Fehlermeldungen und Wünsche bitte ausschließlich an UDF@adhoc-data.de

Inhaltsverzeichnis

- F_DQM
- F_SQM
- F_TAB

Vergleichen
- F_EQUALSTRING

Suchen / Ermitteln
- F_FINDWORD
- F_FINDNTHWORD
- F_FINDWORDINDEX
- F_STRINGLENGTH
- F_STRINGLISTITEM

Mathematische Funktionen (31 Funktionen)

Erzeugen
- F_INTRANDOM

Umwandeln
- F_CONVERTFROM33
- F_CONVERTTO33
- F_CONVERTFROMBASE
- F_CONVERTTOBASE
- F_HEXTOINT
- F_INTTOHEX
- F_DEGTORAD
- F_RADTODEG

Formatieren
- F_DOLLARVAL
- F_CONVERTTODOLLAR
- F_FIXEDPOINT
- F_FIXEDPOINTLANG
- F_ROUND
- F_ROUNDFLOAT
- F_TRUNCATE
- F_ZAHLRUNDEN

Berechnen
- F_ABS
- F_DOUBLEABS
- F_INTEGERABS
- F_FACT
- F_FACTORIAL
- F_MODULO
- F_PROZENTE

Vergleichen
- F_EQUALFLOAT
- F_EQUALINTEGER
- F_MAXNUM
- F_MAX
- F_MINNUM
- F_MIN
- F_ISDIVISIBLEBY

Datum-Zeit Funktionen (66 Funktionen)

Funktionen zum Rechnen mit Datum-Uhrzeit

F_ADDYEAR
F_ADDMONTH
F_ADDWEEK
F_ADDDAY
F_ADDHOUR
F_ADDMINUTE
F_ADDSECOND

Funktionen zum Berechnen der Differenz von zwei Datum-Uhrzeit

F_AGEINYEARS
F_AGEINYEARSTHRESHOLD
F_AGEINMONTHS
F_AGEINMONTHSTHRESHOLD
F_AGEINWEEKS
F_AGEINWEEKSTHRESHOLD
F_AGEINDAYS
F_AGEINDAYSTHRESHOLD
F_AGEINHOURS
F_AGEINHOURSTHRESHOLD
F_AGEINMINUTES
F_AGEINMINUTESTHRESHOLD
F_AGEINSECONDS
F_AGEINSECONDSTHRESHOLD
F_YEARSBETWEEN
F_MONTHSBETWEEN
F_WEEKSBETWEEN
F_DAYSBETWEEN
F_HOURSBETWEEN
F_MINUTESBETWEEN
F_SECONDSBETWEEN
F_DAYOFYEAR
F_DAYOFMONTH
F_DAYSOFMONTH
F_LASTDAY
F_DAYOFWEEK
F_DTIME

Funktionen zur (formatierten) Ausgabe von Datum-Uhrzeit

F_CMONTHLONG
F_CMONTHLONGLANG
F_CMONTHSHORT
F_CMONTHSHORTLANG
F_CDOWLONG
F_CDOWLONGLANG
F_CDOWSHORT
F_CDOWSHORTLANG
F_GFORMATD
F_YEAR
F_QUARTER
F_MONTH
F_WEEK
F_HOUR

- F_MINUTE
- F_SECOND
- F_YEAROFYEAR
- F_WEEKOFYEAR
- F_WOY
- F_ENCODEDATE
- F_ENCODETIME
- F_ENCODETIMESTAMP
- F_STRTOTIME
- F_STRIPDATE
- F_STRIPTIME

Funktionen zum Testen (zweier) Datum-Uhrzeit

- F_EQUALDATE
- F_EQUALDATETIME
- F_ISLEAPYEAR
- F_MAXDATE
- F_MINDATE
- F_OSTERDATUM
- F_ZEITDIFFERENZ

BLOb Funktionen (16 Funktionen)

BLOb Konvertierungen

- F_BLOBASPCHAR
- F_STRBLOB
- F_BLOB2EXCEL

BLOb Bearbeitungen

- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBLEFT
- F_BLOBMID
- F_BLOBRIGHT
- F_BLOBREPLACESTRING
- F_BLOBSUBSTR
- F_BLOBLINE

BLOb Berechnungen

- F_BLOBSIZE
- F_BLOBMAXSEGMENTLENGTH
- F_BLOBSEGMENTCOUNT
- F_BLOBLINE_COUNT

BLOb Vergleiche

- F_BLOBCOMPARE

UUID-Funktionen (14 Funktionen)

UUID Erzeugen

- F_UUID1MAC
- F_UUID1RAND
- F_UUID4
- F_UUID1MACCOMPR
- F_UUID1RANDCOMPR
- F_UUID4COMPR

UUID Umwandeln

- F_UUID2UUIDCOMPR

1. Kompatibilität zwischen Windows und Linux um Datenbanken mittels Backup/Restore portieren zu können. Dazu gehören sowohl die gleichen Funktionsnamen wie auch die gleichen entrypoints sowie das die Funktionen die gleichen Ergebnisse liefern.
   Der Modul-Name unter Windwos heißt "FreeAdhocUDF.dll" unter Linux "FreeAdhocUDF.so".
   Der "modul_name" in der UDF-Definition heißt immer "FreeAdhocUDF", Windows sucht dann automatisch eine FreeAdhocUDF.dll, Linux eine FreeAdhocUDF.so.
   Unter Linux ist auf Groß-/Kleinschreibung des modul_name sowie der Name der Lib zu achten.
2. Kompatibilität zwischen der "ursprünglichen" FreeUDFLib (in Delphi, 1998 von Gregory Deatz), der FreeUDFLibC (nach C portierte Delphi-Version, 1999 von Gregory Deatz), der FreeUDFLib von AvERP (in Delphi, enthält einige Erweiterungen) und der GrUDF (in Delphi und Kylix 2004 von Torsten Grundke und Gerd Kroll) dergestalt, dass in Projekte mit einem der vier UDFs diese durch die FreeAdhocUDF ersetzt werden können.
3. Es wurden Korrekturen zu den ursprünglichen UDFs vorgenommen, wo unter Windows was Falsches, unter Linux was Richtiges rauskam. Diese Korrekturen stehen natürlich nicht in der "alten" FreeUDFLib sondern erst in der neuen FreeAdhocUDF zur Verfügung.
4. Es wurden Korrekturen an den ursprünglichen UDFs vorgenommen, wo unter Windows was Anderes als unter Linux rauskam.
   - F_AGEINWEEKS
     bisherige Funktion der FreeUDFLibC macht den Abstand in Tagen, teilt durch 7 und rundet dann auf. Führt bei 20.08.2004 zu 21.08.2004 zu 1 statt wie unter Windows (FreeUDFLib) zu 0. Geänderte Funktion jetzt wie unter Windows.
5. Optimierung des C-Codes, z. T. komplett neu geschrieben.
6. FreeAdhocUDF wurde ursprünglich getestet auf
   - InterBaseSS 6.02 unter Windows XP Professional und Windows 2000 Advanced Server
   - InterBase 7.1 SP2 unter Windows XP Professional und Windows 2000 Advanced Server
   - InterBase 7.1 SP2 unter Mandrake Linux 10.0 / Mandriva Linux 10.1 / Mandriva Linux 2006
   - InterBase 7.5 unter WindowsXP Professional
   - InterBase 7.5 unter Mandriva Linux 2006
   - FireBirdSS 1.5.2 unter Windows XP Professional und Windows 2000 Advanced Server
   - FireBirdSS 1.5.2 unter Mandrake Linux 10.0 / Mandriva Linux 10.1
   - FireBirdSS 1.5.2 unter SuSe Linux 8.1 (benutze FreeAdhocUDF.so_SuSe81_FB15 !)
   - FireBirdSS 1.5.2 unter SuSe EnterpriseServer 8 (benutze FreeAdhocUDF.so_SuSe81_FB15 !)
   - FireBirdSS 1.5.2 unter SuSe Linux 10.0
   - FireBirdSS 1.5.2 unter Ubuntu Server 5.10
   - FireBirdSS 1.5.2 unter Kubuntu 5.10
   - FireBirdSS 2.0.0 RC2 unter Windows 2000 Advanced Server
   - FireBirdSS 2.0.0 RC2 unter Mandriva Linux 2006

   Für neueren Versionen siehe jeweils unter Versonen

Unter Linux sind die so-Dateien event. zum Gebrauch nach "FreeAdhocUDF.so" umzubenennen.

Version (ohne VersionsNr.) vom 23.08.2004
stellte Kompatibilität zwischen Windows und Linux für InterBase her und war kompatibel mit der FreeUDFLib und der FreeUDFLibC. Sie enthielt außerdem einige neue Funktionen.

- F_LOWER
- F_UPPER
- F_MAXNUM
- F_MINNUM
- F_CMONTHLONGLANG
- F_CMONTHSHORTLANG
- F_CDOWLONGLANG
- F_CDOWSHORTLANG
- F_DAYSOFMONTH
- F_DTIME
- F_GFORMATD
- F_TELEFONNR
- F_IF

Version “adhoc 20051016" vom 16.10.2005
stellte Kompatibilität zwischen Windows und Linux für InterBase und FireBird her und ist kompatible zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP und GrUDF. Sie enthält neue Funktionen, die es bisher nicht in einer der vier UDFs gab, die aber nützlich sind.

- F_VERSION
- F_EUROVAL
- F_ROUND
- F_DIGITS
- F_STRIPSTRINGHOLD
- F_FACT
- F_FIXEDPOINTLANG
- F_AGEINYEARS
- F_AGEINYEARSTHRESHOLD
- F_AGEINHOURS
- F_AGEINHOURSTHRESHOLD
- F_AGEINMINUTES
- F_AGEINMINUTESTHRESHOLD
- F_AGEINSECONDS
- F_AGEINSECONDSTHRESHOLD
- F_YEARSBETWEEN
- F_MONTHSBETWEEN
- F_WEEKSBETWEEN
- F_DAYSBETWEEN
- F_HOURSBETWEEN
- F_MINUTESBETWEEN
- F_SECONDSBETWEEN
- F_ADDWEEK
- F_ADDDAY
- F_ADDHOUR
- F_ADDMINUTE
- F_ADDSECOND

- F_ENCODEDATE
- F_ENCODETIME
- F_ENCODETIMESTAMP
- F_LASTDAY
- F_STRRM
- F_STRCOPY

Version “adhoc 20051231" vom 31.12.2005
behebt folgende Fehler:
- F_OSTERDATUM (berechnete den Dienstag nach Ostern statt den Ostersonntag)
- UDF-Definitionsscript (Entrypoint von F_MAX und F_MIN war falsch)
- Anpassung Handbuch

Version “adhoc 20060302" vom 02.03.2006
fügt neu hinzu:
- F_STRINGLISTITEM
- F_LF
- F_STR2EXCEL
- F_BLOB2EXCEL
- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBREPLACESTRING
- korrigiert Fehler in F_ADDYEAR, F_ADDMONTH, F_ADDWEEK, F_ADDDAY, F_ADDHOUR, F_ADDMINUTE, F_ADDSECOND
- erweitert F_FINDWORD und F_FINDNTHWORD um Sonderzeichen (äöü...)
- Anpassung Handbuch

Version “adhoc 20060306" vom 06.03.2006
fügt neu hinzu:
- enthält nun eine Variante zu F_SUBSTR wg. Kompatibilität zur (Delphi-)FreeUDFLib

Version “adhoc 20060516" vom 16.05.2006
- angepaßt auf InterBase 7.5
- stellt Lauffähigkeit ab FireBird 2.0 RC1 her (compiliert mit RC2)
- unter Windows ab FireBird 1.5 nur noch eine FreeAdhocUDF-FireBird-Variante notwendig
- unter Windows ab InterBase 6 nur noch eine FreeAdhocUDF-InterBase-Variante notwendig
- unter Linux für FireBird 1.5 nur noch für SuSe 8.1 eine eigene Variante notwendig
- unter Linux für In terBase nur noch eine FreeAdhocUDF-InterBase-Variante notwendig
- SuSe 8.1 mit InterBase 6 nicht mehr länger unterstützt (getestet)
- korrigiert Fehler in F_AGEINYEARSTHRESHOLD (Parameter vertauscht)
- korrigiert Fehler in F_ADDMONTH
- Fehlerbeseitigung und Anpassung Handbuch

Diese Version wurde getestet auf
- InterBaseSS 6.02 unter Windows XP Professional und Windows 2000 Advanced Server
- InterBase 7.1 SP2 unter Windows XP Professional und Windows 2000 Advanced Server
- InterBase 7.1 SP2 unter Mandrake Linux 10.0 / Mandriva Linux 10.1 / Mandriva Linux 2006
- InterBase 7.5 unter WindowsXP Professional
- InterBase 7.5 unter Mandriva Linux 2006
- FireBirdSS 1.5.2 unter Windows XP Professional und Windows 2000 Advanced Server
- FireBirdSS 1.5.2 unter Mandrake Linux 10.0 / Mandriva Linux 10.1

- FireBirdSS 1.5.2 unter SuSe Linux 8.1 (benutze FreeAdhocUDF.so_SuSe81_FB15 !)
- FireBirdSS 1.5.2 unter SuSe EnterpriseServer 8 (benutze FreeAdhocUDF.so_SuSe81_FB15 !)
- FireBirdSS 1.5.2 unter SuSe Linux 10.0
- FireBirdSS 1.5.2 unter Ubuntu Server 5.10
- FireBirdSS 1.5.2 unter Kubuntu 5.10
- FireBirdSS 2.0.0 RC2 unter Windows 2000 Advanced Server
- FireBirdSS 2.0.0 RC2 unter Mandriva Linux 2006

Version “adhoc 20060919 vom 19.09.2006
Interims-Version - nicht veröffentlicht
Workaround für Fehler (Server-Absturz) für InterBase in
- F_BLOBASPCHAR
- F_BLOBCAT
- F_BLOBCATSTR

Version “adhoc 20060925" vom 25.09.2006
behebt Fehler (auch Server-Absturz) für InterBase und FireBird in:
- F_BLOBASPCHAR
- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBREPLACESTRING

behebt Fehler für Linux
- F_DAYSOFMONTHS

fügt neu hinzu:
- F_BLOBMAXSEGMENTLENGTH
- F_BLOBSEGMENTCOUNT
- F_BLOBLINE
- F_BLOBLINE_COUNT
- F_BLOBCOMPARE
- F_BLOBSUBSTR

Anpassung Handbuch
(neue Funktionen) getestet auf
- InterBase 7.1 SP2 unter Windows 2000 Advanced Server
- InterBase 7.1 SP2 unter Mandriva Linux 2006
- InterBase 7.5 SP1 unter Windows XP Professional
- InterBase 7.5 SP1 unter Mandriva Linux 2006
- FireBird 1.5.2 unter SuSe Linux 8.1
- FireBird 1.5.2 unter SuSe Linux 10.0
- FireBird 1.5.2 unter Mandriva Linux 2006
- FireBird 2.0 RC4 unter Windows XP Professional
- FireBird 2.0 RC4 Mandriva Linux 2006

Version “adhoc 20061020" vom 31.10.2006
fügt neu hinzu:
- F_NBSP
- F_DQM
- F_SQM
- F_TAB
- F_INTRANDOM
- F_UUID1MAC

- F_UUID1RAND
- F_UUID1MACCOMPR
- F_UUID1RANDCOMPR
- F_UUID4
- F_UUID4COMPR
- F_UUID2UUIDCOMPR
- F_UUIDCOMPR2UUID
- F_UUIDCOMPARE
- F_UUID1TIMESTAMP
- F_UUID1COMPRTIMESTAMP
- F_UUID1MACMAC
- F_UUID1COMPRMAC
- F_UUIDVERSION

Anpassung Handbuch

(neue Funktionen) getestet auf

- InterBase 7.1 SP2 unter Windows 2000 Advanced Server
- InterBase 7.1 SP2 unter Mandriva Linux 2006
- InterBase 7.5 SP1 unter Windows XP Professional
- InterBase 7.5 SP1 unter Mandriva Linux 2006
- InterBase 2007 unter Windows XP Professional
- FireBird 1.5.2 unter Windows 2000 Advanced Server
- FireBird 1.5.2 unter SuSe 10.0
- FireBird 1.5.2 unter Mandriva Linux 2006
- FireBird 2.0 RC5 unter Windows XP Professional
- FireBird 2.0 RC5 unter Mandriva Linux 2006

GENERAL SOFTWARE LICENSE AGREEMENT
CAUTION: THE COPYING, MODIFICATION, TRANSLATION OR DISTRIBUTION OF THE OBJECT CODE, PROGRAM, SOFTWARE OR SOURCE CODE IMPLIES ACCEPTANCE OF THE TERMS OF THIS GENERAL SOFTWARE PROGRAM LICENSE AGREEMENT. YOU SHOULD READ CAREFULLY THE FOLLOWING TERMS AND CONDITIONS BEFORE YOU COPY, MODIFY, TRANSLATE OR DISTRIBUTE THE OBJECT CODE, PROGRAM, SOFTWARE OR SOURCE CODE.

1.0 DEFINITIONS

1.1 Licensee - The person who has the privilege to copy, modify, translate or distribute the object code, program, software and source code, subject to the terms and conditions of this General Software License Agreement.

1.2 Object Code - The version of a computer program in machine language, and therefore, ready to be executed by the computer.

1.3 Program - A sequence of instructions for executions by a computer.

1.4 Software - The computer program plus program documentation, if applicable.

1.5 Source Code - The version of a computer program in assembly language or high-level language, and therefore, not ready to be executed by the computer.

1.6 Work - All forms of tangible or intangible property, based whole, in part or derived from the object code, program, software or source code.

1.7 You - The person who has the privilege to copy, modify, translate or distribute the object code, program, software and source code, subject to the terms and conditions of this General Software License Agreement.

2.0 LICENSE

2.1 The copyright holder hereby extends a license to you to use its copyrighted object code, program, software and source code, subject to the terms and conditions of this General Software License Agreement.

