LibreOffice verwenden. JDBC-Treiber sind in der Programmiersprache Java geschrieben. Dabei unterscheidet man vier Typen:

1. JDBC-ODBC bridge plus ODBC driver
2. Native-API partly-Java driver
Die Verbindung wird über einen Datenbanktreiber hergestellt. Der JDBC-Treiber ruft die entsprechenden Datenbank-Client-Funktionen auf. Der DBMS-Client muss dabei installiert sein.

3. JDBC-Net pure Java driver
Hier werden durch den JDBC-Treiber die Aufrufe in das entsprechende Protokoll übersetzt und dann von einem entsprechenden Server (middleware) in das DBMS-Protokoll übersetzt und an das DBMS geschickt.

4. Native-protocol pure Java driver

JPEG

JPG
Sehen Sie unter JPEG nach.

JRE

LibreOffice ist ohne JRE lauffähig, JRE ist jedoch für die LibreOffice-Komponente Base wichtig.

Kerning
Kerning ist die englische Bezeichnung für eine Unterschneidung oder Spationierung. Darunter versteht man das Verringern oder Vergrößern des Abstandes zwischen Buchstabenpaaren zum optischen Ausgleich des Schriftbildes, z. B. bei „W“ und „a“.

In Kerning-Tabellen ist vermerkt, welche Buchstabenpaare mehr oder weniger Abstand benötigen. Diese Tabellen sind in der Regel Bestandteil der jeweiligen Schrift.

Kontextmenü

LGPL
Die GNU Lesser General Public License (LGPL) ist von der GPL abgeleitet, erlaubt aber im Gegensatz zur GPL, Bibliotheken etc. in eigene (proprietäre) Software einzubinden.

Link
Ein Link (oder Hyperlink) verknüpft den Inhalt mit einem anderen Dokument oder einem Abschnitt im gleichen Dokument. Links können sich hinter Texten, Symbolen oder Bildern verborgen.

Makro
Ein Makro ist eine Abfolge von Befehlschritten, die in einer Programmiersprache wie LibreOffice Basic, JavaScript, BeanShell oder Python definiert werden. Ein Makro kann über den Makrorekorder oder über eigene Textzeilenprogrammierung erstellt und über die Menü- oder Symbolleiste abgerufen werden.
Markieren
Durch Markieren legen Sie fest, welche Teile eines Textes, einer Tabelle oder eines Bildes Sie bearbeiten wollen. Hierzu führen Sie mit gedrückter Maustaste den Mauszeiger über den gewünschten Bereich.

MariaDB
MariaDB ist ein relationales Datenbankverwaltungssystem, welches als Fork (Abspaltung) von MySQL entstanden ist. MariaDB steht unter GPL Lizenz, die Client-Bibliotheken unter LGPL.

MS-DOS
MS-DOS, kurz für Microsoft Disk Operating System, war Microsofts erstes Betriebssystem für PCs.
Es wurde ursprünglich für den Intel-Prozessor 8086/8088 entwickelt und war in den späten 1980er und frühen 1990er Jahren das dominierende Betriebssystem für Einzelplatzrechner. Die früheren Windows-Versionen 1.0 bis 3.11, 95 (4.0), 98 (4.1) und ME (4.9) waren von DOS abhängig. Windows NT und die darauf basierenden Microsoft-Betriebssysteme (Windows 2000, XP, Vista und 7) bauen nicht mehr auf MS-DOS auf und können DOS-Software nicht mehr oder nur noch eingeschränkt ausführen.

MySQL
MySQL ist ein sehr verbreitetes relationales Datenbankverwaltungssystem und steht für viele Betriebssysteme zur Verfügung (Unix, Linux, Windows, Mac OS, …). Es steht unter einem dualen Lizenzsystem, einmal proprietär und einmal unter der GPL. Ursprünglich wurde MySQL von der schwedischen Firma MySQL AB entwickelt und dann von Sun Microsystems aufgekauft, das seinerseits von Oracle gekauft worden ist.

OASIS

OASIS Open Document Format for Office Applications
Sehen Sie unter ODF nach.

Objekt
Objekte sind selbstständig und beeinflussen sich nicht gegenseitig. Jedem Objekt, das Daten enthält, werden bestimmte Befehle zugeordnet. Ein Grafikobjekt wird so mit Befehlen zur Bildbearbeitung versehen, eine Tabelle mit Befehlen zur Berechnung usw.

