LibreOffice
The Document Foundation

Base

Kapitel 2
Datenbank erstellen

Copyright

Dieses Dokument unterliegt dem Copyright © 2015. Die Beitragenden 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

Robert Großkopf Jost Lange Michael Niedermair

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 15.08.2021. Basierend auf der LibreOffice Version 7.2.

Inhalt

Allgemeines bezüglich der Erstellung einer Datenbank 4

Neue Datenbank als interne Datenbank 4

Zugriff auf externe Datenbanken 7

- MySQL/MariaDB-Datenbanken 8
  - Erstellen eines Nutzers und einer Datenbank 8
  - MySQL-Verbindung direkt mit der Extension 9
  - MySQL-Verbindung über JDBC 9
  - MySQL-Verbindung über ODBC 10
  - Verbindung zur MySQL-Datenbank über den Datenbank-Assistenten 11
  - Verbindung zu MySQL/MariaDB über das Internet 20
  - Zugriff auf gespeicherte Prozeduren in MySQL/MariaDB 21
- PostgreSQL 22
  - Erstellen eines Nutzers und einer Datenbank 22
  - Direkte Verbindung zu Base 22
  - PostgreSQL-Verbindung über JDBC und ODBC 24
  - Autoincrement-Werte bei PostgreSQL 25
- dBase-Datenbanken 26
- Tabellendokumente und Tabellen in Writer-Dokumenten 28
  - Daten in Calc bearbeiten und in Base aktualisieren 29
  - Tabellen aus dem Internet über Calc automatisch aktualisiert in Base auslesen 29
- Thunderbird Adressbuch 33
- Texttabellen 34
  - Texttabellen innerhalb einer internen HSQLDB-Datenbank 34
  - Texttabellen als Grundlage für eine eigenständige Datenbankdatei 37
- Firebird 39
  - Erstellen eines Nutzers und einer Datenbank 40
  - Firebird-Verbindung über JDBC 40
  - Firebird-Verbindung über ODBC 41
  - Direkte Verbindung zu einer Firebird-Datei 42
- SQLite 42
  - Erstellen einer Datenbank 42
  - SQLite-Verbindung über ODBC 43
  - Weitere SQLite-Verbindungen 43
- Zugriff auf Access 43

Nachträgliche Bearbeitung der Verbindungseigenschaften 44

Maskierung von Tabellennamen und Feldnamen bei unterschiedlichen Datenbanksystemen 48

Allgemeines bezüglich der Erstellung einer Datenbank

In LibreOffice gibt es das Programmmodul «Base». Dies stellt eine grafische Benutzeroberfläche für Datenbanken zur Verfügung. Daneben existiert, in LibreOffice eingebunden, die interne Datenbank «HSQLDB» sowie die interne Datenbank «Firebird». Diese internen Versionen der Datenbanken können nur als Datenbanken von einem Nutzer betrieben werden. Die gesamten Daten sind in der *.odb-Datei mit abgespeichert und diese Datei ist nur für einen Benutzer nicht schreibgeschützt zu öffnen.

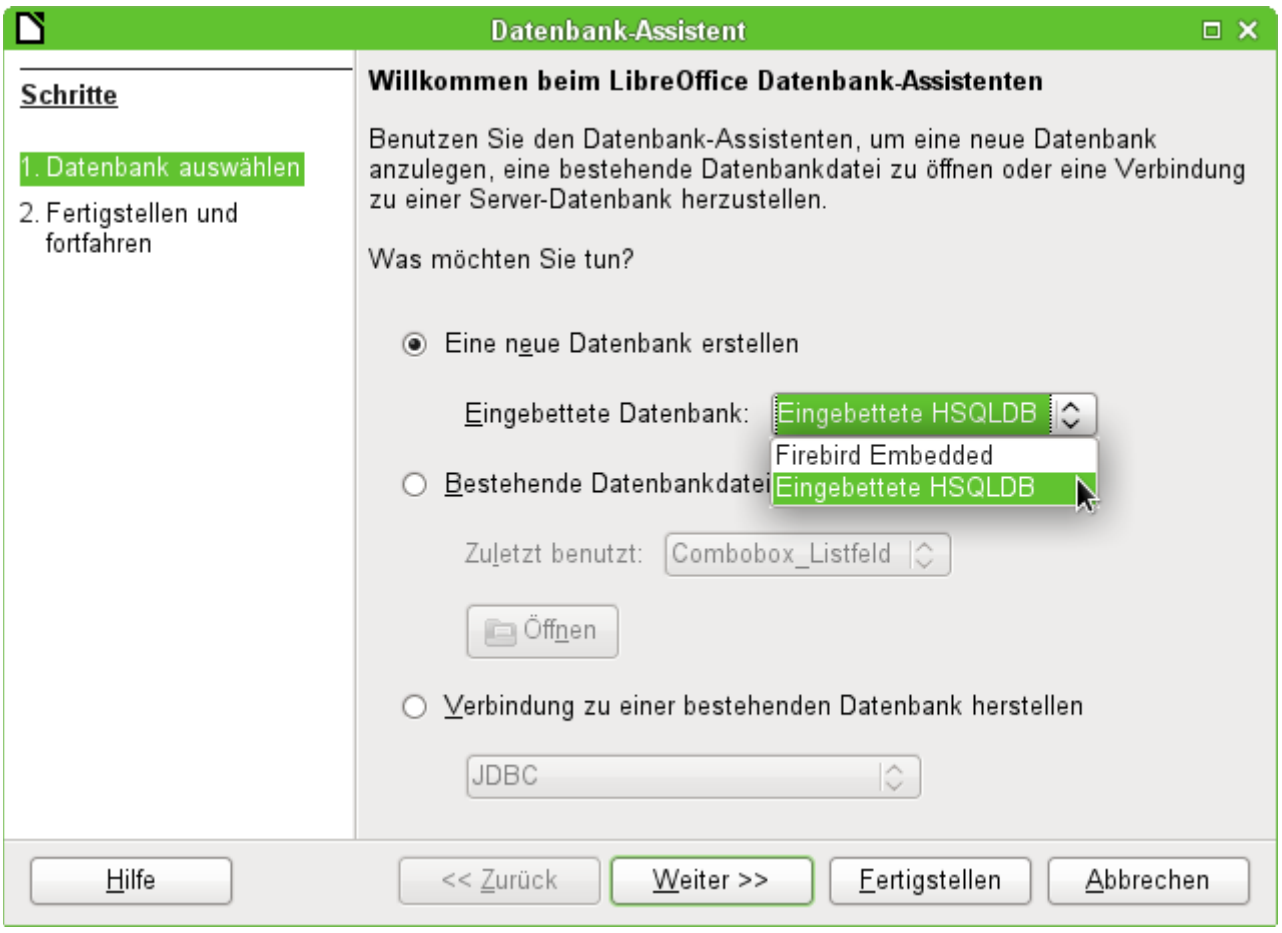
Neue Datenbank als interne Datenbank

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 Datenbank umwandeln, auf die dann auch mehrere User gleichzeitig zugreifen können, wenn der Datenbankserver läuft. Eine Beschreibung dazu in Bezug auf die HSQLDB erfolgt im Anhang dieses Handbuches im Kapitel «Datenbankverbindung zu einer externen HSQLDB».

Hinweis	Die interne HSQLDB benötigt (wie auch der ReportBuilder und die Assistenten) zwingend Java. Häufigster Fehler bei der Erkennung von Java durch LibreOffice ist, dass Nutzer LibreOffice in einer 64bit-Version installiert haben, Java aber in der 32bit-Version. LibreOffice und Java müssen in der gleichen Version vorliegen.

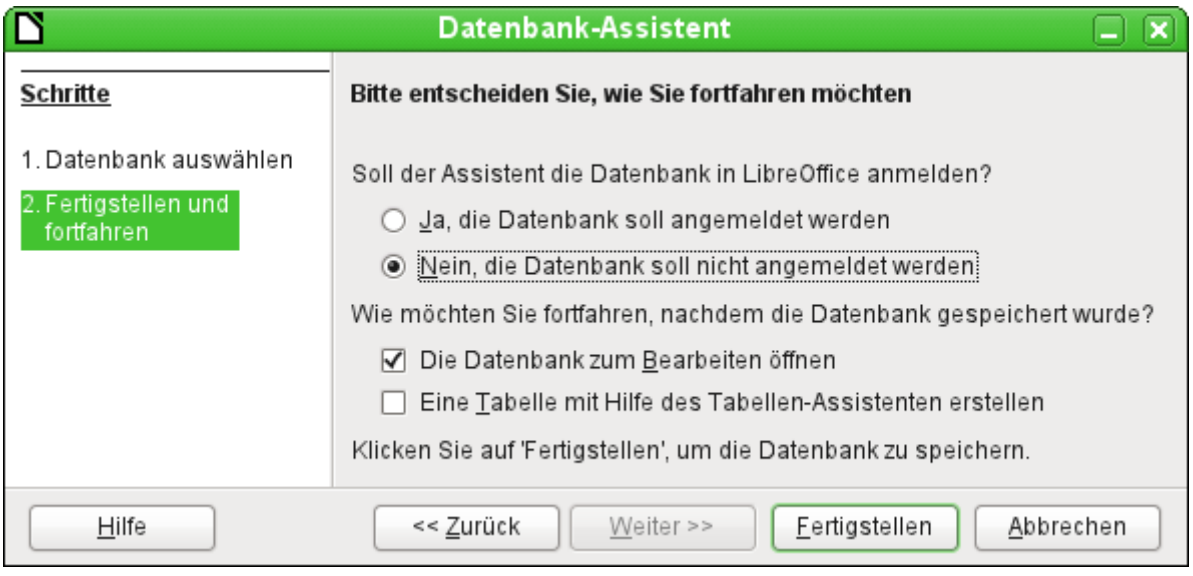
Hinweis	Bei der Nutzung großer Datenmengen kann es mit **Java** zu Geschwindigkeitsverlusten führen. Dies kann von der verwendeten Java-Version abhängen. Ein einfacher Test bei direktem Ausführen von SQL ergibt für eine Tabelle von 15 Spalten und 30.000 Datensätzen deutliche Unterschiede für die Zeit, die zum Scrollen zum letzten Datensatz notwendig sind. Die interne HSQLDB braucht hier fast 30 Sekunden, Firebird hingegen schafft das in 1 Sekunde. Auch bei externen Datenbanken lässt sich das in Abhängigkeit von dem Treiber beobachten. Die MariaDB mit der direkten Verbindung von LibreOffice aus liegt hier mit Firebird gleichauf. Die gleiche Datenbank, jetzt aber über JDBC mit LibreOffice verbunden, ist genauso langsam wie die interne 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, bis Version LO 6.0 müssen dafür aber noch die experimentellen Funktionen aktiviert werden. Auch ab LO 6.4.3 ist dies wieder notwendig, da die Integration weiter Probleme bereitet.

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.



Eine in LibreOffice angemeldete Datenbank kann von den anderen Programmteilen von LibreOffice als Datenquelle genutzt werden (z. B. Serienbrief). Diese Anmeldung kann aber auch

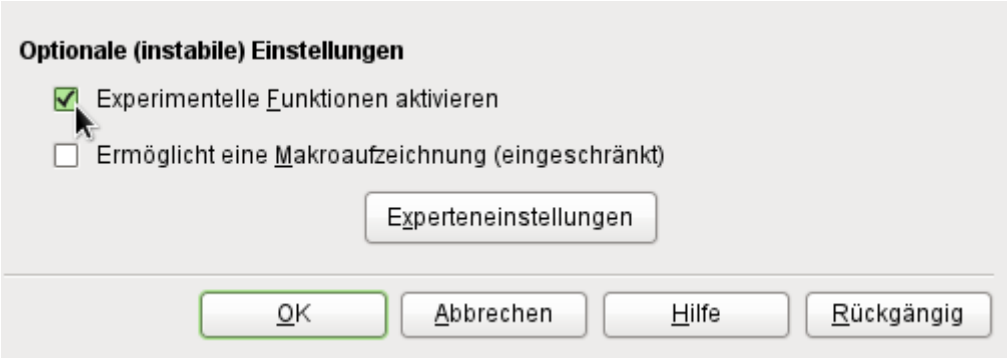
zu einem späteren Zeitraum erfolgen. Deshalb ist an dieser Stelle erst einmal das Optionsfeld **Nein, die Datenbank soll nicht angemeldet werden**, ausgewählt.

Bei den Markierfeldern ist lediglich «Die Datenbank zum Bearbeiten öffnen» markiert. Assistenten zur Erstellung von Tabellen, Abfragen usw. werden in diesem Handbuch nicht genutzt. Ihre Funktion ist lediglich für den einfachen Einstieg in die Materie geeignet.

Die Datenbank wird fertiggestellt, indem direkt noch einem Namen und einem Speicherort gefragt wird. An der entsprechenden Stelle 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. Diese Datenbank könnte in Zukunft die interne HSQLDB ersetzen. Zur Vereinfachung des Wechsels wird an einem Migrationstool gearbeitet. Dieses Tool bietet die Migration von HSQLDB zu Firebird an, wenn über **Extras → Optionen → LibreOffice → Erweitert → Optionale (instabile) Einstellungen** die experimentellen Funktionen aktiviert werden:



Die HSQLDB ist auch für LibreOffice 7.0 die Standarddatenbank für jeden Nutzer; die Firebird-Datenbank hat mit LO 5.3 ein Update auf die aktuelle Firebird-Version 3.0 vollzogen, ist aber bisher nur experimentell zu empfehlen. Die Datenbank-Beispiele beziehen sich weiterhin auf die HSQLDB, sind aber so angepasst, dass die meisten Funktionen direkt auf Firebird übertragbar sind. Gegebenenfalls werden Alternativen in Firebird direkt aufgezeigt.

Hinweis	Die Nutzung von Firebird ist mit der Version LO 6.4.3 wieder in den experimentellen Status zurückversetzt worden. Einige Fehler der internen Version sind so beschaffen, dass sie bei Abfragen gleich die ganze Datenbank zum Absturz bringen lassen und den Inhalt von Tabellen erst nach einem Neustart von Base wieder anzeigen.

Vorsicht

Mit den experimentellen Funktionen erscheint bei jedem Öffnen einer internen HSQLDB-Datenbank (über den Zugriff auf den Tabellencontainer) ab LO 6.1 der folgende Dialog:

Hier ist äußerste Vorsicht geboten! Später drücken und dann erst einmal eine Sicherungskopie der Datenbankdatei machen. Auf keinen Fall einfach bestätigen! Die Migration funktioniert nicht einwandfrei, kann dies auch vermutlich nie.

1. Sicherheitskopie der HSQLDB-Datenbankdatei anfertigen.
2. Funktionen nach der im Angang des Handbuches unter «Migration HSQLDB → Firebird» stehenden Liste so weit wie möglich anpassen.
3. Ansichten, die nicht direkt umgestellt werden können, vom SQL-Code her kopieren und als Abfragen abspeichern.
4. Tabellennamen und Spaltennamen dürfen in Firebird nur maximal 31 Zeichen lang sein. Gegebenenfalls also anpassen.
5. Reine Texttabellen (eingebundene *.csv-Tabelle etc.) sind unter Firebird nicht möglich, müssen also anderweitig ersetzt werden.

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
- MySQL
- ODBC
- Oracle JDBC
- PostgreSQL
- Firebirddatei
- Tabellendokument

- Text
- sowie verschiedene Adressbücher

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.

Bei manchen Datenbanktypen können keine neuen Daten eingegeben werden. Sie sind deshalb nur zum Suchen von Daten geeignet (z.B. Tabellendokument, Adressbücher). Die Beschreibungen in den folgenden Kapiteln beziehen sich ausschließlich auf die Verwendung von LibreOffice Base mit der internen Datenbank HSQLDB. Die meisten Ausführungen lassen sich auf Datenbanken wie MySQL, PostgreSQL etc. übertragen und entsprechend anwenden.

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

Hinweis	In Abhängigkeit vom benutzten Treiber ist es möglich, gleichzeitig auf mehrere Datenbanken des jeweiligen Servers zuzugreifen. Wird z.B. in MySQL/MariaDB mit der direkten Verbindung ein Kontakt zu einer Datenbank hergestellt, so werden in Base gleichzeitig alle anderen Datenbanken auf dem Server angezeigt, bei denen der angegebene Nutzer Leserecht hat. Es ist also problemlos möglich, Daten von einer Datenbank zu einer anderen zu kopieren, zusammenhängende Abfragen durchzuführen usw.

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.

Hinweis	In MySQL und MariaDB ist es möglich, Daten in Tabellen ohne ein Primärschlüsselfeld einzugeben und zu ändern. Die GUI von Base zeigt diese Tabellen zwar an, bietet aber keine Eingabe- bzw. Änderungsmöglichkeit. Wer Tabellen ohne Primärschlüssel verwenden möchte, kann stattdessen über **Extras → SQL**, oder innerhalb von Formularen über Makros, die Tabellen mit Daten versorgen.

Erstellen eines Nutzers und einer Datenbank

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

1. Der Administrationsuser in MySQL heißt «root». Für Linuxnutzer ist hier wichtig, dass es sich dabei nicht um den Administrator des Linuxsystems handelt. Diesem Nutzer wird direkt nach der Installation erst einmal ein Passwort zugewiesen. Dies kann je nach System auch schon vorher bei der Installation erfolgt sein. In der Konsole wird nach der Anmeldung als «sudo»
   `mysql -u root -p`
   eingegeben. Am Anfang ist kein Passwort vorhanden, so dass bei der Eingabeaufforderung auch nur *Enter* gedrückt wird. Gegebenenfalls muss **`-p`** für die Passworteingabe auch weg gelassen werden.
   Es erscheint die Eingabeaufforderung
   `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.
```
SET PASSWORD FOR root@localhost=PASSWORD('Passwort');
SET PASSWORD FOR root@host=PASSWORD('Passwort');
```
Windows-Nutzer geben statt der zweiten Zeile
```
SET PASSWORD FOR root@'%'=PASSWORD('Passwort');
```
ein.

