LibreOffice
The Document Foundation

Base

Anhang

Copyright

Mitwirkende/Autoren

Jochen Schiffers Robert Großkopf Jost Lange

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 13.05.2012. Basierend auf der LibreOffice Version 3.5.

Anmerkung für Macintosh Nutzer

Einige Tastenbelegungen (Tastenkürzel) und Menüeinträge unterscheiden sich zwischen der Macintosh Version und denen für Windows- und Linux-Rechnern. Die unten stehende Tabelle gibt Ihnen einige grundlegende Hinweise dazu. Eine ausführlichere Aufstellung dazu finden Sie in der Hilfedatei des jeweiligen Moduls.

Windows/Linux	***entspricht am Mac***	***Effekt***
Menü-Auswahl **Extras** → **Optionen**	LibreOffice → Einstellungen	Zugriff auf die Programmoptionen
Rechts-Klick	Control+Klick	Öffnen eines Kontextmenüs
Ctrl (Control) oder Strg (Steuerung)	⌘ *(Command)*	Tastenkürzel in Verbindung mit anderen Tasten
F5	Shift+⌘+F5	öffnet den Dokumentnavigator Dialog
F11	⌘+T	öffnet den Formatvorlagen Dialog

Inhalt

Barcode

Um die Barcode-Druckfunktion nutzen zu können, muss der Font "ean13.ttf" installiert sein. Dieser Font ist frei verfügbar.

EAN13-Barcodes können mittels "ean13.ttf" folgendermaßen erstellt werden:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Zahl	Großbuchstaben, A=0 B=1 usw.						*	Kleinbuchstaben, a=0 b=1 usw.						+

Siehe hierzu die Abfrage "Barcode_EAN13_ttf_Bericht" der Beispieldatenbank "Medien_ohne_Makros"

Datentypen des Tabelleneditors

Ganzzahlen

Typ	***Zusatz***	***HSQLDB***	***Umfang***	***Speicherbedarf***
Tiny Integer	TINYINT	TINYINT	2^8 = 256 | - 128 bis + 127	1 Byte
Small Integer	SMALLINT	SMALLINT	2^{16} = 65536 | - 32768 bis + 32767	2 Byte
Integer	INTEGER	INTEGER | INT	2^{32} = 4294967296 | - 2147483648 bis + 2147483647	4 Byte
BigInt	BIGINT	BIGINT	2^{64}	8 Byte

Fließkommazahlen

Typ	***Zusatz***	***HSQLDB***	***Umfang***	***Speicherbedarf***
Dezimal	DECIMAL	DECIMAL	Unbegrenzt, durch GUI auf 50 Stellen, einstellbar, feste Nachkommastellen, exakte Genauigkeit	variabel
Zahl	NUMERIC	NUMERIC	Unbegrenzt, durch GUI auf 50 Stellen, einstellbar, feste Nachkommastellen, exakte Genauigkeit	variabel
Float	FLOAT	(Double wird stattdessen genutzt)		
Real	REAL	REAL		
Double	DOUBLE	DOUBLE [PRECISION] | FLOAT	Einstellbar, nicht exakt, 15 Dezimalstellen maximal	8 Byte

Text

Typ	*Zusatz*	*HSQLDB*	*Umfang*	*Speicherbedarf*
Text	VARCHAR	VARCHAR	einstellbar	variabel
Text	VARCHAR_IGNORECASE	VARCHAR_IGNORECASE	Einstellbar, Auswirkung auf Sortierung	variabel
Text (fix)	CHAR	CHAR | CHARACTER	Einstellbar, Rest zum tatsächlichen Text wird mit Leerzeichen aufgefüllt	fest
Memo	LONGVARCHAR	LONGVARCHAR		variabel

Zeit

Typ	*Zusatz*	*HSQLDB*	*Umfang*	*Speicherbedarf*
Datum	DATE	DATE		4 Byte
Zeit	TIME	TIME		4 Byte
Datum/Zeit	TIMESTAMP	TIMESTAMP | DATETIME	Einstellbar (0, 6 – 6 bedeutet mit Millisekunden)	8 Byte

Sonstige

Typ	*Zusatz*	*HSQLDB*	*Umfang*	*Speicherbedarf*
Ja/Nein	BOOLEAN	BOOLEAN | BIT		
Binärfeld (fix)	BINARY	BINARY	Wie Integer	fest
Binärfeld	VARBINARY	VARBINARY	Wie Integer	variabel
Bild	LONGVARBINARY	LONGVARBINARY	Wie Integer	variabel, für größere Bilder gedacht
OTHER	OTHER	OTHER | OBJECT		

Eingebaute Funktionen und abgespeicherte Prozeduren

Numerisch

Da hier mit Fließkommazahlen gerechnet wird empfiehlt es sich gegebenenfalls auf die Einstellung der Felder bei einer Abfrage zu achten. Meist ist hier die Anzeige der Nachkommazahlen begrenzt, so dass eventuell überraschende Ergebnisse zustande kommen. Dann wird z.B. in Spalte1 0,00, in Spalte2 1000 angezeigt. In Spalte3 soll dann Spalte1 * Spalte2 stehen – dort steht plötzlich 1.

ABS(d)	Gibt des absoluten Wert einer Zahl wieder, entfernt also ggf. das Minus-Vorzeichen.

ACOS(d)	Gibt den Arcuscosinus wieder.
ASIN(d)	Gibt den Arcussinus wieder.
ATAN(d)	Gibt den Arcustangens wieder.
ATAN2(a,b)	Gibt den Arcustangens über Koordinaten wieder. "a" ist der Wert der x-Achse, "b" der Wert der y-Achse
BITAND(a,b)	Sowohl die binäre Schreibweise von "a" als auch die binäre Schreibweise von "b" müssen an der gleichen Stelle eine "1" stehen haben, damit die "1" in das Ergebnis übernommen wird. BITAND(3,5) ergibt 1; 0011 AND 0101 = 0001
BITOR(a,b)	Entweder die binäre Schreibweise von "a" oder die binäre Schreibweise von "b" müssen an der gleichen Stelle eine "1" stehen haben, damit die "1" in das Ergebnis übernommen wird. BITAND(3,5) ergibt 7; 0011 OR 0101 = 0111
CEILING(d)	Gibt die kleinste Ganzzahl an, die nicht kleiner als d ist.
COS(d)	Gibt den Cosinus wieder.
COT(d)	Gibt den Cotangens wieder.
DEGREES(d)	Gibt zu Bogenmaßen die Gradzahl wieder.
EXP(d)	Gibt e^d (e: (2.718...)) wieder.
FLOOR(d)	Gibt die größte Ganzzahl an, die nicht größer als d ist.
LOG(d)	Gibt den natürlichen Logarithmus zur Basis 'e' wieder.
LOG10(d)	Gibt den Logarithmus zur Basis 10 wieder.
MOD(a,b)	Gibt den Rest als Ganzzahl wieder, der bei der Division von 2 Ganzzahlen entsteht. MOD(11,3) ergibt 2, weil 3*3+2=11
PI()	Gibt π (3.1415...) wieder.
POWER(a,b)	a^b , POWER(2,3) = 8, weil $2^3 = 8$
RADIANS(d)	Gibt zu den Gradzahlen das Bogenmaß wieder.
RAND()	Gibt eine Zufallszahl x größer oder gleich 0.0 und kleiner als 1.0 wieder.
ROUND(a,b)	Rundet a auf b Stellen nach dem Dezimalzeichen.
ROUNDMAGIC(d)	Löst Rundungsprobleme, die durch Fließkommazahlen entstehen. 3.11-3.1-0.01 ist eventuell nicht genau 0, wird aber als 0 in der GUI angezeigt. ROUNDMAGIC macht daraus einen tatsächlichen 0-Wert.
SIGN(d)	Gibt -1 wieder, wenn "d" kleiner als 0 ist, 0 wenn d==0 und 1 wenn "d" größer als 0 ist.
SIN(A)	Gibt den Sinus eines Bogenmaßes wieder.
SQRT(d)	Gibt die Quadratwurzel wieder.
TAN(A)	Gibt den Tangens eines Bogenmaßes wieder.
TRUNCATE(a,b)	Schneidet "a" auf "b" Zeichen nach dem Dezimalpunkt ab. TRUNCATE(2.37456,2) = 2.37

