LibreOffice
The Document Foundation

Base

Kapitel 7 Datenbank-Anbindung

Copyright

Mitwirkende/Autoren

Robert Großkopf | Jost Lange | Jochen Schiffers

Michael Niedermair

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 15.08.2022. Basierend auf der LibreOffice Version 7.4.

Inhalt

Kapitel 7 Datenbank-Anbindung 1
- Allgemeines zur Datenbank-Anbindung 4
- Anmeldung der Datenbank 4
- Datenquellenbrowser 5
  - Daten in Text 7
  - Daten in Felder 10
  - Seriendruck 10
  - Aktuelle Dokument-Datenquelle 11
  - Explorer ein/aus 11
- Serienbrieferstellung 11
- Erstellung von Etiketten 17
- Serienbriefe und Etiketten direkt erstellen 21
  - Serienbrieffelder mit der Maus erstellen 21
  - Serienbrieffelder über Feldbefehle erstellen 21
- Externe Formulare 23
  - Vorteil externer Formulare 25
  - Nachteil externer Formulare 25
- Datenbanknutzung in Calc 26
  - Daten in Calc einfügen 26
  - Daten aus Calc in eine Datenbank exportieren 30
- Daten von einer Datenbank zu einer anderen konvertieren 31
- Daten über die Zwischenablage in eine Tabelle einfügen 31
- Datenimport aus PDF-Formularen 32
  - Erstellen eines PDF-Formulars 32
  - Auslesen der Daten aus dem PDF-Formular 33

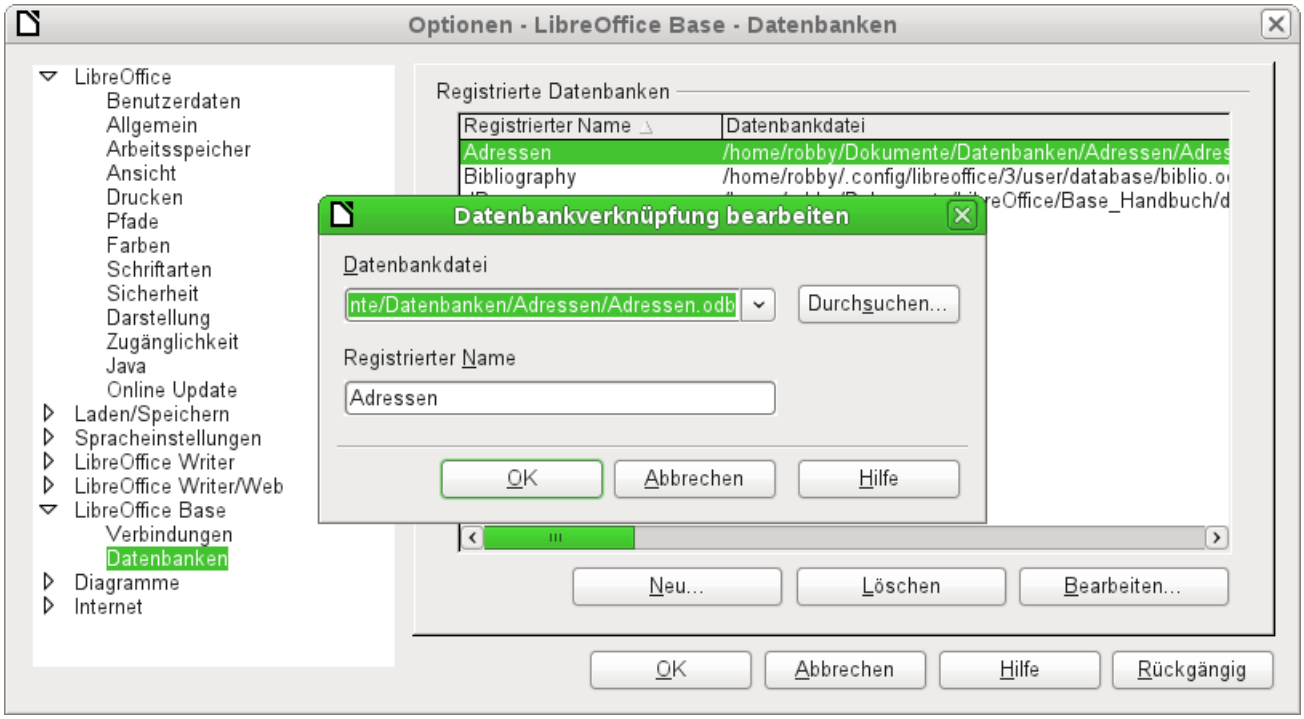
Allgemeines zur Datenbank-Anbindung

Base lässt sich in LibreOffice Writer und LibreOffice Calc auf verschiedene Weise als Datenquelle nutzen. Die Nutzung von Base ist dabei nicht unbedingt an eine Anmeldung der Datenbank in den Einstellungen von LibreOffice gebunden. So interagieren externe Formulare auch direkt mit Base, sofern dort als Datenquelle der Weg zur Datenbank angegeben wird.

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 eingängigerer 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.

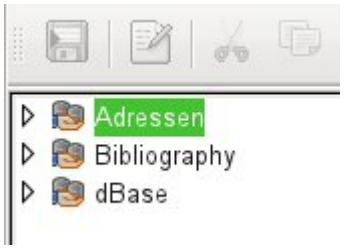
Hinweis	Wird eine Datenbankverknüpfung bearbeitet, so lässt die Verknüpfung sich anschließend zumindest bis einschließlich LO 7.3 nur mit verändertem registrierten Namen speichern: *Bug 149195*. Für den Dialog ist der ursprüngliche registrierte Name bereits vergeben. Um eine Abspeicherung mit gleichem Namen wie ursprünglich zu erreichen muss also der Dialog erneut aufgerufen werden. Jetzt wird wieder der ursprüngliche Name gewählt und die Abspeicherung mit ursprünglichem Namen gelingt.

Datenquellenbrowser

Über den Datenquellenbrowser im Writer oder in Calc erhalten sie Zugriff auf Tabellen und Abfragen der registrierten Datenbanken unter dem dort angegebenen registrierten Namen. Der Browser öffnet sich über **Ansicht → Datenquellen** oder nach dem Drücken der Tastenkombination *Strg+Umschalt+F4* oder über das entsprechende, in der Grundeinstellung nicht eingeblendete, Symbol in der Standard-Symbolleiste.

Hinweis	Die folgende Beschreibung des Datenquellenbrowser richtet sich nach den Funktionen im **Writer** aus. Für **Calc** siehe *Datenbanknutzung in Calc.*

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.

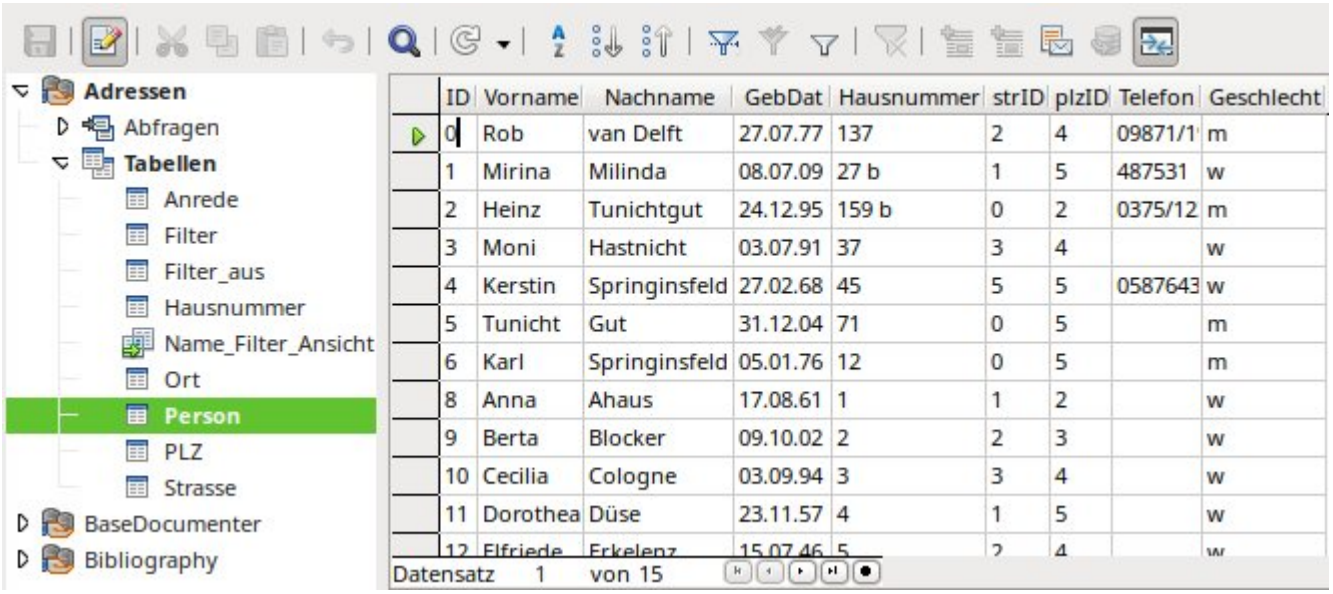


Durch einen Klick auf das ⊳ - Zeichen vor den Datenbankbezeichnungen öffnet sich die Datenbank und zeigt an, dass sich darin ein Unterordner für Abfragen und ein Unterordner für Tabellen befindet. Die weiteren Unterordner der Datenbank werden nicht zur Verfügung gestellt. Interne Formulare sowie Berichte 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.



Von den Zeichen in der Symbolleiste sind viele bereits aus der Eingabe von Daten in Tabellen bekannt. Neu ist insbesondere der letzte Abschnitt: **Daten in Text**, **Daten in Felder**, **Seriendruck**, **Aktuelle Dokument-Datenquelle**, **Explorer ein/aus**.

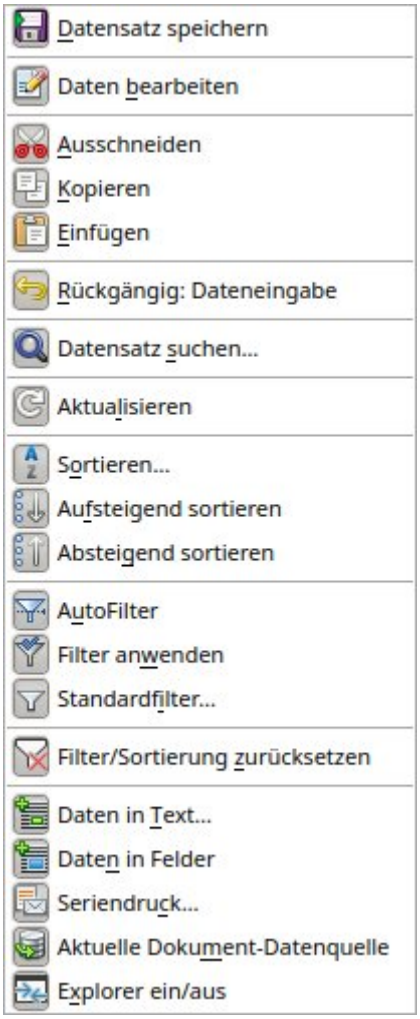
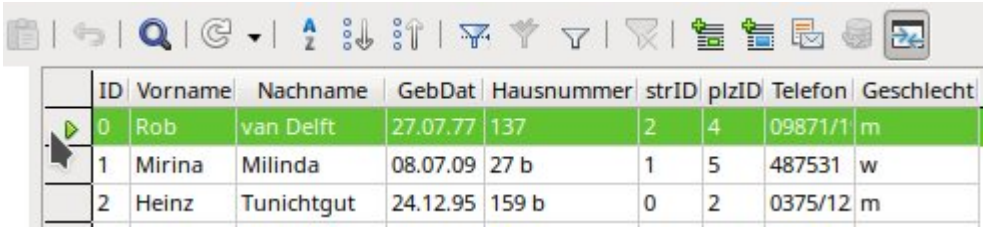


Abbildung 1: Symbolleiste Datenquellenbrowser

Hinweis	Der **Datenquellenbrowser** eignet sich auch für die Dateneingabe. Dies sollte bei ~~Firebird~~ so lange **nicht zur Eingabe und Änderung von Daten** genutzt werden, wie diese Datenbank noch das Abspeichern in der *.odb-Datei erfordert. Sonst scheinen die Daten geändert, werden aber nicht gesichert.

Daten in Text

Die Funktion **Daten in Text** steht zur Verfügung, sobald ein Datensatz markiert wurde.



	ID	Vorname	Nachname	GebDat	Hausnummer	strID	plzID	Telefon	Geschlecht
▶	0	Rob	van Delft	27.07.77	137	2	4	09871/1	m
	1	Mirina	Milinda	08.07.09	27 b	1	5	487531	w
	2	Heinz	Tunichtgut	24.12.95	159 b	0	2	0375/12	m

Abbildung 2: Markierung eines Datensatzes

Wird jetzt **Daten in Text** angewählt, so erscheint ein Assistent, der die erforderlichen Formatierungen vornimmt:

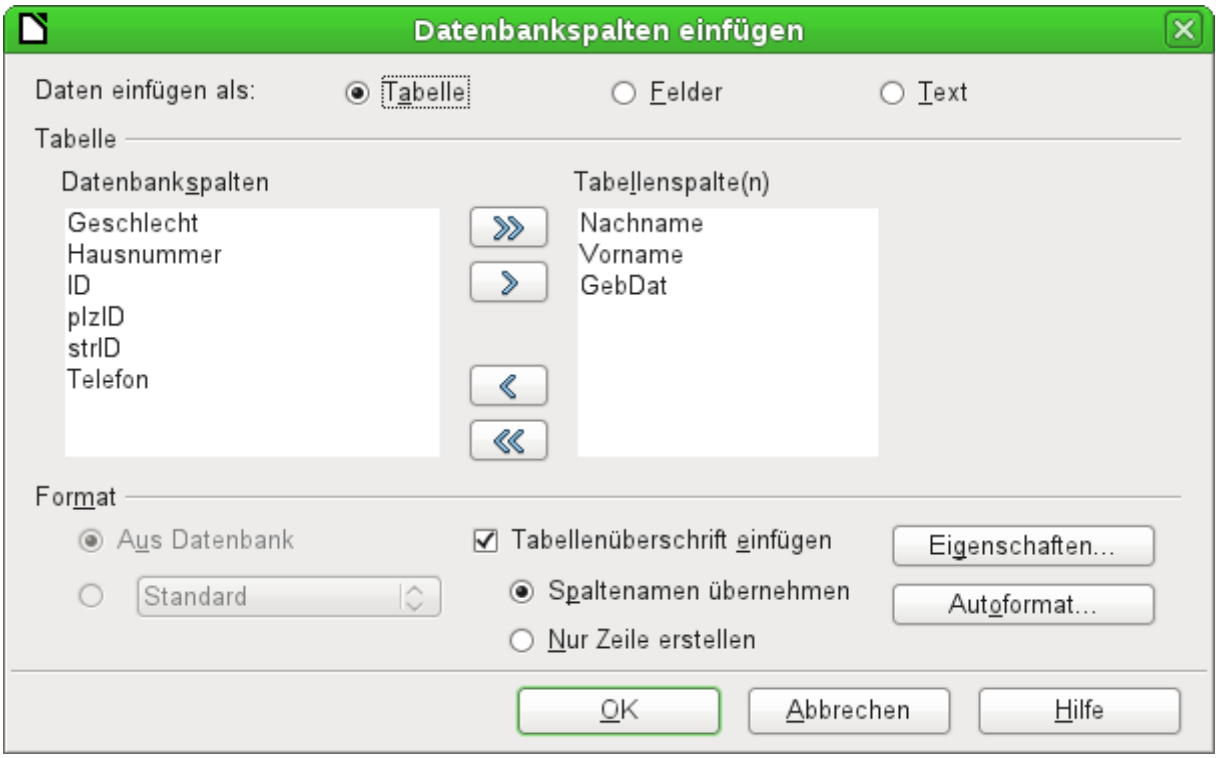


*Abbildung 3: **Daten Einfügen als → Tabelle***

Für das Einfügen von Daten in den Text stehen die Möglichkeiten der Darstellung in Tabellenform, als einzelne Felder oder als Gesamttext zur Verfügung.

