International Language Support (INTL)

Adriano dos Santos Fernandes

This chapter describes the new international language support interface that was introduced with Firebird 2. Since then, a number of additions and improvements have been added, including the ability to implement UNICODE collations from external libraries generically. New DDL syntax has been introduced to assist with this task, in the form of the CREATE COLLATION statement.

New INTL interface for non-ASCII character sets

A. dos Santos Fernandes

Originally described by N. Samofatov, Firebird 2's new interface for international character sets features many 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 does not specifically include a CHARACTER SET clause to use this default.

At attachment time you normally 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 follow different rules from those applicable to other character sets 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 that the new system brings 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 Firebird 1.5.x, only the ASCII-equivalent characters are uppercased in any character set's default (binary) collation order, which is the one that 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 Firebird 2 the result is:

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

back to top of page

Maximum string length

In v.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 character set-related 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 file /intl/fbintl.conf is an example of a manifest file. The following snippet is an excerpt from /intl/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>
```

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

back to top of page

New character sets/collations

Two character sets introduced in Firebird 2 will be of particular interest if you have struggled with the shortcomings of UNICODE_FSS in past versions.

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
```

back to top of page

Developments in v.2.1

The 2.1 release sees further capabilities implemented for

- using ICU charsets through fbintl.
- UNICODE collation (charset_UNICODE) being available for all fbintl charsets.
- using collation attributes.
- CREATE/DROP COLLATION statements.
- SHOW COLLATION and collation extraction in isql.
- Verifying that text blobs are well-formed.
- Transliterating text blobs automatically.

ICU character sets

All non-wide and ASCII-based character sets present in ICU can be used by Firebird 2.1. To reduce the size of the distribution kit, we customize ICU to include only essential character sets and any for which there was a specific feature request.

If the character set you need is not included, you can replace the ICU libraries with another complete module, found at our site or already installed in your operating system.

Registering an ICU character set module

To use an alternative character set module, you need to register it in two places:

1. in the server's language configuration file, intl/fbintl.conf,
2. in each database that is going to use it.

Registering a character set on the server

Using a text editor, register the module in intl/fbintl.conf, as follows.-

```
<charset          NAME>
    intl_module      fbintl
    collation NAME   [REAL-NAME]
</charset>
```

Registering a character set in a database

To register the module in a database, run the procedure sp_register_character_set, the source for which can be found in misc/intl.sql beneath your Firebird 2.1 root.

Using the stored procedure:

A sample

Here is the sample declaration in fbintl.conf:

```
<charset        GB>
    intl_module    fbintl
    collation      GB GB18030
</charset>
```

The stored procedure takes two arguments: a string that is the character set's identifier as declared in the configuration file and a smallint that is the maximum number of bytes a single character can occupy in the encoding. For our example:

```
execute procedure sp_register_character_set ('GB', 4);
```

The CREATE COLLATION statement

Syntax for CREATE COLLATION

```
CREATE COLLATION <name>
    FOR <charset>
    [ FROM <base> | FROM EXTERNAL ('<name>') ]
    [ NO PAD | PAD SPACE ]
    [ CASE SENSITIVE | CASE INSENSITIVE ]
    [ ACCENT SENSITIVE | ACCENT INSENSITIVE ]
    [ '<specific-attributes>' ]
```

Note: Specific attributes should be separated by a semicolon and are case sensitive.

Examples

```
/* 1 */
CREATE COLLATION UNICODE_ENUS_CI
    FOR UTF8
    FROM UNICODE
    CASE INSENSITIVE
    'LOCALE=en_US';

/* 2 */
CREATE COLLATION NEW_COLLATION
    FOR WIN1252
    PAD SPACE;

/* NEW_COLLATION should be declared in .conf file in $root/intl directory */
```

back to top of page

The UNICODE collations

The UNICODE collations (case sensitive and case insensitive) can be applied to any character set that is present in fbintl. They are already registered in fbintl.conf, but you need to register them in the databases, with the desired associations and attributes.

Naming conventions

The naming convention you should use is charset_collation. For example,

```
create collation win1252_unicode
  for win1252;

create collation win1252_unicode_ci
  for win1252
    from win1252_unicode
    case insensitive;
```

Note: The character set name should be as in fbintl.conf (i.e. ISO8859_1 instead of ISO88591, for example).

back to top of page

Specific attributes for collations

Note: Some attributes may not work with some collations, even though they do not report an error.

DISABLE-COMPRESSIONS

Disable compressions (aka contractions) changing the order of a group of characters.