ODBC
Open DataBase Connectivity (ODBC) ist ein Protokoll für den Zugriff auf Datenbank-Systeme durch Anwendungen. Dabei kommt die Abfragesprache SQL zum Einsatz. In LibreOffice können Sie von Fall zu Fall entscheiden, ob die Abfragen durch Eingabe von SQL-Befehlen oder anhand der interaktiven Hilfe erfolgen sollen. Bei letzterer Methode definieren Sie die Abfrage per Mausklick und LibreOffice übersetzt sie dann automatisch in SQL.
ODF

OLE

OpenDocument
Sehen Sie unter ODF nach.

OpenGL

PDF

Ziel war es, ein Dateiformat für elektronische Dokumente zu schaffen, das diese unabhängig vom ursprünglichen Anwendungsprogramm, vom Betriebssystem oder von der Hardwareplattform originalgetreu weitergeben kann. Ein Leser einer PDF-Datei soll das Dokument immer in der Form betrachten und ausdrucken können, die der Autor festgelegt hat. Die typischen Konvertierungsprobleme (wie zum Beispiel veränderter Seitenumbruch oder falsche Schriftarten) beim Austausch eines Dokuments zwischen verschiedenen Anwendungsprogrammen entfallen.

Neben Text, Bildern und Grafik kann eine PDF-Datei auch Hilfen enthalten, die die Navigation innerhalb des Dokumentes erleichtern. Dazu gehören zum Beispiel anklickbare Inhaltsverzeichnisse und miniaturisierte Seitenvorschauen.


PDF-Dateien sind nicht auf PCs beschränkt, sondern kommen auch z. B. in Druckmaschinen zum Einsatz. Des Weiteren gibt es spezielle Erweiterungen zur Langzeitarchivierung.

Pixel
Das Bild eines digitalen Fotos und eines Monitores setzt sich aus Punkten (Pixeln) zusammen. Die Anzahl der Bildpunkte wird in einem Zahlenpaar angegeben.

Pixelgrafik
Eine Pixelgrafik, auch Rastergrafik genannt, ist eine Form der Beschreibung eines Bildes in Form von computerlesbaren Daten. Rastergrafiken bestehen aus einer rasterförmigen Anord-
nung von Bildpunkten (Pixel genannt), denen jeweils eine Farbe zugeordnet ist. Die Hauptmerkmale einer Rastergrafik sind daher die Bildgröße (Breite und Höhe gemessen in Pixeln) sowie die Farbtiefe.


PostgreSQL


Primärschlüssel

Ein Primärschlüssel (PK – primary key) dient zur eindeutigen Kennzeichnung eines Datenbankfeldes. Meist wird für den Primärschlüssel automatisch auch ein Index erstellt. Diese eindeutige Identifikation von Datenbankfeldern wird bei einer relationalen Datenbank verwendet, bei denen von einer Tabelle auf die Daten einer anderen Tabelle zugegriffen werden kann. Wird von einer anderen Tabelle auf einen Primärschlüssel verwiesen, so bezeichnet man ihn als Fremdschlüssel (FK – foreign key). Durch die beiden Schlüssel wird die Referentielle Integrität sichergestellt, was vereinfacht bedeutet, dass der Datensatz des Fremdschlüssels nur auf existierende Datensätze des Primärschlüssels verweisen darf. Dies wird vom DBMS-System sichergestellt.

In LibreOffice definieren Sie Primärschlüssel in der Entwurfsansicht einer Tabelle, indem Sie im Kontextmenü eines Zeilenkopfes für das ausgewählte Feld den entsprechenden Befehl wählen.

PNG

"Portable Network Graphics" (deutsch: portable Netzwerkgrafik) ist ein Rastergrafik-Format, das eine Verlustfreie Kompression benutzt. Es wurde als freier Ersatz für das ältere, bis zum Jahr 2004 mit Patentforderungen belastete Format GIF entworfen und ist weniger komplex als TIFF. Die Dateien werden mit einem wählbaren Faktor und, im Gegensatz zu JPG, stets verlustfrei komprimiert. PNG unterstützt neben unterschiedlichen Farbtiefen auch Transparenz per Alphakanal.