Die obige Abbildung zeigt den Aufbau **Daten einfügen als → Tabelle**. Bei Zahlenfeldern und Datumsfeldern kann das Format der Datenbank durch ein eigenes gewähltes Format geändert werden. Ansonsten erfolgt die Formatierung mit der Auswahl der Tabellenfelder. Die Reihenfolge der Felder wird über die Pfeiltasten festgelegt.

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.

Über den Button Autoformat werden einige vorformatierte Tabellenansichten angeboten. Bis auf das Vorschlagsformat «*Standard*» können alle hier enthaltenen Formatierungen umbenannt werden.

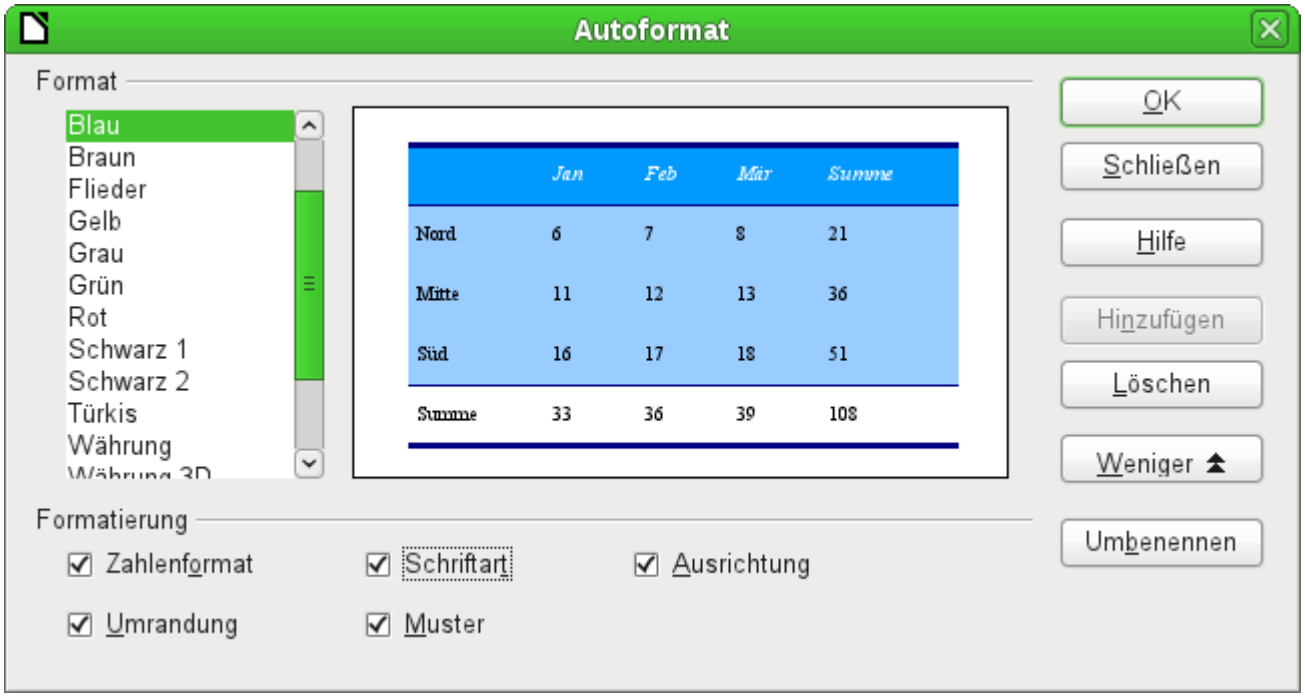


Abbildung 4: Über Autoformat *stehen Tabellenformate zur Auswahl*

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.

Über **Daten einfügen als Felder** wird die Möglichkeit geboten, in einem kleinen Editor die verschiedenen Tabellenfelder hintereinander in einem Text zu positionieren. Dem erstellten Text kann außerdem eine Absatzvorlage zugewiesen werden. Auch hier ist die Formatierung von Datums- und Zahlenwerten separat wählbar, kann aber auch direkt aus den Tabelleneinstellungen der Datenbank gelesen werden.

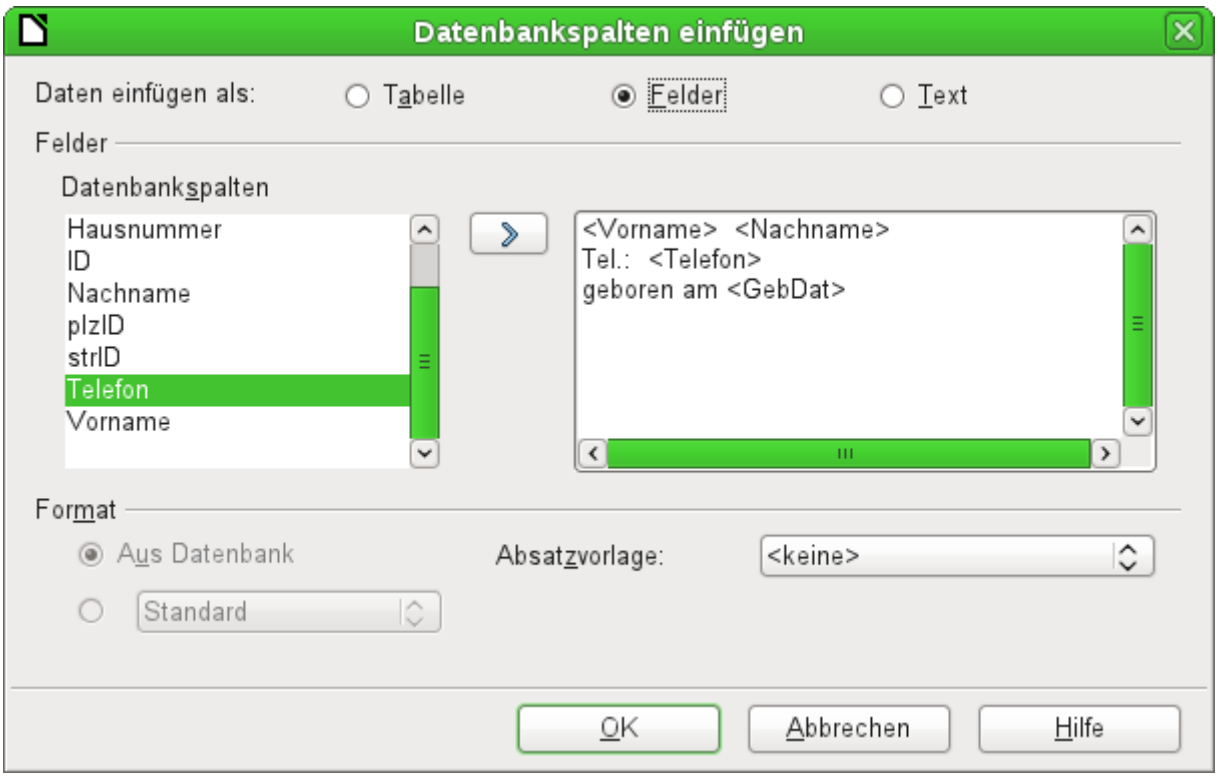


Abbildung 5: ***Daten einfügen als → Felder*** *entspricht auch dem Fenster für* ***Daten einfügen als → Text***

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 in sofern, 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:

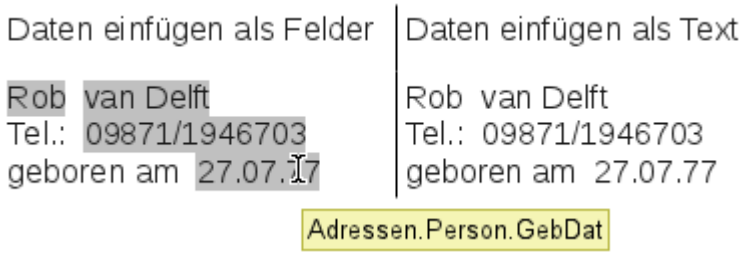


Abbildung 6: Vergleich: Daten als Felder - Daten als Text

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.

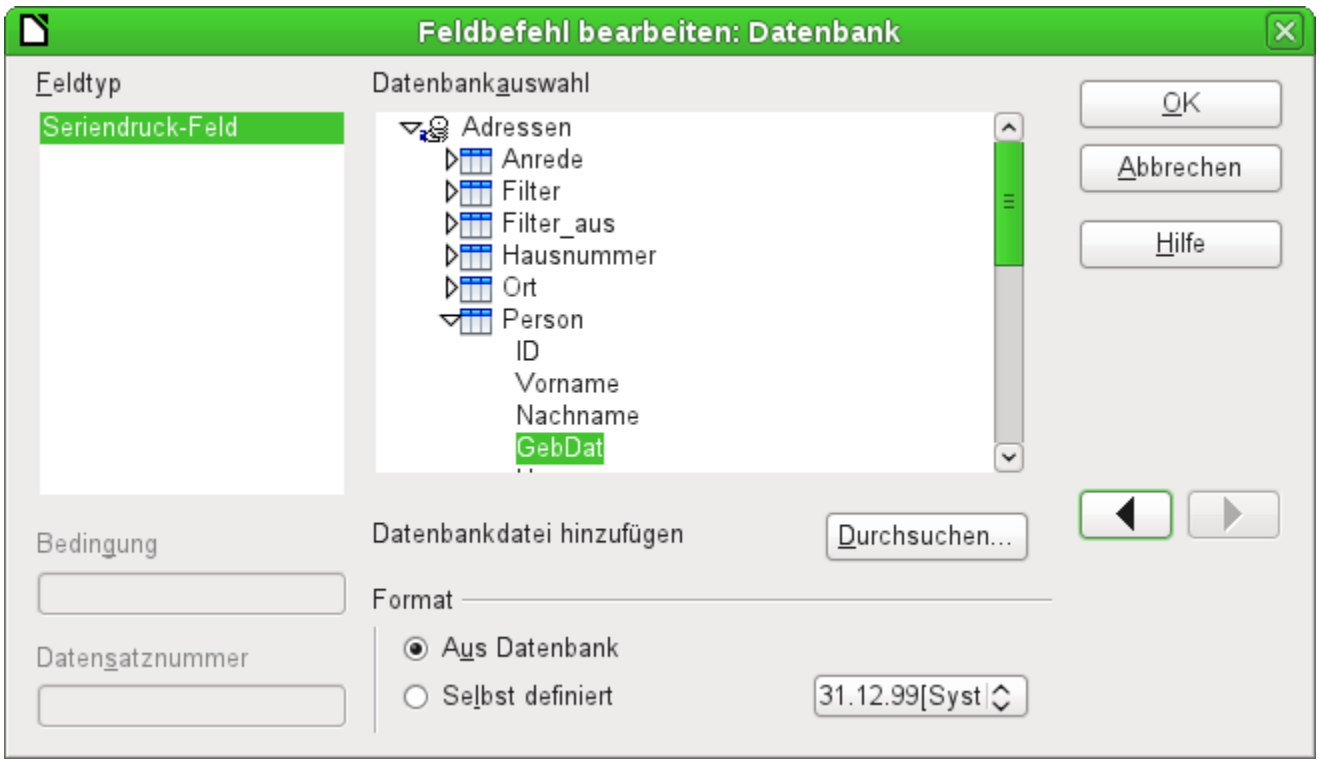


Abbildung 7: Doppelklick auf ein eingefügtes Feld zeigt die Eigenschaften des Seriendruckfeldes

Ein Doppelklick z.B. auf das Feld "*GebDat*" öffnet schließlich folgende Übersicht. Hier wird dann klar, welches Feld letztlich über die Funktion **Daten einfügen als → Felder** erstellt wurde. Es ist der gleiche Feldtyp, der auch über **Einfügen → Feldbefehl → Weitere Feldbefehle → Datenbank** erzeugt wird.

