**Documentation (description et test de requêtes SQL) FreeAdhocUDF** état 2006-10-31

Traduction de Jean Lhoir, Bruxelles, Belgique et Renate Theuring

*Rapports d'erreurs et souhaits uniquement à UDF@adhoc-data.de*

**Table des matières**


Objectif de FreeAdhocUDF
Versions
Licence
Description des fonctions de FreeAdhocUDF (144 fonctions)
Avertissement
Fonctions de chaînes de caractères (47 fonctions)
Conversions
F_LOWER
F_ANSILOWERCASE
F_UPPER
F_ANSIUPPERCASE
F_PROPERCASE
F_CHARACTER
F_ENCRYPTMD5
F_SOUNDEX
F_GENERATESNDXINDEX
F_GSOUNDEX
F_TELEFONNR
F_DIGITS
F_STR2EXCEL
Traitements
F_COLLATEBR
F_COPY
F_EUROVAL
F_HOLDSTRING
F_LEFT
F_LINEWRAP
F_LTRIM
F_LRTRIM
F_MID
F_PADLEFT
F_PADRIGHT
F_REPLACE
F_REPLACESTRING
F_RIGHT
F_RTRIM
F_STRIPSTRING
F_STRIPSTRINGHOLD
F_SUBSTR
F_STRCOPY
F_STRRM
Constructions
F_CRLF
F_LF
F_SPACE
F_STROFCHAR

- F_NBSP
- F_DQM
- F_SQM
- F_TAB

Comparaisons
- F_EQUALSTRING

Recherches et informations
- F_FINDWORD
- F_FINDNTHWORD
- F_FINDWORDINDEX
- F_STRINGLENGTH
- F_STRINGLISTITEM

Fonctions mathématiques (31 fonctions)

Constructions
- F_INTRANDOM

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

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

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

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

Fonctions de dates et heures (66 fonctions)

Fonctions de calcul de dates et heures

- F_ADDYEAR
- F_ADDMONTH
- F_ADDWEEK
- F_ADDDAY
- F_ADDHOUR
- F_ADDMINUTE
- F_ADDSECOND

Fonctions de calcul de différences de dates et d'heures

- 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

Fonctions de formatage de dates et heures

- 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

Fonctions de tests et de comparaisons de dates et heures

F_EQUALDATE
F_EQUALDATETIME
F_ISLEAPYEAR
F_MAXDATE
F_MINDATE
F_OSTERDATUM
F_ZEITDIFFERENZ

Fonctions BLOb (16 fonctions)

Conversions de champs BLOb

F_BLOBASPCHAR
F_STRBLOB
F_BLOB2EXCEL

Traitement de champs BLOb

F_BLOBCAT
F_BLOBCATSTR
F_BLOBLEFT
F_BLOBMID
F_BLOBRIGHT
F_BLOBSIZE
F_BLOBREPLACESTRING

BLOb Calculs

F_BLOBSIZE
F_BLOBMAXSEGMENTLENGTH
F_BLOBSEGMENTCOUNT
F_BLOBLINE_COUNT

BLOb Comparaisons

F_BLOBCOMPARE

Fonctions UUID (14 fonctions)

UUID Constructions

F_UUID1MAC
F_UUID1RAND
F_UUID4
F_UUID1MACCOMPR
F_UUID1RANDCOMPR
F_UUID4COMPR

UUID Conversions

F_UUID2UUIDCOMPR

F_UUIDCOMPR2UUID
UUID lire
F_UUIDVERSION
F_UUID1TIMESTAMP
F_UUID1COMPRTIMESTAMP
F_UUID1MACMAC
F_UUID1COMPRMAC
UUID Comparaisons
F_UUIDCOMPARE
Fonctions diverses (2 fonctions)
F_IF
F_VERSION
Annexe
Fonctions non encore prises en compte à ce jour
Possibilité d'échanger les UDF malgré dépendances

**Objectif de FreeAdhocUDF** retour

1. Compatibilité entre Windows et Linux afin de pouvoir transposer les banques de données au moyen de Backup / Restore. Cela implique de conserver les noms de fonctions et les points d'entrée (entry points) ainsi que de fournir des résultats équivalents. Sous Windows, le module porte le nom de «FreeAdhocUDF.dll» et sous Linux «FreeAdhocUDF.so».
   Le nom de module dans la Définition-SQL s'appelle «FreeAd hocUDF»; Windows recherche automatiquement *FreeAdhocUDF.dll*, tandisque Linux recherche *FreeAdhocUDF.so*.
   Sous Linux, la distinction des majuscules et minuscules est importante pour le nom de module comme pour le nom de librairie (Lib).
2. En ce qui concerne les anciennes versions, c'est-à-dire la version originale FreeUDFLib (en Delphi 1998, de Gregory Deatz), FreeUDFLibC (version Delphi recodée en C en 1999, de Gregory Deatz), FreeUDFLib de AvERP (en Delphi, contenant quel ques ajouts) et GrUDF (en Delphi et Kylix 2004, de Torsten Grundke et Gerd Kroll), la compatibilité permet de passer de chacune d'elles à FreeAdhocUDF.
3. Des corrections ont été apportées aux UDF d'origine lorsque les résultats en Windows étaient erronés alors qu'ils étaient corrects en Linux. Ces corrections apparais sent seulement dans la nouvelle version FreeAdhocUDF et pas dans les anciennes.
4. Des corrections ont été apportées aux UDF d'origine lorsque les résultats en Windows se présentent différemment de ceux de Linux.
   - F_AGEINWEEKS
   précédemment la fonction de FreeUDFLibC calculait l'intervalle en jours divisé par 7 et en suite arrondi. Cela conduisait par exemple à un résultat de 1 entre les dates 20-08-2004 et 21-08-2004, alors que sous Windows (FreeUDFLib) le résultat est 0.
5. Optimisation du code C, en partie complètement réécrit.
6. À l'o ri gine, FreeAdhocUDF a été tes té sur
   - InterBaseSS 6.02 sous Windows XP Professional et Windows 2000 Advanced Server
   - InterBase 7.1 SP2 sous Windows XP Professional et Windows 2000 Advanced Server
   - InterBase 7.1 SP2 sous Mandrake Linux 10.0 / Mandriva Linux 10.1
   - InterBase 7.5 sous Windows 2000 Advanced Server
   - InterBase 7.5 sous Mandriva Linux 10.2
   - FireBirdSS 1.5.2 sous Windows XP Professional et Windows 2000 Advanced Server
   - FireBirdSS 1.5.2 sous Mandrake Linux 10.0 / Mandriva Linux 10.1
   - FireBirdSS 1.5.2 sous SuSe Linux 8.1 (utilise FreeAdhocUDF.so_SuSe81_FB15 !)
   - FireBirdSS 1.5.2 sous SuSe EnterpriseServer 8 (utilise FreeAdhocUDF.so_SuSe81_FB15 !)
   - FireBirdSS 1.5.2 sous SuSe Linux 10.0
   - FireBirdSS 1.5.2 sous Ubuntu Server 5.10
   - FireBirdSS 1.5.2 sous Kubuntu 5.10
   - FireBirdSS 2.0.0RC2 sous Windows 2000 Advanced Server
   - FireBirdSS 2.0.0RC2 sous Mandriva Linux 10.1
   Pour les tests des nou vel les ver sions, se ré fé rer à Versions.

**Sous Linux, les fichiers so doivent éventuellement changer de nom, soit "FreeAdhocUDF.so" pour être utilisés.**

**Version (sans numéro) du 23.08.2004**
Elle établit la compatibilité entre Windows et Linux pour InterBase et était compatible avec la FreeUDFLib et la FreeUDFLibC. Elle comprenait en plus quelques nouvel les fonctions.

- 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" du 16.10.2005**
Ellr assure la compatibilité entre Windows et Linux pour InterBase et FireBird et est compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP et GrUDF. Elle comprend de nouvelles fonctions qui n'étaient reprises par aucune version précédente, mais qui sont cependant utiles.

- 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" du 31.12.2005**
Corrige l'erreur suivante:
- F_OSTERDATUM (fournissait le mardi suivant Pâques au lieu du dimanche de Pâques)
- Script de déclaration UDF (les points d'entrée Entrypoint de F_MAX et de F_MIN étaient erronés)
- Mise à jour de la documentation.

**Version "adhoc 20060302" du 03.02.2006**
Comporte les ajouts suivants:
- F_STRINGLISTITEM
- F_LF
- F_STR2EXCEL
- F_BLOB2EXCEL
- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBREPLACESTRING
- correction à F_ADDYEAR, F_ADDMONTH, F_ADDWEEK, F_ADDDAY, F_ADDHOUR, F_ADDMINUTE, F_ADDSECOND
- plus de possibilités F_FINDWORD et F_FINDNTHWORD par les caractères spéciaux (Ç, ë)
- Mise à jour de la documentation.

**Version “adhoc 20060306" du 06.03.2006**
Contient une variante concernant F_SUBSTR pour compatibilité conc. Delphi FreeUDFLib.

**Version “adhoc 20060513" du 16.05.2006**
- compatibilité à partir de InterBase 7.5
- compatibilité à partir de FireBird 2.0 RC1 (actuellement RC2)
- sous Windows à partir de FireBird 1.5 une variante de FreeAdhocUDF-FireBird est nécessaire
- sous Windows à partir de InterBase 6 une variante de FreeAdhocUDF-InterBase est nécessaire
- sous Linux pour FireBird 1.5 une variante spécifique à SuSe 8.1 est nécessaire
- sous Linux pour InterBase une variante FreeAdhocUDF-InterBase est nécessaire
- SuSe 8.1 avec InterBase 6 n’est plus supportée
- correction d’une erreur dans F_AGEINYEARSTHRESHOLD (inversion de paramètres)
- correction d’une erreur dans F_ADDMONTH
- correction d’erreurs et adaptation du manuel

fonctions testées à
- InterBaseSS 6.02 sous Windows XP Professional et Windows 2000 Advanced Server
- InterBase 7.1 SP2 sous Windows XP Professional et Windows 2000 Advanced Server
- InterBase 7.1 SP2 sous Mandrake Linux 10.0 / Mandriva Linux 10.1
- InterBase 7.5 sous Windows 2000 Advanced Server
- InterBase 7.5 sous Mandriva Linux 10.2
- FireBirdSS 1.5.2 sous Windows XP Professional et Windows 2000 Advanced Server
- FireBirdSS 1.5.2 sous Mandrake Linux 10.0 / Mandriva Linux 10.1
- FireBirdSS 1.5.2 sous SuSe Linux 8.1 (utilise FreeAdhocUDF.so_SuSe81_FB15 !)

- FireBirdSS 1.5.2 sous SuSe EnterpriseServer 8 (utilise FreeAdhocUDF.so_SuSe81_FB15 !)
- FireBirdSS 1.5.2 sous SuSe Linux 10.0
- FireBirdSS 1.5.2 sous Ubuntu Server 5.10
- FireBirdSS 1.5.2 sous Kubuntu 5.10
- FireBirdSS 2.0.0RC2 sous Windows 2000 Advanced Server
- FireBirdSS 2.0.0RC2 sous Mandriva Linux 10.1

**Version “adhoc 20060919 du 19.09.2006**
Version intérims - pas publiée
Workaround pour erreurs (plantage de serveur) pour InterBase en
- F_BLOBASPCHAR
- F_BLOBCAT
- F_BLOBCATSTR

**Version “adhoc 20060925" du 25.09.2006**
élimine des erreurs (aussi plantage de serveur) pour InterBase et FireBird en:
- F_BLOBASPCHAR
- F_BLOBCAT
- F_BLOBCATSTR
- F_BLOBREPLACESTRING

élimine erreurs pour Linux
- F_DAYSOFMONTHS

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

(Nouvelles fonctions) testées à
- InterBase 7.1 SP2 sous Windows 2000 Advanced Server
- InterBase 7.1 SP2 sous Mandriva 2006
- InterBase 7.5 SP1 sous Windows XP Professional
- InterBase 7.5 SP1 sous Mandriva 2006
- FireBird 1.5.2 sous SuSe 8.1
- FireBird 1.5.2 sous SuSe 10.0
- FireBird 1.5.2 sous Mandriva 2006
- FireBird 2.0 RC4 sous Windows XP Professional
- FireBird 2.0 RC4 sous Mandriva 2006

**Version “adhoc 20061030" du 31.10.2006**
rajoute:
- F_NBSP
- F_DQM
- F_SQM
- F_TAB
- F_INTRANDOM
- F_UUID1MAC
- F_UUID1RAND
- F_UUID1MACCOMPR

