We are looking for someone, who can translate this documentation better we can :-)

Bugreports and wishes please only to UDF@adhoc-data.de

Index of contents

- Compare
  - F_EQUALSTRING
- Search / Identify
  - F_FINDWORD
  - F_FINDNTHWORD
  - F_FINDWORDINDEX
  - F_STRINGLENGTH
  - F_STRINGLISTITEM

Math functions (31 functions)

- Transform
  - F_CONVERTFROM33
  - F_CONVERTTO33
  - F_CONVERTFROMBASE
  - F_CONVERTTOBASE
  - F_HEXTOINT
  - F_INTTOHEX
  - F_DEGTORAD
  - F_RADTODEG
- Formate
  - F_DOLLARVAL
  - F_CONVERTTODOLLAR
  - F_FIXEDPOINT
  - F_FIXEDPOINTLANG
  - F_ROUND
  - F_ROUNDFLOAT
  - F_TRUNCATE
  - F_ZAHLRUNDEN
- Calculate
  - F_ABS
  - F_DOUBLEABS
  - F_INTEGERABS
  - F_FACT
  - F_FACTORIAL
  - F_MODULO
  - F_PROZENTE
- Compare
  - F_EQUALFLOAT
  - F_EQUALINTEGER
  - F_MAXNUM
  - F_MAX
  - F_MINNUM
  - F_MIN
  - F_ISDIVISIBLEBY

Date-time functions (66 functions)

- Functions to calculate with date-time
  - F_ADDYEAR
  - F_ADDMONTH
  - F_ADDWEEK
  - F_ADDDAY
  - F_ADDHOUR
  - F_ADDMINUTE
  - F_ADDSECOND
- Functions to calculate the difference of two dates-times
  - 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

Functions for the (formatted) output of date-time

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

Functions for testing (two) dates-times

F_EQUALDATE
F_EQUALDATETIME
F_ISLEAPYEAR
F_MAXDATE
F_MINDATE

1. Compatibility between windows and Linux to portate databases by backup/Restore. In addition belong both deliver the same function names as well as the same entrypoints as well as that the Functions the same results.
   The module-name in Windwos is named “FreeAdhocUDF.dll” in Linux “FreeAdhocUDF.so”. The “module name” in the SQL-definition is always named “FreeAdhocUDF”, windows surge automatically the FreeAdhocUDF.dll, Linux the FreeAdhocUDF.so
   In Linux you must pay attention to large-/little spelling of the module name as well as the name of the Lib.
2. Compatibility between the “original” FreeUDFLib (in Delphi, 1998 of Gregory Deatz), the FreeUDFLibC (to C portierte Delphi-version, 1999 of Gregory Deatz), the FreeUDFLib of AvERP (in Delphi, contains some expansions) and the GrUDF (in Delphi and Kylix 2004 of Torsten Grundke and Gerd Kroll) to such an extent that into projects with one of the four UDFs these can be replaced by the FreeAdhocUDF.
3. The original UDFs were corrected, where in windows something incorrect, in Linux something came out correct. These corrections do not be available in the “old” FreeUDFLib but until the new FreeAdhocUDF.
4. The original UDFs were corrected at, came out where in windows what other as in Linux.
   - F_AGEINWEEKS
     previous function of the FreeUDFLibC makes the distance in days, divides through 7 and rounds off then to. Leads in 20.08.2004 to 21.08.2004 to 1 instead of 0 in windows (FreeUDFLib). Changed function now like in windows
5. Optimization of the C-code, in part completely newly written.
6. At we startet FreeAdhocUDF was tested on
   - InterBaseSS 6.02 under Windows XP Professional and Windows 2000 Advanced Server
   - InterBase 7.1 under Windows XP Professional and Windows 2000 Advanced Server
   - InterBase 7.1 under Mandrake Linux 10.0 / Mandriva Linux 10.1
   - InterBase 7.5 under Windows2000 Advanced Server
   - InterBase 7.5 under Mandriva Linux 10.2
   - FireBirdSS 1.5.2 under Windows XP Professional and Windows 2000 Advanced Server
   - FireBirdSS 1.5.2 under Mandrake Linux 10.0 / Mandriva Linux 10.1
   - FireBirdSS 1.5.2 under SuSe Linux 8.1 (use FreeAdhocUDF.so SuSe81FB15 !)
   - FireBirdSS 1.5.2 under SuSe EnterpriseServer 8 (use FreeAdhocUDF.so SuSe81FB15 !)
   - FireBirdSS 1.5.2 under SuSe Linux 10.0
   - FireBirdSS 1.5.2 under Ubuntu Server 5.10
   - FireBirdSS 1.5.2 under Kubuntu 5.10
   - FireBirdSS 2.0.0 RC2 under Windows 2000 Advanced Server
   - FireBirdSS 2.0.0 RC2 under Mandriva Linux

   New versions are tested look under versions

   The attached *.so are allways renamed to “FreeAdhocUDF.so” for use.

Version (without VersionsNo.) of 2004-08-23
produced compatibility between windows and Linux for InterBase and was Compatible with the FreeUDFLib and the FreeUDFLibC. It contained moreover some new Functions. .

- 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" of 2005-10-16
manufactured compatibility between Windows and Linux for InterBase and FireBird and is compatible to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP and GrUDF. It contains new Functions, which did not give it so far in one of the four UDFs, which are however useful.

- 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" from 2005-12-31
bugfixed on following bugs:
- F_OSTERDATUM (calulates Thuesday after Easter insted of Easter-Sunday)
- UDF-definitionscript (wrong Entrypoint of F_MAX and F_MIN)
- update documentation

Version "adhoc 20060302" from 2006-03-02
adds the following:
- F_STRINGLISTITEM
- F_LF
- F_STR2EXCEL
- F_BLOB2EXCEL
- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBREPLACESTRING
- corrects a bug in F_ADDYEAR, F_ADDMONTH, F_ADDWEEK, F_ADDDAY, F_ADDHOUR, F_ADDMINUTE, F_ADDSECOND
- extended version of F_FINDWORD and F_FINDNTHWORD additional character (äöü...)
- update of the manual

Version “adhoc 20060306" from 2006-03-06
now contains two versions for F_SUBSTR.

Version “adhoc20060516" from 2006-05-16
- addapted/tested with InterBase 7.5
- works from FireBird 2.0 RC1 on (compiled with RC2)
- for Windows from FireBird 1.5 only one FreeAdhocUDF-FireBird version is necessary .
- for Windows from InterBase 6 only one FreeAdhocUDF-FireBird version is i necessary
- only for Linux and FireBird 1.5 a version a special version is necessary
- for Linux and InterBase just one version of FreeAdhocUDF is necessary
- no more support for Interbase 6 and SuSe 8.1
- corrects a bug in F_AGEINYEARSTHRESHOLD (Parameters changed)
- corrects a bug in F_ADDMONT
- error correction and update of the manual

This version was testet on
- InterBaseSS 6.02 under Windows XP Professional and Windows 2000 Advanced Server
- InterBase 7.1 under Windows XP Professional and Windows 2000 Advanced Server
- InterBase 7.1 under Mandrake Linux 10.0 / Mandriva Linux 10.1
- InterBase 7.5 under Windows2000 Advanced Server
- InterBase 7.5 under Mandriva Linux 10.2
- FireBirdSS 1.5.2 under Windows XP Professional and Windows 2000 Advanced Server
- FireBirdSS 1.5.2 under Mandrake Linux 10.0 / Mandriva Linux 10.1
- FireBirdSS 1.5.2 under SuSe Linux 8.1 (use FreeAdhocUDF.so SuSe81FB15 !)
- FireBirdSS 1.5.2 under SuSe EnterpriseServer 8 (use FreeAdhocUDF.so SuSe81FB15 !)
- FireBirdSS 1.5.2 under SuSe Linux 10.0
- FireBirdSS 1.5.2 under Ubuntu Server 5.10
- FireBirdSS 1.5.2 under Kubuntu 5.10
- FireBirdSS 2.0.0 RC2 under Windows 2000 Advanced Server
- FireBirdSS 2.0.0 RC2 under Mandriva Linux

Version “adhoc 20060919 from 2006-09-19
Interims-version - not published
Workaround for bug (Server-hangup) for InterBase during
- F_BLOBASPCHAR
- F_BLOBCAT

- F_BLOBCATSTR

Version “adhoc 20060925" from 2006-09-25
corrects bug (also server-hangup) for InterBase and FireBird during:
- F_BLOBASPCHAR
- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBREPLACESTRING

corrects bug for Linux
- F_DAYSOFMONTHS

adds:
- F_BLOBMAXSEGMENTLENGTH
- F_BLOBSEGMENTCOUNT
- F_BLOBLINE
- F_BLOBLINE_COUNT
- F_BLOBCOMPARE
- F_BLOBSUBSTR

update of the manual
(new functions) tested on
- InterBase 7.1 SP2 under Windows 2000 Advanced Server
- InterBase 7.1 SP2 under Mandriva 2006
- InterBase 7.5 SP1 under Windows XP Professional
- InterBase 7.5 SP1 under Mandriva 2006
- FireBird 1.5.2 under SuSe 8.1
- FireBird 1.5.2 under SuSe 10.0
- FireBird 1.5.2 under Mandriva 2006
- FireBird 2.0 RC4 under Windows XP Professional
- FireBird 2.0 RC4 under Mandriva 2006

