Base Handbuch

Anhang

LibreOffice 7.6

Dokumentationen zu LibreOffice unter de.libreoffice.org

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 01.08.2023. Basierend auf der Version LibreOffice 7.6.

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 ~~Firebird~~	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} = 18446744073709551615 | - 9223372036854775808 bis + 9223372036854775807	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)	Einstellbar, nicht exakt, 7 Dezimalstellen maximal	4 Byte
Real	REAL	REAL Firebird: BLOB		
Double	DOUBLE	DOUBLE [PRECISION] | FLOAT	Einstellbar, nicht exakt, 15 Dezimalstellen maximal	8 Byte

Text

Typ	Zusatz	HSQLDB	Umfang	Speicherbedarf
Text	VARCHAR	VARCHAR	Einstellbar FIREBIRD: 32000, in der internen Version 8000 Zeichen	variabel
Text	VARCHAR_IGNORECASE	VARCHAR_IGNORECASE ~~FIREBIRD~~	Einstellbar, Auswirkung auf Sortierung, ignoriert Unterschiede zwischen Groß- und Kleinschreibung	variabel
Text (fix)	CHAR	CHAR | CHARACTER	Einstellbar, Rest zum tatsächlichen Text wird mit Leerzeichen aufgefüllt	fest
Memo	LONGVARCHAR	LONGVARCHAR FIREBIRD: CLOB	unbegrenzt	variabel

Zeit

Typ	Zusatz	HSQLDB	Umfang	Speicherbedarf
Datum	DATE	DATE		4 Byte
Zeit	TIME	TIME	FIREBIRD: 1/10000 Sekunden maximal: 23:59:59	4 Byte
Datum/Zeit	TIMESTAMP	TIMESTAMP | DATETIME	HSQLDB: Einstellbar (0, 6 – 6 bedeutet 1/1000Sekunden)	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 FIREBIRD: BLOB[1]	LONGVARBINARY FIREBIRD: BLOB	LONGVARBINARY FIREBIRD: BLOB	Wie Integer	variabel, für größere Bilder gedacht
OTHER	OTHER	OTHER | OBJECT		

In den Tabellendefinitionen und bei der Änderung von Datentypen in Abfragen mit den Funktionen *CONVERT* oder *CAST* werden bei einigen Datentypen Informationen zur Anzahl an Zeichen (a), zur Genauigkeit (g, entspricht der Gesamtzahl an Ziffern) und zu den Dezimalstellen (d) erwartet: CHAR(a), VARCHAR(a), DOUBLE(g), NUMERIC(g,d), DECIMAL(g,d) und TIMESTAMP(g).

1 In Firebird funktioniert nur der Feldtyp BLOB [BLOB] korrekt mit dem grafischen Kontrollfeld von Formularen.

HSQLDB: TIMESTAMP(g) kann nur die Werte '0' oder '6' annehmen. Die Genauigkeit des Timestamps kann *nur direkt über den SQL-Befehl* eingegeben werden. Sollen also Zeitangaben im Sportbereich eingetragen werden, so ist TIMESTAMP(6) über **Extras → SQL** voreinzustellen.

Bei der Migration von HSQLDB nach FIREBIRD wird aus dem Feld für ein Bild ein Datentyp **`Bild [BLOB]`** erstellt. Dieser Datentyp arbeitet leider nicht korrekt mit dem Grafischen Kontrollfeld zusammen, so dass die Bilder nicht mehr angezeigt und auch nicht vergrößert dargestellt werden können. Abhilfe: Der Name der importierten Tabelle wird geändert und die gesamte Tabelle wird kopiert. Beim Einfügen wird die Tabelle mit altem Namen neu erstellt. Jetzt wird beim Datentyp für das Bild der Datentyp **`Blob [BLOB]`** gewählt. Dieser Datentyp zeigt Bilder (und ggf. auch andere Dokumente) wieder korrekt an.

Datentypen in StarBasic

Zahlen

Typ	***Entspricht in HSQLDB***	***Startwert***	***Anmerkung***	***Speicherbedarf***
Integer	SMALLINT	0	2^{16} = - 32768 bis + 32767	2 Byte
Long	INTEGER	0	2^{32} = - 2147483648 bis + 2147483647	4 Byte
Single		0.0	Dezimaltrenner: «.»	4 Byte
Double	DOUBLE	0.0	Dezimatrenner: «.»	8 Byte
Currency	Ähnlich DECIMAL, NUMERIC	0.0000	Währung, 4 Dezimalstellen fest	8 Byte

Sonstige

Typ	***Entspricht in HSQLDB***	***Startwert***	***Anmerkung***	***Speicherbedarf***
Boolean	BOOLEAN	False	1 = „ja“, alles andere: „nein“	1 Byte
Date	TIMESTAMP	00:00:00	Datum und Zeit	8 Byte
String	VARCHAR	Leerer String	bis 65536 Zeichen	variabel
Object	OTHER	Null		variabel
Variant		Leer	Kann jeden (anderen) Datentyp annehmen	variabel

Vor allem bei Zahlenwerten besteht große Verwechselungsgefahr. In der Datenbank steht z.B. häufig im Primärschlüssel der Datentyp «INTEGER». Wird jetzt per Makro ausgelesen, so muss dort die aufnehmende Variable den Typ «Long» haben, da diese vom Umfang her mit «INTEGER» aus der Datenbank übereinstimmt. Der entsprechende Auslesebefehl heißt dann auch «getLong».

Eingebaute Funktionen und abgespeicherte Prozeduren

In der eingebauten HSQLDB bzw. in Firebird sind die folgenden Funktionen verfügbar. Ein paar Funktionen können leider nur dann genutzt werden, wenn in der Abfrage «SQL-Kommando direkt ausführen» gewählt wurde. Dies verhindert dann gleichzeitig, dass die Abfragen editierbar bleiben.

Funktionen, die mit der grafischen Benutzeroberfläche zusammenarbeiten, sind gekennzeichnet mit GUI. Funktionen, die nur über «SQL-Kommando direkt ausführen» ansprechbar sind, sind gekennzeichnet mit ~~GUI~~.

Bei den Funktionen gibt es solche, die mit der HSQLDB laufen, nicht aber mit der eingebauten Version von Firebird - und umgekehrt. Ist der Datenbankname grün dargestellt, so ist die Funktion verfügbar und gibt ordnungsgemäße Werte aus. Ist der Datenbankname rot und durchgestrichen dargestellt, so ist die Funktion nicht verfügbar oder aber, was deutlich bedenklicher ist, verfügbar, nur eben mit falschen Ergebnissen. Das ist dann mit dem Zusatz ~~(Bug)~~ gekennzeichnet.

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) HSQLDB FIREBIRD	Gibt des absoluten Wert einer Zahl wieder, entfernt also ggf. das Minus-Vorzeichen. GUI
ACOS(d) HSQLDB FIREBIRD	Gibt den Arcuscosinus wieder. Bereich: [-1, 1]. GUI
ASIN(d) HSQLDB FIREBIRD	Gibt den Arcussinus wieder. Bereich: [-1, 1]. GUI
ATAN(d) HSQLDB FIREBIRD	Gibt den Arcustangens wieder. GUI
ATAN2(a,b) HSQLDB FIREBIRD	Gibt den Arcustangens über Koordinaten wieder. 'a' ist der Wert der x-Achse, 'b' der Wert der y-Achse GUI
BITAND(a,b) HSQLDB BIN_AND(a,b) FIREBIRD	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 GUI
BITOR(a,b) HSQLDB BIN_OR(a,b) FIREBIRD	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. **BITOR(3,5)** ergibt 7; 0011 OR 0101 = 0111 GUI
BIN_SHL(a,b) ~~HSQLDB~~ FIREBIRD	a*2b
BIN_SHR(a,b) ~~HSQLDB~~ FIREBIRD	a/2b

BITXOR(a,b) HSQLDB BIN_XOR(a,b) FIREBIRD	Entweder die binäre Schreibweise von 'a' oder die binäre Schreibweise von 'b', nicht aber beide, müssen an der gleichen Stelle eine '1' stehen haben, damit die '1' in das Ergebnis übernommen wird. **BITXOR(3,5)** ergibt 6; 0011 XOR 0101 = 0110 GUI
CEIL(d) ~~HSQLDB~~ FIREBIRD CEILING(d) HSQLDB FIREBIRD	Gibt die kleinste Ganzzahl an, die nicht kleiner als d ist. GUI
COS(d) HSQLDB FIREBIRD	Gibt den Cosinus wieder. GUI
COSH(d) ~~HSQLDB~~ FIREBIRD	Gibt den Cosinus-Hyperbolicus wieder. GUI
COT(d) HSQLDB FIREBIRD	Gibt den Cotangens wieder. GUI
DEGREES(d) HSQLDB ~~FIREBIRD~~	Gibt zu Bogenmaßen die Gradzahl wieder. GUI
EXP(d) HSQLDB FIREBIRD	Gibt ed (e: (2.718...)) wieder. GUI
FLOOR(d) HSQLDB FIREBIRD	Gibt die größte Ganzzahl an, die nicht größer als d ist. GUI
LOG(d) HSQLDB ~~FIREBIRD~~ LN(d) ~~HSQLDB~~ FIREBIRD	Gibt den natürlichen Logarithmus zur Basis 'e' wieder. GUI
LOG(x,y) ~~HSQLDB~~ FIREBIRD	Gibt den Logarithmus zur Basis x wieder. **LOG(2,8)** ergibt 3, weil $2^3 = 8$ GUI
LOG10(d) HSQLDB FIREBIRD	Gibt den Logarithmus zur Basis 10 wieder. GUI
MOD(a,b) HSQLDB FIREBIRD	Gibt den Rest als Ganzzahl wieder, der bei der Division von 2 Ganzzahlen entsteht. **MOD(11,3)** ergibt 2, weil 3*3+2=11 GUI
PI() HSQLDB FIREBIRD	Gibt π (3.1415...) wieder. GUI
POWER(a,b) HSQLDB FIREBIRD	ab , **POWER(2,3)** = 8, weil $2^3 = 8$ GUI
RADIANS(d) HSQLDB ~~FIREBIRD~~	Gibt zu den Gradzahlen das Bogenmaß wieder. GUI
RAND() HSQLDB FIREBIRD	Gibt eine Zufallszahl x größer oder gleich 0.0 und kleiner als 1.0 wieder. GUI
ROUND(a,b) HSQLDB FIREBIRD	Rundet a auf b Stellen nach dem Dezimalzeichen. GUI
ROUNDMAGIC(d) HSQLDB ~~FIREBIRD~~	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. GUI

SIGN(d) HSQLDB FIREBIRD	Gibt -1 wieder, wenn 'd' kleiner als 0 ist, 0 wenn 'd' = 0 und 1 wenn 'd' größer als 0 ist. GUI
SIN(A) HSQLDB FIREBIRD	Gibt den Sinus eines Bogenmaßes wieder. GUI
SINH(d) ~~HSQLDB~~ FIREBIRD	Gibt den Sinus-Hyperbolicus wieder. GUI
SQRT(d) HSQLDB FIREBIRD	Gibt die Quadratwurzel wieder. GUI
TAN(A) HSQLDB FIREBIRD	Gibt den Tangens eines Bogenmaßes wieder. GUI
TANH(d) ~~HSQLDB~~ FIREBIRD	Gibt den Tangens-Hyperbolicus wieder. GUI
TRUNCATE(a,b) HSQLDB ~~FIREBIRD~~ TRUNC(a,b) ~~HSQLDB~~ FIREBIRD	Schneidet 'a' auf 'b' Zeichen nach dem Dezimalpunkt ab. TRUNCATE(2.37456,2) = 2.37 GUI
Text	
ASCII(s) HSQLDB ASCII_VAL(s) FIREBIRD	Gibt den ASCII-Code des ersten Buchstaben des Strings wieder. GUI
BIT_LENGTH(s) HSQLDB FIREBIRD	Gibt die Länge des Textes s in Bits wieder. HSQLDB: Jedes Zeichen wird als 16bit ausgegeben. Firebird: Normale Zeichen 8bit, Sonderzeichen 16bit. GUI
CHAR(c) HSQLDB ASCII_CHAR(c) FIREBIRD	Gibt den Buchstaben wieder, der zu dem ASCII-Code c gehört. Dabei geht es nicht nur um Buchstaben, sondern auch um Steuerzeichen. **`ASCII_CHAR()`** ist auf die Zahlen 0 – 255 begrenzt. **`CHAR(13)`** erzeugt in einer Abfrage einen Zeilenumbruch, der in mehrzeiligen Feldern eines Formulars oder in Berichten sichtbar wird. GUI
CHAR_LENGTH(s) CHARACTER_LENGTH() HSQLDB FIREBIRD	Gibt die Länge des Textes in Buchstaben wieder. GUI
CHAR_TO_UUID (ascii_uuid) ~~HSQLDB FIREBIRD (BUG)~~	Ein String mit 36 Zeichen und einem '-' an Position 9, 14, 19 und 24 und ansonsten gültigen Hexadezimalzeichen wird in einen String ohne '-' und in Großschreibung geändert. CHAR_TO_UUID('A0bF4E45-3029-2a44-D493-4998c9b439A3') soll ergeben: A0BF4E4530292A44D4934998C9B439A3 GUI
CONCAT(str1,str2) HSQLDB ~~FIREBIRD~~	Verbindet str1 + str2 GUI
'str1'||'str2'||'str3' HSQLDB FIREBIRD	Verbindet str1 + str2 + str3, einfachere Alternative zu CONCAT GUI
'str1'+'str2'+'str3' HSQLDB ~~FIREBIRD~~	Verbindet str1 + str2 + str3, einfachere Alternative zu CONCAT GUI

DIFFERENCE(s1,s2) HSQLDB ~~FIREBIRD~~	Gibt den «Klangunterschied» 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 GUI

GEN_UUID
~~HSQLDB~~ FIREBIRD
	Ergibt einen einzigartigen String mit 16 Byte Länge erstellen.
	`001 SELECT UUID_TO_CHAR(GEN_UUID()) AS "UUID"`
	`002 FROM "RDB$DATABASE"`
	GUI

HASH(s) ~~HSQLDB~~ FIREBIRD	Erstellt einen Hash (Streuwert) für einen eingegebenen Text. Das Ergebnis ist eine BIGINT-Zahl. GUI
HEXTORAW(s) HSQLDB ~~FIREBIRD~~	Übersetzt Hexadezimalcode in andere Zeichen. Der Hexadezimalcode muss dabei aus einer durch 4 teilbaren Anzahl an Zeichen bestehen. **`HEXTORAW('0041')`** = 'A' (siehe auch den Hexadezimalcode über **Einfügen → Sonderzeichen** im Writer) GUI
INSERT(s1,start,len,s2) HSQLDB ~~FIREBIRD~~ OVERLAY(s1 PLACING s2 FROM start [FOR len]) ~~HSQLDB~~ FIREBIRD	Gibt einen Text wieder, bei dem Teile ersetzt werden. Beginnend mit «start» wird über eine Länge «len» aus dem Text s1 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')`** oder **`OVERLAY( 'Bundesbahn' PLACING 's und B' FROM 3 FOR 5)`** 'Bus und Bahn'. GUI (HSQLDB) ~~GUI (FIREBIRD)~~
LCASE(s) HSQLDB ~~FIREBIRD~~	Wandelt den String in Kleinbuchstaben um. GUI
LEFT(s,count) HSQLDB FIREBIRD	Gibt die mit count angegebene Zeichenanzahl vom Beginn des Textes s wieder. GUI
LENGTH(s) HSQLDB ~~FIREBIRD~~	Gibt die Länge eines Textes in Anzahl der Buchstaben wieder. GUI
LIST ([ALL | DISTINCT] expression [, separator]) ~~HSQLDB~~ FIREBIRD	Fasst alle Einträge zu einem Feldinhalt zusammen, die in einem Feld (in Abhängigkeit von der Gruppierung anderer Felder) stehen. **`SELECT "Nachname", CAST( LIST( "Vorname", ', ' ) AS VARCHAR ( 8000 ) ) AS "Vornamen" FROM "Tabelle" GROUP BY "Nachname"`** Die Umformung in den VARCHAR-Dateityp war bei Einführung von LO 5.3 noch notwendig. Mit LO 5.3.1 war dieser Bug behoben. GUI
LOCATE(search,s[,start]) HSQLDB ~~FIREBIRD~~	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. GUI
LOWER(s) HSQLDB FIREBIRD	Wie LCASE(s) GUI