2.2 This license is applicable to the object code, program, software and source code distributed under the terms of this General Software License Agreement, any work containing the object code, program, software or source code distributed under the terms of this General Software License Agreement, any modification of the object code, program, software or source code distributed under the terms of this General Software License Agreement, any translation of the object code, program, software or source code distributed under the terms of this General Software License Agreement and any work containing a modification or translation of the object code, program, software or source code distributed pursuant to the terms and conditions of this General Software License Agreement.

2.3 You may copy, modify, translate and distribute the object code, program, software or source code distributed under the terms of this General Software License Agreement, subject to the terms and

conditions of this General Software License Agreement.

2.4 If you copy, modify, translate or distribute the object code, program, software or source code distributed under the terms of this General Software License Agreement, you must publish and make known in a clear and conspicuous manner on each copy, modification, translation or distribution of the object code, program, software or source code that the copy, modification, translation or distribution of the object code, program, software or source code is subject to the terms and conditions of this General Software License Agreement and provide a copy of this General Software License Agreement with each copy, modification, translation or distribution of the object code, program, software or source code.

2.5 If you derive, publish or distribute any work that is based whole or in part on the object code, program, software or source code distributed under the terms of this General Software License Agreement, or any modification or translation thereof, you must publish and make known in a clear and conspicuous manner on each such work that the work is subject to the terms and conditions of this General Software License Agreement and provide a copy of this General Software License Agreement with each work.

2.6 If you copy, modify, translate or distribute the object code, program, software or source code distributed under the terms and conditions of this General Software License Agreement, you must provide clear and conspicuous notice that you have copied, modified, translated or distributed the object code, program, software or source code distributed under the terms of this General Software License Agreement, and indicate the date of each such copy, modification, translation or distribution.

2.7 If you copy, modify, translate or distribute the object code, program, software or source code distributed under the terms of this General Software License Agreement, or publish or distribute any work that is derived, in whole or in part, from any copy, modification, translation or distribution of the object code, program, software or source code distributed under the terms of this General Software License Agreement, you cannot impose any further obligations or restrictions on any third person or entity other than what is contained in this General Software License Agreement.

3.0 NO WARRANTY

3.1 THE OBJECT CODE, PROGRAM, SOFTWARE AND SOURCE CODE ARE PROVIDED "AS IS" WITHOUT ANY WARRANT OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR USE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE OBJECT CODE, PROGRAM, SOFTWARE AND SOURCE CODE IS WITH YOU. SHOULD THE OBJECT CODE, PROGRAM, SOFTWARE AND SOURCE CODE PROVE DEFECTIVE, YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

4.0 LIMITATION OF DAMAGES

4.1 IN NO EVENT WILL THE COPYRIGHT HOLDER OR ANY OTHER PERSON OR ENTITY BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, COMPENSATORY, GENERAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR THE INABILITY TO USE THE OBJECT CODE, PROGRAM, SOFTWARE AND SOURCE CODE, EVEN IF THE COPYRIGHT HOLDER OR ANY OTHER PERSON OR ENTITY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

5.0 MISCELLANEOUS

5.1 The article and paragraph headings appearing in this General Software License Agreement have been asserted for the purpose of convenience and ready reference. They do not purport to, and shall not be deemed to define, limit, or extend the scope or intent of the articles and paragraphs to which they pertain.

5.2 This General Software License Agreement embodies the entire agreement respecting its subject matter. There are no promises, terms, conditions or obligations other than those expressly set forth herein. Unless otherwise expressly set forth herein, this General Software License Agreement supersedes all previous communications, representations, agreements, either verbal or written, warranties, promises, covenants or undertakings. 5.3 This General Software License Agreement shall not be modified, altered, amended or supplemented, except in writing signed by all parties hereto.

5.4 This General Software License Agreement shall be governed by the laws of the State of New Jersey.

Vorbemerkung

Es gibt (u.a.) unterschiedliche Limits in InterBase und FireBird sowie in deren Versionen, sprich jede einzelne Spalte eines SQLs darf nicht mehr als das Spalten-Limit UND die Zeile, also der komplette Datensatz des SQLs nicht mehr als das Zeilen-Limit ergeben.
Zum Zeilen-Limit werden alle Ausgabe-Spalten mit deren DEFINIERTEM Wert zusammengerechnet, also nicht mit dem Wert des FeldINHALTes:
- ein Datenbankfeld mit seinem definiertem Wert - z.B. VARCHAR(100), 2 Bytes für SMALLINT etc.
- eine UDF mit dem definierten Rückgabewert - z.B. 100 Bytes für ... RETURN VARCHAR(100) ...
- eine StoredProcedure dto.

	column limit Spalten-Limit	row limit Zeilen-Limit
InterBase 6.02	32 kB	32 kB
InterBase 7.x	32 kB	64 kB
FireBird 1.x	32 kB	32 kB
FireBird 2.x	32 kB	40 kB

Damit z.B. folgende Konstruktion funktioniert:
select F_REPLACE(F_REPLACE(F_REPLACE(F_REPLACE(‘abcdefg’, ‘a’, ‘x‘), ‘b’, ‘y’), ‘c’, ‘z’), ‘d’, ‘-‘) from TABLE ... muß die DEFINITON für F_REPLACE wie folgt aussehen:

```
DECLARE EXTERNAL FUNCTION F_REPLACE
   CSTRING(8190),
   CSTRING(254),
   CSTRING(254)
   RETURNS CSTRING(8190) FREE_IT
   ENTRY_POINT 'replace' MODULE_NAME 'FreeAdhocUDF';
```

darf der “RETURNS CSTRING(8190)” nicht mehr als 8190 sein, weil in EINER Spalte EINE Funktion 4-fach verschachtelt ist -> 4*8190 = 32760
Soll mein SQL außerdem noch eine Tabellen-Spalte erhalten, also z.B.
Select SPALTE, F_REPLACE(F_REPLACE(F_REPLACE(...
darf bei obiger Definition (8190) das F_REPLACE nur DREI mal verschachtelt auftreten, damit das Limit “32 kB pro Datensatz” nicht überschritten wird, weil bei 4-fach verschachtelt schon 32 kB für diese Spalte das Datensatz-Limit erreicht hätte.
Da es in einem Projekt aber verschiedenste Bedingungen für die einzelnen UDFs gibt, ist es eine Möglichkeit, GLEICHE UDFs mehrfach zu definieren:
- F_REPLACE mit RETURN CSTRING(254)
- F_REPLACE4 mit RETURN CSTRING(4095)
- F_REPLACE8 mit RETURN CSTRING(8190)
- F_BIGREPLACE mit RETURN CSTRING(32760)

Nun hat mal “für alle Gelegenheiten” die richtige Funktion zur Verfügung.

String-Funktionen - Konvertierungen

F_CHARACTER

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input INT
Output CSTRING(2)
Parameter 1 ASCII-Code
Gibt das Zeichen des ASCII-Codes von Parameter 1 aus
TestSQL

```
SELECT 'B' AS ISCORRECT, F_CHARACTER(66) FROM RDB$DATABASE;
SELECT 'ä' AS ISCORRECT, F_CHARACTER(228) FROM RDB$DATABASE;
```

F_CHR

Kompatibilität zu GrUDF
identisch zu F_CHARACTER

F_ENCRYPTMD5

Kompatibilität zu FreeUDFLibC
Input CSTRING(128)
Output CSTRING(33)
Parameter 1 String, der verschlüsselt werden soll.
Verschlüsselt nach dem MD5 Algorithmus.
TestSQL

```
SELECT 'e7d31845480111fdba3316129e166860' AS ISCORRECT, F_ENCRYPTMD5('Pauline') FROM
RDB$DATABASE;
```

F_PROPERCASE

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, der umgewandelt werden soll
Wandelt den String bzgl. Groß-/Kleinschreibung so um, dass jedes Wort mit einem Großbuchstaben beginnt und alle anderen Buchstaben Kleinbuchstaben sind. Wandelt auch Sonderzeichen (nicht nur deutsche ä ö ü). Ein kleines ‘ß’ bleibt natürlich ein kleines ‘ß’.
TestSQL

```
SELECT 'Dies Ist Ein Test Äh' AS ISCORRECT, F_PROPERCASE('dies ist ein test äh') FROM
RDB$DATABASE;
```

F_GSOUNDEX

Funktion von adhoc
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String
Ermittelt einen deutsch-phonetischen String von Parameter 1. Kann z.B. zur Dubletten-Suche dienen.
TestSQL

```
SELECT 'MAYR' AS ISCORRECT, F_GSOUNDEX('Meier'), F_GSOUNDEX('Maier'),
F_GSOUNDEX('Mayer') FROM RDB$DATABASE;
```

F_GENERATESNDXINDEX
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP
Input CSTRING(8190)
Output CSTRING(6)
Parameter 1 String
Ermittelt den (Ur-) Soundex vom String.
Soundex ist ein phonetischer Algorithmus zur Indexierung von Wörtern und Phrasen nach ihrem Klang in der englischen Sprache. Gleichklingende Wörter sollen dabei zu einer identischen Zeichenfolge codiert werden. Berücksichtigt ab der 2. Stelle gezählt nur die ersten 6 Konsonanten bei der Ermittlung, wobei mehrfaches Auftreten nur einmal berücksichtigt wird (im Beispiel sind das l,m,y,w,r,d). Deswegen ist er ungeeignet, um längere Strings zu vergleichen.
TestSQL
SELECT 'H4564' AS ISCORRECT, F_SOUNDEX('Hello my world') FROM RDB$DATABASE;
SELECT 'H4564' AS ISCORRECT, F_SOUNDEX('Hello my world on my earth') FROM RDB$DATABASE;

F_SOUNDEX
Funktion von adhoc
identisch zu F_GENERATESNDXINDEX

F_ANSILOWERCASE
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, der umgewandelt werden soll
Wandelt alle Zeichen in Kleinbuchstaben um, incl. Sonderzeichen
TestSQL
SELECT 'schöner tag' AS ISCORRECT, F_ANSILOWERCASE('SchÖner TAG') FROM RDB$DATABASE;
Anmerkung:
Diese Funktion ist unter FireBird 2 nicht mehr notwendig, da es nun auch LOWER(..) Gibt und die Sonderzeichen korrekt behandelt werden.

F_LOWER
Funktion von adhoc
identisch mit F_ANSILOWERCASE

F_ANSIUPPERCASE
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, der umgewandelt werden soll
Wandelt alle Zeichen in Großbuchstaben um, incl. Sonderzeichen (im Unterschied zu dem SQL-Befehl UPPER(...))
TestSQL
SELECT 'SCHÖNER TAG' AS ISCORRECT, UPPER('Schöner Tag'), F_ANSIUPPERCASE('Schöner Tag') FROM RDB$DATABASE;
Anmerkung:
Diese Funktion ist unter FireBird 2 nicht mehr notwendig, da auch UPPER(..) Sonderzeichen korrekt behandelt.

F_UPPER
Funktion von adhoc
identisch mit ANSIUPPERCASE

F_TELEFONNR
Funktion von adhoc
Input CSTRING(32760), INT
Output CSTRING(32760)
Parameter 1 String, der eine TelefonNr. (oder Ähnliches, aus dem alles bis auf die Zahlen entfernt werden soll) enthält
Parameter 2 definiert die Anzahl der Ziffern am Ende, die durch * ersetzt werden
Entfernt alle nicht-nummerischen- und Leerzeichen aus der Telefon-Nr.
Steht am Anfang ein “49" wird es durch ein “+49" ersetzt.
Beginnt der String mit einem “+”, bleibt dies erhalten.
Anmerkung:
Diese Funktion dient dazu, um in einem String gespeicherte und “wild” formatierte TelefonNr. für TAPI an ein/e Telefon(anlage) zu übergeben (die selber diese “Intelligenz” nicht besitzt)
TestSQL
SELECT '0232653***' AS ISCORRECT, F_TELEFONNR(' (0232) / 6535-35', 3) FROM RDB$DATABASE;
SELECT '+001232653***' AS ISCORRECT, F_TELEFONNR('+001 (232) / 6535-35', 3) FROM RDB$DATABASE;
SELECT '+49232653***' AS ISCORRECT, F_TELEFONNR('+49 (232) / 6535-35', 3) FROM RDB$DATABASE;

F_DIGITS
Kompatibilität zu GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String 1
Entfernt aus dem String 1 alle nicht-numerischen Zeichen (z.B. für TAPI)
TestSQL
SELECT '0232653535' AS ISCORRECT, F_DIGITS(' (0232) / 6535-35') FROM RDB$DATABASE;

F_STR2EXCEL
Funktion von adhoc
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, der für Excel umgewandelt werden soll
Um auch mehrzeilige Texte und Texte, die Doppelhochkommata enthalten, aus längeren Strings nach Excel exportieren zu können, bedarf es einiger Umformungen. Diese Funktion erledigt das:
- fügt am Anfang und Ende des Strings jeweils ein Doppelhochkomma hinzu
- verdoppelt alle im Text selber vorkommenden Doppelhochkommata
- entfernt alle CHR(13) aus dem String
- begrenzt den übergebenen String auf 32760 Zeichen (Grenze von Excel für ein Feld)
TestSQL
SELECT '"1.Zeile ""Paul"" und' || F_LF() || '2.Zeile"' AS ISCORRECT, F_STR2EXCEL('1.Zeile "Paul" und' || F_CRLF() || '2.Zeile') FROM RDB$DATABASE;
Anmerkung:
Da es eigentlich keinen Sinn macht, einen in der Datenbank vorhandenen (sehr) langen Text in eine

Excel-Zelle zu exportieren, dürfte diese Funktion (erst) in Kombination mit F_LEFT bzw. F_RIGHT erst praktikabel sein. Bsp.:
SELECT F_RIGHT(F_STR2EXCEL(STRFeldTAGEBUCH), 1000) FROM ... exportiert z.B. die letzten 1000 Zeichen des Tagebuchs.

F_COLLATEBR
Kompatibilität zu GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, indem die Sonderzeichen umgewandelt werden sollen
Wandelt bestimmte „Umlaute“ zu einen vorgegebenen Zeichen
á, â, ã, à, ä, å, Á, Â, Ã, À, Ä, Å => A
é, ê, è, ë, É, Ê, È, Ë => E
í, î, ì, ï, Í, Î, Ì, Ï => I
ó, ô, õ, ò, ö, Ó, Ô, Õ, Ò, Ö => O
ú, û, ù, ü,Ú, Û, Ù, Ü => U
ç, Ç => C
ñ, Ñ => N
ý, ÿ, Ý, Ÿ => Y
Anmerkung:
im Unterschied zur GrUDF wandelt Ÿ NICHT nach Y um (in Linux nicht vorhanden?)!
TestSQL
```
SELECT 'AAAAAAAAAAAA' AS ISCORRECT, F_COLLATEBR('áâãàäåÁÂÃÀÄÅ')
FROM RDB$DATABASE;
SELECT 'EEEEEEEE' AS ISCORRECT, F_COLLATEBR('éêèëÉÊÈË') FROM RDB$DATABASE;
SELECT 'IIIIIIII' AS ISCORRECT, F_COLLATEBR('íîìïÍÎÌÏ') FROM RDB$DATABASE;
SELECT 'OOOOOOOOOO' AS ISCORRECT, F_COLLATEBR('óôõòöÓÔÕÒÖ')
FROM RDB$DATABASE;
SELECT 'UUUUUUUU' AS ISCORRECT, F_COLLATEBR('úûùüÚÛÙÜ') FROM RDB$DATABASE;
SELECT 'CC' AS ISCORRECT, F_COLLATEBR('çÇ') FROM RDB$DATABASE;
SELECT 'NN' AS ISCORRECT, F_COLLATEBR('ñÑ') FROM RDB$DATABASE;
```

F_COPY
Kompatibilität zu GrUDF
Input CSTRING(8190), INT, INT
Output CSTRING(8190)
Parameter 1 String, aus dem eine Zeichenkette ermittelt werden soll
Parameter 2 Position, an der der zu ermittelnde String beginnt
Parameter 3 Länge des zu ermittelnden Strings
Gibt die Anzahl der Buchstaben (Parameter 3) des eingegebenen Text ab der eingegebenen Buchstabennummer (Parameter 2) wieder. Zählung beginnt für Parameter 2 bei 1
Gleiche Funktion wie F_MID. Zählung beginnt bei F_COPY bei 1, nicht bei 0 wie bei F_MID.
TestSQL
```
SELECT  'tag' AS ISCORRECT, F_COPY('Geburtstagsparty', 8, 3) FROM RDB$DATABASE;
```

F_EUROVAL
Funktion von adhoc
Input DOUBLE
Output CSTRING(254)
Parameter 1 Flieskommazahl
Gibt EUR nach dem eingegebenen Parameter aus.
TestSQL
```
SELECT '15.47 EUR' AS ISCORRECT, F_EUROVAL(15.47) FROM RDB$DATABASE;
```

F_HOLDSTRING
Kompatibilität zu FreeUDFLibC
Input CSTRING(32760), CSTRING(254)
Output CSTRING(32760)
Parameter 1 String 1
Parameter 2 String 2 (Aufflistung aller Zeichen, die nicht entfernt werden sollen)
Entfernt aus dem String alle die Zeichen, die nicht im String 2 angegeben sind.
Die Reihenfolge der Zeichen im String 2 spielt keine Rolle.
Identisch zu F_STRIPSTRINGHOLD
Gegenstück zu F_STRIPSTRING
TestSQL
SELECT 'ieiteietet' AS ISCORRECT, F_HOLDSTRING('Dies ist ein Test Text', 'iet') FROM RDB$DATABASE;
SELECT 'ieiteietet' AS ISCORRECT, F_HOLDSTRING('Dies ist ein Test Text', 'tei') FROM RDB$DATABASE;

F_LEFT
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GruDF
Input CSTRING(8190), INT
Output CSTRING(8190)
Parameter 1 String
Parameter 2 Länge des auszugebenden Strings
Gibt den String beschnitten auf die Anzahl Zeichen von links von Parameter 2 aus.
Zählung beginnt bei 1
TestSQL
SELECT 'Dies i' AS ISCORRECT, F_LEFT('Dies ist ein Test', 6) FROM RDB$DATABASE;