Version “adhoc 20061020" from 2006-10-31
adds:
- F_NBSP
- F_DQM
- F_SQM
- F_TAB
- F_GETRANDOM
- F_GETUUID
- F_GETGUID
- F_UUID2GUID
- F_GUID2UUID
- F_UUIDTIMESTAMP
- F_GUIDTIMESTAMP
- F_UUIDMAC
- F_GUIDMAC

update of the manual
(new functions) tested on
- InterBase 7.1 SP2 under Windows 2000 Advanced Server
- InterBase 7.1 SP2 under Mandriva Linux 2006
- InterBase 7.5 SP1 under Windows XP Professional
- InterBase 7.5 SP1 under Mandriva Linux 2006
- InterBase 2007 under Windows XP Professional
- FireBird 1.5.2 under Windows 2000 Advanced Server
- FireBird 1.5.2 under SuSe 10.0
- FireBird 1.5.2 under Mandriva Linux 2006
- FireBird 2.0 RC5 under Windows XP Professional
- FireBird 2.0 RC5 under 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.

preliminary note

There are various limits for Interbase and for Firebird, as well as between their versions.
The data size of columns in a SQL query cannot exceed the
column limit AND the row limit. The full row size of the SQL cannot exceed the maximum row size limit.
The size of a query is the sum of each defined column sizes. Not the sum of the real field content:
- a column with ist defined size - for example VARCHAR(100), 2 bytes for SMALLINT etc.
- a UDF with the defined output-value - for example 100 bytes for ...RETURN VARCHAR(100)...
- a StroredProcedure s.o.

	column limit	row 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

In order that the following construction works:
select F_REPLACE(F_REPLACE(F_REPLACE(F_REPLACE('abcdefg', 'a', 'x'), 'b', 'y'), 'c', 'z'), 'd', '-') from TABLE...
The definition of the F_REPLACE must looks like:

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

The return string cannot be larger than 8190 bytes, because the function is used 4 times in ONE SQL
-> 4 * 8190 = 32760 bytes
Should the SQL contain also table-columns like
SELECT COLUMN, F_REPLACE(F_REPLACE(F_REPLACE(....
may the F_REPLACE appear just 3 times for don't exceeding the limit of 32kB.
Due to the fact that you want to use the UDF in many ways for each project, it's possible to define one UDF multiple :
- F_REPLACE mit RETURN CSTRING(254)
- F_REPLACE4 mit RETURN CSTRING(4095)
- F_REPLACE8 mit RETURN CSTRING(8190)
- F_BIGREPLACE mit RETURN CSTRING(32760)

So you have the right UDF for all circumstances.

String-Functions - Conversions

F_CHARACTER
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input INT
Output CSTRING(2)
Parameter 1 ASCII-Code
The sign of the ASCII-code of parameter 1 issues
TestSQL
SELECT 'B' AS ISCORRECT, F_CHARACTER(66) FROM RDB$DATABASE
SELECT 'ä' AS ISCORRECT, F_CHARACTER(228) FROM RDB$DATABASE

F_CHR
Compatibility to GrUDF
identically to F_CHARACTER

F_ENCRYPTMD5
Compatibility to FreeUDFLibC
Input CSTRING(128)
Output CSTRING(33)
Parameter 1 String that is supposed to be coded.
Coded after the MD5 algorithm.
TestSQL
SELECT 'e7d31845480111fdba3316129e166860' AS ISCORRECT, F_ENCRYPTMD5('Pauline') FROM RDB$DATABASE;

F_PROPERCASE
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, which is to be converted
It converts the string concerning upper and lower case printing capability in such a way that each word starts with a capital letter and all other letters are small letters. Also special character (not only German ä ö ü) changes. A small 'ß' remains naturally a small '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
Determines the one German-phonetic string of parameter 1. Can serve e.g. for the doublet search.
TestSQL
SELECT 'MAYR' AS ISCORRECT, F_GSOUNDEX('Meier'), F_GSOUNDEX('Maier'), F_GSOUNDEX('Mayer') FROM RDB$DATABASE

F_GENERATESNDXINDEX
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input CSTRING(8190)
Output CSTRING(6)
Parameter 1 String
Determines the (original -) Soundex of the string.
Soundex is a phonetic algorithm to indexation of words and cliches after its sound in the English language. Equalequivalent words are to be coded thereby to an identical character sequence.
Considered starting from the 2. place counted only the first 6 consonants with the determination, whereby repeated occurrence is only once considered (in the example the l,m,y,w,r,d). Therefore it is unsuitable, around longer strings to compare.
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
Function of adhoc
identically to F_GENERATESNDXINDEX

F_ANSILOWERCASE
Compatibility to FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, which is to be converted
All indications in small letters change around inclusive special characters.
TestSQL
SELECT 'schöner tag' AS ISCORRECT, F_ANSILOWERCASE('SchÖner TAG') FROM RDB$DATABASE
annotation:
This function is no more necessary for FireBird 2, because of the LOWER(..) which handle special characters right, too.

F_LOWER
Function of adhoc
identically to F_ANSILOWERCASE

F_ANSIUPPERCASE
Compatibility to FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, which is to be converted
All indications convert inclusive into capital letters. Special character (in contrast to the SQL instruction UPPER(...))
TestSQL
SELECT 'SCHÖNER TAG' AS ISCORRECT, UPPER('Schöner Tag'), F_UPPER('Schöner Tag') FROM RDB$DATABASE
annotation:
This function is no more necessary for FireBird 2, because of the UPPER(..) which handle special characters right, too

F_UPPER
Function of adhoc, identically to ANSIUPPERCASE

F_TELEFONNR
Function of adhoc

Input CSTRING(32760), INT
Output CSTRING(32760)
Parameter 1 String, some telephone No. (or something similar, from which everything is to be removed up to the numbers) contains.
Parameter 2 defines the number of numbers at the end, which are replaced through *
Removes all nonnumerical and blank from the telephone number. "49" stands it by "+49" is replaced at the beginning. If the string starts with one "+", this remains.
Note: This function serves for it, in order in a string stored and "wildly" formatted telephonenumber. for TAPI to a Telephfone(system) to hand over (this "intelligence" does not possess)
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
Compatibility to GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String 1
Remove all nonnumerical indications from string 1 (e.g. for TAPI)
TestSQL
SELECT '0232653535' AS ISCORRECT, F_DIGITS(' (0232) / 6535-35') FROM RDB$DATABASE

F_STR2EXCEL
Function from adhoc
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 string to convert for Excel.
To convert multiline texts and texts with converted commas to Excel, it is required to transform the string.
This fuction will do the following:
- appends a double inverted comma at the beginning and the end of the string
- doubles all inverted commas
- deletes all CHR(13) in the string
- limits the input string to 32760 characters (Limitation in Excel)

TestSQL
SELECT '"1.Zeile ""Paul"" und' || F_LF() || '2.Zeile"' AS ISCORRECT, F_EXCELSTRING('1.Zeile "Paul" und' || F_CRLF() || '2.Zeile') FROM RDB$DATABASE
Note:
Actually it doesn't make any sense to export a very large text from the database to excel cell,
this function may be practicable in combinatin with F_LEFT or F_RIGHT.
For instance:
SELECT F_RIGHT(F_STR2EXCEL(STRFeldTAGEBUCH), 1000) FROM ...export the last 1000 charachters of the field "STRFeldTAGEBUCH"

F_COLLATEBR
Compatibility to GrUDF
Input CSTRING(32760)
Output CSTRING(32760)
Parameter 1 String, where the special characters are to be converted
Changes intended "umlauts" for given indications
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
Note:
in the difference to the GrUDF does not change Ÿ after Y over (in Linux missing?)!
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
Compatibility to GrUDF
Input CSTRING(8190), INT, INT
Output CSTRING(8190)
Parameter 1 String, from which a character string is to be determined
Parameter 2 Position, at which the string which can be determined starts
Parameter 3 Length of string which can be determined
The number of letters (parameter 3) shows entered text starting from the entered letter number (parameter 2). Counting starts for parameter 2 with 1. Same function as F_MID. Counting starts for F_COPY with 1, not with 0 as for F_MID.
TestSQL
SELECT 'tag' AS ISCORRECT, F_COPY('Geburtstagsparty', 8, 3) FROM RDB$DATABASE;

F_EUROVAL
Function of adhoc
Input DOUBLE
Output CSTRING(254)
Parameter 1 Floating Point
Set EUR after parameter 1
TestSQL
SELECT '15.47 EUR' AS ISCORRECT, F_EUROVAL(15.47) FROM RDB$DATABASE

F_HOLDSTRING
Compatibility to FreeUDFLibC
Input CSTRING(32760), CSTRING(254)
Output CSTRING(32760)
Parameter 1 String 1

Parameter 2 String 2 (The sequence of the indications in the string 2 does not play a role.)
Identically to F_STRIPSTRINGHOLD
Counterpart to F_STRIPSTRING
Note:
In a SQL F_HOLDSTRING and F_STRIPSTRINGHOLD may not use at the same time.
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GruDF
Input CSTRING(8190), INT
Output CSTRING(8190)
Parameter 1 String
Parameter 2 Length of the string which can be spent
The string cut on the number indication from left gives from parameter 2 out.
Counting starts with 1
TestSQL
SELECT 'Dies i' AS ISCORRECT, F_LEFT('Dies ist ein Test', 6) FROM RDB$DATABASE

F_LINEWRAP
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input CSTRING(32760), INT, INT
Output CSTRING(32760)
Parameter 1 String
Parameter 2 Startposition
Parameter 3 Measure
Spends all words of the string, begin with the initial position, which is together not longer than the measure. Counting starts with 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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String, whose blanks are to be removed at the beginning
Remove all simple blanks at the beginning of the string, not the protected blanks (with < ALT > < 255 > entered)
TestSQL
SELECT 'Dies ist ein Test' AS ISCORRECT, F_LTRIM(' Dies ist ein Test') FROM RDB$DATABASE
F_LRTRIM / F_BIGLRTRIM
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String, whose blanks are to be removed at the beginning and at the end.
Remove all simple blanks at the beginning and at the end of the string. Entfernt nicht die geschützten Leerzeichen (mit <ALT> <255> eingegebene)
F_LRTRIM und F_BIGLRTRIM dürfen nicht in einem SQL verwendet werden.
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

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(8190), INTEGER, INTEGER
Output	CSTRING(8190)
Parameter 1	String, from which a character string is to be determined
Parameter 2	Startposition of the string which can be determined
Parameter 3	Length of the string which can be determined

The number of letters (parameter 3) shows entered text starting from the entered letter number (parameter 2). Counting starts for parameter 2 with 0.