2. Als Sicherheitsmaßnahme werden alle eventuell bestehenden anonymen Nutzer gelöscht.
```
DELETE FROM mysql.user WHERE User='';
DELETE FROM mysql.db WHERE User='';
FLUSH PRIVILEGES;
```
3. Eine Datenbank mit dem Namen «libretest» wird erstellt.
```
CREATE DATABASE libretest;
```
4. Alle Rechte an der Datenbank «libretest» werden dem Nutzer «lotest» übergeben, der sich durch das Passwort «libre» ausweisen soll.
```
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

Hinweis	Seit LO 6.2 ist die direkte Verbindung zu MySQL/MariaDB in LibreOffice integriert. Eine Installation der Erweiterung wie im folgenden beschrieben ist nicht mehr erforderlich.

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* (Stand: 1/11) zu finden.
- Eine auch unter anderen Systemen lauffähige Erweiterung für Versionen nach 3.3.4 bis einschließlich 4.0.* steht unter *http://extensions.openoffice.org/en/project/aoo-my-sdbc* zum Download.
- Für Version LO 4.0 ist eine Extension für Windows über *https://dl.dropboxusercontent.com/u/193133/mysql-connector-ooo.oxt* downloadbar.
- Für Version LO 4.1 und 4.2 ist eine Extension für Windows über *http://extensions.libreoffice.org/extension-center/mysql-native-connector-for-libreoffice-4.x* downloadbar.
- Für die Versionen ab 4.1 ist eine Extension für Linux über *http://extensions.libreoffice.org/extension-center/mysql-native-connector* downloadbar.

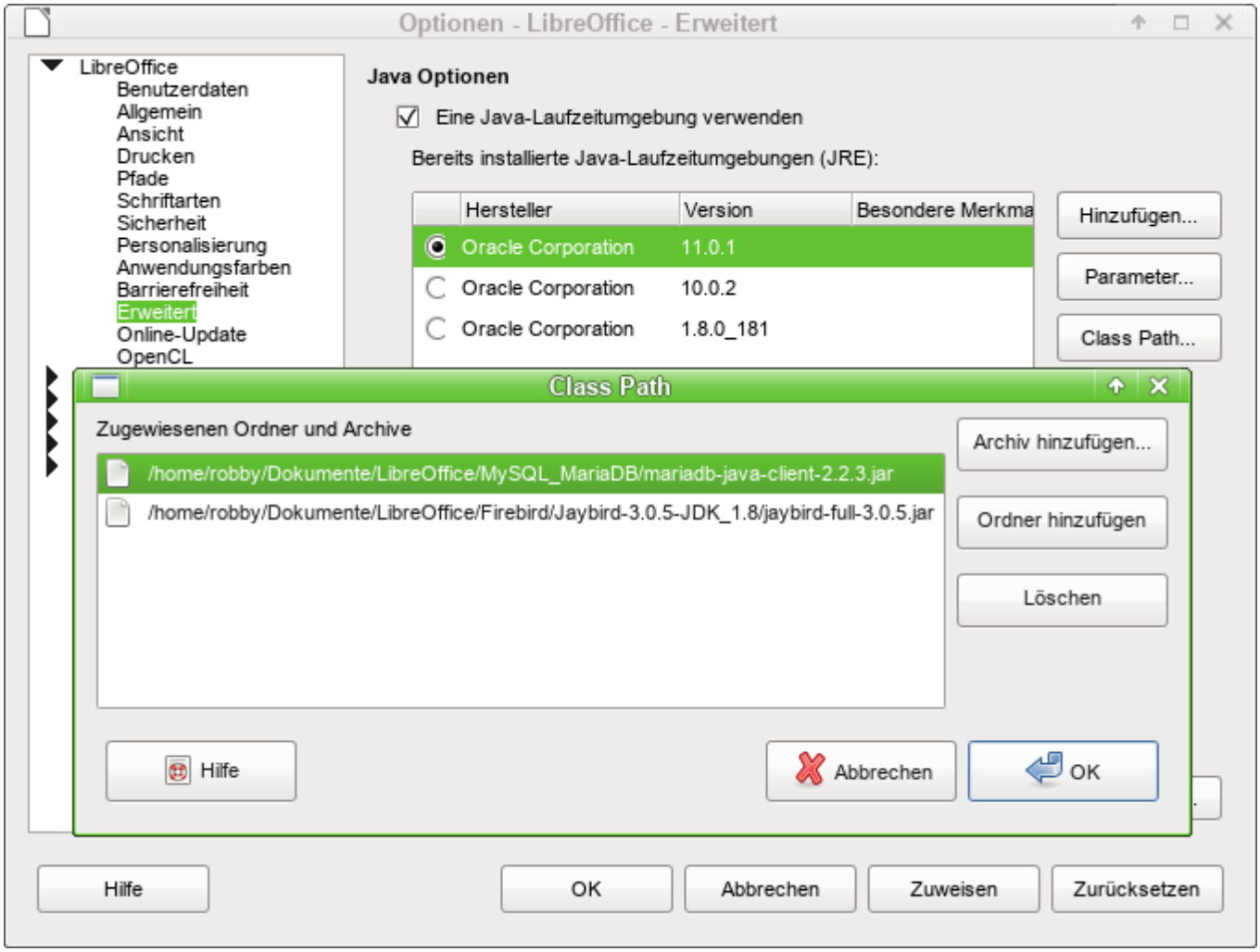
Vorsicht	Ein Kontakt zu LO 4.1.* kommt zwar mit dem AOO-Connector zustande, aber da sich ab der Version LO 4.1 die Beschreibung der Sekundenbruchteile von LO und AOO unterscheiden, werden falsche Werte im Bereich der Zeiten und auch von kompletten Timestamp-Feldern angezeigt. Verwenden sie also bitte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

Als allgemeiner Zugang zu MySQL ist der Zugang über JDBC oder ODBC zu wählen. Um den JDBC-Zugang nutzen zu können, wird der mysql-connector-java.jar in der jeweils zur Datenbank

passenden Fassung benötigt. Für MySQL kann das entsprechende Paket, das diesen Connector enthält, hier herunter geladen werden: *https://dev.mysql.com/downloads/connector/j/* . Dieses Java-Archiv wird am besten in das Verzeichnis kopiert, aus dem auch die aktuelle Java-Version von LibreOffice geladen wird. Als Speicherort bietet sich das Unterverzeichnis «...Javapfad.../lib/ext/» an. In Linux-Systemen wird dies durch die Softwareverwaltung oft automatisch erledigt.

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



Über **Class Path → Archiv hinzufügen…** wird die zu der Datenbank passende *.jar-Datei gesucht und in LibreOffice eingebunden.

Der Treiber wird bis Version 5.1 mit **`com.mysql.jdbc.Driver`** angesprochen. Neuere Treiber benötigen den Eintrag **`com.mysql.cj.jdbc.Driver.`**

Für die MariaDB bietet sich der JDBC-Treiber von *https://downloads.mariadb.com/Connectors/java/* an. Der Treiber kann ebenso in den Class-Pfad eingebunden werden und wird mit **`org.mariadb.jdbc.Driver`** angesprochen.

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.1 nicht findet. Hier existiert in den meisten Systemen inzwischen die libodbc.so.2 . Auf diese Datei muss ein entsprechender Verweis mit dem Namen libodbc.so.1 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/lib64/libmyodbc5.so
```

odbc.ini

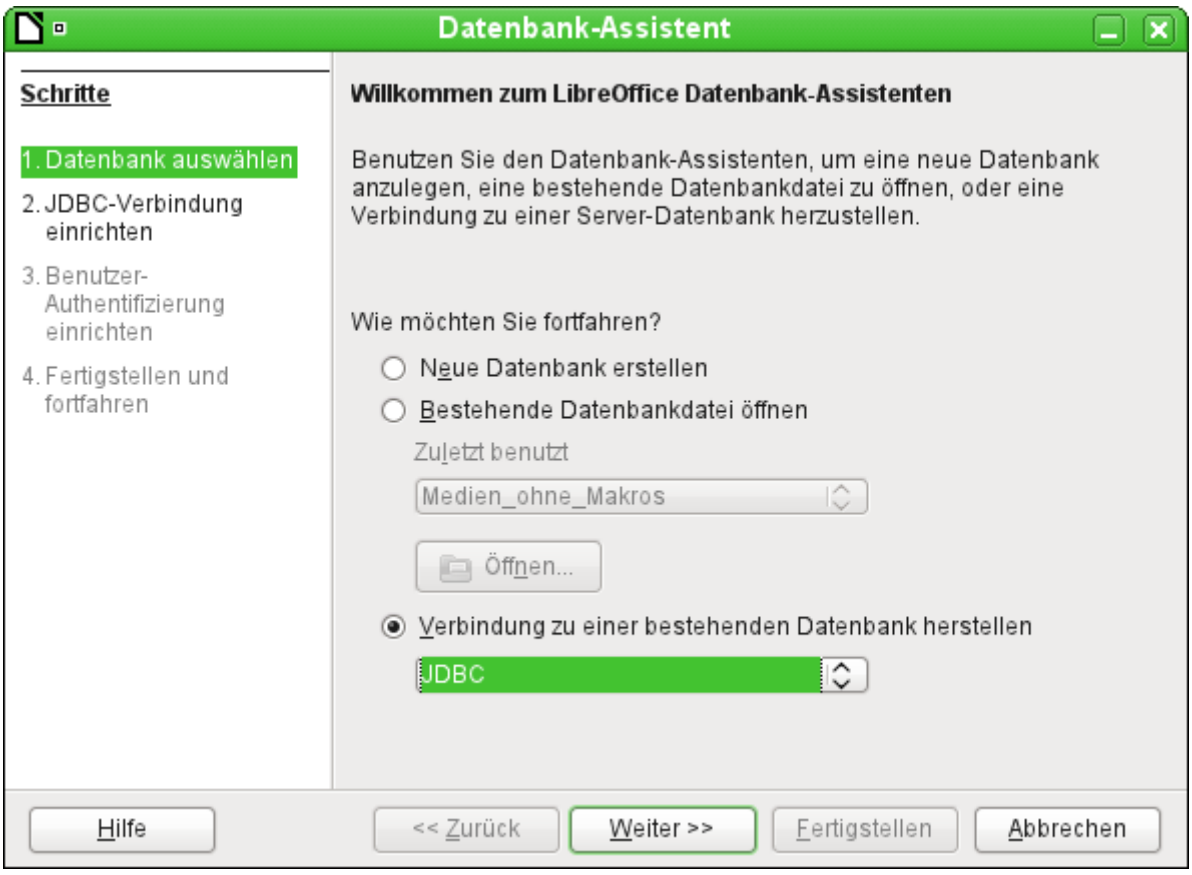
```
[MySQL-test]
Description = MySQL database test
Driver = MySQL
Server = localhost
Database = libretest
Port = 3306
Socket =
Option = 3
Charset = UTF8
```

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:



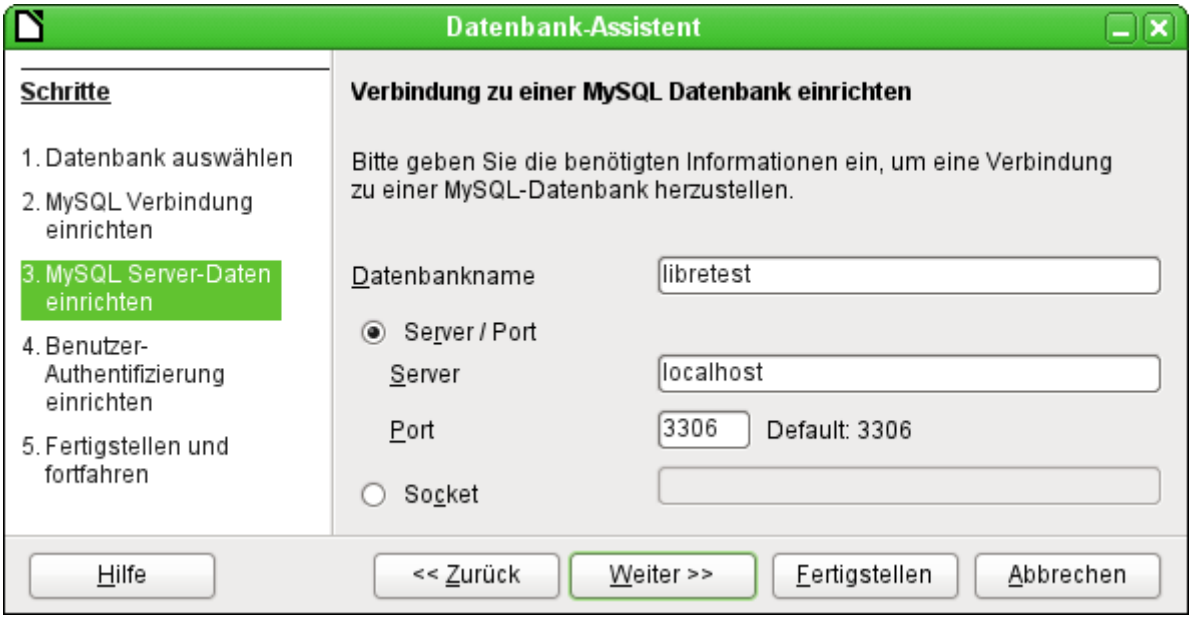
Über «*Neue Datenbank erstellen*» ist nur die Erstellung einer Datenbank im internen HSQLDB-Format möglich. Die Zusammenarbeit mit anderen Datenbanken kann nur erfolgen, wenn die Datenbank selbst bereits existiert. Dazu muss also **Verbindung zu einer bestehenden Datenbank herstellen** gewählt werden.

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[1] MySQL-Connector als direkte Verbindung installiert ist oder unterstützt wird. Ist die Erweiterung für eine direkte Verbindung installiert, so wird «Direkte Verbindung» standardmäßig voreingestellt.

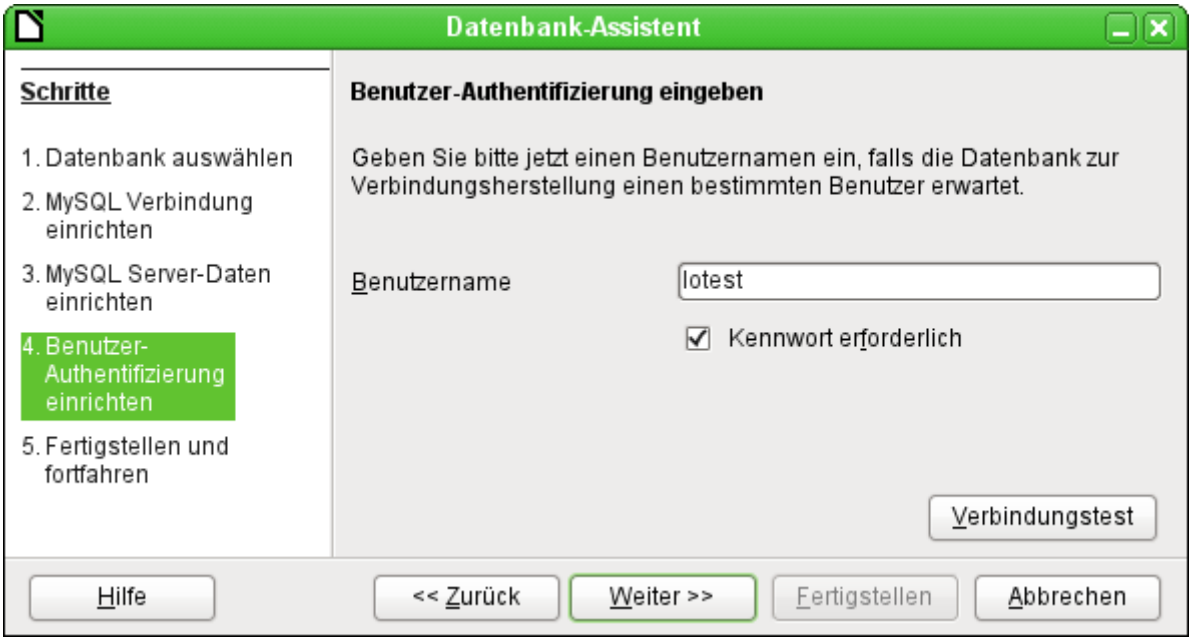
Die direkte Verbindung ist diejenige, die von der Geschwindigkeit her am besten gewählt werden sollte. Allerdings hat sie auch ein paar kleine Bugs, die gegebenenfalls besonders berücksichtigt werden müssen. So ist es z. B. dringend erforderlich, in Formularen bei Textfeldern die maximale Länge einzustellen, weil sonst die Eingabe einfach auf den bisher maximal eingegebenen Wert gekürzt wird.



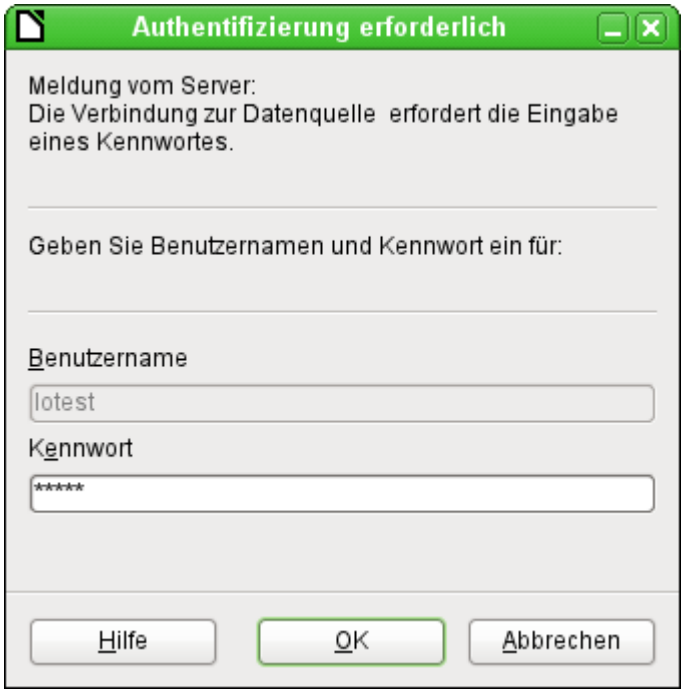
Der Datenbankname muss bekannt sein. Befindet sich der Server auf dem gleichen Rechner wie die Benutzeroberfläche, unter der die Datenbank erstellt wird, so kann als Server «localhost» gewählt werden. Ansonsten kann die IP-Adresse (z.B. 192.168.0.1) oder auch je nach Netzwerkstruktur der Rechnername 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.

1 «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?

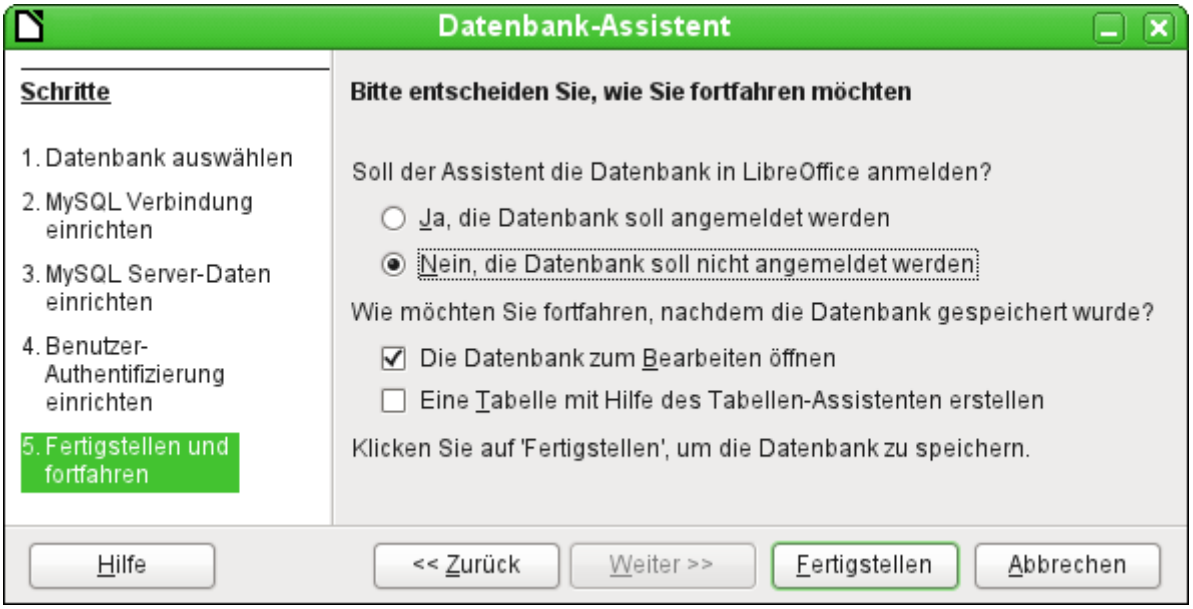


Jede über das Netz erreichbare Datenbank sollte mit einem Benutzernamen und einem Kennwort geschützt sein. Hier wird direkt getestet, ob die Verbindung klappt. Wichtig ist natürlich, dass der entsprechende Benutzer in MySQL bzw. der MariaDB entsprechend für den benannten Server eingerichtet wurde.