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:

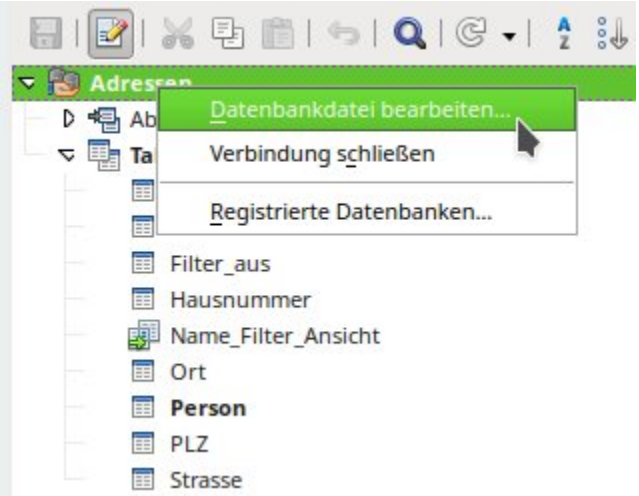
Daten einfügen als Felder	Daten einfügen als Text
Mirina Milinda Tel.: 487531 geboren am 08.07.09	Rob van Delft Tel.: 09871/1946703 geboren am 27.07.77

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 dem Button Seriendruck 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

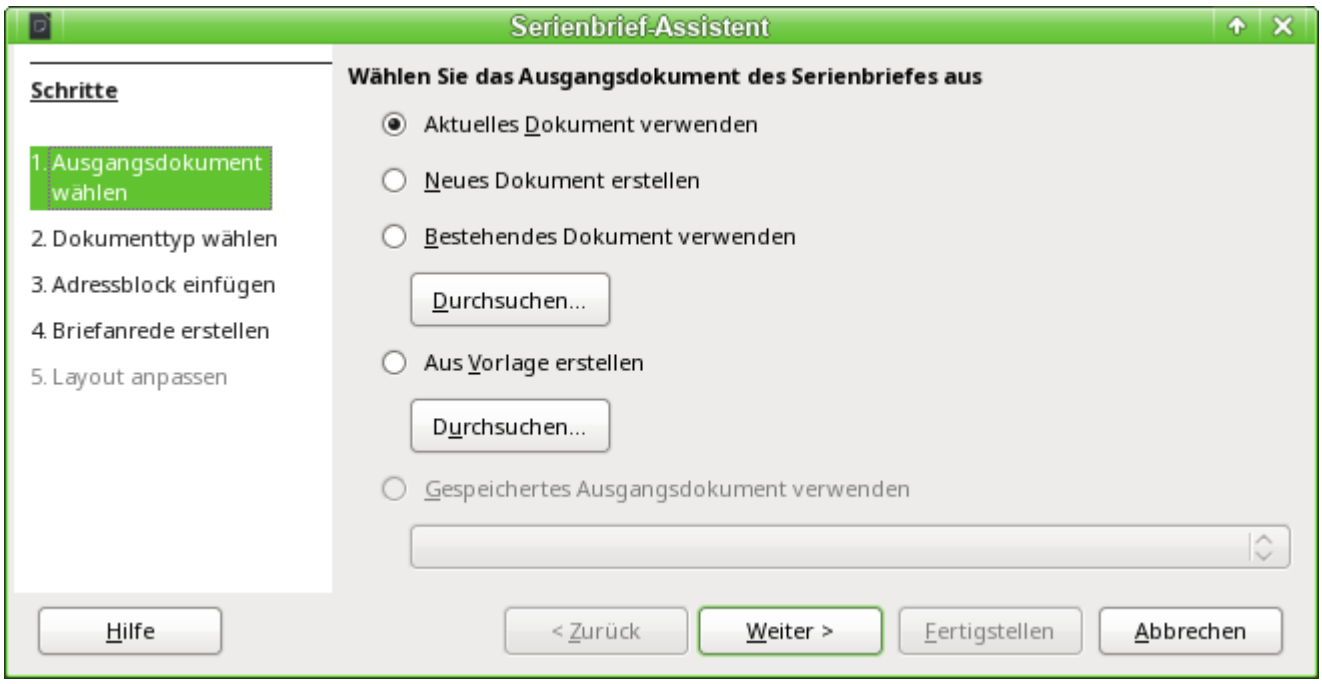
Ein Klick auf den Button Aktuelle Dokument-Datenquelle öffnet direkt die Ansicht auf die Tabelle, die die Grundlage für die Daten bildet, die in das Dokument eingefügt wurden. Im obigen Beispiel zeigt sich also sofort die Tabelle "*Person*" aus der Datenbank "*Adressen*".

Explorer ein/aus

Mit Betätigung des Buttons Explorer ein/aus 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.

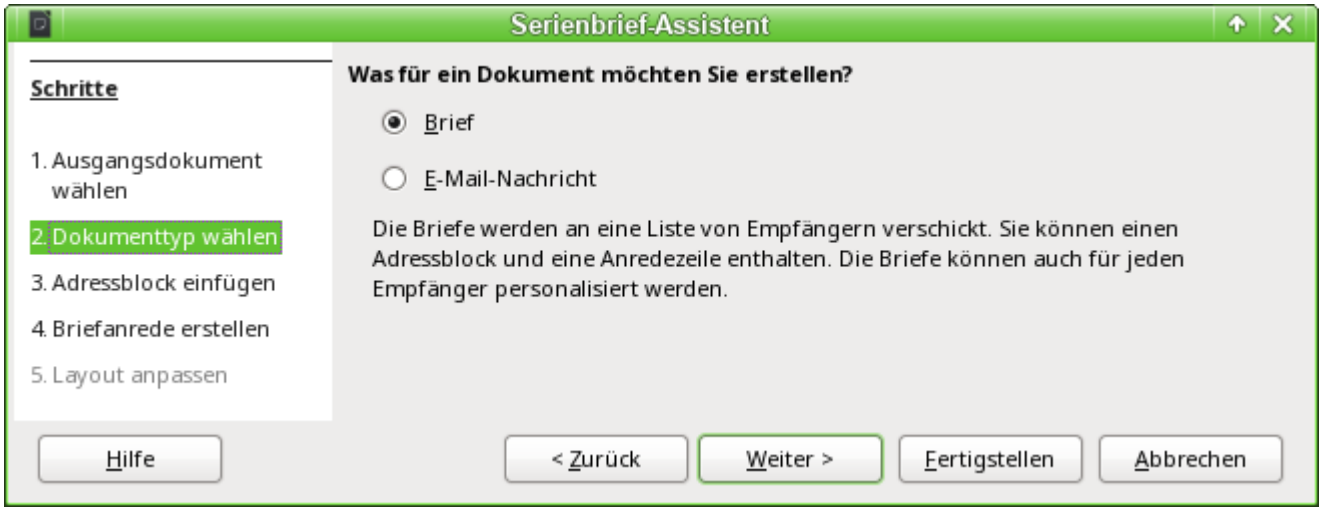
Serienbrieferstellung

Über den Datenbankbrowser ist auch der Serienbriefassistent zugänglich. Mit diesem Assistenten wird kleinschrittig das Adressfeld und die Anrede anhand einer Datenquelle nachvollzogen. Prinzipiell ist die Erstellung von solchen Feldern natürlich auch ohne den Assistenten möglich. Hier wird einmal beispielhaft der gesamte Assistent durchgearbeitet.

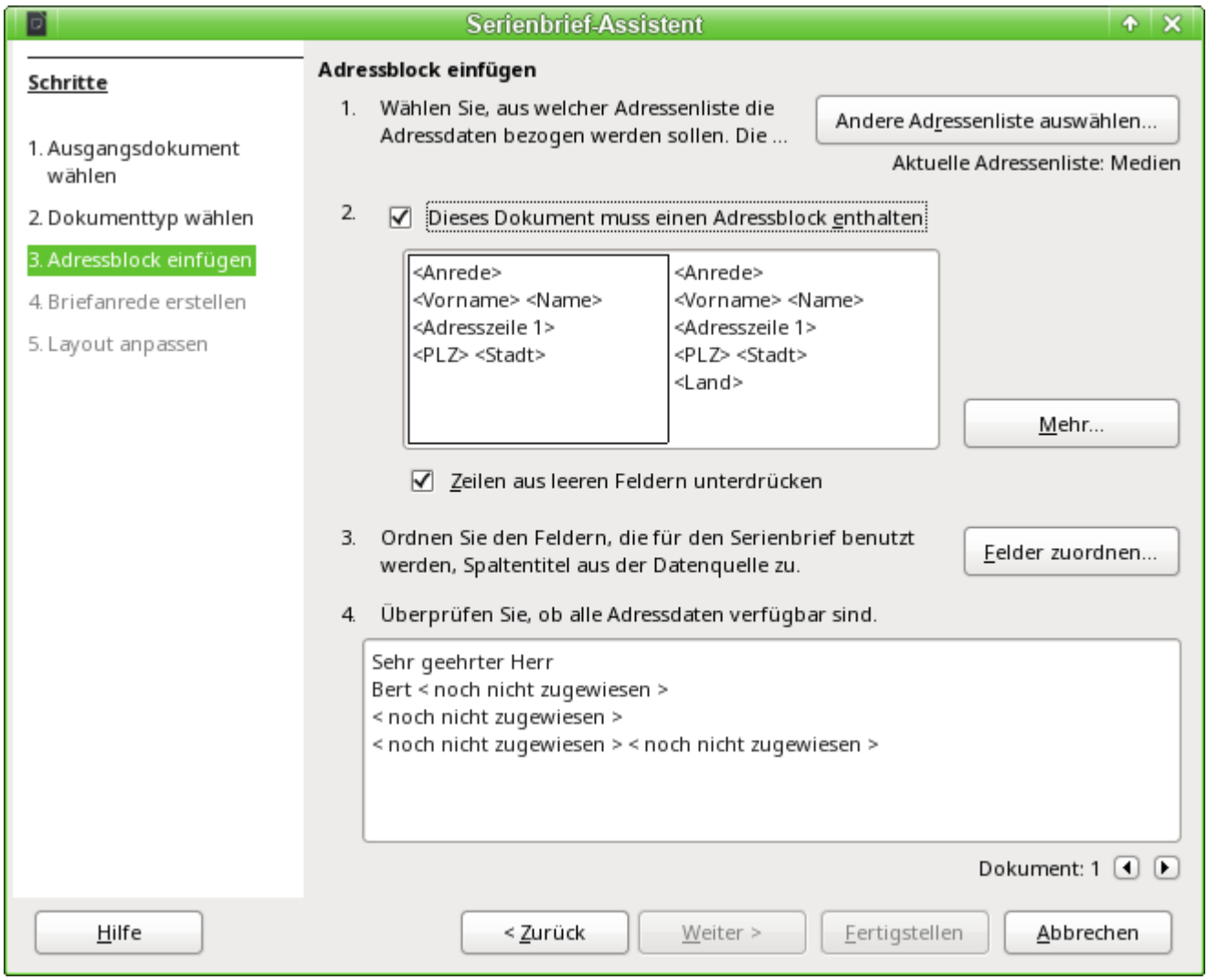


Das **Ausgangsdokument** des Serienbriefs ist das Dokument, mit dem die **Datenbankfelder verknüpft** werden.

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.

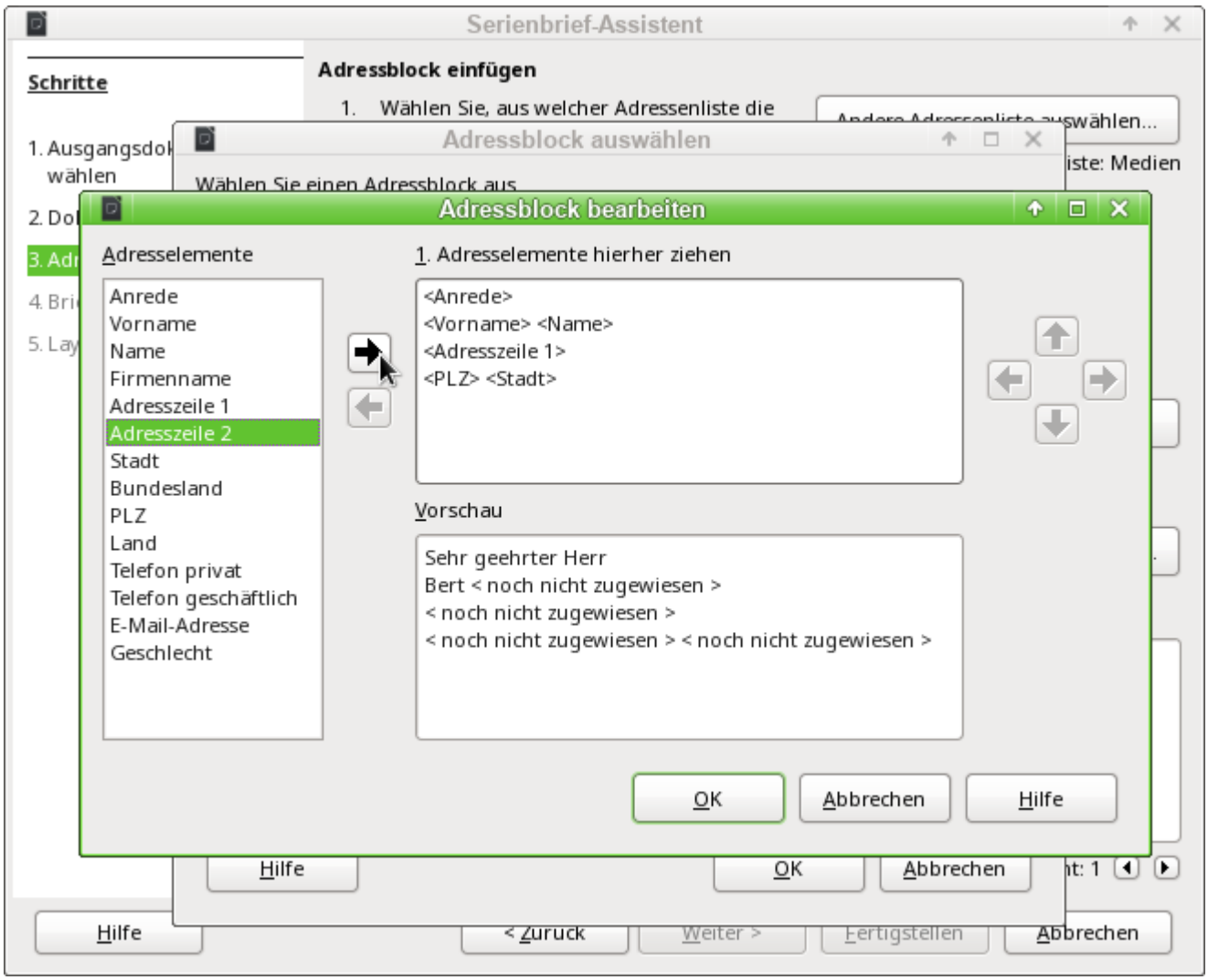
Unter Punkt 2 wird das prinzipielle Aussehen des Adressblocks festgelegt. Dieser Adressblock kann entsprechend über den Button Mehr angepasst werden. Siehe hierzu die nächste Abbildung.

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 Felder des Assistenten.

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 < *noch nicht zugewiesen* > , 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.

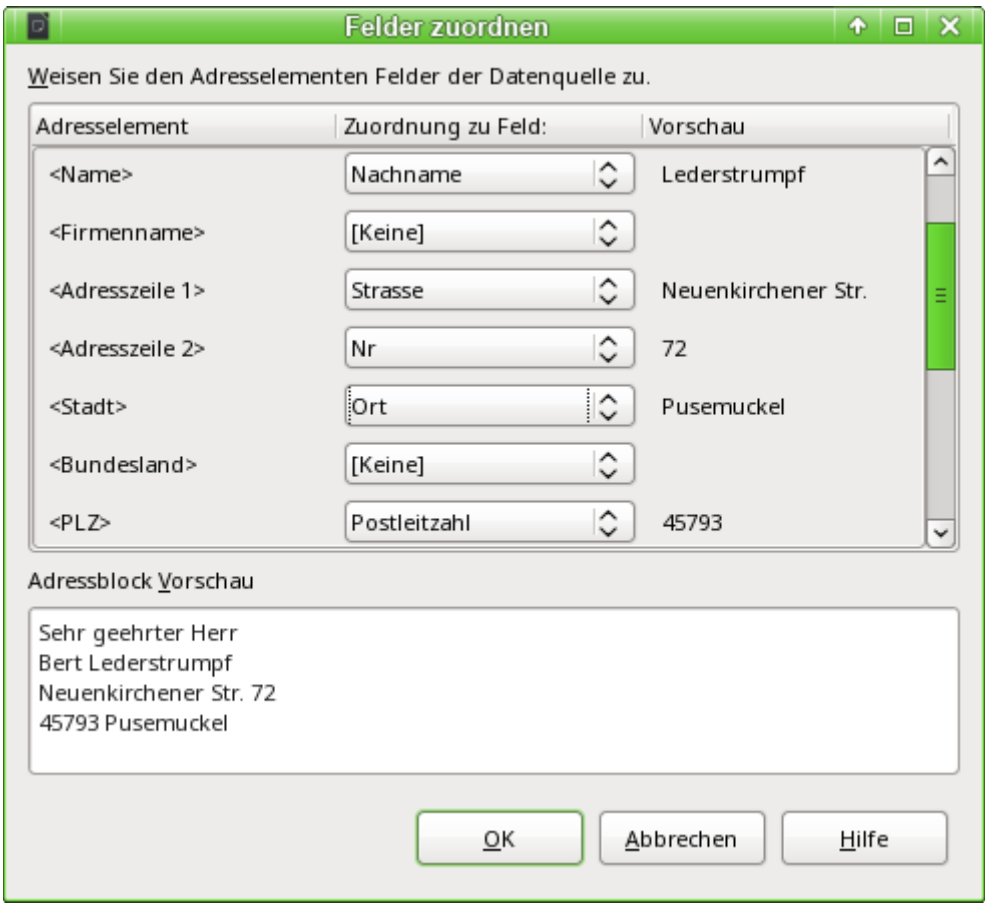


Im Hintergrund ist zu erkennen, dass bei der Nachbearbeitung zuerst einmal wieder eine erweitere Auswahl von Adressblöcken präsentiert wird. Der passendste Adressblock wurde hier ausgewählt und entsprechend bearbeitet.