Comparisons F_COPY

TestSQL

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

F_PADLEFT

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(32760), CSTRING(16), INT
Output	CSTRING(32760)
Parameter 1	String 1 (with the indications from string 2 to the length of parameter 3 to be filled up)
Parameter 2	String 2 (with the string 1 to be filled up)
Parameter 3	Length of the string, up to which is to be filled up

Fills up the string 1 left from the string with the indications from string 2 to the overall length of indications of parameter 3.

If you enter more than one character in string 2, the filling with characters of string 2 starts from right and abort, if the required number of the complete characters will be reached. (Look at. 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

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(32760), CSTRING(16), INT
Output	CSTRING(32760)
Parameter 1	String1 (with the indications from string 2 to the length by parameter 3 to be filled up)
Parameter 2	String 2 (with the string 1 to be filled up)
Parameter 3	Length of the stringer, up to which is to be filled up

Fills up the string 1 right from the string with the indications from string 2 to the overall length of indications of parameter 3.

If you enter more than one character in string 2, the filling with characters of string 2 starts from right and abort, if the required number of the complete characters will be reached. (Look at. 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

Compatibility to FreeUDFLibC

Input	CSTRING(32760), CSTRING(254), CSTRING(254)
Output	CSTRING(32760)
Parameter 1	the string, as a character string is to be exchanged
Parameter 2	the string which can be exchanged
Parameter 3	the string which can be set

Replace all charakters from parameter 2 through the charakters from parameter 3

Easy version of the funktion F_REPLACESTRING only without the chance to change the string only onetimes and independent of upper and lower case.

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
Compatibility to FreeUDFLib AvERP, GrUDF,
Input CSTRING(32760), CSTRING(254), CSTRING(254), INT, INT
Output CSTRING(32760)
Parameter 1 the string, as a character string is to be exchanged
Parameter 2 the string which can be exchanged
Parameter 3 the string which can be set
Parameter 4 0 = only exchange the first occurrence, 1 = all occurrences exchange
Parameter 5 0 = consider upper and lower case, 1 = not consider
Replaced in a string one/all character string/s from parameter 2 by another character string from parameter 3 and knows also upper-lower case.
Considers replaced in a string a character string by another
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190), INT
Output CSTRING(8190)
Parameter 1 String
Parameter 2 Numbers of charakters from right
Return the Number rightmost characters from parameter 2. Counting starts by 1.
TestSQL

```
SELECT 'n Test' AS ISCORRECT, F_RIGHT('Dies ist ein Test', 6) FROM RDB$DATABASE;
```

F_RTRIM
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190)
Output CSTRING(8190)
Parameter 1 String, whose right whitespace ought to clear
Trim the right side of parameter 1 of all whitespace. In other words, trim all trailing whitespace.(with <ALT> <255> entered)
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(32760), CSTRING(254)
Output CSTRING(32760)
Parameter 1 String 1 (from that all indications from string 2 to be removed)
Parameter 2 String 2 (Indications to be removed)
Strip string 1 of all characters listed in string 2, the sequence of charakters in string 2 is irrelevant.
Analog to 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
Compatibility to ????, identically to F_HOLDSTRING
Input CSTRING(32760), CSTRING(254)
Output CSTRING(32760)
Parameter 1 String 1 (from that all indications from string 2 to be removed)
Parameter 2 String 2 (Indications to be removed)
Strip string 1 of all characters listed in string 2, the sequence of charakters in string 2 is irrelevant
Differently expressed: gives only the indications from string 1 (in the sequence of string 1), which are indicated in string 2.
Counterpart to F_STRIPSTRING
In a SQL may be used not at the same time F_HOLDSTRING and F_STRIPSTRINGHOLD.
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input CSTRING(8190), CSTRING(254)
Output INT
Parameter 1 String1 (in that the position by parameter 2 to be determined)
Parameter 2 String 2 (its position in parameter 1 to be determined)
Spends the first position in the string 1, with which the string 2 starts.
Counting starts with 0.
TestSQL
SELECT 9 AS ISCORRECT, F_SUBSTR('Pauline fährt in Urlaub', 'ähr') FROM RDB$DATABASE
SELECT 4 AS ISCORRECT, F_SUBSTR('Pauline fährt in Urlaub', 'i') FROM RDB$DATABASE

F_SUBSTR / F_BIGSUBSTR

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(8190), CSTRING(1024)
Output	INT
Parameter 1	String1 (in that the position by parameter 2 to be determined)
Parameter 2	String 2 (its position in parameter 1 to be determined)

Spends the first position in the string 1, with which the string 2 starts.
Counting starts with 0.
Notice:
In the original FreeUDFLib (1998 by Gregory Deatz) the order of the input-parameters are swapped to the other string-funktions and to the c-version (1999 by Gregory Deatz) of the FreeUDFLibC. To maintain backward compatibility for both versions there are two different “entrypoints” you can use in the DECLARE-Script of F_SUBSTR.
To be compatible with the (Delphi-)FreeUDFLib, the FreeUDFLib AvERP and the GrUDF (which is compatible to the Delphi-FreeUDFLib) use the entrypoint “strsub”, for compatibility to FreeUDFLibC use 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

Compatibility to ????

Input	CSTRING(8190), INT, INT
Output	CSTRING(8190)
Parameter 1	String, from which a character string is to be determined
Parameter 2	Position, at which the string which can be determined starts
Parameter 3	Length of the string which can be determined

The number of letters (parameter 3) shows entered text starting from the entered letter number (parameter 2). Counting starts for parameter 2 with 1. Same function as F_MID and F_COPY. Counting starts with 1.
TestSQL

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

F_STRRM

Compatibility to ????

Input	CSTRING(8190), INT
Output	CSTRING(8190)
Parameter 1	String, from which a place is to be removed
Parameter 2	Place in the string, which is to be removed (counting starts with 0)

Removes the indication in the entered place of the entered string.
TestSQL

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

F_CRLF
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input without
Output CSTRING(3)
Parameter 1 without -> “()”
Carriage Return and Linefeet - produce the indications CHR(13) + CHR(10)
TestSQL
SELECT ' ABC' || F_CRLF() || '123' FROM RDB$DATABASE;
Ergebnis: ‘ABC’ (1. line)
‘123’ (2. line)
SELECT 'first line' || F_CRLF() || 'second line' FROM RDB$DATABASE;

F_LF
Function from adhoc
Input none
Output CSTRING(2)
Parameter 1 without “()”
Line feed. Creates the character CHR(10)
Identical to F_CHARACTER(10).
TestSQL
SELECT 'ABC' || F_LF() || '123' FROM RDB$DATABASE;

F_SPACE
Compatibility to GrUDF
Input INTEGER
Output CSTRING(32760)
Parameter 1 Number of blanks
Returns a blank chain (CHR(32)) with certain indicated number.
TestSQL
SELECT F_STRINGLENGTH(F_SPACE(10)) || ' Leerzeichen' AS ISCORRECT, F_SPACE(10) FROM RDB$DATABASE;

F_STROFCHAR
Compatibility to GrUDF
Input CSTRING(1), INT
Output CSTRING(32760)
Parameter 1 String, which is to be repeated
Parameter 2 Number of repetitions
Returns a string with the indicated number of repeated indications.
TestSQL
SELECT F_STRINGLENGTH(F_STROFCHAR('A', 10)) || ' mal A' AS ISCORRECT,
F_STROFCHAR('A', 10)
FROM RDB$DATABASE;

F_NBSP
Function from adhoc
Input none
Output CSTRING(2)
Parameter 1 without “()”
Non-braking-space. Creates the character CHR(160)
Identical to F_CHARACTER(160).
TestSQL
SELECT 'ABC 123' AS ISCORRECT, 'ABC' || F_NBSP() || '123' FROM RDB$DATABASE;

F_DQM
Function from adhoc
Input none
Output CSTRING(2)
Parameter 1 without “()”
Doube-quote-mark. Creates the character CHR(34)
Identical to F_CHARACTER(34).
TestSQL
SELECT 'ABC"123' AS ISCORRECT, 'ABC' || F_DQM() || '123' FROM RDB$DATABASE;

F_SQM
Function from adhoc
Input none
Output CSTRING(2)
Parameter 1 without “()”
Single-quote-mark. Creates the character CHR(39)
Identical to F_CHARACTER(39).
TestSQL
SELECT 'ABC<einfaches Hochkoma>123' AS ISCORRECT, 'ABC' || F_SQM() || '123' FROM RDB$DATABASE;

F_TAB
Function from adhoc
Input none
Output CSTRING(2)
Parameter 1 without “()”
Tabulator. Creates the character CHR(9)
Identical to F_CHARACTER(9).
TestSQL
SELECT 'ABC<TAB>123' AS ISCORRECT, 'ABC' || F_TAB() || '123' FROM RDB$DATABASE;

F_EQUALSTRING
Compatibility to FreeUDFLib AvERP, GrUDF,
Input CSTRING(8190), CSTRING(8190)
Output INT
Parameter 1 String 1
Parameter 2 String 2
Examined if string 1 equal to string 2.
Ergebnis 1 = is equal, 0 = is not equal
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input CSTRING(32760), INT
Output CSTRING(254)
Parameter 1 String, in which another string is to be looked for.
Parameter 2 Position, at which the search in the string should starts.
Returns a word or a partial word , which stands to the accordingly indicated Positon. Positioning starts here with 0, that means 1.position = 0 and searches up to the first blank.
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
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input CSTRING(32760), INT
Output CSTRING(254)
Parameter 1 String, in which another string is to be looked for
Parameter 2 String-number, which to be searched (nth word)
Returns the nth word. Counting starts at 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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input CSTRING(32760), INT
Output INT
Parameter 1 String
Parameter 2 length of the string which should be examined
It examines whether the position from Paramter 2 exists in the string.
If the poition is found in the string, the result is the position itself, if not found the result is -1.
Counting starts at 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

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Input	CSTRING(32760)
Output	INT
Parameter 1	String, whose length is to be determined

Determines the length of a stringer. Considered with it also simple blanks at the end..