LPAD(s,len[,pad]) ~~HSQLDB~~ FIREBIRD	Der String s wird mit den in pad angegebenen Zeichen bis zur Länge len von links an aufgefüllt. Ist pad nicht angegeben, so wird ein Leerzeichen genutzt. Ist s länger als len, so wird s auf len von rechts aus gekürzt. GUI
LTRIM(s) HSQLDB ~~FIREBIRD~~	Entfernt führende Leerzeichen und nicht druckbare Zeichen von einem Text. GUI
OCTET_LENGTH(s)	Gibt die Länge eines Textes in Bytes an. Dies entspricht dem doppelten Wert der Zeichenanzahl. HSQLDB Dies entspricht dem UTF8-Wert der Zeichenanzahl. Sonderzeichen haben eine Länge von 2. FIREBIRD GUI
POSITION(s1 IN s2) HSQLDB FIREBIRD POSITION(s1, s2[, start]) ~~HSQLDB~~ FIREBIRD	Wenn der erste Text in dem zweiten enthalten ist wird die Position des ersten Textes wiedergeben, ansonsten 0. Bei der 2. Version kann auch der Startpunkt der Suche festgelegt werden. Dies könnte statt einer Suchmöglichkeit mit «LIKE» besonders bei umfangreichen Texten genutzt werden. GUI
RAWTOHEX(s) HSQLDB ~~FIREBIRD~~	Verwandelt in die Hexadezimalschreibweise, Umkehr von HEXTO-RAW() GUI
REPEAT(s,count) HSQLDB ~~FIREBIRD~~	Wiederholt den Text s count Mal GUI
REPLACE(s1,replace,s2) HSQLDB FIREBIRD	Ersetzt alle vorkommenden Textstücke mit dem Inhalt replace im Text s1 durch den Text s2 GUI
REVERSE(s) ~~HSQLDB~~ FIREBIRD	Schreibt den String verkehrt herum. GUI
RIGHT(s,count) HSQLDB FIREBIRD	Umgekehrt zu LEFT; gibt die mit count angegebene Zeichenzahl vom Textende aus wieder. GUI
RPAD(s,len[,pad]) ~~HSQLDB~~ FIREBIRD	Der String s wird mit den in pad angegebenen Zeichen bis zur Länge len von rechts an aufgefüllt. Ist pad nicht angegeben, so wird ein Leerzeichen genutzt. Ist s länger als len, so wird s auf len von rechts aus gekürzt. GUI
RTRIM(s) HSQLDB ~~FIREBIRD~~	Entfernt alle Leerzeichen und nicht druckbaren Zeichen am Tex-tende. GUI
SOUNDEX(s) HSQLDB ~~FIREBIRD~~	Gibt einen Code von 4 Zeichen wieder, die dem Klang von s ent-sprechen sollen – passt zu der Funktion DIFFERENCE() GUI
SPACE(count) HSQLDB ~~FIREBIRD~~	Gibt die in count angegebene Zahl an Leertasten wieder. GUI
SUBSTR(s,start[,len]) HSQLDB ~~FIREBIRD~~	Kürzel für SUBSTRING GUI

SUBSTRING(s,start[,len]) HSQLDB ~~FIREBIRD~~ SUBSTRING(s FROM start [FOR len]) HSQLDB FIREBIRD	Gibt den Text s ab der Startposition wieder. (1=links) . Wird die Länge ausgelassen, so wird der gesamte Text wiedergegeben. Auch bei einer größeren Länge als der Textlänge wird der restliche Text wiedergegeben. HSQLDB: Startposition < 0 beginnt von rechts rückwärts. GUI Liefert den Teil eines Textes ab der in FROM angegebenen Start-position, optional in der in FOR angegebenen Länge. Steht im Feld "Name" z.B. 'Roberta', so ergibt **SUBSTRING("Name" FROM 3 FOR 3)** den Teilstring 'bert'. FIREBIRD: Startposition muss > 0 sein. GUI
TRIM([{LEADING | TRAI-LING | BOTH}] FROM s) HSQLDB FIREBIRD	Nicht druckbare Sonderzeichen und Leerzeichen werden entfernt. GUI
UCASE(s) HSQLDB ~~FIREBIRD~~	Wandelt den String in Großbuchstaben um. GUI
UPPER(s) HSQLDB FIREBIRD	Wie UCASE(s) GUI
UUID_TO_CHAR ~~HSQLDB~~ FIREBIRD	Macht aus einer 16-Byte UUID eine 36 Zeichen lange UUID-Kette mit Bindestrichen. Sonderzeichen haben eine Länge von 2 Byte, so dass dafür die Zeichenzahl gekürzt werden müsste. **UUID_TO_CHAR('LibreOffice Base')** ergibt 4C696272-654F-6666-6963-652042617365 GUI
Datum/Zeit	
CURDATE() HSQLDB ~~FIREBIRD~~	Gibt das aktuelle Datum wieder. GUI
CURRENT_DATE HSQLDB FIREBIRD	Synonym für CURDATE(), SQL-Standard GUI
CURTIME() HSQLDB ~~FIREBIRD~~	Gibt die aktuelle Zeit wieder. GUI
CURRENT_TIME HSQLDB FIREBIRD	Synonym für CURTIME(), SQL-Standard GUI
CURRENT_TIMESTAMP HSQLDB FIREBIRD	Synonym für NOW(), SQL-Standard, FIREBIRD liefert eine Genauigkeit von 1/1000 Sekunden GUI

DATEADD(string, n, datetime) DATEADD(n string TO datetime) ~~HSQLDB~~ FIREBIRD	Addiert zu einer Datums bzw. Datumszeitangabe eine entsprechende Zeit, die über die Ganzzahl n und den string festgelegt wird. Der Eintrag in string entscheidet darüber, in welcher Einheit der Unterschied wiedergegeben wird: millisecond, second, minute, hour, day, week, month, year. Die string-Eingaben dürfen nicht mit einfachen Anführungszeichen maskiert sein. ~~GUI~~ Alternativ ist für Tagesangaben in FIREBIRD möglich: "Datum" + "Zeit" Timestamp GUI "Datum" + 1 Datum, Integerzahl wird als Tag gewertet GUI "Datum" – 1 Datum, Integerzahl wird als Tag gewertet GUI "Zeit" + 1 Zeit, Integerzahl wird als Sekunde gewertet GUI "Zeit" – 1 Zeit, Integerzahl wird als Sekunde gewertet GUI "Zeitstempel" + 2.75 Zeitstempel, Ganzzahl wird als Datum, Nachkommastellen als Bruchteil des Tages gewertet, hier also 2 Tage und 18 Stunden GUI "Zeitstempel" – 2.75 Zeitstempel GUI Nur als Differenz, nicht als Summe funktionieren: "Datum1" – "Datum2" Differenz in Tagen GUI "Zeit1" – "Zeit2" Differenz in Sekunden GUI "Zeitstempel1" – "Zeitstempel2" Differenz in Tagen und Bruchteilen von Tagen GUI Besonderheit: CURRENT_DATE – "Datum" ~~GUI~~ CAST(CURRENT_DATE AS DATE) – "Datum" GUI
DATEDIFF(string, datetime1, datetime2) HSQLDB FIREBIRD DATEDIFF(string FROM datetime1 TO datetime2) ~~HSQLDB~~ FIREBIRD	Datumsunterschied zwischen zwei Datums- bzw. Datumszeitangaben. HSQLDB: 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. GUI Firebird: Der Eintrag in string entscheidet darüber, in welcher Einheit der Unterschied wiedergegeben wird: millisecond, second, minute, hour, day, week, month, year. Die string-Eingaben dürfen nicht mit einfachen Anführungszeichen maskiert sein. ~~GUI~~
DAY(date) HSQLDB ~~FIREBIRD~~	Gibt den Tag im Monat wieder. (1-31) GUI
DAYNAME(date) HSQLDB ~~FIREBIRD~~	Gibt den englischen Namen des Tages wieder. GUI
DAYOFMONTH(date) HSQLDB ~~FIREBIRD~~	Gibt den Tag im Monat wieder. (1-31), Synonym für DAY() GUI
DAYOFWEEK(date) HSQLDB ~~FIREBIRD~~	Gibt den Wochentag als Zahl wieder. (1 bedeutet Sonntag) GUI
DAYOFYEAR(date) HSQLDB ~~FIREBIRD~~	Gibt den Tag im Jahr wieder. (1-366) GUI
HOUR(time) HSQLDB ~~FIREBIRD~~	Gibt die Stunde wieder. (0-23) GUI

MINUTE(time) HSQLDB ~~FIREBIRD~~	Gibt die Minute wieder. (0-59) GUI
MONTH(date) HSQLDB ~~FIREBIRD~~	Gibt den Monat wieder. (1-12) GUI
MONTHNAME(date) HSQLDB ~~FIREBIRD~~	Gibt den englischen Namen des Monats wieder. GUI
NOW() HSQLDB ~~FIREBIRD~~ DATE 'NOW', TIME 'NOW', TIMESTAMP 'NOW' ~~HSQLDB~~ FIREBIRD	Gibt das aktuelle Datum und die aktuelle Zeit zusammen als Zeitstempel wieder. Stattdessen kann auch CURRENT_TIMESTAMP genutzt werden. GUI Kurzform **`TIMESTAMP 'NOW'`** usw.: ~~GUI~~ **`Time 'Now'`** liefert in FIREBIRD eine Genauigkeit von 1/1000 Sekunden Langform **`CAST('NOW' AS TIMESTAMP)`** usw.: GUI
QUARTER(date) HSQLDB ~~FIREBIRD~~	Gibt das Quartal im Jahr wieder. (1-4) GUI
SECOND(time) HSQLDB ~~FIREBIRD~~	Gibt die Sekunden einer Zeitangabe wieder. (0-59) GUI
WEEK(date) HSQLDB ~~FIREBIRD~~	Gibt die Woche des Jahres wieder. (1-53) GUI
YEAR(date) HSQLDB ~~FIREBIRD~~	Gibt das Jahr aus einer Datumseingabe wieder. GUI
TODAY HSQLDB ~~FIREBIRD~~	Synonym für CURDATE() GUI
DATE 'TODAY' ~~HSQLDB~~ FIREBIRD	Gibt das aktuelle Datum an, Langform: CAST('TODAY' AS DATE) GUI (Langform) ~~GUI~~ (Kurzform)
DATE 'YESTERDAY' ~~HSQLDB~~ FIREBIRD	Gibt das Datum von Gestern an, Langform: CAST('YESTERDAY' AS DATE) GUI (Langform) ~~GUI~~ (Kurzform)
DATE 'TOMORROW' ~~HSQLDB~~ FIREBIRD	Gibt das Datum von Morgen an, Langform: CAST('TOMORROW' AS DATE) GUI (Langform) ~~GUI~~ (Kurzform)

TO_CHAR(datetime, string) HSQLDB ~~FIREBIRD~~	Setzt eine Datums- bzw. Timestampeingabe in einen entsprechenden String um. Folgende Zeichen sind definiert: 'Y' bis 'YYYY' (Jahr, 1 bis 4 Stellen) 'IY' bis 'IYYY' (Jahr nach ISO, 2 bis 4 Stellen) 'MM' (Monat, zweistellig) 'MON' (Monatsname, Kurzfassung, berücksichtigt GUI-Sprache) 'MONTH' (Monatsname, Langfassung) 'w' (Woche des Jahres) 'W' (Woche des Monats) 'IW' (Woche des Jahres nach ISO-Standard) 'd' (Tag der Woche) 'D' (Tagesname, Kurzfassung, berücksichtigt GUI-Sprache) 'DD' oder 'dd' (Tag des Monats, zweistellig) 'H' (Stunde von 0-23) 'HH' (Stunde von 0-11) 'm' (Minute) 's' (Sekunde) 'a' (AM oder PM für Stunde von 0-11) viele andere Zeichen werden direkt übernommen: **`TO_CHAR(CURRENT_DATE,'D, DD.MON.YYYY')`** = Sa, 05.Nov.2016 Die Funktion läuft nur mit bereits eingegebenen Datumswerten, die aus der Datenbank gelesen werden. GUI
EXTRACT(string FROM datetime) HSQLDB FIREBIRD	Kann viele der Datums- und Zeitfunktionen ersetzen. Gibt das Jahr, den Monat, den Tag usw. von einem Datums- bzw. Datumszeitwert wieder. **`EXTRACT(DAY FROM "Datum")`** gibt den Tag im Monat wieder. HSQLDB: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND Firebird zusätzlich: WEEK, WEEKDAY, YEARDAY, MILLISECOND (~~GUI~~). GUI **Hinweis:** Die EXTRACT-Funktion scheint für die GUI bessere Ergebnisse zu liefern als die Kurzfunktionen YEAR(). Bei einer Formularkonstruktion, in der Formular und Unterformular über Felder verbunden wurden, die beide über YEAR("Datum") erstellt wurden, erschien im Unterformular kein Wert. Erst mit EXTRACT(YEAR FROM "Datum") arbeitete das Unterformular wie gewünscht.

Datenbankverbindung

DATABASE() HSQLDB ~~FIREBIRD~~	Gibt den kompletten Pfad und Namen der Datenbank, die zu dieser Verbindung gehört, wieder. GUI
USER() HSQLDB ~~FIREBIRD~~ USER ~~HSQLDB~~ FIREBIRD	Gibt den Benutzernamen dieser Verbindung wieder. Der Nutzername ist dann von Bedeutung, wenn die Datenbank in eine externe Datenbank umgewandelt werden soll. Standard in HSQLDB: SA Standard in Firebird: SYSDBA ~~GUI (HSQLDB)~~ GUI (FIREBIRD)
CURRENT_USER HSQLDB FIREBIRD	SQL Standardfunktion, Synonym für USER(). Zu beachten ist, dass hier keine Klammern zu setzen sind. GUI

CURRENT_CONNECTION ~~HSQLDB~~ FIREBIRD	Gibt einen INTEGER-Wert für die aktuelle Datenbankverbindung wieder. ~~GUI~~
CURRENT_ROLE ~~HSQLDB~~ FIREBIRD	Gibt in der Regel NONE für die aktuelle Datenbankverbindung wieder. Funktioniert nicht bei der Abfrage von Tabellen, sondern nur bei direkter Abfrage an die Datenbank selbst. ~~GUI~~
CURRENT_TRANSACTION ~~HSQLDB~~ FIREBIRD	Gibt die Transaktionsnummer für die aktuelle Datenbankverbindung wieder. Funktioniert nicht bei der Abfrage von Tabellen, sondern nur bei direkter Abfrage an die Datenbank selbst. ~~GUI~~
IDENTITY() HSQLDB ~~FIREBIRD~~	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. GUI
RDB$GET_CONTEXT('name', 'varname') ~~HSQLDB~~ FIREBIRD	Als 'name' können angegeben werden: SYSTEM, USER_SESSION, USER_TRANSACTION. USER_SESSION und USER_TRANSACTION sind zum Start leer und können mit beliebigen Variablen belegt werden. SYSTEM hat folgende vorbelegten 'varname': DB_NAME Pfad zu der ausgepackten Firebird-Datenbank NETWORK_PROTOCOL 'TCPv4', 'WNET', 'XNET' oder NULL CLIENT_ADDRESS abhängig von PROTOCOL, intern NULL CURRENT_USER wie CURRENT_USER direkt CURRENT_ROLE wie CURRENT_ROLE direkt SESSION_ID wie CURRENT_CONNECTION direkt TRANSACTION_ID wie CURRENT_TRANSACTION direkt ISOLATION_LEVEL 'READ COMMITTED', 'SNAPSHOT' oder 'CONSISTENCY'. ENGINE_VERSION Firebird-Server, LO 5.3: '3.0.0' ~~GUI~~
RDB$SET_CONTEXT ('name', 'varname', wert | NULL) ~~HSQLDB~~ FIREBIRD	Ist das Gegenstück zu RDB$GET_CONTEXT(). Als 'name' können aber nur USER_SESSION und USER_TRANSACTION angegeben werden. SYSTEM kann nicht beschrieben werden. **RDB$SET_CONTEXT('USER_SESSION', 'Autor', 'Robert')** schreibt die Variable über eine Abfrage nach USER_SESSION. In der aktuellen Zeile der Abfrage ist der Wert der Variablen noch NULL oder eben der vorhergehende Wert. mit **RDB$GET_CONTEXT('USER_SESSION', 'Autor')** ist in den folgenden Zeilen der Abfrage der Wert 'Robert' verfügbar. ~~GUI~~