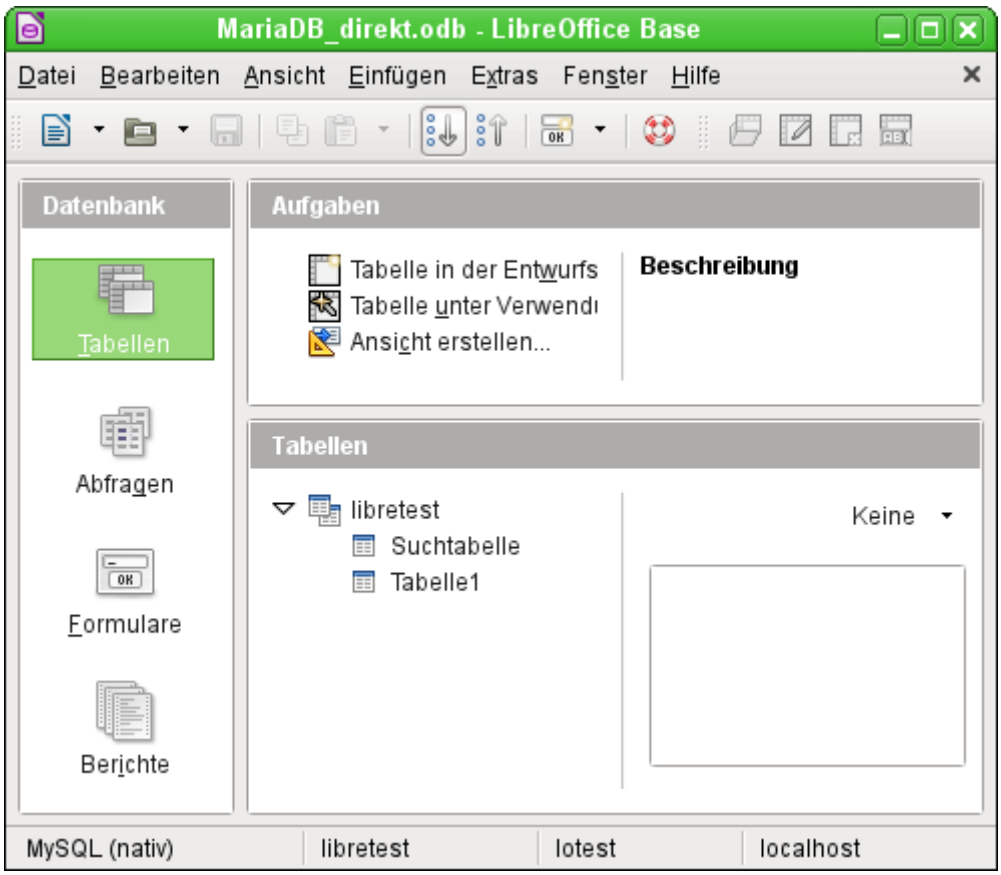
Dieses Fenster erscheint anschließend bei jedem Start der Datenbankdatei, wenn das erste Mal auf die MySQL-Datenbank zugegriffen wird.

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, für die der Nutzer mindestens eine Leseberechtigung hat.

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" ...
```

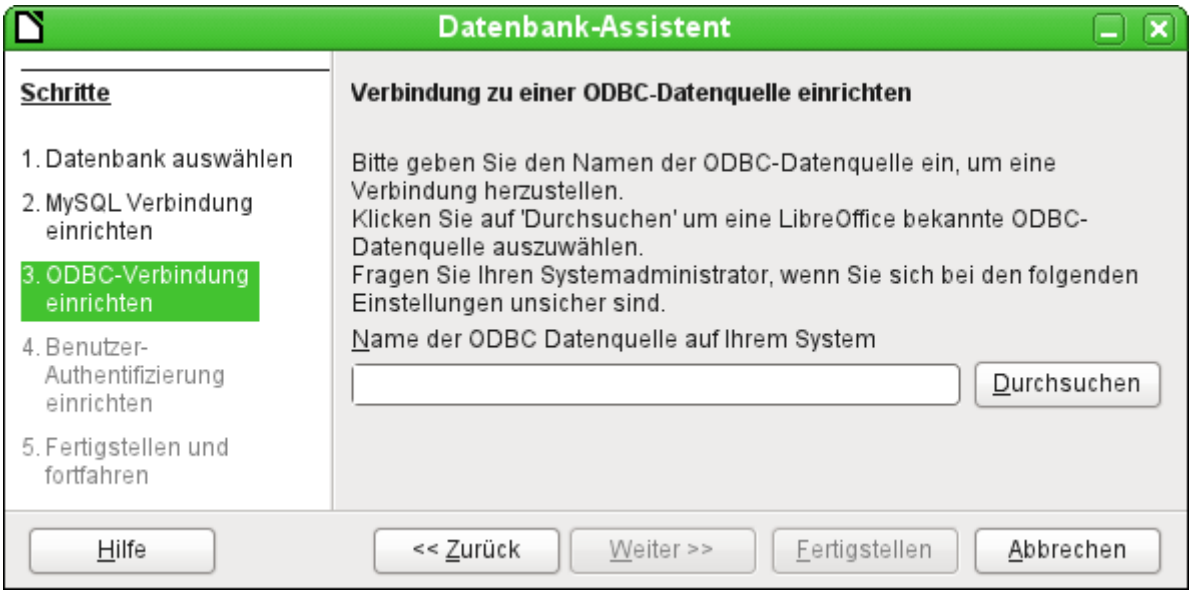
Hier ist es also auf jeden Fall notwendig, der Kombination aus Datenbankname und Tabellenname mit dem Zusatz «AS» einen alternativen Alias-Namen zuzuweisen. Genaueres siehe dazu in dem Kapitel «Verwendung eines Alias in Abfragen».

Hinweis	Durch diese Aliaszuweisung werden Abfragen, die aus mehreren Tabellen bestehen, nicht mehr editierbar, auch wenn alle anderen Bedingungen erfüllt sind. Dann muss in der durch die GUI erstellten Abfrage im SQL-Code **`"test"."Klasse" AS`** (also der Hinweis auf den Datenbanknamen) entfernt werden. Dann gelingt auch eine bearbeitbare Abfrage über mehrere Tabellen.

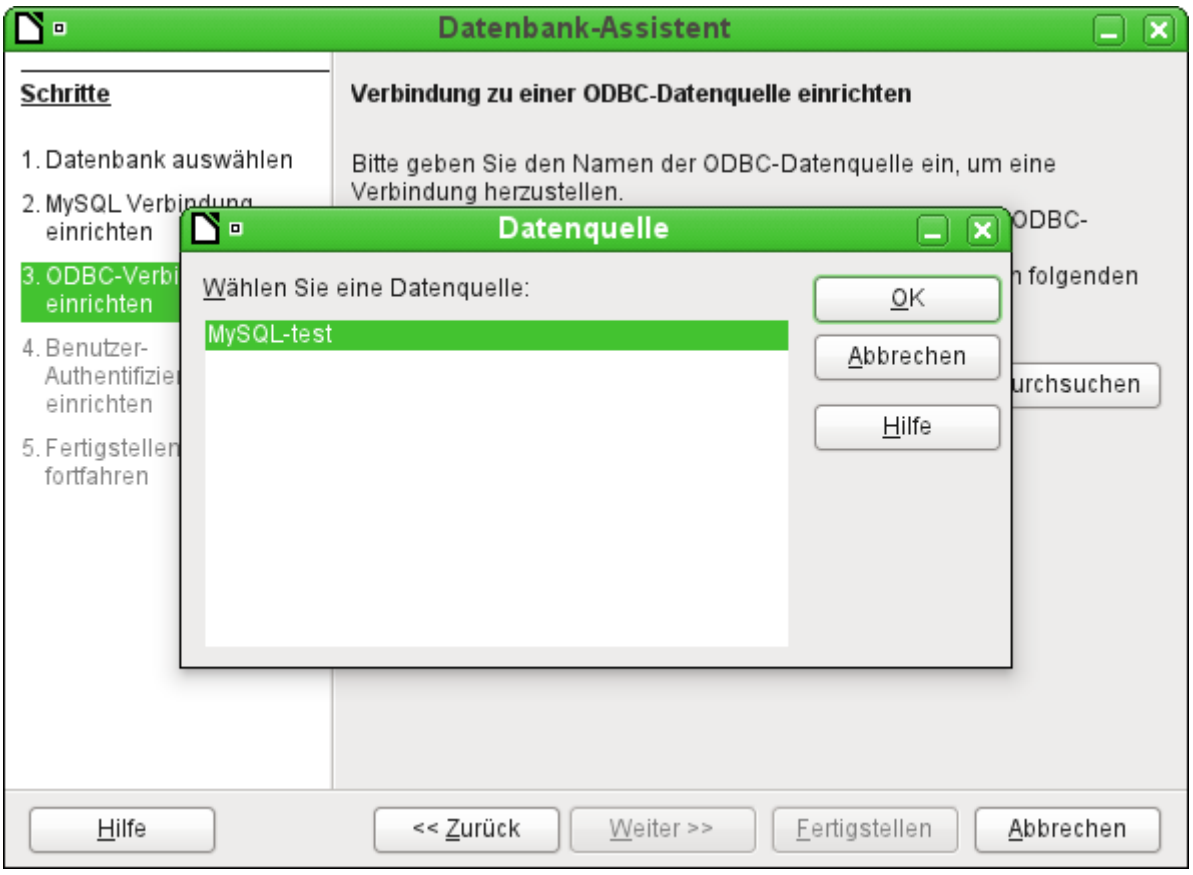
Tabellen können in der Datenbank erstellt und gelöscht werden. Automatisch hochzählende Zahlenfelder («Autowert») funktionieren und lassen sich auch bei der Tabellenerstellung auswählen. Sie starten bei MySQL mit dem Wert 1.

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:



Die ODBC Datenquelle hat nicht unbedingt den gleichen Namen wie die Datenbank in MySQL selbst. Hier muss der Name eingetragen werden, der auch in der Datei «odbc.ini» steht. Die einfachste Möglichkeit besteht hier darin, über den Button Durchsuchen den Namen aus der «odbc.ini» direkt auszulesen.

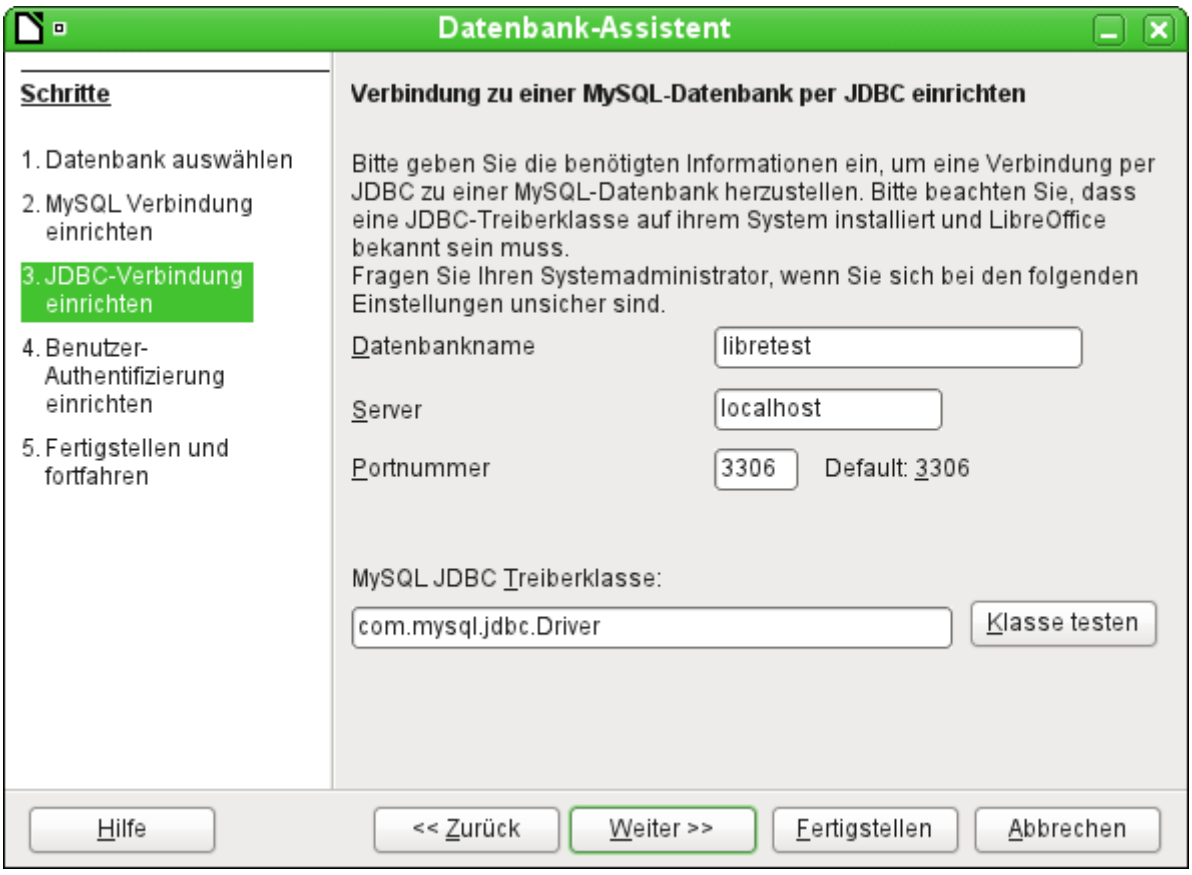


Beim Test erscheint der Name aus der «odbc.ini». Auch wenn hier wieder nur mit einer Datenbank verknüpft wird, können andere Tabellen des MySQL-Servers sehr wohl gelesen werden.

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.

Über «Klasse testen» wird überprüft, ob das Archiv mysql-connector-java.jar über Java erreichbar ist. Entweder muss dieses Archiv im Pfad der ausgewählten Java-Version liegen oder direkt in LibreOffice eingebunden werden.

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

Hinweis	Bei Verwendung des MariaDB-Treibers muss die JDBC-Treiberklasse mit «org.mariadb.jdbc.Driver» angegeben werden

Hinweis	Der JDBC-Zugang mit MySQL/MariaDB kann auch direkt über die Auswahl **JDBC** statt des Untermenüs **MySQL → JDBC** erfolgen. Dies kann besonders dann sinnvoll sein, wenn dem Treiber Parameter mitgegeben werden sollen. Ohne Parameter trennt z.B. der JDBC-Treiber wie auch der direkte Treiber zu einer MySQL-Datenquelle im Internet nach recht kurzen Pausenzeiten die Verbindung. Mit der folgenden Einstellung wird dies vermieden: jdbc:mysql://«Host der Datenbank»:3306/«Datenbankname»?autoReconnect=true Bei neueren Treibern (ab Version 8.*) für MySQL ist die Verbindung wegen einer Zeitzoneneinstellung nur über die Angabe von Parametern möglich, wenn nicht Servereinstellungen beeinflusst werden können: jdbc:mysql://localhost/«Datenbankname»?autoReconnect=true&useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC Dies bietet alle erforderlichen Einstellungen, zusätzlich auch noch die Einstellung des Zeichensatzes. Eine Übersicht aller Einstellungen bietet *https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-reference-configuration-properties.html#connector-j-reference-set-config*

Verbindung zu MySQL/MariaDB über das Internet

Die verschiedenen Verbindungen zu MySQL/MariaDB wurden oben jeweils mit der Beziehung zum örtlichen Server «localhost» getestet. Sie funktionieren grundsätzlich auch als Verbindungen zu Datenbanken eines Providers im Internet. Allerdings ist hier zu beachten, dass der Server des Providers für Kontakte von außerhalb ein **`wait_timeout`** eingestellt hat. Nach einer entsprechenden Zeit ohne Datenaustausch wird dann die Verbindung unterbrochen. Je nach Einstellung des Servers kann es erlaubt sein, diese Zeitspanne zu beeinflussen. Wenn über **Extras → SQL** die Eingabe

```
SET SESSION wait_timeout=600;
```

erfolgreich abgeschlossen wird, dann beträgt die Zeitspanne jetzt 600 Sekunden oder 10 Minuten. Sobald in dieser Zeit wieder eine Abfrage läuft wird die Zeitspanne bis zur nächsten Unterbrechung wieder auf 10 Minuten gesetzt.

Ist die Einstellung der Zeitspanne nicht möglich, so hilft das folgende Makro:

```
GLOBAL boStop AS BOOLEAN

SUB Reconnect
    DIM oDatasource AS OBJECT
    DIM oConnection AS OBJECT
    DIM oSQL_Command AS OBJECT
    boStop = false
    DO
        WAIT 30000 'Zeitangabe in Millisekunden
        oDatasource = thisDatabaseDocument.CurrentController
        oConnection = oDatasource.ActiveConnection()
        oSQL_Command = oConnection.createStatement()
        oSQL_Command.executeQuery("SELECT NOW()")
    LOOP WHILE boStop = false
END SUB