F_LINEWRAP
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input CSTRING(32760), INT, INT
Output CSTRING(32760)
Parameter 1 String
Parameter 2 Startposition
Parameter 3 Spaltenbreite
Gibt alle Wörter des Strings aus, beginnen bei der Startposition, der zusammen nicht länger als die Spaltenbreite sind
Zählung beginnt bei 0.
TestSQL
SELECT 'alle zu einer Geburtstagsparty' AS ISCORRECT, F_LINEWRAP('Wir gehen alle zu einer Geburtstagsparty', 10, 30) FROM RDB$DATABASE;
SELECT 'alle zu einer' AS ISCORRECT, F_LINEWRAP('Wir gehen alle zu einer Geburtstagsparty', 10, 29) FROM RDB$DATABASE;

F_LTRIM
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String, dessen Leerzeichen zu Beginn entfernt werden sollen
Entfernt alle einfachen Leerzeichen zu Beginn des Strings, nicht die geschützten Leerzeichen (mit <ALT> <255> eingegebene)
TestSQL
SELECT 'Dies ist ein Test' AS ISCORRECT, F_LTRIM(' Dies ist ein Test') FROM RDB$DATABASE;

F_LRTRIM / F_BIGLRTRIM
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String, dessen Leerzeichen zu Beginn und am Ende entfernt werden sollen
Entfernt alle einfachen Leerzeichen zu Beginn und am Ende des Strings.
Entfernt nicht die geschützten Leerzeichen (mit <ALT> <255> eingegebene)
TestSQL
SELECT 'Dies ist ein Test' AS ISCORRECT, F_STRINGLENGTH(F_LRTRIM(' Dies ist ein Test ')) AS ANZ_ZEICHEN, F_LRTRIM(' Dies ist ein Test ') FROM RDB$DATABASE;

F_MID
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190), INTEGER, INTEGER
Output CSTRING(8190)
Parameter 1 String, aus dem eine Zeichenkette ermittelt werden soll
Parameter 2 Position, an der der zu ermittelnde String beginnt
Parameter 3 Länge des zu ermittelnden Strings
Gibt die Anzahl der Buchstaben (Parameter 3) des eingegebenen Text ab der eingegebenen Buchstabennummer (Parameter 2) wieder. Zählung beginnt für Parameter 2 bei 0.
S. auch F_COPY
TestSQL
SELECT 'tag' AS ISCORRECT, F_MID('Geburtstagsparty', 7, 3)
FROM RDB$DATABASE;

F_PADLEFT

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(4095), CSTRING(16), INT
Output	CSTRING(4095)
Parameter 1	String1 (der mit den Zeichen aus String 2 auf die Länge von Parameter 3 aufgefüllt werden soll)
Parameter 2	String 2 (mit dem String 1 aufgefüllt werden soll)
Parameter 3	Länge des Strings, bis auf die aufgefüllt werden soll

Füllt den String 1 links vom String mit dem/den Zeichen aus String 2 auf die Gesamtlänge von Zeichen von Parameter 3 auf

Werden in String 2 mehr als ein Zeichen eingegeben, beginnt die Auffüllung mit den Zeichen aus String 2 von rechts und bricht ab, wenn die geforderte Anzahl von Gesamtzeichen erreicht ist (s. 2. TestSQL)

TestSQL

```
SELECT 'XXXDies ist ein Test' AS ISCORRECT, F_PADLEFT('Dies ist ein Test', 'X', 20) FROM RDB$DATABASE;
SELECT 'xXxDies ist ein Test' AS ISCORRECT, F_PADLEFT('Dies ist ein Test', 'Xx', 20) FROM RDB$DATABASE;
```

F_PADRIGHT

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(4095), CSTRING(16), INT
Output	CSTRING(4095)
Parameter 1	String1 (der mit den Zeichen aus String 2 auf die Länge von Parameter 3 aufgefüllt werden soll)
Parameter 2	String 2 (mit dem String 1 aufgefüllt werden soll)
Parameter 3	Länge des Strings, bis auf die aufgefüllt werden soll

Füllt den String 1 rechts vom String mit dem/den Zeichen aus String 2 auf die Gesamtlänge von Zeichen von Parameter 3 auf

Werden in String 2 mehr als ein Zeichen eingegeben, beginnt die Auffüllung mit den Zeichen aus String 2 von links und bricht ab, wenn die geforderte Anzahl von Gesamtzeichen erreicht ist (s. 2. TestSQL)

TestSQL

```
SELECT 'Dies ist ein TestXXX' AS ISCORRECT, F_PADRIGHT('Dies ist ein Test', 'X', 20) FROM RDB$DATABASE;
SELECT 'Dies ist ein TestXxX' AS ISCORRECT, F_PADRIGHT('Dies ist ein Test', 'Xx', 20) FROM RDB$DATABASE;
```

F_REPLACE

Kompatibilität zu FreeUDFLibC

Input	CSTRING(32760), CSTRING(254), CSTRING(254)
Output	CSTRING(32760)
Parameter 1	der String, indem eine Zeichenkette ausgetauscht werden soll
Parameter 2	der auszutauschende String
Parameter 3	der String, der gesetzt werden soll

Ersetzt in einem String alle Zeichenketten aus Parameter 2 durch eine andere Zeichenkette aus Parameter 3

Einfache Version der Funktion F_REPLACESTRING nur ohne Möglichkeit, den String nur 1 Mal und unabhängig von Groß-/Kleinschreibung auszutauschen.

TestSQL

```
SELECT 'Dies ist ein Versuch zwei Versuch drei Versuch vier TEST' AS ISCORRECT,
F_REPLACE('Dies ist ein Test zwei Test drei Test vier TEST', 'Test', 'Versuch') FROM
RDB$DATABASE;
```

F_REPLACESTRING

Kompatibilität zu FreeUDFLib AvERP, GrUDF,

Input	CSTRING(32760), CSTRING(254), CSTRING(254), INT, INT
Output	CSTRING(32760)
Parameter 1	der String, indem eine Zeichenkette ausgetauscht werden soll
Parameter 2	der auszutauschende String
Parameter 3	der String, der gesetzt werden soll
Parameter 4	0 = nur das erste Vorkommen austauschen, 1 = alle Vorkommen austauschen
Parameter 5	0 = Groß-/Kleinschreibung berücksichtigen, 1 = nicht berücksichtigen

Ersetzt in einem String eine/alle Zeichenkette/n aus Parameter 2 durch eine andere Zeichenkette aus Parameter 3 und kann auch Groß-Kleinschreibung berücksichtigen

Ersetzt in einem String eine Zeichenkette durch eine andere

TestSQL

```
SELECT 'Dies ist ein Versuch zwei Test drei Test vier TEST' AS ISCORRECT,
F_REPLACESTRING('Dies ist ein Test zwei Test drei Test vier TEST', 'Test', 'Versuch', 0, 0) FROM
RDB$DATABASE;
SELECT 'Dies ist ein Versuch zwei Test drei Test vier TEST' AS ISCORRECT,
F_REPLACESTRING('Dies ist ein Test zwei Test drei Test vier TEST', 'Test', 'Versuch', 0, 1) FROM
RDB$DATABASE;
SELECT 'Dies ist ein Versuch zwei Versuch drei Versuch vier TEST' AS ISCORRECT,
F_REPLACESTRING('Dies ist ein Test zwei Test drei Test vier TEST', 'Test', 'Versuch', 1, 0) FROM
RDB$DATABASE;
SELECT 'Dies ist ein Versuch zwei Versuch drei Versuch vier Versuch' AS ISCORRECT,
F_REPLACESTRING('Dies ist ein Test zwei Test drei Test vier TEST', 'Test', 'Versuch', 1, 1) FROM
RDB$DATABASE;
```

F_RIGHT
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190), INT
Output CSTRING(8190)
Parameter 1 String
Parameter 2 Anzahl der Zeichen von rechts
Gibt den String beschnitten auf die Anzahl Zeichen gezählt von rechts von Parameter 2 aus, Zählung beginnt bei 1
TestSQL
SELECT 'n Test' AS ISCORRECT, F_RIGHT('Dies ist ein Test', 6) FROM RDB$DATABASE;

F_RTRIM
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String, dessen rechte Leerzeichen entfernt werden sollen
Entfernt alle einfachen Leerzeichen am Ende des Strings, nicht die geschützten Leerzeichen (mit <ALT> <255> eingegebene)
TestSQL
SELECT 'Dies ist ein Test' AS ISCORRECT, F_STRINGLENGTH(F_RTRIM('Dies ist ein Test ')) AS ANZ_ZEICHEN, F_RTRIM('Dies ist ein Test ') FROM RDB$DATABASE;

F_STRIPSTRING
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(32760), CSTRING(254)
Output CSTRING(32760)
Parameter 1 String 1 (aus dem alle Zeichen von String 2 entfernt werden sollen)
Parameter 2 String 2 (Zeichen die entfernt werden sollen)
Entfernt aus dem String 1 alle die Zeichen, die im String 2 angegeben sind, wobei die Reihenfolge der Zeichen im String 2 keine Rolle spielt
Gegenstück zu F_HOLDSTRING
TestSQL
SELECT 'Ds s n Ts Tx' AS ISCORRECT, F_STRIPSTRING('Dies ist ein Test Text', 'iet') FROM RDB$DATABASE;
SELECT 'Ds s n Ts Tx' AS ISCORRECT, F_STRIPSTRING('Dies ist ein Test Text', 'tei') FROM RDB$DATABASE;

F_STRIPSTRINGHOLD

Kompatibilität zu ????, identisch zu F_HOLDSTRING

Input CSTRING(32760), CSTRING(254)

Output CSTRING(32760)

Parameter 1 String 1 (aus dem alle Zeichen von String 2 entfernt werden sollen)

Parameter 2 String 2 (Zeichen die entfernt werden sollen)

Entfernt aus dem String 1 alle die Zeichen, die nicht im String 2 angegeben sind, wobei die Reihenfolge der Zeichen im String 2 keine Rolle spielt

Anders ausgedrückt: Gibt nur die Zeichen aus String 1 aus (in der Reihenfolge von String 1), die in String 2 angegeben sind.

Gegenstück zu F_STRIPSTRING

TestSQL

```
SELECT 'ieiteietet' AS ISCORRECT, F_STRIPSTRINGHOLD('Dies ist ein Test Text', 'iet') FROM RDB$DATABASE;
SELECT 'ieiteietet' AS ISCORRECT, F_STRIPSTRINGHOLD('Dies ist ein Test Text', 'tei') FROM RDB$DATABASE;
```

F_SUBSTR / F_BIGSUBSTR

Kompatibilität zu FreeUDFLibC (FreeUDFLib, FreeUDFLib AvERP, GrUDF s. Anmerkung)

Input CSTRING(8190), CSTRING(1024)

Output INT

Parameter 1 String1 (in dem die Position von Parameter 2 ermittelt werden soll)

Parameter 2 String 2 (dessen Position in Parameter 1 ermittelt werden soll)

Gibt die erste Position im String 1 aus, bei der der String 2 beginnt.

Zählung beginnt bei 0.

Anmerkung:

In der originalen FreeUDFLib (1998 by Gregory Deatz) ist die Reihenfolge der Eingabe-Parameter vertauscht gegenüber den anderen String-Funktionen und vor allem wie in der C-Portierung (1999 by Gregory Deatz) der FreeUDFLibC. Um nun Kompatibilität für beide Varianten herzustellen, gibt es zwei verschiedene “Entrypoints”, die in dem DECLARE-Script von F_SUBSTR verwendet werden können. Für eine Kompatibilität zur (Delphi-)FreeUDFLib, zur FreeUDFLib AvERP und zur GrUDF (die wiederum ihrerseits Kompatible zur Delphi-FreeUDFLib ist) verwendet man den Entrypoint “strsub”, für eine Kompatibilität zur FreeUDFLibC den Entrypoint “substr”.

TestSQL (Version FreeUDFLibC)

```
SELECT 9 AS ISCORRECT, F_SUBSTR('Pauline fährt in Urlaub', 'ähr') FROM RDB$DATABASE;
```

TestSQL (Version FreeUDFLib, GrUDF)

```
SELECT 9 AS ISCORRECT, F_SUBSTR('ähr', 'Pauline fährt in Urlaub') FROM RDB$DATABASE;
```

F_STRCOPY

Kompatibilität zu ????

Input CSTRING(8190), INT, INT

Output CSTRING(8190)

Parameter 1 String, aus dem eine Zeichenkette ermittelt werden soll

Parameter 2 Position, an der der zu ermittelnde String beginnt

Parameter 3 Länge des zu ermittelnden Strings

Gibt die Anzahl der Buchstaben (Parameter 3) des eingegebenen Text ab der eingegebenen Buchstabennummer (Parameter 2) wieder. Zählung beginnt für Parameter 2 bei 1

Gleiche Funktion wie F_MID und F_COPY. Zählung beginnt bei 1.

TestSQL

```
SELECT  'tag' AS ISCORRECT, F_STRCOPY('Geburtstagsparty', 7, 3) FROM RDB$DATABASE;
```

F_STRRM

Kompatibilität zu ????

Input	CSTRING(8190), INT
Output	CSTRING(8190)
Parameter 1	String, aus dem eine Stelle entfernt werden soll
Parameter 2	Stelle im String, die entfernt werden soll (Zählung beginnt bei 0)

Entfernt das Zeichen an der eingegebenen Stelle des eingegebenen Strings,

TestSQL

```
SELECT 'ies ist ein Test' AS ISCORRECT, F_STRRM('Dies ist ein Test', 0) FROM RDB$DATABASE;
```

F_CRLF
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input ohne
Output CSTRING(3)
Parameter 1 ohne -> “()”
Zeilenabschluß und Zeilenumbruch. Erzeugt die Zeichen CHR(13) + CHR(10)
TestSQL
SELECT ' ABC' || F_CRLF() || '123' FROM RDB$DATABASE;
Ergebnis: ‘ABC’ (1.Zeile)
‘123’ (2. Zeile)
SELECT 'erste Zeile' || F_CRLF() || 'zweite Zeile' FROM RDB$DATABASE;

F_LF
Funktion von adhoc
Input ohne
Output CSTRING(2)
Parameter 1 ohne -> “()”
Zeilenumbruch. Erzeugt das Zeichen CHR(10).
Identisch zu F_CHARAFCTER(10).
TestSQL
SELECT 'ABC' || F_LF() || '123' FROM RDB$DATABASE;

F_SPACE
Kompatibilität zu GrUDF
Input INTEGER
Output CSTRING(32760)
Parameter 1 Anzahl der Leerzeichen
Gibt eine Leerzeichenkette (CHR(32)) mit bestimmter angegebener Anzahl zurück.
TestSQL
SELECT F_STRINGLENGTH(F_SPACE(10)) || ' Leerzeichen und ein x' AS ISCORRECT,
F_SPACE(10) || 'x' FROM RDB$DATABASE;

F_STROFCHAR
Kompatibilität zu GrUDF
Input CSTRING(1), INT
Output CSTRING(32760)
Parameter 1 String, der wiederholt werden soll
Parameter 2 Anzahl der Wiederholungen
Gibt einen String mit der angegebenen Anzahl wiederholter Zeichen zurück.
TestSQL
SELECT F_STRINGLENGTH(F_STROFCHAR('A', 10)) || ' mal A' AS ISCORRECT,
F_STROFCHAR('A', 10) FROM RDB$DATABASE;

F_NBSP
Funktion von adhoc
Input ohne
Output CSTRING(2)
Parameter 1 ohne -> “()”
Geschütztes Leerzeichen (non-braking-space). Erzeugt das Zeichen CHR(160).
Identisch zu F_CHARAFCTER(160).
TestSQL
SELECT 'ABC 123' AS ISCORRECT, 'ABC' || F_NBSP() || '123' FROM RDB$DATABASE;

F_DQM
Funktion von adhoc
Input ohne
Output CSTRING(2)
Parameter 1 ohne -> “()”
Doppeltes Anführungszeichen (doube-quote-mark). Erzeugt das Zeichen CHR(34).
Identisch zu F_CHARAFCTER(34).
TestSQL
SELECT 'ABC"123' AS ISCORRECT, 'ABC' || F_DQM() || '123' FROM RDB$DATABASE;

F_SQM
Funktion von adhoc
Input ohne
Output CSTRING(2)
Parameter 1 ohne -> “()”
Einfaches Anführungszeichen (single-quote-mark). Erzeugt das Zeichen CHR(39).
Identisch zu F_CHARAFCTER(39).
TestSQL
SELECT 'ABC<einfaches Hochkoma>123' AS ISCORRECT, 'ABC' || F_SQM() || '123' FROM RDB$DATABASE;

F_TAB
Funktion von adhoc
Input ohne
Output CSTRING(2)
Parameter 1 ohne -> “()”
Tabulator. Erzeugt das Zeichen CHR(9).
Identisch zu F_CHARAFCTER(9).
TestSQL
SELECT 'ABC<TAB>123' AS ISCORRECT, 'ABC' || F_TAB() || '123' FROM RDB$DATABASE;

F_EQUALSTRING

Kompatibilität zu FreeUDFLib AvERP, GrUDF,

Input	CSTRING(8190), CSTRING(8190)
Output	INT
Parameter 1	String 1
Parameter 2	String 2

Überprüft ob String 1 gleich String 2 ist.

Ergebnis 1 = ist gleich, 0 = ist nicht gleich

TestSQL

```
SELECT 1 AS ISCORRECT, F_EQUALSTRING('Pauline ist im Urlaub', 'Pauline ist im Urlaub') FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_EQUALSTRING('Pauline ist im Urlaub', 'Paul ist im Urlaub') FROM RDB$DATABASE;
```

F_FINDWORD

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP

Input CSTRING(32760), INT

Output CSTRING(254)

Parameter 1 String, in dem ein anderer String gesucht werden soll

Parameter 2 Position, an der die Suche im String beginnen soll

Gibt ein Wort oder Teilwort zurück, welches an der entsprechend angegebenen Positon steht.