System	
IFNULL(exp,value) HSQLDB ~~FIREBIRD~~	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. IFNULL ist eine wichtige Funktion, wenn Felder durch Rechnung oder CONCAT miteinander verbunden werden. Der Inhalt des Ergebnisses wäre NULL, wenn auch nur ein Wert NULL ist. **`"Nachname"||', '||"Vorname"`** würde für Personen, bei denen z.B. der Eintrag für "Vorname" fehlt, ein leeres Feld, also NULL ergeben. **`"Nachname"||IFNULL(', '||"Vorname",'')`** würde stattdessen auch nur "Nachname" ausgeben. GUI
CASEWHEN(exp,v1,v2) HSQLDB ~~FIREBIRD~~ IIF(exp,v1,v2) ~~HSQLDB~~ FIREBIRD	Wenn exp wahr ist wird v1 zurückgegeben, sonst v2. Stattdessen kann auch CASE WHEN genutzt werden. **`CASEWHEN("a">10,'Ziel erreicht','noch üben')`** gibt 'Ziel erreicht' aus, wenn der Inhalt des Feldes "a" größer als 10 ist. GUI
CONVERT(term,type) HSQLDB ~~FIREBIRD~~ CAST(term AS type) HSQLDB FIREBIRD	Wandelt term in einen anderen Datentyp um. **`CONVERT("a",DECIMAL(5,2))`** macht aus dem Feld "a" ein Feld mit 5 Ziffern, davon 2 Nachkommastellen. Ist die Zahl zu groß, so wird ein Fehler ausgegeben. GUI
COALESCE(expr1, expr2, expr3,...) HSQLDB FIREBIRD	Wenn expr1 nicht NULL ist, wird expr1 wiedergegeben, ansonsten wird expr2 überprüft, danach dann expr3 usw. Sämtliche Ausdrücke müssen zumindest einen ähnlichen Datentyp haben. So geht die alternative Darstellung von Ganzzahlen und Fließkommazahlen, aber nicht auch noch des eines Datums- oder Zeitwertes. **`COALESCE("Spitzname", "Vorname", 'Herrn/Frau')`** wählt den Spitznamen, wenn dieser vorhanden ist, ansonsten den Vornamen und wenn dieser auch nicht bekannt ist, dann allgemein 'Herrn/Frau'. GUI
NULLIF(v1,v2) HSQLDB FIREBIRD	Wenn v1 gleich v2 ist, wird NULL wiedergegeben, ansonsten v1. Die Daten müssen vom Typ her vergleichbar sein. GUI
CASE v1 WHEN v2 THEN v3 [ELSE v4] END HSQLDB FIREBIRD DECODE(v1, v2, v3, v4) ~~HSQLDB~~ FIREBIRD	Wenn v1 gleich v2 ist, wird v3 wiedergegeben. Sonst wird v4 wiedergegeben oder NULL, wenn kein ELSE formuliert ist. ~~GUI (HSQLDB)~~ GUI (FIREBIRD)
CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END HSQLDB FIREBIRD	Wenn expr1 wahr ist wird v1 zurückgegeben. [Optional können weitere Fälle angegeben werden] Sonst wird v4 wiedergegeben oder NULL, wenn kein ELSE formuliert ist. **`CASE WHEN DAYOFWEEK("Datum")=1 THEN 'Sonntag' WHEN DAYOFWEEK("Datum")=2 THEN 'Montag' … END`** könnte per SQL den Tagesnamen ausgeben, der sonst in der Funktion nur in Englisch verfügbar ist. GUI
MAXVALUE(expr [, expr ...]) ~~HSQLDB~~ FIREBIRD	Sucht den höchsten Wert im Vergleich zu verschiedenen Ausdrücken heraus. Ergibt NULL, wenn ein Wert leer ist. Im Gegensatz zur Aggregatfunktion MAX() können hier die Werte verschiedener Felder miteinander verglichen werden. GUI

MINVALUE(expr [, expr ...]) ~~HSQLDB~~ FIREBIRD	Sucht den niedrigsten Wert im Vergleich zu verschiedenen Ausdrücken heraus. Ergibt NULL, wenn ein Wert leer ist. Im Gegensatz zur Aggregatfunktion MIN() können hier die Werte verschiedener Felder miteinander verglichen werden. GUI

Window-Funktionen bei Firebird

Window-Funktionen[2] filtern nicht die Ergebnismenge einer Abfrage. Die Ergebnisse der Funktionen werden zusätzlich in entsprechenden Spalten angegeben.

Window-Funktionen können in der **SELECT**-Liste oder in der **ORDER BY**-Definition vorkommen. In den Window-Funktionen ist eine **OVER**-Klausel zwingend enthalten. In dieser Klausel können Partitionierungen und Sortierungen vorgenommen werden.

<Funktionsname>() OVER (PARTITION BY … | ORDER BY …)

Ranking-Funktionen

RANK() OVER (ORDER BY ...)

	ID	Name	Laufzeit	RANK
▶	3	Markus	01:37:32	1
	2	Linda	01:39:12	2
	5	Klaus	01:39:12	2
	1	Erich	01:42:54	4
	4	Nina	01:43:17	5
+				

Datensatz 1 von 5

```
SELECT "ID", "Name", "Laufzeit",
RANK() OVER (ORDER BY "Laufzeit") FROM "tbl_Starter"
```

Kann zur Bestimmung der Reihenfolge in einer Datenmenge eingesetzt werden. Gleiche Werte werden dabei mit der gleichen Position versehen. Die Position wird wie bei Sportwettkämpfen erstellt. Richtet sich nach der Sortierung, die angegeben wird.
GUI

RANK() OVER (PARTITION BY ... ORDER BY ...)

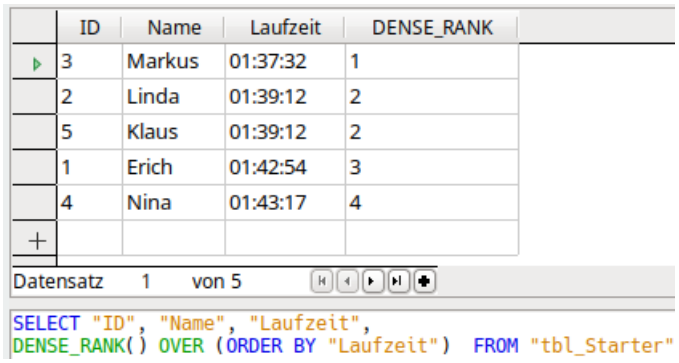
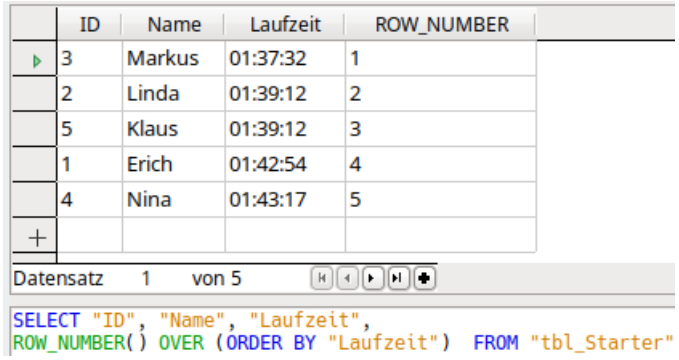
	ID	Name	Laufzeit	Geschlecht	RANK
▶	3	Markus	01:37:32	m	1
	5	Klaus	01:39:12	m	2
	1	Erich	01:42:54	m	3
	2	Linda	01:39:12	w	1
	4	Nina	01:43:17	w	2
+					

Datensatz 1 von 5

```
SELECT "ID", "Name", "Laufzeit", "Geschlecht",
RANK() OVER (PARTITION BY "Geschlecht" ORDER BY "Laufzeit")
FROM "tbl_Starter"
```

Gleiche Funktion wie oben, nur mit der Ergänzung der Partitionierung. Hier ist die Reihenfolge abhängig vom Feld "Geschlecht".

2 Siehe die Sprachreferenz für Firebird 3.0: *https://www.firebirdsql.org/en/reference-manuals/*

DENSE_RANK() OVER (ORDER BY ...)	 Kann zur Bestimmung der Reihenfolge in einer Datenmenge eingesetzt werden. Gleiche Werte werden dabei mit der gleichen Position versehen. Dabei rücken tiefer liegende Werte auf. Richtet sich nach der Sortierung, die angegeben wird. GUI
ROW_NUMBER() OVER (ORDER BY ...)	 Kann zur Bestimmung der laufenden Nummer in einer Datenmenge eingesetzt werden. Gleiche Werte werden dabei nach dem Primärschlüssel zusätzlich sortiert. Richtet sich nach der Sortierung, die angegeben wird. GUI
Navigationsfunktionen	

FIRST_VALUE(...) OVER (ORDER BY ...)	`001 SELECT "ID", "Name", "Laufzeit",`
	`002 FIRST_VALUE("Laufzeit") OVER (ORDER BY "Laufzeit")`
	`003 FROM "Starter"`
	Hier wird der Wert für die niedrigste Laufzeit neben angezeigten aktuellen Laufzeit angegeben. Es könnte also direkt die Differenz zum ersten Platz ausgerechnet werden.
	GUI

LAG(...,[offset,[default]]) OVER (ORDER BY ...)	`001 SELECT "ID", "Name", "Laufzeit",`
	`002 LAG("Laufzeit") OVER (ORDER BY "Laufzeit")`
	`003 FROM "Starter"`
	Hier wird der Wert für die Laufzeit angezeigt, die die Person gelaufen hat, die direkt vorher ins Ziel gekommen ist. Es könnte also direkt die Differenz zur vorher durchs Zeile gekommenen Person ausgerechnet werden.
	Wird ein **`offset`** angegeben, so kann auch der Abstand zu weiter vorher durchs Ziel gekommenen Personen ermittelt werden. Als **`default`** wird ohne Angabe **`NULL`** gesetzt, ansonsten der korrekt formatierte Wert für das entsprechende Feld, hier z. B. '00:00:00'.
	GUI

LAST_VALUE(...) OVER (ORDER BY ...)	Hier wird der zuletzt ausgelesene Wert wieder gegeben. Dies ist also nicht der Wert für die zuletzt durchs Ziel gekommene Person, sondern der aktuelle Wert, der auch so in der Laufzeit steht. GUI
LEAD(...,[offset,[default]]) OVER (ORDER BY ...)	Dies ist die Umkehrung der Funktion von LAG. Es wird die Zeit angezeigt, die die Person gelaufen ist, die direkt nach der aktuellen Position ins Ziel gekommen ist. Wird ein **offset** angegeben, so kann auch der Abstand zu weiter hinten durchs Ziel gekommenen Personen ermittelt werden. Als **default** wird ohne Angabe **NULL** gesetzt, ansonsten der korrekt formatierte Wert für das entsprechende Feld, hier z. B. '00:00:00'. GUI

NTH_VALUE(...,offset) [FROM {FIRST | LAST}] OVER (ORDER BY ...)	`001 SELECT "ID", "Name", "Laufzeit",`
	`002 NTH_VALUE("Laufzeit",1) OVER (ORDER BY "Laufzeit")`
	`003 FROM "Starter"`
	Ein **offset** muss angegeben werden. Die obige Abfrage gibt mit dem **offset** '1' genau das gleiche Ergebnis wie **FIRST_VALUE**, kann durch das **offset** aber auf die verschiedenen anderen Werte eingestellt werden. Standardmäßig wird vom ersten Wert (**FROM FIRST**) aus ermittelt. Ansonsten vom zuletzt ermittelten Wert.
	GUI

Aggregatfunktionen als Windowfunktionen

Alle Aggregatfunktionen können auch als Windowfunktionen erstellt werden. Damit entfällt die Gruppierungsanweisung (die darüber hinaus die einzelnen Datensätze nicht mehr komplett anzeigt) oder die Notwendigkeit, innerhalb einer Abfrage eine separate Unterabfrage zu erstellen. Alle Aggregatfunktionen können genutzt werden. Hier nur ein Beispiel für die Nutzung von **SUM**.

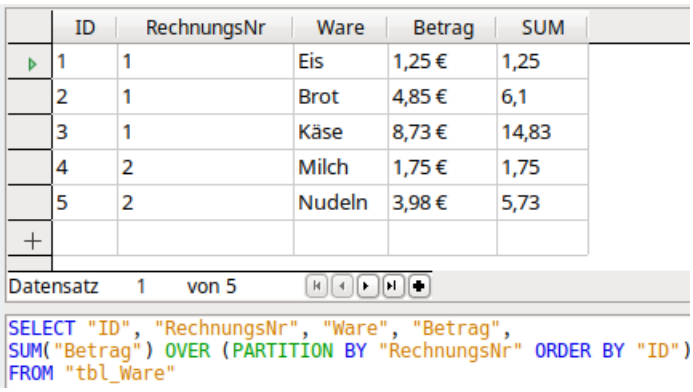
SUM(...) OVER (PARTITION BY ...)

	ID	RechnungsNr	Ware	Betrag	SUM
▶	1	1	Eis	1,25 €	14,83
	2	1	Brot	4,85 €	14,83
	3	1	Käse	8,73 €	14,83
	4	2	Milch	1,75 €	5,73
	5	2	Nudeln	3,98 €	5,73
+					

Datensatz 1 von 5

```
SELECT "ID", "RechnungsNr", "Ware", "Betrag",
SUM("Betrag") OVER (PARTITION BY "RechnungsNr")
FROM "tbl_Ware"
```

Die Summierung erfolgt nach der Ausgabe des gesamten Inhaltes der Tabelle. Keine Feld muss gruppiert werden. Die Summe wird für das Feld "Betrag" abhängig von der "RechnungsNr" berechnet.
GUI

SUM(…) OVER (PARTITION BY … ORDER BY …)	 Wird zusätzlich eine Sortierung vorgenommen, so wird die laufende Summe ermittelt. In der 2. Zeile also die Summe von Zeile 1 und 2, in der 3. Zeile dann die Summe von Zeile 1 und 2 und 3. Solch ein Abfrageergebnis ist sonst nur durch korrelierende Unterabfragen möglich. GUI

Migration HSQLDB → Firebird

Die folgenden Probleme können bei der Migration von Datenbanken der internen HSQLDB 1.8.0.10 zu der internen Firebird 3.0 auftauchen.