Valid for collations of narrow character sets.

Format: DISABLE-COMPRESSIONS={0 | 1}

Example

```
DISABLE-COMPRESSIONS=1
```

DISABLE-EXPANSIONS

Disable expansions changing the order of a character to sort as a group of characters.

Valid for collations of narrow character sets.

Format: DISABLE-EXPANSIONS={0 | 1}

Example

DISABLE-EXPANSIONS=1

ICU-VERSION

Specify what version of ICU library will be used. Valid values are the ones defined in the config file (intl/fbintl.conf) in entry intl_module/icu_versions.

Valid for UNICODE and UNICODE_CI.

Format: ICU-VERSION={default | major.minor}

Example

```
ICU-VERSION=3.0
```

LOCALE

Specify the collation locale.

Valid for UNICODE and UNICODE_CI. Requires complete version of ICU libraries.

Format: LOCALE=xx_XX

Example

```
LOCALE=en_US
```

MULTI-LEVEL

Uses more than one level for ordering purposes.

Valid for collations of narrow character sets.

Format: MULTI-LEVEL={0 | 1}

Example

```
MULTI-LEVEL=1
```

SPECIALS-FIRST

Order special characters (spaces, symbols, etc) before alphanumeric characters.

Valid for collations of narrow character sets.

Format: SPECIALS-FIRST={0 | 1}

Example

```
SPECIALS-FIRST=1
```

back to top of page

Collation changes in v.2.1

Spanish

ES_ES (as well as the new ES_ES_CI_AI) collation automatically uses attributes DISABLE-COMPRESSIONS=1;SPECIALS-FIRST=1.

Note: The attributes are stored at database creation time, so the changes do not apply to databases with ODS < 11.1.

The ES_ES_CI_AI collation was standardised to current usage.

UTF-8

Case-insensitive collation for UTF-8. See feature request CORE-972.

back to top of page

Metadata text conversion

Firebird versions 2.0.x had two problems related to character sets and metadata extraction:

1. When creating or altering objects, text associated with metadata was not transliterated from the client character set to the system (UNICODE_FSS) character set of these BLOB columns. Instead, raw bytes were stored there.

The types of text affected were PSQL sources, descriptions, text associated with constraints and defaults, and so on.

Note: Even in the current version (2.1 Beta 1) the problem can still occur if CREATE or ALTER operations are performed with the connection character set as NONE or UNICODE_FSS and you are using non-UNICODE_FSS data.

1. In reads from text BLOBs, transliteration from the BLOB character set to the client character set was not being performed.

Repairing your metadata text

If your metadata text was created with non-ASCII encoding, you need to repair your database in order to read the metadata correctly after upgrading it to v.2.1.

Important: The procedure involves multiple passes through the database, using scripts. It is strongly recommended that you disconnect and reconnect before each pass.

The database should already have been converted to ODS11.1 by way of a gbak backup and restore.

Before doing anything, make a copy of the database.

In the examples that follow, the string $fbroot$ represents the path to your Firebird installation root directory, e.g. /opt/firebird.

Create the procedures in the database

```
[1] isql /path/to/your/database.fdb
[2] SQL> input '$fbroot$/misc/upgrade/metadata/metadata_charset_create.sql';
```

Check your database

```
[1] isql /path/to/your/database.fdb
[2] SQL> select * from rdb$check_metadata;
```

The rdb$check_metadata procedure will return all objects that are touched by it.

- If no exception is raised, your metadata is OK and you can go to the section Remove the upgrade procedures.

- Otherwise, the first bad object is the last one listed before the exception.

Fixing the metadata

To fix the metadata, you need to know in what character set the objects were created. The upgrade script will work correctly only if all your metadata was created using the same character set.

```
[1] isql /path/to/your/database.fdb
[2] SQL> input
'$fbroot$/misc/upgrade/metatdata/metadata_charset_create.sql';
[3] SQL> select * from rdb$fix_metadata('WIN1252'); -- replace WIN1252 by
your charset
[4] SQL> commit;
```

The rdb$fix_metadata procedure will return the same data as rdb$check_metadata, but it will change the metadata texts.

Important: It should be run once!

After this, you can remove the upgrade procedures.

Remove the upgrade procedures

```
[1] isql /path/to/your/database.fdb
[2] SQL> input '$fbroot$/misc/upgrade/metadata/metadata_charset_drop.sql';
```

Supported character sets

See Appendix B at the end of these notes, for a full listing of the supported character sets.