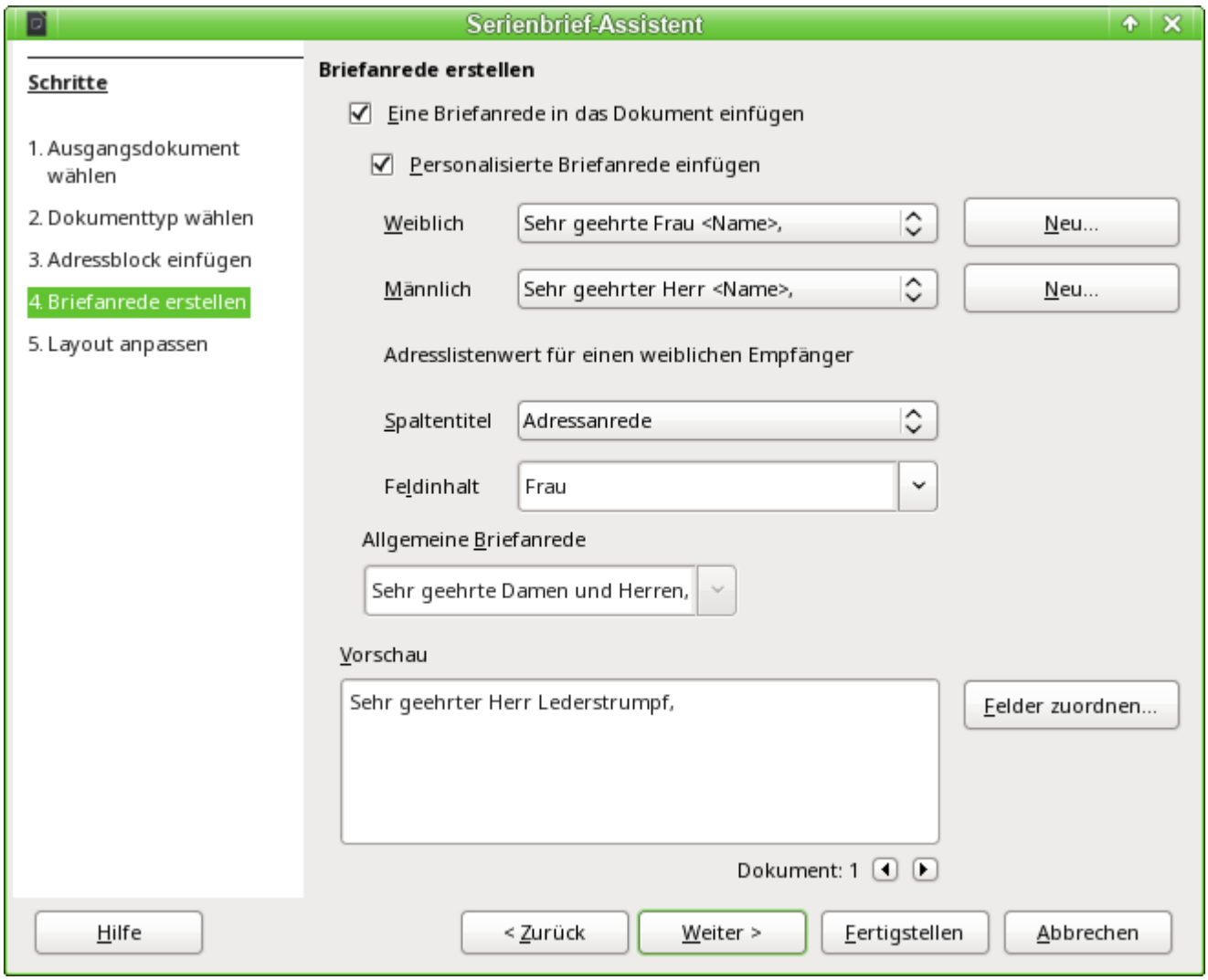
Mit den Pfeiltasten werden Felder in den Adressblock übernommen bzw. aus dem Adressblock entfernt. Der Adressblock kann nicht direkt editiert werden. Stattdessen wird die Anordnung der Felder über die auf der rechten Seite sichtbaren Pfeiltasten hergestellt.

Für die Adressanrede wurde einfach das Element «*Anrede*» eingefügt. Bis auf den Vornamen müssen jetzt noch alle anderen Felder entsprechend zugewiesen werden.

In unserem Fall muss auch das Element «*Anrede*» anders zugewiesen werden, da ein entsprechendes Feld mit etwas anderem Inhalt in der Abfrage existiert und hier aufgrund der Namensgleichheit als "Adressanrede" benutzt wird. Die "Adressanrede" für Rob lautet in der Abfrage der Datenbank «Herrn», die "Anrede" «Herr» für eine gegebenenfalls gesondert zu erstellende Anredezeile zum Beginn des Briefinhaltes.



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.



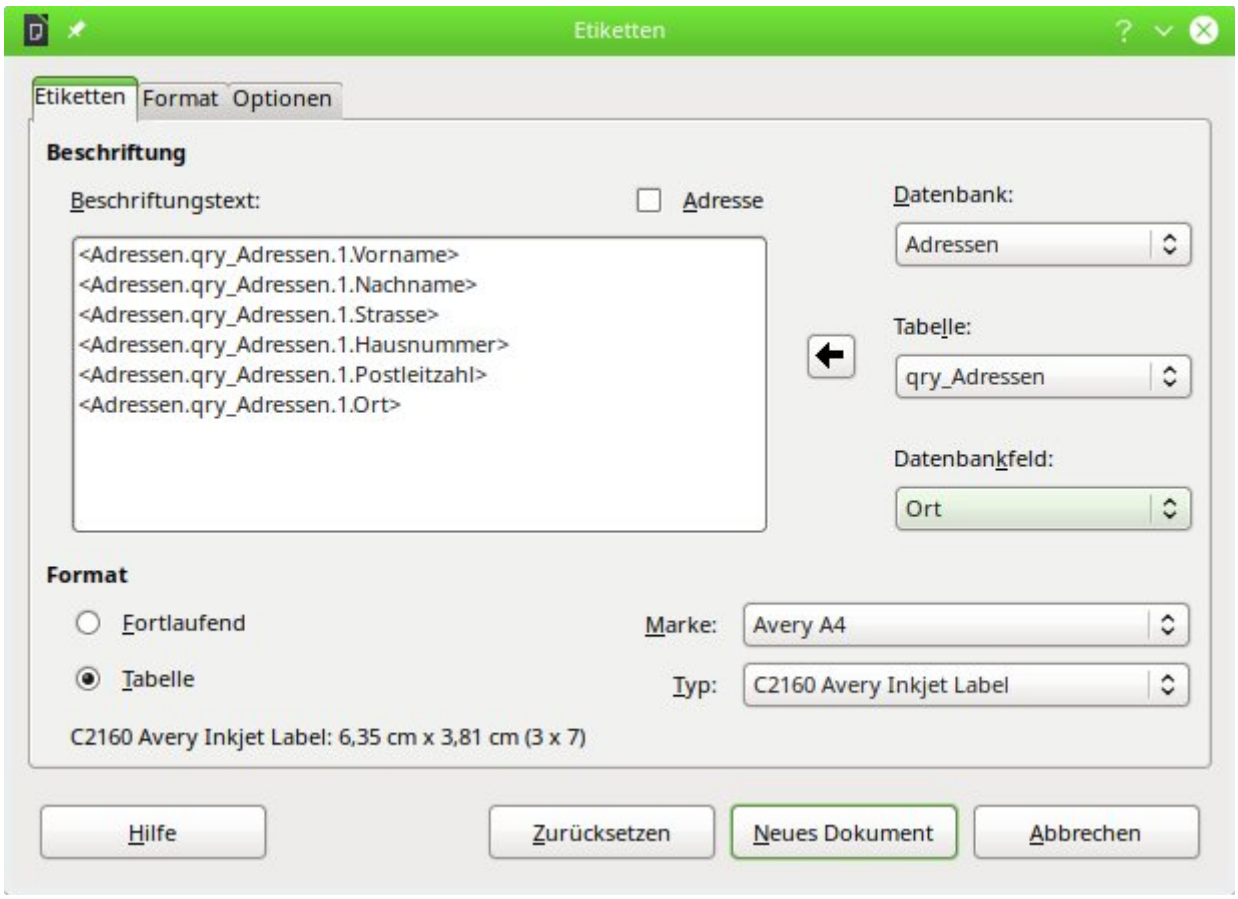
Die wesentlichen Datenbankeinstellungen werden mit dem 4. Schritt eigentlich schon beendet. Hier geht es lediglich darum, aus welchem Feld das Geschlecht des Adressaten abgelesen werden soll. Dies war bereits so benannt, dass nur noch der Feldinhalt für den weiblichen Empfänger gekennzeichnet werden musste.

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.

In der Regel ist das Dokument erst einmal eine Rohform, so dass es für den Gebrauch im Writer noch weiter bearbeitet werden kann.

Erstellung von Etiketten

Über **Dateien → Neu → Etiketten** wird der Assistent zur Erstellung von Etiketten gestartet. Es öffnet sich ein Fenster, das alle Formatierungs- und Inhaltsfragen zu den Etiketten abfragt, bevor die Etiketten selbst erstellt werden. Die Einstellungen dieses Fensters werden in den persönlichen Einstellungen des Nutzers gespeichert.



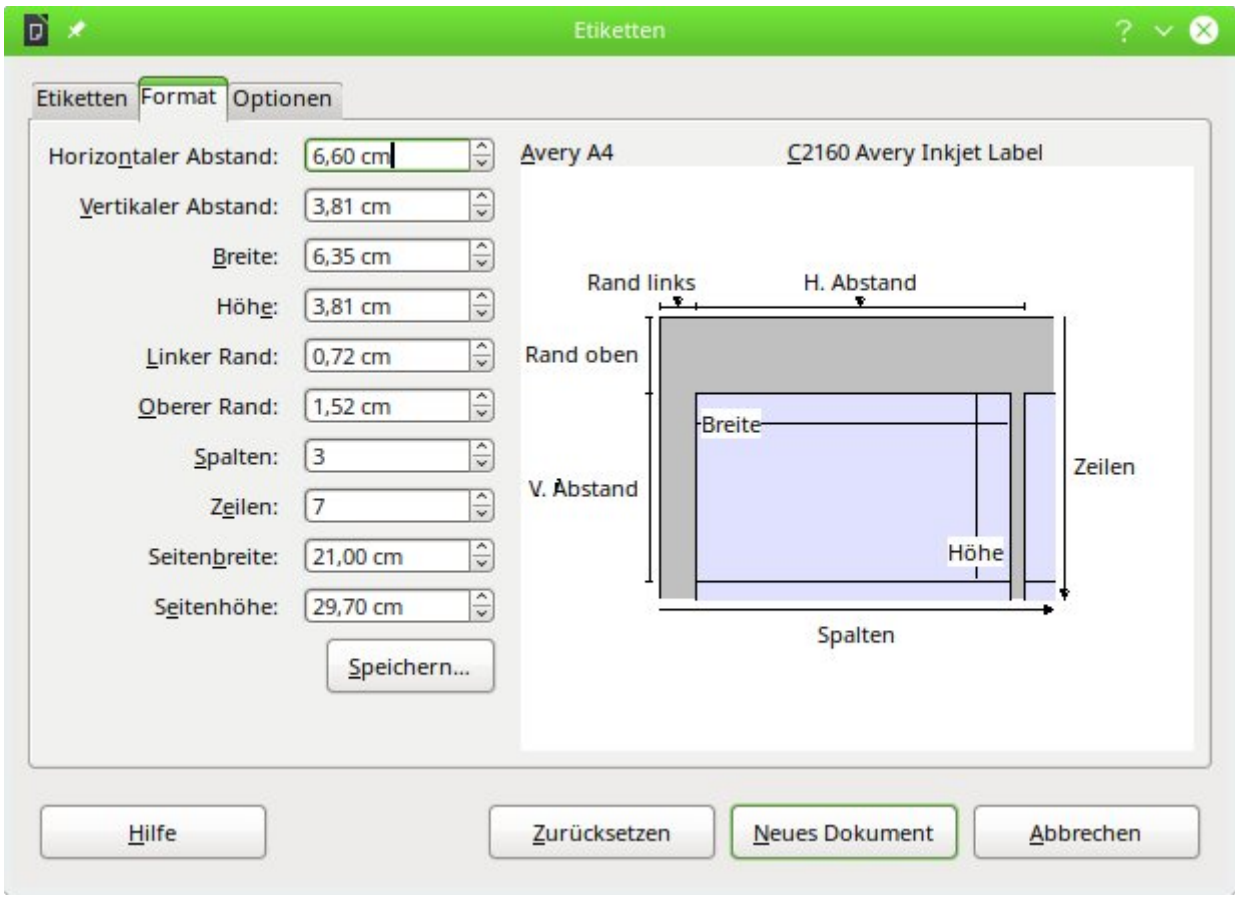
Die Grundeinstellungen zum Inhalt werden im Reiter **Etiketten** erstellt. Wird hier statt des Datenbankinhaltes über das Markierfeld **Adresse** gewählt, so wird jedes Etikett mit dem gleichen Inhalt ausgefüllt, der aus den Einstellungen in **Extras → Optionen → LibreOffice → Benutzerdaten** gelesen wird.

Als Beispieldatenbank dient hier wieder die **Datenbank → Adressen**. Auch wenn über dem nächsten Selektionsfeld der Begriff «*Tabelle*» steht, werden hier **Tabellen und Abfragen** gelistet, wie sie im Datenquellenbrowser verfügbar sind.