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

Adaptation du manuel

(Nouvelles fonctions) testées à

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

**Avertissement**

Il y a des différences de limites entre Interbase et Firebird et aussi entre leurs versions. Chaque champ d'une requête SQL doit respecter la limite des champs ET l'enregistrement complet de la requête ne peut dépasser la limite d'un enregistrement.
Le calcul de la limite d'un enregistrement s'opère par l'addition des dimensions spécifiées pour les champs et non par les dimensions des valeurs contenues dans ces champs:
- un champ de table avec sa définition, par exemple VARCHAR(100), 2 by tes pour SMALLINT, etc.
- une fonction UDF avec une valeur de retour, par exemple 100 bytes pour … RETURN VARCHAR(100) …
- une procédure stoquée, idem.

| | column limit<br>limite de champ | row limit<br>limite d'enregistrement |
|---|---|---|
| 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 |

Pour que la requête de l'exemple suivant puisse fonctionner:
select F_REPLACE(F_REPLACE(F_REPLACE(F_REPLACE('abcdefg', 'a', 'x'), 'b', 'y'), 'c', 'z'), 'd', '-') from TABLE ...la définition (DECLARE) de F_REPLACE doit être la suivante:

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

"RETURNS CSTRING(8190)" ne peut plus être 8190, car un champ se reproduit par 4 fois par récurrence --> 4 * 8190 = 32760.
La requête doit en particulier comprendre un champ, comme par exemple Select CHAMP, F_REPLACE(F_REPLACE(F_REPLACE(... de telle sorte que la requête initiale avec une définition de 8190 n'aura plus recours par récurrence que 3 fois à F_REPLACE. Ainsi la limite de 32 kB de l'enregistrement ne sera pas dépassée, alors que 4 fois aurait conduit à un dépassement des 32 kB.
Comme une application peut vomprendre plusieurs contraintes pour une seule fonction UDF, une possibilité est de déclarer plusieurs fois la même fonction:
- F_REPLACE avec RETURN CSTRING(254)
- F_REPLACE4 avec RETURN CSTRING(4095)
- F_REPLACE8 avec RETURN CSTRING(8190)
- F_BIGREPLACE avec RETURN CSTRING(32760)

Ainsi on dispose pour toutes les conditions de la fonction correcte.

**F_CHARACTER**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée INT
Sortie CSTRING(2)
Paramètre 1 Code ASCII
Fournit le signe du code ASCII correspondant au paramètre 1 par TestSQL
TestSQL
SELECT 'B' AS ISCORRECT, F_CHARACTER(66) FROM RDB$DATABASE
SELECT 'ä' AS ISCORRECT, F_CHARACTER(228) FROM RDB$DATABASE

**F_CHR**
Compatible avec GrUDF
identique à F_CHARACTER

**F_ENCRYPTMD5**
Compatible avec FreeUDFLibC
Entrée CSTRING(128)
Sortie CSTRING(33)
Paramètre 1 Chaîne à encrypter
Encrypté selon l'algorithme MD5
TestSQL
SELECT 'e7d31845480111fdba3316129e166860' AS ISCORRECT, F_ENCRYPTMD5('Pauline') FROM RDB$DATABASE;

**F_PROPERCASE**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée CSTRING(32760)
Sortie CSTRING(32760)
Paramètre 1 Chaîne à transformer
Capitalise la chaîne: chaque mot commence par une lettre haut de casse alors que les autres sont bas de casse. Transforme également les lettres accentuées. Le signe «ß» allemand reste bien entendu inchangé.
TestSQL
SELECT 'Dies Ist Ein Test Äh' AS ISCORRECT, F_PROPERCASE('dies ist ein test äh') FROM RDB$DATABASE

**F_GSOUNDEX**
Fonction de adhoc
Entrée CSTRING(8190)
Sortie CSTRING(8190)
Paramètre 1 Chaîne
Construit le code Soundex allemand de la chaîne du paramètre 1. Peut servir à rechercher les doublons.
TestSQL
SELECT 'MAYR' AS ISCORRECT, F_GSOUNDEX('Meier'), F_GSOUNDEX('Maier'), F_GSOUNDEX('Mayer') FROM RDB$DATABASE

**F_GENERATESNDXINDEX**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée CSTRING(8190)
Sortie CSTRING(6)
Paramètre 1 Chaîne
Construit le code original Soundex de la chaîne.
Soundex est un code bâti par un algorithme phonétique et sert à indexer des mots et des phrases selon le son de la prononciation anglaise. Des mots sont ainsi classés selon le code produit rassemblant des mots différents de prononciation voisine.
À partir de la seconde position, seules les 6 premières consonnes sont prises en compte et une seule fois si plusieurs mêmes consonnes se suivent (selon l'exemple l, m, y, w, r, d). Cette codification ne permet pas la comparaison de longues chaînes de caractères
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**
Fonction de adhoc
identique à F_GENERATESNDXINDEX

**F_ANSILOWERCASE**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée CSTRING(32760)
Sortie CSTRING(32760)
Paramètre 1 Chaîne à transformer
Transforme tous les signes en lettres minuscules, y compris les lettres accentuées.
TestSQL
SELECT 'schöner tag' AS ISCORRECT, F_ANSILOWERCASE('SchÖner TAG') FROM RDB$DATABASE
Remarque:
Cette fonction n'est plus nécessaire sous FireBird 2, car LOWER(..) traite aussi correctement les caractères spéciaux ou accentués.

**F_LOWER**
Fonction de adhoc, identique à F_ANSILOWERCASE

**F_ANSIUPPERCASE**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée CSTRING(32760)
Sortie CSTRING(32760)
Paramètre 1 Chaîne à transformer
Transforme tous les signes en lettres majuscules, y compris les lettres accentuées (contrairement à la fonction UPPER(…) de SQL).
TestSQL
SELECT 'SCHÖNER TAG' AS ISCORRECT, UPPER('Schöner Tag'), F_UPPER('Schöner Tag') FROM RDB$DATABASE
Remarque:
Cette fonction n'est plus nécessaire sous FireBird 2, car UPPER(..) traite aussi correctement les caractères spéciaux ou accentués.

**F_UPPER**
Fonction de adhoc
identique à ANSIUPPERCASE

**F_TELEFONNR**
Fonction de adhoc
Entrée CSTRING(32760), INT
Sortie CSTRING(32760)
Paramètre 1 Paramètre 1 Chaîne comprenant un numéro de téléphone (ou quelque chose de semblable dont tout doit être enlevé vant les chiffres).
Paramètre 2 Paramètre 2 définit le nombre de chiffres à la fin, qui doivent être remplacés par *.
Enlève tous les caractères non numériques ainsi que les espaces d'un numéro de téléphone. Si le numéro commence par '49', il commencera par '+49'.
Si le numéro donné commence par '+', ce signe sera conservé.
Remarque:
Cette fonction est conçue pour formater un numéro de téléphone mal adapté à la norme TAPI et le fournir à une installation téléphonique qui n'est pas équipée pour faire elle-même ce formatage.
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**
Compatible avec GrUDF
Entrée CSTRING(32760)
Sortie CSTRING(32760)
Paramètre 1 Chaîne 1
Enlève tous les signes non numériques de la chaîne 1 (par exemple pour TAPI).
TestSQL
SELECT '0232653535' AS ISCORRECT, F_DIGITS(' (0232) / 6535-35') FROM RDB$DATABASE

**F_STR2EXCEL**
Fonction de adhoc
Entrée CSTRING(32760)
Sortie CSTRING(32760)
Paramètre 1 Chaîne à transformer pour Excel
Pour pouvoir exporter vers Excel du texte de plusieurs lignes ou du texte contenant des guillemets doubles hauts (") venant de longues chaînes, certaines adaptations sont nécessaires. Cette fonction agit dans ce sens. Elle:
- ajoute au début et à la fin de la chaîne un guillemet double haut (")
- redouble chaque guillemet double haut (") qui se trouvait dans le texte
- enlève tous les CHR(13) de la chaîne
- tronque la chaîne pour ne pas dépasser 32760 caractères (limite d'un champ dans 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

Remarque:
Comme cela n’a pas de sens d’exporter vers Excel un (très) long texte venant d’une base de données, on doit tout d’abord le traiter avec les fonctions F_LEFT ou F_RIGHT pour rendre l’opération possible. Par exemple:
SELECT F_RIGHT(F_STR2EXCEL(STRFeldAgenda), 1000) FROM ... exporte, par exemple, les derniers 1000 caractères d’un champ d’un agenda.

**F_COLLATEBR**

Compatible avec GrUDF
Entrée CSTRING(32760)
Sortie CSTRING(32760)
Paramètre 1 Chaîne dans laquelle des signes particuliers doivent êtretransformés.
Transformation signifie ici substitution d'une lettre accentuée par une lettre majuscule non accentuée.
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
Remarque:
au contraire de GrUDF Ÿ n'est par tranformé en Y (n'existe pas en Linux?)
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**

Compatible avec GrUDF
Entrée CSTRING(8190), INT, INT
Sortie CSTRING(8190)
Paramètre 1 Chaîne donnée dans laquelle une chaîne partielle sera traitée
Paramètre 2 Position de début de la chaîne partielle à l'intérieur de la chaîne donnée
Paramètre 3 Longueur de la chaîne à l'intérieur de la chaîne donnée
Fournit une chaîne de caractères partielle dont la longueur est définie au paramètre 3 et extraite de la chaîne donnée à partir de la lettre dont la position est donnée par le paramètre 2. Le comptage pour ce paramètre commence en position 1.
Même fonction que F_MID. Mais la première lettre est en position 1, tandis que pour F_MID elle est en position 0.
TestSQL

```
SELECT  'tag' AS ISCORRECT, F_COPY('Geburtstagsparty', 8, 3) FROM RDB$DATABASE;
```

**F_EUROVAL**
Fonction de adhoc
Entrée DOUBLE
Sortie CSTRING(254)
Paramètre 1 Nombre à virgule flottante
Place EUR après le nombre donné.
TestSQL
SELECT '15.47 EUR' AS ISCORRECT, F_EUROVAL(15.47) FROM RDB$DATABASE

**F_HOLDSTRING**
Compatible avec FreeUDFLibC
Entrée CSTRING(32760), CSTRING(254)
Sortie CSTRING(32760)
Paramètre 1 Chaîne 1
Paramètre 2 Chaîne 2 (comprend tous les caractères qui ne doivent pas être supprimés de la chaîne 1)
Enlève de la chaîne 1 tous les caractères qui ne sont pas dans la chaîne 2. L'ordre des caractères dans la chaîne 2 ne joue aucun rôle.
Identique à F_STRIPSTRINGHOLD
Pendant de F_STRIPSTRING
Remarque:
Une requête SQL ne peut utiliser à la fois F_HOLDSTRING et F_STRIPSTRINGHOLD
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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GruDF
Entrée CSTRING(8190), INT
Sortie CSTRING(8190)
Paramètre 1 Chaîne
Paramètre 2 Longueur de la chaîne demandée
Fournit une partie de la chaîne d'entrée depuis son premier caractère et d'une longueur donnée par le paramètre 2.
TestSQL
SELECT 'Dies i' AS ISCORRECT, F_LEFT('Dies ist ein Test', 6) FROM RDB$DATABASE

**F_LINEWRAP**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée CSTRING(32760), INT, INT
Sortie CSTRING(32760)
Paramètre 1 Chaîne
Paramètre 2 Position de départ
Paramètre 3 Justification
Founit tous les mots de la chaîne à partir de la position de départ, qui ne dépassent pas la justification.
On compte à partir de 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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée CSTRING(8190)
Sortie CSTRING(8190)
Paramètre 1 Chaîne dont on désire supprimer les espaces en tête
Enlève tous les espaces en tête de la chaîne donnée, tout en conservant les espaces protégés (créés avec <ALT> <255>)

**F_LRTRIM / F_BIGLRTRIM**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée CSTRING(8190)
Sortie CSTRING(8190)
Paramètre 1 Chaîne dont on désire supprimer les espaces en tête et en queue
Enlève tous les espaces en tête et en queue de la chaîne donnée, tout en conservant les espaces protégés (créés avec <ALT> <255>)
TestSQL
SELECT 'Dies ist ein Test' AS ISCORRECT, F_LTRIM(' Dies ist ein Test') FROM RDB$DATABASE

**F_MID**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée CSTRING(8190), INTEGER, INTEGER
Sortie CSTRING(8190)
Paramètre 1 Chaîne hors de laquelle on désire une chaîne partielle
Paramètre 2 Position du début de la chaîne partielle dans la chaîne donnée
Paramètre 3 Longueur de la chaîne partielle
Fournit une chaîne partielle de la chaîne donnée, de longueur indiquée par le paramètre 3, et à partir de la position donnée par le paramètre 2. Le premier caractère de la chaîne donnée est la
position 0.
TestSQL
SELECT 'tag' AS ISCORRECT, F_MID('Geburtstagsparty', 7, 3) FROM RDB$DATABASE;

**F_PADLEFT**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée CSTRING(32760), CSTRING(16), INT

Sortie CSTRING(32760)

Paramètre 1 Chaîne 1 (que l'on désire faire précéder de la chaîne du paramètre 2 autant de fois que nécessaire pour atteindre la longueur donnée par le paramètre 3)

Parameter 2 Chaîne 2 (qui sert à remplir la chaîne 1)

Paramètre 3 Longueur à atteindre par le remplisage

Remplit la chaîne 1 à gauche avec les caractères de la chaîne 2 jusqu'à ce que la longueur spécifiée par le paramètre 3 soit atteinte

Si la chaîne 2 comprend plus de 1 caractère, le remplissage débute avec les caractères de droite de la chaîne 2 et s'arrête lorsque la longueur totale spécifiée est atteinte (voir second test ci-dessous).

