New features for text data

New string functions

Two new string functions were added:

LOWER()

A. dos Santos Fernandes

LOWER() returns the input argument converted to all lower-case characters.

Example

```
isql -q -ch dos850

SQL> create database 'test.fdb';
SQL> create table t (c char(1) character set dos850);
SQL> insert into t values ('A');
SQL> insert into t values ('E');
SQL> insert into t values ('Á');;
SQL> insert into t values ('É');
SQL>
C      LOWER
====== ======
A      a
E      e
Á      á
É      é
```

TRIM()

A. dos Santos Fernandes

TRIM trims characters (default: blanks) from the left and/or right of a string.

Syntax Pattern

```
TRIM <left paren> [ [ <trim specification> ] [ <trim character> ]
  FROM ] <value expression> <right paren>

   <trim specification> ::= LEADING | TRAILING | BOTH

   <trim character> ::= <value expression>
```

Rules

1. If <trim specification> is not specified, BOTH is assumed.
2. If <trim character> is not specified, ' ' is assumed.
3. If <trim specification> and/or <trim character> is specified, FROM should be specified.
4. If <trim specification> and <trim character> is not specified, FROM should not be specified.

Examples

A)

```
select
   rdb$relation_name,
   trim(leading 'RDB$' from rdb$relation_name)
from rdb$relations
   where rdb$relation_name starting with 'RDB$';
```

B)

```
select
   trim(rdb$relation_name) || ' is a system table'
from rdb$relations
   where rdb$system_flag = 1;
```

back to top of page

New string size functions

A. dos Santos Fernandes

Three new functions will return information about the size of strings:

- BIT_LENGTH returns the length of a string in bits.
- CHAR_LENGTH/CHARACTER_LENGTH returns the length of a string in characters.
- OCTET_LENGTH returns the length of a string in bytes.

Syntax Pattern

These three functions share a similar syntax pattern, as follows.-

```
<length function> ::=
{ BIT_LENGTH | CHAR_LENGTH | CHARACTER_LENGTH | OCTET_LENGTH } ( <value
expression> )
```

Example

```
select
   rdb$relation_name,
   char_length(rdb$relation_name),
   char_length(trim(rdb$relation_name))
```

```
from rdb$relations;
```

back to top of page

New INTL interface for non-ASCII character sets

A. dos Santos Fernandes

A feature of Firebird 2 is the introduction of a new interface for international character sets.

Originally described by N. Samofatov, the new interface features a number of enhancements that have been implemented by me.

Architecture

Firebird allows character sets and collations to be declared in any character field or variable declaration. The default character set can also be specified at database create time, to cause every CHAR/VARCHAR declaration that doesn't specifically included a CHARACTER SET clause to use it.

At attachment time you can specify the character set that the client is to use to read strings. If no “client” (or “connection”) character set is specified, character set NONE is assumed.

Two special character sets, NONE and OCTETS, can be used in declarations. However, OCTETS cannot be used as a connection character set. The two sets are similar, except that the space character of NONE is ASCII 0x20, whereas the space character OCTETS is 0x00. NONE and OCTETS are “special” in the sense that they do not follow the rule that other charsets do regarding conversions.

- With other character sets, conversion is performed as CHARSET1→UNICODE→CHARSET2.
- With NONE/OCTETS the bytes are just copied: NONE/OCTETS→CHARSET2 and CHARSET1→NONE/OCTETS.

Enhancements

Enhancements include:

Well-formedness checks

Some character sets (especially multi-byte) do not accept just any string. Now, the engine verifies that strings are well-formed when assigning from NONE/OCTETS and when strings sent by the client (the statement string and parameters).

Uppercasing

In FB 1.5.x only ASCII characters are uppercased in a character set's default (binary) collation order, which is used if no collation is specified.

For example,

```
isql -q -ch dos850

SQL> create database 'test.fdb';
SQL> create table t (c char(1) character set dos850);
SQL> insert into t values ('a');
SQL> insert into t values ('e');
SQL> insert into t values ('á');
SQL> insert into t values ('é');
SQL>
SQL> select c, upper(c) from t;

C      UPPER
====== ======
a      A
e      E
á      á
é      é
```

In FB 2.0 the result is:

```
C      UPPER
====== ======
a      A
e      E
á      Á
é      É
```

Maximum string length

In FB 1.5.x the engine does not verify the logical length of multi-byte character set (MBCS) strings. Hence, a UNICODE_FSS field takes three times as many characters as the declared field size, three being the maximum length of one UNICODE_FSS character).

This has been retained for compatibility for legacy character sets. However, new character sets (UTF8, for example) do not inherit this limitation.

sqlsubtype and attachment character set

When the character set of a CHAR or VARCHAR column is anything but NONE or OCTETS and the attachment character set is not NONE, the *sqlsubtype* member of an XSQLVAR pertaining to that column now contains the attachment (connection) character set number instead of the column's character set.

Enhancements for BLOBs

Several enhancements have been added for text BLOBs.

COLLATE clauses for BLOBs

A DML COLLATE clause is now allowed with BLOBs.

Example

```
select blob_column from table
where blob_column collate unicode = 'foo';
```

Full equality comparisons between BLOBs

Comparison can be performed on the entire content of a text BLOB.

Character set conversion for BLOBs

Conversion between character sets is now possible when assigning to a BLOB from a string or another BLOB.

back to top of page

INTL Plug-ins

Character sets and collations are installed using a manifest file.

The manifest file should be put in the $rootdir/intl with a .conf extension. It is used to locate character sets and collations in the libraries. If a character set/collation is declared more than once, it is not loaded and the error is reported in the log.