QR-Code

Der QR-Code (Quick Response Code) ist ein zweidimensionaler Code, der ursprünglich für die Markierung von Baugruppen und Komponenten entwickelt worden ist. Der Code enthält redundante Daten, so dass in einem gewissen Umfang eine Fehlerkorrektur möglich ist, wenn beispielsweise durch Verschmutzung nicht alles erkennbar ist (Level H erlaubt, dass 30% wiederhergestellt werden können). Heute wird der QR-Code nicht nur als Produktetikette verwendet, sondern beispielsweise für URLs (Verweise auf Webseiten), als Visitenkarte (VCARD) usw.

Rastergrafik

Siehe Sie unter Pixelgrafik nach.

Registerhaltigkeit


Wenn Sie einen Absatz, eine Absatzvorlage oder eine Seitenvorlage als registerhaltig definieren, werden die Grundlinien der betroffenen Zeichen an einem vertikalen Seitenraster ausge-
richtet. Dabei spielt die Schriftgröße oder das Vorhandensein einer Grafik keine Rolle. Sie kön-
nen die Einstellung für dieses Raster auch als eine Eigenschaft der Seitenvorlage festlegen.

Relationale Datenbank

Eine relationale Datenbank ist ein Datenbanksystem, in dem Daten in Form miteinander-
verbundener Tabellen verwaltet werden. Die Daten können in verschiedener Weise abgefragt wer-
den, ohne dass die zugrunde liegenden Tabellen reorganisiert werden müssen.

Ein relationales Datenbankverwaltungssystem (RDBMS) ist ein Programm, mit dem Sie relation-
ionale Datenbanken erstellen, aktualisieren und verwalten können. Ein RDBMS akzeptiert SQL-
Anweisungen, die entweder vom Benutzer eingegeben werden oder in einer Anwendung ent-
halten sind, und erzeugt, aktualisiert oder ermöglicht den Zugriff auf Datenbanken.

Als typisches Beispiel lässt sich eine Datenbank mit Kunden-, Verkaufs- und Rechnungstabell-
en heranziehen. In der Rechnungstabelle sind nicht die eigentlichen Kunden- oder Verkaufs-
daten, sondern Referenzen, in Form von relationalen Verknüpfungen, oder Relationen auf die
Tabellenfelder mit den entsprechenden Kunden- und Verkaufsdaten (z. B. das Kundenum-
mernfeld aus der Kundentabelle) enthalten.

Relativ speichern

In einigen Dialogen (z. B. Bearbeiten → AutoText) können Sie wählen, ob eine Datei relativ
oder absolut gespeichert werden soll.

Wenn Sie sich für das relative Speichern entscheiden, werden Referenzen auf eine eingebet-
tete Grafik oder ein anderes Objekt im Dokument relativ zur Position im Dateisystem gespei-
chert. In diesem Fall spielt es keine Rolle, wo die referenzierte Verzeichnisstruktur eingetragen
ist. Solange die Referenz auf derselben Festplatte bzw. demselben Volume bleibt, werden die
Dateien unabhängig vom Speicherort immer aufgefunden. Dies ist für solche Dokumente von
besonderer Bedeutung, die auch auf Computern mit einer möglicherweise ganz anderen Ver-
zeichnisstruktur oder anderen Laufwerks- oder Volume-Namen verwendet werden sollen. Auch
für das Anlegen von Verzeichnisstrukturen auf einem Internetserver empfiehlt es sich, Doku-
mente relativ zu speichern.

Vergleichen Sie auch mit Absolut speichern.

RTF

RTF (Rich Text Format) ist ein für den Austausch von Textdateien entwickeltes Dateiformat. Es
zeichnet sich dadurch aus, dass Formatierungsinformationen in direkt lesbare Textdaten kon-
vertiert werden. Leider entstehen dabei im Vergleich zu anderen Formaten recht große
Dateien.

Schusterjungenregelung

Schusterjungen und Hurenkinder sind historische Begriffe aus der Typographie, die seit langem
benutzt werden.

Ein Schusterjunge ist die erste Zeile eines Absatzes, die alleine am unteren Rand der Vorseite
steht. Mit einem Textdokument von LibreOffice können Sie diese unschöne Erscheinung auto-
matisch für die gewünschte Absatzvorlage vermeiden. Dabei können Sie sogar wählen, wie
viele Zeilen mindestens immer zusammen auf einer Seite gehalten werden sollen.

Dem gegenüber steht die Hurenkinderregelung.