- Tabellennamen, Spaltennamen usw. sind in der HSQLDB von der Länge nicht begrenzt. Bei Firebird hingegen gilt eine Grenze von maximal 31 Zeichen.
- Texttabellen können in der HSQLDB zum Lesen, Schreiben und Ändern von Daten genutzt werden. Zur Zeit funktionieren Texttabelle in Firebird hingegen gar nicht, weil die Standardeinstellungen dafür bei der internen Datenbank ausgeschaltet sind. Selbst bei funktionierenden Texttabellen unterscheidet sich das Format der Tabellen, so dass z.B. keine *.csv-Dateien benutzt werden können. Außerdem ist das Ändern und Löschen von Daten in Firebird-Texttabellen nicht erlaubt.
- Unterformulare funktionieren nach der Nutzung des Migrationsassistenten nicht. Hier muss die Datenbankdatei mit einem Packprogramm geöffnet werden. Die content.xml muss zum Bearbeiten geöffnet werden. Der Eintrag **<db:driver-settings** db:system-driver-settings="" db:base-dn="" db:parameter-name-substitution="false"**/>** benötigt statt des **false** ein **true**. Entweder ist hier also **true** einzutragen oder der gesamte Inhalt db:parameter-name-substitution="false" zu löschen.
  Alternativ kann auch das folgende Makro von der migrierten Base-Datei aus gestartet werden:
  ```
  SUB FB_Parameter
     DIM oSettings AS OBJECT
     oSettings = ThisComponent.DataSource.Settings
     oSettings.ParameterNameSubstitution = True
  END SUB
  ```
  Nach einmaligem Start des Makros ist der Eintrag in der Base-Datei komplett verschwunden, wenn die Base-Datei anschließend abgespeichert wird.
- Sollen Bilder in der Datenbank gespeichert werden, so ist der Datentyp **BLOB [BLOB]** statt **Bild [BLOB]** zu währen. Nur Inhalte aus **BLOB [BLOB]** werden im grafischen Kontrollfeld angezeigt.

Hinweis

Eine Umstellung auf Firebird mit dem Migrationsassistenten kann auch wieder rückgängig gemacht werden. Die alte HSQLDB ist weiter in der Datenbankdatei enthalten. Aus
`<db:connection-resource xlink:href="sdbc:embedded:firebird" xlink:type="simple"/>`
in der content.xml muss wieder
`<db:connection-resource xlink:href="sdbc:embedded:hsqldb" xlink:type="simple"/>`
erstellt werden.
Nach dieser Änderung kann die Datei «firebird.fbk» in dem Verzeichnis «database» wieder entfernt werden.
Das Ganze geht natürlich auch über ein Makro[3]:

```
001  SUB FirebirdMigrationRemove
002     DIM oDoc AS OBJECT
003     DIM oDataSource AS OBJECT
004     DIM oSettings AS OBJECT
005     DIM oDocumentSubStorage AS OBJECT
006     DIM sURL AS STRING
007     oDoc = ThisComponent
008     sURL = "sdbc:embedded:hsqldb"
009     oDataSource = ThisComponent.DataSource
010     oDataSource.URL = sURL
011     oSettings = oDataSource.Settings
012     oSettings.JavaDriverClassPath = sURL
013     REM 4 = write mode
014     oDocumentSubStorage = oDoc.getDocumentSubStorage("database", 4)
015     oDocumentSubStorage.removeElement("firebird.fbk")
016     oDocumentSubStorage.commit()
017  END SUB
```

Aber Vorsicht: Die interne «firebird.fbk» ist jetzt auf jeden Fall gelöscht.

Im Folgenden sind Funktionen aufgelistet, die in Firebird nicht mit dem gleichen Namen existieren. Die erste Spalte enthält die Funktionsbezeichnung bei der Hsqldb, die zweite Spalte Funktionsbezeichnungen, die alternativ in der Hsqldb und in Firebird verwendet werden können. Existiert so eine Alternative nicht, so muss die dritte Spalte greifen, die Funktionen beschreibt, die in Firebird die gleiche Bedeutung haben wie die Funktionen der Hsqldb in der ersten Spalte.

Numerisch		
HSQLDB	***HSQLDB und Firebird***	***Firebird***
BITAND(a,b)		BIN_AND(a,b)
BITOR(a,b)		BIN_OR(a,b)
BITXOR(a,b)		BIN_XOR(a,b)
DEGREES(d)	<value for radians>*360/(2*PI())	
LOG(d)		LN(d)
RADIANS(d)	<value for degrees>*2*PI()/360	
ROUNDMAGIC(d)		~~Keine Funktion vorhanden~~
TRUNCATE(a,b)		TRUNC(a,b)

3 Dank an Ratslinger, *https://ask.libreoffice.org/u/ratslinger*

TEXT		
HSQLDB	***HSQLDB und Firebird***	***Firebird***
ASCII(s)		ASCII_VAL(s)
BIT_LENGTH(s)	Gibt die Texlänge wieder. HSQLDB: 16bit pro Zeichen Firebird: 8bit, Sonderzeichen 16bit. Gleiche Funktion, aber unterschiedliche Zählweise	BIT_LENGTH(s)
CHAR(c)		ASCII_CHAR© (nur 0 - 255)
CONCAT(str1,str2) 'str1'+'str2'+'str3'	'str1'||'str2'||'str3'	
DIFFERENCE(s1,s2)		~~Keine Funktion vorhanden~~
HEXTORAW(s)		~~Keine Funktion vorhanden~~
INSERT(s1,start,len,s2)		OVERLAY(s1 PLACING s2 FROM start [FOR len])
LCASE(s)	LOWER(s)	
LENGTH(s)	CHAR_LENGTH(s) CHARACTER_LENGTH()	
LOCATE(search,s[,start])	POSITION(search IN s)	POSITION(search,s[,start])
LTRIM(s)	TRIM(LEADING FROM s)	
OCTET_LENGTH(s)	Gleiche Funktion, aber unterschiedliche Zählweise	OCTET_LENGTH(s)
RAWTOHEX(s)		~~Keine Funktion vorhanden~~
REPEAT(s,count)		~~Keine Funktion vorhanden~~
RTRIM(s)	TRIM(TRAILING FROM s)	
SOUNDEX(s)		~~Keine Funktion vorhanden~~
SPACE(count)		~~Keine Funktion vorhanden~~
SUBSTR(s,start[,len]) SUBSTRING(s,start[,len])	SUBSTRING(s FROM start [FOR len])	
UCASE(s)	UPPER(s)	
Datum/Zeit		
HSQLDB	***HSQLDB und Firebird***	***Firebird***
CURDATE(), TODAY	CURRENT_DATE	
CURTIME()	CURRENT_TIME	
DATEDIFF(string, datetime1, datetime2) 'ms'='millisecond', 'ss'='second', 'mi'='minute', 'hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'		DATEDIFF(string, datetime1, datetime2) millisecond, second, minute, hour, day, week, month, year Nur die Langversion ohne «''»

DAY(date) DAYOFMONTH(date)	EXTRACT(DAY FROM date-time)	
DAYNAME(date)		~~Keine Funktion vorhanden~~
DAYOFWEEK(date)		EXTRACT(WEEKDAY FROM datetime)
DAYOFYEAR(date)		EXTRACT(YEARDAY FROM datetime)
HOUR(time)	EXTRACT(HOUR FROM date-time)	
MINUTE(time)	EXTRACT(MINUTE FROM date-time)	
MONTH(date)	EXTRACT(MONTH FROM date-time)	
MONTHNAME(date)		~~Keine Funktion vorhanden~~
NOW()	CURRENT_TIMESTAMP	
QUARTER(date)	CEILING(EXTRACT(MONTH FROM date)/3.00)	
SECOND(time)	EXTRACT(SECOND FROM datetime)	
WEEK(date)		EXTRACT(WEEK FROM date-time)
YEAR(date)	EXTRACT(YEAR FROM date-time)	
TO_CHAR(datetime, string)		~~Keine Funktion vorhanden~~
Datenbankverbindung		
HSQLDB	***HSQLDB und Firebird***	***Firebird***
DATABASE()		RDB$GET_CONTEXT('SYS-TEM', 'DB_NAME')
USER()	CURRENT_USER	
IDENTITY()		~~Keine Funktion vorhanden~~
System		
HSQLDB	***HSQLDB und Firebird***	***Firebird***
IFNULL(exp,value)	COALESCE(expr1,expr2,expr3,...)	
CASEWHEN(exp,v1,v2)	CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END	IIF(exp,v1,v2) (nur wenn CASE WHEN zu kompliziert sein sollte)
CONVERT(term,type)	CAST(term AS type)	

Steuerzeichen zur Nutzung in Abfragen

In Abfragen lassen sich Felder miteinander verknüpfen. Aus zwei Feldern in

```
001 SELECT "Vorname", "Nachname" FROM "Tabelle"
```

wird durch

```
001 SELECT "Vorname"||' '||"Nachname" FROM "Tabelle"
```

ein Feld. Hier wird noch ein Leerzeichen mit eingefügt. Natürlich lassen sich hier alle möglichen beliebigen Zeichen einfügen. Solange diese in `''` stehen werden sie als Text interpretiert. Manchmal ist es aber auch sinnvoll, Zeilenumbrüche z.B. für einen Bericht einzufügen. Deshalb hier eine kleine Liste von Steuerzeichen, die entsprechend durch einen Blick auf *http://de.wikipedia.org/wiki/Steuerzeichen* erweitert werden kann.

CHAR(9)	Horizontaler Tabulator	
CHAR(10)	Zeilenvorschub	Erzeugt in Serienbriefen und im Report-Builder in einem Feld einen Zeilenumbruch (Linux, Unix, Mac) Um den Zeilenumbruch auch in Abfragen anzuzeigen, muss das Feld in ein LONGVARCHAR-Feld umgewandelt werden.
CHAR(13)	Wagenrücklauf	Zeilenumbruch zusammen mit dem Zeilenvorschub in Windows. **`CHAR(13)||CHAR(10)`** ist auch für Linux, Mac usw. möglich, daher die universellere Variante.

Bei FIREBIRD ist statt **`CHAR( ) ASCII_CHAR( )`** erforderlich.

Einige uno-Befehle zur Nutzung mit einer Schaltfläche

Einer Schaltfläche können verschiedene uno-Befehle direkt zugeordnet werden. Dafür muss unter **Eigenschaften: Schaltfläche → Aktion → Dokument/Webseite öffnen** gewählt werden sowie z. B. als **URL → .uno:RecSearch** zum Öffnen der Suchfunktion eingetragen werden. Manchmal muss zusätzlich beachtet werden, dass **Fokussieren bei Klick → Nein** angewählt wird, wenn bestimmte Aktionen direkt auf ein Formularfeld zugreifen sollen, das dafür den Focus braucht, wie z. B. **.uno:Paste**, das den Inhalt aus der Zwischenablage einfügen kann.

Die folgende Liste gibt nur wenige Befehle wieder. Sämtliche Befehle aus der Navigationsleiste sind ja bereits in der Schaltfläche so verfügbar, könnten aber auch über die uno-Befehle erstellt werden. Viele Befehle lassen sich auch über den Makrorekorder ermitteln, der häufig diese uno-Befehle über einen Dispatcher aufruft. UNO-Befehle können auch z. B. über **Extras → Anpassen → Symbolleisten → Beschreibung** in jedem LO-Dokument ermittelt werden.

Uno-Befehl	***Anwendung für ...***
.uno:RecSearch	Öffnen der Suchfunktion im Formular
.uno:Paste	Zwischenablage einfügen, nur mit **Fokussieren bei Klick → Nein**
.uno:Copy	Kopiert den markierten Inhalt in die Zwischenablage, nur mit **Fokussieren bei Klick → Nein**
.uno:Print	Druckdialog für das Formular öffnen
.uno:PrintDefault	Drucken mit dem Standarddrucker ohne Dialog
.uno:NextRecord	Nächster Datensatz
.uno:PrevRecord	Vorhergehender Datensatz
.uno:FirstRecord	Erster Datensatz
.uno:LastRecord	Letzter Datensatz
.uno:NewRecord	Neuer Datensatz
.uno:RecSave	Datensatz speichern

.uno:DeleteRecord	Lösche den Datensatz
.uno:RecUndo	Nehme die letzten Eingaben zurück.
.uno:OpenURL	Öffne die angezeigte URL
.uno:Refresh	Angezeigte Daten aktualisieren
.uno:ViewFormAsGrid	Formular: Datenquelle als Tabelle aufrufen

Informationstabellen der HSQLDB

Innerhalb von HSQLDB-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.

```
001 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
```

Die folgende Abfrage gibt z.B. eine komplette Übersicht über alle in der Datenbank genutzten Tabellen mit Feldtypen, Primärschlüsseln und Fremdschlüsseln:

```
001 SELECT
002    "A"."TABLE_NAME",
003    "A"."COLUMN_NAME",
004    "A"."TYPE_NAME",
005    "A"."NULLABLE",
006    "B"."KEY_SEQ" AS "PRIMARYKEY",
007    "C"."PKTABLE_NAME" || '.' || "C"."PKCOLUMN_NAME" AS "FOREIGNKEY FOR"
008 FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" AS "A"
009    LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS" AS "B"
010    ON ( "B"."TABLE_NAME" = "A"."TABLE_NAME" AND "B"."COLUMN_NAME" =
          "A"."COLUMN_NAME" )
011    LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_CROSSREFERENCE" AS "C"
012    ON ( "C"."FKTABLE_NAME" = "A"."TABLE_NAME" AND "C"."FKCOLUMN_NAME" =
          "A"."COLUMN_NAME" )
013 WHERE "A"."TABLE_SCHEM" = 'PUBLIC'
```

Informationstabellen der Firebird-Datenbank

Die Firebird-Informationstabellen sind deutlich mehr gesplittet als die der HSQLDB. So stehen z.B. Tabellennamen, Feldnamen und Feldtypen nicht zusammen in einer Tabelle, sondern müssen über mehrere Tabellen hinweg zusammen gebracht werden:

```
001 SELECT
002    "a".RDB$RELATION_NAME AS "Tables",
003    "a".RDB$FIELD_NAME AS "Fields",
004    "c".RDB$TYPE_NAME AS "Types",
005    "a".RDB$Field_POSITION AS "Fieldposition",
006    "a".RDB$NULL_FLAG AS "Nullflag"
007 FROM RDB$RELATION_FIELDS AS "a", RDB$FIELDS AS "b", RDB$TYPES AS "c"
008 WHERE "a".RDB$FIELD_SOURCE = "b".RDB$FIELD_NAME
009    AND "b".RDB$FIELD_TYPE = "c".RDB$TYPE
010    AND "c".RDB$FIELD_NAME = 'RDB$FIELD_TYPE'
011    AND "a".RDB$SYSTEM_FLAG = 0
012 ORDER BY "Tables", "Fieldposition"
```

Hiermit wird eine Übersicht über alle selbst erstellten Tabellen mit den Feldnamen, Feldtypen, Feldpositionen innerhalb der Tabelle und der Information, ob sie leer sein dürfen, ermöglicht. Um den Feldtypen zuordnen zu können muss also über die Tabelle "RDB$FIELD_TYPE" zur Tabelle "RDB$TYPE" verbunden werden.

Leider stimmen die dort aufgeführten Typen nur mit denen überein, die intern von Firebird genutzt werden. Sie lauten anders als die, die in dem Tabelleneditor vorkommen. Hier ein Code, um die entsprechende Übersicht zu bekommen, die nur über Nummern aus den Informationstabellen auslesbar ist:

```
001 SELECT TRIM("a".RDB$RELATION_NAME) AS "Tabelle",
002    "a".RDB$FIELD_NAME AS "Feld",
003    TRIM(CASE "b".RDB$FIELD_TYPE||'|'|| COALESCE("b".RDB$FIELD_SUB_TYPE,0)
004       WHEN '7|0' THEN 'SMALLINT'
005       WHEN '8|0' THEN 'INTEGER'
006       WHEN '8|1' THEN 'NUMERIC'
```

```
007       WHEN '8|2' THEN 'DECIMAL'
008       WHEN '10|0' THEN 'FLOAT'
009       WHEN '12|0' THEN 'DATE'
010       WHEN '13|0' THEN 'TIME'
011       WHEN '14|0' THEN 'CHAR'
012       WHEN '16|0' THEN 'BIGINT'
013       WHEN '27|0' THEN 'DOUBLE PRECISION'
014       WHEN '35|0' THEN 'TIMESTAMP'
015       WHEN '37|0' THEN 'VARCHAR'
016       WHEN '261|0' THEN 'BLOB'
017       WHEN '261|1' THEN 'BLOB Text'
018       WHEN '261|2' THEN 'BLOB BLR'
019       WHEN '261|3' THEN 'BLOB ACL'
020       END) AS "SQL_Datentyp",
021    COALESCE(
022       "b".RDB$CHARACTER_LENGTH,
023       "b".RDB$FIELD_PRECISION||','||
024       ("b".RDB$FIELD_SCALE*-1)) AS "Laenge,Nachkommastellen"
025 FROM RDB$RELATION_FIELDS AS "a", RDB$FIELDS AS "b"
026 WHERE "a".RDB$FIELD_SOURCE = "b".RDB$FIELD_NAME
027    AND "a".RDB$SYSTEM_FLAG = 0
```

Diese Abfrage gibt alle genutzten Tabellen mit den Feldnamen, den Feldtypen und der Feldlänge bzw. Genauigkeit bei Dezimalzahlen an.