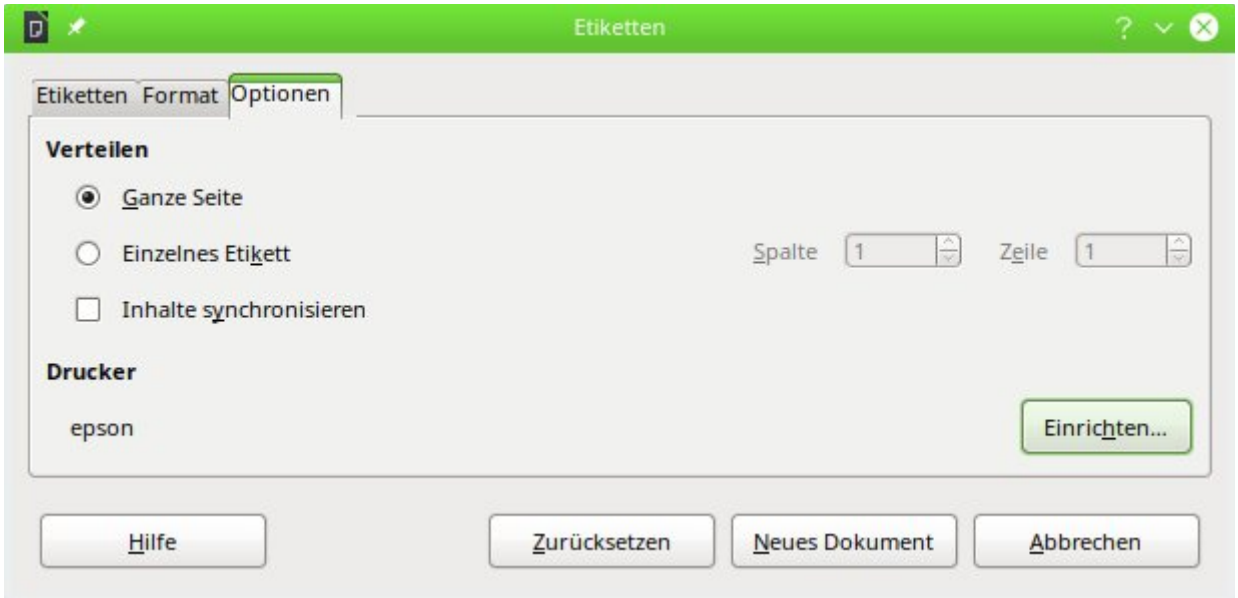
Über die Pfeiltasten werden die einzelnen Datenbankfelder in den Editor eingefügt. Die Bezeichnung für das Datenbankfeld "*Vorname*" wird hier zu *<Adressen.qry_Adressen.1.Vorname>*. Die Reihenfolge ist also *<Datenbank.Tabelle.1.Datenbankfeld>*.

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 komplett und gut sichtbar gedruckt werden kann.

Das Format kann im Reiter **Etiketten** vorgewählt werden. Hier sind viele Etikettenmarken eingearbeitet, so dass meist keine weiteren Einstellungen im Reiter **Format** notwendig sind.



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 wäre hier bei Etiketten der Breite 7,00 cm, die Seitenbreite geringfügig größer als 3*7,00 cm = 21,00 cm zu stellen. Erst dann werden 3 Etiketten nebeneinander auf einer Seite ausgegeben, wenn gleichzeitig der linke Rand 0 cm und der horizontale Abstand 7,00 cm ist.



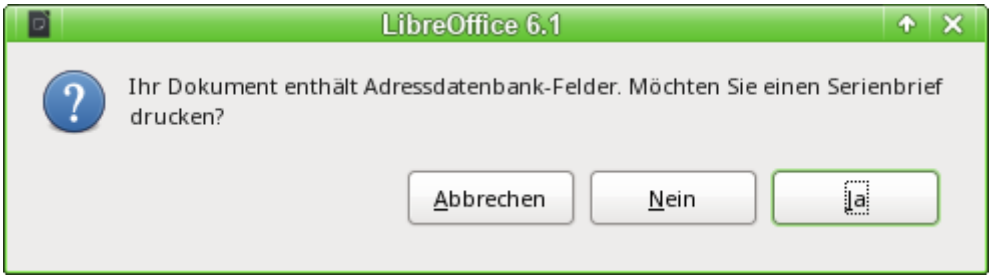
Unter dem Reiter **Zusätze** kann schließlich noch eingestellt werden, ob nur ein einzelnes Etikett oder eine ganze fortlaufende Seite erstellt werden soll. Die fortlaufende Seite wird dann, beginnend mit dem ersten Datensatz, mit Daten aus der Datenbank bestückt. Sind mehr Datensätze

vorhanden als auf eine Seite passen, so wird automatisch das nächste Blatt mit den fortlaufenden Daten versehen.

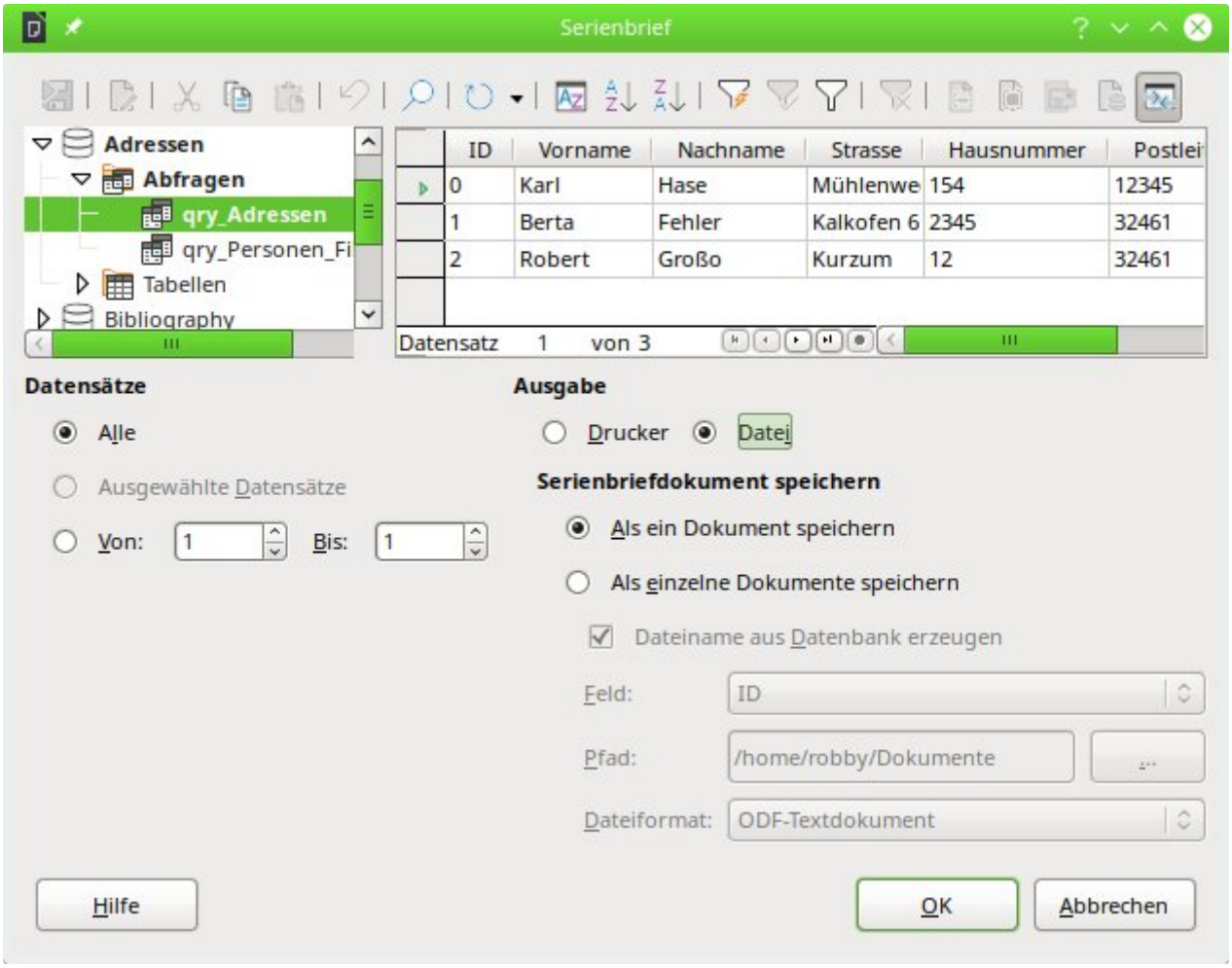
Mit dem Markierfeld **Inhalte synchronisieren** werden alle Etiketten miteinander verbunden, so dass die anschließenden Änderungen im Layout einer Etikette auf die anderen Etiketten übernommen werden. Um den editierten Inhalt zu übertragen, muss lediglich die eingeblendete Schaltfläche mit dem Aufdruck 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.



Ist die Abfrage ausgesucht und sind die entsprechenden Datensätze ausgewählt (hier: **Alle**), so kann mit dem Druck begonnen werden. Ratsam vor allem bei ersten Tests ist **Ausgabe → Datei**. So wird der Inhalt als ein Dokument gespeichert wird. Die Speicherung in mehrere Dokumente

dient hier nicht dem Etikettendruck, sondern dem Druck von Briefen an unterschiedliche Personen, die so separat nachbearbeitet werden können.

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

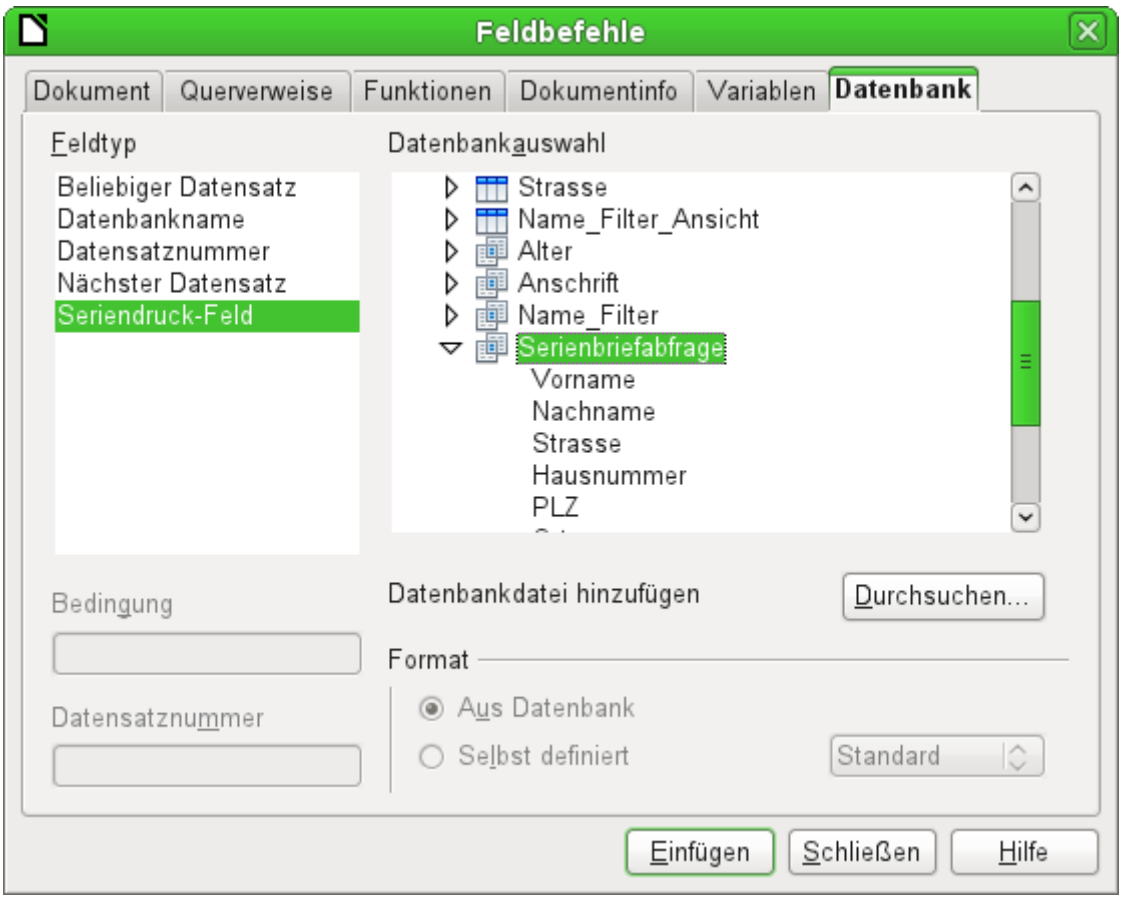
Serienbrief-Felder können mit der Maus aus dem Datenbankbrowser heraus erstellt werden:

Der Tabellenkopf wird mit der linken Maustaste markiert. Die Maustaste wird gedrückt gehalten und der Cursor in das Textdokument gezogen. Der Cursor verändert sein Symbol zu einem Einfügesymbol. In dem Textdokument wird schließlich das Serienbrieffeld eingefügt, das hier in der kompletten Beschreibung über **Ansicht → Feldname** sichtbar gemacht wurde.

Auch aus einer nicht angemeldeten Datenbank heraus können Serienbrieffelder mit der Maus direkt erstellt werden. Hierzu muss die Datenbank neben dem Writer-Dokument geöffnet sein. Ist eine Tabelle oder Abfrage ausgewählt, so wird mit der linken Maustaste der jeweilige Tabellenkopf markiert und in das Writer-Dokument hinüber gezogen. Der Ablauf entspricht also dem im vorherigen Abschnitt geschilderten Vorgehen.