SQL

SQL (Structured Query Language) ist eine Sprache zur Definition von Datenstrukturen und zur
Spezifikation von Abfragen einer relationalen Datenbank. In LibreOffice haben Sie die Möglich-
keit, Abfragen entweder in SQL oder mithilfe der Maus zu definieren.

SQL (damals SEQUEL) wurde im Jahre 1968 im Rahmen einer studentischen Arbeit in Berkley
entwickelt und 1986 als ANSI-Standard (SQL1) verabschiedet. Die letzte Änderung war 2011,

SQL teilt sich in folgende Unterkategorien auf:

- **DDL**: Data Definition Language
  
  z. B. CREATE TABLE

- **DCL**: Data Control Language
  
  GRANT, REVOKE

- **DML**: Data Manipulation Language
  
  z. B. INSERT, UPDATE, DELETE

- **DQL**: Data Query Language
  
  SELECT

- **TCL**: Transaction Control Language
  
  COMMIT, ROLLBACK

### SQL-Datenbank

Eine SQL-Datenbank ist ein Datenbank-System, das eine SQL-Schnittstelle bietet. SQL-Datenbanken werden oft in Client/Server-Netzwerken eingesetzt, in denen verschiedene Clients auf einen zentralen Server (z. B. einen SQL-Server) zugreifen, daher bezeichnet man sie auch als SQL-Server-Datenbanken oder kurz SQL-Server.


### SQL-Server

Sehen Sie unter **SQL-Datenbank** nach.

### SWF


### Symbolleiste

Ein kleines Fenster mit mehreren Symbolen (Icons), mit denen gängige Aufgaben erledigt werden. Die Symbolleisten können in LibreOffice an beliebigen Stellen positioniert werden.

### TIF

Sehen Sie unter **TIFF** nach.

### TIFF


Größter Nachteil von TIFF ist seine Komplexität. Die Vielfalt möglicher gültiger TIFF-Dateien kann nur schwer von einzelnen Programmen unterstützt werden. In der Spezifikation des Dateiformates ist deswegen eine Untermenge gültiger TIFF-Dateien definiert, die jedes TIFF-fähige Programm verarbeiten können sollte, genannt Baseline TIFF.
Unicode-Zeichensatz


- UTF-8: 00 00 FE FF
- UTF-16LE: FF FE
- UTF-16BE: FE FF
- UTF-32LE: FF FE 00 00
- UTF-32BE: 00 00 FE FF

UTF-8

UTF-8 (Abk. für 8-bit UCS Transformation Format) ist die am weitesten verbreitete Kodierung vom Unicode-Zeichensatz.


UTF-16

UTF-16 ist eine Kodierung vom Unicode-Zeichensatz, die für die häufig gebrauchten Zeichen optimiert ist. Es ist das älteste der Unicode-Kodierungsformate und wird standardmäßig in Java intern verwendet, daher hat ein char in Java 16 Bit.

UTF-32

Vektorgrafik
Eine Vektorgrafik ist eine Computergrafik, die aus grafischen Primitiven wie Linien, Kreisen, Polygonen oder allgemeinen Kurven (Splines) zusammengesetzt ist. Meist sind mit Vektorgrafiken Darstellungen gemeint, deren Primitiven sich zweidimensional in der Ebene beschreiben lassen. Eine Bildbeschreibung, die sich auf dreidimensionale Primitiven stützt, wird eher 3D-Modell oder Szene genannt.


Verknüpfung
Verknüpfungen dienen zum schnellen Aufrufen von Dateien auf der Arbeitsoberfläche. Mit einem einfachen oder einem Doppelklick (je nach Einstellung des Betriebssystems) öffnet sich das entsprechende Programm.

Verlustbehafette Kompression
Bei der verlustbehafteten Kompression wird versucht, den Informationsverlust unmerklich oder wenigstens ästhetisch erträglich zu halten. Diese Methoden nutzen aus, dass kleine Farbänderungen für das Auge nicht sichtbar sind. Ähnlich wie bei der verlustbehafetten Audiokomprimierung basiert die Bildkomprimierung auf einem Modell der menschlichen Wahrnehmung. Der Komprimierungsalgorithmus soll bevorzugt die Bildinformationen entfernen, die über die Aufnahmefähigkeit der menschlichen Bildwahrnehmung hinausgehen. Das Wahrnehmungsmodell ist jedoch, im Gegensatz zur Audiokompression, nicht explizit formuliert und in die Algorithmen eingearbeitet, sondern mehr intuitiv.