Die Ausgabe in dieser Abfrage erfolgt in festen Spaltenbreiten. Von der HSQLDB ist es eher so, dass nur die Anzahl der lesbaren Zeichen wiedergegeben werden. Falls also eine Ausgabe nach den Bezeichnern ausgewertet werden soll, so sind die Leerzeichen durch **TRIM** zu entfernen.

Automatisch generierte Werte wie in der HSQLDB werden bei Firebird über Generatoren erstellt. Um den Startwert eines solchen Generators neu einstellen zu können ist der Name des Generators erforderlich. Die folgende Abfrage liefert den Generatornamen für alle Tabellen, die von dem Nutzer erstellt wurden:

```
001 SELECT RDB$FIELD_NAME, RDB$RELATION_NAME, RDB$GENERATOR_NAME
002 FROM RDB$RELATION_FIELDS
003 WHERE RDB$GENERATOR_NAME IS NOT NULL
```

Ansichten werden auch von Firebird unterstützt. Die folgende Abfrage liefert den Namen und die SQL-Formulierung für jede Ansicht:

```
001 SELECT RDB$RELATION_NAME, RDB$VIEW_SOURCE
002 FROM RDB$RELATIONS
003 WHERE RDB$VIEW_SOURCE IS NOT NULL
```

Die Firebird-Systemtabellen starten alle mit den Anfangsbuchstaben RDB$:

```
RDB$BACKUP_HISTORY
RDB$CHARACTER_SETS
RDB$CHECK_CONSTRAINTS
RDB$COLLATIONS
RDB$DATABASE
RDB$DEPENDENCIES
RDB$EXCEPTIONS
RDB$FIELDS
RDB$FIELD_DIMENSIONS
RDB$FILES
RDB$FILTERS
RDB$FORMATS
RDB$FUNCTIONS
RDB$FUNCTION_ARGUMENTS
RDB$GENERATORS
RDB$INDICES
RDB$INDEX_SEGMENTS
```

```
RDB$LOG_FILES
RDB$PAGES
RDB$PROCEDURES
RDB$PROCEDURE_PARAMETERS
RDB$REF_CONSTRAINTS
RDB$RELATIONS
RDB$RELATION_CONSTRAINTS
RDB$RELATION_FIELDS
RDB$ROLES
RDB$SECURITY_CLASSES
RDB$TRANSACTIONS
RDB$TRIGGERS
RDB$TRIGGER_MESSAGES
RDB$TYPES
RDB$USER_PRIVILEGES
RDB$VIEW_RELATIONS
```

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.

Formulare, Abfragen und Berichte können, sofern eine Vorversion der Datenbank gesichert wurde, über die Zwischenablage in eine neue Datenbank kopiert werden. Lässt sich allerdings, aus welchen Gründen auch immer, eine aktuelle Datenbankdatei nicht mehr öffnen, so ist das Hauptproblem: Wie komme ich (hoffentlich) an die Daten.

Hinweis

Bevor es an die Wiederherstellung einer Datenbank-Archivdatei geht sollte erst einmal geklärt werden, ob nicht irgendwo im System doch eine Sicherungskopie existiert, die nicht bewusst angelegt wurde.
Eine Möglichkeit ist der Backup-Pfad von LibreOffice: **Extras → Optionen → LibreOffice → Pfade → Sicherungskopien** gibt darüber Aufschluss. Falls dieser Pfad noch nicht existiert wurde er noch nicht genutzt.
Eine weitere Möglichkeit ist das temporäre Verzeichnis des Betriebssystems: **Extras → Optionen → LibreOffice → Pfade → Temporäre Dateien** dürfte hier den richtigen Pfad angeben. Unter Linux ist dies **/tmp**. In diesem Pfad befinden sich Verzeichnisse, die mit lu….tmp gekennzeichnet sind, also von der Namensgebung so aussehen, als wären es Dateinamen mit der Endung «tmp». In diesen Pfaden befinden sich Dateien von LibreOffice, sofern die Dateien gerade geöffnet sind und bearbeitet werden. Wird LibreOffice normal beendet, so werden diese Pfade wieder gelöscht. Alte Pfade weisen also darauf hin, dass das Programm zu dem Zeitpunkt nicht korrekt beendet wurde. Leider haben die Dateien in diesen Pfaden nicht die ursprünglichen Namen. Unter Linux lassen sich aber Dateitypen auch mit der Endung *.tmp problemlos den Programmtypen zuordnen.

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. Oft wird erst beim Schließen der Datei die gesamte Datenbank in die Datei zurückgeschrieben und gepackt. Lediglich Eingaben, die direkt in die Tabellen erfolgen, werden auch direkt in die Datenbankdatei gesichert. Seit LO 5.1 kann außerdem die Datenbank noch über den Button Speichern gesichert werden. Dies

war vorher nicht der Fall, da der Button nicht den Zugang zu einem Untermenü anbot und deshalb inaktiv war.

Wiederherstellung der Datenbank-Archivdatei

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:

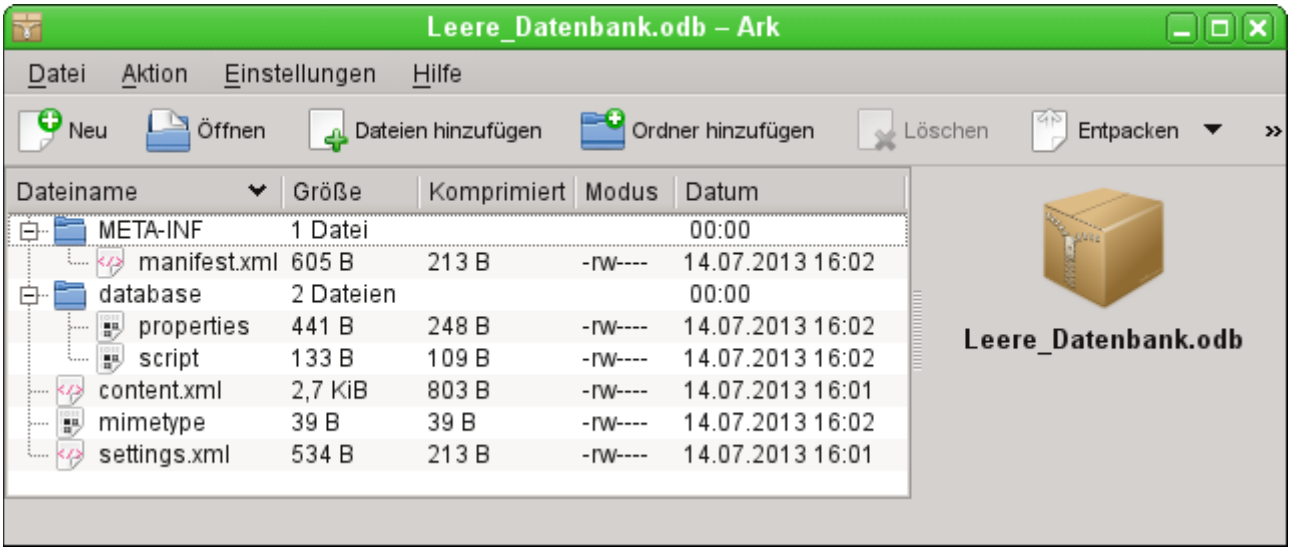


Abbildung 1: Aufbau einer Datenbankdatei ohne Tabellen, Abfragen, Formulare und Berichte

4. Die Datenbankdatei muss ausgepackt werden. Die entscheidenden Informationen für die Daten liegen im Unterverzeichnis «database» in den Dateien «data» und «script».
5. Gegebenenfalls empfiehlt es sich, die Datei «script» einmal anzuschauen und auf 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. 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*

Weitere Informationen zur Datenbank-Archivdatei

Eine Datenbank-Archivdatei enthält im tatsächlichen Gebrauch neben dem grundlegenden Verzeichnis für die Datenbank und dem für das OpenDocument-Format vorgeschriebenen Verzeichnis «META-INF» noch weitere Verzeichnisse, um Formulare und Berichte abzuspeichern. Eine Beschreibung zum grundsätzliche Aufbau des OpenDocument-Formates ist u.a. unter *http://de.wikipedia.org/wiki/OpenDocument* zu finden.

Die folgende Übersicht zeigt eine Datenbank, die Tabellen ein Formular und einen Bericht enthält. Nicht offen sichtbar ist hier, dass auch eine Abfrage zur Datenbank gehört. Solche Abfragen werden nicht in separaten Verzeichnissen gespeichert, sondern sind in der Datei «content.xml» enthalten. Die dafür notwendigen Informationen beschränken sich schließlich auf eine einfache SQL-Formulierung.

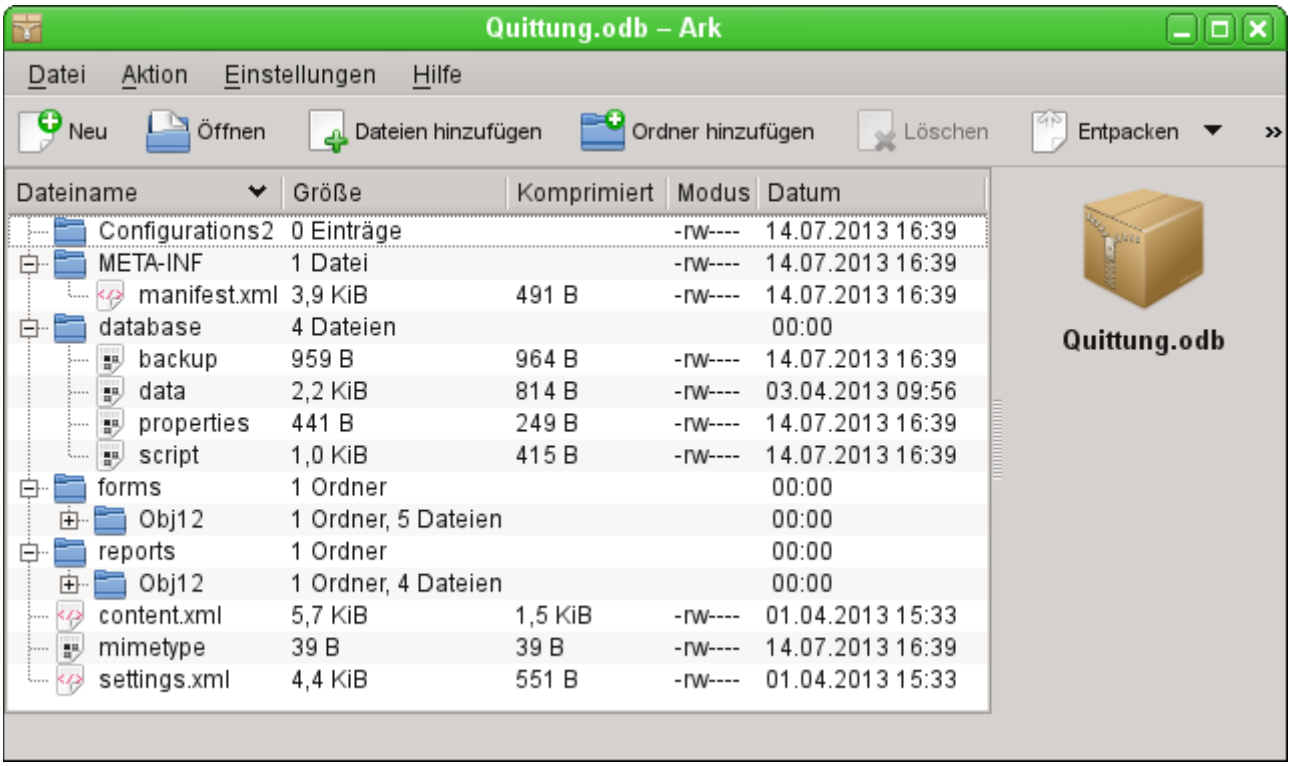


Abbildung 2: Datenbankdatei, die neben der Datenbank auch Informationen zu einem Formular und einem Bericht abgespeichert hat.

Hier einige der Dateien aus der Datenbank-Archivdatei im Überblick:

mimetype

```
004  application/vnd.oasis.opendocument.base
```

Diese kleine Textdatei enthält lediglich den Hinweis, dass es sich bei der Archivdatei um eine Datenbankdatei im OpenDocument-Format handelt.

content.xml einer Datenbank ohne Inhalt

```
005  <?xml version="1.0" encoding="UTF-8"?>
     <office:document-content
     xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
     xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
     xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
     xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
     xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0"
     xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
```

```
xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0"
xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0"
xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0"
xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0"
xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0"
xmlns:ooo="http://openoffice.org/2004/office"
xmlns:ooow="http://openoffice.org/2004/writer"
xmlns:oooc="http://openoffice.org/2004/calc"
xmlns:dom="http://www.w3.org/2001/xml-events"
xmlns:db="urn:oasis:names:tc:opendocument:xmlns:database:1.0"
xmlns:xforms="http://www.w3.org/2002/xforms"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rpt="http://openoffice.org/2005/report"
xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2"
xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:grddl="http://www.w3.org/2003/g/data-view#"
xmlns:tableooo="http://openoffice.org/2009/table"
xmlns:drawooo="http://openoffice.org/2010/draw"
xmlns:calcext="urn:org:documentfoundation:names:experimental:calc:xmlns:calcext:1.0
"
 xmlns:field="urn:openoffice:names:experimental:ooo-ms-interop:xmlns:field:1.0"
xmlns:formx="urn:openoffice:names:experimental:ooxml-odf-interop:xmlns:form:1.0"
xmlns:css3t="http://www.w3.org/TR/css3-text/"
office:version="1.2">
   <office:scripts/>
   <office:font-face-decls/>
   <office:automatic-styles/>
   <office:body>
      <office:database>
         <db:data-source>
            <db:connection-data>
               <db:connection-resource xlink:href="sdbc:embedded:hsqldb"/>
               <db:login db:is-password-required="false"/>
            </db:connection-data>
            <db:driver-settings
               db:system-driver-settings=""
               db:base-dn=""
               db:parameter-name-substitution="false"/>
            <db:application-connection-settings
               db:is-table-name-length-limited="false"
               db:append-table-alias-name="false"
               db:max-row-count="100">
               <db:table-filter>
                  <db:table-include-filter>
                     <db:table-filter-pattern>%</db:table-filter-pattern>
                  </db:table-include-filter>
               </db:table-filter>
            </db:application-connection-settings>
         </db:data-source>
      </office:database>
   </office:body>
</office:document-content>
```

Zu Beginn wird die xml-Version und der verwendete Zeichensatz geklärt. Der gesamte nachfolgende Inhalt wird ohne Absatz direkt in einer Zeile ausgegeben. In der oben erstellten Übersicht wird hoffentlich der Inhalt etwas klarer. Zusammengehörige Elemente werden in «*Tags*» gefasst.