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**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée CSTRING(32760), CSTRING(16), INT

Sortie CSTRING(32760)

Paramètre 1 Chaîne 1 (qui sera complétée par les caractères de la chaîne 2 jusqu'à atteindre la longueur spécifiée par le paramètre 3)

Paramètre 2 Chaîne 2 (avec laquelle on complète la chaîne 1)

Paramètre 3 Longueur de chaîne à atteindre par le remplissage

Complète la chaîne 1 à droite avec le caractère ou les caractères de la chaîne 2 pour atteindre la longueur totale spécifiée par le paramètre 3

Si la chaîne 2 comprend plus de 1 caractère, le remplissage débute avec les caractères de gauche de la chaîne 2 et s'arrête lorsque la longueur totale spécifiée est atteinte (voir second test ci-dessous).

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**

Compatible avec FreeUDFLibC

Entrée CSTRING(32760), CSTRING(254), CSTRING(254)

Sortie CSTRING(32760)

Paramètre 1 Chaîne dans laquelle le remplacement va s'opérer

Paramètre 2 Chaîne partielle à remplacer

Paramètre 3 Chaîne servant à remplacer la chaîne du paramètre 2

Remplace dans une chaîne toutes les chaînes partielles identiques à celle spécifiée au paramètre 2 par la chaîne spécifiée au paramètre 3.

Simplification de la fonction F_REPLACESTRING sans la possibilité de faire un seul remplacement et qui respecte les majuscules et minuscules spécifiées au paramètre 2.

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**

Compatible avec FreeUDFLib AvERP, GrUDF,

Entrée CSTRING(32760), CSTRING(254), CSTRING(254), INT, INT

Sortie CSTRING(32760)

Paramètre 1 Chaîne dans laquelle le remplacement va s'opérer

Paramètre 2 Chaîne partielle à remplacer

Paramètre 3 Chaîne servant à remplacer la chaîne du paramètre 2

Paramètre 4 0 = seulement le premier remplacement, 1 = tous les remplacements possibles

Paramètre 5 0 = respecter la casse des caractères du paramètre 2, 1 = ne pas respecter

Remplace dans une chaîne une ou toutes les chaînes partielles identiques à celle spécifiée au paramètre 2 par la chaîne spécifiée au paramètre 3. Peut tenir compte ou pas de la casse des caractères du paramètre 2.

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**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée CSTRING(8190), INT

Sortie CSTRING(8190)

Paramètre 1 Chaîne

Paramètre 2 Nombre de caractères à droite

Ne conserve à la fin de la chaîne que le nombre de caractères spécifié au paramètre 2

TestSQL

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

**F_RTRIM**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée CSTRING(8190)
Sortie CSTRING(8190)
Paramètre 1 Chaîne à laquelle on enlève les espaces de la fin
Enlève les espaces à la fin de la chaîne tout en préservant les espaces protégés (créés par <ALT> <255>)
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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée CSTRING(32760), CSTRING(254)
Sortie CSTRING(32760)
Paramètre 1 Chaîne 1 (de laquelle tous les caractères de la chaîne 2 sont à enlever)
Paramètre 2 Chaîne 2 (Caractères qui sont à enlever)
Enlève de la chaîne 1 tous les caractères contenus dans la chaîne 2, quel que soit leur ordre dans cette chaîne 2.
Pendant de 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**
Compatible avec ????, identique à F_HOLDSTRING
Entrée CSTRING(32760), CSTRING(254)
Sortie CSTRING(32760)
Paramètre 1 Chaîne 1 (de laquelle tous les caractères de la chaîne 2 sont à enlever)
Paramètre 2 Chaîne 2 (Caractères qui sont à enlever)
Enlève de la chaîne 1 tous les caractères qui ne sont pas compris dans la chaîne 2, quel que soit leur ordre dans cette chaîne 2.
Dit autrement: Renvoie seulement les caractères de la chaîne 1 dans leur ordre et qui sont compris dans la chaîne 2.
Pendant de F_STRIPSTRING
Une requête SQL ne peut comprendre à la fois F_HOLDSTRING et 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 / F_BIGSUBSTR**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée CSTRING(8190), CSTRING(1024)

Sortie INT

Paramètre 1 Chaîne 1 (dans laquelle la position de la chaîne 2 va être recherchée)

Paramètre 2 Chaîne 2 (dont on recherche la position dans la chaîne 1)

Fournit la position de la première occurrence de la chaîne 2 dans la chaîne 1. Le comptage commence à 0.

Remarque:

Dans la version originale de “FreeUDFLib” (1998 de Gregory Deatz) la séquence des paramètres d'entrée n'est pas la même que dans les autres fonctions String ainsi que dans la version portable C “FreeUDFLibC” (1999 by Gregory Deatz). Afin de maintenir la compatibilité avec ces deux variantes, deux “Entrypoints” différents figurent dans le script DECLARE de la fonction IF_SUBSTR.

Pour la compatibilité avec “FreeUDFLib” de Delphi, avec “FreeUDFLib AvERP” et avec “GrUDF” (qui est de son côté compatible avec “FreeUDFLib” de Delphi) on utilise le Entrypoint “strsub” alors que pour la compatibilité avec “FreeUDFLibC”, c'est le 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**

Compatible avec ????

Input CSTRING(8190), INT, INT

Output CSTRING(8190)

Paramètre 1 Chaîne de laquelle une partie va être extraite

Paramètre 2 Position du début de la chaîne recherchée

Paramètre 3 Longueur de la chaîne recherchée

Fournit la chaîne partielle à la longueur spécifiée au paramètre 3 et commençant au caractère dont le numéro est spécifié au paramètre 2. Le comptage commence à 1.

Même fonction que F_MID et F_COPY.

Le comptage commence à 1.

TestSQL

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

pas correct sous Linux

**F_STRRM**

Compatible avec ????

Entrée CSTRING(8190), INT

Sortie CSTRING(8190)

Paramètre 1 Chaîne de laquelle une position sera supprimée

Paramètre 2 Position dans la chaîne qui va être supprimée (le comptage commence à 0)

Supprime le caractère de la chaîne à la position spécifiée

TestSQL

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

**F_CRLF**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée aucune
Sortie CSTRING(3)
Paramètre 1 sans -> "()"
Retour chariot et nouvelle ligne. Fournit les signes CHR(13) + CHR(10)
TestSQL
SELECT ' ABC' || F_CRLF() || '123' FROM RDB$DATABASE;
Résultat: 'ABC' (1e ligne)
'123' (2e ligne)
SELECT '1e ligne' || F_CRLF() || '2e ligne' FROM RDB$DATABASE;

**F_LF**
Fonction de adhoc
Entrée aucune
Sortie CSTRING(2)
Paramètre 1 sans -> "()"
Marquage de fin de ligne. Met en oeuvre le caractère CHR(10).
Identique à F_CHARACTER(10)
TestSQL
SELECT 'ABC' || F_LF() || '123' FROM RDB$DATABASE;

**F_SPACE**
Compatible avec GrUDF
Entrée INTEGER
Sortie CSTRING(32760)
Paramètre 1 nombre d'espaces
Fournit une chaîne d'espaces (CHR(32)) de la longueur spécifiée par le paramètre
TestSQL
SELECT F_STRINGLENGTH(F_SPACE(10)) || ' Leerzeichen' AS ISCORRECT, F_SPACE(10) FROM RDB$DATABASE;

**F_STROFCHAR**
Compatible avec GrUDF
Entrée CSTRING(1), INT
Sortie CSTRING(32760)
Paramètre 1 Chaîne qui doit être répétée
Paramètre 2 Nombre des répétitions
Fournit une chaîne répétant le caractère spécifiée en paramètre 1 et de la longueur spécifiée en paramètre 2
TestSQL
SELECT F_STRINGLENGTH(F_STROFCHAR('A', 10)) || ' mal A' AS ISCORRECT,
F_STROFCHAR('A', 10)
FROM RDB$DATABASE;

**F_NBSP**
Fonction de adhoc
Entrée aucune
Sortie CSTRING(2)
Paramètre 1 aucun -> «()»
Espace insécable. Produit le caractère CHR(160).
Identique à F_CHARACTER(160).
TestSQL
SELECT 'ABC 123' AS ISCORRECT, 'ABC' || F_NBSP() || '123' FROM RDB$DATABASE;

**F_DQM**
Fonction de adhoc
Entrée aucune
Sortie CSTRING(2)
Paramètre 1 aucun -> «()»
Guillemet double ouvrant ou fermant (format droit). Fournit le caractère CHR(34).
Identique à F_CHARACTER(34).
TestSQL
SELECT 'ABC"123' AS ISCORRECT, 'ABC' || F_DQM() || '123' FROM RDB$DATABASE;

**F_SQM**
Fonction de adhoc
Entrée aucune
Sortie CSTRING(2)
Paramètre 1 aucun -> «()»
Apostrophe droite (non typographique). Fournit le caractère CHR(39).
Identique à F_CHARACTER(39).
TestSQL
SELECT 'ABC<Apostrophe droite>123' AS ISCORRECT, 'ABC' || F_SQM() || '123' FROM RDB$DATABASE;

**F_TAB**
Fonction de adhoc
Entrée aucune
Sortie CSTRING(2)
Paramètre 1 aucun -> «()»
Tabulateur. Fournit le caractère CHR(9).
Identique à F_CHARACTER(9).
TestSQL
SELECT 'ABC<TAB>123' AS ISCORRECT, 'ABC' || F_TAB() || '123' FROM RDB$DATABASE;

**F_EQUALSTRING**

Compatible avec FreeUDFLib AvERP, GrUDF,

| | |
|---|---|
| Entrée | CSTRING(8190), CSTRING(8190) |
| Sortie | INT |
| Paramètre 1 | Chaîne 1 |
| Paramètre 2 | Chaîne 2 |

Vérifie si la chaîne 1 est égale à la chaîne 2

Résultat 1 = est égale, 0 = pas égale

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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée CSTRING(32760), INT
Sortie CSTRING(254)
Paramètre 1 Chaîne dans laquelle une autre chaîne sera recherchée
Paramètre 2 Position à partir de laquelle la recherche doit commencer
Fournit un mot ou une partie de mot qui existe à la position spécifiée. L'origine de la position est ici 0, c'est-à-dire que la première position est 0 et que la fonction recherche le premier espace suivant.
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**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée CSTRING(32760), INT
Sortie CSTRING(254)
Paramètre 1 Chaîne dans laquelle on recherche une autre chaîne
Paramètre 2 Numéro d'ordre de la chaîne recherchée (numéro d'ordre du mot dans la chaîne donnée)
Fournit le ne mot. Le comptage commence à 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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée CSTRING(32760), INT
Sortie INT
Paramètre 1 Chaîne
Paramètre 2 pour vérifier la longueur de la chaîne
Cherchez l'endroit indiqué au paramètre 2 dans la chaîne. Si cet endroit existe dans la chaîne, le résultat est cet endroit; si cet endroit n'est pas dans la chaîne le résultat est –1.
Le comptage commence à 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**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée CSTRING(32760)

Sortie INT

Paramètre 1 Chaîne dont on recherche la longueur

Recherche la longueur d'une chaîne, en prenant en compte les espaces éventuels à la fin.

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 avec GrUDF

Entrées CSTRING(32760), CSTRING(254)

Sortie CSTRING(1024)

Paramètre 1 Liste de chaînes au format «n=…», «m=…», etc.

Paramètre 2 Valeur recherchée dans la liste

Recherche dans une chaîne de caractères, construite sous la forme d'une liste (NAME=Value), la ligne NAME=Value correspondant au nom donné.