Technisch bedingt ist eine Wiederherstellung des Originalzustandes nicht mehr möglich.
Im Gegensatz dazu steht die Verlustfreie Kompression.

Verlustfreie Kompression
Bei der verlustfreien Kompression geht keine Information verloren. Die Daten werden nur anders als vorher organisiert, indem bestimmte Redundanzen erkannt und zusammengefasst werden. Zum Beispiel können sich wiederholende Bitfolgen einmal in einem Wörterbuch abgelegt und dann nur noch durch ihre Nummer repräsentiert werden. Es können beliebige allgemeine Komprimierungsverfahren verwendet werden, die sich auch auf andere Arten von Daten wie Text anwenden lassen.

Dadurch, das keine Information verloren geht, ist die Wiederherstellung des Originalzustandes jederzeit wieder möglich.
Im Gegensatz dazu steht die Verlustbehafette Kompression.

Windows-Zeichensatz


XHTML
XHTML (Extensible Hypertext Markup Language) ist eine erweiterte Version des HTML.
**XML**


**Zahlensystem**

Ein Zahlensystem ist durch die Anzahl der zur Darstellung von Zahlen verfügbaren Zeichen gekennzeichnet. So basiert beispielsweise das Dezimalsystem auf zehn Ziffern (0 bis 9 – Basis 10), das Binär- oder Dualzahlensystem auf zwei Ziffern (0 und 1 – Basis 2) und das Hexadezimalsystem auf 16 Zeichen (0 bis 9 und A bis F – Basis 16).

**Ziehen&Ablegen**


**Zwischenablage**

Die Zwischenablage ist ein virtueller Speicher zum Austausch von Daten zwischen mehreren Programmen. Mit Herunterfahren des PCs werden die Daten der Zwischenablage gelöscht.
# Stichwortverzeichnis

<table>
<thead>
<tr>
<th>Abfragen</th>
<th>23, 225</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beziehungsdefinition</td>
<td>236</td>
</tr>
<tr>
<td>Bezugstabellen</td>
<td>264</td>
</tr>
<tr>
<td>Eingabe</td>
<td>225</td>
</tr>
<tr>
<td>Eingabemöglichkeiten</td>
<td>257</td>
</tr>
<tr>
<td>Ergebnis</td>
<td>229</td>
</tr>
<tr>
<td>Erweiterungen mit SQL</td>
<td>242</td>
</tr>
<tr>
<td>Funktionen</td>
<td>233</td>
</tr>
<tr>
<td>in Formularen verwenden</td>
<td>256</td>
</tr>
<tr>
<td>Korrelierte Unterabfrage</td>
<td>263</td>
</tr>
<tr>
<td>Parameter</td>
<td>261</td>
</tr>
<tr>
<td>per GUI</td>
<td>225</td>
</tr>
<tr>
<td>SQL</td>
<td>230</td>
</tr>
<tr>
<td>Tabellenansichten</td>
<td>268</td>
</tr>
<tr>
<td>Unterabfragen</td>
<td>262</td>
</tr>
<tr>
<td>Zeilenumbuch</td>
<td>381</td>
</tr>
<tr>
<td>Zusammenfassung</td>
<td>267</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addon</th>
<th>301</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alter ermitteln</td>
<td>372</td>
</tr>
<tr>
<td>andocken</td>
<td>519</td>
</tr>
<tr>
<td>ANSI</td>
<td>519</td>
</tr>
<tr>
<td>API</td>
<td>519</td>
</tr>
<tr>
<td>Austauschformat</td>
<td>520</td>
</tr>
<tr>
<td>AutoHide</td>
<td>519</td>
</tr>
<tr>
<td>Barcode</td>
<td>490, 520</td>
</tr>
<tr>
<td>Benutzeroberfläche</td>
<td>98</td>
</tr>
<tr>
<td>Berichte</td>
<td>23</td>
</tr>
<tr>
<td>Beziehungen</td>
<td>21, 92</td>
</tr>
<tr>
<td>Eins-zu-Eins (1:1)</td>
<td>93</td>
</tr>
<tr>
<td>Eins-zu-Viele (1:n)</td>
<td>92</td>
</tr>
<tr>
<td>Viele-zu-Viele (n:m)</td>
<td>93</td>
</tr>
<tr>
<td>Bézierkurven</td>
<td>520</td>
</tr>
<tr>
<td>Bildkompression</td>
<td>520</td>
</tr>
<tr>
<td>Codeschnipsel</td>
<td>372</td>
</tr>
<tr>
<td>Comboboxen</td>
<td>429</td>
</tr>
<tr>
<td>Complex Text Layout</td>
<td>520</td>
</tr>
<tr>
<td>Container</td>
<td>18</td>
</tr>
<tr>
<td>CTL</td>
<td>520</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Daten</th>
<th>520</th>
</tr>
</thead>
<tbody>
<tr>
<td>aus Calc exportieren</td>
<td>345</td>
</tr>
<tr>
<td>Einfügen als Felder</td>
<td>324, 326</td>
</tr>
<tr>
<td>Felder</td>
<td>326</td>
</tr>
<tr>
<td>Gruppieren</td>
<td>381</td>
</tr>
<tr>
<td>in Calc einfügen</td>
<td>343</td>
</tr>
<tr>
<td>Text</td>
<td>322</td>
</tr>
</tbody>
</table>