Positionierung beginnt hier bei 0, d.h. 1.Stelle = 0 und sucht bis zum ersten Leerzeichen.

TestSQL

```
SELECT 'ABC' AS ISCORRECT, F_FINDWORD('ABC 123 45678 9123', 0) FROM
RDB$DATABASE;
SELECT 'BC' AS ISCORRECT, F_FINDWORD('ABC 123 45678 9123', 1) FROM RDB$DATABASE;
SELECT '123' AS ISCORRECT, F_FINDWORD('ABC 123 45678 9123', 3) FROM RDB$DATABASE;
SELECT '123' AS ISCORRECT, F_FINDWORD('ABC 123 45678 9123', 4) FROM RDB$DATABASE;
```

F_FINDNTHWORD

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP

Input CSTRING(32760), INT

Output CSTRING(254)

Parameter 1 String, in dem ein anderer String gesucht werden soll

Parameter 2 String-Nr. , die gesucht werden soll (n-tes Wort)

Gibt das n-Wort wieder. Zählung beginnt bei 0.

TestSQL

```
SELECT 'ABC' AS ISCORRECT, F_FINDNTHWORD('ABC 123 45678 9123', 0)
FROM RDB$DATABASE;
SELECT '123' AS ISCORRECT, F_FINDNTHWORD('ABC 123 45678 9123', 1)
FROM RDB$DATABASE;
```

F_FINDWORDINDEX

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP

Input CSTRING(32760), INT

Output INT

Parameter 1 String

Parameter 2 zu überprüfenden Länge des Strings

Sucht die in Parameter 2 angegebene Stelle im String. Ist diese Stelle im String vorhanden, ist das Ergebnis diese Stelle, ist die Stelle nicht im String vorhanden, ist das Ergebnis -1.

Zählung beginnt bei 0.

TestSQL

```
SELECT  12 AS ISCORRECT, F_FINDWORDINDEX('Geburtstagsparty', 12) FROM
RDB$DATABASE;
SELECT  -1 AS ISCORRECT, F_FINDWORDINDEX('Geburtstagsparty', 16) FROM
RDB$DATABASE;
```

F_STRINGLENGTH / F_BIGSTRINGLENGTH

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input CSTRING(32760)

Output INT

Parameter 1 String, dessen Länge ermittelt werden soll

Ermittelt die Länge eines Strings, berücksichtigt dabei auch einfache Leerzeichen am Ende.

F_STRINGLENGTH und F_BIGSTRINGLENGTH dürfen nicht in einem SQL verwendet werden.

TestSQL

```
SELECT 17 AS ISCORRECT, F_STRINGLENGTH('Dies ist ein Test') FROM RDB$DATABASE;
SELECT 19 AS ISCORRECT, F_STRINGLENGTH('Dies ist ein Test  ') FROM RDB$DATABASE;
```

F_STRINGLISTITEM

Kompatibilität zu GrUDF

Input CSTRING(32760), CSTRING(254)

Output CSTRING(1024)

Parameter 1 Stringliste im Format ”n=..”,”m=..”, u.s.w.

Parameter 2 Wert aus der Liste, der ausgegeben werden soll

Sucht in einer Zeichenkette, die wie eine Stringliste (NAME=Value) aufgebaut ist, nach dem Namen und gibt dann vollständig NAME=Value zurück.

TestSQL

```
SELECT '2=gelb' AS ISCORRECT, F_STRINGLISTITEM('"1=blau","2=gelb","3=grün","4=rot"', '2')
FROM RDB$DATABASE;
```

F_INTRANDOM

Funktion von adhoc

Input	INT, INT
Output	INT
Parameter 1	min. Größe der Zufallszahl
Parameter 2	max. Größe der Zufallszahl

Erzeugt eine Integer-Zufallszahl im Bereich zwischen Parameter 1 und Parameter 2

TestSQL

```
SELECT F_INTRANDOM(100, 10000) FROM RDB$DATABASE;
```

Anmerkung:

Durch Initialiserung des Generators mit der aktuellen Systemzeit und der ProzessID ist es ausgeschlossen, dass zwei Aufrufe innerhalb einer Sekunde die gleiche Zufallszahl liefern.

F_CONVERTFROM33
Kompatibilität zu FreeUDFLibC
Input CSTRING(254)
Output INT
Parameter 1 Zahl im 33-Zahlensystems, die konvertiert werden soll als String
Konvertiert eine Zahl im 33-Zahlensystems ins Dezimal-System
TestSQL
SELECT 1000, F_CONVERTFROM33('WB') FROM RDB$DATABASE;

F_CONVERTTO33
Kompatibilität zu FreeUDFLibC
Input INT
Output CSTRING(254)
Parameter 1 Zahl im Dezimal-System, die ins 33-Zahlensystem konvertiert werden soll
Konvertiert eine Dezimal-Zahl ins 33-Zahlensystem
TestSQL
SELECT 'WB', F_CONVERTTO33(1000) FROM RDB$DATABASE;

F_CONVERTFROMBASE
Input CSTRING(32), INT, CSTRING(8)
Output INT
Parameter 1 Zahl eines Zahlensystems, die konvertiert werden soll als String
Parameter 2 Basis der zu konvertierenden Zahl (z.B. 2 für Binär-System)
Parameter 3 alle Ziffern, die im Zahlensystem gültig sind als String (z.B. ‘01234567' für Oktal-System)
Konvertiert eine Zahl eines beliebigen Zahlensystems ins Dezimal-System
TestSQL
SELECT 3, F_CONVERTFROMBASE('11', 2, '01') FROM RDB$DATABASE;
SELECT 9, F_CONVERTFROMBASE('11', 8, '01234567') FROM RDB$DATABASE;

F_CONVERTTOBASE
Kompatibilität zu FreeUDFLibC
Input INT, INT, CSTRING(254)
Output CSTRING(254)
Parameter 1 Dezimal-Zahl, die konvertiert werden soll
Parameter 2 Basis des Sytem, in das konvertiert werden soll (z.B. 2 für Binär-System)
Parameter 3 alle Ziffern, die im Zahlensystem gültig sind als String (z.B. ‘01234567' für Oktal-System)
Konvertiert eine Dezimal-Zahl in ein beliebiges Zahlensystem
TestSQL
SELECT '11', F_CONVERTTOBASE(3, 2, '01') FROM RDB$DATABASE;

F_HEXTOINT
Kompatibilität zu GrUDF
Input CSTRING(20)
Output INTEGER
Parameter 1 Hexwert
Wandelt einen Hexwert in einen Integerwert um.
TestSQL
SELECT 13 AS ISCORRECT, F_HEXTOINT('000000000d') FROM RDB$DATABASE;
SELECT 13 AS ISCORRECT, F_HEXTOINT('d') FROM RDB$DATABASE;
SELECT 13 AS ISCORRECT, F_HEXTOINT('D') FROM RDB$DATABASE;

F_INTTOHEX
Kompatibilität zu GrUDF
Input INTEGER, INTEGER
Output CSTRING(254)
Parameter 1 umzuwandelnde Ganzzahl
Parameter 2 Anzahl Stellen (links aufgefüllt mit führenden Nullen)
Wandelt einen Intergerwert in einen Hexwert um.
TestSQL
SELECT '000000000d' AS ISCORRECT, F_INTTOHEX(13, 10) FROM RDB$DATABASE;

F_DEGTORAD
Kompatibilität zu GrUDF
Input DOUBLE
Output DOUBLE
Parameter 1 Winkel in (Alt-)Grad
Wandelt einen Winkel von (Alt-)Grad in Radiant (Bogenmaß) um
TestSQL
SELECT '3.1415.. = PHI' AS ISCORRECT, F_DEGTORAD(180) FROM RDB$DATABASE;

F_RADTODEG
Kompatibilität zu GrUDF
Input DOUBLE
Output DOUBLE
Wandelt einen Winkel von Radiant(Bogenmaß) in (Alt-)Grad um
TestSQL
SELECT 80.2140913183152 AS ISCORRECT, F_RADTODEG(1.4) FROM RDB$DATABASE;
IB71Win 80,2140913183153
IB71Lin 80,2140913183152
IB75Win 80,2140913183153
IB75Lin
FB15Win
FB15Lin 80,2140913183152
FB20Win 80,2140913183153
FB20Lin

F_DOLLARVAL
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP
Input DOUBLE
Output CSTRING(32)
Parameter 1 Flieskommawert
Wandelt einen Flieskommawert in einen Dollar-Zahlenwert (gerundet auf 2 Nachkommastellen) um.
TestSQL
SELECT '$15.68' AS ISCORRECT, F_CONVERTTODOLLAR(15.678), F_DOLLARVAL(15.678)
FROM RDB$DATABASE;

F_CONVERTTODOLLAR
Kompatibilität zu FreeUDFLibC
Identisch mit F_DOLLARVAL

F_FIXEDPOINT
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input DOUBLE, INT
Output CSTRING(32)
Parameter 1 zu rundende Flieskommazahl
Parameter 2 Anzahl Stellen, auf die gerundet werden soll
Rundet die Flieskommazahl auf Anzahl der Stellen Parameter 2.
In der String-Ausgabe ist das Dezimal-Trennzeichen ein “.”.
TestSQL
SELECT '12345.5' AS ISCORRECT, F_FIXEDPOINT(12345.46, 1) FROM RDB$DATABASE;

F_FIXEDPOINTLANG
Funktion von adhoc
Input DOUBLE, INT , CSTRING(1), CSTRING(1)
Output CSTRING(32)
Parameter 1 zu rundende Flieskommazahl
Parameter 2 Anzahl Stellen, auf die gerundet werden soll
Parameter 3 Zeichen für Dezimaltrennung
Parameter 4 Zeichen für Tausendertrennung
Rundet die Flieskommazahl auf Anzahl der Stellen von Parameter 2 mit definierbaren Dezimal- und Tausender-Trennzeichen
TestSQL
SELECT '12.345,5' AS ISCORRECT, F_FIXEDPOINTLANG(12345.46, 1, ',', '.') FROM RDB$DATABASE;

F_ROUND
Kompatibilität zu FreeUDFLibC
Input DOUBLE
Output INT
Parameter 1 auf Ganzahl zu rundende Flieskommazahl
Rundet eine Flieskommazahl auf einen ganzahligen Wert.
TestSQL
SELECT 16 AS ISCORRECT, F_ROUND(15.567) FROM RDB$DATABASE;

F_ROUNDFLOAT

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP

Input DOUBLE, DOUBLE

Output DOUBLE

Parameter 1 zu rundende Flieskommazahl

Parameter 2 Pattern, wie Flieskommazahl aussehen soll (z.B. 0.01); rundet auf das nächste Vielfache des 2. Parameters

Rundet eine Flieskommazahl.

TestSQL

```
SELECT 15.55 AS ISCORRECT, F_ROUNDFLOAT(15.567, 0.05) FROM RDB$DATABASE;
SELECT 15.56 AS ISCORRECT, F_ROUNDFLOAT(15.567, 0.02) FROM RDB$DATABASE;
SELECT 15.57 AS ISCORRECT, F_ROUNDFLOAT(15.567, 0.01) FROM RDB$DATABASE;
```

F_TRUNCATE

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input DOUBLE

Output INT

Parameter 1 abzuschneidende Flieskommazahl

Schneidet die Nachkommastellen einer Flieskommazahl ab.

TestSQL

```
SELECT 15 AS ISCORRECT, F_TRUNCATE(15.567) FROM RDB$DATABASE;
```

F_ZAHLRUNDEN

Kompatibilität zu FreeUDFLib AvERP, GrUDF

Input DOUBLE, INT

Output DOUBLE

Parameter 1 zur rundender Wert

Parameter 2 Anzahl der Stellen, auf die gerundet werden soll

Rundet den Fließkomma-Zahlenwert auf die angegebene Anzahl Stellen.

TestSQL

```
SELECT 14.5 AS ISCORRECT, F_ZAHLRUNDEN(14.4935, 1) FROM RDB$DATABASE;
SELECT 14.49 AS ISCORRECT, F_ZAHLRUNDEN(14.4935, 2) FROM RDB$DATABASE;
SELECT 14.494 AS ISCORRECT, F_ZAHLRUNDEN(14.4935, 3) FROM RDB$DATABASE;
SELECT 14.4935 AS ISCORRECT, F_ZAHLRUNDEN(14.4935, 6) FROM RDB$DATABASE;
```

F_ABS
Kompatibilität zu GrUDF
Identisch mit F_DOUBLEABS

F_DOUBLEABS
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input DOUBLE
Output DOUBLE
Parameter 1 zu berechnenden Flieskommazahl
Ermittelt den Absolutwert (positiver Wert) einer Flieskommazahl
SELECT 15.678 AS ISCORRECT, F_DOUBLEABS(15.678) FROM RDB$DATABASE;
SELECT 15.678 AS ISCORRECT, F_DOUBLEABS(-15.678) FROM RDB$DATABASE;

F_INTEGERABS
Kompatibilität zu FreeUDFLibC
Input INT
Output INT
Parameter 1 zu berechnenden Ganzzahl
Ermittelt den Absolutwert (positiver Wert) einer Ganzzahl
SELECT 15 AS ISCORRECT, F_INTEGERABS(15) FROM RDB$DATABASE;
SELECT 15 AS ISCORRECT, F_INTEGERABS(-15) FROM RDB$DATABASE;

F_FACT
Funktion von adhoc
Input INT
Output DOUBLE
Parameter 1 Ganzzahl, von der die Fakultät gebildet werden soll.
Berechnet die Fakultät (das Produkt aller natürlichen Zahlen von 1 bis zum Argument)
TestSQL
SELECT 6.000 AS ISCORRECT, F_FACT(3) FROM RDB$DATABASE;

F_FACTORIAL
Kompatibilität zu GrUDF
Identisch mit F_FACT

F_MODULO
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF,
Input INT, INT
Output INT
Parameter 1 Divident (die Zahl, die geteilt wird)
Parameter 2 Divisor (die Zahl, durch die geteilt wird)
Gibt den Modulo (Rest aus der Division zweier Ganzzahlen) von Parameter 1 dividiert durch Parameter 2 an.
TestSQL
SELECT 2 AS ISCORRECT, F_MODULO(5, 3) FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_MODULO(6, 3) FROM RDB$DATABASE;

F_PROZENTE
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input DOUBLE, DOUBLE
Output DOUBLE
wieviel % des 2. vom 1.
Z.B. VK-Preis zu EK-Preis
Test-SQL
SELECT 14.000 AS ISCORRECT, F_PROZENTE(100.00, 14.0) FROM RDB$DATABASE;

F_EQUALFLOAT
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 Flieskommawert 1
Parameter 2 Flieskommawert 2
Vergleicht zwei Fließkomma-Zahlenwerte auf Gleichheit
Ergebnis 1 = ist gleich, 0 = ist ungleich
TestSQL
```
SELECT 1 AS ISCORRECT, F_EQUALFLOAT(14.00, 14.0) FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_EQUALFLOAT(14.00, 14.01) FROM RDB$DATABASE;
```

F_EQUALINTEGER
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input INT, INT
Output INT
Parameter 1 Ganzzahlwert 1
Parameter 2 Ganzzahlwert 2
Vergleicht zwei Integer-Zahlenwerte auf Gleichheit
Ergebnis 1 = ist gleich, 0 = ist ungleich
Anmerkung:
Wird (irrtümlich) ein Parameter als DOUBLE eingegeben, wird erst auf INT gerundet und dann verglichen!
TestSQL
```
SELECT 1 AS ISCORRECT, F_EQUALINTEGER(14, 14) FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_EQUALINTEGER(14, 15) FROM RDB$DATABASE;
SELECT 1 AS ISCORRECT, F_EQUALINTEGER(14, 14.49) FROM RDB$DATABASE;
```

F_MAXNUM
Funktion von adhoc
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 Flieskommazahl 1
Parameter 2 Flieskommazahl 2
Gibt die Größere der beiden eingegebenen Flieskommazahlen zurück.
TestSQL
```
SELECT 15.346 AS ISCORRECT, F_MAXNUM(15.345, 15.346) FROM RDB$DATABASE;
```

F_MAX
Kompatibilität zu GrUDF
Identisch mit F_MAXNUM

F_MINNUM
Funktion von adhoc
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 Flieskommazahl 1
Parameter 2 Flieskommazahl 2
Gibt die Kleinere der beiden eingegebenen Flieskommazahlen zurück.
TestSQL
SELECT 15.345 AS ISCORRECT, F_MINNUM(15.345, 15.346) FROM RDB$DATABASE;

F_MIN
Kompatibilität zu GrUDF
Identisch mit F_MINNUM

F_ISDIVISIBLEBY
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP
Input INT, INT
Output INT
Parameter 1 Divident (die Zahl, die geteilt wird)
Parameter 2 Divisor (die Zahl, durch die geteilt wird)
Ermittelt, ob die Division eine Ganzzahl ergibt.
Ergebnis ist teilbar durch = 1, ist nicht teilbar durch = 0
TestSQL
SELECT 1 AS ISCORRECT, F_ISDIVISIBLEBY(15, 3) FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_ISDIVISIBLEBY(15, 4) FROM RDB$DATABASE;

Datum-Zeit-Funktionen - Funktionen zum Rechnen mit Datum-Uhrzeit

F_ADDYEAR

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Datum optional Uhrzeit
Parameter 2 Anzahl Jahre zum Aufaddieren
Addiert zum Datum die Anzahl Jahre.
Bei negativem Parameter 2 werden die Jahre subtrahiert.
TestSQL

```
SELECT '01.10.2008 15:03:01' AS ISCORRECT, F_ADDYEAR('01.10.2005 15:03:01', 3)
FROM RDB$DATABASE;
SELECT '01.10.2002 15:03:01' AS ISCORRECT, F_ADDYEAR('01.10.2005 15:03:01', -3)
FROM RDB$DATABASE;
```