TestSQL

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

**F_INTRANDOM**

Fonction de adhoc

| | |
|---|---|
| Entrée | aucune |
| Sortie | CSTRING(2) |
| Parameter 1 | Paramètre 1 valeur minimum du nombre aléatoire |
| Parameter 2 | Paramètre 2 valeur maximum du nombre aléatoire |

Fournit un nombre entier aléatoire compris entre les valeurs des paramètres 1 et 2.

TestSQL

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

Remarque:

Grâce à l'initialisation par l'horloge système et le processeur, il est exclu que deux appels à la fonction endéans une seconde puisse fournir le même nombre aléatoire.

**F_CONVERTFROM33**
Compatible avec FreeUDFLibC
Entrée CSTRING(254)
Sortie INT
Paramètre 1 Nombre en base 33 en chaîne qui est à convertir
Convertit un nombre en base 33 dans le système décimal
TestSQL
SELECT 1000, F_CONVERTFROM33('WB') FROM RDB$DATABASE;

**F_CONVERTTO33**
Compatible avec FreeUDFLibC
Entrée INT
Sortie CSTRING(254)
Paramètre 1 Nombre en système décimal qui est à convertir en base 33
Convertit un nombre décimal en base 33.
TestSQL
SELECT 'WB', F_CONVERTTO33(1000) FROM RDB$DATABASE;

**F_CONVERTFROMBASE**
Compatible avec ???
Entrées CSTRING(32), INT, CSTRING(8)
Sortie INT
Paramètre 1 Nombre d'un système quelconque sous forme de chaîne, à convertir
Paramètre 2 Base du nombre à convertir (par exemple 2 pour le système binaire)
Paramètre 3 Tous les chiffres valables dans le système spécifié au paramètre 2 (par exemple, '01234567' pour le système octal)
Convertir un nombre exprimé dans un système quelconque vers le système décimal.
TestSQL
SELECT 3, F_CONVERTFROMBASE('11', 2, '01') FROM RDB$DATABASE;
SELECT 9, F_CONVERTFROMBASE('11', 8, '01234567') FROM RDB$DATABASE;

**F_CONVERTTOBASE**
Compatibilité avec FreeUDFLibC
Entrées INT, INT, CSTRING(254)
Sortie CSTRING(254)
Paramètre 1 Nombre décimal à convertir
Paramètre 2 Base du système dans lequel on convertit (par exemple 2 pour le système binaire)
Paramètre 3 Tous les chiffres valables dans le système spécifié au paramètre 2 (par exemple, '01234567' pour le système octal)
Convertit un nombre décimal dans un système quelconque
TestSQL
SELECT '11', F_CONVERTTOBASE(3, 2, '01') FROM RDB$DATABASE;

**F_HEXTOINT**
Compatible avec GrUDF
Entrée CSTRING(20)
Sortie INTEGER
Paramètre 1 Valeur en hexadécimal
Convertit une valeur de hexadécimal en valeur entière.
TestSQL
SELECT 1 AS ISCORRECT, F_HEXTOINT('0000000001') FROM RDB$DATABASE;

**F_INTTOHEX**
Compatible avec GrUDF
Entrée INTEGER, INTEGER
Sortie CSTRING(254)
Paramètre 1 Entier à convertir
Paramètre 2 Nombre de chiffres décimaux
Convertit une valeur entière en hexadécimal.
TestSQL
SELECT ',0000000001' AS ISCORRECT, F_INTTOHEX(1,10) FROM RDB$DATABASE;

**F_DEGTORAD**
Compatible avec GrUDF
Entrée DOUBLE
Sortie DOUBLE
Paramètre 1 Angle en degrés
Convertit un angle de degrés en radian (mesure de l'arc).
TestSQL
SELECT '3.1415.. = PHI' AS ISCORRECT, F_DEGTORAD(180) FROM RDB$DATABASE;

**F_RADTODEG**
Compatible avec GrUDF
Entrée DOUBLE
Sortie DOUBLE
Convertit un angle de radian (mesure de l'arc) en degrés.
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**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée DOUBLE
Sortie CSTRING(32)
Paramètre 1 Valeur en virgule flottante
Convertit une valeur en virgule flottante en une valeur en dollars (arrondie à 2 décimales).
TestSQL
SELECT '$15.68' AS ISCORRECT, F_CONVERTTODOLLAR(15.678), F_DOLLARVAL(15.678)
FROM RDB$DATABASE;

**F_CONVERTTODOLLAR**
Compatible avec FreeUDFLibC
Identique à F_DOLLARVAL.

**F_FIXEDPOINT**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée DOUBLE, INT
Sortie CSTRING(32)
Paramètre 1 nombre en virgule flottante à arrondir
Paramètre 2 Nombre de chiffres décimaux du nombre arrondi
Arrondit un nombre en virgule flottante avec un nombre de décimales spécifiée au paramètre 2.
Dans la chaîne obtenue en sortie, le séparateur décimal est “.”.
TestSQL
SELECT '12345.5' AS ISCORRECT, F_FIXEDPOINT(12345.46, 1) FROM RDB$DATABASE

**F_FIXEDPOINTLANG**
Fonction de adhoc
Entrée DOUBLE, INT , CSTRING(1), CSTRING(1)
Sortie CSTRING(32)
Paramètre 1 nombre en virgule flottante à arrondir
Paramètre 2 Nombre de chiffres décimaux pour arrondir
Paramètre 3 Signe de séparateur décimal
Paramètre 4 Signe de séparateur de milliers
Arrondit le nombre en virgule flottante en tenant compte de nombre de chiffres décimaux du paramètre 2 et des séparateurs (décimal et milliers) définis en paramètres 3 et 4.
TestSQL
SELECT '12.345,5' AS ISCORRECT, F_FIXEDPOINTLANG(12345.46, 1, ',', '.') FROM RDB$DATABASE

**F_ROUND**
Compatible avec FreeUDFLibC
Entrée DOUBLE
Sortie INT
Paramètre 1 Nombre en virgule flottante à convertir en valeur entière
Arrondit un nombre en virgule flottante en un nombre entier.
TestSQL
SELECT 16 AS ISCORRECT, F_ROUND(15.567) FROM RDB$DATABASE

**F_ROUNDFLOAT**

Compatible avec FreeUDFLib, FreeUDFLib AvERP

Entrée DOUBLE, DOUBLE

Sortie DOUBLE

Paramètre 1 nombre en virgule flottante à arrondir

Paramètre 2 Modèle montarnt le format du nombre à virgule flottante (par exemple 0.01); arrondit au multiple suivant du 2e paramètre

Arrondit un nombre en virgule flottante.

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;
```

Ne fonctionne pas sous Linux

**F_TRUNCATE**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée DOUBLE

Sortie INT

Paramètre 1 nombre à virgule flottante à tronquer

Supprime la partie décimale d'un nombre à virgule flottante.

TestSQL

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

**F_ZAHLRUNDEN**

Compatible avec FreeUDFLib AvERP, GrUDF

Entrée DOUBLE, INT

Sortie DOUBLE

Paramètre 1 valeur à arrondir

Paramètre 2 Nombre de chiffres décimaux pour l'arrondi.

Arrondit le nombre à virgule flottante en fonction du nombre de chiffres décimaux spécifié au paramètre 2.

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**
Compatible avec GrUDF
Identique à F_DOUBLEABS
F_ABS et F_DOUBLEABS ne peuvent pas être utilisés dans une même requête SQL.

**F_DOUBLEABS**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée DOUBLE
Sortie DOUBLE
Paramètre 1 Nombre à virgule flottante entrant dans le calcul.
Calcule la valeur absolue (valeur positive) d'un nombre à virgule flottante
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**
Compatible avec FreeUDFLibC
Entrée INT
Sortie INT
Paramètre 1 Nombre entier à traiter
Calcule la valeur absolue (valeur positive) d'un nombre entier
SELECT 15 AS ISCORRECT, F_INTEGERABS(15) FROM RDB$DATABASE;
SELECT 15 AS ISCORRECT, F_INTEGERABS(-15) FROM RDB$DATABASE;

**F_FACT**
Fonction de adhoc
Entrée INT
Sortie DOUBLE
Paramètre 1 Nombre entier dont on demande la factorielle.
Calcule la factorielle (le produit de tous les nombres naturels de 1 à l'argument)
TestSQL
SELECT 6.000 AS ISCORRECT, F_FACT(3) FROM RDB$DATABASE

**F_FACTORIAL**
Compatible avec GrUDF
Identique à F_FACT
F_FACT et F_FACTORIAL ne peuvent pas être utilisées dans une même requête SQL.

**F_MODULO**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF,
Entrée INT, INT
Sortie INT
Paramètre 1 Dividende (le nombre à diviser)
Paramètre 2 Diviseur (le nombre par lequel on divise)
Fournit le Modulo (reste de la division de deux nombres entiers) du paramètre 1 par le paramètre 2.
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**

Compatible avec FreeUDFLib AvERP, GrUDF

Entrée DOUBLE, DOUBLE

Sortie DOUBLE

combien de % du 2e par rapport au 1er

par exemple Prix de vente par rapport au Prix d'achat

Test-SQL

```
SELECT 14.000 AS ISCORRECT, F_PROZENTE(100.00, 14.0) FROM RDB$DATABASE
```

**F_EQUALFLOAT**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée DOUBLE, DOUBLE
Sortie DOUBLE
Paramètre 1 Valeur à virgule flottante 1
Paramètre 1 Valeur à virgule flottante 2
Évalue l'égalité de deux valeurs à virgule flottante
Résultat 1 = égalité, 0 = pas d'égalité
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**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée INT, INT
Sortie INT
Paramètre 1 Nombre entier 1
Paramètre 2 Nombre entier 2
Évalue l'égalité de deux nombres entiers
Résultat 1 = égalité, 0 = pas d'égalité
Remarque:
Si (par erreur) un des paramètres est spécifié en DOUBLE, il sera d'abord arrondi en INT avant la comparaison!
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**
Fonction de adhoc
Entrée DOUBLE, DOUBLE
Sortie DOUBLE
Paramètre 1 Nombre à virgule flottante 1
Paramètre 2 Nombre à virgule flottante 2
Renvoie le plus grand des deux nombres à virgule flottante.
TestSQL

```
SELECT 15.346 AS ISCORRECT, F_MAXNUM(15.345, 15.346) FROM RDB$DATABASE;
```

**F_MAX**
Compatible avec GrUDF
Identique à F_MAXNUM

**F_MINNUM**
Fonction de adhoc
Entrée DOUBLE, DOUBLE
Sortie DOUBLE
Paramètre 1 Nombre à virgule flottante 1
Paramètre 2 Nombre à virgule flottante 2
Renvoie le plus petit des deux nombres à virgule flottante.
TestSQL
SELECT 15.345 AS ISCORRECT, F_MINNUM(15.345, 15.346) FROM RDB$DATABASE;

**F_MIN**
Compatible avec GrUDF
Identique à F_MINNUM

**F_ISDIVISIBLEBY**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée INT, INT
Sortie INT
Paramètre 1 Dividende (le nombre à diviser)
Paramètre 2 Diviseur (le nombre par lequel on divise)
Examine si la division fournit un nombre entier.
Les nombres sont divisibles = 1, ou non divisibles = 0
TestSQL
SELECT 1 AS ISCORRECT, F_ISDIVISIBLEBY(15, 3) FROM RDB$DATABASE;
SELECT 0 AS ISCORRECT, F_ISDIVISIBLEBY(15, 4) FROM RDB$DATABASE;

## Fonctions de dates et heures - fonctions de calcul de dates et heures

### F_ADDYEAR

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP, INT
Sortie TIMESTAMP
Paramètre 1 Date avec heure optionnelle
Paramètre 2 Nombre d'années à ajouter
Ajoute à la date le nombre d'années spécifié.
Si le paramètre 2 est négatif, les années sont soustraites.
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

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP, INT
Sortie TIMESTAMP
Paramètre 1 Date avec heure optionnelle
Paramètre 2 Nombre de mois à ajouter
Ajoute à la date le nombre de mois spécifié
Si le paramètre 2 est négatif, les mois sont soustraits.
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

Fonction de adhoc
Entrée TIMESTAMP, INT
Sortie TIMESTAMP
Paramètre 1 Date avec heure optionnelle
Paramètre 2 Nombre de semaines à ajouter
Ajoute à la date le nombre de semaines spécifié.
Si le paramètre 2 est négatif, les semaines sont soustraites.
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**
Fonction de adhoc
Entrée TIMESTAMP, INT
Sortie TIMESTAMP
Paramètre 1 Date avec heure optionnelle
Paramètre 2 Nombre de jours à ajouter
Ajoute à la date le nombre de jours spécifié.
Si le paramètre 2 est négatif, les jours sont soustraits.
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**
Fonction de adhoc
Entrée TIMESTAMP, INT
Sortie TIMESTAMP
Paramètre 1 Date et heure
Paramètre 2 Nombre d'heures à ajouter
Ajoute à la date et heure le nombre d'heures spécifié.
Si le paramètre 2 est négatif, les heures sont soustraites.
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**
Fonction de adhoc
Entrée TIMESTAMP, INT
Sortie TIMESTAMP
Paramètre 1 Date et heure
Paramètre 2 Nombre de minutes à ajouter
Ajoute à la date et heure le nombre de minutes spécifié.
Si le paramètre 2 est négatif, les minutes sont soustraites.
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**

Fonction de adhoc

Entrée TIMESTAMP, INT

Sortie TIMESTAMP

Paramètre 1 Date et heure

Paramètre 2 Nombre de secondes à ajouter

Ajoute à la date et heure le nombre de secondes spécifié.

Si le paramètre 2 est négatif, les secondes sont soustraites.

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**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)
Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d'années. Option fondée sur la compatibilité.
Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!
TestSQL