SUB StopConnect
    boStop = true
END SUB
```

Zuerst wird eine globale Variable erstellt, damit die Schleife in der Prozedur **`Reconnect`** auch unterbrochen werden kann. In der Prozedur wird alle 30 Sekunden eine einfache Abfrage gestellt,

die nicht weiter ausgewertet wird. So bleibt der Kontakt zum Server erhalten, wenn **`wait_timeout`** größer als 30 Sekunden ist und die Netzverbindung einwandfrei funktioniert. Die Einstellung der Zeitspanne ist hier dem Test überlassen. Mit der Prozedur **`StopConnect`** schließlich lässt sich die Prozedur **`Reconnect`** wieder unterbrechen.

Wird dieses Makro mit dem Öffnen der Datenbank aufgerufen, so muss natürlich innerhalb der ersten 30 Sekunden auf jeden Fall der erste Kontakt zur Datenbank mit Passworteingabe hergestellt werden.

Bei der JDBC-Verbindung lässt sich hier mit dem Parameter **`autoReconnect=true`** eine Verbindung direkt wieder herstellen. Die erste Unterbrechung findet dann trotzdem statt, bei einem zweiten Kontaktversuch ist die Verbindung aber wieder da.

Bei der direkten Verbindung funktioniert das nicht. Stattdessen muss die Datenbankdatei geschlossen und wieder geöffnet werden um erneut einen Kontakt herzustellen. Die Verbindungsdaten wie Nutzername und Passwort müssen aber nicht neu eingegeben werden.

Zugriff auf gespeicherte Prozeduren in MySQL/MariaDB

Auf dem Datenbankserver können neben Tabellen und Ansichten auch Prozeduren gespeichert werden. Werden diese Prozeduren direkt auf der Konsole von MySQL aufgerufen, so können dort zum Teil Tabellenansichten ähnlich einer sonst in Base sichtbaren Ansicht («View») abgerufen werden. In Base sind diese Tabellenansichten nicht sichtbar. Ein Aufruf der Prozeduren unter **Extras → SQL** ist problemlos möglich, nur erfolgt leider keine Ausgabe der anzuzeigenden Inhalte. Um Inhalte der gespeicherten Prozeduren («Stored Procedures») zu sehen kann der folgende Weg beschritten werden:

- Statt einer Prozedur wie
  ```
  CREATE PROCEDURE AlleNamen()
  BEGIN
     SELECT * FROM `Namen`;
  END
  ```
  wird das Ergebnis der Prozedur in eine temporäre Tabelle geschrieben:
  ```
  CREATE PROCEDURE AlleNamen()
  BEGIN
     DROP TEMPORARY TABLE IF EXISTS `TempNamen`;
     CREATE TEMPORARY TABLE `TempNamen` AS SELECT * FROM `Namen`;
  END
  ```
- Anschließend wird unter **Extras → SQL** die Prozedur aufgerufen:
  ```
  CALL AlleNamen();
  ```
- Jetzt befinden sich die Ausgabedaten in einer temporären Tabelle, die nur für den aktuellen Nutzer von MySQL sichtbar ist. Die Ausgabedaten werden in einer Abfrage über
  ```
  SELECT * FROM `TempNamen`
  ```
  sichtbar.

Wie das Ganze etwas besser automatisiert ablaufen kann ist im Kapitel «Makros», Unterkapitel «MySQL-Datenbank mit Makros ansprechen» beschrieben.

Hinweis	Mit der direkten Verbindung lässt sich zur Zeit (LO 5.4.1.2) eine gespeicherte Prozedur wie die oben geschilderte nur einmal aufrufen. Danach kommt die Fehlermeldung **`Commands out of sync; you can't run this command now.`** Die JDBC-Verbindung zur MySQL- bzw. MariaDB ist davon nicht betroffen. Vor der Nutzung von gespeicherten Prozeduren sollte also wenigstens unter **Extras → SQL** zweimal hintereinander die Prozedur mit CALL … testweise abgerufen werden.

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.

1. Dem Nutzer «postgres» muss zuerst ein Passwort zugewiesen werden. Das kann mit dem Administrationstool des Betriebssystems erfolgen.
2. Der Postgre-Server muss vom Administartor gestartet werden, z. B. **`service postgresql start`** oder **`rcpostgresql start`**.
3. Der Nutzer «postgres» muss sich mit
   `su postgres`
   auf der Konsole anmelden.
4. Ein einfacher Datenbanknutzer, hier «lotest», sollte mit Passwortzugang erstellt werden:
   `createuser -P lotest`
5. Damit der Datenbanknutzer anschließend auch auf die zu gründende Datenbank zugreifen kann, muss in der Datei /var/lib/pqsql/data/**pg_hba.conf** ein Eintrag geändert werden. In dieser Datei werden u.a. die Methoden zur Identifizierung der Nutzer auf verschiedenen Ebenen festgelegt. Die Methode, mit der LO-Base kommunizieren kann, ist die Methode «password» oder «md5», nicht, wie voreingestellt, die Methode «ident».
6. Der Systemnutzer «postgres» meldet sich über «psql» an:
   `psql -d template1 -U postgres`
7. Der Systemnutzer «postgres» erstellt die Datenbank «libretest»:
   `CREATE DATABASE libretest;`

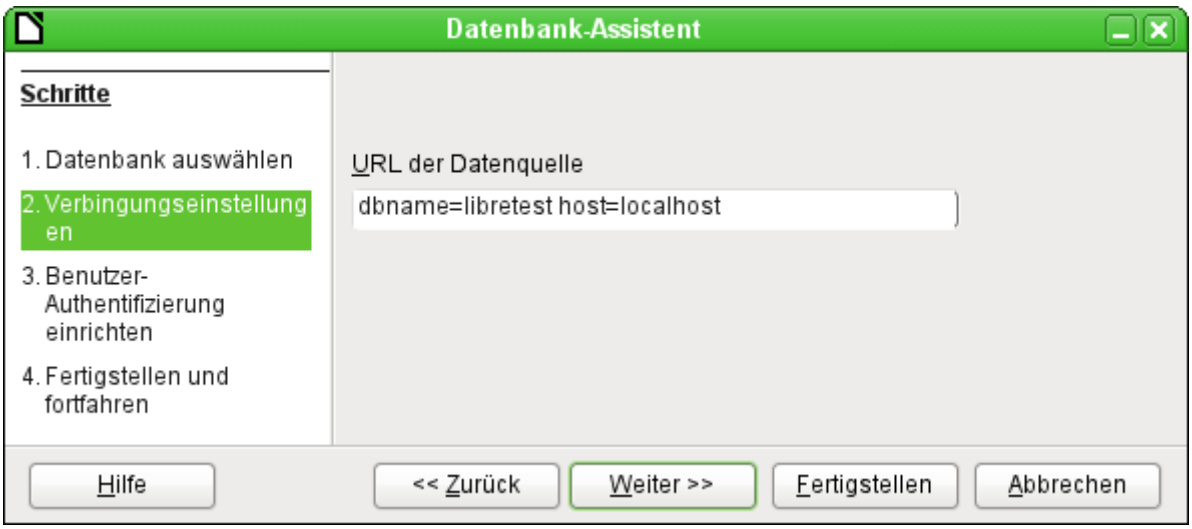
Der Nutzer «lotest» hat nicht die gleichen Rechte wie der Nutzer «postgres». Meldet sich der Nutzer «postgres» direkt beim system über su postgres an, so kann er beispielsweise zusätzliche Schemas für eine Datenbank erstellen:

1. Der Systemnutzer «postgres» meldet sich über «psql» an:
   `psql -d libretest -U postgres`
2. Der Systemnutzer «postgres» erstellt das Schema «reports»:
   `CREATE SCHEMA reports;`
3. Der Systemnutzer «postgres» setzt den Nutzer «lotest» als Eigentümer für das Schema ein:
   `ALTER SCHEMA reports OWNER TO lotest;`

Der Nutzer «lotest» kann jetzt auf dieses Schema zugreifen. Allerdings erscheint es nicht direkt in der Übersicht. Wird aber über die GUI in Base eine Tabelle erstellt, so kann diese in dem Schema «reports» gespeichert werden und anschließend wird das Schema auch in der Liste der Tabellen angezeigt.

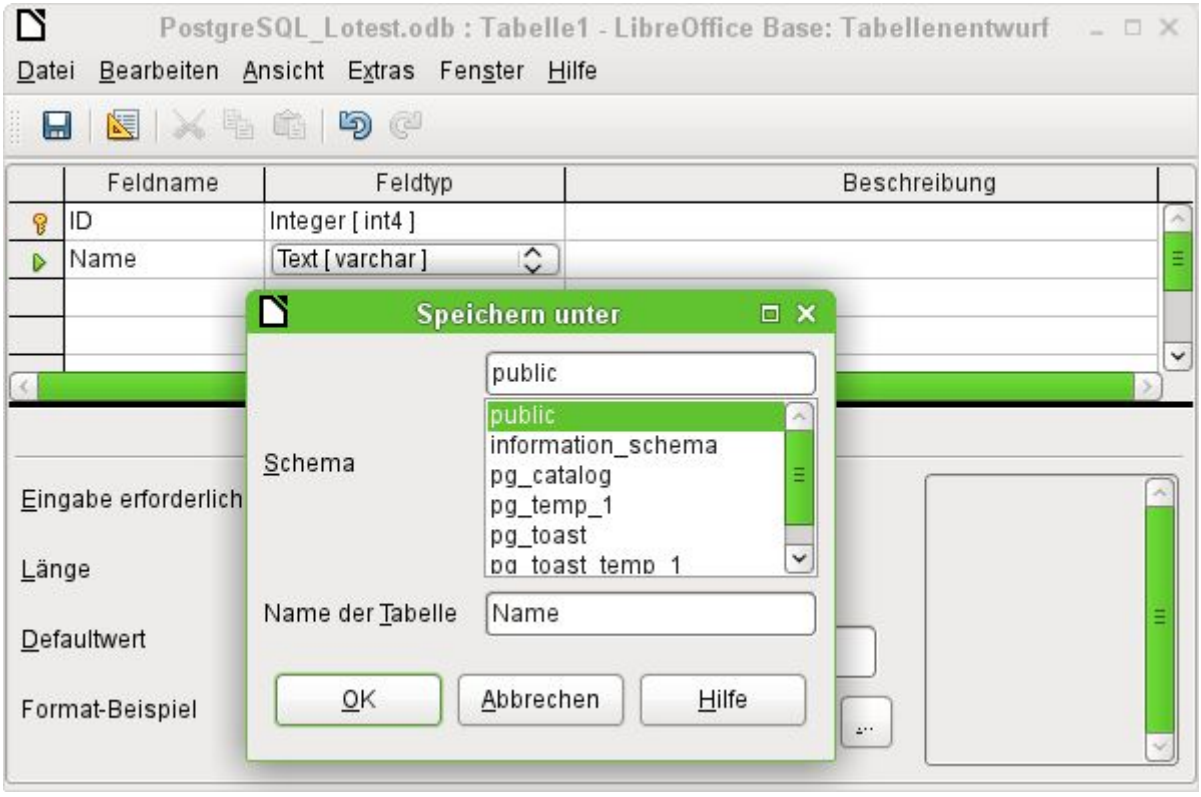
Direkte Verbindung zu Base

Im Assistenten wird der Eintrag «postgres» ausgewählt. Die Verbindungseinstellung geschieht über die Angabe des Datenbanknamens («dbname») und des Hostes. Unter bestimmten Umständen ist es erforderlich, statt der einfachen Angabe eines Hostes den kompletten Host einschließlich des vollständigen Domainnamens anzugeben.



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

Wird die Datenbank das erste Mal geöffnet, so erscheinen für den User erst einmal eine Unmenge an Tabellen aus dem Bereich «information_schema». Diese Tabellen sind ebenso wie die im Bereich «pg_catalog» für den normalen Nutzer nur lesbar, nicht schreibbar. Diese verschiedenen Bereiche bezeichnet PostgreSQL als «Schema». Der Nutzer legt neue Tabellen im Schema «public» an.

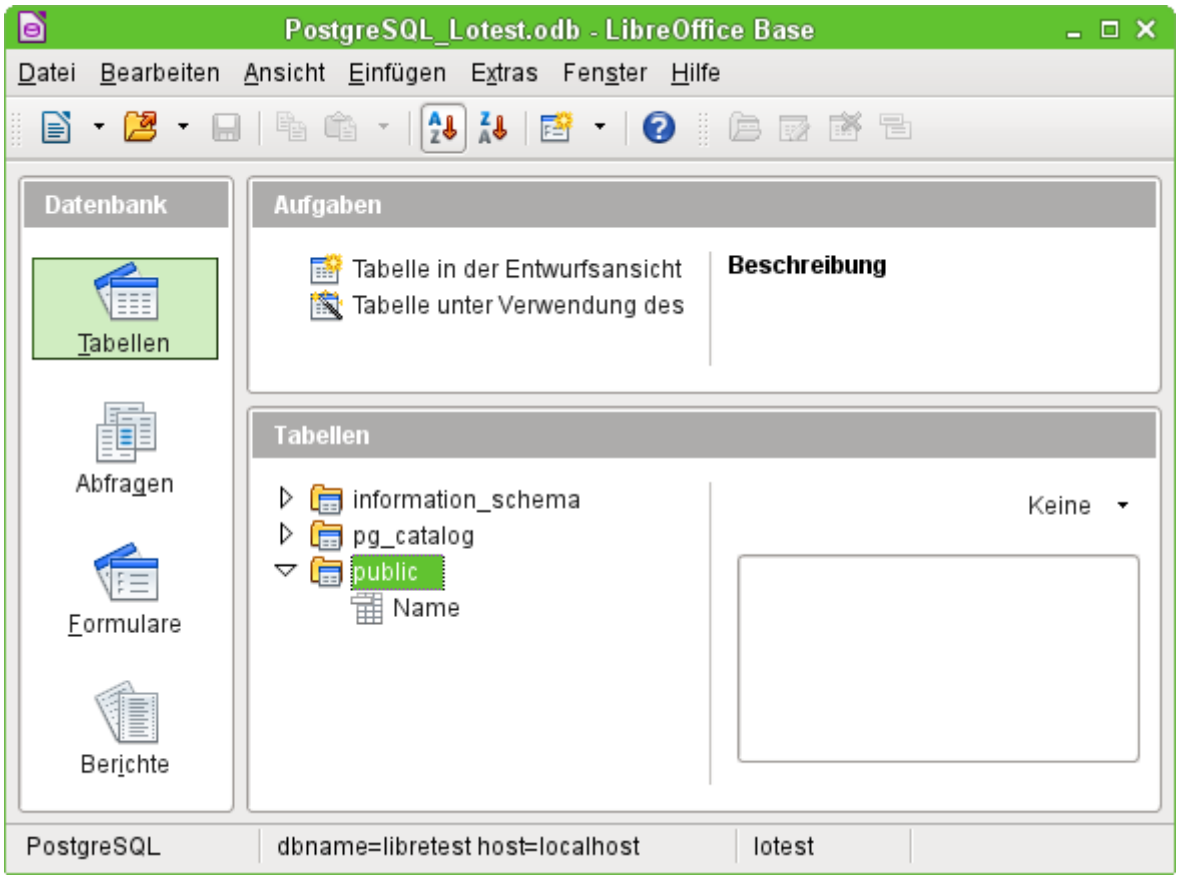


Der Dialog «Speichern unter» zeigt hier verschiedene Schema von PostgreSQL auf. Tatsächlich speicherbar ist allerdings nur in dem Schema «public», solange keine weitreichenderen Nutzerrechte für den entsprechenden Anmeldenamen vergeben wurden.

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».

Bei der Erstellung von Tabellen kann es vorkommen, dass Base Datentypen vorschlägt, die die aktuelle PostgreSQL-Installation nicht verarbeiten kann. So werden standardmäßig die Felder mit dem Feldtyp «Text [character_data]» vorgeschlagen. Den kann PostgreSQL aber nicht verarbeiten. Eine Änderung auf den Typ «Text [varchar]» (siehe oben) schafft hier Abhilfe.



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

Hinweis	Die Ansicht auf andere Verzeichnisse als das Verzeichnis "public" kann auch über **Extras → Tabellenfilter** eingeschränkt werden.

PostgreSQL-Verbindung über JDBC und ODBC

Neben dem direkten Treiber ist natürlich auch die Verbindung über JDBC oder ODBC möglich.

Während der direkte Treiber z. B. keine Erstellung von Ansichten in der GUI erlaubt oder immer mit der Tabellenansicht in dem Tabellenverzeichnis "information_schema" beginnt, lässt die **JDBC**-Verbindung die Erstellung von Ansichten zu und zeigt standardmäßig nur das Tabellenverzeichnis mit dem Schemanamen "public" oder andere selbst erstellte Schemata an. Nur in diesem Bereich werden schließlich neue Tabellen des Nutzers erstellt. Auch ist der JDBC-Treiber der einzige, der in der GUI automatisch hochschreibende Primärschlüsselfelder erzeugt.

Soll eine sichere Verbindung über SSH zu einer entfernten PostgreSQL-Datenbank erfolgen, so ist auch der **JDBC**-Treiber die erste Wahl. Dem direkten Treiber gelingt dies nicht, da er mit der Port-Bezeichnung nichts anfangen kann.

Der **JDBC**-Treiber wiederum hat Probleme bei der Verwendung von Abfragen mit Aliasbezeichnungen für Tabellen. Wenn ein Primärschlüssel automatisch hoch geschrieben werden soll, so wird dies bei Abfragen mit Aliasbezeichnungen nicht korrekt ausgelesen. Dies führt dann zu angeblichen 0-Werten als Primärschlüssel und Datenverlust, wenn nach dem ersten Abspeichern Daten in so einem Datensatz geändert werden.

Aus diesem Grund hier auch noch funktionierende Standardeinstellungen für die **ODBC**-Verbindung von PostgreSQL. Leider ist auch diese Verbindung nicht frei von Mängeln. So lassen sich hier über die GUI keine Ansichten erstellen, solange noch keine Ansicht existiert. Dem kann aber abgeholfen werden, indem eine erste Dummyansicht über **Extras → SQL** erstellt wird:

```
CREATE VIEW "public"."vtest" AS SELECT * FROM "public"."Tabellenname"
```

Wird danach die Datenbankdatei abgespeichert und wieder neu geladen, so können Ansichten auch über die GUI mit der **ODBC**-Verbindung erstellt werden.

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

odbcinst.ini

```
[PSQL]
Description=PostgreSQL
Driver64=/usr/lib64/psqlodbcw.so
UsageCount=1
```

odbc.ini

```
[PostgreSQL-libretest]
Description = PostgreSQL connection to libretest
Driver = PSQL
Database = libretest
Servername = localhost
ReadOnly = No
RowVersioning = No
ShowSystemTables = No
ConnSettings = SET SEARCH_PATH TO libretest;
```

Mit diesen Einstellungen ist eine Verbindung unter Linux möglich. Serielle Felder, die in PostgreSQL der Ersatz für Autoincrement-Werte sind, wurden anstandslos ausgelesen. Auch die Nutzung von Aliasbezeichnungen in Abfragen beeinflussten nicht, wie bei dem JDBC-Treiber, die korrekte Rückgabe der Werte nach Neueingaben.

PostgreSQL hat die Eigenart, dass innerhalb einer Datenbank unterschiedliche Schemata für unterschiedliche Zwecke angelegt werden können. Häufig wird ein anderer Schemaname als «public» beim Import von Daten aus anderen Datenbanken gewählt. Der Eintrag für die ConnSettings bewirkt, dass alle Schemata aus der Datenbank «libretest» angezeigt werden. Soll nur ein Schema angezeigt werden, so wird der Name dieses Schemas, das sich in der Datenbank befindet, eingetragen.

Autoincrement-Werte bei PostgreSQL

PostgreSQL unterstützt nicht direkt Autoincrement-Werte. Stattdessen muss ein Feld, das automatisch hochzählende Werte generieren soll, als Feld des Typs **SERIAL** definiert werden.