Text	
ASCII(s)	Gibt den ASCII-Code des ersten Buchstaben des Strings wieder.
BIT_LENGTH(str)	Gibt die Länge des Textes str in Bits wieder.
CHAR(c)	Gibt den Buchstaben wieder, der zu dem ASCII-Code c gehört.
CHAR_LENGTH(str)	Gibt die Länge des Textes in Buchstaben wieder.
CONCAT(str1,str2)	Verbindet str1 + str2
'str1'||'str2'||'str3' oder 'str1'+'str2'+'str3'	Verbindet str1 + str2 + str3, einfachere Alternative zu CONCAT
DIFFERENCE(s1,s2)	Gibt den ?Klang?unterschied zwischen s1 und s2 wieder. Hier wird lediglich eine Ganzzahl ausgegeben. 0 bedeutet dabei gleichen Klang. So erscheint 'for' und 'four' mit 0 gleich, Kürzen und Würzen wird auf 1 gesetzt, Mund und Mond wieder auf 0
HEXTORAW(s1)	Übersetzt Haxadezimalcode in andere Zeichen
INSERT(s,start,len,s2)	Gibt einen Text wieder, bei dem Teile ersetzt werden. Beginnend mit start wird über eine Länge len aus dem Text s Text ausgeschnitten und durch den Text s2 ersetzt. INSERT('Bundesbahn', 3, 4, 'mmel') macht aus 'Bundesbahn' 'Bummelbahn', wobei die Länge des eingefügten Textes auch ohne weiteres größer als die des ausgeschnittenen Textes sein darf. So ergibt INSERT('Bundesbahn', 3, 5, 's und B') 'Bus und Bahn'.
LCASE(s)	Wandelt den String in Kleinbuchstaben um.
LEFT(s,count)	Gibt die mit count angegebene Zeichenanzahl vom Beginn des Textes s wieder.
LENGTH(s)	Gibt die Länge eines Textes in Anzahl der Buchstaben wieder.
LOCATE(search,s,[start])	Gibt den ersten Treffer für den Begriff aus search in dem Text s wieder. Der Treffer wird numerisch angegeben: (1=left, 0=not found) Die Angabe eines Startes innerhalb des Textes ist optional.
LTRIM(s)	Entfernt führende Leerzeichen und nicht druckbare Zeichen von einem Text.
OCTET_LENGTH(str)	Gibt die Länge eines Textes in Bytes an. Dies entspricht im Prinzip dem doppelten Wert der Zeichenanzahl.
RAWTOHEX(s1)	Verwandelt in die Hexadezimalschreibweise, Umkehr von HEXTORAW()
REPEAT(s,count)	Wiederholt den Text s count Mal
REPLACE(s,replace,s2)	Ersetzt alle vorkommenden Textstücke mit dem Inhalt replace im Text s durch den Text s2
RIGHT(s,count)	Umgekehrt zu LEFT; gibt die mit count angegebene Zeichenzahl vom Textende aus wieder.
RTRIM(s)	Entfernt alle Leerzeichen und nicht druckbaren Zeichen am Textende.
SOUNDEX(s)	Gibt einen Code von 4 Zeichen wieder, die dem Klang von s entsprechen sollen – passt zu der Funktion DIFFERENCE()
SPACE(count)	Gibt die in count angegebene Zahl an Leertasten wieder.

SUBSTR(s,start[,len])	Kürzel für SUBSTRING
SUBSTRING(s,start[,len])	Gibt den Text s ab der Startposition wieder. (1=links) . Wird die Länge ausgelassen, so wird der gesamte Text wiedergegeben.
UCASE(s)	Wandelt den String in Großbuchstaben um.
LOWER(s)	Wie LCASE(s)
UPPER(s)	Wie UCASE(s)
Datum/Zeit	
CURDATE()	Gibt das aktuelle Datum wieder.
CURTIME()	Gibt die aktuelle Zeit wieder.
DATEDIFF(string, datetime1, datetime2)	Datumsunterschied zwischen zwei Datums- bzw. Datumszeitangaben. Der Eintrag in string entscheidet darüber, in welcher Einheit der Unterschied wiedergegeben wird: 'ms'='millisecond', 'ss'='second', 'mi'='minute', 'hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'. Sowohl die Langfassung als auch die Kurzfassung ist für den einzusetzenden string möglich.
DAY(date)	Gibt den Tag im Monat wieder. (1-31)
DAYNAME(date)	Gibt den englischen Namen des Tages wieder.
DAYOFMONTH(date)	Gibt den Tag im Monat wieder. (1-31), Synonym für DAY()
DAYOFWEEK(date)	Gibt den Wochentag als Zahl wieder. (1 bedeutet Sonntag)
DAYOFYEAR(date)	Gibt den Tag im Jahr wieder (1-366).
HOUR(time)	Gibt die Stunde wieder. (0-23)
MINUTE(time)	Gibt die Minute wieder. (0-59)
MONTH(date)	Gibt den Monat wieder. (1-12)
MONTHNAME(date)	Gibt den englischen Namen des Monats wieder.
NOW()	Gibt das aktuelle Datum und die aktuelle Zeit zusammen als Zeitstempel wieder. Stattdessen kann auch CURRENT_TIMESTAMP genutzt werden.
QUARTER(date)	Gibt das Quartal im Jahr wieder. (1-4)
SECOND(time)	Gibt die Sekunden einer Zeitangabe wieder. (0-59)
WEEK(date)	Gibt die Woche des Jahres wieder. (1-53)
YEAR(date)	Gibt das Jahr aus einer Datumseingabe wieder.
CURRENT_DATE	Synonym für CURDATE(), SQL-Standard, *Meldung in Base: Access is denied*
CURRENT_TIME	Synonym für CURTIME(), SQL-Standard
CURRENT_TIMESTAMP	Synonym für NOW(), SQL-Standard

Datenbankverbindung

(Bis auf IDENTITY() in Base ohne Bedeutung, mit der Einstellung **`SQL-Kommando direkt ausführen`** abrufbar)

DATABASE()	Gibt den Namen der Datenbank, die zu dieser Verbindung gehört, wieder.
USER()	Gibt den Benutzernamen dieser Verbindung wieder.
CURRENT_USER	SQL Standardfunktion, Synonym für USER()
IDENTITY()	Gibt den letzten Wert für ein Autowertfeld wieder, das in der aktuellen Verbindung erzeugt wurde. Dies wird bei der Makroprogrammierung genutzt, um aus einem erstellten Primärschlüssel für eine Tabelle einen Fremdschlüssel für eine andere Tabelle zu erstellen.