```
SELECT 3 AS ISCORRECT, F_AGEINYEARS('01.01.2005 15:01:21','01.10.2008 15:01:01') FROM RDB$DATABASE;
SELECT -3 AS ISCORRECT, F_AGEINYEARS('01.01.2005 15:01:21','01.10.2004 15:01:01') FROM RDB$DATABASE
```

**F_AGEINYEARSTHRESHOLD**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Sortie INT
Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)
Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)
Paramètre 3 Valeur minimum
Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)
Paramètre 5 Valeur maximum
Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d'années.
Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum. Option fondée sur la compatibilité.
Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!
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**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF

Entrée TIMESTAMP, TIMESTAMP

Sortie INT

Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)

Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)

Calcule la différence des dates 1 et 2 exprimée en nombre entier de mois. Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**

Compatible avec FreeUDFLib, FreeUDFLib AvERP

Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT

Sortie INT

Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)

Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)

Paramètre 3 Valeur minimum

Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)

Paramètre 5 Valeur maximum

Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)

Calcule la différence des dates 1 et 2 exprimée en nombre entier de mois.

Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum.

Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**

Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP

Entrée TIMESTAMP, TIMESTAMP

Sortie INT

Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)

Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)

Calcule la différence des dates 1 et 2 exprimée en nombre entier de semaines.

Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**

Compatible avec FreeUDFLib, FreeUDFLib AvERP

Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT

Sortie INT

Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)

Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)

Paramètre 3 Valeur minimum

Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)

Paramètre 5 Valeur maximum

Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)

Calcule la différence des dates 1 (paramètre 1) et 2 (paramètre 2) exprimée en nombre entier de semaines. Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum. Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**

Compatible avec FreeUDFLib, FreeUDFLib AvERP

Entrée TIMESTAMP, TIMESTAMP

Sortie INT

Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)

Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)

Calcule la différence des dates 1 et 2 exprimée en nombre entier de jours.

Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**

Compatible avec FreeUDFLib, FreeUDFLib AvERP

Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT

Sortie INT

Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)

Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)

Paramètre 3 Valeur minimum

Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)

Paramètre 5 Valeur maximum

Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)

Calcule la différence des dates 1 (paramètre 1) et 2 (paramètre 2) exprimée en nombre entier de jours. Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum. Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d'heures.
Option fondée sur la compatibilité.
Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!
TestSQL
SELECT 3 AS ISCORRECT, F_AGEINHOURS('01.10.2005 15:01:03','01.10.2005 18:01:03') FROM RDB$DATABASE;

**F_AGEINHOURSTHRESHOLD**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Paramètre 3 Valeur minimum
Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)
Paramètre 5 Valeur maximum
Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d'heures.
Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum. Option fondée sur la compatibilité.
Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!
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**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier de minutes.
Option fondée sur la compatibilité.
Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!
TestSQL
SELECT 14 AS ISCORRECT, F_AGEINMINUTES('01.10.2005 15:01:03','01.10.2005 15:15:03') FROM RDB$DATABASE;

**F_AGEINMINUTESTHRESHOLD**

Fonction de adhoc

Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT

Sortie INT

Paramètre 1 Date et heure 1 (la plus ancienne des 2)

Paramètre 2 Date et heure 2 (la plus tardive des 2)

Paramètre 3 Valeur minimum

Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)

Paramètre 5 Valeur maximum

Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)

Calcule la différence des dates 1 et 2 exprimée en nombre entier de minutes.

Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum. Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

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**

Fonction de adhoc

Entrée TIMESTAMP, TIMESTAMP

Sortie INT

Paramètre 1 Date et heure 1 (la plus ancienne des 2)

Paramètre 2 Date et heure 2 (la plus tardive des 2)

Calcule la différence des dates 1 et 2 exprimée en nombre entier de secondes.

Option fondée sur la compatibilité.

Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!

TestSQL

```
SELECT 20 AS ISCORRECT, F_AGEINSECONDS('01.01.2005 15:01:01','01.01.2005 15:01:21')
FROM RDB$DATABASE;
```

**F_AGEINSECONDSTHRESHOLD**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP, INT, INT, INT, INT
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Paramètre 3 Valeur minimum
Paramètre 4 Valeur minimum utilisée (0 = non, 1 = oui)
Paramètre 5 Valeur maximum
Paramètre 6 Valeur maximum utilisée (0 = non, 1 = oui)
Calcule la différence des dates 1 et 2 exprimée en nombre entier de secondes.
Si le paramètre 4 vaut 1, le résultat sera au moins la valeur minimum. Si le paramètre 6 vaut 1, le résultat sera au plus la valeur maximum. Option fondée sur la compatibilité.
Si la date 2 est plus ancienne que la date 1, le résultat est négatif. En vérité il devrait être nul puisqu'il n'y a pas d'âge négatif!
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**
Compatible avec GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)
Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d'années. Le résultat est toujours positif.
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**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)
Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d'années. Le résultat est toujours positif.
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**
Compatible avec GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)
Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier de semaines.
Le résultat est toujours positif.
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**
Fonction de adhoc
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1 (la plus ancienne des 2)
Paramètre 2 Date et heure optionnelle 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier de jours. Le résultat est toujours positif.
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**
Compatible avec GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier d’heures.
Le résultat est toujours positif.
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**
Compatible avec GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier de minutes.
Le résultat est toujours positif.
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**
Compatible avec GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure 1 (la plus ancienne des 2)
Paramètre 2 Date et heure 2 (la plus tardive des 2)
Calcule la différence des dates 1 et 2 exprimée en nombre entier de secondes.
Le résultat est toujours positif.
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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit le nombre de jours (jour X de l'année) jusqu'à la date spécifiée.
TestSQL

```
SELECT 235 AS ISCORRECT, F_DAYOFYEAR('22.08.2004') FROM RDB$DATABASE;
```

**F_DAYOFMONTH**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit le jour du mois.
TestSQL