```
CREATE TABLE "public"."Test" (
"ID" SERIAL PRIMARY KEY
);
```

Hier wird ein automatisch hochzählendes Feld "ID" als Primärschlüssel definiert. Die entsprechende Eingabe unter **Extras → SQL** erzeugt die Tabelle "Test". Wird anschließend über **Ansicht → Tabellen aktualisieren** die Anzeige der Tabelle ermöglicht, so kann die Tabelle in dem GUI-Editor

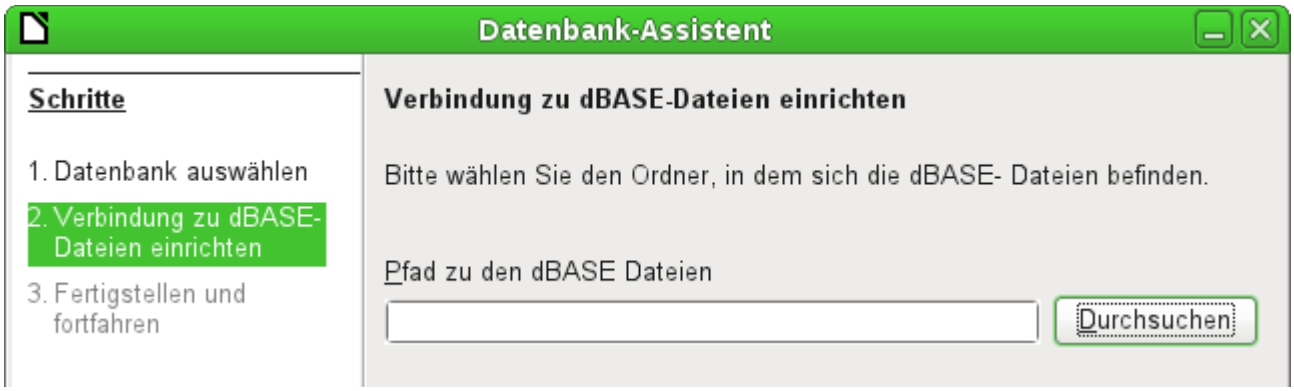
anschließend weiter bearbeitet werden (**rechte Maustaste auf der Tabelle → Bearbeiten**). Der direkte Treiber von PostgreSQL gibt zwar vor, bei INTEGER-Feldern auch einen AutoWert zu erstellen. Wenn dann aber die Tabelle erstellt wurde, dann wird daraus eben nur ein Feld, in das die Werte durch den Nutzer eingetragen werden müssen.

Über den JDBC-Treiber ist es möglich, direkt Felder des Typs **SERIAL** zu erstellen. Unter **Bearbeiten → Datenbank → Erweiterte Einstellungen → Generierte Werte** darf dazu aber **kein Eintrag** erfolgen. Wird dort SERIAL eingetragen, so bricht die Erstellung eines entsprechenden Feldes über die GUI mit einer Fehlermeldung ab.

dBase-Datenbanken

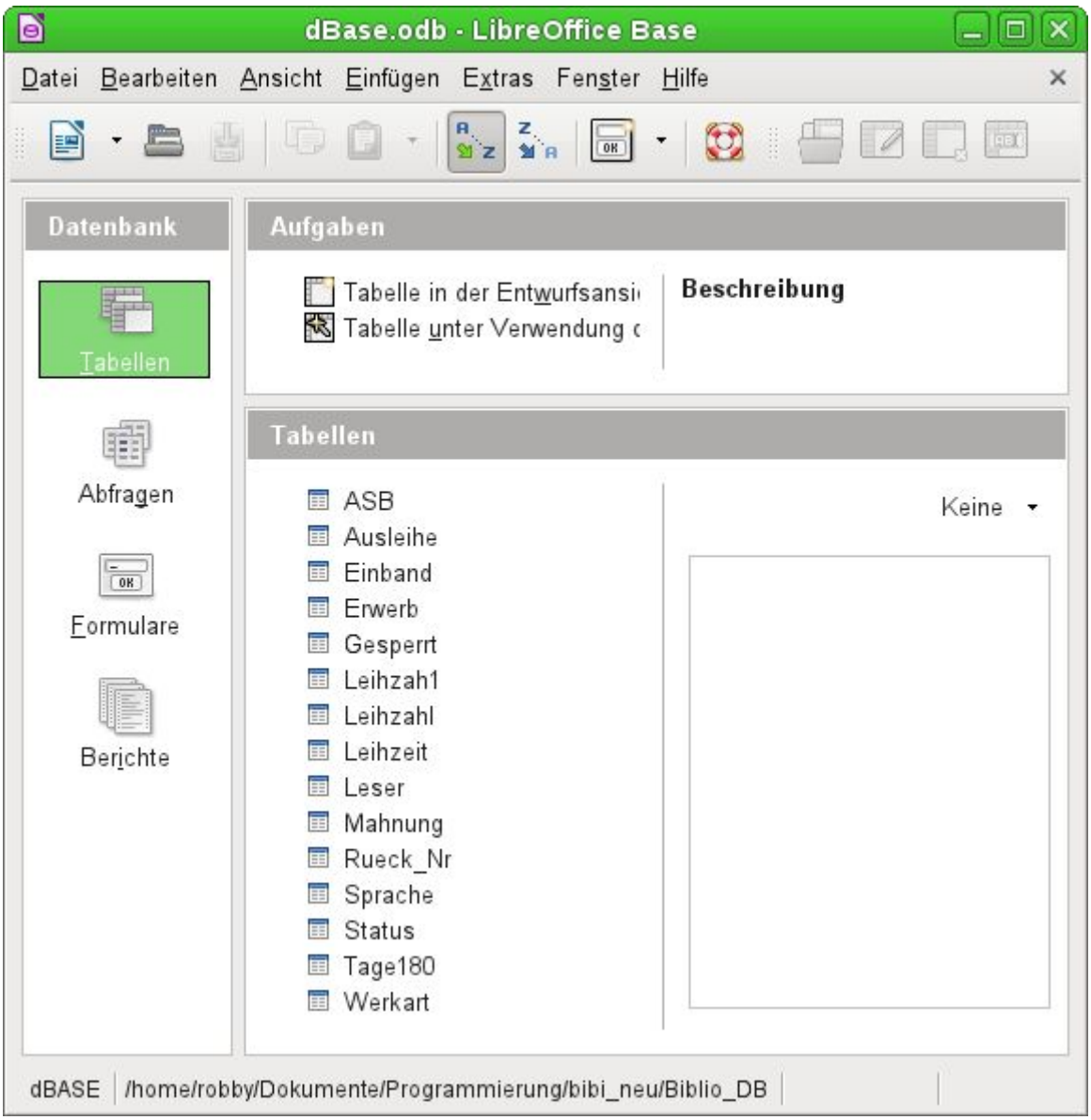
Bei dBase handelt es sich um ein Datenbankformat, dass alle Daten in einem beliebigen Verzeichnis in separaten Tabellen bereitstellt. Verknüpfungen zwischen den Tabellen müssen über die Programmstruktur erledigt werden. Dbase hat keine Möglichkeit, intern zu verhindern, dass z.B. in der Medien-Datenbank ein Medium gelöscht wird, der Verweis darauf aber weiterhin in der Tabelle zum Medienentleih erhalten bleibt.

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*)



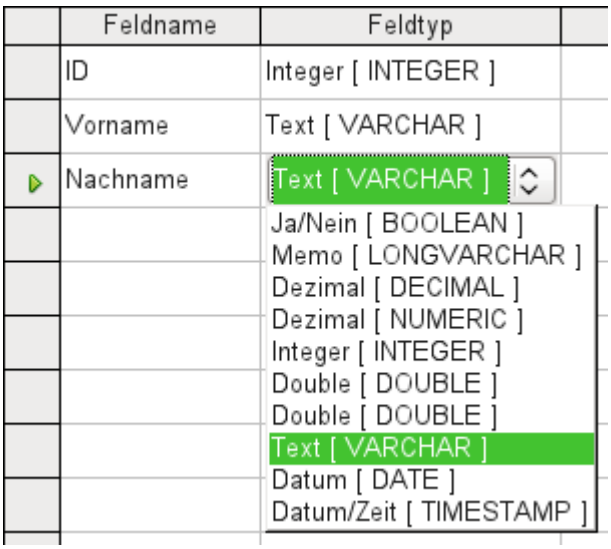
Die Verbindung wird zu einem Verzeichnis hergestellt. Alle *.dbf-Dateien, die sich in diesem Verzeichnis befinden, werden anschließend angezeigt und können über Formulare miteinander verbunden werden.

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



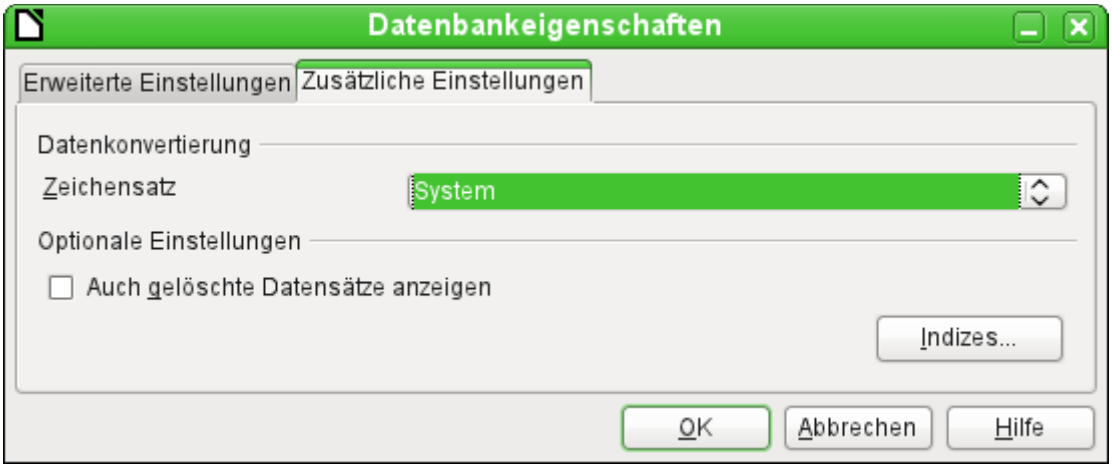
Tabellen können in Base neu erstellt werden. Sie werden dann einfach als neue Dateien in das angegebene Verzeichnis geschrieben.

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.



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:

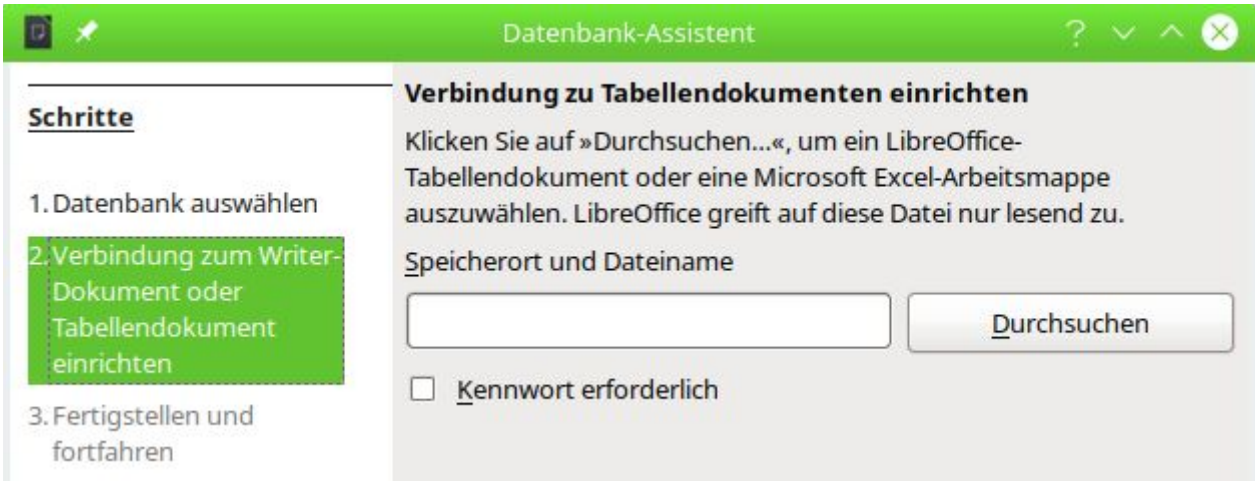


Hinweis	Der Importassistent für dBase hat Probleme, numerische Feldtypen und Ja/Nein-Felder automatisch richtig zu erkennen. (*Bug 53027*) Hier muss entsprechend nachgebessert werden.

Tabellendokumente und Tabellen in Writer-Dokumenten

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.

Auch eine Verbindung zu Writer-Dokumenten ist möglich. Dort werden dann bestehende Tabellen in Base angezeigt. Auch diese sind wie die Calc-Tabellen schreibgeschützt. Der Name der Tabellen kann im Writer-Dokument über den Navigator oder die Tabelleneigenschaften festgelegt werden.



Der Dialog für die Verbindung zu Tabellendokumenten und Writer-Dokumenten ist gleich. Lediglich aus den Schritten ist erkennbar: Beide Dokumenttypen können genutzt werden.

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. Auch Daten, die aus externen Quellen in Calc eingebunden sind (z. B. laufend aktualisierte Webseiten), lassen sich so in Base lesen.

Enthält das Tabellenblatt Datenbankbereiche, sichtbar über den Navigator oder **Daten → Bereich auswählen**, so werden diese Bereiche zusätzlich jeweils als Tabellen angezeigt.

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.

Natürlich lassen sich in Calc vorbereitete Daten sehr wohl nach Base importieren und dort dann entsprechend verwerten. Siehe hierzu das Kapitel «Import von Daten aus anderen Datenquellen».

Daten in Calc bearbeiten und in Base aktualisieren

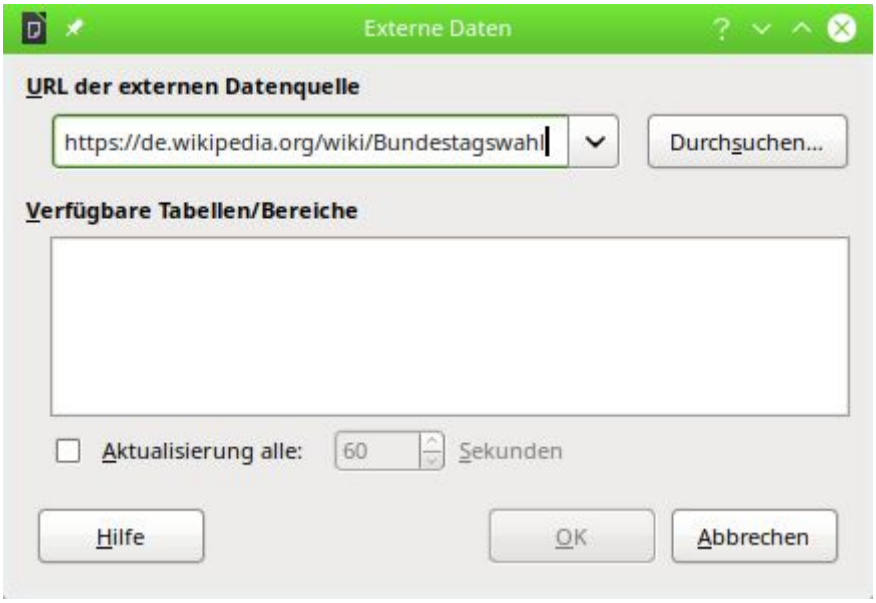
Innerhalb von Base sind Daten aus Calc nicht editierbar. Sollen trotzdem Daten verändert werden, so sind die folgenden Schritte zu beachten:

1. Die Base-Datei muss mit einem Zugriff auf die Tabellen geöffnet sein, bevor die damit verbundene Calc-Datei geöffnet wird.
2. Die Calc-Datei wird mit einem Schreibschutz geöffnet. Hier auf Dokument bearbeiten klicken.
3. Werden Daten im vorhandenen Datenbereich geändert, so muss in der Base-Tabelle der Button zum Aktualisieren (oder **Daten → Aktualisieren**) betätigt werden. Die geänderten Daten sind jetzt in der Base-Datei zu sehen. Vorsicht! Die Daten sind in Calc nicht abgespeichert.
4. Werden Daten unterhalb des Bereiches in einer neuen Zeile hinzugefügt, so bekommt die Tabelle in Base von der zusätzlichen Zeile nichts mit. In diesem Fall die Tabelle schließen und **Ansicht → Tabellen aktualisieren** im Hauptmenü der Base-Datei betätigen. Danach die Tabelle wieder öffnen und die zusätzliche Zeile erscheint.

Tabellen aus dem Internet über Calc automatisch aktualisiert in Base auslesen

In Calc ist es möglich, auf externe Daten aus dem Internet zuzugreifen. Diese Daten lassen sich mit einer beständigen Aktualisierung versehen. So ist es auch möglich, mit Base eben über Calc auf solche Daten zuzugreifen.

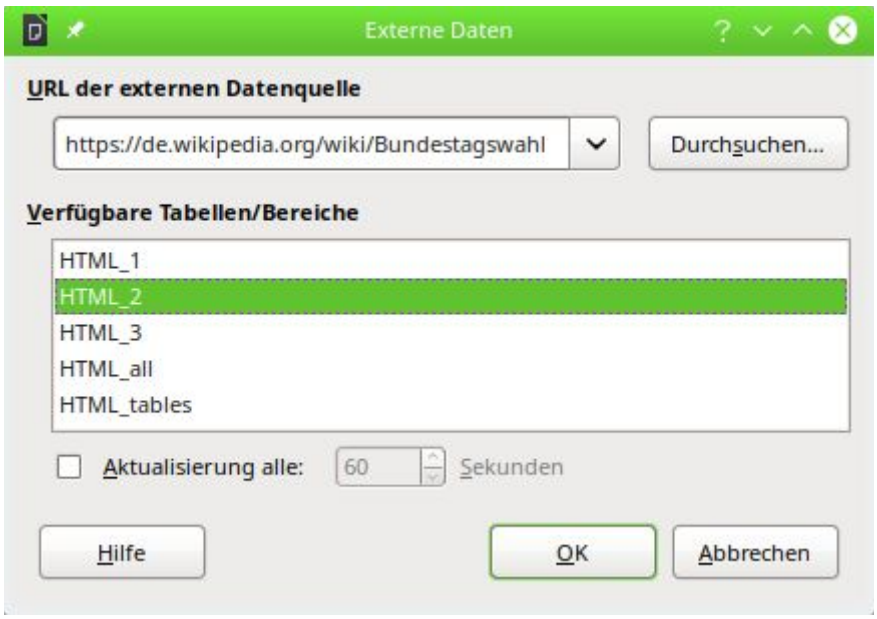
In Calc wird **Tabelle → Verknüpfung zu externen Daten...** aufgerufen.



Hier wurde als Quelle der Wikipediaartikel zu Bundestagswahlen ausgesucht. Bei dem Artikel lohnt sich zwar nicht die Aktualisierung, aber das Prinzip lässt sich daran auch nachvollziehbar erklären.

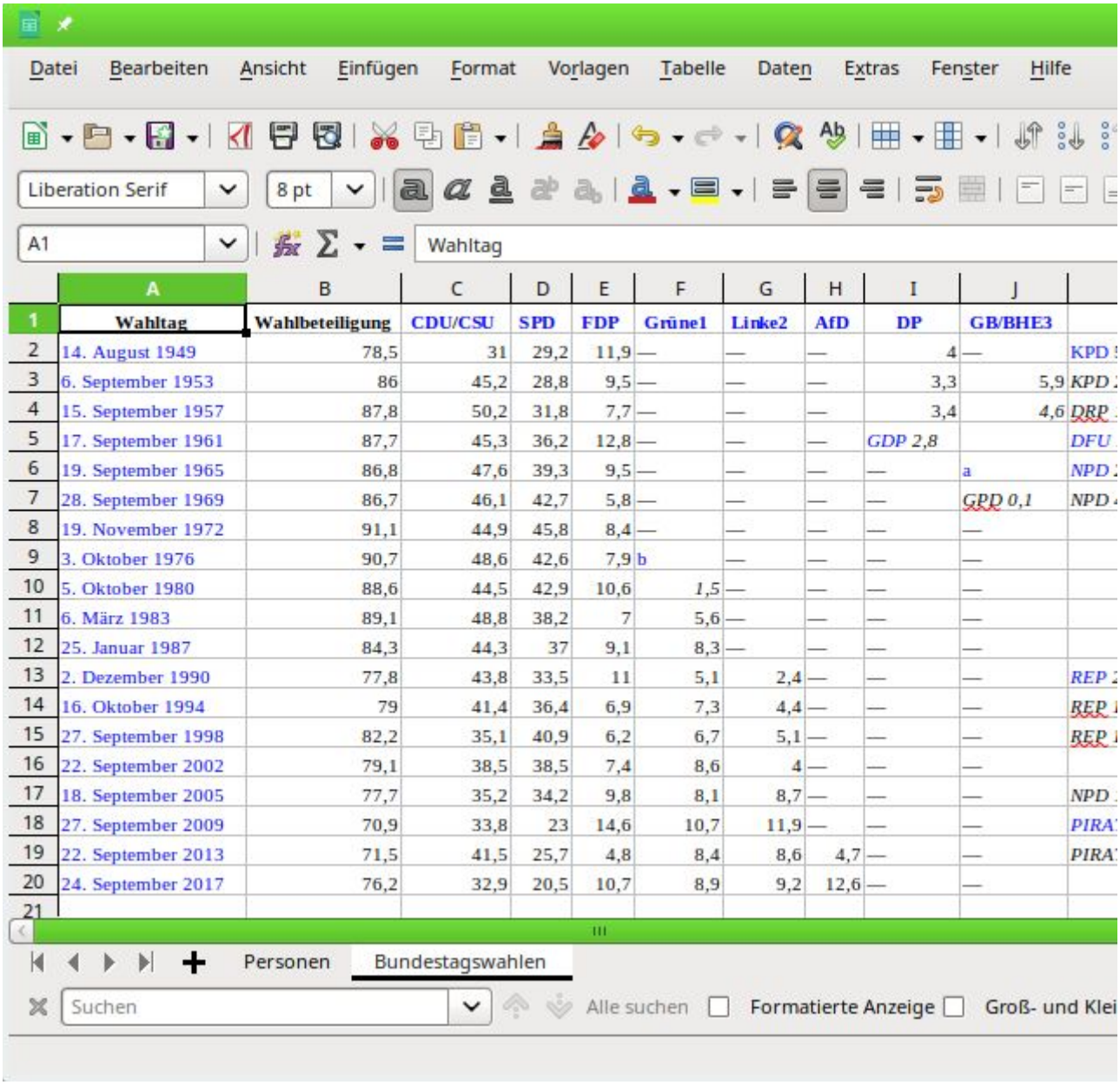
Ist der Link zu der externen Datenquelle eingegeben, so muss diese Eingabe mit einem *Return* betätigt werden. Ansonsten passiert nichts. Erst danach beginnt die Suche nach vorhandenen Inhalten auf der Seite.