Serienbrieffelder über Feldbefehle erstellen

Serienbrief-Felder können über **Einfügen → Feldbefehl → Weitere Feldbefehle → 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 → Funktionen → 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

```
001 [Adressen.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:

```
001 [Adressen.Serienbriefabfrage.Geschlecht] != "w"
002 OR NOT [Adressen.Serienbriefabfrage.Nachname]
```

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 «Korrelierte 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:

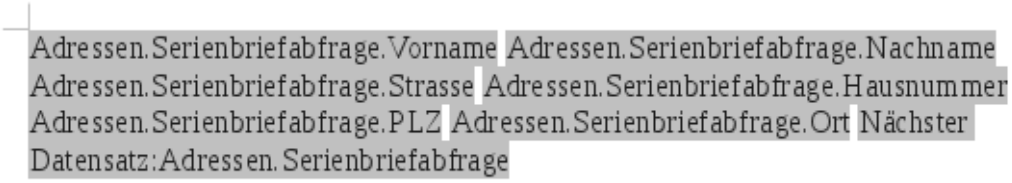


Abbildung 8: Feldbefehle für Etiketten mit fortlaufendem Inhalt

Bei dem letzten Etikett 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». Auch die direkte Auswahl von Elementen über **Formular → Entwurfsmodus** ist seit LO 6.0 möglich. Bei der Erstellung des Formulars erscheint unter **Formular-Eigenschaften → Daten** ein etwas anderer Aufbau als bei Formularen direkt in der Datenbankdatei *.odb:

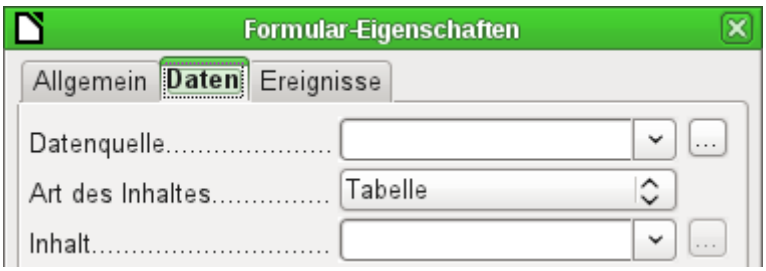


Abbildung 9: Formular mit einer externen ...

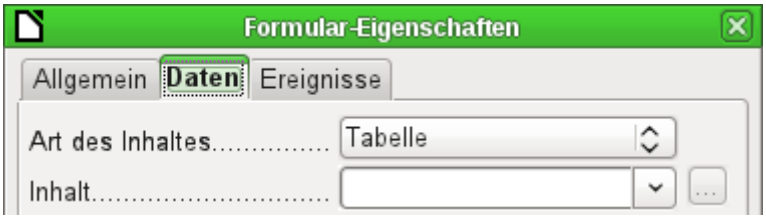


Abbildung 10: ... und einer internen Datenquelle.

Lediglich die **Datenquelle** muss bei externen Formularen zusätzlich ausgewählt werden. Geschieht dies mit dem Button ... rechts von dem Listenfeld zur Datenquelle, so wird der Dateibrowser geöffnet. Eine beliebige *.odb-Datei kann ausgewählt werden. Anschließend steht in dem Feld zur Datenquelle ein Link, der mit der Bezeichnung «file:///» beginnt.

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.

Tipp	Das Erstellen von Formularen kann auch im Schnellverfahren durch Ziehen und Ablegen erfolgen. Dafür ist die Datenbank zu öffnen, die Tabelle bzw. Abfrage aufzusuchen und der jeweilige Tabellenkopf zu markieren. 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. In beiden Fällen wird ein Eingabefeld sowie ein dazugehöriges Beschriftungsfeld mit der Bezeichnung des Feldnamens erzeugt. Beim ersten Einfügen wird gleichzeitig die Verbindung zur Datenquelle erstellt. So kann direkt nach dem Ziehen und Ablegen mit der Eingabe von Daten in diesem Formular begonnen werden.

Tipp	Sind bereits in Base Formulare erstellt worden, so können sie aus Base heraus als externe Formulare abgespeichert werden. Es muss dann lediglich noch die Verbindung zur Datenbank erstellt werden und die Formulare funktionieren außerhalb der *.odb-Datei. Sollten Makros in den Formularen genutzt worden sein, so ist hier natürlich gegebenenfalls noch eine Anpassung notwendig.

Vorsicht

Die Nutzung von externen Formularen kann in Zusammenhang mit FIREBIRD zu Datenverlusten führen. Die Firebird Datenbank benötigt auch in LO 7.1 immer noch die Sicherung der Daten über das Speicher-Symbol der Datenbankdatei.

Nur mittels Makro kann diese Sicherung auch bei externen Formularen erfolgen. Dabei sollte das Makro an die Eigenschaft **Nach der Datensatzaktion** gebunden werden:

```
001  SUB DataSave(oEvent AS OBJECT)
002     oEvent.Source.activeConnection.Parent.flush
003  END SUB
```

Vorteil externer Formulare

Base muss nicht erst geöffnet werden, um mit der Datenbank zu arbeiten. Im Hintergrund ist also nicht immer ein weiteres Fenster der Datenbankoberfläche 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 zugesandt 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.

Tipp	Externe Formulare sollten schreibgeschützt geöffnet werden. Sonst erfolgt beim Schließen die missverständliche Nachfrage, ob das Dokument gespeichert werden soll. Deshalb: **Datei → Eigenschaften → Sicherheit → Datei schreibgeschützt öffnen**. Der Schreibschutz des Betriebssystems erzeugt hingegen immer die Meldung, dass das Dokument schreibgeschützt ist und ob es denn bearbeitet werden soll.

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. Sind die Datenbanken angemeldet, so muss allerdings lediglich der Anmeldename übereinstimmen.

Nur die Formulare sind extern erstellbar, nicht aber Abfragen und Berichte. Ein einfacher Blick auf eine Abfrage muss also über ein Formular erfolgen. Als Darstellung eignet sich da sicher das Tabellenkontrollfeld ganz gut. Ein Bericht hingegen erfordert die Öffnung der Datenbank. Dies kann auch über Makros mit einem «Druck von Berichten aus einem externen Formular her-

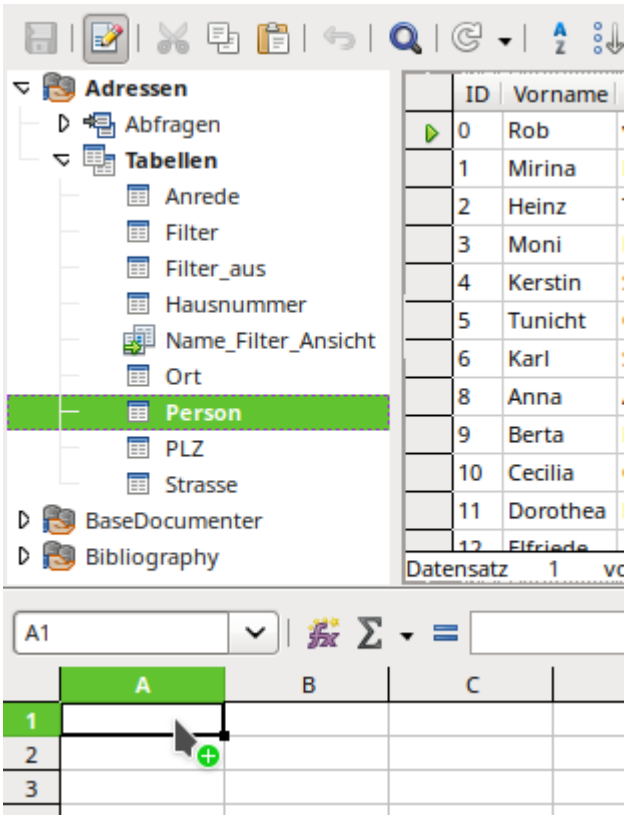
aus» erfolgen. Alternativ dazu kann er zumindest teilweise mit Hilfe von Serienbrieffeldern nachgestellt werden. Zu weiteren Druckmöglichkeiten siehe auch das entsprechende Beispieldatenbankpaket «Serienbrief»[1].

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 Tabelle wird ausgewählt, mit der linken Maustaste markiert und in ein Tabellenblatt von Calc hereingezogen. Mit dem Cursor wird die linke obere Ecke der Tabelle festgelegt. Die Tabelle wird mit Feldbezeichnungen erstellt. Der Datenquellen-Browser bietet bei dieser Aktion nicht erst Daten in Text oder Daten in Felder an.

Die so nach Calc hereingezogenen Daten weisen die folgenden Eigenschaften auf:

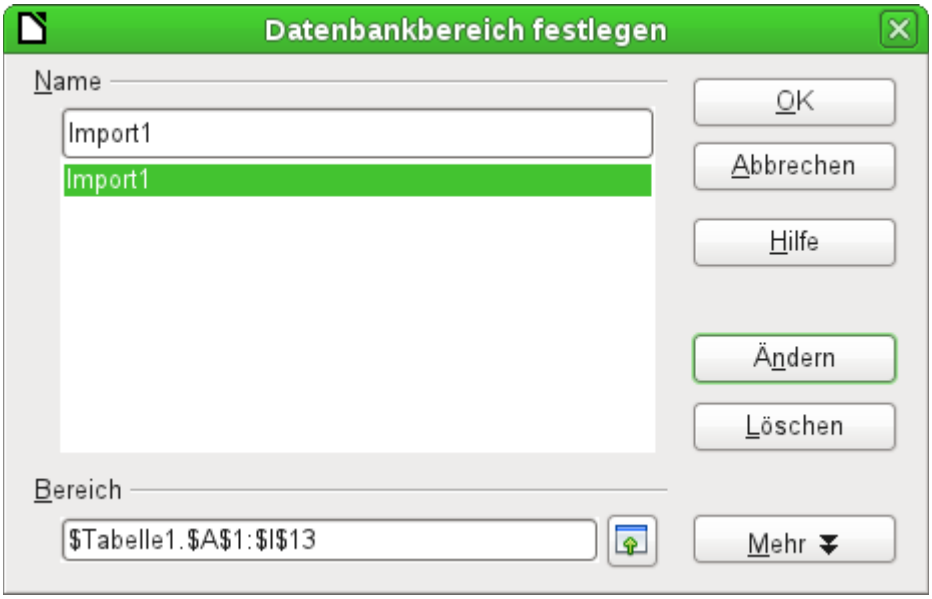
Nicht nur die Daten werden importiert, sondern auch die Eigenschaften der Felder werden ausgelesen und beim Import beachtet. Felder wie z.B. die Hausnummern, die als Textfelder deklariert wurden, werden auch als Text formatiert in Calc eingefügt.

Der Import wird direkt als Bereich eingefügt. Ihm wird standardmäßig der Name Import1 zugewiesen. Über diesen Bereich können die Daten später angesprochen werden. Über **Daten → Bereich aktualisieren** wird der Bereich gegebenenfalls einem neuen Datenstand aus der Datenbank heraus angepasst.

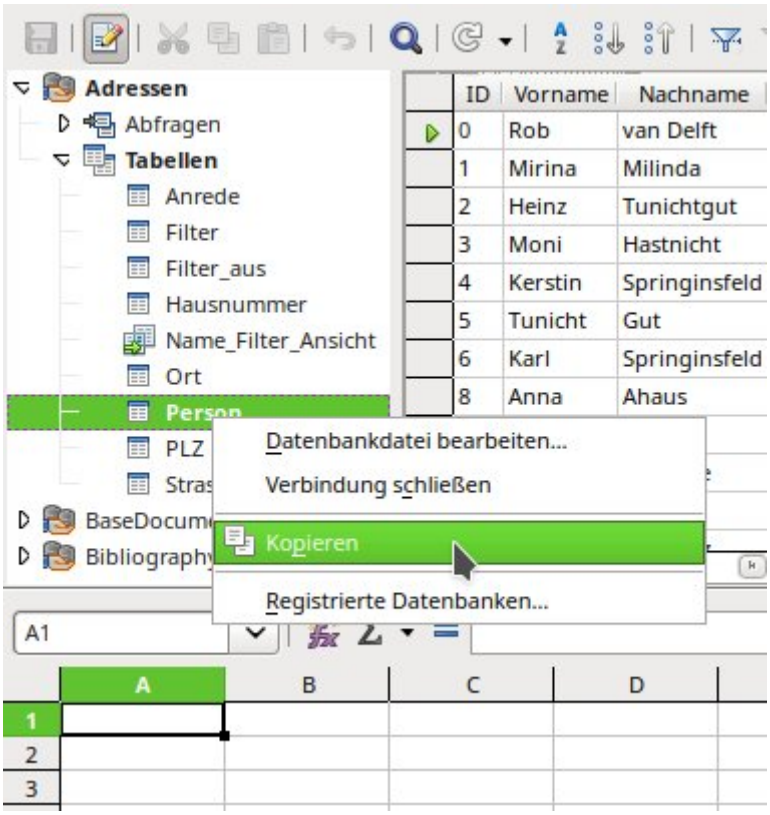
Werden einzelne Datenzeilen oder die komplette Tabelle in der Datenansicht markiert, so kann der Bereich auch über einen Klick auf Daten in Text eingefügt und auch aktualisiert werden.

1 «Serienbrief.zip» ist den Beispieldatenbanken für dieses Handbuch beigefügt.

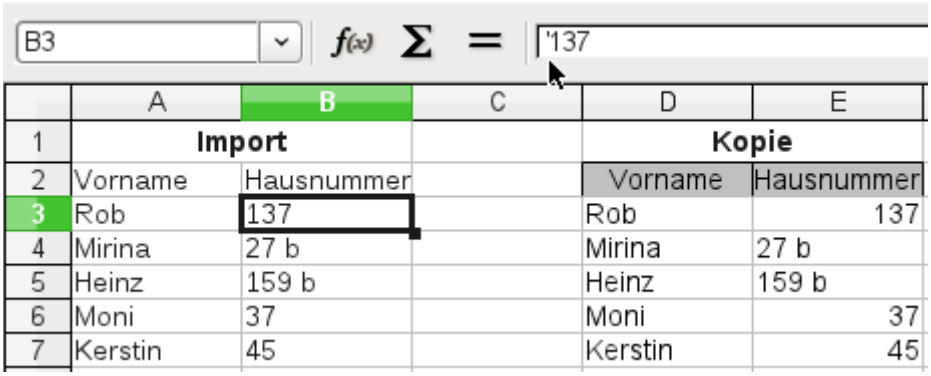
Hierbei werden dann nur die markierten Zeilen der Tabelle neu geschrieben. Alle anderen Zeilen werden aus der Tabelle in Calc entfernt.



Die hereingezogenen Daten sind nicht weiter formatiert als es den Eigenschaften der Datenbankfelder entspricht. Der Bereich ist angegeben, wird aber beim Neueinlesen gegebenenfalls um zusätzlich hinzugekommene Datensätze erweitert.



Über das Kontextmenü der Tabelle kann eine Kopie der Daten erfolgen. Hier wird allerdings kein Importbereich erzeugt, sondern eine Kopie. Die Eigenschaften der Datenfelder werden nicht ausgelesen, sondern von Calc analysiert. Außerdem werden die Feldbezeichnungen als Tabellenköpfe vorformatiert.

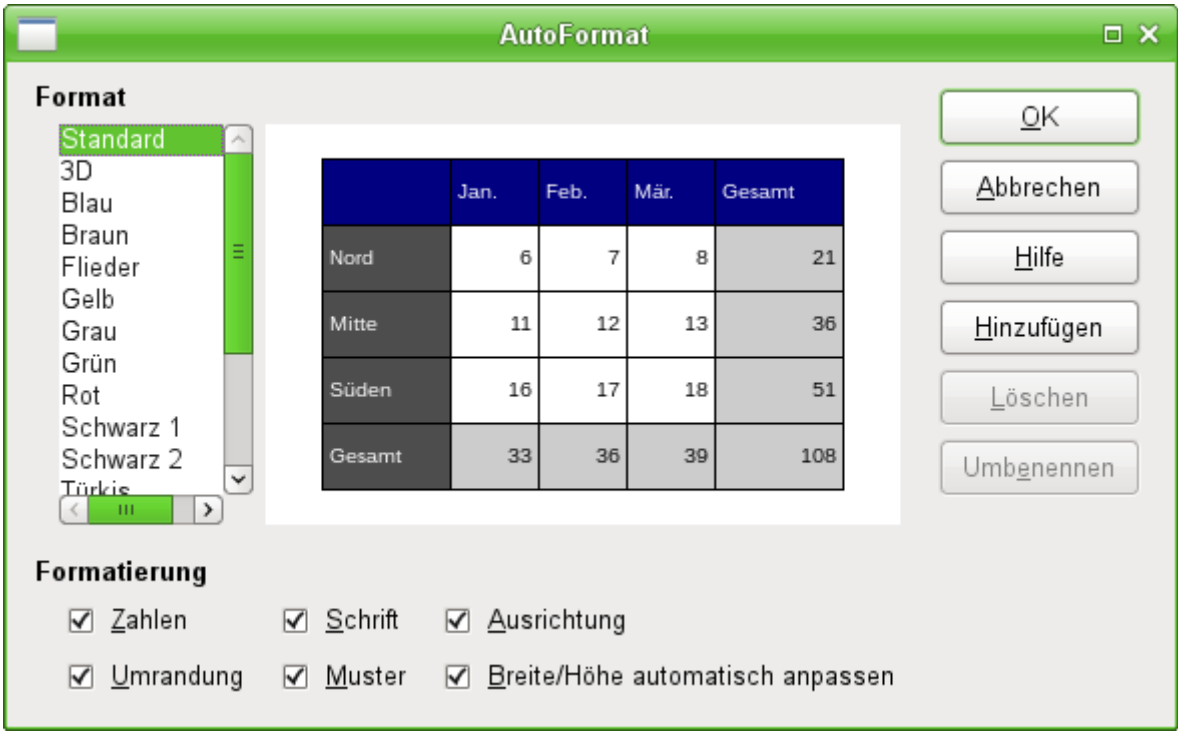


Die Unterschiede zeigen sich besonders bei Feldern, die in der Datenbank als Text formatiert sind. Beim Import macht Calc daraus Textfelder und setzt Zahlen, die es sonst auch als Zahlen interpretieren würde, ein Hochkomma **('137)** voran. Mit diesen Zahlen kann also nicht direkt weiter gerechnet werden. Dies wäre aber auch in Base nicht möglich, da es sich ja tatsächlich um Text handelt.

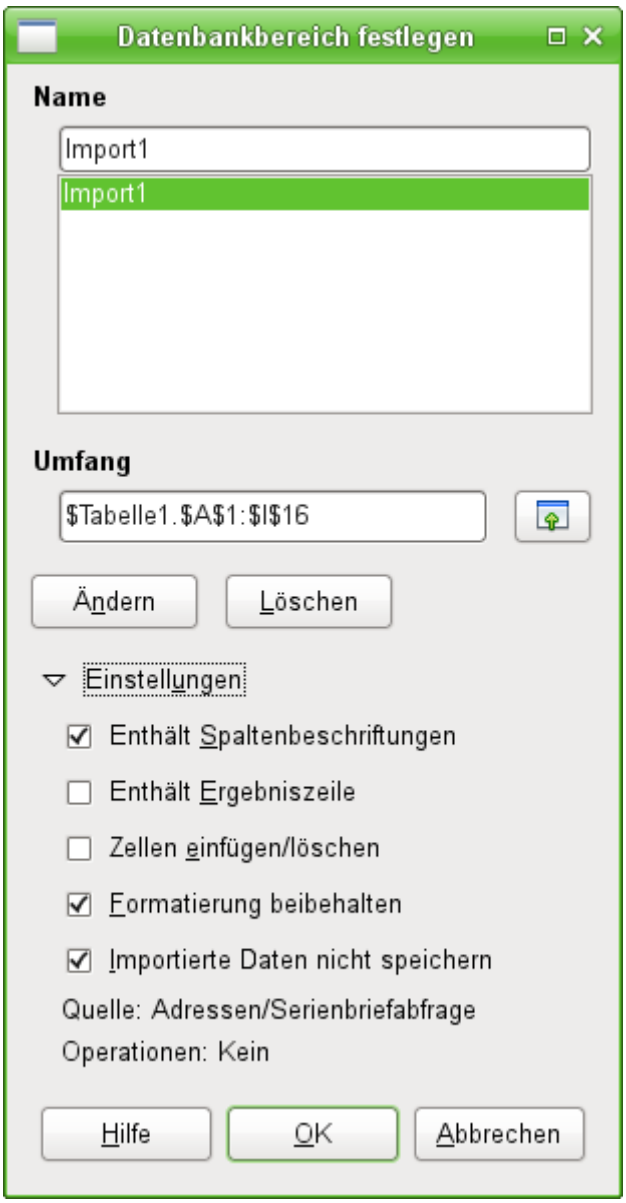
Bei einem eventuellen Export wird das Hochkomma allerdings wieder entfernt, so dass die Daten letztlich in der gleichen Art bestehen bleiben.

Hinweis	**Nur der Import zeigt Daten, die tatsächlich in der Datenbank abgespeichert sind.** Wird in einem Feld der Datenbank eine Dezimalzahl über die Sprachauswahl mit dem Dezimaltrenner «.» formatiert, so wird beim *Import* weiter die Dezimalzahl erkannt. Bei der *Kopie* hingegen wird aus der Dezimalzahl Text. Wird eine Dezimalzahl in Base mit 2 Nachkommastellen angezeigt, obwohl das Feld in der Datenbank tatsächlich 3 Nachkommastellen enthält, so werden bei der *Kopie* nur die sichtbaren 2 Nachkommastellen kopiert. Wird aus einer Base-Datei direkt per *Drag- and Drop* eine Tabelle in Calc gezogen, so ist dies ein *Import*. Kopieren über die *Zwischenablage* erzeugt eine *Kopie*.

Importierte Daten können natürlich im Nachhinein formatiert werden. Hierzu wird der importierte Bereich komplett markiert und dann über **Format → AutoFormat Vorlagen** weiter formatiert:



Diese Formatierung bleibt erhalten, wenn unter **Daten → Bereich festlegen** die folgenden Angaben gemacht werden:



Die **Einstellungen** müssen aufgeklappt werden. Dann sollte **Formatierung beibehalten** ausgewählt werden. Zusätzlich ist noch sinnvoll, **Importierte Daten nicht speichern** zu wählen, wenn eine automatische Abfrage der Daten bei jedem Programmstart erfolgen soll. Dabei werden im übrigen auch Daten aufgenommen, die neu hinzugefügt werden. In sofern ist der Eintrag in **Umfang** nur der Startbereich. Es erfolgt automatisch eine Erweiterung des Bereiches nach unten.

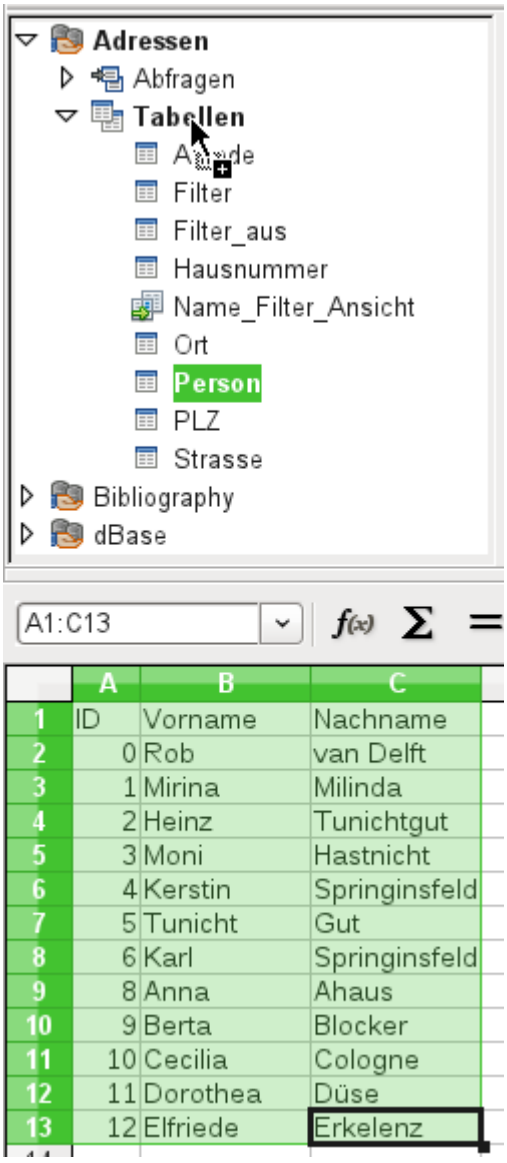
Tipp	Der Import von Daten nach Calc überschreibt den vorherigen Inhalt – und auch die eventuell bereits erledigten Formatierungen, die anschließend direkt für einzelne Felder erfolgt sind. Werden beständig in die gleiche Tabelle Daten exportiert, so empfiehlt es sich, ein separates Tabellenblatt für den Datenimport zu nutzen. Die Daten werden dann über den Bezug **`Tabellenname.Feldbezeichnung`** in einem anderen Tabellenblatt ausgelesen. Die Felder in diesem Tabellenblatt können beliebig vorformatiert werden, ohne dass das Format überschrieben wird.

Tipp	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.

Vorsicht 	Kopien über die Zwischenablage werden in Calc so übernommen, wie sie in der Datenquelle sichtbar sind. Tabellenköpfe werden als Feldbezeichner hervorgehoben, Zahlen mit der Anzahl an Nachkommastellen übernommen, die in Base **angezeigt** werden. Kopien über Drag-and-Drop transportieren den tatsächlichen Inhalt der Datenquelle, also auch z.B. Nachkommastellen von Zahlen, die zur Zeit in der Datenquelle nicht zu sehen, wohl aber vorhanden sind. (*Bug 112023*)

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.



	A	B	C
1	ID	Vorname	Nachname
2	0	Rob	van Delft
3	1	Mirina	Milinda
4	2	Heinz	Tunichtgut
5	3	Moni	Hastnicht
6	4	Kerstin	Springinsfeld
7	5	Tunicht	Gut
8	6	Karl	Springinsfeld
9	8	Anna	Ahaus
10	9	Berta	Blocker
11	10	Cecilia	Cologne
12	11	Dorothea	Düse
13	12	Elfriede	Erkelenz

Der Cursor verändert sein Symbol und deutet an, dass etwas hinzugefügt werden kann.

Es öffnet sich das erste Fenster des Importassistenten. Die weiteren Schritte mit dem Assistenten sind im Kapitel «Tabellen» im Abschnitt «Import von Daten aus anderen Datenquellen» beschrieben.

Hinweis	Der Transport der Daten aus einer Calc-Tabelle in eine Datenbank läuft nicht in der gleichen Weise ab wie der Transport von Daten aus einer Datenbank zu einer anderen Datenbank. Der oben beschriebene Weg führt z.B. dazu, dass Zeilenumbrüche in Feldern der Tabelle nicht in die Tabelle der Datenbank übernommen werden (*Bug 117436*) . Wird stattdessen die Calc-Datei als Datenquelle für eine Datenbank genommen, so kann von dieser Datenbank aus zu einer internen HSQLDB-Datenbank der Tabelleninhalt mit Zeilenumbruch kopiert werden.

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.

Hinweis	Beim Kopieren von Daten einer Datenbank in eine andere müssen die Felder der aufnehmenden Datenbank groß genug sein, um die Daten der abgebenden Datenbank aufzunehmen. Wurden z. B. Bilder in einer HSQLDB gespeichert (Datentyp **`LONGVARBINARY`**, 2^{31} – 1 Byte), so passen diese Bilder nicht unbedingt in ein Feld des Type **`BLOB`** von MariaDB/MySQL (2^{16} – 1 Byte). Zu große Bilder in der HSQLDB führen dann zu einem Abbruch des Imports in MariaDB/MySQL.

Wird gewünscht, 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 möglichst 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

Sind Daten in Tabellenform vorhanden, so können diese über die Zwischenablage und den Assistenten in Base eingefügt werden.

Wird in Base mit einem rechten Mausklick auf die Zieltabelle der Einfügevorgang begonnen, so erscheinen in dem **Kontextmenü** der Maus unter **Kopieren** die Befehle **Einfügen** und **Inhalte einfügen …**. Wird hier **Einfügen** gewählt, so ist beim Importassistenten die entsprechende Tabelle sowie das Anhängen von Daten bereits vorgewählt. **Inhalte einfügen …** schaltet hier lediglich eine Abfrage für einen Importfilter vor. Hier steht 'HTML' und 'RTF' zur Verfügung.

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 Beispiel[2] soll so eine Importmöglichkeit aufzeigen.

Erstellen eines PDF-Formulars

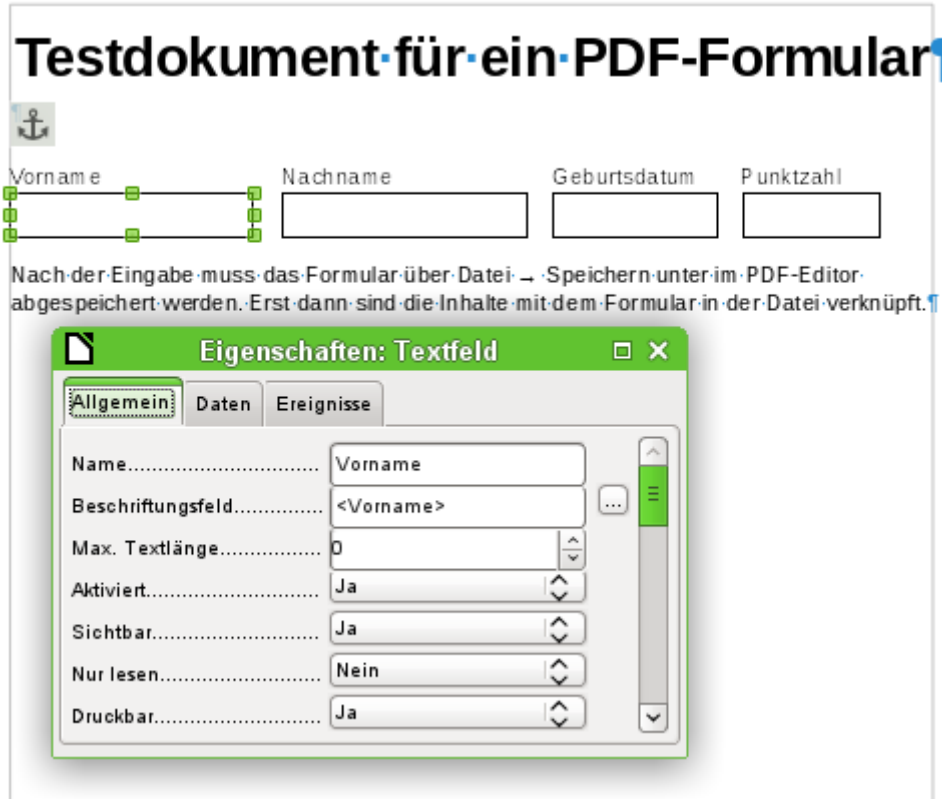
Ein PDF-Formular wird als externes Formular ohne Datenbankanbindung erstellt. Über **Formular → Entwurfsmodus** können die notwendigen Elemente für das Formular eingefügt werden.

Leider werden im PDF-Formular keine Unterschiede zwischen Zahlenfeldern, Datumsfeldern und Textfeldern gemacht. Für das beigefügte Beispiel reicht es also völlig aus, bei jedem Eingabefeld ein Textfeld aufzuziehen. Weitere Feldformatierungen des Writer-Formulars gehen bei dem Export in ein PDF-Formular zwangsläufig verloren.

Grundsätzlich ermöglichen PDF-Formulare nur die folgenden Formularfelder:

- Schaltfläche
- Textfeld
- Markierfeld
- Kombinationsfeld und
- Listenfeld

2 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.

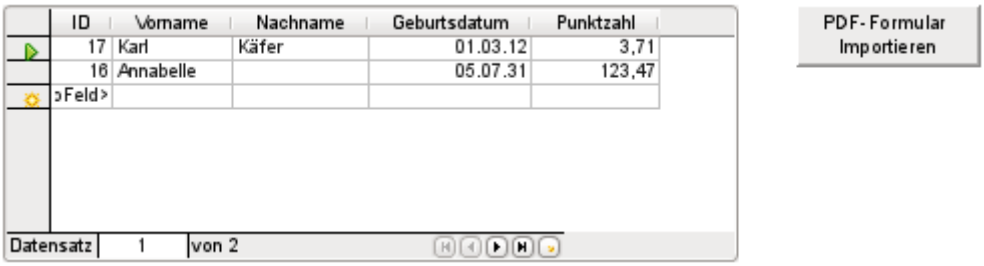
Hilfetexte werden zwar beim Auslesen der Daten mit angezeigt, erscheinen allerdings wohl nicht in jedem *pdf-Viewer.

Hinweis	Der Acrobat Reader hat mit der voreingestellten Standard-Schriftart der Felder Probleme. Hier sollte eine weit verbreitete Schriftart wie Arial gewählt werden. Ist die in den Eingabefeldern definierte Schriftart nicht auf dem System vorhanden, so weigert sich der Acrobat Reader, eine Eingabe in den Feldern zu ermöglichen. Der Fehler liegt auch darin begründet, dass LibreOffice die Schriftarten, die in den Formularfeldern benutzt werden, nicht mit exportiert.

Damit das Formular die Daten anschließend auch enthält, ist nach der Dateneingabe über **Datei → Speichern unter** im PDF-Viewer eine Speicherung der Datei vorzunehmen. Dieser Befehl kann von Viewer zu Viewer unterschiedlich sein. Ohne diese Maßnahme zeigt der Viewer die Daten nach dem Öffnen des Formular auf dem eigenen Rechner zwar an, liest sie aber wohl aus den temporären Dateien des Viewers, nicht aus der *.pdf-Datei direkt. Das Formular bleibt dann beim Transport von einem Rechner auf einen anderen leer.

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.

Zum Auslesen der Daten wird hier auf das OpenSource-Programm «pdftk» zurückgegriffen. Das Programm ist auf jeden Fall frei für Linux und Windows erhältlich. Linux-Distributionen haben meist bereits ein Paket in den Repositories. Windows-User finden das Programm hier: *https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/* .

Unter Linux geht das Auslesen der Daten mit dem folgenden Befehl:

```
004 for i in *.pdf ; do pdftk "$i" dump_data_fields_utf8 >> auswertung.txt ; done
```

Der Befehl liest aus der im gleichen Verzeichnis liegenden *.pdf-Datei die Daten in eine Textdatei mit der Bezeichnung «auswertung.txt».

Die mit «pdftk» ausgelesenen Daten werden in eine Textdatei geschrieben, die wie folgt aussieht:

```
---
FieldType: Text
FieldName: Vorname
FieldFlags: 0
FieldValue: Karl
FieldJustification: Left
---
FieldType: Text
FieldName: Nachname
FieldFlags: 0
FieldValue: Käfer
FieldJustification: Left
---
FieldType: Text
FieldName: Geburtsdatum
FieldNameAlt: Datum mit mindestens zweistelliger Jahreszahl
FieldFlags: 0
FieldValue: 1.3.12
FieldJustification: Left
---
FieldType: Text
FieldName: Punktzahl
FieldNameAlt: Dezimalzahl, 2 Nachkommastellen
FieldFlags: 0
FieldValue: 3,71
FieldJustification: Left

```

Für jedes Feld sind in der Textdatei 5 Zeilen enthalten. Für die Auswertung im Makro sind die Zeilen «FieldName» (Feldname 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.

```
001 SUB PDF_Form_Import(oEvent AS OBJECT)
002     DIM inNumber AS INTEGER
003     DIM stRow AS STRING
```

```
004     DIM i AS INTEGER
005     DIM k AS INTEGER
006     DIM oDatasource AS OBJECT
007     DIM oConnection AS OBJECT
008     DIM oSQL_Command AS OBJECT
009     DIM oResult AS OBJECT
010     DIM stSql AS STRING
011     DIM oDB AS OBJECT
012     DIM oFileAccess AS OBJECT
013     DIM inFields AS INTEGER
014     DIM stFieldName AS STRING
015     DIM stFieldValue AS STRING
016     DIM stFieldType AS STRING
017     DIM stDir AS STRING
018     DIM stDir2 AS STRING
019     DIM stPDFForm AS STRING
020     DIM stFile AS STRING
021     DIM stTable AS STRING
022     DIM inNull AS INTEGER
023     DIM aFiles()
024     DIM aNull()
025     DIM stCommand AS STRING
026     DIM stParameter AS STRING
027     DIM oShell AS OBJECT
```

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.

```
028     inFields = 3
029     stTable = "Name"
030     oDatasource = ThisComponent.Parent.CurrentController
031     If NOT (oDatasource.isConnected()) THEN
032        oDatasource.connect()
033     END IF
034     oConnection = oDatasource.ActiveConnection()
035     oSQL_Command = oConnection.createStatement()
```

Die Datenbankverbindung wurde ermittelt. Anschließend wird der Pfad zu der bestehenden Datenbankdatei im Dateisystem ausgelesen. Mit Hilfe dieses Pfades wird in dem Array **aFiles** der Inhalt des Verzeichnisses ermittelt. Für jede der Dateien des Verzeichnisses wird in einer Schleife nachgesehen, ob die Dateiendung «.pdf» lautet. Groß- und Kleinschreibung sind hier egal, da das Ergebnis der Suche über **LCase** direkt in Kleinschreibung umgesetzt wird.

```
036     oDB = ThisComponent.Parent
037     stDir = Left(oDB.Location,Len(oDB.Location)-Len(oDB.Title))
038     oFileAccess = createUnoService("com.sun.star.ucb.SimpleFileAccess")
039     aFiles = oFileAccess.getFolderContents(stDir, False)
040     FOR k = 0 TO uBound(aFiles())
041        IF LCase(Right(aFiles(k),4)) = ".pdf" THEN
042           stDir2 = ConvertFromUrl(stDir)
043           stPDFForm = ConvertFromUrl(aFiles(k))
```

Um das Kommando für das Auslesen betriebssystemspezifisch adressieren zu können, muss der ursprünglich mit **file://** beginnende Pfadname dem jeweiligen System angepasst werden. Das Startkommando für das Programm «pdftk» ist abhängig vom Betriebssystem entweder mit dem Zusatz «.exe», vielleicht sogar mit dem gesamten Pfad zum Programm wie z.B. «C:\Program Files (x86)\pdftk\pdftk.exe», oder eben ohne Zusatz zu versehen. Über **GetGuiType** wird der Systemtyp ermittelt, wobei '1' für Windows, '3' für MAC und '4' für Unix/Linux steht. Im folgenden wird nur zwischen Windows und allen anderen Systemen unterschieden.

Im Anschluss daran gibt der 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.

```
044         IF GetGuiType = 1 THEN '()
045            stCommand = "pdftk.exe"
046         ELSE
047            stCommand = "pdftk"
048         END IF
049         stParameter = stPDFForm & " dump_data_fields_utf8 output "
050            & stDir2 & "PDF_Form_Data.txt"
051         Shell(stCommand,0,stParameter,True)
052         stFile = stDir & "PDF_Form_Data.txt"
053         i = -1
054         inNumber = FreeFile
```

Mit der Funktion **FreeFile** wird ermittelt, welchen freien Datenkanal das Betriebssystem als nächstes zur Verfügung stellen wird. Dieser Datenkanal wird als eine Integerziffer ausgelesen und direkt zur Öffnung der gerade erstellten PDF-Datendatei genutzt. Die Datei wird über INPUT ausgelesen. Dies ist aus Sicht von LibreOffice gesehen. Daten von außerhalb werden in das Programm LibreOffice eingelesen.

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

Zeile 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 (HSQLDB, SQL-Code für Firebird siehe Anhang). Datumswerte und Dezimalzahlen müssen anders weiter gegeben werden als Texte.

```
058               IF instr(stRow, "FieldName: ") THEN
059                  IF stFieldName = "" THEN
060                     stFieldName = """"+mid(stRow,12)+""""
061                  ELSE
062                     stFieldName = stFieldName & ","""" + mid(stRow,12)+""""
063                  END IF
064                  stSql = "SELECT TYPE_NAME FROM INFORMATION_SCHEMA.SYSTEM_COLUMNS
                        WHERE TABLE_NAME = '" + stTable + "' AND
                        COLUMN_NAME = '" + mid(stRow,12) + "'"
065                  oResult = oSQL_Command.executeQuery(stSql)
066                  WHILE oResult.next
067                     stFieldType = oResult.getString(1)
068                  WEND
069               END IF
```

Hinweis

Der SQL-Code für Firebird ist hier leider wesentlich komplizierter, da Firebird in den Systemtabellen die Feldtypen nicht benennt, sondern nur numerisch aufführt:

```
064 stSql = "SELECT TRIM(CASE ""b"".RDB$FIELD_TYPE||'|'||
       COALESCE(""b"".RDB$FIELD_SUB_TYPE,0) "+ _
       "WHEN '7|0' THEN 'SMALLINT' "+ _
       "WHEN '8|0' THEN 'INTEGER' "+ _
       "WHEN '8|1' THEN 'NUMERIC' "+ _
       "WHEN '8|2' THEN 'DECIMAL' "+ _
       "WHEN '10|0' THEN 'FLOAT' "+ _
       "WHEN '12|0' THEN 'DATE' "+ _
       "WHEN '13|0' THEN 'TIME' "+ _
       "WHEN '14|0' THEN 'CHAR' "+ _
       "WHEN '16|0' THEN 'BIGINT' "+ _
       "WHEN '35|0' THEN 'TIMESTAMP' "+ _
       "WHEN '37|0' THEN 'VARCHAR' "+ _
       "WHEN '261|0' THEN 'BLOB' "+ _
       "WHEN '261|1' THEN 'BLOB Text' "+ _
       "WHEN '261|2' THEN 'BLOB BLR' "+ _
       "WHEN '261|3' THEN 'BLOB ACL' "+ _
       "END) AS ""SQL_Datentyp"" "+ _
       "FROM RDB$RELATION_FIELDS AS ""a"", RDB$FIELDS AS ""b"" "+ _
       "WHERE ""a"".RDB$FIELD_SOURCE = ""b"".RDB$FIELD_NAME "+ _
       "AND ""a"".RDB$RELATION_NAME = '" + stTable + "' "+ _
       "AND ""a"".RDB$FIELD_NAME = '" + mid(stRow,12) + "'"
```

Wie bei den Feldnamen wird auch bei den Werten verfahren. Sie dürfen allerdings nicht in doppelten Anführungszeichen weitergegeben werden, sondern müssen für den SQL-Code entsprechend vorbereitet werden. So muss Schrift in einfache Anführungszeichen gesetzt werden, Datumsangaben SQL-konform umgewandelt werden usw. Dies geschieht durch die extra ausgelagerte Funktion **SQL_Value**.

```
070             IF instr(stRow, "FieldValue: ") THEN
071                IF stFieldValue = "" THEN
072                   stFieldValue = SQL_Value(mid(stRow,13), stFieldType)
073                ELSE
074                   stFieldValue = stFieldValue & "," &
075                      SQL_Value(mid(stRow,13), stFieldType)
076                END IF
077             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 **inFields** 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.

```
078             IF instr(stRow, "FieldJustification:") THEN
079                i = i + 1
080             END IF
081             IF i = inFields THEN
082                aNull = Split(stFieldValue,",")
083                FOR n = 0 TO Ubound(aNull())
084                   IF aNull(n) = "NULL" THEN inNull = inNull + 1
085                NEXT
086                IF inNull < inFields THEN
087                   stSql ="INSERT INTO """ + stTable + """(" + stFieldName + ")"
088                   stSql = stSql + "VALUES (" + stFieldValue + ")"
089                   oSQL_Command.executeUpdate(stSql)
090                END IF
091                stFieldName = ""
```

```
092                     stFieldValue = ""
093                     stFieldType = ""
094                     i = -1
095                     inNull = 0
096                  END IF
097               LOOP
098            CLOSE inNumber
```

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.

```
099            Kill(stFile)
100         END IF
101      NEXT
102      oEvent.Source.Model.Parent.reload()
103   END SUB
```

Enthält ein Text ein einfaches Anführungszeichen «'», so wird es 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`**.

```
001   FUNCTION String_to_SQL(st AS STRING) AS STRING
002      IF InStr(st,"'") THEN
003         st = Join(Split(st,"'"),"''")
004      END IF
005      String_to_SQL = st
006   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 häufig 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 international üblichen erweiterten ISO-Format Format geschrieben werden: YYYY-MM-DD. Es muss also gegebenenfalls 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.

```
007   FUNCTION Date_to_SQLDate(st AS STRING) AS STRING
008      DIM stDay AS STRING
009      DIM stMonth AS STRING
010      DIM stDate AS STRING
011      DIM inYear AS INTEGER
012      stDay = Right("0" & Day(CDate(st)), 2)
013      stMonth = Right("0" & Month(CDate(st)), 2)
014      inYear = Year(CDate(st))
015      IF inYear = 0 THEN
016         inYear = Year(Now())
017      END IF
018      IF inYear > 1000 THEN
019      ELSEIF inYear > 30 THEN
020         inYear = 1900 + inYear
021      ELSE
```

```
022         inYear = 2000 + inYear
023      END IF
024      stDate = inYear & "-" & stMonth & "-" & stDay
025      Date_to_SQLDate = stDate
026   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.

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

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.

```
031      ELSEIF stType = "DATE" THEN
032         IF isDate(st) THEN
033            SQL_Value = "'" & Date_to_SQLDate(st) & "'"
034         ELSE
035            SQL_Value = "NULL"
036         END IF
```

Handelt es sich bei dem Feldtyp um ein Dezimalfeld, so kann es Nachkommastellen geben. Der Dezimaltrenner in Basic und auch in SQL ist allerdings ein Punkt. Entsprechend müssen Zahlen umgewandelt werden, sofern sie ein Komma enthalten. Das Feld kann nur Zahlen aufnehmen, so dass andere Zeichen wie z.B. Maßeinheiten entfernt werden müssen. Dies geschieht mit der Funktion **Val()**. Leider gibt diese Funktion aber den Wert mit dem Dezimaltrenner der eingestellten Sprache zurück, so dass hier anschließend noch das Komma erneut durch einen Punkt ersetzt werden muss.

```
037      ELSEIF stType = "DECIMAL" THEN
038         stValue = Str(Val(Join(Split(st,","),".")))
039         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.

```
040      ELSE
041         SQL_Value = "'" & String_to_SQL(st) & "'"
042      END IF
043   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.