```
SELECT 23 AS ISCORRECT, F_DAYOFMONTH('23.08.2004') FROM RDB$DATABASE;
```

**F_DAYSOFMONTH**
Fonction de adhoc
Entrée INTEGER, INTEGER
Sortie INTEGER
Paramètre 1 Mois en chiffre
Paramètre 2 Année
Fournit le nombre de jour du mois choisi pour l'année choisie.
TestSQL
SELECT 29 AS ISCORRECT, F_DAYSOFMONTH(2, 2004) FROM RDB$DATABASE;

**F_LASTDAY**
Compatible avec GrUDF
(Presque) identique à F_DAYSOFMONTH; seuls l'ordre paramètres est inversé (d'abord puis mois).
Entrées INT, INT
Sortie INTEGER
Paramètre 1 Année
Paramètre 2 Mois en chiffre
Fournit le dernier jour du mois choisi pour l'année choisie.
TestSQL
SELECT 29 AS ISCORRECT, F_LASTDAY(2004, 2) FROM RDB$DATABASE;

**F_DAYOFWEEK**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit le jour de la semaine en valeur numérique de la date choisie. La semaine commence par le dimanche (= 1).
TestSQL
SELECT 1 AS ISCORRECT, F_DAYOFWEEK('22.08.2004') FROM RDB$DATABASE;

**F_DTIME**
Fonction de adhoc
Entrée TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle
Calcule le nombre de jours depuis le 31-12-1899 jusqu'à la date choisie.
Le comptage commence par 0.
TestSQL
SELECT 2 AS ISCORRECT, F_DTIME('03.01.1900') FROM RDB$DATABASE;

**F_CMONTHLONG**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie CSTRING(4)
Paramètre 1 Date et heure optionnelle
Fournit le nom du mois en anglais.
TestSQL
SELECT 'August' AS ISCORRECT, F_CMONTHLONG('23.08.2004') FROM RDB$DATABASE;

**F_CMONTHLONGLANG**
Fonction de adhoc
Entrée TIMESTAMP, CSTRING(2)
Sortie CSTRING(16)
Paramètre 1 Date et heure optionnelle
Paramètre 2 Indicatif de la langue pour le résultat.
de = allemand, uk = anglais, fr = français, es = espagnol, it = italien.
Fournit le nom du mois dans la langue choisie.
TestSQL
SELECT 'Août' AS ISCORRECT, F_CMONTHLONGLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

**F_CMONTHSHORT**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie CSTRING(4)
Paramètre 1 Date et heure optionnelle
Fournit l'abréviation du nom du mois en anglais.
TestSQL
SELECT 'Aug' AS ISCORRECT, F_CMONTHSHORT('23.08.2004') FROM RDB$DATABASE;

**F_CMONTHSHORTLANG**
Fonction de adhoc
Entrée TIMESTAMP, CSTRING(2)
Sortie CSTRING(16)
Paramètre 1 Date et heure optionnelle
Paramètre 2 Indicatif de la langue pour le résultat.
de = allemand, uk = anglais, fr = français, es = espagnol, it = italien.
Fournit l'abréviation du nom du mois dans la langue choisie
TestSQL
SELECT 'Aoû' AS ISCORRECT, F_CMONTHSHORTLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

**F_CDOWLONG**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie CSTRING(16)
Paramètre 1 Date et heure optionnelle
Fournit le nom du jour de la semaine en anglais.
TestSQL
SELECT 'Monday' AS ISCORRECT, F_CDOWLONG('23.08.2004') FROM RDB$DATABASE;

**F_CDOWLONGLANG**
Fonction de adhoc
Entrée TIMESTAMP, CSTRING(2)
Sortie CSTRING(16)
Paramètre 1 Date et heure optionnelle
Paramètre 2 Indicatif de la langue pour le résultat.
de = allemand, uk = anglais, fr = français, es = espagnol, it = italien.
Fournit le nom du jour de la semaine dans la langue choisie.
TestSQL
SELECT 'Lundi' AS ISCORRECT, F_CDOWLONGLANG('23.08.2004', 'fr')
FROM RDB$DATABASE;

**F_CDOWSHORT**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie CSTRING(4)
Paramètre 1 Date et heure optionnelle
Fournit l'abréviation du nom du jour de la semaine en anglais.
TestSQL
SELECT 'Mon' AS ISCORRECT, F_CDOWSHORT('23.08.2004') FROM RDB$DATABASE;

**F_CDOWSHORTLANG**
Fonction de adhoc
Entrée TIMESTAMP, CSTRING(2)
Sortie CSTRING(16)
Paramètre 1 Date et heure optionnelle
Paramètre 2 Indicatif de la langue pour le résultat.
de = allemand, uk = anglais, fr = français, es = espagnol, it = italien.
Fournit l'abréviation du nom du jour de la semaine dans la langue choisie.
TestSQL
SELECT 'Lun' AS ISCORRECT, F_CDOWSHORTLANG('23.08.2004', 'fr') FROM RDB$DATABASE;

**F_GFORMATD**
Fonction de adhoc
Entrée CSTRING(254), TIMESTAMP
Sortie CSTRING(254)
Paramètre 1 d = Jour (1 chiffre si < 10)
dd = Jour, toujours 2 chiffres
m = Mois (1 chiffre si < 10)
mm = Mois, toujours 2 chiffres
yy = Année (1 chiffre si < 10, sinon 2 chiffres)
yyyy = Année, toujours 4 chiffres
h = Heure (1 chiffre si < 10)
hh = Heure, toujours 2 chiffres
n = Minute (1 chiffre si < 10)
nn = Minute, toujours 2 chiffres
s = Seconde (1 chiffre si < 10)
ss = Seconde, toujours 2 chiffres
tous les autres signes repris dans le paramètre 1 resteront à leur place dans le résultat.
Paramètre 2 Date et heure optionnelle Formate la date et l'heure selon le paramètre 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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit l'année de la date choisie.
TestSQL
SELECT 2004 AS ISCORRECT, F_YEAR(' 22.08.2004 14:38:12') FROM RDB$DATABASE;

**F_QUARTER**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit le trimestre de la date choisie.
TestSQL
SELECT 3 AS ISCORRECT, F_QUARTER('23.08.2004 14:38:12') FROM RDB$DATABASE;

**F_MONTH**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit le numéro du mois de la date choisie.
TestSQL
SELECT 8 AS ISCORRECT, F_MONTH('23.08.2004 14:38:12') FROM RDB$DATABASE;

**F_WEEK**
Compatible avec FreeUDFLibC
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit la semaine de la date choisie.
Le comptage commence à la semaine 1 dans laquelle tombe le 1er janvier.
Remarque:
Il existe plusieurs variantes de numérotations des semaines:
La première semaine de l'année est
* celle dans laquelle tombe le 1er janvier
* la première semaine entière de l'année
* la première semaine contenant au moins 4 jours de la nouvelle année (ISO-8601)
Ces 3 variantes se réfèrent toutes à la semaine classique.
Des déviations existent comme en Grande-Bretagne où une année fiscale commence toujours le 6 avril.
La norme internationale ISO-8601 (1973) fixe le début de la semaine au lundi. La première semaine de l'année doit comprendre au minimum quatre jours. Cette définition équivaut à considérer la première semaine comme étant celle contenant le 4 janvier, ou encore la semaine qui contient le premier jeudi de l'année.
En Allemagne depuis 1976, le début de la semaine a été fixé au lundi: DIN 1355 (1974), DIN EN 28601 (1993).
D'après les normes citées, une année a 53 semaines lorsqu'elle commence ou qu'elle finit par un jeudi.
TestSQL
SELECT 41 AS ISCORRECT, F_WEEK('02.10.2005 14:38:12') FROM RDB$DATABASE;

**F_HOUR**
Compatible avec GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure
Fournit l'heure contenue dans le paramètre 1.
Si le paramètre 1 ne contient que la date, alors l'heure sera 0.
TestSQL
SELECT 14 AS ISCORRECT, F_HOUR('23.08.2004 14:38:12') FROM RDB$DATABASE;

**F_MINUTE**
Compatible avec GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure
Fournit les minutes contenues dans la date et heure spécifiée.
Si la date est seule fournie, le résultat sera 0.
TestSQL
SELECT 38 AS ISCORRECT, F_MINUTE('23.08.2004 14:38:12') FROM RDB$DATABASE;

**F_SECOND**
Compatible avec GrUDF
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure
Fournit les secondes contenues dans la date et heure spécifiée.
Si la date est seule fournie, le résultat sera 0.
TestSQL
SELECT 12 AS ISCORRECT, F_SECOND('23.08.2004 14:38:12') FROM RDB$DATABASE;

**F_YEAROFYEAR**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
(voir F_YEAR)
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure
Fournit l'année de la date spécifiée.
TestSQL
SELECT 2004 AS ISCORRECT, F_YEAROFYEAR(' 22.08.2004 14:38:12') FROM RDB$DATABASE;

**F_WEEKOFYEAR**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
(siehe F_WEEK)
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle
Fournit le numéro de semaine de l'année.
TestSQL
SELECT 34 AS ISCORRECT, F_WEEKOFYEAR('22.08.2004 14:38:12') FROM RDB$DATABASE;

**F_WOY**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée TIMESTAMP
Sortie CSTRING(7)
Paramètre 1 Date et heure
Fournit l'année et la semaine de l'année concaténées.
TestSQL
SELECT '200434' AS ISCORRECT, F_WOY('22.08.2004 14:38:12') FROM RDB$DATABASE;

**F_ENCODEDATE**
Compatible avec GrUDF
Entrée INT, INT, INT
Sortie DATE
Paramètre 1 Année
Paramètre 2 Mois
Paramètre 3 Jour
Construit une valeur DATE à partir de année, mois, jour
TestSQL
SELECT '20.02.2004' AS ISCORRECT, F_ENCODEDATE(2004, 2, 20) FROM RDB$DATABASE

**F_ENCODETIME**
Compatible avec GrUDF
Entrée INT, INT, INT
Sortie TIME
Paramètre 1 Heures
Paramètre 2 Minutes
Paramètre 3 Secondes
Construit une valeur TIME à partir des heures, minutes et secondes.
TestSQL
SELECT '09:45:53' AS ISCORRECT, F_ENCODETIME(9, 45, 53) FROM RDB$DATABASE

**F_ENCODETIMESTAMP**
Compatible avec GrUDF
Entrées INT, INT, INT, INT, INT, INT
Sortie TIMESTAMP
Paramètre 1 Année
Paramètre 2 Mois
Paramètre 3 Jour
Paramètre 4 Heures
Paramètre 5 Minutes
Paramètre 6 Secondes
Construit une valeur TIMESTAMP à partir des année, mois, jour, heure, minute, seconde.
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**
Compatible avec FreeUDFLibC
Entrée CSTRING(11)
Sortie TIME
Paramètre 1 heure «anglophile» sous forme de chaîne
Convertit l'heure «anglophile» (05:04:01 AM) en notation 24 heures. Le format d'entrée peut indiquer les éléments en un ou deux chiffres et la mention «AM» ou «PM» peut suivre avec ou sans espace, mais cette mention doit être en majuscules.
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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP
Sortie TIMESTAMP
Paramètre 1 Date et heure
Fournit l'heure spécifiée associée à la date 31-12-1899 (date 0).
Remarque:
Pour recevoir en retour l'heure seule, utiliser 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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée TIMESTAMP
Sortie TIMESTAMP
Paramètre 1 Date et heure
Fournit la date spécifiée associée à l'heure 00:00:00.
Remarque:
Pour obtenir seulement la date, exécuter 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**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1
Paramètre 2 Date et heure optionnelle 2
Compare l'égalité de deux dates-heures, mais en ne considérant que la partie date.
Résultat 1 = sont égales, 0 =ne sont pas égales
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**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée TIMESTAMP, TIMESTAMP
Sortie INT
Paramètre 1 Date et heure optionnelle 1
Paramètre 2 Date et heure optionnelle 2
Compare l'égalité de deux dates-heures en tenant compte à la fois de la date et de l'heure.
Résultat 1 = sont égales, 0 =ne sont pas égales
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**
Compatible avec FreeUDFLibC
Entrée TIMESTAMP
Sortie INTEGER
Paramètre 1 Date et heure optionnelle 1
Indique si l'année est bissextile.
Résultat 1 = année bissextile, 0 = non bissextile
Algorithme Y2k-fest (2000 était une année bissextile)
TestSQL

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

**F_MAXDATE**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée TIMESTAMP, TIMESTAMP
Sortie TIMESTAMP
Paramètre 1 Date et heure optionnelle 1
Paramètre 2 Date et heure optionnelle 2
Fournit la plus tardive des deux dates spécifiées.
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**
Compatible avec FreeUDFLib, FreeUDFLibC, FreeUDFLib AvERP
Entrée TIMESTAMP, TIMESTAMP
Sortie TIMESTAMP
Paramètre 1 Date et heure optionnelle 1
Paramètre 2 Date et heure optionnelle 2
Fournit la plus ancienne des deux dates spécifiées.
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**
Compatible avec FreeUDFLib AvERP, GrUDF
Entrée INTEGER
Sortie DATE
Paramètre 1 Année pour laquelle on recherche la date de Pâques.
Fournit la date du dimanche de Pâques de l'année spécifiée
TestSQL
SELECT '27.03.2005' AS ISCORRECT, F_OSTERDATUM(2005)
FROM RDB$DATABASE;

**F_ZEITDIFFERENZ**

Compatible avec FreeUDFLib AvERP, GrUDF

Entrée TIMESTAMP, TIMESTAMP, CSTRING(1)

Sortie DOUBLE

Paramètre 1 Date et heure optionnelle 1

Paramètre 2 Date et heure optionnelle 2

Paramètre 3 Forme de sortie

t = différence en jours

h = différence en heures

m = différence en minutes

s = différence en secondes

toutes les autres valeurs renvoient toujours 0.

Calcule la différence entre la date 1 et la date 2; le résultat est une valeur en virgule flottante tenant compte du choix du paramètre 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
```

## Fonctions BLOb - Conversions BLOb

### F_BLOBASPCHAR

Compatible avec FreeUDFLib, FreeUDFLib AvERP, GrUDF
Entrée BLOB
Sortie CSTRING(32760)
Paramètre 1 TextBlob qui doit être converti en VarChar.
Convertit TextBlob en VarChar
TestSQL (pour TestISO.GDB)
SELECT 'ein einzeiliger TextBLOb' AS ISCORRECT, TEXTBLOB,
F_BLOBASPCHAR(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 1

### F_STRBLOB

Compatible avec FreeUDFLib, GrUDF
Entrée CSTRING(32760)
Sortie BLOB
Paramètre 1 VarChar qui doit être converti en TextBlob.
Convertit VarChar en TextBlob

### F_BLOB2EXCEL

Fonction de adhoc
Entrée BLOb
Sortie BLOb
Paramètre 1 BLOb à transformer pour Excel
Pour pouvoir exporter vers Excel du texte de plusieurs lignes ou du texte contenant des guillemets doubles hauts (") venant de chaînes BLOb, certaines adaptations sont nécessaires. Cette fonction agit dans ce sens. Elle:
- ajoute au début et à la fin de la chaîne un guillemet double haut (")
- redouble chaque guillemet double haut (") qui se trouvait dans le texte
- enlève tous les CHR(13) de la chaîne
- tronque la chaîne BLOb pour ne pas dépasser 32760 caractères (limite d'un champ dans 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
Remarque:
Comme cela n'a pas de sens d'exporter vers Excel un (très) long texte venant d'une base de données, on doit tout d'abord le traiter avec les fonctions F_LEFT ou F_RIGHT pour rendre l'opération possible. Par exemple:
SELECT F_RIGHT(F_BLOB2EXCEL(BLObFeldAGENDA), 1000) FROM ... exporte, par exemple, les derniers 1000 caractères d'un champ BLOb d'un agenda.

**F_BLOBCAT**
Compatible avec GrUDF
Entrée BLOB, BLOB
Sortie BLOB
Paramètre 1 Texte Blob à concaténer avec le texte Blob du paramètre 2
Paramètre 2 Texte Blob à concaténer à la suite du texte Blob du paramètre 1
Concatène le contenu du second champ BLOb au premier pour former un tout.
La fonction ajoute un CRLF à la fin du premier BLOb avant l'ajout du second.
TestSQL (pour TestISO.GDB)
INSERT INTO BLOBTEST (TEXTBLOB) SELECT F_BLOBCAT(TEXTBLOB, (SELECT TEXTBLOB FROM BLOBTEST WHERE BLOBTESTID = 2)) FROM BLOBTEST WHERE BLOBTESTID = 1
Crée un nouveau enregistrement dans tableau BLOBTEST, où le champ TEXTBLOP résulte des contenus assemblés de TEXTBLOB de l'enregistrement avec le TEXTBLOBID 1 et l'enregistrement avec le TEXTBLOBID 2.

**F_BLOBCATSTR**
Compatibilité avec GrUDF
Entrées BLOB, CSTRING(32760)
Sortie BLOB
Paramètre 1 Texte BLOb à concaténer avec le chaîne de caractères du paramètre 2
Paramètre 2 Chaîne de caractères à concaténer au texte BLOb du paramètre 1
Concatène le contenu d'un champ BLOb et d'une chaîne de caractères et place le résultat dans un champ BLOb.
La fonction insère un CRLF à la fin du premier champ BLOb avec d'y concaténer la chaîne du paramètre 2
TestSQL (pour TestISO.GDB)
INSERT INTO BLOBTEST (TEXTBLOB) SELECT F_BLOBCATSTR(TEXTBLOB, 'Diese Zeile wurde angehängt') FROM BLOBTEST WHERE BLOBTESTID = 1
Crée dans la table BLOBTEST un nouvel enregistrement, où le champ TEXTBLOB résulte des contenus assemblés de TEXTBLOB de l'enregistrement avec TEXTBLOBID 1 et de la chaîne "Diese Zeile wurde angehängt".