Mit den Anfangsdefinitionen werden durch «xmlns» (XML-Namespace) die Namensräume umschrieben, auf die innerhalb dieser Datei zugegriffen werden kann. Anschließend werden etwas konkretere Angaben zum Inhalt gemacht. Hier ist dann z.B. ersichtlich, dass es sich um eine interne HSQLDB-Datenbank handelt und die Angabe eines Passwortes nicht erforderlich ist.

content.xml einer Datenbank mit Inhalt
Der folgende Inhalt ist nur ein Auszug der content.xml-Datei und soll nur die Struktur klären.

```
006 <office:scripts/>
    <office:font-face-decls>
       <style:font-face style:name="F" svg:font-family=""/>
    </office:font-face-decls>
    <office:automatic-styles>
       <style:style
          style:name="co1"
          style:family="table-column"
          style:data-style-name="N0"/>
       <style:style
          style:name="co2"
          style:family="table-column"
          style:data-style-name="N107"/>
       <style:style style:name="ce1" style:family="table-cell">
          <style:paragraph-properties fo:text-align="start"/>
       </style:style>
       <number:number-style style:name="N0" number:language="de" number:country="DE">
          <number:number number:min-integer-digits="1"/>
       </number:number-style>
       <number:currency-style
          style:name="N107P0"
          style:volatile="true"
          number:language="de"
          number:country="DE">
          <number:number
             number:decimal-places="2"
             number:min-integer-digits="1"
             number:grouping="true"/>
          <number:text> </number:text>
          <number:currency-symbol
             number:language="de"
             number:country="DE">€
          </number:currency-symbol>
       </number:currency-style>
```

Hier wird ein Feld als Währungsfeld festgelegt. Die Anzahl der Dezimalstellen werden genannt, der Abstand zwischen Zahlen und Währungssymbol sowie das Währungssymbol selbst.

```
007    <number:currency-style
          style:name="N107"
          number:language="de"
          number:country="DE">
          <style:text-properties fo:color="#ff0000"/>
          <number:text>-</number:text>
          <number:number
             number:decimal-places="2"
             number:min-integer-digits="1"
             number:grouping="true"/>
          <number:text> </number:text>
          <number:currency-symbol
             number:language="de"
             number:country="DE">€
          </number:currency-symbol>
          <style:map style:condition="value()&gt;=0" style:apply-style-name="N107P0"/>
       </number:currency-style>
```

Im zweiten Abschnitt erfolgt die Festlegung, dass bis zu einem bestimmten Wert die Währung in der Farbe Rot («#ff0000») erscheinen soll.

```
008 </office:automatic-styles>
    <office:body>
       <office:database>
          <db:data-source>
```

Dieser Eintrag entspricht mit allen Unterpunkten dem aus der oben beschriebenen content.xml einer Datenbank-Archivdatei ohne Inhalt.

```
009         </db:data-source>
            <db:forms>
               <db:component
                  db:name="Quittung"
                  xlink:href="forms/Obj12"
                  db:as-template="false"/>
            </db:forms>
```

Die Datenbank-Archivdatei enthält einen Unterordner, in dem die Details zu einem Formular abgespeichert sind. Das Formular ist auf der Benutzeroberfläche mit dem Namen «Quittung» verzeichnet.

```
010         <db:reports>
               <db:component
                  db: name="Quittung"
                  xlink:href="reports/Obj12"
                  db:as-template="false"/>
            </db:reports>
```

Die Datenbank-Archivdatei enthält einen Unterordner, in dem die Details zu einem Bericht abgespeichert sind. Der Bericht ist auf der Benutzeroberfläche ebenfalls mit dem Namen «Quittung» verzeichnet.

```
011         <db:queries>
               <db:query
                  db:name="Verkauf_berechnet"
                  db:command="SELECT &quot;a&quot;.*, ( SELECT &quot;Preis&quot; *
                     &quot;a&quot;.&quot;Anzahl&quot; FROM &quot;Ware&quot; WHERE
                     &quot;ID&quot; = &quot;a&quot;.&quot;Ware_ID&quot; ) AS
                     &quot;Anzahl*Preis&quot; FROM &quot;Verkauf&quot; AS
                     &quot;a&quot;"/>
            </db:queries>
```

Sämtliche Abfragen werden direkt in der content.xml gespeichert. «"» steht dabei für ein doppeltes Anführungszeichen oben «"». Die oben stehende Abfrage ist in diesem Beispiel eigentlich recht umfangreich und besteht aus vielen korrelierenden Unterabfragen. Sie ist hier nur verkürzt wiedergegeben.

```
012         <db:table-representations>
               <db:table-representation db:name="Quittung"/>
               <db:table-representation db:name="Verkauf"/>
               <db:table-representation db:name="Ware">
                  <db:columns>
                     <db:column
                        db:name="ID"
                        db:style-name="co1"
                        db:default-cell-style-name="ce1"/>
                     <db:column
                        db:name="MWSt"
                        db:style-name="co1"
                        db:default-cell-style-name="ce1"/>
                     <db:column
                        db:name="Preis"
                        db:help-message="Angabe als Nettopreis"
                        db:style-name="co2"
                        db:default-cell-style-name="ce1"/>
                     <db:column
                        db:name="Ware"
                        db:style-name="co1"
                        db:default-cell-style-name="ce1"/>
                  </db:columns>
               </db:table-representation>
            </db:table-representations>
```

Wie sollen verschiedene Tabellen von der Ansicht her erscheinen? An dieser Stelle wird das Erscheinungsbild bestimmter Spalten gespeichert; in diesem Beispiel wurden Einstellungen der Tabelle "Ware" mit ihren einzelnen Feldern "ID", "MWSt" usw. abgespeichert. Hier wurde zum einen beim Preis eine Zusatzinformation angegeben. Zum anderen ist eine Formatierung der Angabe vorgenommen worden. Der **style-name** 'co2' entspricht beim Preis dem **style-name**, der zu Beginn der content.xml definiert wurde und mit dem **data-style-name** 'N107' verbunden ist. Der **data-style-name** 'N107' wiederum verweist auf die Formatierung des Feldes als Währungsfeld.

```
013     </office:database>
    </office:body>
```

Grundsätzlich ist in der content.xml der Inhalt der Abfragen und Informationen zum Erscheinungsbild der Tabellen direkt gespeichert. Außerdem ist eine Definition der Verbindung zur Datenbank enthalten. Schließlich kommen noch Verweise auf Formulare und Berichte hinzu.

settings.xml

```
014 <?xml version="1.0" encoding="UTF-8"?>
    <office:document-settings
    xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
    xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
    xmlns:svg="http://www.w3.org/2000/svg"
    xmlns:config="urn:oasis:names:tc:opendocument:xmlns:config:1.0"
    xmlns:ooo="http://openoffice.org/2004/office"
    xmlns:db="urn:oasis:names:tc:opendocument:xmlns:database:1.0"
    office:version="1.2"/>
```

Bei einer Datenbank ohne weiteren Inhalt stehen hier nur die Grunddefinitionen. Mit Inhalt werden hier aber die unterschiedlichsten Einstellungen abgespeichert. Nach dem Start mit der obigen Definition sind folgende Einstellungen in der Beispieldatei abgespeichert:

```
015 <office:settings>
      <config:config-item-set config:name="ooo:view-settings">
        <config:config-item-set config:name="Queries">
          <config:config-item-set config:name="Verkauf_berechnet">
            <config:config-item-set config:name="Tables">
              <config:config-item-set config:name="Table1">
                <config:config-item config:name="WindowName"
                  config:type="string">Verkauf</config:config-item>
                <config:config-item config:name="WindowLeft"
                  config:type="int">153</config:config-item>
                <config:config-item config:name="ShowAll"
                  config:type="boolean">true</config:config-item>
                <config:config-item config:name="WindowTop"
                  config:type="int">17</config:config-item>
                <config:config-item config:name="WindowWidth"
                  config:type="int">120</config:config-item>
                <config:config-item config:name="WindowHeight"
                  config:type="int">120</config:config-item>
                <config:config-item config:name="ComposedName"
                  config:type="string">Verkauf</config:config-item>
                <config:config-item config:name="TableName"
                  config:type="string">Verkauf</config:config-item>
              </config:config-item-set>
            </config:config-item-set>
            <config:config-item config:name="SplitterPosition"
              config:type="int">105</config:config-item>
            <config:config-item config:name="VisibleRows"
              config:type="int">1024</config:config-item>
          </config:config-item-set>
        </config:config-item-set>
      </config:config-item-set>
      <config:config-item-set config:name="ooo:configuration-settings">
        <config:config-item-set config:name="layout-settings">
```

```
            <config:config-item-set config:name="Tables">
               <config:config-item-set config:name="Table1">
                  <config:config-item config:name="WindowName"
                     config:type="string">Verkauf</config:config-item>
                  <config:config-item config:name="WindowLeft"
                     config:type="int">186</config:config-item>
                  <config:config-item config:name="ShowAll"
                     config:type="boolean">false</config:config-item>
                  <config:config-item config:name="WindowTop"
                     config:type="int">17</config:config-item>
                  <config:config-item config:name="WindowWidth"
                     config:type="int">120</config:config-item>
                  <config:config-item config:name="WindowHeight"
                     config:type="int">120</config:config-item>
                  <config:config-item config:name="ComposedName"
                     config:type="string">Verkauf</config:config-item>
                  <config:config-item config:name="TableName"
                     config:type="string">Verkauf</config:config-item>
               </config:config-item-set>
               <config:config-item-set config:name="Table2">
                  ... (identische config:type-Punkte wie "Table1"
                  <config:config-item config:name="TableName"
                     config:type="string">Ware</config:config-item>
               </config:config-item-set>
               <config:config-item-set config:name="Table3">
                  ... (identische config:type-Punkte wie "Table1"
                  <config:config-item config:name="TableName"
                     config:type="string">Quittung</config:config-item>
               </config:config-item-set>
            </config:config-item-set>
         </config:config-item-set>
      </config:config-item-set>
   </office:settings>
```

Die gesamte Übersicht bezieht sich auf verschiedene Ansichten der Fenster für die (eine) Abfrage "Verkauf berechnet" und für die Tabellen "Verkauf", "Ware" und "Quittung". Die letzten beiden wurden hier nur verkürzt wiedergegeben. Würden diese Einstellungen bei einer defekten *.odb-Datei fehlen, so wäre das also nicht weiter von Bedeutung. Sie würden wieder erstellt, wenn die entsprechenden Fenster das nächste Mal geöffnet werden.

META-INF/manifest.xml

```
016 <?xml version="1.0" encoding="UTF-8"?>
   <manifest:manifest
      xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
      <manifest:file-entry
         manifest:full-path="/"
         manifest:media-type="application/vnd.oasis.opendocument.base"/>
      <manifest:file-entry
         manifest:full-path="database/script"
         manifest:media-type=""/>
      <manifest:file-entry
         manifest:full-path="database/properties"
         manifest:media-type=""/>
      <manifest:file-entry
         manifest:full-path="settings.xml"
         manifest:media-type="text/xml"/>
      <manifest:file-entry
         manifest:full-path="content.xml"
         manifest:media-type="text/xml"/>
   </manifest:manifest>
```

Bei dieser Datei im Unterverzeichnis META-INF handelt es sich um ein Inhaltsverzeichnis der gesamten Datenbank-Archivdatei. Da es sich bei der oben gezeigten Datei um die Datei handelt, die in der leeren Datenbank (1) enthalten ist, gibt es hier nur 5 Datei-Einträge («file-entry»). Bei der mit Formular und Bericht versehenen Datenbank-Archivdatei sind die Einträge in der META-INF entsprechend umfangreicher.

database/properties

```
017 #HSQL Database Engine 1.8.0.10
018 #Sun Jul 14 18:02:08 CEST 2013
019 hsqldb.script_format=0
020 runtime.gc_interval=0
021 sql.enforce_strict_size=true
022 hsqldb.cache_size_scale=8
023 readonly=false
024 hsqldb.nio_data_file=false
025 hsqldb.cache_scale=13
026 version=1.8.0
027 hsqldb.default_table_type=cached
028 hsqldb.cache_file_scale=1
029 hsqldb.lock_file=true
030 hsqldb.log_size=10
031 modified=no
032 hsqldb.cache_version=1.7.0
033 hsqldb.original_version=1.8.0
034 hsqldb.compatible_version=1.8.0
```

Die properties-Datei enthält die Grundeinstellungen für die interne HSQL Datenbank. Siehe dazu auch das folgende Kapitel.

database/script

```
035 SET DATABASE COLLATION "German"
036 CREATE SCHEMA PUBLIC AUTHORIZATION DBA
037 CREATE USER SA PASSWORD ""
038 GRANT DBA TO SA
039 SET WRITE_DELAY 60
```

In der script-Datei finden sich die Standardeinstellungen für die Verbindung zur Datenbank, zur benutzten Sprache usw. Hier erscheint auch der später erwähnte Benutzer «SA».

In einer mit Inhalt gefüllten Datenbank werden in dieser Datei die Grundlagen für die Tabellendefinitionen gespeichert:

```
040 SET DATABASE COLLATION "German"
041 CREATE SCHEMA PUBLIC AUTHORIZATION DBA
```

Die Tabellen werden definiert, bevor der Datenbanknutzer definiert wird. Zuerst werden die Tabellen mit ihren Feldern im Cache erstellt.

```
042 CREATE CACHED TABLE "Ware"
043    ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
044    PRIMARY KEY,"Ware" VARCHAR(50),"Preis" DECIMAL(8,2),"MWSt" TINYINT)
045 CREATE CACHED TABLE "Verkauf"
046    ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
047    PRIMARY KEY,"Anzahl" TINYINT,"Ware_ID" INTEGER,"Quittung_ID" INTEGER,
048    CONSTRAINT SYS_FK_59 FOREIGN KEY("Ware_ID") REFERENCES "Ware"("ID"))
049 CREATE CACHED TABLE "Quittung"
050    ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL
051    PRIMARY KEY,"Datum" DATE)
```

Anschließend werden noch Änderungen an den Tabellen vorgenommen, damit die Beziehungen («REFERENCES») stimmig sind

```
052 ALTER TABLE "Verkauf" ADD CONSTRAINT SYS_FK_76 FOREIGN KEY("Quittung_ID")
053    REFERENCES "Quittung"("ID")
054 SET TABLE "Ware" INDEX'608 20'
055 SET TABLE "Verkauf" INDEX'1872 1656 1872 12'
056 SET TABLE "Quittung" INDEX'2232 1'
```

Nach der Einstellung der Position des Indexes in der data-Datei (erscheint nur hier in der script-Datei, wird nie direkt in SQL eingegeben!) werden die automatisch hoch schreibenden Felder der Tabellen («AutoWert») so eingestellt, dass sie die nächsten Werte bei Neueingaben erstellen. So ist z.B. der letzte eingetragene Wert im Feld "ID" der Tabelle "Ware" die Nummer 19. Das automatische Hochschreiben beginnt also mit der Nummer 20.

```
057 ALTER TABLE "Ware" ALTER COLUMN "ID" RESTART WITH 20
```

```
058  ALTER TABLE "Verkauf" ALTER COLUMN "ID" RESTART WITH 12
059  ALTER TABLE "Quittung" ALTER COLUMN "ID" RESTART WITH 1
060  CREATE USER SA PASSWORD ""
061  GRANT DBA TO SA
062  SET WRITE_DELAY 60
```

Behebung von Versionsproblemen

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-Versionen auftauchen. Wird eine externe HSQLDB genutzt, so ist der sicherste Weg der über das hsqldb.jar-Archiv, das mit LO 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 Schwierigkeiten hat, zwischen interner und externer HSQLDB zu unterscheiden und Meldungen von einem Versionskonflikt produziert.

Es muss als externe Datenbank die mitgelieferte hsqldb.jar-Datei genutzt werden. Außerdem muss aus der *.odb-Datei das database-Verzeichnis extrahiert werden. Die Datei properties hat hier einen Eintrag, der in LO 3.3 zu dieser Fehlermeldung führt:

```
010  version=1.8.1
```

steht in Zeile 11.