System

IFNULL(exp,value)	Wenn exp NULL ist wird value zurückgegeben, sonst exp. Stattdessen kann als Erweiterung auch COALESCE() genutzt werden. Exp und value müssen den gleichen Datentyp haben.
CASEWHEN(exp,v1,v2)	Wenn exp wahr ist wird v1 zurückgegeben, sonst v2. Stattdessen kann auch CASE WHEN genutzt werden. CASE WHEN funktioniert in Zusammenhang mit der GUI auch besser.
CONVERT(term,type)	Wandelt term in einen anderen Datentyp um.
CAST(term AS type)	Synonym zu CONVERT()
COALESCE(expr1,expr2, expr3,...)	Wenn expr1 nicht NULL ist wird expr1 wiedergegeben, ansonsten wird expr2 überprüft, danach dann expr3 usw.
NULLIF(v1,v2)	Wenn v1 gleich v2 ist wird NULL wiedergegeben, ansonsten v1.
`CASE v1 WHEN v2 THEN v3 [ELSE v4] END`	Wenn v1 gleich v2 ist wird v3 wiedergegeben. Sonst wird v4 wiedergegeben oder NULL, wenn kein ELSE formuliert ist.
`CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END`	Wenn expr1 wahr ist wird v1 zurückgegeben. [Optional können weitere Fälle angegeben werden] Sonst wird v4 wiedergegeben oder NULL, wenn kein ELSE formuliert ist.
`EXTRACT ({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM <Datums- oder Zeitwert>)`	Kann viele der Datums- und Zeitfunktionen ersetzen. Gibt das Jahr, den Monat, den Tag usw. von einem Datums- bzw. Datumszeitwert wieder.
`POSITION(<string expression> IN <string expression>)`	Wenn der erste Text in dem zweiten enthalten ist wird die Position des ersten Textes wiedergeben, ansonsten 0
`SUBSTRING(<string expression> FROM <numeric expression> [FOR <numeric expression>])`	Liefert den Teil eines Textes ab der in FROM angegebenen Startposition, optional in der in FOR angegebenen Länge.

`TRIM([{LEADING | TRAILING | BOTH}] FROM <string expression>)`	Nicht druckbare Sonderzeichen und Leerzeichen werden entfernt.

Informationstabellen der HSQLDB

Innerhalb von Datenbanken wird in dem Bereich *"INFORMATION_SCHEMA"* die Information über alle Tabelleneigenschaften sowie ihre Verbindung untereinander abgelegt. Diese Informationen ermöglichen in Base bei der Erstellung von Makros, Prozeduren mit weniger Parametern zu starten. Eine Anwendung findet sich in der Beispieldatenbank unter anderem im Modul *"Wartung"* in der Prozedur *"Tabellenbereinigung"* für die Ansteuerung des Dialoges.

In einer Abfrage können die einzelnen Informationen sowie sämtliche dazugehörigen Felder auf die folgende Art ermittelt werden.

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_ALIASES"
```

Im Gegensatz zu einer normalen Tabelle ist es hier notwendig, dem jeweiligen folgenden Begriff *"INFORMATION_SCHEMA"* voranzustellen.

```
SYSTEM_ALIASES
SYSTEM_ALLTYPEINFO
SYSTEM_BESTROWIDENTIFIER
SYSTEM_CACHEINFO
SYSTEM_CATALOGS
SYSTEM_CHECK_COLUMN_USAGE
SYSTEM_CHECK_CONSTRAINTS
SYSTEM_CHECK_ROUTINE_USAGE
SYSTEM_CHECK_TABLE_USAGE
SYSTEM_CLASSPRIVILEGES
SYSTEM_COLUMNPRIVILEGES
SYSTEM_COLUMNS
SYSTEM_CROSSREFERENCE
SYSTEM_INDEXINFO
SYSTEM_PRIMARYKEYS
SYSTEM_PROCEDURECOLUMNS
SYSTEM_PROCEDURES
SYSTEM_PROPERTIES
SYSTEM_SCHEMAS
SYSTEM_SEQUENCES
SYSTEM_SESSIONINFO
SYSTEM_SESSIONS
SYSTEM_SUPERTABLES
SYSTEM_SUPERTYPES
SYSTEM_TABLEPRIVILEGES
SYSTEM_TABLES
SYSTEM_TABLETYPES
SYSTEM_TABLE_CONSTRAINTS
SYSTEM_TEXTTABLES
SYSTEM_TRIGGERCOLUMNS
SYSTEM_TRIGGERS
SYSTEM_TYPEINFO
SYSTEM_UDTATTRIBUTES
SYSTEM_UDTS
```