**F_BLOBLEFT**
Compatible avec FreeUDFLib, FreeUDFLib Fournit le début du texte spécifié en une chaîne de la longueur donnée au paramètre 2.
Le comptage commence à 1.
TestSQL (pour TestISO.GDB)
SELECT 'ein einzeiliger' AS ISCORRECT, F_BLOBLEFT(TEXTBLOB, 15) FROM BLOBTEST WHERE BLOBTESTID = 1

**F_BLOBMID**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée BLOB, INT, INT
Sortie CSTRING(32760)
Paramètre 1 TextBlob dont on va extraire une chaîne
Paramètre 2 Position du début de la chaîne à extraire
Paramètre 3 Longueur de la chaîne à extraire
Extrait du texte fourni une chaîne de caractères d'une longueur spécifiée au paramètre 3 et commençant à la position spécifiées au paramètre 2. On compte à partir de 0 pour le paramètre 2.
TestSQL (pour TestISO.GDB)
SELECT 'zwei' AS ISCORRECT, F_BLOBMID(TEXTBLOB, 4, 4) FROM BLOBTEST WHERE BLOBTESTID = 2

**F_BLOBRIGHT**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée BLOB, INT
Sortie CSTRING(32760)
Paramètre 1 TextBlob
Paramètre 2 Nombre de caractères à partir de la droite.
Extrait de la fin du texte donné un nombre de caractères spécifié au paramètre 2; le comptage commence à 1.
TestSQL (pour TestISO.GDB)
SELECT 'dritten Zeile' AS ISCORRECT, F_BLOBRIGHT(TEXTBLOB, 13) FROM BLOBTEST WHERE BLOBTESTID = 3

**F_BLOBREPLACESTRING**
Fonction de adhoc
Entrées BLOB, CSTRING(254), CSTRING(254), INT, INT
Sortie BLOB
Paramètre 1 Le texte BLOb dans lequel une chaîne va être modifiée
Paramètre 2 La chaîne à modifier
Paramètre 3 La chaîne servant au remplacement
Paramètre 4 0 = un seul remplament, 1 = remplacement de toutes les chaînes de paramètre 2 trouvées
Paramètre 5 0 = distinction des lettres majuscules et minuscules, 1 = pas de distinction
Remplace une ou plusieurs fois la chaîne du paramètre 2 trouvée par la chaîne du paramètre 3. Elle peut prendre en compte les différences entre lettres majuscules et minuscules.
TestSQL (pour 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**
Fonction de adhoc
Entrée BLOB, CSTRING(1024)
Sortie INT
Paramètre 1 TextBLOb (dans lequel la position du paramètre 2 doit être trouvée)
Paramètre 2 chaîne de caractères 2 (position de laquelle doit être trouvée dans paramètre 1)
Indique la 1ère position dans BLOb, à laquelle la chaîne de caractères 2 commence
Le comptage commence à 0.
TestSQL (pour TestISO.GDB)
SELECT 4 AS ISCORRECT, F_BLOBSUBSTR(TEXTBLOB, 'einzeiliger') FROM BLOBTEST WHERE BLOBTESTID = 1

**F_BLOBLINE**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée BLOB, INT
Sortie CSTRING(32760)
Paramètre 1 TextBLOb
Paramètre 2 N° de la ligne à reproduire
Reproduit la ligne <paramètre 2> du TextBLOb
TestSQL (pour TestISO.GDB)
SELECT 'mit einer zweiten Zeile' AS ISCORRECT, F_BLOBLINE(TEXTBLOB, 2) FROM BLOBTEST WHERE BLOBTESTID = 3

**F_BLOBSIZE**
Compatible avec FreeUDFLib, FreeUDFLib AvERP
Entrée BLOB
Sortie INT
Paramètre 1 TextBlob
Dans le cas d'un TextBlob, fournit la longueur (analogue à F_STRINGLENGTH); dans le cas d'un BinairBlob, la taille des données.
TestSQL (pour TestISO.GDB)
Pour TextBLOb:
SELECT 50 AS ISCORRECT, F_BLOBSIZE(TEXTBLOB), F_STRINGLENGTH(F_BLOBASPCHAR(TEXTBLOB)) FROM BLOBTEST WHERE BLOBTESTID = 2
Pour BinärBLOb:
SELECT 1426 AS ISCORRECT, F_BLOBSIZE(BINAERBLOB) FROM BLOBTEST WHERE BLOBTESTID = 4

**F_BLOBMAXSEGMENTLENGTH**
Compatible avec FreeUDFLib
Entrée BLOB
Sortie INT
Paramètre 1 TextBLOb ou BinärBLOb
Détecte le nombre de bytes du plus grand segment BLOB
TestSQL (pour TestISO.GDB)
Pour TextBLOb:
SELECT 16384 AS ISCORRECT, F_BLOBMAXSEGMENTLENGTH(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 8
Pour BinärBLOb:
SELECT 16384 AS ISCORRECT, F_BLOBMAXSEGMENTLENGTH(BINAERBLOB) FROM BLOBTEST WHERE BLOBTESTID = 7

**F_BLOBSEGMENTCOUNT**
Compatible avec FreeUDFLib
Entrée BLOB
Sortie INT
Paramètre 1 TextBLOb oder BinärBLOb
Ermittelt die Anzahl der BLOB-Segmente
TestSQL (pour TestISO.GDB)
Für TextBLOb:
SELECT 3 AS ISCORRECT, F_BLOBSEGMENTCOUNT(TEXTBLOB) FROM BLOBTEST WHERE BLOBTESTID = 8
Für BinärBLOb:
SELECT 2 AS ISCORRECT, F_BLOBSEGMENTCOUNT(BINAERBLOB) FROM BLOBTEST WHERE BLOBTESTID = 7

**F_BLOBLINECOUNT**
Fonction de adhoc
Entrée BLOB
Sortie INT
Paramètre 1 TextBLOb
Détecte le nombre de lignes dans un TextBLOb
TestSQL (pour TestISO.GDB)
SELECT 3 AS ISCORRECT, F_BLOBLINECOUNT(TEXTBLOB) FROM BLOBTEST
WHERE BLOBTESTID = 3

**F_BLOBCOMPARE**
Compatible avec FreeUDFLib, GrUDF (BLOBICOMP)
Entrée BLOB
Sortie INT
Paramètre 1 TextBLOb
Compare deux BLOBs Binaires l'un avec l'autre
TestSQL (pour 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

Remarque (voir http://fr.wikipedia.org/wiki/UUID):
Un identifiant unique universel (UUID) est un repère d'identification normalisé utilisé dans le développement de logiciels. La norme émane de l'Open Software Foundation (OSF) comme une partie du Distributed Computing Environment (DCE). L'objectif de L'UUID est de fournir un repère d'identification unique au monde pour les systèmes partagés sans avoir recours à une coordination centrale contraignante. Un UUID est codé sur 16 octets et divisé en 5 groupes de signes. Dans sa forme normalisée, l'UUID apparaît comme suit:
550e8400-e29b-11d4-a716-446655440000
Bien que l'unicité d'un UUID fourni ne soit pas garantie, le clé de 2128 ou 3,4028*1038 est si élevée que la probabilité d'obtenir deux UUID identiques est pratiquement nulle. C'est ainsi que les UUID peuvent être utilisés sans contrôle centralisé et sans risque qu'un UUID représente des applications différentes. Ces données marquées par un UUID peuvent être introduites dans une base de données sans risque de conflit. La forme la plus répandue de la norme UUID est celle de Microsoft: Globally Unique Identifier (GUID).
Dans la version originale (Version 1) l'UUID est formé de l'adresse MAC que l'ordinateur génère et du nombre d'intervalles de cent nanosecondes écoulé depuis le début du calendrier grégorien selon l'horloge de l'ordinateur. En pratique l'algorithme est compliqué. Ce schéma a été critiqué car il manque de confidentialité en donnant l'identité de l'ordinateur et la date de création.
5 ver sions d'UUID sont dé fi nies dans la norme RFC4122 1 Basée sur l'hor loge de l'or di na teur a avec

l'

a dresse MAC unique se lon la norme IEEE 802

b avec une adresse MAC gé nérée par des nom bres aléa toi res se lon la norme IEEE 802

2 Ver sion de sé cu ri té DCE (avec les UIDs de POSIX)
3 Basée sur les es pa ces de noms (ha chage cryp to gra phique MD5)
4 Gé nérée par des nom bres aléa toi res
5 Basée sur les es pa ces de noms (ha chage cryp to gra phique SHA-1).

Seu les les ver sions 1 et 4 per met tent de réa li ser, par un même al go rithme, la dé cen tra li sa tion de la gé né ra tion d'un iden ti fiant unique qui soit com pa tible à la fois avec Win dows et Li nux. Ce n'est pas le cas des ver sions 3 et 5, car la ga rantie d'u ni ci té d'un es pace de nom (URL) n'est pas ab solue.
Comme cela dé pend de l'ap pli ca tion de sa voir si l'i den ti té de l'or di na teur et la date de créa tion peu vent ou non être ac ces si bles, 3 fonc tions de gé né ra tion d'UUID ont été pré vues:
F_UUID1MAC
Version 1 avec adresse MAC réelle
Iden ti té de l'or di na teur et date de créa tion dé ce la bles
F_UUID1RAND
Version 1 avec adresse MAC gé nérée par nom bres aléa toi res
Iden ti té de la date de créa tion seule dé ce lable
F_UUID4
Version 4
Plus d'i den ti té dé ce lable.
Jan New by est l'au teur d'une UUIDLIB qui met en oeuvre la ver sion 1b, mais dans la quelle l'a dresse MAC gé nérée est com primée au lieu d'u ti li ser le masque pré vu par la norme RFC4122. L'UUID ain si gé néré com prend alors 22 ca rac tè res au lieu de 36, ce qui donnerait à Fi re Bird / Inter base (?) un gain de vi tesse.
Nous avons éga le ment fait usage de cet al go rithme dans les fonc tions de conver sion, de sorte que pour cha cune des trois ver sions d'UUID gé né rées il y ait une ver sion com primée dis po nible (qui cor res ponde aux ca rac té ris ti ques de la ver sion de base):
F_UUID1MACCOMPR, F_UUID1RANDCOMPR, F_UUID4COMPR.

**F_UUID1MAC**
Fonction de adhoc
Entrée au cune
Sortie CSTRING(36)
Pro duit un iden ti fiant unique uni ver sel (UUID) de ver sion 1a.
TestSQL
SELECT F_UUID1MAC() FROM RDB$DATABASE;
Remarque:
S'il n'y a pas d'a dresse MAC dis ponible sur le Server, ac tionne la fonction F_UUID1RAND

**F_UUID1RAND**
Ré sul tat com pa tible avec la fonc tion GUID_CREATE de uui dlib
Entrée au cune
Sortie CSTRING(36)
Pro duit un iden ti fiant unique uni ver sel (UUID) de ver sion 1b.
TestSQL
SELECT F_UUID1RAND() FROM RDB$DATABASE;

**F_UUID4**
Fonction de adhoc
Entrée au cune
Sortie CSTRING(36)
Pro duit un iden ti fiant unique universel (UUID) de version 4.
TestSQL
SELECT F_UUID4() FROM RDB$DATABASE;

Re marque (à pro pos de uui dlib):
CREATE_UUID() gé nère un guid de 22 ca rac tè res sous forme com primée qui per mute les sec tions com po san tes afin de per mettre la com pres sion du préfixe des in dex de Firebird. Tous les ca rac tè res uti li sés dans cette UUID sont des ca rac tè res va li des pour URL.

**F_UUID1MACCOMPR**
Fonction de adhoc
Entrée au cune
Sortie CSTRING(22)
Pro duit un iden ti fiant unique uni ver sel (UUID) com pri mé de ver sion 1a.
TestSQL
SELECT F_UUID1MACCOMPR() FROM RDB$DATABASE;

**F_UUID1RANDCOMPR**
Ré sul tat com pa tible avec la fonc tion UUID_CREATE de uui dlib
Entrée au cune
Sortie CSTRING(22)
Pro duit un iden ti fiant unique uni ver sel (UUID) com pri mé de ver sion 1b.
TestSQL
SELECT F_UUID1RANDCOMPR() FROM RDB$DATABASE;

**F_UUID4COMPR**
Fonction de adhoc
Entrée au cune
Sortie CSTRING(22)
Pro duit un iden ti fiant unique uni ver sel (UUID) com pri mé de ver sion 4.
TestSQL
SELECT F_UUID4COMPR() FROM RDB$DATABASE;

**F_UUID2UUIDCOMPR**

Ré sul tat com pa tible avec la fonc tion GUID_TO_UUID de uui dlib

Entrée CSTRING(250)

Sortie CSTRING(22)

Pa ra mètre 1 UUID valide

Convertit un iden ti fiant unique universel (UUID) quelle que soit la ver sion en un UUID comprimé.

TestSQL (pour TestISO.GDB)

Trans mis sion comme pa ra mètre d'un UUID com pri mé au lieu d'un UUID normal:

SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID2UUIDCOMPR(UUIDCOMPR) FROM UUIDTEST;

Trans mission comme paramètre d'une chaîne de ca rac tè res quel conque au lieu d'un UUID normal:

SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID2UUIDCOMPR('1235abc de') FROM UUIDTEST;

Trans mission comme paramètre d'un UUID normal valide:

SELECT 'this is a valid input' AS ISCORRECT, F_UUID2UUIDCOMPR(UUID) FROM UUIDTEST;

**F_UUIDCOMPR2UUID**

Ré sul tat com pa tible avec la fonc tion UUID_TO_GUID de uui dlib

Entrée CSTRING(250)

Sortie CSTRING(36)

Paramètre 1 UUID com primé va lide

Conver tit un UUID com pri mé res pec tant la norme de com pres sion en un UUID normal.