Diese Zeile ist zu ändern auf

```
010  version=1.8.0
```

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

Weitere Tipps

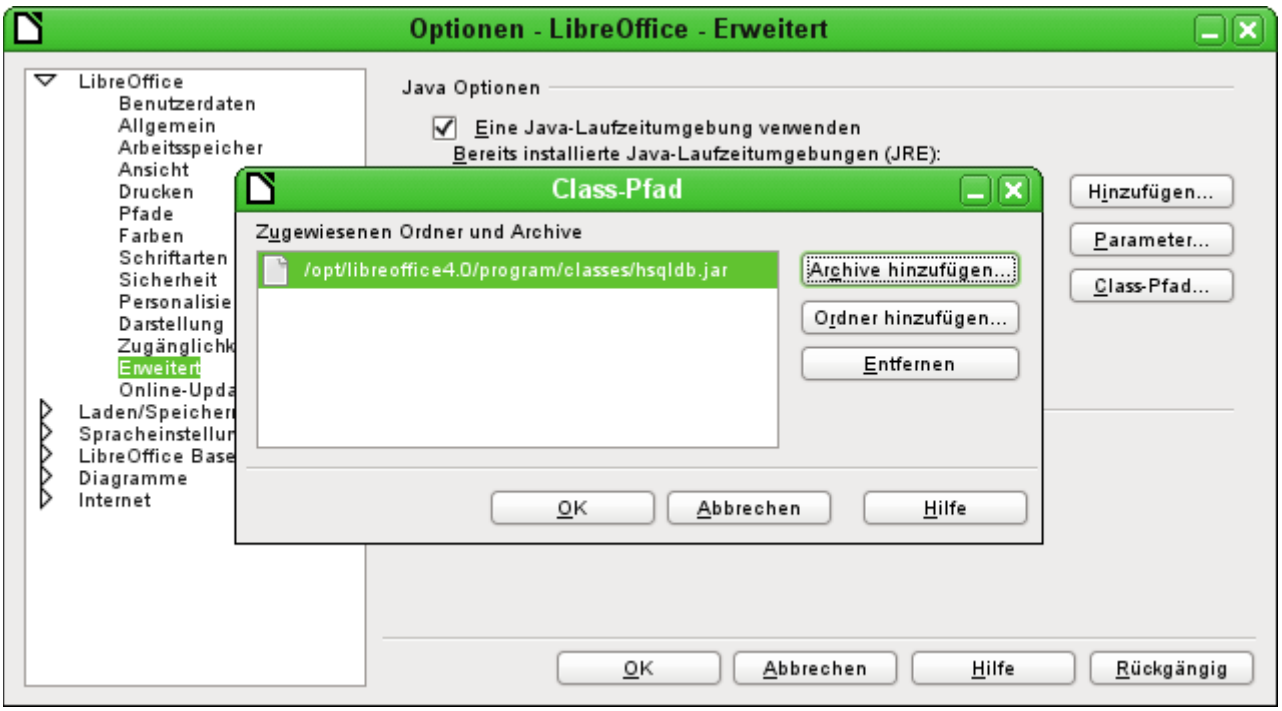
Wenn aus irgendwelchen Gründen wohl die Datenbankdatei geöffnet wird, aber kein Zugang mehr zu den Tabellen existiert, kann direkt über **Extras → SQL** der Befehl **`SHUTDOWN SCRIPT`** eingegeben werden. Anschließend wird die Datenbank geschlossen und neu gestartet. Das Ganze funktioniert aber nicht, wenn bereits ein «Error im Script-file» gemeldet wird.

Die Daten der Datenbank liegen in der *.odb-Datei im Unterverzeichnis «database». Hier gibt es eine Datei «data» und eine Datei «backup». Ist die Datei «data» defekt, so kann sie über die Datei «backup» wiederhergestellt werden. Hierzu muss die im Verzeichnis «database» liegende Datei «properties» bearbeitet werden. Hier gibt es eine Zeile «modified=no». Diese muss umgeschrieben werden zu «modified=yes». Das zeigt dem System an, dass die Datenbank nicht korrekt beendet wurde. Jetzt wird aus der komprimierten «backup»-Datei beim Neustart eine neue «data»-Datei erstellt.

Datenbankverbindung zu einer externen HSQLDB

Die interne HSQLDB unterscheidet sich erst einmal nicht von der externen Variante. Wenn, wie im Folgenden beschrieben, erst einmal nur der Zugriff auf die Datenbank nach außerhalb gelegt werden soll, dann ist keine Serverfunktion erforderlich. Hier reicht schlicht das Archiv, was in LO mitgeliefert wurde. Es liegt im LO-Pfad 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. Diese anderen Versionen sollten allerdings nur genutzt werden, wenn Klarheit darüber besteht, wie Versionsprobleme zwischen interner und externer Variante behoben werden können.



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.

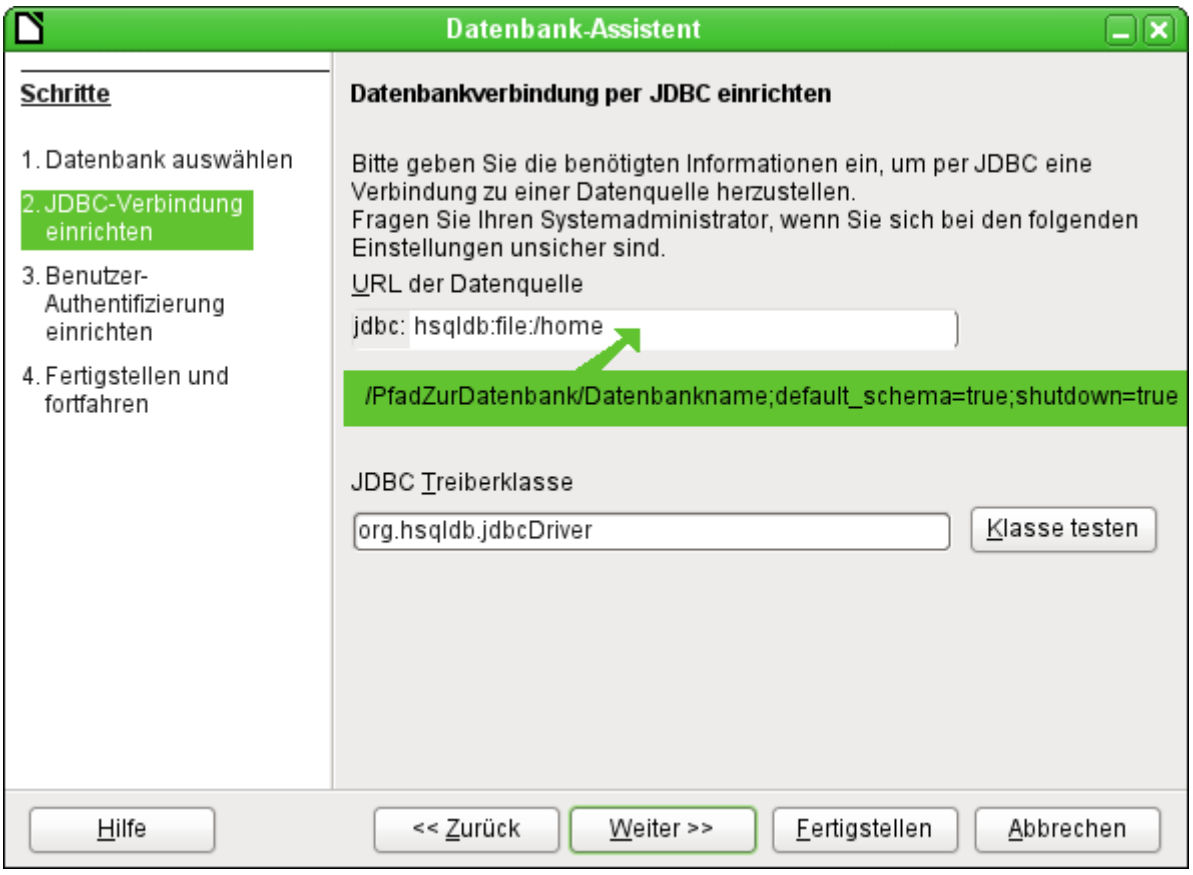
Wichtig, damit auch Daten in die Datenbank über die GUI geschrieben werden können, muss: ergänzend neben dem Datenbanknamen «**;default_schema=true**» stehen. Dies kann noch durch ein «**;shutdown=true**» ergänzt werden, damit die DB nach dem Schließen von LO heruntergefahren wird.

Also:

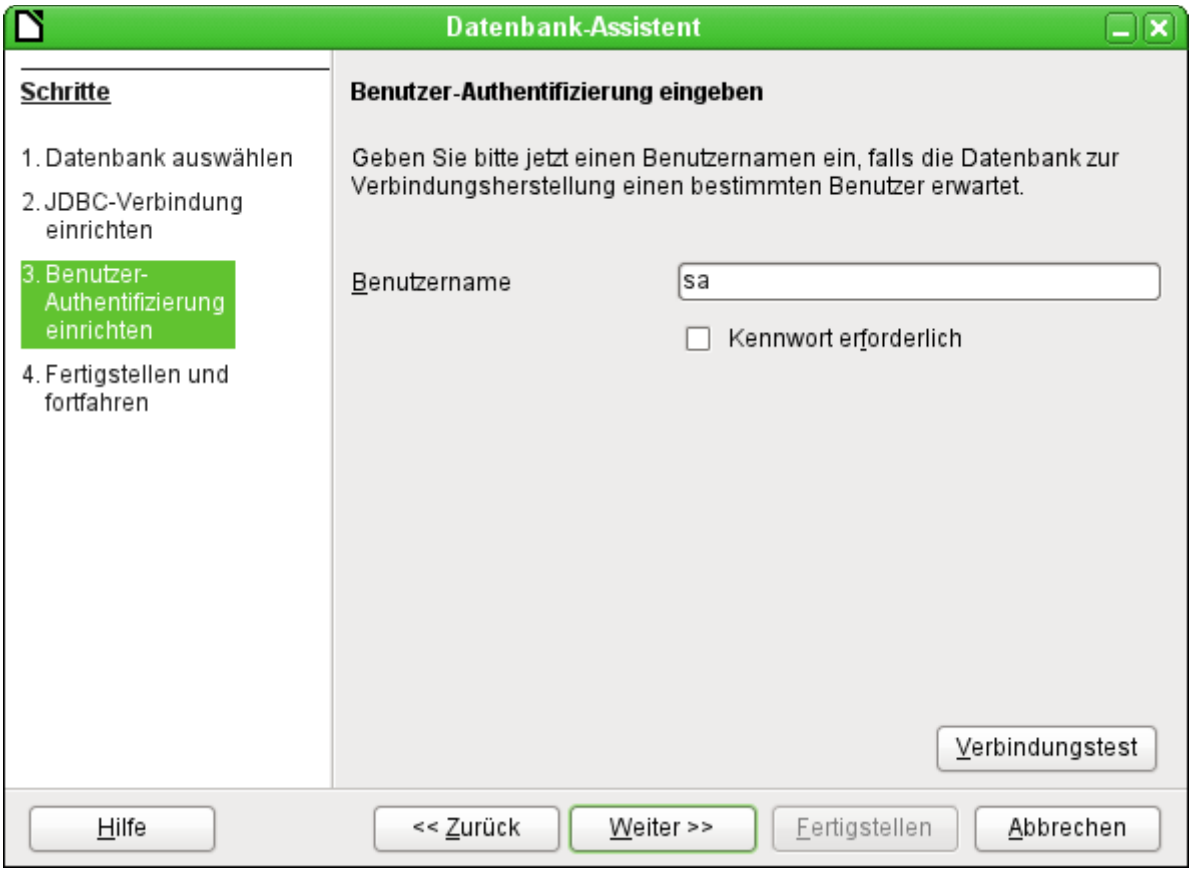
```
001 jdbc:hsqldb:file:/home/PfadZurDatenbank/Datenbankname;default_schema=true;
    shutdown=true
```

In dem Ordner befinden sich die Dateien

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



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



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 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.
Ebenso wenig kann eine externe Datenbank, die einmal mit einer Version der 2er-Reihe bearbeitet wurde, anschließend wieder mit der externen Version 1.8.* bearbeitet werden, die kompatibel zu LibreOffice ist.
Auch kann bei der Nutzung einer anderen Version in der externen Datenbank die Version der internen Datenbank überschrieben werden. Der Aufruf als interne Datenbank ist dann unmöglich.

Parallelinstallation von interner und externer HSQLDB

Die Einbindung der externen Datei hsqldb.jar in den Class-Pfad kann bei unterschiedlichen Versionen dazu führen, dass interne Datenbanken anschließend nicht mehr zu öffnen sind. Base kommt mit den gleichlautenden Treibern nicht zurecht und will die externe Variante auch für die internen Datenbanken nutzen. Das geht beim ersten Öffnen noch gut. Beim zweiten Öffnen kommt dann aber die Meldung, dass die Datenbank nicht mehr zu öffnen ist, da sie mit einer neueren Version der HSQLDB geschrieben wurde.

Dem kann abgeholfen werden, indem für die externen Datenbanken die Datei hsqldb.jar nicht über den Class-Pfad in LO fest eingegeben wird, sondern stattdessen der Class-Pfad für die jeweilige Datenbankdatei über ein Makro geschrieben wird, wie auch *hier* (http://forum.openoffice.org/en/forum/viewtopic.php?f=40&t=61155) zu lesen ist.

```
001 SUB Start
002    Const cPath = "/home/robby/public_html/hsqldb_test/hsqldb.jar"
003    DIM oDataSource AS OBJECT
004    DIM oSettings AS OBJECT
005    DIM sURL AS STRING
006    sURL = ConvertToURL(cPath)
007    oDataSource = ThisComponent.DataSource
008    oSettings = oDataSource.Settings
009    oSettings.JavaDriverClassPath = sURL
010 END SUB
```

Hier wird die Datei «hsqldb.jar» unter Linux in dem o.g. Pfad abgelegt. Anschließend wird dieser Pfad der momentan geöffneten Datenbank als Treiberdatei zugewiesen.

Dieses Makro wird nach dem Öffnen der *.odb-Datei einmal aufgerufen. Es schreibt dann in die in der *.odb-Datei befindlichen Datei content.xml die entsprechende Verbindung zur Java-Klasse:

```
001 <db:data-source-settings>
002    <db:data-source-setting
003       db:data-source-setting-is-list="false"
004       db:data-source-setting-name="JavaDriverClass"
005       db:data-source-setting-type="string">
006    <db:data-source-setting-value>
007       org.hsqldb.jdbcDriver
008    </db:data-source-setting-value>
009    </db:data-source-setting>
010    <db:data-source-setting
011       db:data-source-setting-is-list="false"
012       db:data-source-setting-name="JavaDriverClassPath"
013       db:data-source-setting-type="string">
```

```
014         <db:data-source-setting-value>
015            file:///home/robby/public_html/hsqldb_test/hsqldb.jar
016         </db:data-source-setting-value>
017      </db:data-source-setting>
018   </db:data-source-settings>
```

Letztlich könnte also auch ohne das Makro ein entsprechender Pfad direkt in die content.xml der *.odb-Datei eingetragen werden. Nur ist dieser Weg für den Normalnutzer sicher nicht so komfortabel zu handhaben.

Änderung der Datenbankverbindung zur externen HSQLDB

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 Inhalte zu suchen, die leider in einer durchlaufenden Zeile stehen, es sei denn, man verwendet für diese Arbeit einen speziellen sog. XML-Editor:

```
001 <db:connection-data>
002    <db:connection-resource xlink:href="sdbc:embedded:hsqldb"/>
003    <db:login db:is-password-required="false"/>
004 </db:connection-data>
005 <db:driver-settings
006    db:system-driver-settings=""
007    db:base-dn=""
008    db:parameter-name-substitution="false"/>
```

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

```
001 <db:connection-data>
002    <db:connection-resource
003       xlink:href="jdbc:hsqldb:file:/home/robby/Dokumente/Datenbanken/hsqldb_data/
004       medien;default_schema=true;shutdown=true"/>
005    <db:login db:user-name="sa" db:is-password-required="false"/>
006 </db:connection-data>
007 <db:driver-settings
008    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 dem Rechner wieder.
Es wird darin der Zugang zu 2 Datenbanken geregelt, nämlich der ursprünglichen Standard-Datenbank (die als neue Datenbank genutzt werden kann) als auch der Datenbank, die aus der *.odb-Datei extrahiert wurde.