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

Compatible to GrUDF

Input	CSTRING(32760), CSTRING(254)
Output	CSTRING(1024)
Parameter 1	stringlist formatted like ”n=..”,”m=..”, u.s.w.
Parameter 2	Value from the list to be shown

Searchs the string, build like a stringlist (Name=Value), for the Name and returns the complete Name=Value

TestSQL

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

Math Functions - Convert

F_CONVERTFROM33
Compatibility to FreeUDFLibC
Input CSTRING(254)
Output INT
Parameter 1 Number in the 33- number-system, which is to be converted as string
Converts a number in the 33- number-system into the decimal-system
TestSQL
SELECT 1000, F_CONVERTFROM33('WB') FROM RDB$DATABASE;

F_CONVERTTO33
Compatibility to FreeUDFLibC
Input INT
Output CSTRING(254)
Parameter 1 Number in the decimal-system, which is to be converted into the 33- number-system.
Converts a decimal-number into the 33- number-system
TestSQL
SELECT 'WB', F_CONVERTTO33(1000) FROM RDB$DATABASE;

F_CONVERTFROMBASE
Input CSTRING(32), INT, CSTRING(8)
Output INT
Parameter 1 Number of a number-system, which is to be converted as string.
Parameter 2 Basis of the number which can be converted (for example 2 for binary-system).
Parameter 3 all numbers, which are valid in the number-system as string (e.g. ‘01234567' for oktal-system).
Converts a number of any number-system into the decimal-system
TestSQL
SELECT 3, F_CONVERTFROMBASE('11', 2, '01') FROM RDB$DATABASE;
SELECT 9, F_CONVERTFROMBASE('11', 8, '01234567') FROM RDB$DATABASE;

F_CONVERTTOBASE
Compatibility to FreeUDFLibC
Input INT, INT, CSTRING(254)
Output CSTRING(254)
Parameter 1 Dezimal-number, which is to be converted
Parameter 2 Basis of system, into which is to be converted(e.g. 2 for binäry-system)
Parameter 3 all numbers, which are valid in the number-system as string (e.g. ‘01234567' for oktal-system).
Converts a decimal-number into any number-system
TestSQL
SELECT '11', F_CONVERTTOBASE(3, 2, '01') FROM RDB$DATABASE;

F_HEXTOINT
Compatibility to GrUDF
Input CSTRING(20)
Output INTEGER
Parameter 1 Hexworth
Change a Hexworth into a Integerworth.
TestSQL
SELECT 1 AS ISCORRECT, F_HEXTOINT('0000000001') FROM RDB$DATABASE;

F_INTTOHEX
Compatibility to GrUDF
Input INTEGER, INTEGER
Output CSTRING(254)
Parameter 1 integer which can be converted
Parameter 2 amount of digits (left filled of with 0)
Change an Intergerword into a Hexword.
TestSQL
SELECT ',0000000001' AS ISCORRECT, F_INTTOHEX(1,10) FROM RDB$DATABASE;

F_DEGTORAD
Compatibility to GrUDF
Input DOUBLE
Output DOUBLE
Parameter 1 Angle in (Alt-)dergee
Changes an angle from (Alt-)degree in radian (radian measure).
TestSQL
SELECT '3.1415.. = PHI' AS ISCORRECT, F_DEGTORAD(180) FROM RDB$DATABASE;

F_RADTODEG
Compatibility to GrUDF
Input DOUBLE
Output DOUBLE
Changes an angle from radian (radian measure) in (Alt-)degree.
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

Compatibility to FreeUDFLib, FreeUDFLib AvERP

Input	DOUBLE
Output	CSTRING(32)
Parameter 1	Floating Point

Converts a floatingpoint value into a dollar value (rounded on 2 right-of-comma positions).

TestSQL

```
SELECT '$15.68' AS ISCORRECT, F_CONVERTTODOLLAR(15.678), F_DOLLARVAL(15.678)
FROM RDB$DATABASE;
```

F_CONVERTTODOLLAR

Compatibility to FreeUDFLibC

Identically to F_DOLLARVAL.

F_FIXEDPOINT

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP

Input	DOUBLE, INT
Output	CSTRING(32)
Parameter 1	Floating Point which can be rounded
Parameter 2	Number of places, on which is to be rounded.

Rounds the Floating Point on number of places (parameter 2).

In the string-expenditure is the decimal-separator ".".

TestSQL

```
SELECT '12345.5' AS ISCORRECT, F_FIXEDPOINT(12345.46, 1) FROM RDB$DATABASE
```

F_FIXEDPOINTLANG

Function of adhoc

Input	DOUBLE, INT , CSTRING(1), CSTRING(1)
Output	CSTRING(32)
Parameter 1	Floatingpoint which can be rounded
Parameter 2	Number of places, on which is to be rounded.
Parameter 3	Indication of decimal-separation
Parameter 4	Indication of thousands seperator

Rounds the floatingpoint to the number of places from Parameter 2 with definable decimal- and thousands seperator

TestSQL

```
SELECT '12.345,5' AS ISCORRECT, F_FIXEDPOINTLANG(12345.46, 1, ',', '.') FROM
RDB$DATABASE
```

F_ROUND

Compatibility to FreeUDFLibC

Input	DOUBLE
Output	INT
Parameter 1	floatingpoint to be rounded to an integer

Rounds the floatingpoint to a integer value.

TestSQL

```
SELECT 16 AS ISCORRECT, F_ROUND(15.567) FROM RDB$DATABASE
```

F_ROUNDFLOAT
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 floatingpoint to be rounded
Parameter 2 outputformate for floatingpoint valuel (e.g. 0.01); rounds on the next multiple of the 2. Parameter
Rounds a floatingpoint.
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input DOUBLE
Output INT
Parameter 1 floatingpoint to be cut off
Cuts the right-of-comma positions from a floatingpoint.
TestSQL

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

F_ZAHLRUNDEN
Compatibility to FreeUDFLib AvERP, GrUDF
Input DOUBLE, INT
Output DOUBLE
Parameter 1 to rounding value
Parameter 2 Number of places, on which is to be rounded
Rounds the floatingpoint value to the entered number of places.
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
Compatibility to GrUDF
Identically to F_DOUBLEABS
F_ABS and F_DOUBLEABS may not be used in the same SQL.

F_DOUBLEABS
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input DOUBLE
Output DOUBLE
Parameter 1 to calculat the Floating Point
Determines the absolute value (more positively value) of a Floating Point
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
Compatibility to FreeUDFLibC
Input INT
Output INT
Parameter 1 integer which can be computed.
Determines the absolute value (positive value) of an integer.
SELECT 15 AS ISCORRECT, F_INTEGERABS(15) FROM RDB$DATABASE;
SELECT 15 AS ISCORRECT, F_INTEGERABS(-15) FROM RDB$DATABASE;

F_FACT
Function of adhoc
Input INT
Output DOUBLE
Parameter 1 Integer, by which the faculty is to be formed.
Computes the faculty (the product of all natural numbers from 1 to the argument)
TestSQL
SELECT 6.000 AS ISCORRECT, F_FACT(3) FROM RDB$DATABASE

F_FACTORIAL
Compatibility to GrUDF
Identically to F_FACT
F_FACT and F_FACTORIAL may not be used in the same SQL.

F_MODULO
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF,
Input INT, INT
Output INT
Parameter 1 Divident (the number, which will divided)
Parameter 2 Divisor (the number, by which one divides)
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
Compatibility to FreeUDFLib AvERP, GrUDF
Input DOUBLE, DOUBLE
Output DOUBLE
howmuch % the 2. from 1.
e.g. selling price to purchase price
Test-SQL
SELECT 14.000 AS ISCORRECT, F_PROZENTE(100.00, 14.0) FROM RDB$DATABASE

F_EQUALFLOAT
Compatibility to FreeUDFLib AvERP, GrUDF
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 floatingpoint value 1
Parameter 2 floatingpoint value 2
compares two floatingpoint values on equality
Result: 1 = is equal, 0 = is not equal
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
Compatibility to FreeUDFLib AvERP, GrUDF
Input INT, INT
Output INT
Parameter 1 Integer value 1
Parameter 2 Integer value 2
Compares two Integer-values on equality.
Result: 1 = is equal, 0 = is not equal
Note: If a parameter is entered as DOUBLE (erroneously), will first rounds on INT and compares then!
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
Function of adhoc
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 Floating Point 1
Parameter 2 Floating Point 2
Returns the larger one of the two entered Floating Points.
TestSQL
SELECT 15.346 AS ISCORRECT, F_MAXNUM(15.345, 15.346) FROM RDB$DATABASE;

F_MAX
Compatibility to GrUDF
Identically to F_MAXNUM
F_MAX and F_MAXNUM may not be used in the same SQL.

F_MINNUM
Function of adhoc
Input DOUBLE, DOUBLE
Output DOUBLE
Parameter 1 Floating Point 1
Parameter 2 Floating Point 2
Returns the smaller one of the two entered Floating Points.
TestSQL
SELECT 15.345 AS ISCORRECT, F_MINNUM(15.345, 15.346) FROM RDB$DATABASE;

F_MIN
Compatibility to GrUDF
Identically to F_MINNUM
F_MIN and F_MINNUM may not be used in the same SQL.