Wenn Calc entsprechende Informationen aus der Seite geladen hat erscheinen zuerst die Importoptionen. Da die Tabellen in Wikipedia in dem gewählten Beispiel sowieso eine Mischung aus Text und Zahlen darstellen wird auch Base daraus nur Textfelder machen. Deswegen ist hier z. B. das Erkennen weiterer Zahlen nicht aktiviert worden.



Die Tabellen in dem Wikipediaartikel werden von oben nach unten durchnummeriert. Hier ist die zweite Tabelle **HTML_2** ausgewählt, die die prozentualen Anteile für die verschiedenen Parteien bei sämtlichen Bundestagswahlen aufzeigt. Die unteren beiden Einträge sind für die Nutzung in Base nicht geeignet. Bei **HTML_all** wird der gesamte Inhalt ohne Rücksicht auf Tabellen geladen, bei **HTML_tables** alle 3 Tabellen auf eine Seite des Calc-Dokumentes gepackt, so dass Base daraus versuchen würde, eine Gesamttabelle zu erstellen.

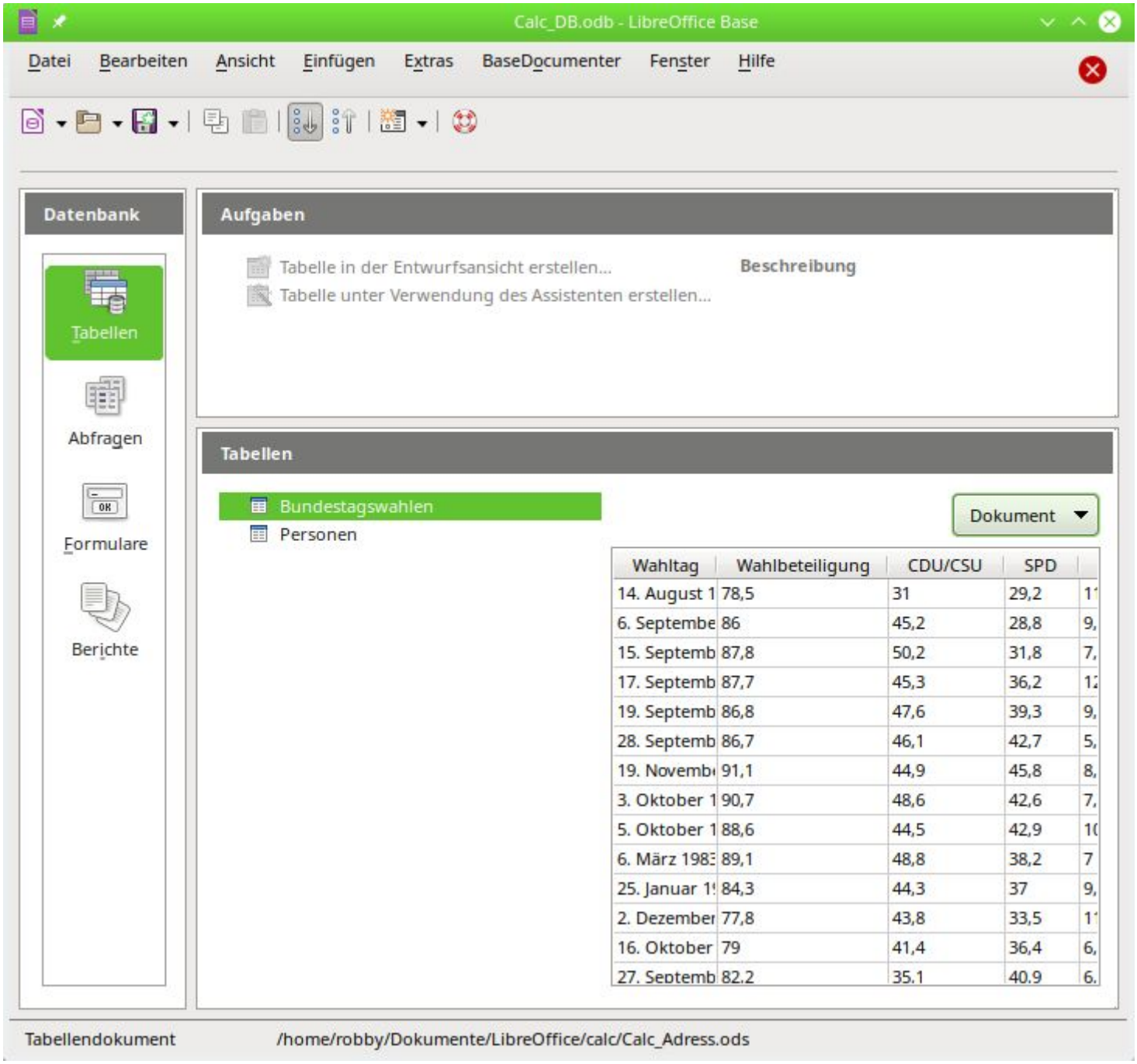
Jetzt könnte noch die automatische Aktualisierung eingestellt werden, so dass z. B. alle 60 Sekunden im Netz nachgeschaut wird, ob die Daten noch Bestand haben. So etwas macht wohl eher bei Aktienkursen als bei einer Tabelle zur Bundestagswahl Sinn.



	A	B	C	D	E	F	G	H	I	J	
1	Wahltag	Wahlbeteiligung	CDU/CSU	SPD	FDP	Grüne1	Linke2	AfD	DP	GB/BHE3	
2	14. August 1949	78,5	31	29,2	11,9	—	—	—	4	—	KPD
3	6. September 1953	86	45,2	28,8	9,5	—	—	—	3,3	5,9	KPD
4	15. September 1957	87,8	50,2	31,8	7,7	—	—	—	3,4	4,6	DRP
5	17. September 1961	87,7	45,3	36,2	12,8	—	—	—	GDP 2,8		DFU
6	19. September 1965	86,8	47,6	39,3	9,5	—	—	—	—	a	NPD
7	28. September 1969	86,7	46,1	42,7	5,8	—	—	—	—	GPD 0,1	NPD
8	19. November 1972	91,1	44,9	45,8	8,4	—	—	—	—	—	
9	3. Oktober 1976	90,7	48,6	42,6	7,9	b	—	—	—	—	
10	5. Oktober 1980	88,6	44,5	42,9	10,6	1,5	—	—	—	—	
11	6. März 1983	89,1	48,8	38,2	7	5,6	—	—	—	—	
12	25. Januar 1987	84,3	44,3	37	9,1	8,3	—	—	—	—	
13	2. Dezember 1990	77,8	43,8	33,5	11	5,1	2,4	—	—	—	REP
14	16. Oktober 1994	79	41,4	36,4	6,9	7,3	4,4	—	—	—	REP
15	27. September 1998	82,2	35,1	40,9	6,2	6,7	5,1	—	—	—	REP
16	22. September 2002	79,1	38,5	38,5	7,4	8,6	4	—	—	—	
17	18. September 2005	77,7	35,2	34,2	9,8	8,1	8,7	—	—	—	NPD
18	27. September 2009	70,9	33,8	23	14,6	10,7	11,9	—	—	—	PIRA
19	22. September 2013	71,5	41,5	25,7	4,8	8,4	8,6	4,7	—	—	PIRA
20	24. September 2017	76,2	32,9	20,5	10,7	8,9	9,2	12,6	—	—	
21											

Die so eingelesenen Daten brauchen nicht weiter formatiert zu werden. das geschieht in Base. Für die Datenbankdatei ist lediglich wichtig, wie die Tabelle, hier «Bundestagswahlen» heißt. Denn dies ist auch der so erkannte Tabellenname.

Bei Tabellen, die so in Calc eingelesen werden, werden die besten Ergebnisse erzielt, wenn die Vorlage im Internet die Daten sauber in Zeilen und Spalten darstellt. Zeilenumbrüche in einer Zelle führen zu zusätzlichen Datenzeilen, die dann leere Felder in den anderen Spalten hervorrufen. Zusätzliche unbedarfte Formatierungen der Urheber einer solchen Tabelle können dazu führen, dass nach jeder Zeile mit Inhalt eine Leerzeile entsteht. Das werden dann auch leere Tabellenzeilen in Base. Ein Löschen von Zeilen bringt hier nichts, da schließlich der Inhalt regelmäßig aktualisiert wird. Da müsste dann schon innerhalb von Calc aus der Importtabelle in eine weitere Tabelle eingelesen werden, die diese Formatierungsfehler ausgleicht.

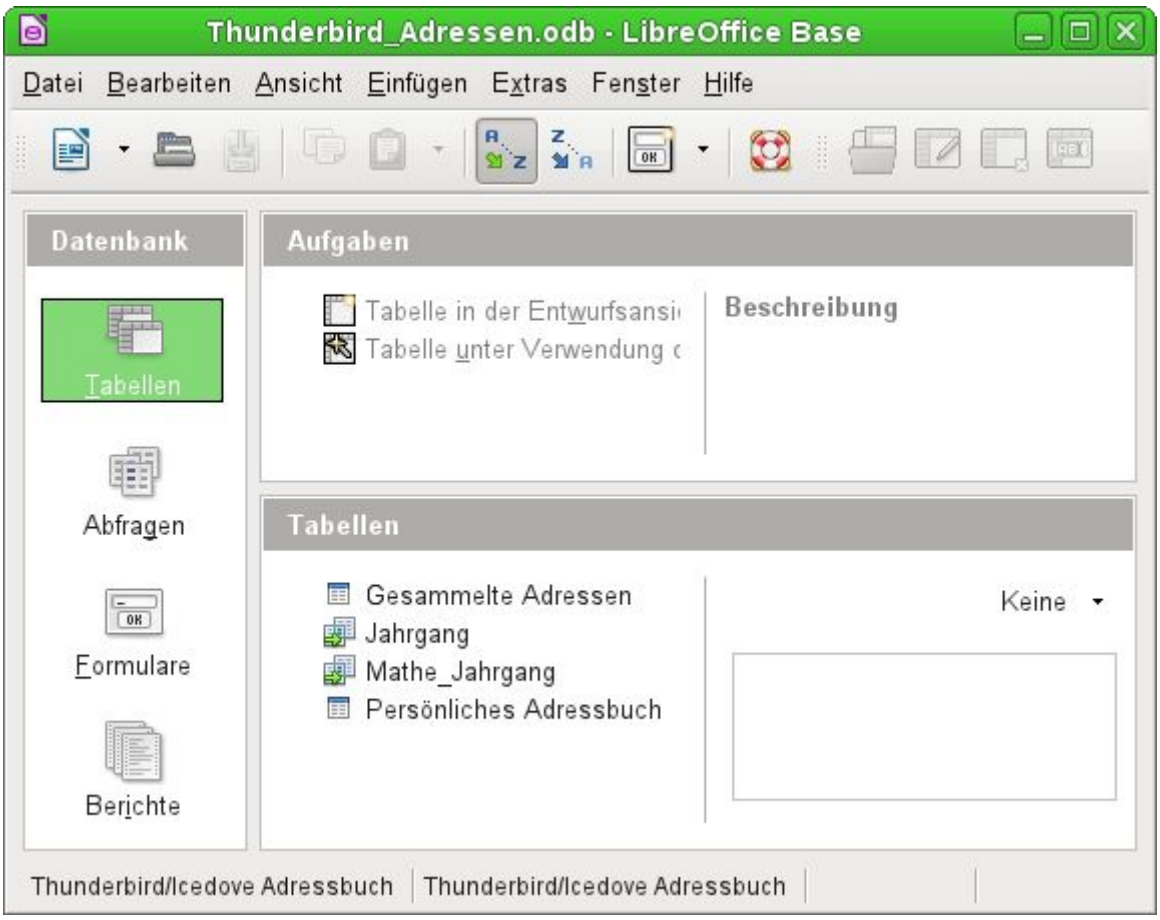


In der Datenbank stehen jetzt die Namen der Tabelle aus dem Calc-Dokument. Hier wurde die Dokumentansicht für die Tabellen aktiviert, so dass ausschnitthaft direkt zu sehen ist, welcher Inhalt in der Tabelle angezeigt wird.

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.

Hinweis	Die direkte Verbindung funktioniert nur zu Versionen vor Thunderbird 78. Die im Jahr 2020 erschienen Version ist auf SQLite als Datenbankmodul umgestiegen. Zum Kontakt zu dieser Datenbank siehe das Kapitel zu *SQLite*.



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.

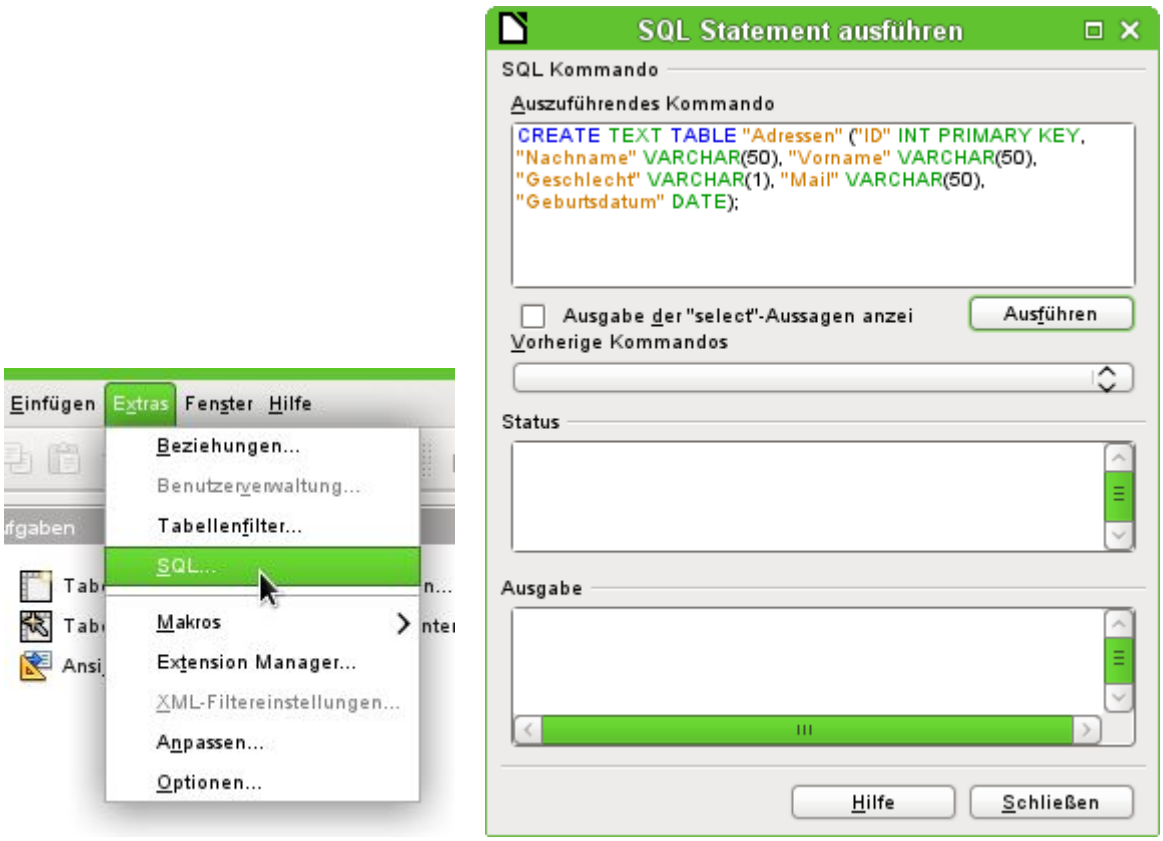
Texttabellen

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

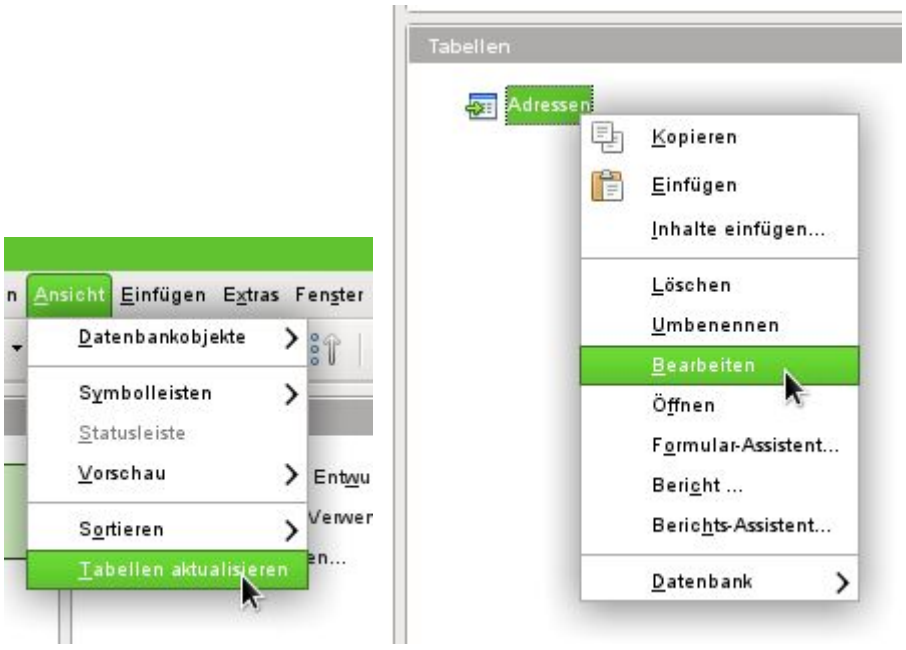
Texttabellen innerhalb einer internen HSQLDB-Datenbank

Ein gängiges Austauschformat zwischen Datenbanken ist das *.csv-Format. Daten werden hier in einer durch einfache Textbearbeitungsprogramme les- und schreibbaren Form gespeichert. Einzelne Felder werden durch Kommata getrennt. Enthält ein Feld Text mit einem Komma, so werden die Textfelder mit doppelten Anführungszeichen versehen. Ein neuer Datensatz beginnt nach einem Zeilenumbruch.

Soll zum Beispiel der Inhalt eines Adressbuches, das nicht direkt von den Treibern von Base unterstützt wird, eingelesen werden, so kann dies entweder über einen Import einer solchen *.csv-Datei erfolgen (hierzu 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.



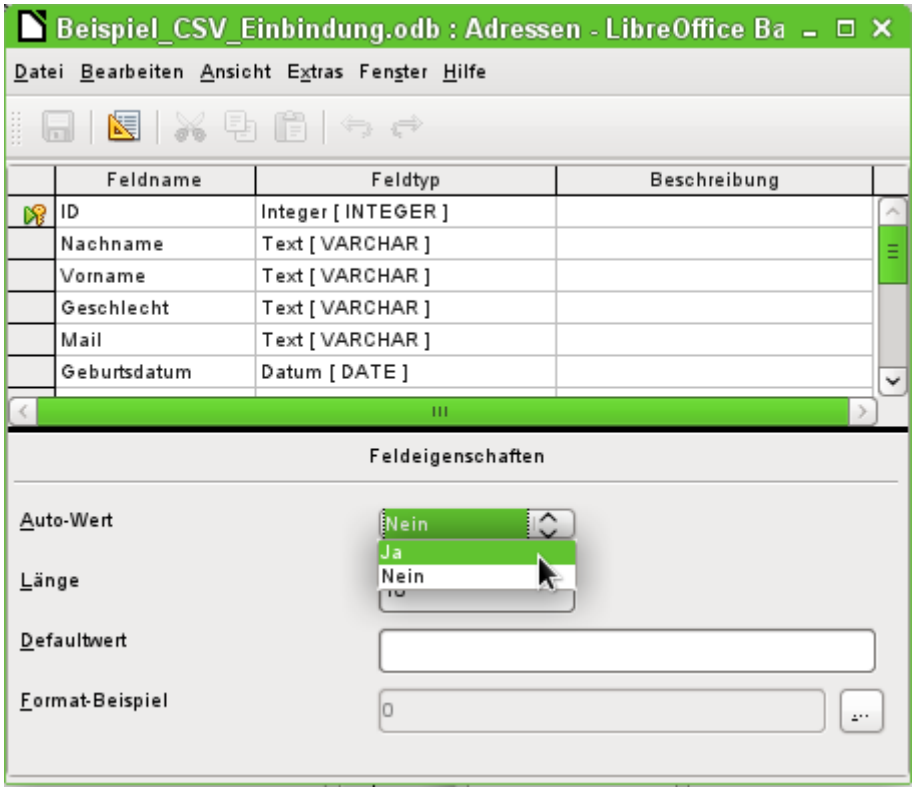
Der Zugriff zu einer Texttabelle kann nicht über die grafische Benutzeroberfläche erstellt werden.[2] Stattdessen wird über **Extras → SQL** eine Texttabelle erstellt. Die Felder in der Texttabelle müssen dem entsprechen, was die Texttabelle in der entsprechenden Reihenfolge liefert. Das Feld "ID" muss 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.

2 Diesem Handbuch liegt die Datenbank «Beispiel_CSV_Einbindung.odb» bei.

Das Symbol der Tabelle macht deutlich, dass es sich nicht um eine «normale» Tabelle der Datenbank handelt.



Die Tabelle wurde zum Bearbeiten geöffnet und das Primärschlüsselfeld auf einen automatisch hoch zählendes Feld umgestellt. Dies kann auch direkt mit der Eingabe des SQL-Befehls erfolgen:

```
CREATE TEXT TABLE "Adressen" ("ID" INT GENERATED BY DEFAULT AS
IDENTITY, ...
```

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";
```[3]

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.

3 Die Angabe «encoding=UTF-8» funktioniert bei vielen Systemen. Eventuell muss auch statt «UTF-8» «ansi» gewählt werden.

- 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 jedes Feld in doppelte Anführungszeichen gefasst werden, so ist «all_quoted=true» zu setzen.

Zu weiteren Parametern siehe: *http://hsqldb.org/doc/guide/ch09.html#set_table_source-section* .

Mit

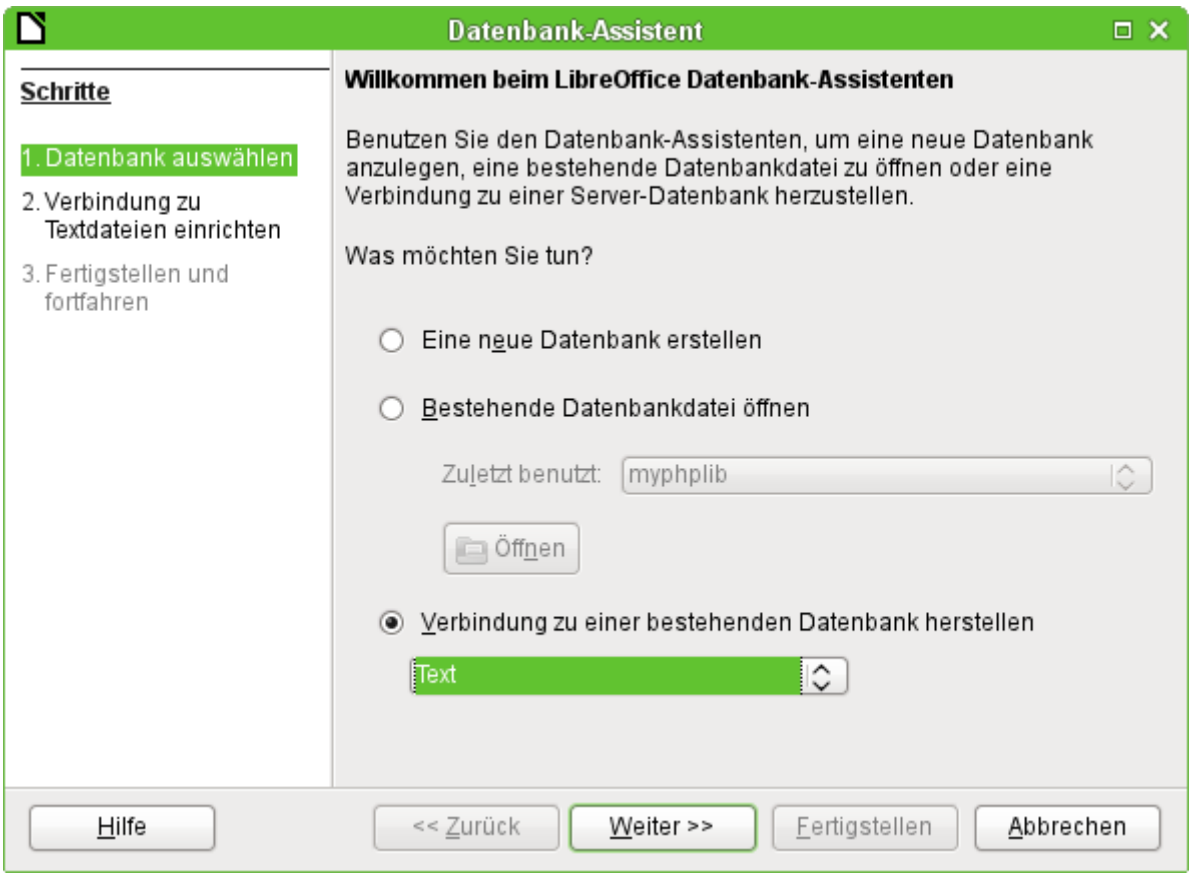
```
SET TABLE "Adressen" READONLY TRUE;
```

wird schließlich davor geschützt, dass in die Tabelle überhaupt etwas geschrieben wird. So steht dann die Tabelle als normale Adresstabelle 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.

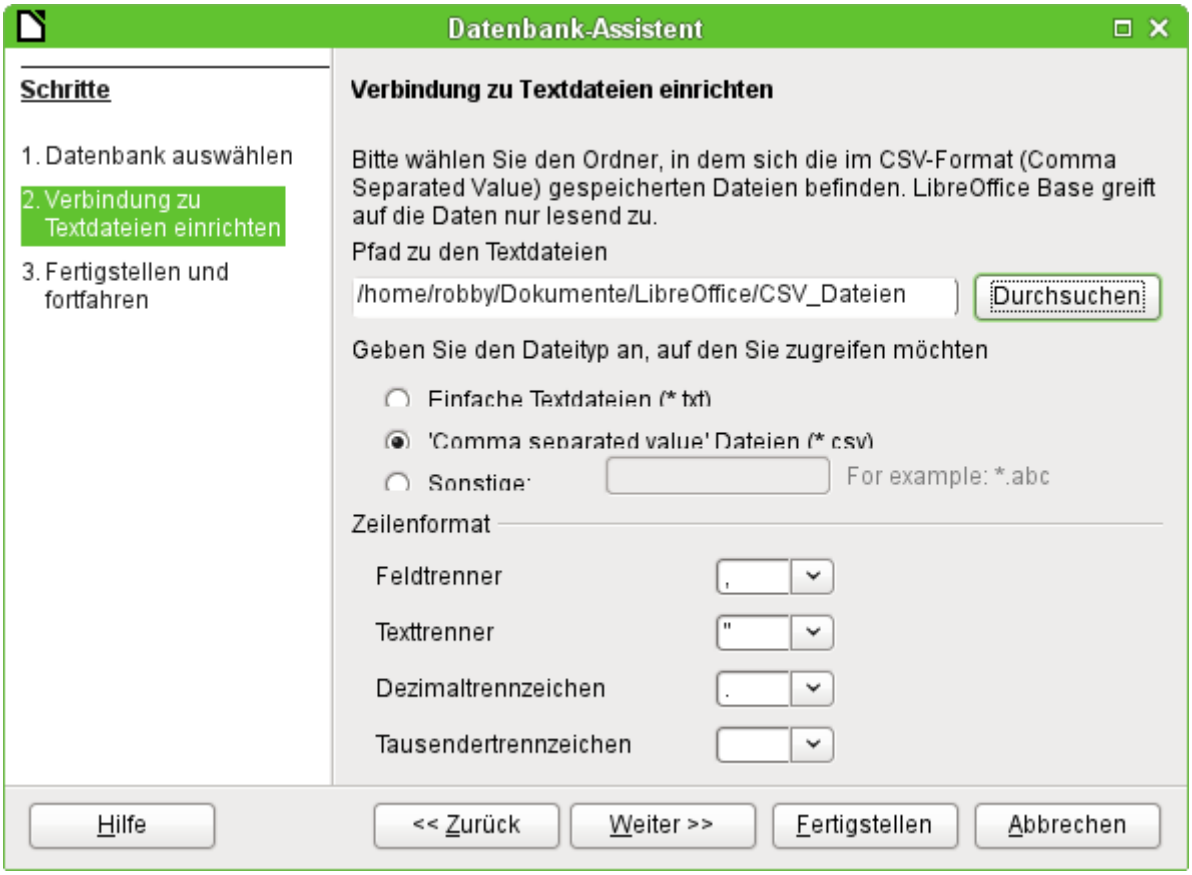
Hinweis	Texttabellen innerhalb der internen HSQLDB scheinen zur Zeit die einzige Möglichkeit zu sein, Zeilenumbrüche aus externen Tabellen wie z.B. einer Calc-Tabelle beim Import in eine HSQLDB zu erhalten: Die Calc-Tabellen müssen jeweils als einzelne *.csv-Dateien exportiert werden. Zeilenköpfe sind zu entfernen. Für jede *.csv-Datei wird eine Texttabelle erstellt. Mehrzeilige Texte müssen in LONGVARCHAR-Spalten abgelegt sein. Anschließend werden diese Textdateien wieder kopiert und als normale Tabellen in die HSQLDB-Datenbank eingefügt.

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.

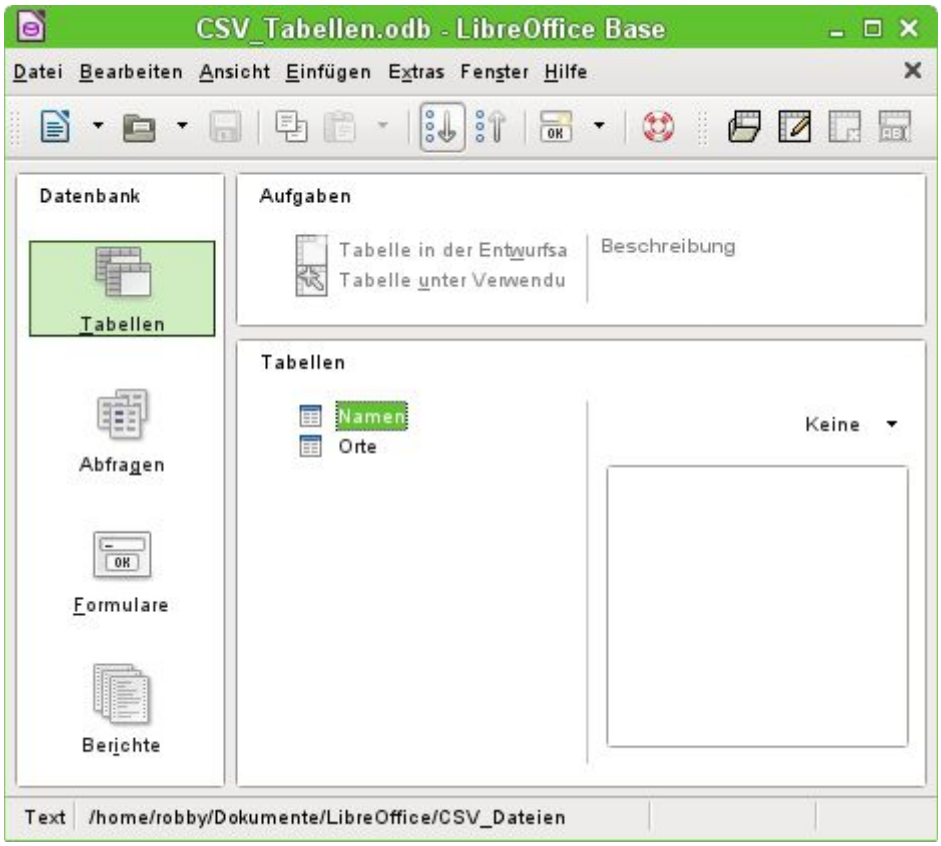


Startpunkt ist die Verbindung zu einer bestehenden Datenbank. Hier ist als Format «Text» gewählt.



Der Pfad zu den Textdateien wird ausgesucht. Innerhalb des Verzeichnisses werden später alle dem Dateityp entsprechenden Dateien aufgelistet. Für das *.csv-Format ist hier die zweite Option gewählt.

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



In der Tabellenansicht sind dann alle Tabellen mit dem Dateinamen aus dem aufgezeigten Verzeichnis ohne die Dateinamenserweiterung zu sehen. Die Aufgaben zum Erstellen von Tabellen stehen nicht zur Verfügung. Die Tabellen selbst sind nur lesbar, nicht schreibbar.

Auch der Zugriff auf die Tabellen mit Abfragen ist beschränkt auf eine Tabelle und keine Funktionen.

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 wird ab LO 6.1 schrittweise durch eine interne Firebird-Datenbank ersetzt. Seit Version LO 4.2.* ist die interne Firebird-Datenbank erst einmal nur als «experimentelle Funktion» zusätzlich wählbar. Das liegt einfach daran, dass zu dem Zeitpunkt längst noch nicht alle Funktionen richtig integriert werden konnten. Um aber zu sehen, was Firebird leisten kann, wird hier die Verbindung zu einer externen Firebird-Datenbank dargestellt.

Da die Dokumentation dazu nicht ganz so umfangreich ist wie z.B. zu MySQL oder PostgreSQL hier zuerst 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 15.0 zu einer funktionierenden Datenbank mit Firebird geführt:

1. Notwendig für die Installationsschritte bis zur funktionierenden Datenbank sind Administratorrechte auf dem Rechner. Dem Systemuser «firebird» sollte ein Passwort zugeordnet werden.
2. Der Nutzer «firebird» muss sich mit
   `su firebird`
   auf der Konsole anmelden.
3. **`sysdba`** ist der Nutzername für den Admin-Account. Das Standardpasswort ist **`masterkey`**. Dies sollte in Produktivumgebungen geändert werden.
4. Eine Passwortänderung geht im Terminal als Nutzer «firebird» über:
   `isql-fb -user sysdba`
5. Danach geht es weiter in der SQL-Konsole. Zuerst muss eine leere Datenbank erstellt werden, damit auf die weiteren Befehle zugegriffen werden kann:**`SQL> CREATE DATABASE 'fbtest.fdb';`**
   `SQL> CREATE USER SYSDBA password 'neuespasswort';`
   SQL> **`commit;`**
6. Erst nach diesen Einstellungen kann der Server gestartet werden.
7. Ein neuer Nutzer sollte erstellt werden:
   SQL> **`connect localhost:employee user SYSDBA password neuespasswort;`**
   Als Antwort hierauf erscheint die Meldung
   `Database: localhost:employee, User: SYSDBA`
   Hiernach wird dann eingegeben:
   SQL> **`CREATE USER lotest password 'libre';`**
   Der neue Nutzer mit dem Namen **`lotest`** und dem Passwort **`libre`** wird also erst erstellt, wenn eine Verbindung zu einer Datenbank, hier der zu Firebird gehörigen festen Beispieldatenbank «employee», existiert.
8. Danach kann die Datenbank für den neuen Nutzer erstellt werden:
   SQL> **`CREATE DATABASE 'libretest.fdb' user 'lotest' password 'libre';`**
9. Auf die direkte Nachfrage nach dem Commit muss mit y geantwortet werden:
   `Commit current transaction (y/n)?y`
   `Committing.`
10. Mit
    SQL> **`quit;`**
    wird die Eingabe im SQL-Modus des Tools **`isql-fb`** beendet.

Werden die Angaben als Systemadministrator Root getätigt, so ist die Datenbank nicht dem richtigen Benutzer für den Netzwerkbetrieb zugeordnet. Die Datenbank muss als Nutzer und als Gruppe «firebird» eingetragen bekommen. Sonst klappt der Kontakt später nicht.

Firebird-Verbindung über JDBC

Zuerst muss das Jar-Archiv in LO eingebunden werden. Irritierend ist, dass es kein Archiv mit einer Bezeichnung firebird-*.jar gibt. Der aktuelle JDBC-Treiber ist auf der Seite *http://www.firebirds-ql.org/en/jdbc-driver/* zu finden. Der Treiber beginnt mit der Bezeichnung Jaybird...

Aus der *.zip-Datei muss nur das Archiv **jaybird-full-3.0.5.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:

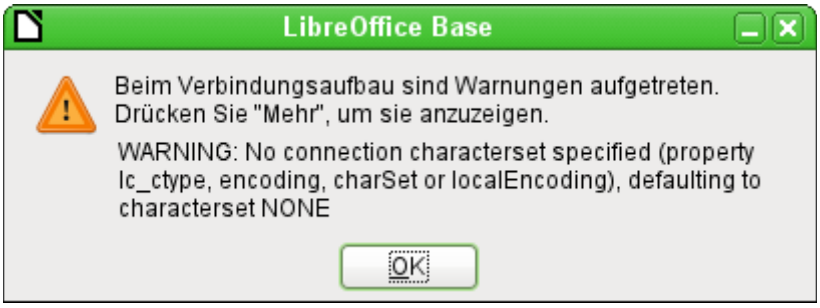
JDBC URL **`jdbc:firebirdsql://host[:port]/<path_or_alias>`**

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:



Bei der Erstellung von Tabellen muss darauf geachtet werden, dass die Formatierung der entsprechenden Felder («Feldeigenschaften») von Beginn an stimmt. Bei allen Zahlenwerten setzt sonst LO als Default-Format merkwürdigerweise ein Währungsfeld ein.

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

Aus dem Netz muss zuerst der passende ODBC-Treiber heruntergeladen werden: *http://www.firebirdsql.org/en/odbc-driver/*. Dieser Treiber besteht meist aus einer einfachen Datei, der libOdbcFb.so. Diese Datei wird meist mit Administratorrechten in einen vom System allgemein zugänglichen Pfad gelegt. Sie muss ausführbar sein.

Bei Linux-Systemen kann es wegen fehlender Bibliotheken zu einigen Problemen führen. Siehe hierzu auch *http://www.firebirdsql.org/manual/de/qsg2-de-installing.html* Die lassen sich aber durch entsprechende Symlinks beheben.

Die libOdbcFb.so benötigt die libodbc.so.1. Hier existiert in den meisten Systemen inzwischen die libodbc.so.2 . Auf diese Datei muss ein entsprechender Verweis mit dem Namen libodbc.so.1 in dem gleichen Verzeichnis abgespeichert werden.

Auch die libgds.so wird anschließend als fehlend angemahnt. Dies kann durch einen Link zu /usr/lib64/libfbclient.so.2 gelöst werden.

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

odbcinst.ini

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

odbc.ini

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

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.*

Die ODBC-Verbindung ist leider zur Zeit nur sehr eingeschränkt funktionstüchtig. Beim Erstellen von Tabellen stürzt LO ab. Beim Eingeben von Werten in ein Integer-Feld wird bei Linux-64-bit-Versionen an die Datenbank ein leeres Feld weitergegeben. Dies liegt daran, dass der ODBC-Treiber Integer-Werte wie Long-Werte behandelt. Dies ist bei 64bit-Linux-Systemen aber falsch. Siehe *https://bugs.documentfoundation.org/show_bug.cgi?id=45881#c2*. Wird statt Integer-Werten aber z.B. mit BigInt-Werten gearbeitet, so ist das kein Problem.

Insgesamt klappt die Zusammenarbeit mit dem ODBC-Treiber unter 64bit-Linux-Systemen häufig nicht brauchbar. Da funktioniert der JDBC-Treiber eindeutig besser.

Direkte Verbindung zu einer Firebird-Datei

Wird eine Serverdatenbank nicht benötigt, dann ist auch die direkte Verbindung zu einer Firebird-Datei möglich. Hier muss lediglich, wie z.B. bei Tabellendokumenten, der Pfad zu der Firebird-Datei herausgesucht werden. Die Verbindung wird dann automatisch hergestellt. Die Firebird-Datei muss dazu allerdings ab LibreOffice 5.3 mit einer Firebird-Version ab Firebird 3 erstellt worden sein.

Aus einer internen Datenbankdatei lässt sich eine externe Datenbankdatei erstellen:

- Das temporäre Verzeichnis, das in LibreOffice definiert ist, muss mit dem Dateimanager aufgesucht werden.
- Die Datenbank mit der internen Firebird-Datenbankdatei muss geöffnet werden. In dieser Datei liegt zur Zeit die Datei «firebird.fbk», eine Backupdatei der Daten.
- Der Tabellencontainer muss geöffnet werden, damit die Verbindung zur internen Datenbankdatei erstellt wird.
- Jetzt wird automatisch in dem temporären Verzeichnis ein Verzeichnis erstellt, das wiederum mehrere Unterverzeichnisse hat. Dieses Verzeichnis ist nicht an dem Namen, sondern an dem Zeitpunkt der Erstellung zu erkennen.
- In einem dieser Unterverzeichnisse steckt neben «firebird.fbk» die daraus erstellte «firebird.fdb». Diese Datei kann jetzt kopiert und als externe Datenbankdatei weiter verwandt werden. Ein Passwortschutz für diese Datei besteht nicht.
- Nach dem Schließen der internen Datenbankdatei wird das temporäre Verzeichnis automatisch gelöscht.

SQLite

SQLite-Datenbanken bestehen aus einer einzigen Datei. Diese Datei kann irgendwo im Dateisystem gespeichert werden und muss nur über einen entsprechenden Treiber mit Base verbunden werden.

Erstellen einer Datenbank

Das Erstellen einer Datenbank beschränkt sich darauf, eine leere Datei in einem Verzeichnis abzulegen. Es ist nicht einmal erforderlich, der leeren Datei eine bestimmte Dateiendung zu geben. Unter Linux reicht es, eine einfache Textdatei zu erstellen. Die Datei wird dann mit dem Namen libretest.db versehen. Bei der Erstellung kann es passieren, dass bereits Absätze vorhanden sind. Diese Absätze sollten gelöscht werden, so dass der Inhalt der Datei 0 Byte groß ist.