```
001 # Hsqldb Server cfg file.
002 # See the Advanced Topics chapter of the Hsqldb User Guide.
003
004 server.database.0   file:data/db0
005 server.dbname.0     firstdb
006 server.urlid.0      db0-url
007
008 server.database.1   file:data/medien
009 server.dbname.1     medien
010 server.urlid.1      medien-url
011
012 server.silent       true
013 server.trace        false
014
015 server.port         9001
016 server.no_system_exit         true
```

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

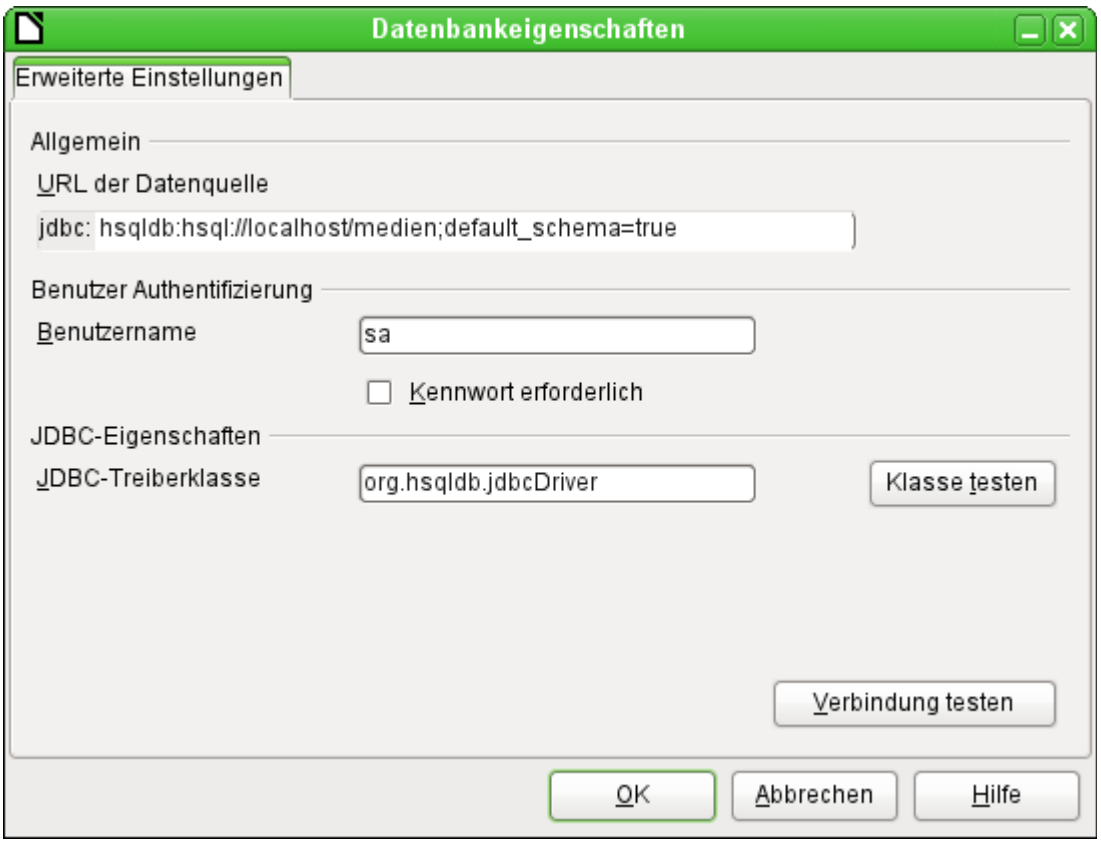
Die beiden Datenbanken werden mit folgenden Zugängen angesprochen:

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

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

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

Nun erfolgt die Serververbindung über LO. Im Hauptfenster von Base wird **Bearbeiten → Datenbank → Eigenschaften** aufgesucht.



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

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

```
001 jdbc:hsqldb:hsql://192.168.0.20/medien;default_schema=true
```

bzw.:

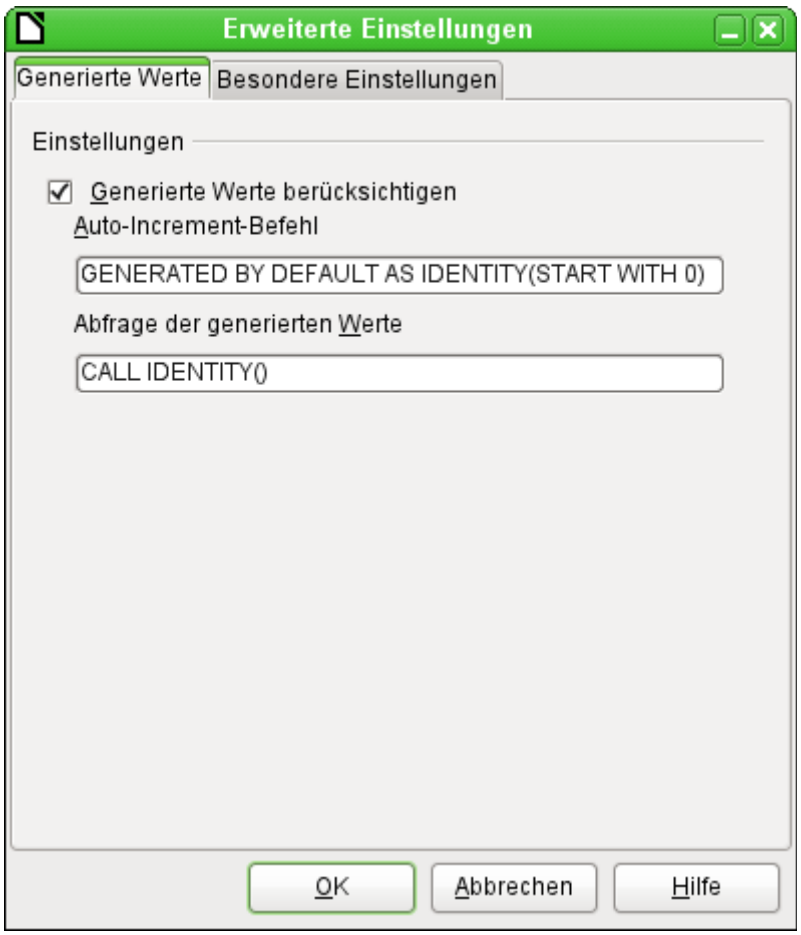
```
001 jdbc:hsqldb:hsql://lin_serv/medien;default_schema=true
```

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

Für die Extraktion und Erzeugung einer Verbindung zur externen neuesten Version der HSQLDB steht auch eine Extension zur Verfügung, die diese Aufgabe erledigen soll: *http://hsqldb.org/web/openoffice.html*

Autoinkrementwerte mit der externen HSQLDB

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



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

Hinweis

In LO ist 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:

```
001 ALTER TABLE "Tabellenname"
002 ALTER COLUMN "ID" INT GENERATED BY DEFAULT AS
    IDENTITY(START WITH 0)
```

... wobei davon ausgegangen wird, dass das Primärschlüsselfeld den Namen "ID" hat.

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

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

Umgang mit der internen Firebird-Datenbank

Die interne Firebird-Datenbank ist bisher teilweise nur als experimentelle Funktion verfügbar. Um solch eine Datenbank zu erstellen, muss **Extras → Optionen → LibreOffice → Erweitert → Optionale (instabile) Einstellungen → Experimentelle Funktionen aktivieren** eingeschaltet sein. Schon dieser Weg zeigt auf, dass solch eine Datenbank noch nicht für den täglichen Gebrauch geeignet ist.

Der experimentelle Modus ist für den Betrieb bereits bestehender Firebird-Datenbanken ab LO 6.1 nicht erforderlich. Mit der Version LO 6.4.3 ist Firebird allerdings erst einmal wieder in den experimentellen Status zurück versetzt worden.

Über den folgenden Link können die wesentlichsten Bugs der internen Firebird-Datenbank zusammen mit LibreOffice eingesehen werden: *Gemeldete Bugs für Firebird in Zusammenhang mit Base* (https://bugs.documentfoundation.org/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&bug_status=NEEDINFO&component=Base&known_name=Firebird_open&list_id=522642&product=LibreOffice&query_based_on=Firebird_open&query_format=advanced&short_desc=Firebird&short_desc_type=allwordssubstr) .

Folgende Besonderheit fällt dem Nutzer gegenüber der HSQLDB direkt auf:

Werden neue Daten eingegeben, so werden diese bis zu LibreOffice-Version 7.6 nicht automatisch in der Datenbankdatei abgespeichert. Der Speicherbutton wird bei jeder Dateneingabe erneut aktiviert. Bei der eingebauten HSQLDB ist die Abspeicherung nach Dateneingaben nicht notwendig.

Funktionserweiterungen und -änderungen in Base im Laufe der LO-Versionen

LO 3.6

Der Befehl **`CHECKPOINT DEFRAG`** wird automatisch beim Schließen der Datenbankdatei ausgeführt. Damit wird die Datenbank auf minimale Größe zusammengeschrumpft. Vorher nahmen bereits gelöschte Datenzeilen weiterhin Platz in der Datenbank ein.

LO 4.1

Im **Abfrageeditor** wurde die Einstellung des **Limits** für die auszugebenden Daten möglich. Diese Einstellung arbeitet so auch mit der GUI und berührt nicht die Editierbarkeit von Abfragen.

Bei **Listenfeldern** ist die Angabe des **gebundenen Feldes** auch als 0 (gleiches Feld wie angezeigt) und als -1 (Position des ausgewählten Feldes) möglich.

Wird mit Makros auf ein **Listenfeld** zugegriffen, so ergibt seit dieser Version der **currentValue** den Wert, der an die Datenbank weitergegeben wird, und nicht unbedingt den Wert, den das Feld in dem Formular anzeigt.

Der **Report-Designer** ist in LO als Erweiterung integriert, so dass er nicht mehr als Erweiterung in den Erweiterungsdialogen erscheint.

LO 4.1.2

Der Zugriff auf **Datumswerte** mit Makros in Formularen wurde geändert. Der Datumswert wird jetzt im Datumsfeld als eine Kombination von Tag, Monat und Jahr wiedergegeben und nicht als ISO-Zahlenwert: oFeld.CurrentValue.Year ist so z.B. die Jahresangabe.

LO 4.2

Access2Base ist standardmäßig als Erweiterung in LO integriert.

Firebird ist als experimentelle Datenbank in LO eingebaut. Die HSQLDB bleibt aber weiterhin die Standarddatenbank. Zu einer externen Firebird-Datei kann eine Verbindung auch direkt erstellt werden.

LO 4.3.1

Aliaszuweisungen in der grafischen Benutzeroberfläche von LO erfolgen ohne den Zusatz **AS**. Die Funktionsweise der internen HSQLDB ist davon nicht berührt. Oracle konnte mit dieser Form der Aliaszuweisung nicht umgehen.

LO 4.4

Parameter in Abfragen: Ein leerer Parameter wurde bisher als leerer Text in Abfragen weiter gegeben, innerhalb von Formularen aber als **NULL** angesehen. Dies wurde geändert. Ein leerer Parameter ist jetzt auch in Abfragen **NULL** und kann also z.B. mit **IFNULL** abgefragt werden.

LO 5.0

Grafisches Kontrollfeld: Das grafische Kontrollfeld konnte bis zu diesem Zeitpunkt nur Bildformate einlesen und darstellen. Mit der Version 5.0 ist das Einlesen bzw. Verknüpfen sämtlicher Dateien möglich. Bei *.pdf-Dateien wird in dem Kontrollfeld die 1. Seite dargestellt. Bei unbekannten Formaten bleibt die Anzeige im Kontrollfeld einfach leer.

LO 5.3

Interne Firebird Datenbank: Die interne Datenbank wechselt von der Version 2.5 auf die Version 3.0.0. Die Version 3.0.0 hat keine Abwärtskompatibilität, so dass vorher erstellte interne Firebird-Datenbanken mit der Version LO 5.3 nicht geöffnet werden können.
Die neue Firebird-Version hat ein besonderes Archiv-Format. Ältere Firebird-Datenbanken können nur über das Entpacken der *.odb-Datei und ein Umwandeln mit einer externen Firebird-Datenbank in das Archivformat überführt werden. Ansonsten bietet es sich an, Tabellen älterer Datenbanken ggf. in eine Calc-Datei zu überführen und von dort in die neue Version zu importieren.
Die Verbindung zu einer externen Firebird-Datei funktioniert ab dieser Version nur noch mit Firebird-3-Datenbankdateien.

Hinweis

Die Umwandlung von der alten Firebird Datenbank zur Version 3.0 kann lt. Firebird-Homepage mit Hilfe einer Firebird 2.5 und einer Firebird 3.0 - Serverinstallation erfolgen (*http://www.firebirdsql.org/file/documentation/release_notes/html/en/3_0/rnfb30-compat-upgrade-secdb.html*)

LO 6.0

Symbolleiste «Formular»: Die Symbolleiste wurde mit Elementen aus dem Bereich «Weitere Symbole» erweitert. Bei den weiteren Symbolen fehlt das Tabellen-Kontrollelement. Dieses ist jetzt über den neuen Menüpunkt **Formular → Tabellen-Steuerelement** abrufbar.

LO 6.1

Interne Firebird Datenbank: Für bestehende interne Firebird-Datenbanken ist der experimentelle Status beendet. Neue interne Firebird Datenbanken können weiterhin nur über den experimentellen Modus erstellt werden. Dies bedeutet auch, dass nach weiteren intensiven Tests in einer der kommenden Versionen von LO die Unterstützung für die interne HSQLDB entfallen wird.
Der Migrationsassistent für die Migration der Daten von interner HSQLDB zu interner Firebird-Datenbank ist in dieser Version nur experimentell nutzbar. Vor der Anwendung der Migration sollte unbedingt eine Datensicherung erfolgen.

LO 6.2

MySQL-Verbindung: Die direkte Verbindung zu MySQL- bzw. MariaDB-Datenbanken, die vorher nur als separate Extension verfügbar war, wird jetzt direkt in LO mit eingebaut. Grundlage der Verbindung ist allerdings der MariaDB-C-Connector, der von der LGPL-Lizenz her zu LibreOffice passt.

LO 6.4

Report Builder: Zellen können jetzt mit einer automatischen Zeilenhöhe versehen werden, so dass nicht mehr ein rotes Dreieck am rechten Rand auf fehlenden Inhalt hinweisen muss.

LO 6.4.3

Firebird: Eingebettete Firebird Datenbanken sind nur noch erstellbar, wenn unter **Extras → Optionen → LibreOffice → Erweitert → Optionale Funktionen → Experimentelle Funktionen aktivieren** eingeschaltet ist. Dies liegt daran, dass zu viele kleine Bugs den Betrieb beeinträchtigen und die interne HSQLDB weiterhin besser mit der GUI zusammen arbeitet.

LO 7.0

Formularbearbeitung: Formulare werden seit diesem Update in der ODF-Version 1.3 erweitert gespeichert. Ein mit LO 7.0 abgespeichertes Formular sollte nicht mit einer älteren Version von LO bearbeitet werden. Das Formular ist nach dem Abspeichern sonst nicht mehr aufrufbar. Dieser Bug wurde in den nachfolgenden Versionen behoben.

LO 7.6

Firebird: Eingaben neuer Daten werden jetzt wie bei der internen HSQLDB direkt abgespeichert. Eine zusätzliche Betätigung des Speicherbuttons ist nicht mehr notwendig.