F_ISDIVISIBLEBY
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input INT, INT
Output INT
Parameter 1 Divident (the number, which will divided)
Parameter 2 Divisor (the number, by which one divides)
Determined whether the division results an integer.
Result: is divisibly through = 1, is not divisibly through = 0
TestSQL
SELECT 1 AS ISCORRECT, F_ISDIVISIBLEBY(15, 3) FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_ISDIVISIBLEBY(15, 4) FROM RDB$DATABASE;

Date-Time-Functions - Functions to calculate with date-time

F_ADDYEAR

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Date optionally time
Parameter 2 Number of years for adding
Added to the date the number of years.
With negative parameter 2 the years will subtracted.
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

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Date optionally time
Parameter 2 Number of months for adding
Added to the date the number of months.
With negative parameter 2 the months will subtracted.
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

Function of adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Date optionally time
Parameter 2 Number of weeks for adding
Added to the date the number of weeks.
With negative parameter 2 the weeks will subtracted.
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
Function of adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Date optionally time
Parameter 2 Number of days for adding
Added to the date the number of days.
With negative parameter 2 the days will subtracted.
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
Function of adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Date Time
Parameter 2 Number of houres for adding.
Added to the date/time the number of houres.
With negative parameter 2 the houres will subtracted.
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
Function of adhoc
Input TIMESTAMP, INT
Output TIMESTAMP
Parameter 1 Date Time
Parameter 2 Number of minutes for adding.
Added to the date/time the number of minutes.
With negative parameter 2 the minutes will subtracted.
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

Function of adhoc

Input	TIMESTAMP, INT
Output	TIMESTAMP
Parameter 1	Date Time
Parameter 2	Number of seconds for adding.

Added to the date/time the number of seconds.

With negative parameter 2 the seconds will subtracted.