```
SYSTEM_USAGE_PRIVILEGES
SYSTEM_USERS
SYSTEM_VERSIONCOLUMNS
SYSTEM_VIEWS
SYSTEM_VIEW_COLUMN_USAGE
SYSTEM_VIEW_ROUTINE_USAGE
SYSTEM_VIEW_TABLE_USAGE
```

Datenbankreparatur für *.odb-Dateien

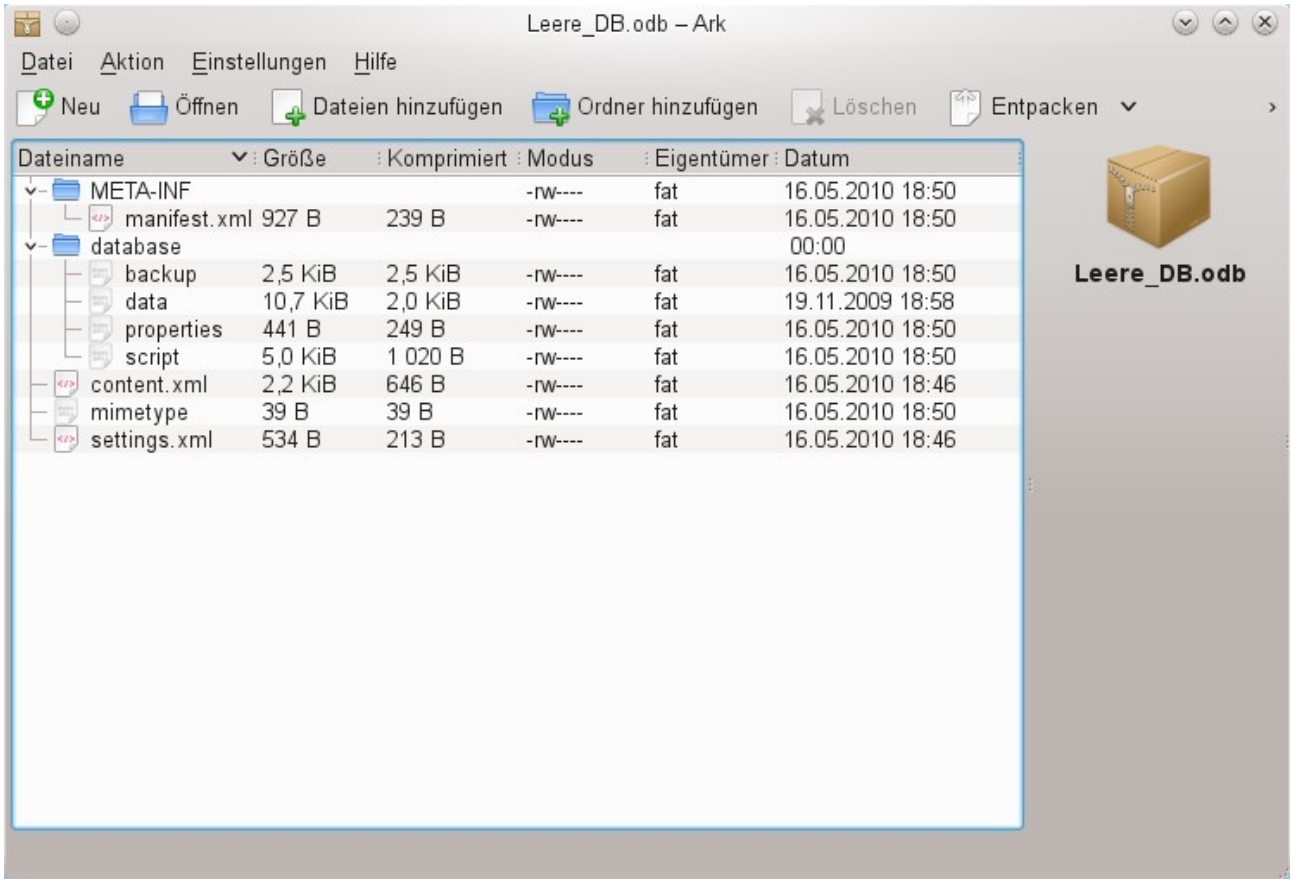
Regelmäßige Datensicherung sollte eigentlich Grundlage für den Umgang mit dem PC sein. Sicherheitskopien sind so der einfachste Weg, auf einen halbwegs aktuellen Datenstand zurückgreifen zu können. Doch in der Praxis mangelt es eben häufig an dieser Stelle.

Bei plötzlichen Abstürzen des PC kann es passieren, dass geöffnete Datenbanken von LO (interne Datenbank HSQLDB) nicht mehr zu öffnen sind. Stattdessen wird beim Versuch, die Datenbank zu öffnen, nach einem entsprechenden Filter für das Format gefragt.

Das Ganze liegt daran, dass Teile der Daten der geöffneten Datenbank im Arbeitsspeicher liegen und lediglich temporär zwischengespeichert werden. Erst beim Schließen der Datei wird die gesamte Datenbank in die Datei zurückgeschrieben und gepackt.

Um eventuell doch noch an die Daten zu kommen, kann das folgende Verfahren hilfreich sein:

1. Fertigen sie eine Kopie ihrer Datenbank für die weiteren Schritte an.
2. Versuchen sie die Kopie mit einem Packprogramm zu öffnen. Es handelt sich bei der *.odb-Datei um ein gepacktes Format, ein Zip-Archiv. Lässt sich die Datei so nicht direkt öffnen, so funktioniert das Ganze vielleicht auch über die Umbenennung der Endung von *.odb zu *.zip.
   Funktioniert das Öffnen nicht, so ist vermutlich von der Datenbank nichts mehr zu retten.
3. Folgende Verzeichnisse sehen sie nach dem Öffnen einer Datenbankdatei im Packprogramm auf jeden Fall:



4. Die Datenbankdatei muss ausgepackt werden. Die entscheidenden Informationen für die Daten liegen im Unterverzeichnis "database" in den Dateien "data" und "script".
5. Gegebenenfalls empfiehlt es sich, die Datei "script" einmal anzuschauen und auf Ungereimtheiten zu überprüfen. Dieser Schritt kann aber auch erst einmal zum Testen übersprungen werden. Die "script"-Datei enthält vor allem die Beschreibung der Tabellenstruktur.
6. Gründen sie eine neue, leere Datenbankdatei und öffnen diese Datenbankdatei mit dem Packprogramm.
7. Ersetzen sie die Dateien "data" und "script" aus der neuen Datenbankdatei durch die unter "4." entpackten Dateien.
8. Das Packprogramm muss nun geschlossen werden. War es, je nach Betriebssystem, notwendig, die Dateien vor dem Öffnen durch das Packprogramm nach *.zip umzubenennen, so ist das jetzt wieder nach *.odb zu wandeln.
9. Öffnen sie die Datenbankdatei jetzt mit LO oder OOo. Sie können hoffentlich wieder auf ihre Tabellen zugreifen.
10. Wie weit sich jetzt auch Abfragen, Formulare und Berichte auf ähnliche Weise wiederherstellen lassen, bleibt dem weiteren Testen überlassen.

Siehe hierzu auch: *http://user.services.LO oder OOo.org/en/forum/viewtopic.php?f=83&t=17125*

Wenn, wie auf den folgenden Seiten beschrieben, die externe HSQLDB verwendet wird, kann eventuell ein weiteres Problem mit den *.odb-Dateien in Verbindung mit manchen LO oder OOo-Versionen auftauchen. Wird eine externe HSQLDB genutzt, so ist der sicherste Weg der über das hsqldb.jar-Archiv, das mit LO oder OOo mitgeliefert wird. Wird ein anderes Archiv verwendet, so kann das dazu führen, dass die internen Datenbanken plötzlich nicht mehr zugänglich sind. Dies liegt daran, dass LO oder OOo Schwierigkeiten hat, zwischen interner und externer HSQLDB zu unterscheiden und Meldungen von einem Versionskonflikt produziert. OOo 3.1.1 scheint hiermit

keine Probleme zu haben, OOo 3.3 und LO 3.3 leider schon. Eine aktuelle Version des Programms bedeutet hier nicht unbedingt eine geringere Zahl an Problemstellen.

Lassen sich interne Datenbanken nicht mehr öffnen so hilft pragmatisch erst einmal nur, OOo 3.1.1 oder LO ab der Version 3.5 zu nutzen. Ansonsten muss als externe Datenbank die mitgelieferte hsqldb.jar-Datei genutzt werden. Außerdem muss aus der *.odb-Datei das database-verzeichnis extrahiert werden. Die Datei properties hat hier einen Eintrag, der in LO 3.3 und OOo 3.3 zu dieser Fehlermeldung führt:

```
version=1.8.1
```

steht in Zeile 11.

Diese Zeile ist zu ändern auf

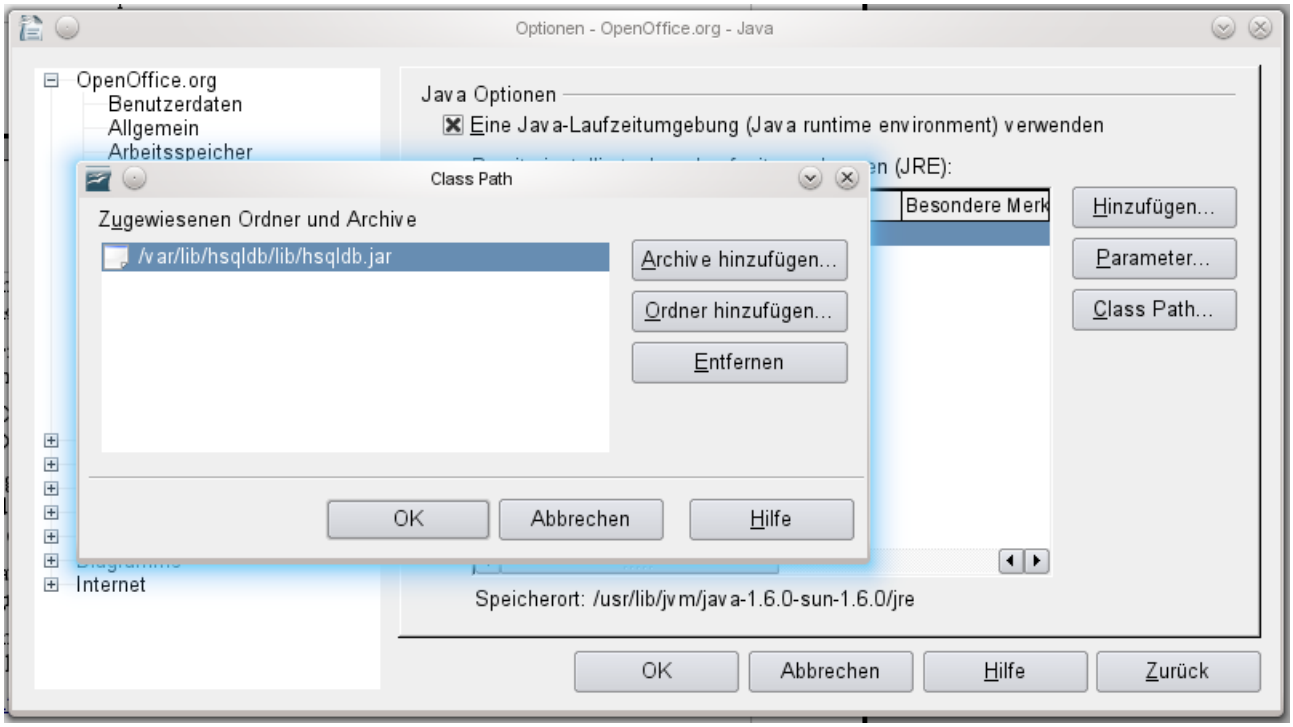
```
version=1.8.0
```

Danach ist das database-Verzeichnis wieder in das *.odb-Päckchen einzulesen und die Datenbank lässt sich auch wieder unter LO 3.3 und OOo 3.3 öffnen.

Datenbankverbindung zu einer externen HSQLDB

Die interne HSQLDB unterscheidet sich erst einmal nicht von der externen Variante. Wenn, wie im Folgenden beschrieben, erst einmal nur der Zugriff auf die Datenbank nach außerhalb gelegt werden soll, dann ist keine Serverfunktion erforderlich. Hier reicht schlicht das Archiv, was in LO oder OOo mitgeliefert wurde. Im LO- oder OOo-Pfad liegt unter /program/classes/hsqldb.jar. Die Verwendung dieses Archivs ist die sicherste Variante, da dann keine Versionsprobleme auftauchen.

Die externe HSQLDB steht unter *http://hsqldb.org/* zum Download frei zur Verfügung. Ist die Datenbank installiert, so sind in LO oder OOo folgende Schritte zu vollziehen:



Der Datenbanktreiber muss, sofern er nicht in dem Pfad der Java-Runtime liegt, als ClassPath unter Extras – Optionen – Java hinzugefügt werden.

Die Verbindung zu der externen Datenbank erfolgt über JDBC. Die Datenbankdateien sollen in einem bestimmten Verzeichnis abgelegt werden. Dieses Verzeichnis kann beliebig gewählt werden. Es liegt in dem folgenden Beispiel im home-Ordner. Nicht angegeben ist hier der weitere Verzeichnisverlauf sowie der Name der Datenbank.

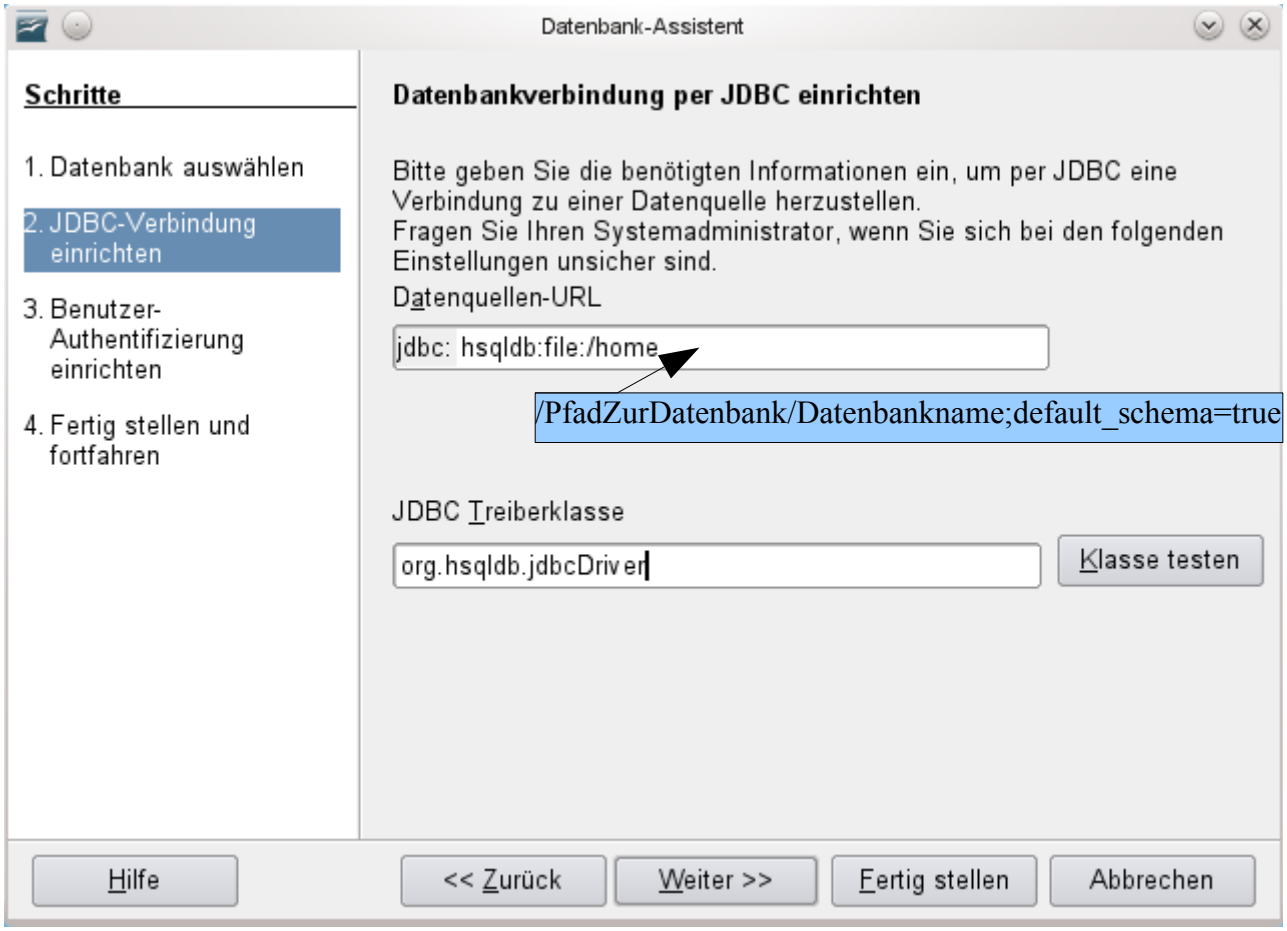
Wichtig, damit auch Daten in die Datenbank über die GUI geschrieben werden können, muss: ergänzend neben dem Datenbanknamen **`";default_schema=true"`** stehen.

Also:

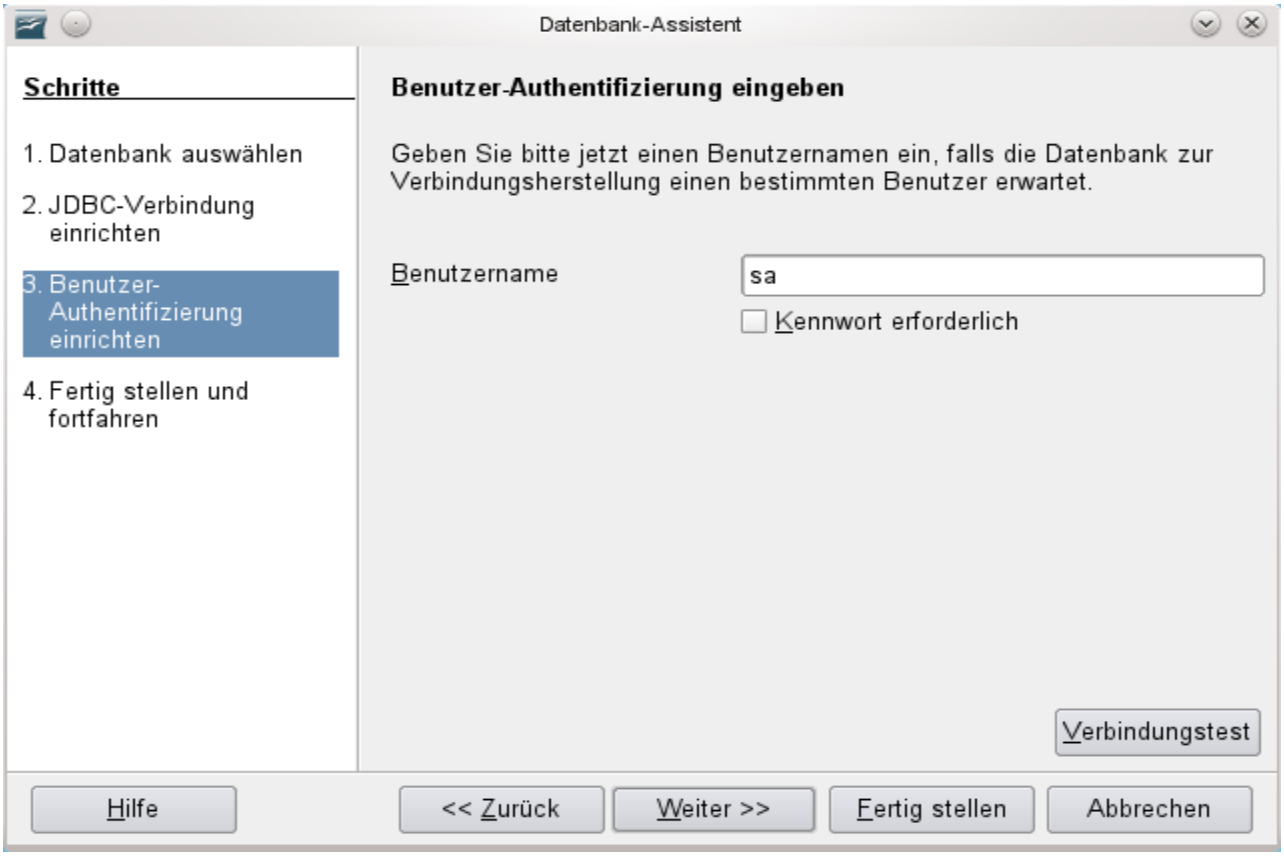
```
jdbc:hsqldb:file:/home/PfadZurDatenbank/Datenbankname;default_schema=true
```

In dem Ordner befinden sich die Dateien