Natürlich lässt sich eine leere Datei auch auf der Konsole erstellen, die bei der Installation des Treibers mitinstalliert wird:

```
sqlite3 libretest.db
```

erstellt die leere Datenbankdatei. Dabei wird dann die SQLite-Konsole geöffnet. Die Datei ist noch nicht in dem Verzeichnis sichtbar, in dem der Befehl ausgeführt wird. Auf der Konsole wird jetzt nur noch

```
.exit
```

eingegeben und die leere Datei wird geschrieben.

SQLite-Verbindung über ODBC

Unter Linux wird der ODBC-Treiber des Systems über die Paketverwaltung installiert. Die Einstellungen in der odbcinst.ini werden dabei automatisch vorgenommen.

odbcinst.ini

```
[SQLITE3]
Description=SQLite ODBC 3.X
Driver=/usr/lib64/libsqlite3odbc.so
Setup=/usr/lib64/libsqlite3odbc.so
Threading=2
FileUsage=1
UsageCount=1
```

In der odbc.ini wird dann die Verbindung zu der vorher erstellten Datei vorgenommen. Die Datei wird anschließend über die entsprechende Bezeichnung, hier Sqlite3-libretest, über ODBC mit Base verbunden.

odbc.ini

```
[Sqlite3-libretest]
Description = SQLite database libreoffice test
Driver= SQLITE3
Database= /home/<nutzername>/Dokumente/LibreOffice/libretest.db
```

Weitere SQLite-Verbindungen

Es existieren zwar JDBC-Treiber für SQLite. Der Zugriff auf eine SQLite-Datenbankdatei bricht unter LibreOffice-Base aber leider mit der Fehlermeldung ab:

```
SQLite only supports TYPE_FORWARD_ONLY Cursors
```

Auch eine direkte Verbindung zu SQLite gibt es für LibreOffice bisher nicht.

Zugriff auf Access

Während unter Windows der direkte Zugriff auf Access-Datenbanken möglich ist muss unter Linux ein entsprechender Treiber die Verbindung herstellen. Hier bietet sich die JDBC-Verbindung über UcanAccess an: *http://ucanaccess.sourceforge.net/site.html.*

Der JDBC-Treiber kann wie unter «*MySQL-Verbindung über JDBC*» beschrieben eingebunden und genutzt werden. Die *.zip-Datei muss entpackt werden. Das entstehende Verzeichnis wird komplett mit den Unterverzeichnissen benötigt. Im Unterverzeichnis loader befindet sich ucanload.jar. Diese Datei muss über den Class-Path direkt eingebunden werden.

Für die Angaben bei der JDBC-Installation sind folgende Parameter wichtig:

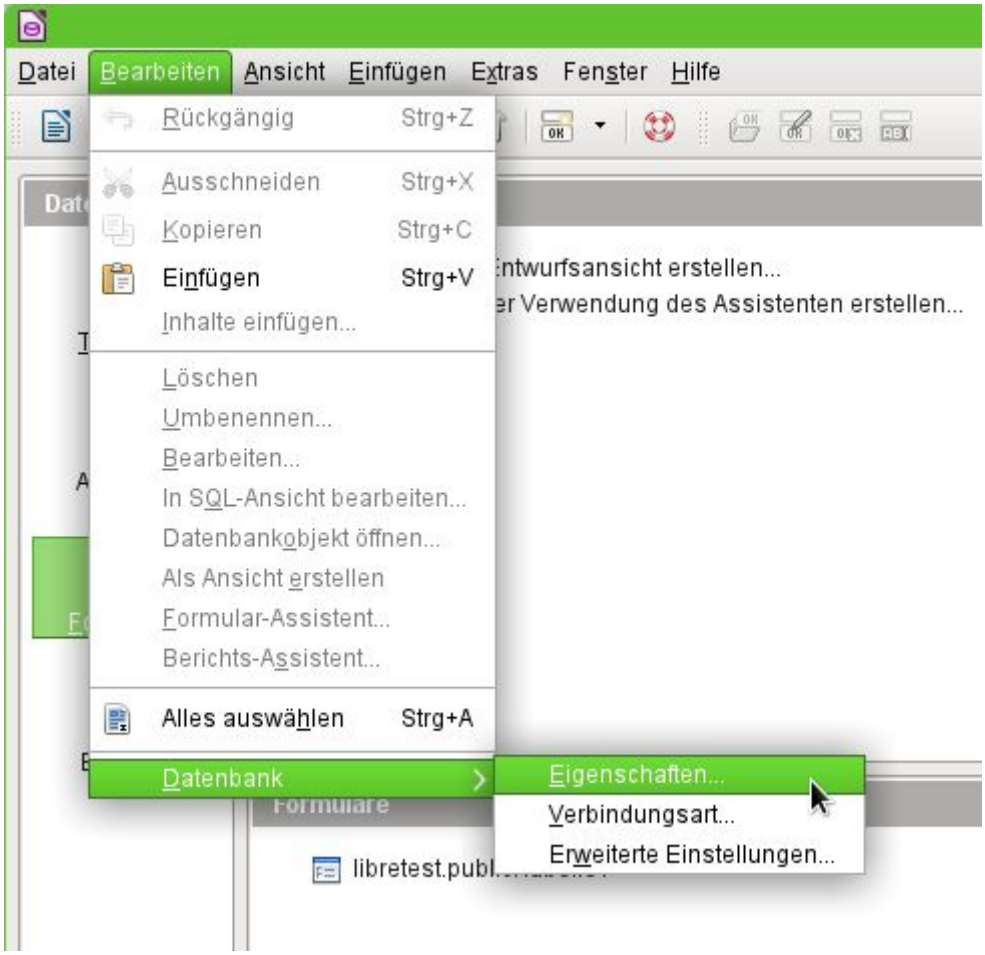
JDBC URL **`jdbc:ucanaccess:///home/<path_or_alias>`**

Treiberbezeichnung: **`net.ucanaccess.jdbc.UcanloadDriver`**

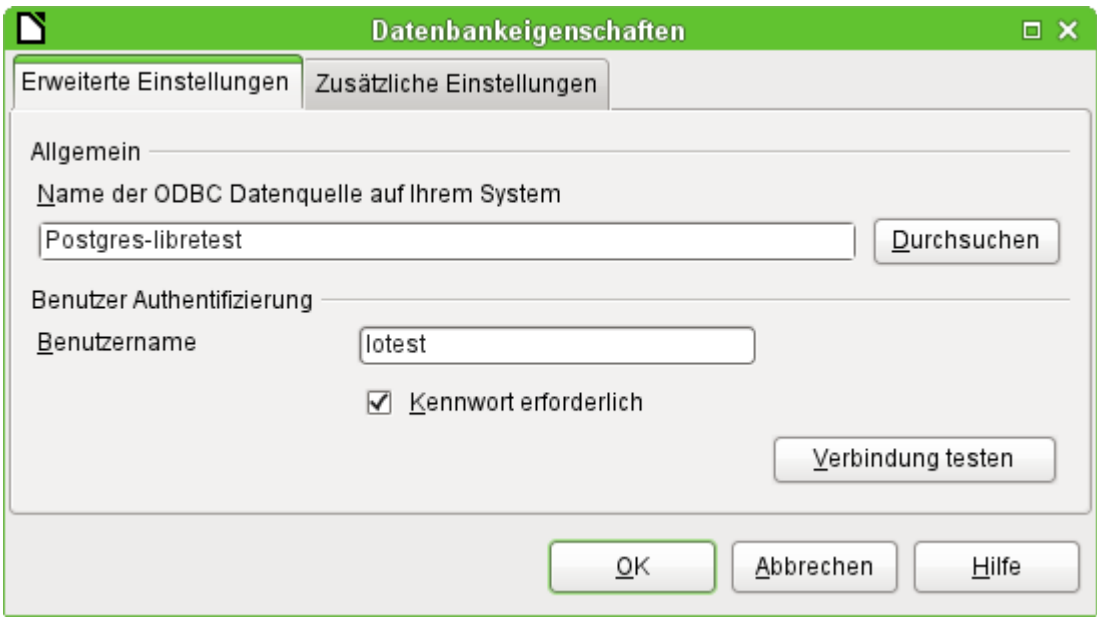
Nachträgliche Bearbeitung der Verbindungseigenschaften

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 Verbindungsparametern zur Datenbank geändert worden.

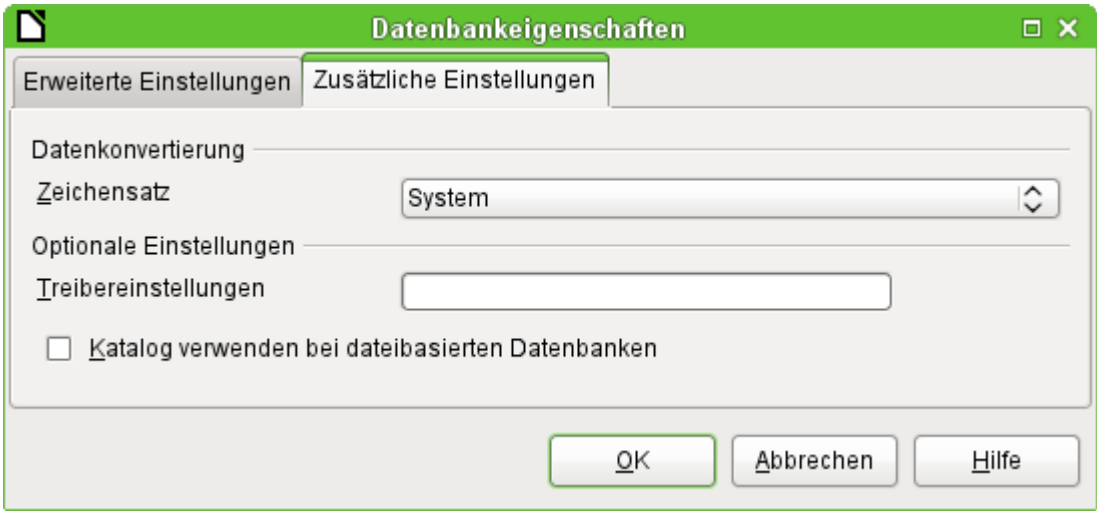
Die folgenden Screenshots zeigen die Möglichkeit auf, wie z.B. die Verbindungsparameter zu einer externen PostgreSQL-Datenbank im Nachhinein geändert werden können.



Über **Bearbeiten → Datenbank** werden die Einstellungsmöglichkeiten «Eigenschaften», «Verbindungsart» und «Erweiterte Einstellung» aufgerufen.

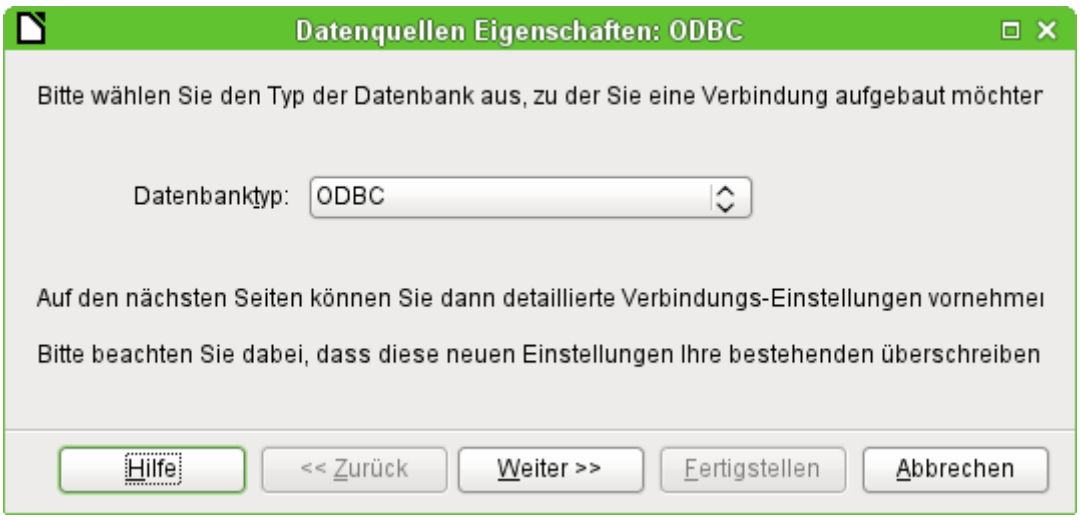


Falls sich der Name der Datenquelle geändert hat, kann dies hier geändert werde. Bei einer Verbindung über ODBC wird ja der Name, mit dem die Datenbank aufgerufen werden kann, in der odbc.ini festgelegt. Der Name ist in der Regel nicht gleich dem eigentlichen Datenbanknamen in PostgreSQL.

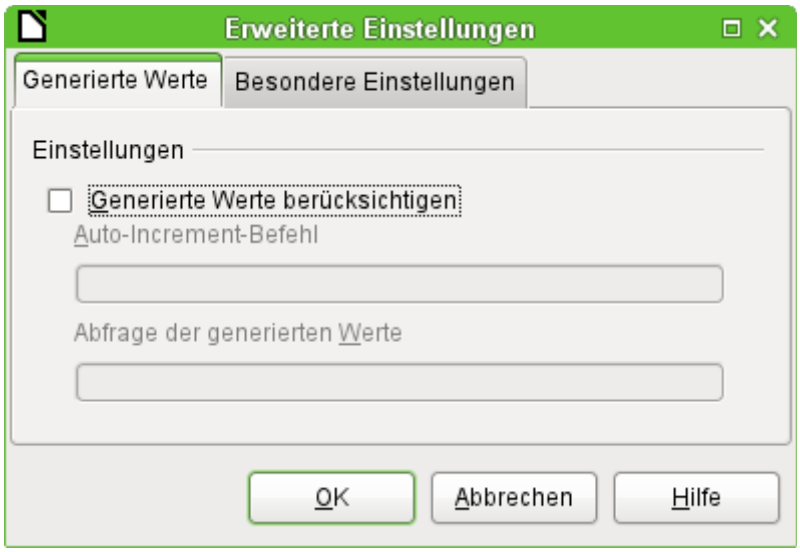


Gibt es Probleme mit dem Zeichensatz? Diese Probleme können eventuell in dem zweiten Reiter der Datenbankeigenschaften gelöst werden.

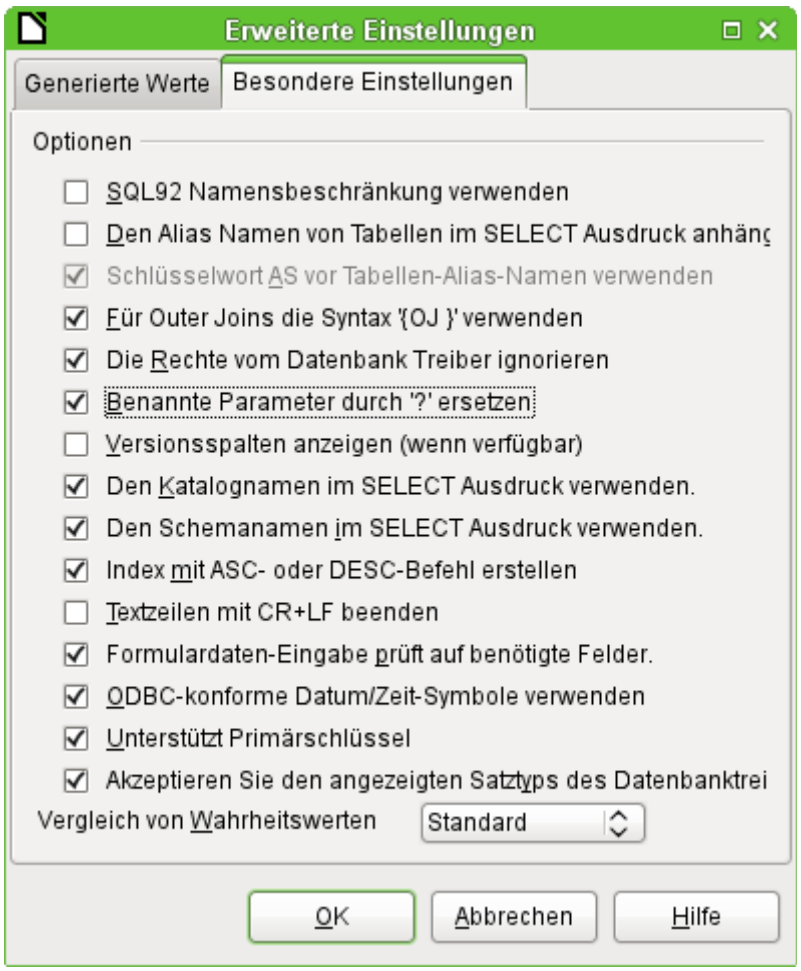
Auch eine zusätzliche spezielle Einstellung des Treibers ist möglich, wenn eventuell ein Parameter ausgeführt werden soll, der zur Zeit nicht bereits in der odbc.ini steht.



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 → Erweitere Einstellungen** erreichbaren besonderen Einstellungen 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.

Leider sind manchmal nicht alle möglichen erweiterten Einstellungen tatsächlich in der GUI verfügbar. Gegebenenfalls kann dann der Zugriff auf die in der *.odb-Datei befindlichen content.xml helfen. Hier ein Beispiel, das gerade beim Umstieg von LO 6.0 zu LO 6.1 Probleme bereitete: Unterabfragen in MySQL waren nicht mehr möglich, weil die Weitergabe des verbindenden Wertes (Parameter) unterbunden wurde. Das gleiche Problem hat auch Firebird, wenn die Firebird-Datenbank über den Migrationsassistenten erstellt wurde - zumindest in allen Versionen LO 6.*.

Der nicht funktionierende Code:

```
<db:driver-settings db:system-driver-settings="" db:base-dn=""
db:parameter-name-substitution="false"/>
```

Wird dieser Code geändert auf

```
<db:driver-settings db:system-driver-settings="" db:base-dn=""/>
```

so funktionieren die Unterformulare wieder mit der *.odb-Datei.

Alternativ kann auch das folgende Makro von der migrierten Base-Datei aus gestartet werden:

```
SUB FB_Parameter
    DIM oSettings AS OBJECT
```

```
    oSettings = ThisComponent.DataSource.Settings
    oSettings.ParameterNameSubstitution = True
END SUB
```

Nach einmaligem Start des Makros ist der Eintrag in der Base-Datei komplett verschwunden, sofern die Base-Datei einmal abgespeichert wurde.

Maskierung von Tabellennamen und Feldnamen bei unterschiedlichen Datenbanksystemen

Grundsätzlich ist es in SQL möglich, Tabellennamen und Feldnamen ohne Maskierung zu schreiben. Ein Code wie

```
SELECT VORNAME, NACHNAME FROM PERSONENTABELLE
```

funktioniert genauso wie

```
SELECT "VORNAME", "NACHNAME" FROM "PERSONENTABELLE"
```

Je nach Datenbanksystem setzt Base die Maskierung mit doppelten Anführungszeichen automatisch im Abfrageeditor und auch bei der Eingabe über **Extras → SQL**. Bei den internen Datenbanksystemen **HSQLDB und Firebird** kann es aber schon Probleme geben, wenn Tabellennamen und Feldnamen nicht grundsätzlich groß geschrieben werden. Werden die folgenden Eingaben in dem Abfrageeditor im SQL-Modus eingegeben und dabei die direkte Ausführung von SQL ausgewählt, so führt das bei einer Eingabe zu einer Fehlermeldung:

```
SELECT "Vorname", "Nachname" FROM "PersonenTabelle"
```

funktioniert, wenn eben die Felder auch Kleinbuchstaben enthalten. Das kann auch auf Sonderzeichen ausgedehnt werden.

```
SELECT Vorname, Nachname FROM PersonenTabelle
```

funktioniert hingegen nicht, weil der SQL-Code direkt an die Datenbank weiter gegeben wird ohne die Bezeichner zu maskieren. Die interne Datenbank sucht dann nach einer Tabelle und Feldern in Großbuchstaben, die aber nicht vorhanden sind.

Bei externen Datenbanken hängt die Maskierung von dem entsprechenden Datenbanksystem ab. Für **MySQL/MariaDB** schreibt Base direkt die passende Maskierung in den SQL-Code, der durch den Abfrageeditor erstellt wird:

```
SELECT `Vorname`, `Nachname` FROM `PersonenTabelle`
```

Die Maskierung für MySQL und MariaDB ist also hier nicht das doppelte Anführungszeichen sondern der Gravis `.