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;
```

F_AGEINYEARS
Function of adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) years of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
TestSQL

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

F_AGEINYEARSTHRESHOLD
Function of adhoc
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) years of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) months of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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

Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) months of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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

Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) weeks of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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

Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) weeks of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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

Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) days of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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

Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) days of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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
Function of adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) hours of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
TestSQL
SELECT 3 AS ISCORRECT, F_AGEINHOURS('01.10.2005 15:01:03','01.10.2005 18:01:03') FROM RDB$DATABASE;

F_AGEINHOURSTHRESHOLD
Function of adhoc
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) hours of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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
Function of adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) minutes of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
TestSQL
SELECT 14 AS ISCORRECT, F_AGEINMINUTES('01.10.2005 15:01:03','01.10.2005 15:15:03') FROM RDB$DATABASE;

F_AGEINMINUTESTHRESHOLD
Function of adhoc
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) minutes of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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
Function of adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) seconds of date 1 to date 2.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
TestSQL
SELECT 20 AS ISCORRECT, F_AGEINSECONDS('01.01.2005 15:01:01','01.01.2005 15:01:21') FROM RDB$DATABASE;

F_AGEINSECONDSTHRESHOLD
Function of adhoc
Input TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Parameter 3 Minimum value
Parameter 4 Minimum value will used (0 = no, 1 = yes)
Parameter 5 Maximum value
Parameter 6 Maximum value will used (0 = no, 1 = yes)
Calculate the time difference in (integral) seconds of date 1 to date 2.
If parameter 4 is = 1, the output is minimum the minimum value.
If parameter 6 is = 1, the output is maximum the maximum value.
For compatibility reasons:
If date 2 is smaller than date 1, the result is negative. Actually it would have to be 0, because there is no negative age!
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
Compatibility to GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) years of date 1 to date 2.
Results are always positive numbers.
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
Function of adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) months of date 1 to date 2.
Results are always positive numbers.
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
Compatibility to GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) weeks of date 1 to date 2.
Results are always positive numbers.
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
Function of adhoc
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) days of date 1 to date 2.
Results are always positive numbers.
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
Compatibility to GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) hours of date 1 to date 2.
Results are always positive numbers.
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
Compatibility to GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) minutes of date 1 to date 2.
Results are always positive numbers.
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
Compatibility to GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 (smaller) Date optionally time 1
Parameter 2 (bigger) Date optionally time 2
Calculate the time difference in (integral) seconds of date 1 to date 2.
Results are always positive numbers.
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the number of days of the year (day number x of the year) up to the selected date.
TestSQL
```
SELECT 235 AS ISCORRECT, F_DAYOFYEAR('22.08.2004') FROM RDB$DATABASE;
```

F_DAYOFMONTH
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 Date optionally time
Returns the number of days of the month
TestSQL
```
SELECT 23 AS ISCORRECT, F_DAYOFMONTH('23.08.2004') FROM RDB$DATABASE;
```

F_DAYSOFMONTH
Function of adhoc
Input INTEGER, INTEGER
Output INTEGER
Parameter 1 month
Parameter 2 year
Returns the number of days for the selected month and year.
TestSQL
```
SELECT 29 AS ISCORRECT, F_DAYSOFMONTH(2, 2004) FROM RDB$DATABASE;
```

F_LASTDAY
Compatibility to GrUDF
(Nearly) identically to F_DAYSOFMONTH, only the parameter is changed (first year, then month)
Input INT, INT
Output INTEGER
Parameter 1 year
Parameter 2 month
Returns the last day of the selected month and year.
SELECT 29 AS ISCORRECT, F_LASTDAY(2004, 2) FROM RDB$DATABASE;

F_DAYOFWEEK
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the day of the week as numerical value of the selected date.
Week starts with sunday (= 1)
TestSQL
SELECT 1 AS ISCORRECT, F_DAYOFWEEK('22.08.2004') FROM RDB$DATABASE;

F_DTIME
Function of adhoc
Input TIMESTAMP
Output INT
Parameter 1 date optionally time
Determines the number of days of the selected date since 31.12.1899.
Counting starts at 0.
TestSQL
SELECT 2 AS ISCORRECT, F_DTIME('03.01.1900') FROM RDB$DATABASE;

F_CMONTHLONG
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(4)
Parameter 1 date optionally time
Returns the month in English.
TestSQL
SELECT 'August' AS ISCORRECT, F_CMONTHLONG('23.08.2004') FROM RDB$DATABASE;

F_CMONTHLONGLANG
Function of adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 date optionally time
Parameter 2 Language identifier for the output
de = german, uk = english, fr = french, es = spanish, it =italian
Returns the month in the selected language.
TestSQL
SELECT 'Août' AS ISCORRECT, F_CMONTHLONGLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

F_CMONTHSHORT
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(4)
Parameter 1 date optionally time
Returns the month shortened in English.
TestSQL
SELECT 'Aug' AS ISCORRECT, F_CMONTHSHORT('23.08.2004') FROM RDB$DATABASE;

F_CMONTHSHORTLANG
Function of adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 date optionally time
Parameter 2 Language identifier for the output
de = german, uk = english, fr = french, es = spanish, it =italian
Returns the month shortened in the selected language.
TestSQL
SELECT 'Aoû' AS ISCORRECT, F_CMONTHSHORTLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

F_CDOWLONG
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(16)
Parameter 1 date optionally time
Returns the day of week in English.
TestSQL
SELECT 'Monday' AS ISCORRECT, F_CDOWLONG('23.08.2004') FROM RDB$DATABASE;

F_CDOWLONGLANG
Function of adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 date optionally time
Parameter 2 Language identifier for the output
de = german, uk = english, fr = french, es = spanish, it =italian
Returns the day of week in the selected language.
TestSQL
SELECT 'Lundi' AS ISCORRECT, F_CDOWLONGLANG('23.08.2004', 'fr')
FROM RDB$DATABASE;

F_CDOWSHORT
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output CSTRING(4)
Parameter 1 date optionally time
Returns the day of week shortened in English.
TestSQL
SELECT 'Mon' AS ISCORRECT, F_CDOWSHORT('23.08.2004') FROM RDB$DATABASE;

F_CDOWSHORTLANG
Function of adhoc
Input TIMESTAMP, CSTRING(2)
Output CSTRING(16)
Parameter 1 date optionally time
Parameter 2 Language identifier for the output
de = german, uk = english, fr = french, es = spanish, it =italian
Returns the day of week shortened in the selected language.
TestSQL
SELECT 'Lun' AS ISCORRECT, F_CDOWSHORTLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

F_GFORMATD
Function of adhoc
Input CSTRING(254), TIMESTAMP
Output CSTRING(254)
Parameter 1 d = Day possibly one-digit
dd = Day always in two digits
m = Month possibly one-digit
mm = Month always in two-digits
yy = Year always in two digits
yyyy = Year always in four digits
h = Hour possibly one-digit
hh = Hour always in two digits
n = Minute possibly one-digit
nn = Minute always in two digits
s = Second possibly one-digit
ss = Second always in two digits
all other indications will indicate according to their task in parameters 1 in the respective place.
Parameter 2 date
Formats the date according to 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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the Year of the selected date.
TestSQL
SELECT 2004 AS ISCORRECT, F_YEAR(' 22.08.2004 14:38:12') FROM RDB$DATABASE;

F_QUARTER
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the quarter of the selected date.
TestSQL
SELECT 3 AS ISCORRECT, F_QUARTER('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_MONTH
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the month of the selected date.
TestSQL
SELECT 8 AS ISCORRECT, F_MONTH('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_WEEK
Compatibility to FreeUDFLibC
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the week of the selected date.
Counting starts at week 1 which contains the 1. January.
Note:
During the numbering of weeks there are different variations:
The first week of the year is
* that one, into which the 1. January falls
* the first complete week of the year
* the first week, into which at least 4 days of the new year fall. (ISO 8601)
These 3 variations refer all to the classical week.
There are deviations e.g. in Great Britain, where there is a tax year, which always starts on 6 April.
The international standard ISO 8601 (1973) specifies Monday as week beginning. The first week of the year must contain at least four days. This is equivalent with the week, which contains the 4. January, or which week, which contains the first Thursday of the year.
In Germany it is fixed since 1976 that the week starts with Monday: DIN 1355 (1974), DIN EN 28601 (1993).
According to the aforementioned standards the year has 53 calendar weeks, if it starts or ends with one Thursday.
TestSQL
```
SELECT 41 AS ISCORRECT, F_WEEK('02.10.2005 14:38:12') FROM RDB$DATABASE;
```

F_HOUR
Compatibility to GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date time
Returns the hours of time of the selected date.
If you entered only one date, the hour is = 0.
TestSQL
```
SELECT 14 AS ISCORRECT, F_HOUR('23.08.2004 14:38:12') FROM RDB$DATABASE;
```

F_MINUTE
Compatibility to GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date time
Returns the minutes of time of the selected date.
If you entered only one date, the minute is = 0.
TestSQL
```
SELECT 38 AS ISCORRECT, F_MINUTE('23.08.2004 14:38:12') FROM RDB$DATABASE;
```

F_SECOND
Compatibility to GrUDF
Input TIMESTAMP
Output INTEGER
Parameter 1 date time
Returns the seconds of time of the selected date.
If you entered only one date, the second is = 0.
TestSQL
SELECT 12 AS ISCORRECT, F_SECOND('23.08.2004 14:38:12') FROM RDB$DATABASE;

F_YEAROFYEAR
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
(siehe F_YEAR)
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the year of date.
TestSQL
SELECT 2004 AS ISCORRECT, F_YEAROFYEAR(' 22.08.2004 14:38:12') FROM
RDB$DATABASE;

F_WEEKOFYEAR
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
(also look at F_WEEK)
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns the week of year.
TestSQL
SELECT 34 AS ISCORRECT, F_WEEKOFYEAR('22.08.2004 14:38:12') FROM
RDB$DATABASE;

F_WOY