```
Datenbankname.backup
Datenbankname.data
Datenbankname.properties
Datenbankname.script
Datenbankname.log
```



Weiter geht es mit der Angabe des Standardnutzers, sofern nichts an der HSQLDB-Konfiguration geändert wurde:



Damit ist die Verbindung erstellt und es kann auf die Datenbank zugegriffen werden.

Vorsicht

Wird eine externe Datenbank mit einer Version HSQLDB 2.* bearbeitet, so kann sie anschließend nicht mehr in eine interne Datenbank unter LibreOffice oder OpenOffice umgewandelt werden. Dies liegt an den zusätzlichen Funktionen, die in der Version 1.8.* noch nicht vorhanden sind. Dadurch endet der Aufruf mit der Version 1.8.* bereits beim Einlesen der Script-Datei der Datenbank.

Ebensowenig kann eine externe Datenbank, die einmal mit einer Version der 2er-Reihe bearbeitet wurde, anschließend wieder mit der Version 1.8.* bearbeitet werden, die kompatibel zu Libre- und OpenOffice ist.

Änderung der Datenbankverbindung zur externen HSQLDB

Die interne HSQL-Datenbank hat den Nachteil, dass die Abspeicherung der Daten innerhalb eines gepackten Archivs erfolgt. Erst mit dem Packen werden alle Daten festgeschrieben. Dies kann leichter zu Datenverlust führen als es bei der Arbeit mit einer externen Datenbank der Fall ist. Im folgenden werden die Schritte gezeigt, die notwendig sind, um den Umstieg einer bestehenden Datenbank vom *.odb-Päckchen zur externen Version in HSQL zu erreichen.

Aus einer Kopie der bestehenden Datenbank wird das Verzeichnis "database" extrahiert. Der Inhalt wird in das oben beschriebene frei wählbare Verzeichnis kopiert. Dabei sind die enthaltenen Dateien um den Datenbanknamen zu ergänzen:

```
Datenbankname.backup
Datenbankname.data
Datenbankname.properties
Datenbankname.script
Datenbankname.log
```

Jetzt muss noch die "content.xml" aus dem *.odb-Päckchen extrahiert werden. Hier sind mit einem einfachen Texteditor die folgenden Zeilen zu suchen:

```
<db:connection-data><db:connection-resource
xlink:href="sdbc:embedded:hsqldb"/><db:login db:is-password-
required="false"/></db:connection-data><db:driver-settings/>
```

Diese Zeilen sind mit der Verbindung zur externen Datenbank zu ersetzen, hier der Verbindung zu einer Datenbank mit dem Namen "verein", die jetzt im Verzeichnis "hsqldb_data" liegt.

```
<db:connection-data><db:connection-resource
xlink:href="jdbc:hsqldb:file:/home/robby/Dokumente/Datenbanken/hsqldb_
data/verein;default_schema=true"/><db:login db:user-name="sa" db:is-
password-required="false"/></db:connection-data><db:driver-settings
db:java-driver-class="org.hsqldb.jdbcDriver"/>
```

Falls, wie oben geschrieben, die Grundkonfiguration der HSQLDB nicht angetastet wurde stimmt auch der Nutzername und die nicht erforderliche Passworteinstellung.

Nach Änderung des Codes muss die content.xml wieder in das *.odb-Päckchen eingepackt werden. Das Verzeichnis "database" ist in dem Päckchen jetzt überflüssig. Die Daten werden in Zukunft durch die externe Datenbank zur Verfügung gestellt.

Änderung der Datenbankverbindung für einen Mehrbenutzerbetrieb

Für die Mehrbenutzerfunktion muss die HSQLDB über einen Server zur Verfügung gestellt werden. Wie die Installation des Servers erfolgt ist je nach Betriebssystem unterschiedlich. Für OpenSuSE war nur ein entsprechendes Paket herunter zu laden und der Server zentral über YAST zu starten (Runlevel-Einstellungen). Nutzer anderer Betriebssysteme und Linux-Varianten finden sicher geeignete Hinweise im Netz.

Im Heimatverzeichnis des Servers, unter SuSE /var/lib/hsqldb, befinden sich unter anderem ein Verzeichnis "data", in dem die Datenbank abzulegen ist, und eine Datei "server.properties", die den Zugang zu den (eventuell also auch mehreren) Datenbanken in diesem Verzeichnis regelt.

Die folgenden Zeilen geben den kompletten Inhalt dieser Datei auf meinem Rechner wieder.
Es wird darin der Zugang zu 2 Datenbanken geregelt, nämlich der ursprünglichen Standard-Datenbank (die als neue Datenbank genutzt werden kann) als auch der Datenbank, die aus der *.odb-Datei extrahiert wurde.

```
# Hsqldb Server cfg file.
# See the Advanced Topics chapter of the Hsqldb User Guide.

server.database.0   file:data/db0
server.dbname.0     firstdb
server.urlid.0      db0-url

server.database.1   file:data/verein
server.dbname.1     verein
server.urlid.1      verein-url