The symbol $(this) is used to indicate the same directory as the manifest file and the library extension should be omitted.

Example of a section from fbintl.conf

```
<intl_module fbintl>
  filename $(this)/fbintl
 </intl_module>

<charset ISO8859_1>
   intl_module fbintl
   collation ISO8859_1
    collation DA_DA
   collation DE_DE
   collation EN_UK
   collation EN_US
   collation ES_ES
   collation PT_BR
   collation PT_PT
</charset>

<charset WIN1250>
   intl_module fbintl
   collation WIN1250
   collation PXW_CSY
   collation PXW_HUN
   collation PXW_HUNDC
</charset>
```

back to top of page

New character sets/collations

UTF8 character set

The UNICODE_FSS character set has a number of problems: it's an old version of UTF8 that accepts malformed strings and does not enforce correct maximum string length. In FB 1.5.x UTF8 is an alias to UNICODE_FSS.

Now, UTF8 is a new character set, without the inherent problems of UNICODE_FSS.

UNICODE collations (for UTF8)

UCS_BASIC works identically to UTF8 with no collation specified (sorts in UNICODE code-point order). The UNICODE collation sorts using UCA (Unicode Collation Algorithm).

Sort order sample:

```
isql -q -ch dos850

SQL> create database 'test.fdb';
SQL> create table t (c char(1) character set utf8);
SQL> insert into t values ('a');
SQL> insert into t values ('A');
SQL> insert into t values ('á');
SQL> insert into t values ('b');
SQL> insert into t values ('B');
SQL> select * from t order by c collate ucs_basic;

C
======
A
B
a
b
```

```
á

SQL> select * from t order by c collate unicode;

C
======
a
A
á
b
B
```

Brazilian collations

Two case-insensitive/accent-insensitive collations were created for Brazil: WIN_PTBR (for WIN1252) and PT_BR (for ISO8859_1).

Sort order and equality sample

```
isql -q -ch dos850

SQL> create database 'test.fdb';
SQL> create table t (c char(1) character set iso8859_1 collate pt_br);
SQL> insert into t values ('a');
SQL> insert into t values ('A');
SQL> insert into t values ('á');
SQL> insert into t values ('b');
SQL> select * from t order by c;

C
======
A
a
á
b

SQL> select * from t where c = 'â';

C
======
a
A
â
```

Drivers

New character sets and collations are implemented through dynamic libraries and installed in the server with a manifest file in the intl subdirectory. For an example, see fbintl.conf.

Not all implemented character sets and collations need to be listed in the manifest file. Only those listed are available and duplications are not loaded.

Adding more character sets to a database

For installing additional character sets and collations into a database, the character sets and collations should be registered in the database's system tables (rdb$character_sets and rdb$collations). The file misc/intl.sql, in your Firebird 2 installation, is a script of stored procedures for registering and unregistering them.

back to top of page

New character sets and collations implemented

ES_ES_CI_AI for ISO8859_1 character set

A. dos Santos Fernandes

Spanish language case- and accent-insensitive collation for ISO8859_1 character set.

KOI8-R

O. Loa, A. Karyakin

Russian language character set and dictionary collation.

KOI8-U

O. Loa, A. Karyakin

Ukrainian language character set and dictionary collation.

WIN1257_LV

O. Loa, A. Karyakin

Latvian dictionary collation.

WIN1257_LT

O. Loa, A. Karyakin

Lithuanian dictionary collation.

WIN1257_EE

O. Loa, A. Karyakin

Estonian dictionary collation.

UTF8

A. dos Santos Fernandes

Unicode 4.0 support with UTF8 character set and collations UCS_BASIC and UNICODE.

Brazilian collations

A. dos Santos Fernandes, P. H. Albanez

1. Collation PT_BR for ISO8859_character set.
2. Collation WIN_PTBR for WIN1252 character set.

Bosnian Collation

F. Hasovic

New Bosnian language collation BS_BA was added for WIN1250 character set.

Czech Collations

I. Prenosil, A. dos Santos Fernandes

1. WIN_CZ: case-insensitive Czech language collation for WIN1250 character set.
2. WIN_CZ_CI_AI: case-insensitive, accent-insensitive Czech language collation for WIN1250 character set.

Vietnamese Character Set

Nguyen The Phuong, A. dos Santos Fernandes

Charset WIN1258 for Vietnamese language.

Polish Collation

Jaroslaw Glowacki, A. dos Santos Fernandes

Added new collation ISO_PLK for ISO8859_2 charset (Polish language).

back to top of page

Character set bug fixes

A. dos Santos Fernandes

The following bugs related to character sets and collations were fixed:

SF #1073212 An ORDER BY on a big column with a COLLATE clause would terminate the server.

SF #939844 A query in a UNICODE database would throw a GDS Exception if it was longer than 263 characters.

SF #977785 Wrong character lengths were being returned from some multi-byte character sets (UTF-8, East-Asian charsets).

SF #536243 A correct result is now returned when the UPPER() function is applied to a UNICODE_FSS string.

SF #942726 UPPER did not convert aacute to Aacute for **ISO8859_1**.

SF #544630 Some problems were reported when connecting using UNICODE.

SF #540547 Some problems involving concatenation, numeric fields and character set were fixed.

Unregistered bug A query could produce different results, depending on the presence of an index, when the last character of the string was the first character of a compression pair.

Unregistered bug SUBSTRING did not work correctly with a BLOB in a character set.

Unregistered bug Pattern matching with multi-byte BLOBs was being performed in binary mode.

Unregistered bug Connecting with a multi-byte character set was unsafe if the database had columns using a different character set.