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP
Output CSTRING(7)
Parameter 1 date optionally time
Returns year and week of the year supplies as a connected string
TestSQL
SELECT '200434' AS ISCORRECT, F_WOY('22.08.2004 14:38:12') FROM RDB$DATABASE;

F_ENCODEDATE
Compatibility to GrUDF
Input INT, INT, INT
Output DATE
Parameter 1 year
Parameter 2 month
Parameter 3 day
Produce from year, month, day a date value.
TestSQL
SELECT '20.02.2004' AS ISCORRECT, F_ENCODEDATE(2004, 2, 20) FROM RDB$DATABASE

F_ENCODETIME
Compatibility to GrUDF
Input INT, INT, INT
Output TIME
Parameter 1 hours
Parameter 2 minutes
Parameter 3 seconds
Produce from hours, minutes, seconds a current value
SELECT '09:45:53' AS ISCORRECT, F_ENCODETIME(9, 45, 53) FROM RDB$DATABASE

F_ENCODETIMESTAMP
Compatibility to GrUDF
Input I NT, INT, INT, INT, INT, INT
Output TIMESTAMP
Parameter 1 year
Parameter 2 month
Parameter 3 day
Parameter 4 hours
Parameter 5 minutes
Parameter 6 seconds
Produced from year, month, day a date current value
Produced from year, month, day, hours, minutes, seconds a date current value.
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
Compatibility to FreeUDFLibC
Input CSTRING(11)
Output TIME
Parameter 1 anglophile time as string
Converts a anglophile time (05:04:01 AM) into a 24-hours-time.
Inputformat can in each case be with one or two digits and the separation from AM/PM with or without blanks, AM/PM must be entered in capital letters.
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Input TIMESTAMP
Output TIMESTAMP
Parameter 1 date time
Returns date time as 31.12.1899 (date 0) with the selected time.
Note:
To get only the time please use 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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP
Output TIMESTAMP
Parameter 1 date time
Returns the date time as time 00:00:00 from the selected date.
Note:
To get only the date please use 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
Compatibility to FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 date optionally time 1
Parameter 2 date optionally time 2
Compares two date-time-values on equality, but only the date will be considered
Result: 1 = is equal, 0 = is unequal
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
Compatibility to FreeUDFLib AvERP, GrUDF
Input TIMESTAMP, TIMESTAMP
Output INT
Parameter 1 date optionally time 1
Parameter 2 date optionally time 2
Compares two date-time-values on equality, date and time will be considered
Result: 1 = is equal, 0 = is unequal
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
Compatibility to FreeUDFLibC
Input TIMESTAMP
Output INTEGER
Parameter 1 date optionally time
Returns, if the year of the selected date is a leap year.
Result: 1 = leap year, 0 = no leap year
Algorithm Y2k-fest (2000 was a leap year).
TestSQL

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

F_MAXDATE
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP
Output TIMESTAMP
Parameter 1 date optionally time
Parameter 2 date optionally time
Returns the larger one of the two entered dates.
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
Compatibility to FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Input TIMESTAMP, TIMESTAMP
Output TIMESTAMP
Parameter 1 date optionally time
Parameter 2 date optionally time
Returns the smaller one of the two entered dates.
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
Compatibility to FreeUDFLib AvERP, GrUDF
Input INTEGER
Output DATE
Parameter 1 Year, for which the Eastersunday is to be computed.
Returns the date of the Eastersunday of the selected year.
TestSQL
SELECT '27.03.2005' AS ISCORRECT, F_OSTERDATUM(2005)
FROM RDB$DATABASE;

F_ZEITDIFFERENZ

Compatibility to FreeUDFLib AvERP, GrUDF

Input	TIMESTAMP, TIMESTAMP, CSTRING(1)
Output	DOUBLE
Parameter 1	date optionally time
Parameter 2	date optionally time
Parameter 3	Kind of output t = timedifference in days h = timedifference in hours m = timedifference in minutes s = timedifference in seconds all other values always result 0

Calculate the timedifference between date 1 and date 2. Output as floatingpoint like selected kind of output in parameter 3.

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 Functions - BLOb Conversions

F_BLOBASPCHAR

Compatibility to FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input BLOB
Output CSTRING(32760)
Parameter 1 TextBlob, which is to be converted to VarChar
Converts TextBlob to VarChar
TestSQL (for TestISO.GDB)
SELECT 'ein einzeiliger TextBLOb' AS ISCORRECT, TEXTBLOB,
F_BLOBASPCHAR(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 1

F_STRBLOB

Compatibility to FreeUDFLib, FreeUDFLib AvERP, GrUDF
Input CSTRING(32760)
Output BLOB
Parameter 1 VarChar, which is to be converted to TextBlob
Converts VarChar to TextBlob

F_BLOB2EXCEL

Function from adhoc
Input BLOB
Output BLOB
Parameter 1, blob to convert for Excel.
To convert multiline texts and texts with converted commas to Excel, it is required to transform the blob. This fuction will do the following:

- appends a double inverted comma at the beginning and the end of the blob
- doubles all inverted commas in the blob
- deletes all CHR(13) in the blob
- limits the input blob to 32760 characters (Limitation in Excel)

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
Note:
Actually it doesn’t make any sense to export a very large text from the database to excel cell, this function may be practicable in combinatin with F_LEFT or F_RIGHT.
For instance:
SELECT F_RIGHT(F_BLOB2EXCEL(STRFeldTAGEBUCH), 1000) FROM ...export the last 1000 charachters of the field “STRFeldTAGEBUCH”

F_BLOBCAT
Compatible to GrUDF
Input BLOB, BLOB
Output BLOB
Parameter 1 Blob, to be appended with Blob from Paramter 2
Parameter 2 Blob, to be appended with Blob from Paramter 1
Adds to blob values to one output value
TestSQL (for TestISO.GDB)
INSERT INTO BLOBTEST (TEXTBLOB) SELECT F_BLOBCAT(TEXTBLOB, (SELECT TEXTBLOB FROM BLOBTEST WHERE BLOBTESTID = 2)) FROM BLOBTEST WHERE BLOBTESTID = 1
Insert a new record in table BLOBTEST with the content of the field TEXTBLOB form datarow with ID = 1 append with the datarow ID = 2

F_BLOBCATSTR
Compatible to GrUDF
Input BLOB, CSTRING(32760)
Output BLOB
Parameter 1 Blob, to be appended with the string from Paramter 2
Parameter 2 String, to be appended with Blob from Paramter 1
Adds the value of a blobfield with the value of a string field to a blob.
Before appending the second value, the function creates a CRLF.
TestSQL
TestSQL (for TestISO.GDB)
INSERT INTO BLOBTEST (TEXTBLOB) SELECT F_BLOBCATSTR(TEXTBLOB, 'Diese Zeile wurde angehängt') FROM BLOBTEST WHERE BLOBTESTID = 1
Insert a new record in table BLOBTEST with the content of the field TEXTBLOB form datarow with ID = 1 append with tthe string ‘Diese Zeile wurde angehängt'

F_BLOBLEFT
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input BLOB, INT
Output CSTRING(32760)
Parameter 1 Text Blob, which is to be cut
Parameter 2 Length of returned string
Returns a string which is cut by the indicationnumber from left gives from parameter 2.
Counting starts with 1
TestSQL (for TestISO.GDB)
SELECT 'ein einzeiliger' AS ISCORRECT, F_BLOBLEFT(TEXTBLOB, 15) FROM BLOBTEST WHERE BLOBTESTID = 1

F_BLOBMID
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input BLOB, INT, INT
Output CSTRING(32760)
Parameter 1 TextBlob, from which a character string is to be determined
Parameter 2 Position, at which the determined string starts
Parameter 3 Length of the determined string
Returns the number of letters (parameter 3) of the entered text starting from the entered letternumber (parameter 2).
Counting starts for parameter 2 with 0.
TestSQL (for TestISO.GDB)
SELECT 'zwei' AS ISCORRECT, F_BLOBMID(TEXTBLOB, 4, 4) FROM BLOBTEST WHERE BLOBTESTID = 2

F_BLOBRIGHT
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input BLOB, INT
Output CSTRING(32760)
Parameter 1 TextBlob
Parameter 2 Number of indications from right
Returns the string which is cut to the number of indications counted from right from parameter 2.
Counting starts with 1.
TestSQL (for TestISO.GDB)
SELECT 'dritten Zeile' AS ISCORRECT, F_BLOBRIGHT(TEXTBLOB, 13) FROM BLOBTEST WHERE BLOBTESTID = 3

F_BLOBREPLACESTRING
Function from adhoc
Input BLOB, CSTRING(254), CSTRING(254), INT, INT
Output BLOB
Parameter 1 blob to be modified by replacing a string
Parameter 2 the string to be replaced in blob
Parameter 3 the string wich will replace Parameter 2
Parameter 4 0 = just replace the first appearance, 1 = replace all appearance
Parameter 5 0 = case sensitive, 1 = not case sensitive
Replace one/all character strings fromParameter2 of the the blob by the character string from Parameter3.
TestSQL (for 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
Function from adhoc
Input BLOB, CSTRING(1024)
Output INT
Parameter 1 TextBLOb (in which you are looking for the string from parameter 2)
Parameter 2 String 2 (you are looking for in the BLOb)
Returns the position in the BLOb where the string starts.
Counting starts at 0.
TestSQL (for TestISO.GDB)
SELECT 4 AS ISCORRECT, F_BLOBSUBSTR(TEXTBLOB, 'einzeiliger') FROM BLOBTEST WHERE BLOBTESTID = 1

F_BLOBLINE
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input BLOB, INT
Output CSTRING(32760)
Parameter 1 TextBLOb
Parameter 2 No of row for output
Returns the content of the <parameter 2> row
TestSQL (for TestISO.GDB)
SELECT 'mit einer zweiten Zeile' AS ISCORRECT, F_BLOBLINE(TEXTBLOB, 2) FROM BLOBTEST WHERE BLOBTESTID = 3

F_BLOBSIZE
Compatibility to FreeUDFLib, FreeUDFLib AvERP
Input BLOB
Output INT
Parameter 1 TextBlob
For TextBlob: Returns the size/length (similar F_STRINGLENGTH).
For BinarBlob: Returns the size of file in byte.
TestSQL (for TestISO.GDB)
For TextBLOb:
SELECT 50 AS ISCORRECT, F_BLOBSIZE(TEXTBLOB),
F_STRINGLENGTH(F_BLOBASPCHAR(TEXTBLOB)) FROM BLOBTEST WHERE
BLOBTESTID = 2
For binaryBLOb:
SELECT 1426 AS ISCORRECT, F_BLOBSIZE(BINAERBLOB) FROM BLOBTEST WHERE
BLOBTESTID = 4

F_BLOBMAXSEGMENTLENGTH
Compatibility to FreeUDFLib
Input BLOB
Output INT
Parameter 1 TextBLOb oder BinärBLOb
Count Bytes of the BLOB-segment of the maximum segtments
TestSQL (for TestISO.GDB)
For TextBLOb:
SELECT 16384 AS ISCORRECT, F_BLOBMAXSEGMENTLENGTH(TEXTBLOB) FROM
BLOBTEST WHERE BLOBTESTID = 8
For binaryBLOb:
SELECT 16384 AS ISCORRECT, F_BLOBMAXSEGMENTLENGTH(BINAERBLOB) FROM
BLOBTEST WHERE BLOBTESTID = 7

F_BLOBSEGMENTCOUNT
Compatibility to FreeUDFLib
Input BLOB
Output INT
Parameter 1 BLOb
Count of BLOB-segments
TestSQL (for TestISO.GDB)
For TextBLOb:
SELECT 3 AS ISCORRECT, F_BLOBSEGMENTCOUNT(TEXTBLOB) FROM BLOBTEST
WHERE BLOBTESTID = 8
For binaryBLOb:
SELECT 2 AS ISCORRECT, F_BLOBSEGMENTCOUNT(BINAERBLOB) FROM
BLOBTEST WHERE BLOBTESTID = 7

F_BLOBLINECOUNT
Function from adhoc
Input BLOB
Output INT
Parameter 1 TextBLOb
Count rows of a TextBLOb
TestSQL (for TestISO.GDB)
SELECT 3 AS ISCORRECT, F_BLOBLINECOUNT(TEXTBLOB) FROM BLOBTEST
WHERE BLOBTESTID = 3

F_BLOBCOMPARE
Compatibility to FreeUDFLib, GrUDF (BLOBICOMP)
Input BLOB
Output INT
Parameter 1 BLOb
Compares two BLObs
TestSQL (for 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

UUID Functions

go top

Preliminary note (look http://en.wikipedia.org/wiki/UUID):
A Universally Unique Identifier is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. Thus, anyone can create a UUID and use it to identify something with reasonable confidence that the identifier will never be unintentionally used by anyone for anything else. Information labelled with UUIDs can therefore be later combined into a single database without need to resolve name conflicts. The most widespread use of this standard is in Microsoft's Globally Unique Identifiers (GUIDs) which implement this standard.
A UUID is essentially a 16-byte (128 bit) number and in its canonical form a UUID may look like this: 550e8400-e29b-41d4-a716-446655440000
The number of theoretically possible UUIDs is therefore 256^{16} or about 3.4×10^{38}. This means that during a 10 billion year lifetime of the Earth, about 1 trillion UUIDs have to be created every nanosecond to exhaust the number of UUIDs.
The term GUID usually references Microsoft's implementation of the UUID standard, however, many other pieces of software use the term GUID including Oracle Database and Novell eDirectory.