über Zwischenablage einfügen | 347 |
Zusammenfassung | 381 |
Datenbank | 17, 59, 520 |
Anbindung | 320 |
anlegen | 59 |
anmelden | 68 |
Assistent | 64 |
Aufgaben | 355 |
aus Calc importieren | 345 |
Autowerte neu einstellen | 483 |
Benutzername | 67 |
Calc | 342 |
Database management system | 12 |
Datenbankmanagementsystem | 521 |
Datenbanksystem | 520 |
dBASE | 74, 521 |
DBMS | 12, 17, 521 |
DBS | 520 |
Direkte Verbindung | 66 |
Eigenschaften | 483 |
Enterträge bearbeiten | 486 |
extern | 61 |
externe HSQLDB einbinden | 510 |
Firebird | 522 |
Geschwindigkeit | 487 |
HSQLDB | 62, 522 |
Informationstabellen | 500 |
intern | 59 |
JDBC | 62, 71 |
Kennwort | 67 |
Komprimieren | 483 |
Konvertieren | 346 |
Managementssystem | 17 |
MySQL | 59, 525 |
Name | 66 |
neu erstellen | 65 |
odb | 60 |
ODBC | 62, 70 |
ODBMS | 527 |
PostgreSQL | 527 |
RDBMS | 522, 528 |
relational | 528 |
Reparatur | 501 |
übers Netz ansprechen | 66 |
unnötige Einträge | 485 |
Verbindung herstellen | 61 |
Versionsprobleme | 509 |
Verwaiste Einträge ermitteln | 487 |
Wartung | 483 |
Wiederherstellung | 501 |
Datenfilterung | 355 |
Abfrage | 355 |