F_ADDMONTH

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Datum optional Uhrzeit
Parameter 2 Anzahl Monate zum Aufaddieren
Addiert zum Datum die Anzahl Monate.
Bei negativem Parameter 2 werden die Monate subtrahiert.
TestSQL

```
SELECT '01.03.2006 15:03:01' AS ISCORRECT, F_ADDMONTH('01.10.2005 15:03:01', 5)
FROM RDB$DATABASE;
SELECT '01.07.2005 15:03:01' AS ISCORRECT, F_ADDMONTH('01.10.2005 15:03:01', -3) FROM
RDB$DATABASE;
```

F_ADDWEEK

Funktion von adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Datum optional Uhrzeit
Parameter 2 Anzahl Wochen zum Aufaddieren
Addiert zum Datum die Anzahl Wochen.
Bei negativem Parameter 2 werden die Wochen subtrahiert.
TestSQL

```
SELECT '22.10.2005 15:03:01' AS ISCORRECT, F_ADDWEEK('01.10.2005 15:03:01', 3)
FROM RDB$DATABASE;
SELECT '10.09.2005 15:03:01' AS ISCORRECT, F_ADDWEEK('01.10.2005 15:03:01', -3)
FROM RDB$DATABASE;
```

F_ADDDAY
Funktion von adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Datum optional Uhrzeit
Parameter 2 Anzahl Tage zum Aufaddieren
Addiert zum Datum die Anzahl Tage.
Bei negativem Parameter 2 werden die Tage subtrahiert.
TestSQL

```
SELECT '04.10.2005 15:03:01' AS ISCORRECT, F_ADDDAY('01.10.2005 15:03:01', 3)
FROM RDB$DATABASE;
SELECT '28.09.2005 15:03:01' AS ISCORRECT, F_ADDDAY('01.10.2005 15:03:01', -3)
FROM RDB$DATABASE;
```

F_ADDHOUR
Funktion von adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Datum Uhrzeit
Parameter 2 Anzahl Stunden zum Aufaddieren
Addiert zum Datum Uhrzeit die Anzahl Stunden.
Bei negativem Parameter 2 werden die Stunden subtrahiert.
TestSQL

```
SELECT '01.10.2005 18:03:01' AS ISCORRECT, F_ADDHOUR('01.10.2005 15:03:01', 3)
FROM RDB$DATABASE;
SELECT '01.10.2005 12:03:01' AS ISCORRECT, F_ADDHOUR('01.10.2005 15:03:01', -3)
FROM RDB$DATABASE;
```

F_ADDMINUTE
Funktion von adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Datum Uhrzeit
Parameter 2 Anzahl Minuten zum Aufaddieren
Addiert zum Datum Uhrzeit die Anzahl Minuten.
Bei negativem Parameter 2 werden die Minuten subtrahiert.
TestSQL

```
SELECT '01.10.2005 15:06:01' AS ISCORRECT, F_ADDMINUTE('01.10.2005 15:03:01', 3) FROM
RDB$DATABASE;
SELECT '01.10.2005 15:00:01' AS ISCORRECT, F_ADDMINUTE('01.10.2005 15:03:01', -3) FROM
RDB$DATABASE;
```

F_ADDSECOND

Funktion von adhoc

Input	TIMESTAMP, INT
Output	TIMESTAMP
Parameter 1	Datum Uhrzeit
Parameter 2	Anzahl Sekunden zum Aufaddieren

Addiert zum Datum Uhrzeit die Anzahl Sekunden.

Bei negativem Parameter 2 werden die Sekunden subtrahiert.

TestSQL

```
SELECT '01.10.2005 15:03:04' AS ISCORRECT, F_ADDSECOND('01.10.2005 15:03:01', 3) FROM
RDB$DATABASE;
SELECT '01.10.2005 15:02:58' AS ISCORRECT, F_ADDSECOND('01.10.2005 15:03:01', -3) FROM
RDB$DATABASE;
```

Datum-Zeit-Funktionen - Funktionen zum Berechnen der Differenz von zwei Datum-Uhrzeit

F_AGEINYEARS

Funktion von adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum optional Uhrzeit 1
Parameter 2 (größeres) Datum optional Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Jahren von Datum 1 zu Datum 2.
Aus Kompatibilitätsgründen:
Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!
TestSQL

```
SELECT 3 AS ISCORRECT, F_AGEINYEARS('01.01.2005 15:01:21','01.10.2008 15:01:01') FROM RDB$DATABASE;
SELECT -3 AS ISCORRECT, F_AGEINYEARS('01.01.2005 15:01:21','01.10.2002 15:01:01') FROM RDB$DATABASE;
```

F_AGEINYEARSTHRESHOLD

Funktion von adhoc
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (kleineres) Datum optional Uhrzeit 1
Parameter 2 (größeres) Datum optional Uhrzeit 2
Parameter 3 Minimalwert
Parameter 4 Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5 Maximalwert
Parameter 6 Maximalwert wird verwendet (0 = nein, 1 = ja)
Berechnet die Zeitdifferenz in (ganzzahligen) Jahren von Datum 1 zu Datum 2.
Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben
Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben
Aus Kompatibilitätsgründen:
Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!
TestSQL

```
SELECT 3 AS ISCORRECT, F_AGEINYEARSTHRESHOLD('01.10.2005 15:01:03','01.12.2008 15:03:03', 5, 0, 10, 0) FROM RDB$DATABASE;
SELECT 5 AS ISCORRECT, F_AGEINYEARSTHRESHOLD('01.10.2005 15:01:03','01.12.2008 15:03:03', 5, 1, 10, 0) FROM RDB$DATABASE;
SELECT 3 AS ISCORRECT, F_AGEINYEARSTHRESHOLD('01.10.2005 15:01:03','01.12.2008 15:03:03', 5, 0, 10, 1) FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_AGEINYEARSTHRESHOLD('01.10.2005 15:01:03','01.12.2018 15:03:03', 5, 0, 10, 1) FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_AGEINYEARSTHRESHOLD('01.10.2005 15:01:03','01.12.2018 15:03:03', 5, 1, 10, 1) FROM RDB$DATABASE;
```

F_AGEINMONTHS

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	TIMESTAMP, TIMESTAMP
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2

Berechnet die Zeitdifferenz in (ganzzahligen) Monaten von Datum 1 zu Datum 2.

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 9 AS ISCORRECT, F_AGEINMONTHS('01.01.2005 15:01:21','01.10.2005 15:01:01') FROM RDB$DATABASE;
SELECT -9 AS ISCORRECT, F_AGEINMONTHS('01.10.2005 15:01:21','01.01.2005 15:01:01') FROM RDB$DATABASE;
```

F_AGEINMONTHSTHRESHOLD

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP

Input	TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2
Parameter 3	Minimalwert
Parameter 4	Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5	Maximalwert
Parameter 6	Maximalwert wird verwendet (0 = nein, 1 = ja)

Berechnet die Zeitdifferenz in (ganzzahligen) Monaten von Datum 1 zu Datum 2.

Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben

Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 2 AS ISCORRECT, F_AGEINMONTHSTHRESHOLD('01.10.2005 15:01:03','01.12.2005 15:03:03', 5, 0, 10, 0) FROM RDB$DATABASE;
SELECT 5 AS ISCORRECT, F_AGEINMONTHSTHRESHOLD('01.10.2005 15:01:03','01.12.2005 15:03:03', 5, 1, 10, 0) FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_AGEINMONTHSTHRESHOLD('01.01.2005 15:01:03','01.12.2005 15:03:03', 5, 1, 10, 1) FROM RDB$DATABASE;
SELECT -1 AS ISCORRECT, F_AGEINMONTHSTHRESHOLD('01.01.2006 15:01:03','01.12.2005 15:03:03', 5, 0, 10, 0) FROM RDB$DATABASE;
```

F_AGEINWEEKS

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP

Input	TIMESTAMP, TIMESTAMP
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2

Berechnet die Zeitdifferenz in (ganzzahligen) Wochen von Datum 1 zu Datum 2.

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 20 AS ISCORRECT, F_AGEINWEEKS('01.01.2005 15:01:21','15.05.2005 15:01:21') FROM RDB$DATABASE;
SELECT -33 AS ISCORRECT, F_AGEINWEEKS('01.01.2006 15:01:21','15.05.2005 15:01:21') FROM RDB$DATABASE;
```

F_AGEINWEEKSTHRESHOLD

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP

Input	TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2
Parameter 3	Minimalwert
Parameter 4	Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5	Maximalwert
Parameter 6	Maximalwert wird verwendet (0 = nein, 1 = ja)

Berechnet die Zeitdifferenz in (ganzzahligen) Wochen von Datum 1 (Parameter 1) zu Datum 2 (Parameter 2).

Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben

Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 2 AS ISCORRECT, F_AGEINWEEKSTHRESHOLD('01.01.2005 15:01:21','15.01.2005 15:01:21', 5, 0, 10, 0) FROM RDB$DATABASE;
SELECT 5 AS ISCORRECT, F_AGEINWEEKSTHRESHOLD('01.01.2005 15:01:21',’15.01.2005 15:01:21', 5, 1, 10, 0) FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_AGEINWEEKSTHRESHOLD('01.01.2005 15:01:21','15.05.2005 15:01:21', 5, 0, 10, 1) FROM RDB$DATABASE;
SELECT -33 AS ISCORRECT, F_AGEINWEEKSTHRESHOLD('01.01.2006 15:01:21','15.05.2005 15:01:21', 5, 0, 10, 1) FROM RDB$DATABASE
```

F_AGEINDAYS

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP

Input	TIMESTAMP, TIMESTAMP
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2

Berechnet die Zeitdifferenz in (ganzzahligen) Tagen von Datum 1 zu Datum 2.

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 10 AS ISCORRECT, F_AGEINDAYS('01.10.2005 15:01:03','11.10.2005 15:04:03') FROM RDB$DATABASE;
SELECT -20 AS ISCORRECT, F_AGEINDAYS('01.10.2005 15:01:03','11.09.2005 15:04:03') FROM RDB$DATABASE;
```

F_AGEINDAYSTHRESHOLD

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP

Input	TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2
Parameter 3	Minimalwert
Parameter 4	Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5	Maximalwert
Parameter 6	Maximalwert wird verwendet (0 = nein, 1 = ja)

Berechnet die Zeitdifferenz in (ganzzahligen) Tagen von Datum 1 (Parameter 1) zu Datum 2 (Parameter 2).

Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben

Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 10 AS ISCORRECT, F_AGEINDAYSTHRESHOLD('01.10.2005 15:01:03','11.10.2005 15:01:03', 15, 0, 20, 0) FROM RDB$DATABASE;
SELECT 15 AS ISCORRECT, F_AGEINDAYSTHRESHOLD('01.10.2005 15:01:03','11.10.2005 15:01:03', 15, 1, 20, 0) FROM RDB$DATABASE;
SELECT 20 AS ISCORRECT, F_AGEINDAYSTHRESHOLD('01.10.2005 15:01:03','01.11.2005 15:01:03', 15, 0, 20, 1) FROM RDB$DATABASE;
SELECT 20 AS ISCORRECT, F_AGEINDAYSTHRESHOLD('01.10.2005 15:01:03','01.11.2005 15:01:03', 15, 1, 20, 1) FROM RDB$DATABASE;
SELECT 15 AS ISCORRECT, F_AGEINDAYSTHRESHOLD('01.10.2005 15:01:03','01.09.2005 15:01:03', 15, 1, -20, 1) FROM RDB$DATABASE;
SELECT -20 AS ISCORRECT, F_AGEINDAYSTHRESHOLD('01.10.2005 15:01:03','15.09.2005 15:01:03', 15, 0, -20, 1) FROM RDB$DATABASE;
```

F_AGEINHOURS

Funktion von adhoc

Input	TIMESTAMP, TIMESTAMP
Output	INT
Parameter 1	(kleineres) Datum Uhrzeit 1
Parameter 2	(größeres) Datum Uhrzeit 2

Berechnet die Zeitdifferenz in (ganzzahligen) Stunden von Datum 1 zu Datum 2.

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 3 AS ISCORRECT, F_AGEINHOURS('01.10.2005 15:01:03','01.10.2005 18:01:03') FROM RDB$DATABASE;
```

F_AGEINHOURSTHRESHOLD

Funktion von adhoc

Input	TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output	INT
Parameter 1	(kleineres) Datum Uhrzeit 1
Parameter 2	(größeres) Datum Uhrzeit 2
Parameter 3	Minimalwert
Parameter 4	Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5	Maximalwert
Parameter 6	Maximalwert wird verwendet (0 = nein, 1 = ja)

Berechnet die Zeitdifferenz in (ganzzahligen) Stunden von Datum 1 zu Datum 2.

Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben

Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben

Aus Kompatibilitätsgründen:

Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!

TestSQL

```
SELECT 3 AS ISCORRECT, F_AGEINHOURSTHRESHOLD('01.10.2005 15:01:03','01.10.2005 18:01:03', 5, 0, 10, 0) FROM RDB$DATABASE;
SELECT 5 AS ISCORRECT, F_AGEINHOURSTHRESHOLD('01.10.2005 15:01:03','01.10.2005 18:01:03', 5, 1, 10, 0) FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_AGEINHOURSTHRESHOLD('01.10.2005 15:01:03','02.10.2005 18:01:03', 5, 1, 10, 1) FROM RDB$DATABASE;
```

F_AGEINMINUTES
Funktion von adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum Uhrzeit 1
Parameter 2 (größeres) Datum Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Minuten von Datum 1 zu Datum 2.
Aus Kompatibilitätsgründen:
Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!
TestSQL
SELECT 14 AS ISCORRECT, F_AGEINMINUTES('01.10.2005 15:01:03','01.10.2005 15:15:03') FROM RDB$DATABASE;

F_AGEINMINUTESTHRESHOLD
Funktion von adhoc
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (kleineres) Datum Uhrzeit 1
Parameter 2 (größeres) Datum Uhrzeit 2
Parameter 3 Minimalwert
Parameter 4 Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5 Maximalwert
Parameter 6 Maximalwert wird verwendet (0 = nein, 1 = ja)
Berechnet die Zeitdifferenz in (ganzzahligen) Minuten von Datum 1 zu Datum 2.
Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben
Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben
Aus Kompatibilitätsgründen:
Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!
TestSQL
SELECT 14 AS ISCORRECT, F_AGEINMINUTESTHRESHOLD('01.10.2005 15:01:03','01.10.2005 15:15:03', 5, 0, 10, 0) FROM RDB$DATABASE;

F_AGEINSECONDS
Funktion von adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum Uhrzeit 1
Parameter 2 (größeres) Datum Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Sekunden von Datum 1 zu Datum 2.
Aus Kompatibilitätsgründen:
Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!
TestSQL
SELECT 20 AS ISCORRECT, F_AGEINSECONDS('01.01.2005 15:01:01','01.01.2005 15:01:21') FROM RDB$DATABASE;

F_AGEINSECONDSTHRESHOLD

Funktion von adhoc

Input	TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output	INT
Parameter 1	(kleineres) Datum Uhrzeit 1
Parameter 2	(größeres) Datum Uhrzeit 2
Parameter 3	Minimalwert
Parameter 4	Minimalwert wird verwendet (0 = nein, 1 = ja)
Parameter 5	Maximalwert
Parameter 6	Maximalwert wird verwendet (0 = nein, 1 = ja)

Berechnet die Zeitdifferenz in (ganzzahligen) Sekunden von Datum 1zu Datum 2.
Steht Parameter 4 auf 1, wird bei der Ausgabe mindestens der Minimalwert ausgegeben
Steht Parameter 6 auf 1, wird bei der Ausgabe höchstens der Maximalwert ausgegeben
Aus Kompatibilitätsgründen:
Wenn Datum 2 < als Datum 1 ist das Ergebnis negativ. Eigentlich müßte es 0 sein, denn es gibt kein negatives Alter!
TestSQL

```
SELECT 16 AS ISCORRECT, F_AGEINSECONDSTHRESHOLD('01.01.2005 15:01:03','01.01.2005 15:01:19', 5, 0, 10, 0) FROM RDB$DATABASE;
SELECT 5 AS ISCORRECT, F_AGEINSECONDSTHRESHOLD('01.01.2005 15:01:19','01.01.2005 15:01:21', 5, 1, 10, 0) FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_AGEINSECONDSTHRESHOLD('01.01.2005 15:01:03','01.01.2005 15:01:19', 5, 1, 10, 1) FROM RDB$DATABASE;
```

F_YEARSBETWEEN

Kompatibilität zu GrUDF

Input	TIMESTAMP, TIMESTAMP
Output	INT
Parameter 1	(kleineres) Datum optional Uhrzeit 1
Parameter 2	(größeres) Datum optional Uhrzeit 2

Berechnet die Zeitdifferenz in (ganzzahligen) Jahren von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL

```
SELECT 2 AS ISCORRECT, F_YEARSBETWEEN('01.10.2005 15:01:03','11.10.2007 15:01:03') FROM RDB$DATABASE;
SELECT 2 AS ISCORRECT, F_YEARSBETWEEN('11.10.2007 15:01:03','01.10.2005 15:01:03') FROM RDB$DATABASE;
```

F_MONTHSBETWEEN
Funktion von adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum optional Uhrzeit 1
Parameter 2 (größeres) Datum optional Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Monaten von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL
SELECT 1 AS ISCORRECT, F_MONTHSBETWEEN('01.10.2005 15:01:03','11.11.2005 15:01:03')
FROM RDB$DATABASE;
SELECT 1 AS ISCORRECT, F_MONTHSBETWEEN('11.11.2005 15:01:03','01.10.2005 15:01:03')
FROM RDB$DATABASE;

F_WEEKSBETWEEN
Kompatibilität zu GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum optional Uhrzeit 1
Parameter 2 (größeres) Datum optional Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Wochen von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL
SELECT 2 AS ISCORRECT, F_WEEKSBETWEEN('01.10.2005 15:01:03','11.10.2005 15:01:03')
FROM RDB$DATABASE;
SELECT 2 AS ISCORRECT, F_WEEKSBETWEEN('11.10.2005 15:01:03',’01.10.2005 15:01:03')
FROM RDB$DATABASE;

F_DAYSBETWEEN
Funktion von adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum optional Uhrzeit 1
Parameter 2 (größeres) Datum optional Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Tagen von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL
SELECT 10 AS ISCORRECT, F_DAYSBETWEEN('01.10.2005 15:01:03','11.10.2005 15:01:03') FROM RDB$DATABASE;
SELECT 10 AS ISCORRECT, F_DAYSBETWEEN('11.10.2005 15:01:03','01.10.2005 15:01:03') FROM RDB$DATABASE;

F_HOURSBETWEEN
Kompatibilität zu GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum Uhrzeit 1
Parameter 2 (größeres) Datum Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Stunden von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL
```
SELECT 240 AS ISCORRECT, F_HOURSBETWEEN('01.10.2005 15:01:03','11.10.2005 15:04:03')
FROM RDB$DATABASE;
SELECT 240 AS ISCORRECT, F_HOURSBETWEEN('11.10.2005 15:04:03','01.10.2005 15:01:03')
FROM RDB$DATABASE;
```

F_MINUTESBETWEEN
Kompatibilität zu GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum Uhrzeit 1
Parameter 2 (größeres) Datum Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Minuten von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL
```
SELECT 3 AS ISCORRECT, F_MINUTESBETWEEN('01.10.2005 15:01:03','01.10.2005 15:04:03')
FROM RDB$DATABASE;
SELECT 3 AS ISCORRECT, F_MINUTESBETWEEN('01.10.2005 15:04:03', '01.10.2005 15:01:03')
FROM RDB$DATABASE;
```

F_SECONDSBETWEEN
Kompatibilität zu GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (kleineres) Datum Uhrzeit 1
Parameter 2 (größeres) Datum Uhrzeit 2
Berechnet die Zeitdifferenz in (ganzzahligen) Sekunden von Datum 1 zu Datum 2.
Ergibt immer positive Zahlen.
TestSQL
```
SELECT 180 AS ISCORRECT, F_SECONDSBETWEEN('01.10.2005 15:01:03','01.10.2005 15:04:03')
FROM RDB$DATABASE;
SELECT 180 AS ISCORRECT, F_SECONDSBETWEEN('01.10.2005 15:04:03','01.10.2005 15:01:03')
FROM RDB$DATABASE;
```

F_DAYOFYEAR
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Gibt die Anzahl der Tage des Jahres (Tag Nr. x des Jahres) bis zum gewählten Datum aus.
TestSQL
```
SELECT 235 AS ISCORRECT, F_DAYOFYEAR('22.08.2004') FROM RDB$DATABASE;
```

F_DAYOFMONTH
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Gibt den Tag des Monats aus.
TestSQL
SELECT 23 AS ISCORRECT, F_DAYOFMONTH('23.08.2004') FROM RDB$DATABASE;

F_DAYSOFMONTH
Funktion von adhoc
Input INTEGER, INTEGER
Output INTEGER
Parameter 1 Monat
Parameter 2 Jahr
Gibt die Anzahl der Tage des gewählten Monats für das gewählte Jahr aus.
TestSQL
SELECT 29 AS ISCORRECT, F_DAYSOFMONTH(2, 2004) FROM RDB$DATABASE;

F_LASTDAY
Kompatibilität zu GrUDF
(Fast) identisch zu F_DAYSOFMONTH, nur Parameter vertauscht (erst Jahr, dann Monat)
Input INT, INT
Output INTEGER
Parameter 1 Jahr
Parameter 2 Monat
Gibt den letzten Tag des gewählten Monats aus dem gewählten Jahr aus.
SELECT 29 AS ISCORRECT, F_LASTDAY(2004, 2) FROM RDB$DATABASE;

F_DAYOFWEEK
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Gibt den Tag der Woche als Zahlenwert des gewählten Datums aus.
Woche beginnt mit dem Sonntag (= 1)
TestSQL
SELECT 1 AS ISCORRECT, F_DAYOFWEEK('22.08.2004') FROM RDB$DATABASE;

F_DTIME
Funktion von adhoc
Input TIMESTAMP
Output INT
Parameter 1 Datum optional Uhrzeit
Ermittelt die Anzahl der Tage des gewählten Datums seit dem 31.12.1899.
Die Zählung beginnt bei 0.
TestSQL
SELECT 2 AS ISCORRECT, F_DTIME('03.01.1900') FROM RDB$DATABASE;

Datum-Zeit-Funktionen - Funktionen zur (formatierten) Ausgabe von Datum Uhrzeit

F_CMONTHLONG

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(4)
Parameter 1 Datum optional Uhrzeit
Gibt den Monat in Englisch aus.
TestSQL

```
SELECT 'August' AS ISCORRECT, F_CMONTHLONG('23.08.2004') FROM RDB$DATABASE;
```

F_CMONTHLONGLANG

Funktion von adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 Datum optional Uhrzeit
Parameter 2 Sprachkennzeichen für die Ausgabe
de = Deutsch, uk = Englisch, fr = Französich, es = Spanisch, it = Italienisch
Gibt den Monat in der gewählten Sprache aus.
TestSQL

```
SELECT 'Août' AS ISCORRECT, F_CMONTHLONGLANG('23.08.2004', 'fr') FROM
RDB$DATABASE;
```

F_CMONTHSHORT

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(4)
Parameter 1 Datum optional Uhrzeit
Gibt den Monat abgekürzt in Englisch aus.
TestSQL

```
SELECT 'Aug' AS ISCORRECT, F_CMONTHSHORT('23.08.2004') FROM RDB$DATABASE;
```

F_CMONTHSHORTLANG

Funktion von adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 Datum optional Uhrzeit
Parameter 2 Sprachkennzeichen für die Ausgabe
de = Deutsch, uk = Englisch, fr = Französich, es = Spanisch, it = Italienisch
Gibt den Monat abgekürzt in der gewählten Sprache aus.
TestSQL

```
SELECT 'Aoû' AS ISCORRECT, F_CMONTHSHORTLANG('23.08.2004', 'fr') FROM
RDB$DATABASE;
```

F_CDOWLONG
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(16)
Parameter 1 Datum optional Uhrzeit
Gibt den Tag der Woche in Englisch aus.
TestSQL
SELECT 'Monday' AS ISCORRECT, F_CDOWLONG('23.08.2004') FROM RDB$DATABASE;

F_CDOWLONGLANG
Funktion von adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 Datum optional Uhrzeit
Parameter 2 Sprachkennzeichen für die Ausgabe
de = Deutsch, uk = Englisch, fr = Französich, es = Spanisch, it = Italienisch
Gibt den Tag der Woche in der gewählten Sprache aus.
TestSQL
SELECT 'Lundi' AS ISCORRECT, F_CDOWLONGLANG('23.08.2004', 'fr')
FROM RDB$DATABASE;

F_CDOWSHORT
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(4)
Parameter 1 Datum optional Uhrzeit
Gibt den Tag der Woche abgekürzt in Englisch aus.
TestSQL
SELECT 'Mon' AS ISCORRECT, F_CDOWSHORT('23.08.2004') FROM RDB$DATABASE;

F_CDOWSHORTLANG
Funktion von adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 Datum optional Uhrzeit
Parameter 2 Sprachkennzeichen für die Ausgabe
de = Deutsch, uk = Englisch, fr = Französich, es = Spanisch, it = Italienisch
Gibt den Tag der Woche abgekürzt in der gewählten Sprache aus.
TestSQL
SELECT 'Lun' AS ISCORRECT, F_CDOWSHORTLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

F_GFORMATD

Funktion von adhoc

Input CSTRING(254), TIMESTAMP

Output CSTRING(254)

Parameter 1 d = Tag event. einstellig
dd = Tag immer zweistellig
m = Monat event. einstellig
mm = Monat immer zweistellig
yy = Jahr event. zweistellig
yyyy = Jahr immer vierstellig
h = Stunde event. einstellig
hh = Stunde immer zweistellig
n = Minute event. einstellig
nn = Minute immer zweistellig
s = Sekunde event. einstellig
ss = Sekunde immer zweistellig
alle anderen Zeichen werden entsprechend Ihre Angabe in Parameter 1 an der jeweiligen Stelle angezeigt

Parameter 2 Datum

Formatiert das Datum entsprechend dem Parameter 1

TestSQL

SELECT '01-10-2005 15:09:12' AS ISCORRECT, F_GFORMATD('dd-mm-yyyy hh:nn:ss', '01.10.2005 15:09:12') FROM RDB$DATABASE

F_YEAR

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input TIMESTAMP

Output INTEGER

Parameter 1 Datum optional Uhrzeit

Gibt das Jahr des gewählten Datums aus

TestSQL

SELECT 2004 AS ISCORRECT, F_YEAR(' 22.08.2004 14:38:12') FROM RDB$DATABASE;

F_QUARTER

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input TIMESTAMP

Output INTEGER

Parameter 1 Datum optional Uhrzeit

Gibt das Quartal des gewählten Datums aus

TestSQL

SELECT 3 AS ISCORRECT, F_QUARTER('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_MONTH

Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input TIMESTAMP

Output INTEGER

Parameter 1 Datum optional Uhrzeit

Gibt den Monat des gewählten Datums aus

TestSQL

SELECT 8 AS ISCORRECT, F_MONTH('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_WEEK
Kompatibilität zu FreeUDFLibC
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Gibt die Woche des gewählten Datums aus.
Die Zählung beginnt bei der Woche 1 in die der 1. Januar fällt.
Anmerkung:
Bei den Wochennummerierungen gibt es verschiedene Variationen:
Die erste Woche des Jahres ist

* jene, in die der 1. Januar fällt
* die erste vollständige Woche des Jahres
* die erste Woche, in die mindestens 4 Tage des neuen Jahres fallen. (ISO 8601)

Diese 3 Variationen beziehen sich alle auf die klassische Woche.
Abweichungen gibt es z. B. in Großbritannien, wo es ein Steuerjahr gibt, das immer am 6. April beginnt.
Die internationale Norm ISO 8601 (1973) legt als Wochenanfang den Montag fest. Die erste Woche des Jahres muss mindestens vier Tage enthalten. Dies ist gleichbedeutend mit der Woche, die den 4. Januar enthält, oder der Woche, die den ersten Donnerstag des Jahres enthält.
In Deutschland ist seit 1976 festgelegt, dass die Woche mit dem Montag beginnt: DIN 1355 (1974), DIN EN 28601 (1993).
Nach den vorgenannten Normen hat das Jahr 53 Kalenderwochen, wenn es mit einem Donnerstag beginnt oder endet.
TestSQL
SELECT 41 AS ISCORRECT, F_WEEK('02.10.2005 14:38:12') FROM RDB$DATABASE;

F_HOUR
Kompatibilität zu GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum Uhrzeit
Gibt die Stunden der Uhrzeit des gewählten Datums aus.
Wird nur ein Datum eingegeben, ist die Stunde 0.
TestSQL
SELECT 14 AS ISCORRECT, F_HOUR('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_MINUTE
Kompatibilität zu GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum Uhrzeit
Gibt die Minuten der Uhrzeit des gewählten Datums aus.
Wird nur ein Datum eingegeben, ist die Minute 0.
TestSQL
SELECT 38 AS ISCORRECT, F_MINUTE('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_SECOND
Kompatibilität zu GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum Uhrzeit
Gibt die Sekunden der Uhrzeit des gewählten Datums aus.
Wird nur ein Datum eingegeben, ist die Sekunde 0.
TestSQL
SELECT 12 AS ISCORRECT, F_SECOND('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_YEAROFYEAR
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
(siehe F_YEAR)
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Gibt das Jahr des Datums zurück.
TestSQL
SELECT 2004 AS ISCORRECT, F_YEAROFYEAR(' 22.08.2004 14:38:12') FROM RDB$DATABASE;

F_WEEKOFYEAR
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
(siehe F_WEEK)
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Liefert die Woche des Jahres.
TestSQL
SELECT 34 AS ISCORRECT, F_WEEKOFYEAR('22.08.2004 14:38:12') FROM RDB$DATABASE;

F_WOY
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP
Output CSTRING(7)
Parameter 1 Datum optional Uhrzeit
Liefert das Jahr und die Woche des Jahres als einen zusammenhängenden String.
TestSQL
SELECT '200434' AS ISCORRECT, F_WOY('22.08.2004 14:38:12') FROM RDB$DATABASE;

F_ENCODEDATE
Kompatibilität zu GrUDF
Input INT, INT, INT
Output DATE
Parameter 1 Jahr
Parameter 2 Monat
Parameter 3 Tag
Erzeugt aus Jahr, Monat, Tag einen Datumswert
TestSQL
SELECT '20.02.2004' AS ISCORRECT, F_ENCODEDATE(2004, 2, 20) FROM RDB$DATABASE

F_ENCODETIME
Kompatibilität zu GrUDF
Input INT, INT, INT
Output TIME
Parameter 1 Stunden
Parameter 2 Minuten
Parameter 3 Sekunden
Erzeugt aus Stunden, Minuten, Sekunden einen Zeitwert
SELECT '09:45:53' AS ISCORRECT, F_ENCODETIME(9, 45, 53) FROM RDB$DATABASE

F_ENCODETIMESTAMP
Kompatibilität zu GrUDF
Input INT, INT, INT, INT, INT, INT
Output TIMESTAMP
Parameter 1 Jahr
Parameter 2 Monat
Parameter 3 Tag
Parameter 4 Stunden
Parameter 5 Minuten
Parameter 6 Sekunden
Erzeugt aus Jahr, Monat, Tag, Stunden, Minuten, Sekunden einen DatumZeit-Wert.
TestSQL
SELECT '20.02.2004 09:45:53' AS ISCORRECT, F_ENCODETIMESTAMP(2004, 2, 20, 9, 45, 53) FROM RDB$DATABASE
SELECT '20.02.2004 00:00:00' AS ISCORRECT, F_ENCODETIMESTAMP(2004, 2, 20, 0, 0, 0) FROM RDB$DATABASE

F_STRTOTIME
Kompatibilität zu FreeUDFLibC
Input CSTRING(11)
Output TIME
Parameter 1 anglophile Uhrzeit als String
Wandelt eine anglophile Uhrzeit (05:04:01 AM) in eine 24-Stunden-Uhrzeit um.
Eingabeformat kann jeweils 1- oder 2-stellig sein und die Trennung von AM/PM mit oder ohne Leerzeichen erfolgen, AM/PM muß in Großbuchstaben eingegeben werden.
TestSQL
SELECT '05:04:01' AS ISCORRECT, F_STRTOTIME('05:04:01 AM') FROM RDB$DATABASE
SELECT '17:04:01' AS ISCORRECT, F_STRTOTIME('5:4:1PM') FROM RDB$DATABASE

F_STRIPDATE
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output TIMESTAMP
Parameter 1 Datum Uhrzeit
Gibt das Datum Uhrzeit als 31.12.1899 (Datum 0) mit der gewählten Uhrzeit zurück.
Anmerkung:
Um nur die Uhrzeit zu erhalten verwenden Sie CAST(' 01.10.2005 15:00:00' AS TIME)
TestSQL
SELECT '31.12.1899 15:00:00' AS ISCORRECT, F_STRIPDATE(' 01.10.2005 15:00:00') FROM RDB$DATABASE;

F_STRIPTIME
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP
Output TIMESTAMP
Parameter 1 Datum Uhrzeit
Gibt das Datum Uhrzeit als Uhrzeit 00:00:00 vom gewählten Datum zurück.
Anmerkung:
Um nur das Datum zu erhalten verwenden Sie CAST(‘01.10.2005 15:00:00' AS DATE)
TestSQL
SELECT '01.10.2005 00:00:00' AS ISCORRECT, F_STRIPTIME(' 01.10.2005 15:00:00') FROM RDB$DATABASE;

F_EQUALDATE
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 Datum optional Uhrzeit 1
Parameter 2 Datum optional Uhrzeit 2
Vergleicht zwei Datum-Zeit-Werte auf Gleichheit, berücksichtigt dabei aber nur das Datum
Ergebnis 1 = ist gleich, 0 = ist ungleich
TestSQL

```
SELECT 1 AS ISCORRECT, F_EQUALDATE('20.02.2004 10:00:00', '20.02.2004 11:00:00')
FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_EQUALDATE('20.02.2004 10:00:00', '21.02.2004 11:00:00')
FROM RDB$DATABASE;
```

F_EQUALDATETIME
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 Datum optional Uhrzeit 1
Parameter 2 Datum optional Uhrzeit 2
Vergleicht zwei Datum-Zeit-Werte auf Gleichheit, berücksichtigt dabei Datum und Uhrzeit
Ergebnis 1 = ist gleich, 0 = ist ungleich
TestSQL

```
SELECT 1 AS ISCORRECT, F_EQUALDATETIME('20.02.2004 10:00:00', '20.02.2004
10:00:00') FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_EQUALDATETIME('20.02.2004 10:00:00', '20.02.2004
11:00:00') FROM RDB$DATABASE;
```

F_ISLEAPYEAR
Kompatibilität zu FreeUDFLibC
Input TIMESTAMP
Output INTEGER
Parameter 1 Datum optional Uhrzeit
Liefert, ob das Jahr des Gewählten Datums ein Schaltjahr ist.
Ergebnis 1 = ist Schaltjahr, 0 = ist kein Schaltjahr
Algorithmus Y2k-fest (2000 war ein Schaltjahr).
TestSQL