Conceptually, the original (version 1) generation scheme for UUIDs was to concatenate the UUID version with the MAC address of the computer that is generating the UUID, and with the number of 100-nanosecond intervals since the adoption of the Gregorian calendar. In practice, the actual algorithm is more complicated. This scheme has been criticized in that it is not sufficiently 'opaque'; it reveals both the identity of the computer that generated the UUID and the time at which it did so.
In the RFC4122 there are 5 UUID-versions descriped:

1 time-based
  a. with a real MAC-adress descriped in the IEEE 802-standard
  b. with a random-generated MAC-Adresse descriped in the IEEE 802-standard
2 DCE-security-version (with POSIX UIDs)
3 namespace-based (with MD5)
4 random created
5 namespace-based (SHA-1)

For purpose of the locally generated unique IDs for records with compatibility under Windows and Linux with the same algorithm only the versions 1 and 4 can be used (version 3 and 5 not because there is not surely a name-space on the computer):
Because you prefer a decoding or a non-decoding UUID we made 3 different kinds of UUIDs:
F_UUID1MAC
  version 1 with a real MAC-Adresse
  you can decode the creating-time and the MAC-adress of the computer
F_UUID1RAND
  version 1 with a randomly generated MAC-adresse
  only decode of the creating-time possible
F_UUID4
  version 4
  no decoding possible
In the uuidlib from Ian Newby we found a modification of the version 1b (with random generated MAC-adress) which was compressed instead of the output-mask from the RFC4122. This algorithm has advantages on created UUIDs on the same server because it first give out the MAC-adress and then the timestamp. So the index could work quicker.
We also took this algorithm, incl. the transforming-functions, so we have a compressed version of every of the 3 basic-versions.
F_UUID1MACCOMPR, UUID1RANDCOMPR, UUID4COMPR.

F_UUID1MAC
Function from adhoc
Input nothing
Output CSTRING(36)
Creates an Universally Unique Identifier (UUID) version 1a
TestSQL
SELECT F_UUID1MAC() FROM RDB$DATABASE;
Remark:
Could no MAC-adress found on the DB-server, the function creates an UUID like UUID1RAND

F_UUID1RAND
Function from adhoc
Input nothing
Output CSTRING(36)
Creates an Universally Unique Identifier (UUID) version 1b
TestSQL
SELECT F_UUID1RAND() FROM RDB$DATABASE;

F_UUID4
Function from adhoc
Input nothing
Output CSTRING(36)
Creates an Universally Unique Identifier (UUID) version 4
TestSQL
SELECT F_UUID4() FROM RDB$DATABASE;

Notice (from 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
Function from adhoc
Input nothing
Output CSTRING(36)
Creates a compressed Universally Unique Identifier (UUID) version 1a
TestSQL
SELECT F_UUID1MACCOMPR() FROM RDB$DATABASE;

F_UUID1RANDCOMPR
Function from adhoc
Input nothing
Output CSTRING(36)
Creates a compressed Universally Unique Identifier (UUID) version 1b
TestSQL
SELECT F_UUID1RANDCOMPR() FROM RDB$DATABASE;

F_UUID4COMPR
Function from adhoc
Input nothing
Output CSTRING(36)
Creates a compressed Universally Unique Identifier (UUID) version 4
TestSQL
SELECT F_UUID4COMPR() FROM RDB$DATABASE;

F_UUID2UUIDCOMPR
Output-compatible to the function GUID_TO_UUID from the uuidlib
Input CSTRING(250)
Output CSTRING(22)
Parameter 1 real normal UUID
Converts a normal UUID to a compressed UUIDof every version
TestSQL (for TestISO.GDB)
If you takte an compressed UUID for input instead of a normal UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID2UUIDCOMPR(UUIDCOMPR) FROM UUIDTEST;
If you take a normal string for input instead of a normal UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID2UUIDCOMPR('1235abcde') FROM UUIDTEST;
If you take a normal UUID for input (right):
SELECT 'this is valid Input' AS ISCORRECT, F_UUID2UUIDCOMPR(UUID) FROM UUIDTEST;

F_UUIDCOMPR2UUID
Output-compatible to the function UUID_TO_GUID from the uuidlib
Input CSTRING(250)
Output CSTRING(36)
Parameter 1 real compressed UUID
Converts a compressed UUID to a normal UUID of the created version
TestSQL (for TestISO.GDB)
If you takte an normal UUID for input instead of a compressed UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUIDCOMPR2UUID(UUID) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;
If you take a normal string for input instead of a compressed UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUIDCOMPR2UUID('12345abcde') FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;
If you take a compressed UUID for input (right):
SELECT 'this is valid Input' AS ISCORRECT, F_UUIDCOMPR2UUID(UUIDCOMPR) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;

F_UUIDVERSION
Function from adhoc
Input CSTRING(250)
Output CSTRING(20)
Parameter 1 real UUID
Reports the variant and version of the UUID
Possible variants are:

0	NCS (reserved for backwards-compatibility)
10	the actuall variant as descriped in the RFC4122
110	Microsoft (reserved for backwards-compatibility)
111	reserved for further variants

Possible versions are:

1	time-based
	a. with a real MAC-adress descriped in the IEEE 802-standard
	b. with a random-generated MAC-Adresse descriped in the IEEE 802-standard
2	DCE-security-version (with POSIX UIDs)
3	namespace-based (with MD5)
4	random created
5	namespace-based (SHA-1)

The function reports:
V1a
V1b
V3
V4
V5
TestSQL (for TestISO.GDB)
SELECT ART, F_UUIDVERSION(UUID) FROM UUIDTEST ORDER BY UUIDTESTID;

F_UUID1TIMESTAMP
Function from adhoc
Input CSTRING(250)
Output TIMESTAMP
Parameter 1 real UUID version 1a
Reports the timestamp from the UUID when the UUID was created. If the timestamp is’nt real, the function reports 31.12.1899 00:00:00 (Time = 0 - starttime of the internal InterBase/FireBird time)
TestSQL (for TestISO.GDB)
If you takte an compressed UUIID for input instead of a normal UUID:
SELECT '31.12.1899 00:00:00' AS ISCORRECT, F_UUID1TIMESTAMP(UUIDCOMPR) FROM UUIDTEST;
If you take a normal UUID for input (right):
SELECT ZEITSTEMPEL AS ISCORRECT, F_UUID1TIMESTAMP(UUID) FROM UUIDTEST;

F_UUID1COMPRTIMESTAMP
Function from adhoc
Input CSTRING(250)
Output TIMESTAMP
Parameter 1 real compressed UUID version 1a
Reports the timestamp from the UUID when the UUID was created. If the timestamp is'nt real, the function reports 31.12.1899 00:00:00 (Time = 0 - starttime of the internal InterBase/FireBird time)
TestSQL (for TestISO.GDB)
If you takte an normal UUIID for input instead of a compressed UUID:
SELECT '31.12.1899 00:00:00' AS ISCORRECT, F_UUID1COMPRTIMESTAMP(UUID) FROM UUIDTEST;
If you take a compressed UUID for input (right):
SELECT ZEITSTEMPEL AS ISCORRECT, F_UUID1COMPRTIMESTAMP(UUIDCOMPR) FROM UUIDTEST;

F_UUID1MACMAC
Function from adhoc
Input CSTRING(250)
Output CSTRING(17)
Parameter 1 real UUID version 1a
Reports the MAC-adress of the computer from the UUID, where the UUID was created.
TestSQL (for TestISO.GDB)
If you takte an compressed UUIID for input instead of a normal UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID1MACMAC(UUIDCOMPR) FROM UUIDTEST ORDER BY UUIDTESTID;
If you take a normal UUID for input (right):
SELECT F_UUIDVERSION(UUID), F_UUID1MACMAC(UUID) FROM UUIDTEST ORDER BY UUIDTESTID;

F_UUID1MACCOMPRMAC
Function from adhoc
Input CSTRING(250)
Output CSTRING(17)
Parameter 1 real compressed UUID version 1a
Reports the MAC-adress of the computer from the UUID, where the UUID was created.
TestSQL (for TestISO.GDB)
If you takte an normal UUIID for input instead of a compressed UUID:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID1MACCOMPRMAC(UUID) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL ORDER BY UUIDTESTID;
If you take a compressed UUID for input (right):
SELECT F_UUIDVERSION(F_UUIDCOMPR2UUID(UUIDCOMPR)), F_UUID1MACCOMPRMAC(UUIDCOMPR) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL ORDER BY UUIDTESTID;

F_UUID1COMPARE

Function from adhoc

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

Compares two Universally Unique Identifier (UUID) lexically.
Lexically meens here from left to right blockwise,

- first block 1 of the UUID
- second block 2
- third block 3

Even the tested blocks are identically, go on to the next block. The function goes on until there is a difference or the 3. Block is reached.
The function reports:
Die Funktion gibt dann aus:

-1	u1 is lexically bevore u2
0	u1 is equal u2
1	u1 is lexically after u2

Notice, that this is a lexically sorting and no time-sorting!
To get a sorting by creating-time od a UUID version 1:

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

Remark:
You can also compare other UUID than version 1 - but this reports only shit. Only a version 1 UUID contains a timestamp which is used in this function.
TestSQL (for 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
Function from adhoc
Input CSTRING(32), CSTRING(8), CSTRING(32), CSTRING(8190), CSTRING(8190)
Output CSTRING(8190)
Parameter 1 Comparisonstring 1
Parameter 2 Comparisonoperator
=
<>
<
>
<=
>=
Everyone of these operators a "n" (numeric) can be placed in front, if the two comparede strings contain floatingpoint, e.g. n =
Parameter 3 Comparisonstring 2
Parameter 4 if comparison applies, then result = Parameter 4.
Parameter 5 if comparison does not apply, then result = Parameter 5.
"reproduction" of a IF-loop
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
Function from adhoc
Input none
Output CSTRING(254)
Determines the version of FreeAdhocUDF
TestSQL
SELECT F_VERSION() FROM RDB$DATABASE;

Functions, which are not implemented until yet (no demand, other possibility)

from 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

from FreeUDFLib C

all made

from GrUDF

- F_DAYSPAN -> the same result as F_AGEINDAYS
- F_MONTHSPAN -> the same result as F_AGEINMONTH
- F_YEARSPAN -> the same result as F_AGEINYEARS
- F_WEEKSPAN -> the same result as F_AGEINWEEKS
- F_SECONDSPAN -> the same result as F_AGEINSECONDS
- F_MINUTESPAN -> the same result as F_AGEINMINUTES
- F_HOURSPA -> the same result as F_AGEINHOURS
- F_MONTHSTART -> the same result as CAST(‘01.02.2003' AS DATE)
- F_MONTHEND -> the same result as CAST(F_LASTDAY(2006, 2) || ‘02.2003' AS DATE)
- F_DAYFRAC -> the same result as CAST(F_AGEINSECONDS('20.02.2004 00:00:00', '20.02.2004 12:00:00') AS DOUBLE PRECISION) / 86400
- F_DOUBLE -> the same result as CAST(1234.123 AS DOUBLE PRECISION)
- F_FRAC -> the same result as 1234.123 - F_TRUNCATE(1234.12)
- F_TRUNCDEC
- F_CEIL -> look ib_util, free UDF-Functions of InterBase
- F_FLOOR -> look ib_util, free UDF-Functions of InterBase
- F_DIV -> look ib_util, free UDF-Functions of InterBase
- F_EXP
- F_SQRT -> look ib_util, free UDF-Functions of InterBase
- F_POWER
- F_LNXP1
- F_LOG10 -> look ib_util, free UDF-Functions of InterBase
- F_LOG2 -> look ib_util, free UDF-Functions of InterBase
- F_LOGN -> look ib_util, free UDF-Functions of InterBase
- F_PI -> look ib_util, free UDF-Functions of InterBase
- F_COS -> look ib_util, free UDF-Functions of InterBase
- F_SIN -> look ib_util, free UDF-Functions of InterBase
- F_TAN -> look ib_util, free UDF-Functions of InterBase
- F_COTAN -> look ib_util, free UDF-Functions of InterBase
- F_HYPOT -> look ib_util, free UDF-Functions of InterBase
- F_NOT
- F_AND
- F_OR
- F_XOR

- F_SHL
- F_SHR
- F_BLOBCOMP -> s. F_BLOBCOMPARE
- F_BLOBICOMP -> s. F_BLOBCOMPARE to use with other functions
- F_BLOBUPPER
- F_BLOBWRAP
- F_BLOBISEMPTY

Posibility to change UDFs instead of dependendies

go top

If you want to change to FreeAdhocUDF there is often the problem that there are dependendies on the samenamed UDFs (because they where used in ComputedBy fields, triggers, views or stored procedures) and you could not drop them (until InterBase 6 this was possible).
Here's a (not clean but save) trick to do this:

```
/* UDF-dependendies inactivate (here for all who starts with F_) */
UPDATE RDB$DEPENDENCIES SET RDB$DEPENDED_ON_NAME = 'x' ||
RDB$DEPENDED_ON_NAME
WHERE RDB$DEPENDED_ON_TYPE = 15
AND RDB$DEPENDED_ON_NAME STARTING WITH 'F';
/* now drop the UDF-functions */
DROP  EXTERNAL FUNCTION F_.....;
DROP  EXTERNAL FUNCTION F_.....;
...
/* now append FreeAdhocUDF Funktionen with the script */
...
/* activate the (old) dependendies */
UPDATE RDB$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';
```