Anhang: Stichwortverzeichnis 533
<table>
<thead>
<tr>
<th>Stichwort</th>
<th>Seitenzahl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anhang</td>
<td>534</td>
</tr>
<tr>
<td>Stichwortverzeichnis</td>
<td></td>
</tr>
<tr>
<td>Anhang</td>
<td></td>
</tr>
<tr>
<td>Datenquelle</td>
<td>320</td>
</tr>
<tr>
<td>Browser</td>
<td>320</td>
</tr>
<tr>
<td>Calc</td>
<td>320</td>
</tr>
<tr>
<td>Writer</td>
<td>320</td>
</tr>
<tr>
<td>DDE</td>
<td>521</td>
</tr>
<tr>
<td>Defaultwert</td>
<td>98</td>
</tr>
<tr>
<td>DIN</td>
<td>519, 521</td>
</tr>
<tr>
<td>Drag&amp;Drop</td>
<td>532</td>
</tr>
<tr>
<td>Drehfeld</td>
<td>521</td>
</tr>
<tr>
<td>DTP</td>
<td>522</td>
</tr>
<tr>
<td>Dynamic Data Exchange</td>
<td>521</td>
</tr>
<tr>
<td>Erweiterung</td>
<td>522</td>
</tr>
<tr>
<td>Etikettendruck</td>
<td>320</td>
</tr>
<tr>
<td>Extension</td>
<td>522</td>
</tr>
<tr>
<td>FAQ</td>
<td>13, 15</td>
</tr>
<tr>
<td>Feld</td>
<td>22</td>
</tr>
<tr>
<td>foreign key</td>
<td>527</td>
</tr>
<tr>
<td>Formatierung</td>
<td>20</td>
</tr>
<tr>
<td>Ansicht</td>
<td>207</td>
</tr>
<tr>
<td>Daten</td>
<td>144</td>
</tr>
<tr>
<td>Eigenschaften</td>
<td>143</td>
</tr>
<tr>
<td>Entwurf</td>
<td>140</td>
</tr>
<tr>
<td>Ereignisse</td>
<td>145</td>
</tr>
<tr>
<td>erstellen</td>
<td>138</td>
</tr>
<tr>
<td>Fehlermeldungen</td>
<td>214</td>
</tr>
<tr>
<td>Feld</td>
<td>141</td>
</tr>
<tr>
<td>Hauptformular</td>
<td>194</td>
</tr>
<tr>
<td>Kontrollfelder</td>
<td>146</td>
</tr>
<tr>
<td>Name</td>
<td>143</td>
</tr>
<tr>
<td>Navigator</td>
<td>140, 210</td>
</tr>
<tr>
<td>Unterformular</td>
<td>194</td>
</tr>
<tr>
<td>Forum</td>
<td></td>
</tr>
<tr>
<td>Fremdschlüssel</td>
<td>22, 95, 527</td>
</tr>
<tr>
<td>Funktionen</td>
<td>492</td>
</tr>
<tr>
<td>Datenbankverbindung</td>
<td>497</td>
</tr>
<tr>
<td>Datum/Zeit</td>
<td>496</td>
</tr>
<tr>
<td>Numerisch</td>
<td>493</td>
</tr>
<tr>
<td>System</td>
<td>498</td>
</tr>
<tr>
<td>Text</td>
<td>494</td>
</tr>
<tr>
<td>Gallery</td>
<td>522</td>
</tr>
<tr>
<td>GIF</td>
<td>522</td>
</tr>
<tr>
<td>GPL</td>
<td>522, 524f.</td>
</tr>
</tbody>
</table>
Datenbanken komprimieren..........................452
Datenbanksicherungen..............................449
Dialoge.................................................464
DIM......................................................389
Editor......................................................384
Filtern von Datensätzen............................411
Fremdschlüssel übertragen........................431
Kontrollfunktionen....................................437
Mailprogramm.........................................459
Makrosicherheit........................................385
Navigation Formular..................................438
Prozedur..................................................384
Rahmen....................................................389
Suchen von Datensätzen.............................420
Variablen definieren..................................389
verwalten...............................................384
Zugriff auf Elemente..................................391
Zugriff auf Formular.................................390
zuweisen...............................................386
Markieren...............................................525
Mehrfachselektion.....................................182
MS-DOS....................................................525
MySQL......................................................63
JDBC......................................................63
ODBC......................................................63
ODBC-Verbindung.....................................63
Navigationsleiste......................................179
NULL......................................................107, 119
OASIS.......................................................525f.
Objekt......................................................525
ODF.........................................................63, 525
OLE.........................................................526
OpenGL.....................................................526
PDF........................................................526
Pixel.........................................................526
Pixelgrafik.................................................526
PK..........................................................527
PNG........................................................527
Primärschlüssel........................................21, 91, 95, 527
primary key..............................................527
Programmiersprachen..............................
Basic......................................................524
BeanShell...............................................524
JavaScript..............................................524
Python.....................................................524
QR-Code..................................................527
Rastergrafik.............................................526
Referentielle Integrität...............................22
Registerhaltigkeit.....................................527
Relation...................................................21, 237
Report-Designer...........................................
aufrufen...............................................274
Bedingte Anzeige.......................................300
Bedingte Formatierung...............................301
Benutzerdefinierte Funktionen......................298
Dateneigenschaften von Feldern....................290
Diagramme einbinden................................286
Eigenschaften von Feldern..........................282
Formeleingabe.........................................300
Formeleingaben.......................................291
Funktionen..............................................291, 293
RESTRICT.................................................117
RTF.........................................................528
Schnittfläche............................................176
Schlüsselmodel.........................................21
Schusterjungen...........................................523, 528
SEQUEL.....................................................528
Serienbrief..............................................25, 60
Seriendruck.............................................326
Assistent.................................................326
Etiketten.................................................335
Externe Formulare.....................................341
Feldbegriffe............................................339
Felder......................................................338
Serienbrief.............................................327
Serienbriefassistent.................................327
SGI..........................................................526
Spationierung..........................................524
Speicherfresser........................................91
speichern...............................................519
absolut...............................................519
relativ..................................................528
SQL........................................................97, 528
Alias.......................................................252
ALTER COLUMN.....................................483
ALTER TABLE.........................................114, 483
Bedingungen..........................................128
CASCADE...............................................117
CASEWHEN.............................................372
Char.......................................................381
COUNT....................................................378
CREATE TABLE.........................................111
CURDATE...............................................372
DATEDIFF..............................................372
Datenbank.............................................529