```
SELECT 1 AS ISCORRECT, F_ISLEAPYEAR('22.08.2000 14:38:12') FROM
RDB$DATABASE;
```

F_MAXDATE
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP
Output TIMESTAMP
Parameter 1 Datum optional Uhrzeit
Parameter 2 Datum optional Uhrzeit
Gibt das größere der beiden eingegebenen Datums zurück.
TestSQL
SELECT '01.10.2005 15:00:00' AS ISCORRECT, F_MAXDATE('22.08.2000 14:38:12',
'01.10.2005 15:00:00') FROM RDB$DATABASE;

F_MINDATE
Kompatibilität zu FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP
Output TIMESTAMP
Parameter 1 Datum optional Uhrzeit
Parameter 2 Datum optional Uhrzeit
Gibt das kleinere der beiden eingegebenen Datums zurück.
TestSQL
SELECT '22.08.2000 14:38:12' AS ISCORRECT, F_MINDATE('22.08.2000 14:38:12',
'01.10.2005 15:00:00') FROM RDB$DATABASE;

F_OSTERDATUM
Kompatibilität zu FreeUDFLib AvERP, GrUDF
Input INTEGER
Output DATE
Parameter 1 Jahr, für welches der Ostersonntag berechnet werden soll
Gibt das Datum des Ostersonntags des gewählten Jahres zurück.
TestSQL
SELECT '27.03.2005' AS ISCORRECT, F_OSTERDATUM(2005) FROM RDB$DATABASE;

F_ZEITDIFFERENZ

Kompatibilität zu FreeUDFLib AvERP, GrUDF

Input	TIMESTAMP, TIMESTAMP, CSTRING(1)
Output	DOUBLE
Parameter 1	Datum optional Uhrzeit
Parameter 2	Datum optional Uhrzeit
Parameter 3	Art der Ausgabe t = Zeitdifferenz in Tagen h = Zeitdifferenz in Stunden m = Zeitdifferenz in Minuten s = Zeitdifferenz in Sekunden alle anderen Werte ergeben immer 0

Berechnet die Zeitdifferenz von Datum 1 zu Datum 2, Ausgabe als Fließkommazahl in der im Parameter 3 gewählten Ausgabeart

TestSQL

```
SELECT 1.000 AS ISCORRECT, F_ZEITDIFFERENZ('02.10.2005 15:00:00', '01.10.2005 15:00:00', 't') FROM RDB$DATABASE
SELECT 1.125 AS ISCORRECT, F_ZEITDIFFERENZ('02.10.2005 18:00:00', '01.10.2005 15:00:00', 't') FROM RDB$DATABASE
SELECT 26.500 AS ISCORRECT, F_ZEITDIFFERENZ('02.10.2005 18:00:00', '01.10.2005 15:30:00', 'h') FROM RDB$DATABASE
SELECT 1589.500 AS ISCORRECT, F_ZEITDIFFERENZ('02.10.2005 18:00:00', '01.10.2005 15:30:30', 'm') FROM RDB$DATABASE
SELECT 95370.000 AS ISCORRECT, F_ZEITDIFFERENZ('02.10.2005 18:00:00', '01.10.2005 15:30:30', 's') FROM RDB$DATABASE
SELECT 0.000 AS ISCORRECT, F_ZEITDIFFERENZ('02.10.2005 18:00:00', '01.10.2005 15:30:30', 'x') FROM RDB$DATABASE
```

BLOb Funktionen - BLOb Konvertierungen

F_BLOBASPCHAR

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input BLOB
Output CSTRING(32760)
Parameter 1 TextBlob, der nach VarChar konvertiert werden soll
Konvertiert TextBlob nach VarChar
TestSQL (für TestISO.GDB)
SELECT 'ein einzeiliger TextBLOb' AS ISCORRECT, TEXTBLOB,
F_BLOBASPCHAR(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 1

F_STRBLOB

Kompatibilität zu FreeUDFLib, GrUDF
Input CSTRING(32760)
Output BLOB
Parameter 1 VarChar, der nach TextBlob konvertiert werden soll
Konvertiert VarChar nach TextBlob

F_BLOB2EXCEL

Funktion von adhoc
Input BLOb
Output BLOb
Parameter 1 BLOb, der für Excel umgewandelt werden soll
Um auch mehrzeilige Texte und Texte, die Doppelhochkommata enthalten, aus BLObs nach Excel exportieren zu können, bedarf es einiger Umformungen. Diese Funktion erledigt das:
- fügt am Anfang und Ende des Strings jeweils ein Doppelhochkomma hinzu
- verdoppelt alle im Text selber vorkommenden Doppelhochkommata
- entfernt alle CHR(13) aus dem String
- begrenzt den übergebenen BLOb auf 32760 Zeichen (Grenze von Excel für ein Feld)
TestSQL
SELECT '"ein dreizeiliger TextBLOb' || F_LF() || 'mit einer zweiten Zeile' || F_LF() || 'und einer dritten Zeile"' AS ISCORRECT, F_BLOB2EXCEL(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 3
Anmerkung:
Da es eigentlich keinen Sinn macht, einen in der Datenbank vorhandenen (sehr) langen Text in eine Excel-Zelle zu exportieren, dürfte diese Funktion (erst) in Kombination mit F_LEFT bzw. F_RIGHT erst praktikabel sein. Bsp.:
SELECT F_RIGHT(F_BLOB2EXCEL(BLObFeldTAGEBUCH), 1000) FROM ... exportiert z.B. die letzten 1000 Zeichen des Tagebuchs.

BLOb Funktionen - BLOb Bearbeitungen

F_BLOBCAT

Kompatibilität zu GrUDF
Input BLOB, BLOB
Output BLOB
Parameter 1 Text-Blob, der mit Text-BLOb aus Parameter 2 zusammengeführt werden soll
Parameter 2 Text-Blob, der an Text-BLOb aus Parameter 1 angehängt werden soll
Verkettet die Inhalte zweier Blobfelder zu einem Gemeinsamen.
Die Funktion macht nach dem 1. BLOb einen CRLF, bevor sie den 2. BLOb anhängt.
TestSQL (für TestISO.GDB)
INSERT INTO BLOBTEST (TEXTBLOB) SELECT F_BLOBCAT(TEXTBLOB, (SELECT TEXTBLOB FROM BLOBTEST WHERE BLOBTESTID = 2)) FROM BLOBTEST WHERE BLOBTESTID = 1
Erzeugt in Tabelle BLOBTEST einen neuen Datensatz, wo das Feld TEXTBLOB sich aus den zusammengefügten Inhalten von TEXTBLOB des Datensatzes mit der TEXTBLOBID 1 und des Datensatzes mit der TEXTBLOBID 2 ergibt.

F_BLOBCATSTR

Kompatibilität zu GrUDF
Input BLOB, CSTRING(32760)
Output BLOB
Parameter 1 Text-Blob, der mit Text-String aus Parameter 2 zusammengeführt werden soll
Parameter 2 Text-String, der an Text-BLOb aus Parameter 1 angehängt werden soll
Verkettet den Inhalt eines Blobfeldes mit einem String zu einem BLOb.
Die Funktion macht nach dem 1. BLOb einen CRLF, bevor sie den 2. String anhängt.
TestSQL (für TestISO.GDB)
INSERT INTO BLOBTEST (TEXTBLOB) SELECT F_BLOBCATSTR(TEXTBLOB, 'Diese Zeile wurde angehängt') FROM BLOBTEST WHERE BLOBTESTID = 1
Erzeugt in Tabelle BLOBTEST einen neuen Datensatz, wo das Feld TEXTBLOB sich aus den zusammengefügten Inhalten von TEXTBLOB des Datensatzes mit der TEXTBLOBID 1 und des Strings ‘Diese Zeile wurde angehängt'' ergibt.

F_BLOBLEFT

Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input BLOB, INT
Output CSTRING(32760)
Parameter 1 Text-Blob, der beschnitten werden soll
Parameter 2 Länge des auszugebenden Strings
Gibt einen String beschnitten auf die Anzahl Zeichen von links von Parameter 2 aus.
Zählung beginnt bei 1
TestSQL (für TestISO.GDB)
SELECT 'ein einzeiliger' AS ISCORRECT, F_BLOBLEFT(TEXTBLOB, 15) FROM BLOBTEST WHERE BLOBTESTID = 1

F_BLOBMID
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input BLOB, INT, INT
Output CSTRING(32760)
Parameter 1 TextBlob, aus dem eine Zeichenkette ermittelt werden soll
Parameter 2 Position, an der der zu ermittelnde String beginnt
Parameter 3 Länge des zu ermittelnden Strings
Gibt die Anzahl der Buchstaben (Parameter 3) des eingegebenen Text ab der eingegebenen Buchstabennummer (Parameter 2) wieder. Zählung beginnt für Parameter 2 bei 0.
TestSQL (für TestISO.GDB)
SELECT 'zwei' AS ISCORRECT, F_BLOBMID(TEXTBLOB, 4, 4) FROM BLOBTEST WHERE BLOBTESTID = 2

F_BLOBRIGHT
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input BLOB, INT
Output CSTRING(32760)
Parameter 1 TextBlob
Parameter 2 Anzahl der Zeichen von rechts
Gibt den String beschnitten auf die Anzahl Zeichen gezählt von rechts von Parameter 2 aus, Zählung beginnt bei 1
TestSQL (für TestISO.GDB)
SELECT 'dritten Zeile' AS ISCORRECT, F_BLOBRIGHT(TEXTBLOB, 13) FROM BLOBTEST WHERE BLOBTESTID = 3

F_BLOBREPLACESTRING
Funktion von adhoc
Input BLOB, CSTRING(254), CSTRING(254), INT, INT
Output BLOB
Parameter 1 der Text-BLOb, indem eine Zeichenkette ausgetauscht werden soll
Parameter 2 der auszutauschende String
Parameter 3 der String, der gesetzt werden soll
Parameter 4 0 = nur das erste Vorkommen austauschen, 1 = alle Vorkommen austauschen
Parameter 5 0 = Groß-/Kleinschreibung berücksichtigen, 1 = nicht berücksichtigen
Ersetzt in einem Text-BLOb eine/alle Zeichenkette/n aus Parameter 2 durch eine andere Zeichenkette aus Parameter 3 und kann auch Groß-Kleinschreibung berücksichtigen
TestSQL (für TestISO.GDB)
SELECT 'vier einzeiliger TextBLOb' AS ISCORRECT, F_BLOBREPLACESTRING(TEXTBLOB, 'ein', 'vier', 0, 0) FROM BLOBTEST WHERE BLOBTESTID = 1
SELECT 'vier vierzeiliger TextBLOb' AS ISCORRECT, F_BLOBREPLACESTRING(TEXTBLOB, 'ein', 'vier', 1, 0) FROM BLOBTEST WHERE BLOBTESTID = 1

F_BLOBSUBSTR
Funktion von adhoc
Input BLOB, CSTRING(1024)
Output INT
Parameter 1 TextBLOb (in dem die Position von Parameter 2 ermittelt werden soll)
Parameter 2 String 2 (dessen Position in Parameter 1 ermittelt werden soll)
Gibt die erste Position im BLOb aus, bei der der String 2 beginnt.
Zählung beginnt bei 0.
TestSQL (für TestISO.GDB)
SELECT 4 AS ISCORRECT, F_BLOBSUBSTR(TEXTBLOB, 'einzeiliger') FROM BLOBTEST WHERE BLOBTESTID = 1

F_BLOBLINE
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP
Input BLOB, INT
Output CSTRING(32760)
Parameter 1 TextBLOb
Parameter 2 Nr. der Zeile, die wiedergegeben werden soll
Gibt die <Parameter 2> Zeile des TextBLObs wieder
TestSQL (für TestISO.GDB)
SELECT ‘mit einer zweiten Zeile’ AS ISCORRECT, F_BLOBLINE(TEXTBLOB, 2) FROM BLOBTEST WHERE BLOBTESTID = 3

F_BLOBSIZE
Kompatibilität zu FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input BLOB
Output INT
Parameter 1 TextBlob
Gibt bei einem TextBlobs die Anzahl der Zeichen (analog F_STRINGLENGTH) (Zeilenumbrüche CRLF werden als 2 Zeichen gezählt), bei einem BinarBlob die Größe der Datei in Byte an
TestSQL (für TestISO.GDB)
Für TextBLOb:
SELECT 50 AS ISCORRECT, F_BLOBSIZE(TEXTBLOB), F_STRINGLENGTH(F_BLOBASPCHAR(TEXTBLOB)) FROM BLOBTEST WHERE BLOBTESTID = 2
Für BinärBLOb:
SELECT 1426 AS ISCORRECT, F_BLOBSIZE(BINAERBLOB) FROM BLOBTEST WHERE BLOBTESTID = 4

F_BLOBMAXSEGMENTLENGTH
Kompatibilität zu FreeUDFLib
Input BLOB
Output INT
Parameter 1 TextBLOb oder BinärBLOb
Ermittelt die Anzahl der Bytes des größten BLOB-Segmentes
TestSQL (für TestISO.GDB)
Für TextBLOb:
SELECT 16384 AS ISCORRECT, F_BLOBMAXSEGMENTLENGTH(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 8
Für BinärBLOb:
SELECT 16384 AS ISCORRECT, F_BLOBMAXSEGMENTLENGTH(BINAERBLOB) FROM BLOBTEST WHERE BLOBTESTID = 7

F_BLOBSEGMENTCOUNT
Kompatibilität zu FreeUDFLib
Input BLOB
Output INT
Parameter 1 TextBLOb oder BinärBLOb
Ermittelt die Anzahl der BLOB-Segmente
TestSQL (für TestISO.GDB)
Für TextBLOb:
SELECT 3 AS ISCORRECT, F_BLOBSEGMENTCOUNT(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 8
Für BinärBLOb:
SELECT 2 AS ISCORRECT, F_BLOBSEGMENTCOUNT(BINAERBLOB) FROM BLOBTEST WHERE BLOBTESTID = 7

F_BLOBLINECOUNT
Funktion von adhoc
Input BLOB
Output INT
Parameter 1 TextBLOb
Ermittelt die Anzahl der Zeilen in einem TextBLOb
TestSQL (für TestISO.GDB)

```
SELECT 3 AS ISCORRECT, F_BLOBLINECOUNT(TEXTBLOB) FROM BLOBTEST
WHERE BLOBTESTID = 3
```

BLOb Funktionen - BLOb Vergleiche

F_BLOBCOMPARE

Kompatibilität zu FreeUDFLib, GrUDF (BLOBICOMP)
Input BLOB
Output INT
Parameter 1 TextBLOb
Vergleicht zwei BinärBLObs miteinander
TestSQL (für TestISO.GDB)