TestSQL (pour TestISO.GDB)

Transmission comme paramètre d'un UUID normal au lieu d'un UUID comprimé:

SELECT 'INVALID INPUT' AS ISCORRECT, F_UUIDCOMPR2UUID(UUID) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;

Trans mission comme paramètre d'une chaîne de ca rac tè res quel conque au lieu d'un UUID comprimé:

SELECT 'INVALID INPUT' AS ISCORRECT, F_UUIDCOMPR2UUID('12345abcde') FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;

Trans mission comme paramètre d'un UUID com pri mé va lide:

SELECT 'this is valid Input' AS ISCORRECT, F_UUIDCOMPR2UUID(UUIDCOMPR) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL;

**F_UUIDVERSION**
Fonction de adhoc
Entrée CSTRING(250)
Sortie CSTRING(20)
Paramètre 1 UUID va lide
Four nit la va riante et la ver sion de l'UUID.
Les va rian tes pos si bles sont:

- 0 NCS (ré ser vé à la com pa ti bi li té avec les an cien nes ver sions)
- 10 La va riante ac tuelle se lon RFC4122
- 110 Microsoft (ré ser vé à la com pa ti bi li té avec les an cien nes ver sions)
- 111 Ré ser vé aux va rian tes fu tu res

Les ver sions pos si bles sont:

1. Basé sur l'hor loge
   a. Avec une adresse MAC unique se lon la norme IEEE 802
   b. Avec une adresse MAC cal culée par nombre aléa toire se lon la norme IEEE 802
2. Ver sion de sé cu ri té DCE (avec UIDs de POSIX)
3. Basé sur es pa ces de noms avec ha chage MD5
4. Généré par nombre aléa toire
5. Basé sur es pa ces de noms avec ha chage SHA-1.

La fonc tion renvoie:

- V1a
- V1b
- V3
- V4
- V5

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

**F_UUID1TIMESTAMP**
Fonction de adhoc
Entrée CSTRING(250)
Sortie TIMESTAMP
Paramètre 1 UUID va lide de ver sion 1
Four nit, à par tir d'un UUID1, les date et heure aux quel les cet UUID a été créé
Sile pa ra mètre trans mis n'est pas cor rect, la fonc tion renvoie 31.12.1899 00:00:00,
soit un temps 0 cor res pon dant au point de dé part des da tes dans Inter Base / Fi re Bird.
TestSQL (pour TestISO.GDB)
Trans mis sion comme pa ra mètre d'un UUID com primé au lieu d'un UUID normal:
SELECT '31.12.1899 00:00:00' AS ISCORRECT, F_UUID1TIMESTAMP(UUIDCOMPR)
FROM UUIDTEST;
Trans mis sion comme pa ra mètre d'un UUID nor mal va lide:
SELECT ZEITSTEMPEL AS ISCORRECT, F_UUID1TIMESTAMP(UUID) FROM
UUIDTEST;

**F_UUID1COMPRTIMESTAMP**
Fonction de adhoc
Entrée CSTRING(250)
Sortie TIMESTAMP
Paramètre 1 UUID comprimé valide de version 1
Four nit, à par tir d'un UUID1 com pri mé valide, les date et heure aux quel les cet UUID a été créé
Si le pa ra mètre trans mis n'est pas cor rect, la fonc tion renvoie 31.12.1899 00:00:00, soit un temps 0 cor res pon dant au point de dé part des da tes dans Inter Base / Fi re Bird.
TestSQL (pour TestISO.GDB)
Trans mis sion comme pa ra mètre d'un UUID nor mal au lieu d'un UUID comprimé:
SELECT '31.12.1899 00:00:00' AS ISCORRECT, F_UUID1COMPRTIMESTAMP(UUID) FROM UUIDTEST;
Trans mis sion comme pa ra mètre d'un UUID com pri mé va lide:
SELECT ZEITSTEMPEL AS ISCORRECT, F_UUID1COMPRTIMESTAMP(UUIDCOMPR) FROM UUIDTEST;

**F_UUID1MACMAC**
Fonction de adhoc
Entrée CSTRING(250)
Sortie CSTRING(17)
Paramètre 1 UUID valide de version 1a
Four nit l'a dresse MAC de l'or di na teur sur le quel l'UUID a été créé à par tir de UUID1MAC.
TestSQL (pour TestISO.GDB)
Trans mis sion comme pa ra mètre d'un UUID nor mal au lieu d'un UUID comprimé:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID1MACMAC(UUIDCOMPR) FROM UUIDTEST ORDER BY UUIDTESTID;
Trans mis sion comme pa ra mètre d'un UUID com pri mé va lide:
SELECT F_UUIDVERSION(UUID), F_UUID1MACMAC(UUID) FROM UUIDTEST ORDER BY UUIDTESTID;

**F_UUID1COMPRMAC**
Fonction de adhoc
Entrée CSTRING(22)
Sortie CSTRING(17)
Paramètre 1 UUID comprimé valide de version 1a
Four nit l'a dresse MAC de l'or di na teur sur le quel l'UUID a été créé à par tir de UUID1MAC comprimé.
TestSQL (pour TestISO.GDB)
Trans mis sion comme pa ra mètre d'un UUID nor mal au lieu d'un UUID comprimé:
SELECT 'INVALID INPUT' AS ISCORRECT, F_UUID1MACCOMPRMAC(UUID) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL ORDER BY UUIDTESTID;
Trans mis sion comme pa ra mètre d'un UUID com pri mé va lide:
SELECT F_UUIDVERSION(F_UUIDCOMPR2UUID(UUIDCOMPR)), F_UUID1MACCOMPRMAC(UUIDCOMPR) FROM UUIDTEST WHERE UUIDCOMPR IS NOT NULL ORDER BY UUIDTESTID;

**F_UUID1COMPARE**

Fonction de adhoc

| | |
|---|---|
| Entrées | CSTRING(250), CSTRING(250) |
| Sortie | SMALLINT |
| Parameter 1 | 1er UUID Version 1 |
| Parameter 2 | 2e UUID Version 1 |

Compare lexicalement deux identifiants uniques universels (UUID).
Lexicalement signifie de gauche à droite et par bloc, c'est-à-dire que

- on compare d'abord le bloc 1,
- puis le bloc 2,
- et enfin le bloc 3.

Si les blocs 1 ou 2 sont identiques, on passe au bloc suivant. Et la comparaison s'arrête dès qu'une différence est constatée ou lorsque l'on a atteint le bloc 3. La fonction renvoie alors

| | |
|---|---|
| –1 | u1 est lexi ca le ment avant u2 |
| 0 | u1 et u2 sont lexi ca le ment iden ti ques |
| 1 | u1 est lexi ca le ment après u2. |

Noter qu'un classement lexical n'est pas un classement dans le temps! Pour réaliser une comparaison dans le temps d'UUID de version 1 (par exemple classer les UUID d'après leur date de création): SELECT UUID, F_UUID1TIMESTAMP(UUID) FROM UUIDTEST ORDER BY 2 DESC;
Remarque:
Bien entendu il est possible d'utiliser cette fonction pour comparer d'autres versions que la version 1, cependant cela n'a pas de sens puisque seule la version 1 contient des informations de date que l'algorithme peut traiter.
TestSQL (pour 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**
Fonction de adhoc
Entrée CSTRING(32), CSTRING(8), CSTRING(32), CSTRING(8190), CSTRING(8190)
Sortie CSTRING(8190)
Paramètre 1 Chaîne représentant le membre 1
Paramètre 2 Opérateur de comparaison
=
<>
<
>
<=
>=
Chacun de ces opérateurs peut recevoir un préfixe «n» (numérique) lorsque les membres de la comparaison contiennent un nombre à virgule flottante, p. ex. n=
Paramètre 3 Chaîne représentant le membre 2
Paramètre 4 si l'expression comparant les 2 membres est satisfaite, le résultat est dans le 4e paramètre.
Paramètre 5 si l'expression comparant les 2 membres n'est pas satisfaite, le résultat est dans le 5e paramètre.
«Reconstitution» d'une boucle IF
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**
Fonction de adhoc
Entrée aucune
Sortie CSTRING(254)
Fournit la version de FreeAdhocUDF
TestSQL

```
SELECT F_VERSION() FROM RDB$DATABASE;
```

**Fonctions qui n'ont pas encore été prises en compte à ce jour (pas de besoin, autre possibilité)**

**de 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

**de FreeUDFLib C**
tout transposé

**de GrUDF** (18.11.2005)
- F_ASCII
- F_DAYSPAN -> fait la même chose que F_AGEINDAYS
- F_MONTHSPAN -> fait la même chose que F_AGEINMONTH
- F_YEARSPAN -> fait la même chose que F_AGEINYEARS
- F_WEEKSPAN -> fait la même chose que F_AGEINWEEKS
- F_SECONDSPAN -> fait la même chose que F_AGEINSECONDS
- F_MINUTESPAN -> fait la même chose que F_AGEINMINUTES
- F_HOURSPAN -> fait la même chose que F_AGEINHOURS
- F_MONTHSTART -> fait la même chose que CAST('01.02.2006' AS DATE)
- F_MONTHEND -> fait la même chose que CAST(F_LASTDAY(2006, 2) || '02.2006' AS DATE)
- F_DAYFRAC -> fait la même chose que CAST(F_AGEINSECONDS('20.02.2004 00:00:00', '20.02.2004 12:00:00') AS DOUBLE PRECISION) / 86400
  Oder CAST(F_AGEINHOURS('20.02.2004 00:00:00', '20.02.2004 12:00:00') AS DOUBLE PRECISION) / 24
- F_DOUBLE -> fait la même chose que CAST(1234.123 AS DOUBLE PRECISION)
- F_FRAC -> fait la même chose que 1234.12 - F_TRUNCATE(1234.12)
- F_TRUNCDEC
- F_CEIL -> voir ib_util, fonctions UDF gratuites de InterBase
- F_FLOOR -> voir ib_util, fonctions UDF gratuites de InterBase
- F_DIV -> voir ib_util, fonctions UDF gratuites de InterBase
- F_EXP
- F_SQRT -> voir ib_util, fonctions UDF gratuites de InterBase
- F_POWER
- F_LNXP1
- F_LOG10 -> voir ib_util, fonctions UDF gratuites de InterBase
- F_LOG2 -> voir ib_util, fonctions UDF gratuites de InterBase
- F_LOGN -> voir ib_util, fonctions UDF gratuites de InterBase
- F_PI -> voir ib_util, fonctions UDF gratuites de InterBase
- F_COS -> voir ib_util, fonctions UDF gratuites de InterBase
- F_SIN -> voir ib_util, fonctions UDF gratuites de InterBase

- F_TAN -> voir ib_util, fonctions UDF gratuites de InterBase
- F_COTAN -> voir ib_util, fonctions UDF gratuites de InterBase
- F_HYPOT -> voir ib_util, fonctions UDF gratuites de InterBase
- F_NOT
- F_AND
- F_OR
- F_XOR
- F_SHL
- F_SHR
- F_BLOBCOMP -> s. F_BLOBCOMPARE
- F_BLOBICOMP -> s. F_BLOBCOMPARE avec des fonctions ultérieures
  F_BLOBUPPER
- F_BLOBWRAP
- F_BLOBISEMPTY

## Possibilité d'échanger les UDF malgré dépendances

retour

Si on veut changer sur UDF FreeAdhoc il se pose souvent le problème que les UDFs existants homonymes, à cause de dépendances (utilisés dans les champs ComputedBy, trigger, views ou procedures) ne peuvent être effacés (cela a été possible jusqu'à InterBase 6).
A cet effet il y a un truc (pas tout à fait correct, cependant non dangereux):
éteindre temporairement les dépendances

```
/* UDF (ici pour toutes celles qui commencent par un F_) */
UPDATE RDB$DEPENDENCIES SET RDB$DEPENDED_ON_NAME = 'x' ||
RDB$DEPENDED_ON_NAME WHERE RDB$DEPENDED_ON_TYPE = 15
AND RDB$DEPENDED_ON_NAME STARTING WITH 'F';
/* Eliminer alors les fonctions UDF */
DROP  EXTERNAL FUNCTION F_.....;
DROP  EXTERNAL FUNCTION F_.....;
...
/* Ajouter maintenant les fonctions UDF  FreeAdhoc sur leur script  */
...
/* puis rétablir les dépendances (antérieures)  */
UPDATE RDB$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';
```