server.silent       true
server.trace        false

server.port         9001
server.no_system_exit         true
```

Die Datenbank 0 wird mit dem Namen "firstdb" angesprochen, obwohl die einzelnen Dateien in dem Verzeichnis data mit "db0" beginnen. Meine eigene Datenbank habe ich als "Datenbank 1" hinzugefügt. Hier sind Datenbankname und Dateibeginn identisch.

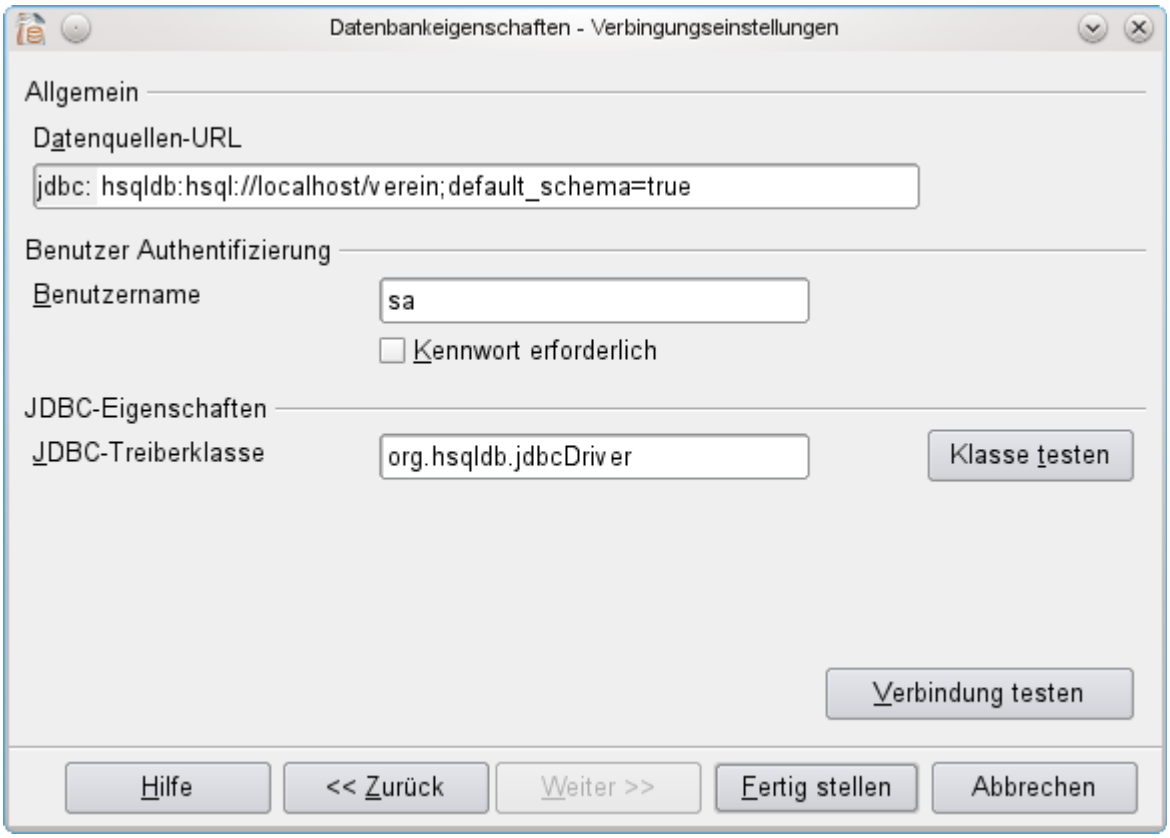
Die beiden Datenbanken werden mit folgenden Zugängen angesprochen:

```
jdbc:hsqldb:hsql://localhost/firstdb;default_schema=true
username sa
password
jdbc:hsqldb:hsql://localhost/verein;default_schema=true
username sa
password
```

Die URL wurde hier bereits jeweils um den für den Schreibzugang über die grafische Benutzeroberfläche von LO oder OOo erforderlichen Zusatz **`";default_schema=true"`** ergänzt.

Wenn tatsächlich im Serverbetrieb gearbeitet werden soll ist natürlich aus Sicherheitsgründen zu überlegen, ob die Datenbank nicht mit einem Passwort geschützt werden soll.

Nun erfolgt die Serververbindung über LO oder OOo:



Mit diesen Zugangsdaten wird auf den Server des eigenen Rechners zugegriffen. Im Netzwerk mit anderen Rechnern müsste dann entweder über Rechnernamen oder die IP-Adresse auf den Server, der ja auf meinem Rechner läuft, zugegriffen werden.

Beispiel: Mein Rechner hat die IP 192.168.0.20 und ist im Netz bekannt mit dem Namen lin_serv. Jetzt ist an anderen Rechnern für die Verbindung zur Datenbank einzugeben:

```
jdbc:hsqldb:hsql://192.168.0.20/verein;default_schema=true
```

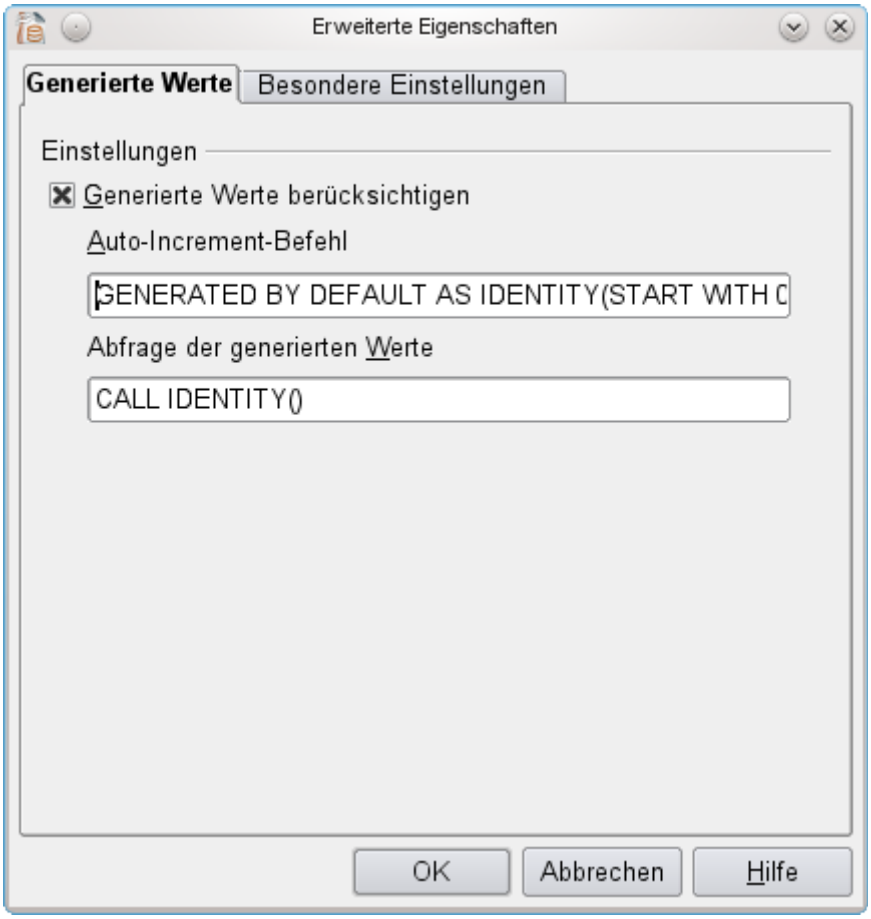
bzw.:

```
jdbc:hsqldb:hsql://lin_serv/verein;default_schema=true
```

Die Datenbank ist nun angebunden und kann beschrieben werden. Schnell taucht allerdings ein zusätzliches Problem auf. Die vorher automatisch generierten Werte werden plötzlich nicht mehr hochgeschrieben. Hier fehlt es noch an einer zusätzlichen Einstellung.

Autoinkrementwerte mit der externen HSQLDB

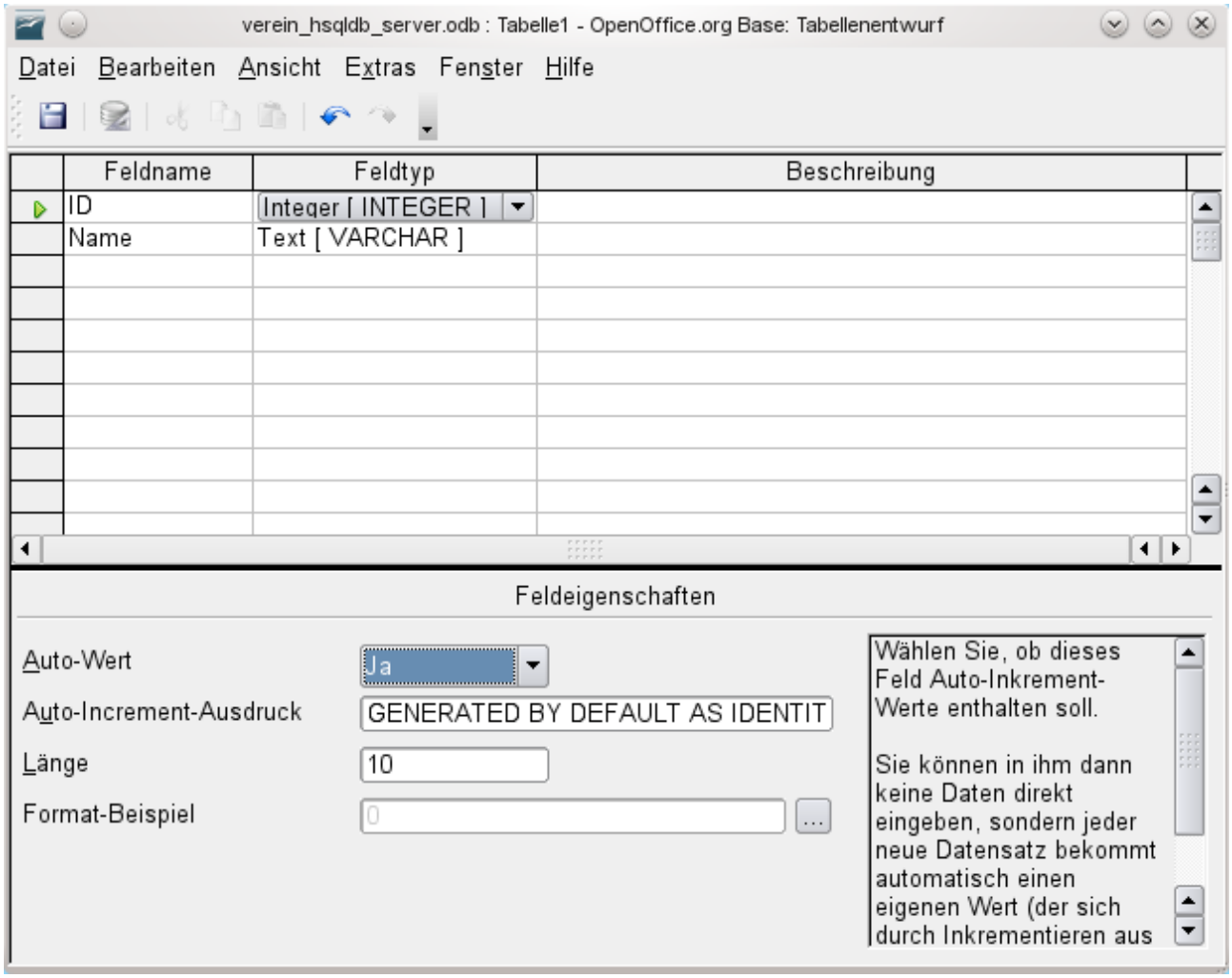
Für die Nutzung der Auto-Werte müssen je nach Version von LO oder OOo bei der Tabellenerstellung verschiedene Wege beschritten werden. Allen gleich ist erst einmal der folgende Eintrag unter **Bearbeiten** → **Datenbank** → **Erweiterte Einstellungen** erforderlich:



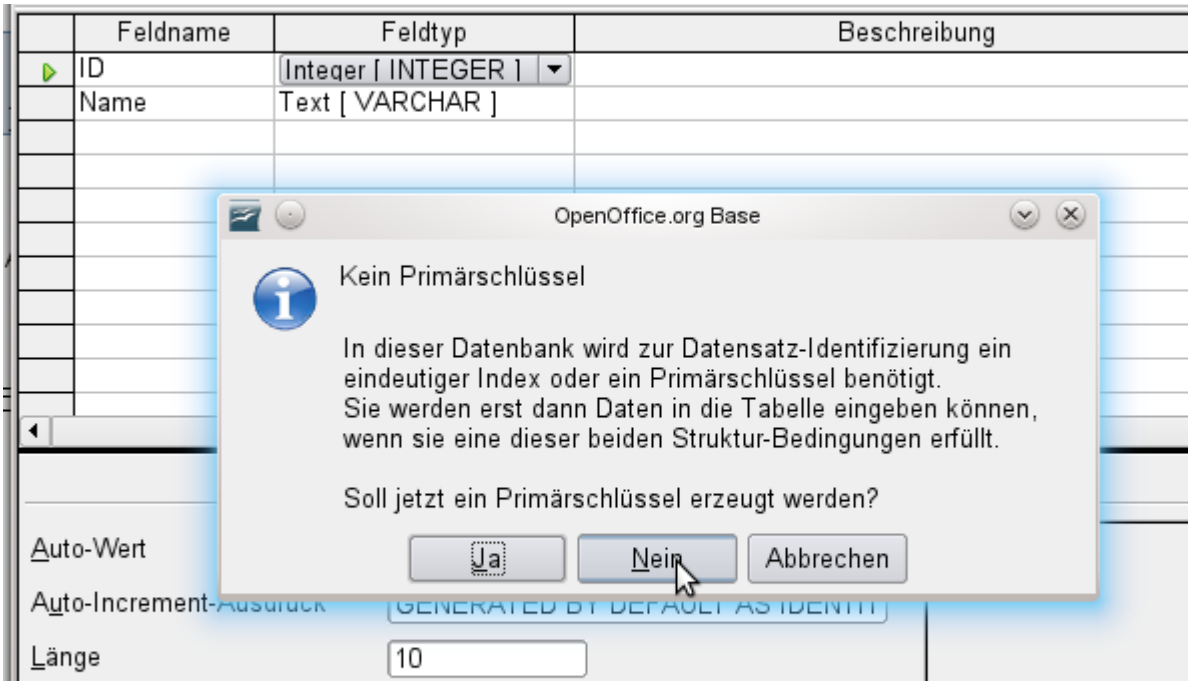
Mit dem Zusatz **`GENERATED BY DEFAULT AS IDENTITY(START WITH 0)`** soll die Funktion des automatisch hochzählenden Wertes für den Primärschlüssel erstellt werden. Die GUI von LO 3.3 und LO 3.4und OOo übernimmt zwar diesen Befehl, schreibt davor aber leider die Anweisung **`NOT NULL`**, so dass die Reihenfolge der Befehlsfolge für die HSQLDB nicht lesbar ist. Hierbei ist zu berücksichtigen, dass die HSQLDB mit dem obigen Befehl ja bereits mitgeteilt bekommt, dass das entsprechende Feld den Primärschlüssel enthält.

Hinweis	In LO 3.3 und LO 3.4 sowie in OOo 3.3 deshalb die Eingabe des Autowertes in der GUI nicht möglich. Nutzer dieser Versionen erstellen zuerst eine Tabelle mit einem Primärschlüsselfeld ohne Autowert und geben dann direkt über **Extras** → **SQL** ein: ALTER TABLE "Tabellenname" ALTER COLUMN "ID" INT GENERATED BY DEFAULT AS `IDENTITY(START WITH 0)` … wobei davon ausgegangen wird, dass das Primärschlüsselfeld den Namen "ID" hat.

Nutzer von OOo in der Version 3.1.1 können auch den folgenden Weg beschreiten:



Das Feld mit der Bezeichnung "ID" hat den Typ "Integer" und bekommt zusätzlich den Auto-Wert zugeschrieben. In der Übersicht erscheint jetzt der einzufügende Ausdruck dazu. Da dieser Ausdruck bereits die Definition des Primärschlüssels enthält wird nicht noch einmal in der GUI die Eigenschaft "Primärschlüssel" zugewiesen.



Auch auf diese Frage muss unbedingt mit "Nein" geantwortet werden. Der Primärschlüssel wird dann trotzdem richtig geschrieben und die Tabelle hat ihren Auto-Wert.

Mit dem Auslesen des letzten Wertes und dem Hochlesen zum nächsten Wert hingegen klappt es in allen Versionen von LO und OOo über den Befehl **`CALL IDENTITY()`**. Dies trifft dann z.B. auf die Lösung zu, die Datenbank zuerst einmal als "*.odb-Päckchen" zu erstellen, gründlich zu testen und danach dann die Datenbanktabellen einfach auszulagern.

Sämtliche Abfragen, Formulare und Berichte lassen sich so weiter nutzen, da die Datenbank für die "*.odb-Datei" weiter auf die gleiche Weise angesprochen wird und eventuell spezifische SQL-Befehle mit der externen HSQLDB weiter gelesen werden können.