```
SELECT 1 AS ISCORRECT, F_BLOBCOMPARE(TEXTBLOB, TEXTBLOB) FROM
BLOBTEST WHERE BLOBTESTID = 3
SELECT 0 AS ISCORRECT, F_BLOBCOMPARE(TEXTBLOB, (SELECT TEXTBLOB FROM
BLOBTEST WHERE BLOBTESTID = 2)) FROM BLOBTEST WHERE BLOBTESTID = 3
```

Vorbemerkung (siehe http://de.wikipedia.org/wiki/UUID):
Ein Universally Unique Identifier (UUID) ist ein Standard für Identifizierer, der in der Softwareentwicklung verwendet wird. Er ist von der Open Software Foundation (OSF) als Teil des Distributed Computing Environment (DCE) standardisiert. Die Absicht hinter UUIDs ist, Informationen in verteilten Systemen ohne großartige zentrale Koordination eindeutig kennzeichnen zu können.
Ein UUID besteht aus einer 16-Byte-Zahl, die in fünf Gruppen unterteilt wird. In ihrer Normalform sieht eine UUID so aus: 550e8400-e29b-11d4-a716-446655440000
Obwohl die Eindeutigkeit für generierte UUID nicht garantiert ist, ist die Gesamtzahl der eindeutigen Schlüssel mit 2^{128} oder $3{,}4028*10^{38}$ so groß, dass die Wahrscheinlichkeit der Erzeugung zwei gleicher UUIDs gegen null geht. Daher können UUIDs beliebig ohne zentrales Kontrollorgan erzeugt und zur Kennzeichnung eingesetzt werden, ohne Gefahr zu laufen, dass gleiche UUID für etwas anderes verwenden wird. Mit UUID markierte Informationen können somit später in einer einzigen Datenbank zusammengeführt werden ohne Bezeichnerkonflikte auflösen zu müssen. Eine weit verbreitete Implementierung des UUID-Standards ist Microsofts Globally Unique Identifier (GUID).
Das originale (Version 1) Generierungsschema für UUID war, die UUID-Version mit der MAC-Adresse des Computers der die UUID generiert und der Anzahl der 100-Nanosekunden-Intervalle seit Beginn des Gregorianischen Kalenders (15.10.1582 00:00:00 Uhr) aneinander zu hängen. In der Praxis ist der eigentliche Algorithmus komplizierter. Dieses Schema wurde kritisiert weil es nicht ausreichend dicht sei; es gibt beides, die Identität des generierenden Computers als auch den Zeitpunkt zu dem er es tat, preis.
Im RFC4122 sind 5 UUID-Versionen definiert

1. Zeit-basierte
   a. mit eindeutiger MAC Adresse nach IEEE 802-Standard
   b. mit per Zufallsgenerator erzeugter MAC-Adresse nach IEEE 802-Standard
2. DCE-Sichereitsversion (mit POSIX UIDs)
3. Names(raum)-basierte (mittels MD5 Kodierung)
4. Per Zufallsgenerator erzeugte
5. Namens(raum)-basierte (mittels SHA-1 Kodierung)

Für den Zweck der dezentralen Generierung von eindeutigen IDs für Datensätze unter Berücksichtigung der Kompatibilität zwischen Windows und Linux mittels gleicher Algorithmen eignen sich nur die Versionen 1 und 4 (Versionen 3 und 5 nicht, weil nicht immer sichergestellt ist, dass ein Namensraum (URL) gegeben ist).
Da es vom Anwendungsfall abhängt, ob eine “Rückschlüsselung” von Erstellungszeit und Erstellungsort erwünscht ist oder nicht, sind 3 verschieden UUID-Generatoren implementiert:
F_UUID1MAC
Version 1 mit echter MAC-Adresse
Rückschlüsselung der Erstellungszeit und der MAC-Adresse möglich
F_UUID1RAND
Version 1 mit per Zufallsgenerator generierter MAC-Adresse
nur Rückschlüsselung des Erstellungszeitpunktes möglich
F_UUID4
Version 4
keinerlei Rückschlüsselung möglich
Darüber hinaus gibt es mit der uuidlib von Ian Newby eine Abwandlung der Version 1b (mit Zufallsgenerator) die komprimiert wurde statt sie nach dem in der RFC4122 vorgegebenen Maske auszugeben. Dieser Algorithmus gibt zuerst die MAC-adresse und dann den Zeitstempel aus, was bei auf dem gleichen Rechner erzeugten UUIDs Vorteile wegen besserer Indexierung bringt. Diesen Grund-Algorithmus haben wir, inclusive der Umwandlungs-Funktionen, ebenfalls übernommen, so dass es zu jeder der 3 (Basis-)UUID-Generatoren noch eine komprimierte Version gibt (für die jeweils die Rückschlüsselung der Basis-Version gilt):
F_UUID1MACCOMPR, UUID1RANDCOMPR, UUID4COMPR.

F_UUID1MAC
Funktion von adhoc
Input nichts
Output CSTRING(36)
Erzeugt einen Universally Unique Identifier (UUID) der Version 1a
TestSQL
SELECT F_UUID1MAC() FROM RDB$DATABASE;
Anmerkung:
Ist auf dem DB-Server keine MAC-Adresse zu finden, erzeugt die Funktion ein F_UUID1RAND

F_UUID1RAND
Ergebnis-kompatibel zur Funktion GUID_CREATE in der uuidlib
Input nichts
Output CSTRING(36)
Erzeugt einen Universally Unique Identifier (UUID) der Version 1b
TestSQL
SELECT F_UUID1RAND() FROM RDB$DATABASE;

F_UUID4
Funktion von adhoc
Input nichts
Output CSTRING(36)
Erzeugt einen Universally Unique Identifier (UUID) der Version 4
TestSQL
SELECT F_UUID4() FROM RDB$DATABASE;

Anmerkung (aus der uuidlib):
create_uuid() creates a 22 char string guid, which is a compressed form where the order of the sections is reversed to allow firebirds index prefix compression to take place. All the characters used in this uuid are valid characters in urls.
F_UUID1MACCOMPR
Funktion von adhoc
Input nichts
Output CSTRING(22)
Erzeugt einen komprimierten Universally Unique Identifier (UUID) der Version 1a
TestSQL
SELECT F_UUID1MACCOMPR() FROM RDB$DATABASE;

F_UUID1RANDCOMPR
Ergebnis-kompatibel zur Funktion UUID_CREATE in der uuidlib
Input nichts
Output CSTRING(22)
Erzeugt einen komprimierten Universally Unique Identifier (UUID) der Version 1b
TestSQL
SELECT F_UUID1RANDCOMPR() FROM RDB$DATABASE;

F_UUID4COMPR
Funktion von adhoc
Input nichts
Output CSTRING(22)
Erzeugt einen komprimierten Universally Unique Identifier (UUID) der Version 4
TestSQL
SELECT F_UUID4COMPR() FROM RDB$DATABASE;

F_UUID2UUIDCOMPR
Ergebnis-kompatibel zur Funktion GUID_TO_UUID in der uuidlib
Input CSTRING(250)
Output CSTRING(22)
Parameter 1 gültige UUID
Wandelt eine UUID (jeder Version) in eine komprimierte UUID um
TestSQL (für TestISO.GDB)
Übergabe einer komprimierten UUID als Input Paramter statt einer normalen UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID2UUIDCOMPR(UUIDCOMPR) FROM UUIDTEST;
Übergabe einer beliebigen Zeichenkette als Input Paramter statt einer normalen UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID2UUIDCOMPR('1235abcde') FROM UUIDTEST;
Übergabe einer normalen UUID als Input Paramter (richtig):
SELECT 'this is valid Input' AS ISCORRECT, F_UUID2UUIDCOMPR(UUID) FROM UUIDTEST;

F_UUIDCOMPR2UUID
Ergebnis-kompatibel zur Funktion UUID_TO_GUID in der uuidlib
Input CSTRING(250)
Output CSTRING(36)
Parameter 1 gültige komprimierte UUID
Wandelt eine komprimierte UUID in eine normale UUID der Version um, die der komprimierten UUID entsprach
TestSQL (für TestISO.GDB)
Übergabe einer normalen UUID als Input Paramter statt einer komprimierten UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUIDCOMPR2UUID(UUID) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;
Übergabe einer beliebigen Zeichenkette als Input Paramter statt einer komprimierten UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUIDCOMPR2UUID('12345abcde') FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;
Übergabe einer komprimierten UUID als Input Paramter (richtig):
SELECT 'this is valid Input' AS ISCORRECT, F_UUIDCOMPR2UUID(UUIDCOMPR) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;

F_UUIDVERSION
Funktion von adhoc
Input CSTRING(250)
Output CSTRING(20)
Parameter 1 gültige UUID
Ermittelt die Variante und Version der UUID
Mögliche Varianten sind:

- 0 NCS (reserviert für Rückwärtskompatibilität)
- 10 die aktuelle Variante nach RFC4122
- 110 Microsoft (reserviert für Rückwärtskompatibilität))
- 111 reserviert für zukünftige Varianten

Mögliche Versionen sind:

- 1 Zeit-basierte
  a. mit eindeutiger MAC Adresse nach IEEE 802-Standard
  b. per Zufallsgenerator erzeugter MAC-Adresse nach IEEE 802-Standard
- 2 DCE-Sichereitsversion (mit POSIX UIDs)
- 3 Names(raum)-basierte (mittels MD5 Kodierung)
- 4 Per Zufallsgenerator erzeugte
- 5 Namens(raum)-basierte (mittels SHA-1 Kodierung)

Die Funktion liefert folgendes zurück:

- V1a
- V1b
- V3
- V4
- V5

TestSQL (für TestISO.GDB)
SELECT ART, F_UUIDVERSION(UUID) FROM UUIDTEST ORDER BY UUIDTESTID;

F_UUID1TIMESTAMP
Funktion von adhoc
Input CSTRING(250)
Output TIMESTAMP
Parameter 1 gültige UUID Version 1
Ermittelt den Zeitstempel, wann die UUID erzeugt wurde aus der UUID1
Bei nicht gültigem Parameter gibt es den Zeitstempel 31.12.1899 00:00:00 aus (Zeit = 0 - Start der internen InterBase/FireBird Zeitrechnung)
TestSQL (für TestISO.GDB)
Übergabe einer komprimierten UUID als Input Paramter statt einer nornalen UUID:
SELECT '31.12.1899 00:00:00' AS ISCORRECT, F_UUID1TIMESTAMP(UUIDCOMPR) FROM UUIDTEST;
Übergabe einer normalen UUID als Input Paramter (richtig):
SELECT ZEITSTEMPEL AS ISCORRECT, F_UUID1TIMESTAMP(UUID) FROM UUIDTEST;

F_UUID1COMPRTIMESTAMP
Funktion von adhoc
Input CSTRING(250)
Output TIMESTAMP
Parameter 1 gültige komprimierte UUID Version 1
Ermittelt den Zeitstempel, wann die UUID erzeugt wurde aus der komprimierten UUID1
Bei nicht gültigem Parameter gibt es den Zeitstempel 31.12.1899 00:00:00 aus (Zeit = 0 - Start der internen InterBase/FireBird Zeitrechnung)
TestSQL (für TestISO.GDB)
Übergabe einer normalen UUID als Input Paramter statt einer komprimierten UUID:
SELECT '31.12.1899 00:00:00' AS ISCORRECT, F_UUID1COMPRTIMESTAMP(UUID) FROM UUIDTEST;
Übergabe einer komprimierten UUID als Input Paramter (richtig):
SELECT ZEITSTEMPEL AS ISCORRECT, F_UUID1COMPRTIMESTAMP(UUIDCOMPR) FROM UUIDTEST;

F_UUID1MACMAC
Funktion von adhoc
Input CSTRING(250)
Output CSTRING(17)
Parameter 1 gültige UUID Version 1a
Ermittelt die MAC-Adresse des Computers, auf dem die UUID erzeugt wurde aus der UUID1MAC
TestSQL (für TestISO.GDB)
Übergabe einer komprimierten UUID als Input Paramter statt einer normalen UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID1MACMAC(UUIDCOMPR) FROM UUIDTEST ORDER BY UUIDTESTID;
Übergabe einer normalen UUID als Input Paramter (richtig):
SELECT F_UUIDVERSION(UUID), F_UUID1MACMAC(UUID) FROM UUIDTEST ORDER BY UUIDTESTID;

F_UUID1MACCOMPRMAC
Funktion von adhoc
Input CSTRING(250)
Output CSTRING(17)
Parameter 1 gültige komprimierte UUID Version 1a
Ermittelt die MAC-Adresse des Computers, auf dem die UUID erzeugt wurde aus der UUID1MACCOMPR
TestSQL (für TestISO.GDB)
Übergabe einer normalen UUID als Input Paramter statt einer komprimierten UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID1MACCOMPRMAC(UUID) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL ORDER BY UUIDTESTID;
Übergabe einer komprimierten UUID als Input Paramter (richtig):
SELECT F_UUIDVERSION(F_UUIDCOMPR2UUID(UUIDCOMPR)), F_UUID1MACCOMPRMAC(UUIDCOMPR) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL ORDER BY UUIDTESTID;

F_UUID1COMPARE

Funktion von adhoc

Input	CSTRING(250), CSTRING(250)
Output	SMALLINT
Parameter 1	1. UUID Version 1
Parameter 2	2. UUID Version 1

Vergleicht zwei Universally Unique Identifier (UUID) lexikalisch.
Lexikalisch heißt hier von links nach rechts Blockweise, also

- zuerst wird der 1. Block der UUID verglichen
- dann der 2. Block
- dann der 3. Block

Wenn die jeweils zu vergleichenden Blocks identisch sind, wird der nächste Block getestet.
Die Funktion durchläuft den Vergleich der jeweiligen Blocks solange, bis ein Unterschied festgestellt wurde oder der 3. Block erreicht wurde.
Die Funktion gibt dann aus:

-1 u1 ist lexikalsich vor u2
0 u1 ist gleich u2
1 u1 ist lexikalisch nach u2

Beachte, dass eine lexikalische Sortierung hier keine zeitliche Sortierung ist!
Um eine zeitliche Sortierung einer UUID Version 1 zu erreichen (um z.B. die UUID nach ihrer Entstehungszeit zu sortieren):

```
SELECT UUID, F_UUID1TIMESTAMP(UUID) FROM UUIDTEST ORDER BY 2 DESC;
```

Anmerkung:
Natürlich könnte man mit dieser Funktion auch andere Versionen als Version 1 vergleichen - raus käme aber nur Unsinn, denn nur eine Version 1 enthält einen echten Zeitstempel, auf den der Algorithmus aufsetzt.
TestSQL (für TestISO.GDB)

```
SELECT -1 AS ISCORRECT, F_UUIDCOMPARE((SELECT UUID FROM UUIDTEST
WHERE UUIDTESTID = 1), (SELECT UUID FROM UUIDTEST WHERE UUIDTESTID = 2))
FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_UUIDCOMPARE((SELECT UUID FROM UUIDTEST
WHERE UUIDTESTID = 2), (SELECT UUID FROM UUIDTEST WHERE UUIDTESTID = 2))
FROM RDB$DATABASE;
SELECT 1 AS ISCORRECT, F_UUIDCOMPARE((SELECT UUID FROM UUIDTEST
WHERE UUIDTESTID = 2), (SELECT UUID FROM UUIDTEST WHERE UUIDTESTID = 1))
FROM RDB$DATABASE;
```

F_IF
Funktion von adhoc
Input CSTRING(32), CSTRING(8), CSTRING(32), CSTRING(8190), CSTRING(8190)
Output CSTRING(8190)
Parameter 1 Vergleichsstring 1
Parameter 2 Vergleichsoperator
=
<>
<
>
<=
>=
Jedem dieser Operatoren kann ein “n” (nummerisch) vorangestellt werden, wenn die zu vergleichende Strings eine Flieskommazahl enthalten, z.B. n=
Parameter 3 Vergleichstring 2
Parameter 4 wenn Vergleich zutrifft, dann Ergebnis das, was in 4. Parameter steht
Parameter 5 wenn Vergleich nicht zutrifft, dann Ergebnis das, was in 5. Parameter steht
“Nachbildung” einer IF-Schleife
TestSQL
```
SELECT 'Parameter 1 ist kleiner' AS ISCORRECT, F_IF('Test', '<=', 'Testa', 'Parameter 1 ist kleiner', 'Parameter 1 ist größer') FROM RDB$DATABASE;
SELECT 'Parameter 1 ist größer' AS ISCORRECT, F_IF('Testb', '<=', 'Testa', 'Parameter 1 ist kleiner', 'Parameter 1 ist größer') FROM RDB$DATABASE;
SELECT 'Parameter 1 ist kleiner' AS ISCORRECT, F_IF('Test1', 'n<=', 'Test2', 'Parameter 1 ist kleiner', 'Parameter 1 ist größer') FROM RDB$DATABASE;
```

F_VERSION
Funktion von adhoc
Input nichts
Output CSTRING(254)
Ermittelt die Version der FreeAdhocUDF
TestSQL
```
SELECT F_VERSION() FROM RDB$DATABASE;
```

Funktionen, die bisher noch NICHT umgesetzt wurden (kein Bedarf, andere Möglichkeit)

aus FreeUDFLib / FreeUDFLib AvERP
- F_IBPassword
- F_GenerateFormattedName
- F_ValidateNameFormat
- F_ValidateRegularExpression
- F_ValidateStringInRE
- F_CloseDebuggerOutput
- F_Debug
- F_IBTempPath
- F_SetDebuggerOutput
- F_ValidateCycleExpression
- F_EvaluateCycleExpression
- F_EvaluateExpression

aus FreeUDFLib C
alle umgesetzt

aus GrUDF (Stand 18.11.2005)
- F_ASCII
- F_DAYSPAN -> macht das Gleiche wie F_AGEINDAYS
- F_MONTHSPAN -> macht das Gleiche wie F_AGEINMONTH
- F_YEARSPAN -> macht das Gleiche wie F_AGEINYEARS
- F_WEEKSPAN -> macht das Gleiche wie F_AGEINWEEKS
- F_SECONDSPAN -> macht das Gleiche wie F_AGEINSECONDS
- F_MINUTESPAN -> macht das Gleiche wie F_AGEINMINUTES
- F_HOURSPAN -> macht das Gleiche wie F_AGEINHOURS
- F_MONTHSTART -> macht das Gleiche wie CAST(‘01.02.2006' AS DATE)
- F_MONTHEND -> macht das Gleiche wie CAST(F_LASTDAY(2006, 2) || ‘02.2006' AS DATE)
- F_DAYFRAC -> macht das Gleiche wie CAST(F_AGEINSECONDS('20.02.2004 00:00:00', '20.02.2004 12:00:00') AS DOUBLE PRECISION) / 86400
  Oder CAST(F_AGEINHOURS('20.02.2004 00:00:00', '20.02.2004 12:00:00') AS DOUBLE PRECISION) / 24
- F_DOUBLE -> macht das Gleiche wie CAST(1234.123 AS DOUBLE PRECISION)
- F_FRAC -> macht das Gleiche wie 1234.12 - F_TRUNCATE(1234.12)
- F_TRUNCDEC
- F_CEIL -> s. ib_util, freie UDF-Funktionen von InterBase
- F_FLOOR -> s. ib_util, freie UDF-Funktionen von InterBase
- F_DIV -> s. ib_util, freie UDF-Funktionen von InterBase
- F_EXP
- F_SQRT -> s. ib_util, freie UDF-Funktionen von InterBase
- F_POWER
- F_LNXP1
- F_LOG10 -> s. ib_util, freie UDF-Funktionen von InterBase
- F_LOG2 -> s. ib_util, freie UDF-Funktionen von InterBase
- F_LOGN -> s. ib_util, freie UDF-Funktionen von InterBase
- F_PI -> s. ib_util, freie UDF-Funktionen von InterBase
- F_COS -> s. ib_util, freie UDF-Funktionen von InterBase
- F_SIN -> s. ib_util, freie UDF-Funktionen von InterBase
- F_TAN -> s. ib_util, freie UDF-Funktionen von InterBase
- F_COTAN -> s. ib_util, freie UDF-Funktionen von InterBase
- F_HYPOT -> s. ib_util, freie UDF-Funktionen von InterBase

- F_NOT
- F_AND
- F_OR
- F_XOR
- F_SHL
- F_SHR
- F_BLOBCOMP -> s. F_BLOBCOMPARE
- F_BLOBICOMP -> s. F_BLOBCOMPARE mit weiteren Funktionen
- F_BLOBUPPER
- F_BLOBWRAP
- F_BLOBISEMPTY

Möglichkeit, UDFs auszutauschen TROTZ Abhängigkeiten

nach oben

Wenn man auf die FreeAdhocUDF wechseln möchte ergibt sich häufig die Problematik, dass sich die vorhandenen gleichlautenden UDFs wegen Abhängigkeiten (sie werden in ComputedBy-Felder, Triggern, Views oder Proceduren verwendet) nicht löschen lassen (bis InterBase 6 ging dies).
Dafür gibt es einen (nicht ganz sauberen aber ungefährlichen) Trick:

```
/* UDF-Abhängigkeiten vorrübergehend ausschalten (hier für alle die mit F_ beginnen) */
UPDATE RDB$DEPENDENCIES SET RDB$DEPENDED_ON_NAME = 'x' ||
RDB$DEPENDED_ON_NAME
WHERE RDB$DEPENDED_ON_TYPE = 15
AND RDB$DEPENDED_ON_NAME STARTING WITH 'F';
/* UDF-Funktionen nun entfernen */
DROP  EXTERNAL FUNCTION F_.....;
DROP  EXTERNAL FUNCTION F_.....;
...
/* jetzt FreeAdhocUDF Funktionen über deren Script hinzufügen */
...
/* dann die (alten) Abhängigkeiten wiederherstellen */
UPDATE RDB$DEPENDENCIES SET RDB$DEPENDED_ON_NAME =
F_MID(RDB$DEPENDED_ON_NAME, 1,
F_STRINGLENGTH(RDB$DEPENDED_ON_NAME))
WHERE RDB$DEPENDED_ON_TYPE = 15
AND RDB$DEPENDED_ON_NAME STARTING WITH 'xF';
```