Anhang: Stichwortverzeichnis

535
DAYOFYEAR ................................................. 372
DCL ......................................................... 529
DDL ......................................................... 529
DELETE ...................................................... 130
direkte Eingabe ........................................ 110
DISTINCT ................................................... 243, 356
DML ......................................................... 529
DQL ......................................................... 529
DROP TABLE .................................................. 117
Fehlercode .................................................. 235
GENERATED BY DEFAULT .......................... 516
GROUP BY .................................................. 234, 243
Group_Concat ............................................. 381
HAVING ....................................................... 243
IF EXISTS ..................................................... 117
IFNULL ....................................................... 355
INSERT INTO ............................................. 129
LIKE ......................................................... 357
LIMIT ......................................................... 243
Listenfelder ............................................... 253
LOWER ......................................................... 358
ORDER BY ................................................... 243, 356
RESTART WITH .......................................... 483
RESTRICT ..................................................... 117
SELECT ....................................................... 230, 243, 355
Server ....................................................... 529
SQL:2011 ..................................................... 529
Sub-Select ............................................... 355
SUM .......................................................... 377
Tabelle erstellen ....................................... 111
Tabelle löschen ......................................... 117
Tabellenänderung ...................................... 114
TCL ........................................................... 529
UNION ......................................................... 243
UPDATE ....................................................... 129
WHERE ......................................................... 243
Streifencode .............................................. 520
Strichcode ............................................... 520
Support ..................................................... 113
SWF ......................................................... 529
Symbolleiste .............................................. 529
Tabellen ..................................................... 21, 91
allgemein .................................................. 91
AutoWert ................................................. 102, 122
Bezeichnung .............................................. 97
Beziehung .................................................. 91
Datentypen ............................................... 91
Defaultwert .............................................. 98
Dokument ................................................... 76
Editor .........................................................
Datentypen ................................................. 490
Eingabe von Daten ....................................... 122
Eltern-Tabelle ........................................... 118
filtern ....................................................... 127
Index ......................................................... 105
NULL ........................................................ 107
Quellen ..................................................... 76
sortieren .................................................... 124
Sortierung .................................................. 105
Suchfunktion ............................................. 125
Timestamp ............................................... 103
Unterschied zu Tabellenkalkulation ........... 91
verknüpfen ............................................... 117
Wiederholungen ....................................... 91
Zeitfelder .................................................... 103
Tabellenkalkulationen ................................ 17
Tabellenkontrollfeld ................................... 189
Thunderbird Adressbuch .............................. 77
TIFF ......................................................... 529
UCS ........................................................... 530
Unicode ..................................................... 530
Unterschneidung ....................................... 524
URL .......................................................... 527
UTF ........................................................ 529
BOM .......................................................... 530
UTF-16 ....................................................... 530
UTF-16BE .................................................... 530
UTF-16LE .................................................... 530
UTF-32 ....................................................... 530
UTF-32BE .................................................... 530
UTF-32LE .................................................... 530
UTF-8 ......................................................... 530
VCARD ....................................................... 527
Vektorgrafik .............................................. 531
Verknüpfung ............................................. 531
View ........................................................ 268, 321
XHTML ..................................................... 531
XML ........................................................ 532
Zahlensystem ............................................ 532
Zeichensatz ..............................................
ANSI ......................................................... 519
ASCII ....................................................... 519
ISO-8859-1 ................................................ 531
ISO-8859-15 .............................................. 531
UTF ........................................................ 530
Windows-1252 ......................................... 531
Zeilenummerierung .................................... 378
Zwischenablage ....................................... 532

Anhang: Stichwortverzeichnis 536
Zu diesem Buch:

Zu den Autoren:
Dieses Buch entstand durch Freiwillige der Libre-Office-Gemeinschaft.


Über LibreOffice:

LibreOffice entsteht aus der kreativen Zusammenarbeit von Entwicklern und Community der Stiftung "The Document Foundation".