Extracting META information from Interbase/Firebird SQL (INFORMATION_SCHEMA)

13 November 2007 22:30

By: http://www.alberton.info/firebird_sql_meta_info.html

The SQL 2003 Standard introduced a new schema called **INFORMATION_SCHEMA**. PostgreSQL, SQL Server and now MySQL have it. ORACLE, DB2, Sybase, Ingres, Informix and other DBMS have something similar, usually called **System Tables**. The INFORMATION_SCHEMA is meant to be a set of views you can query using regular SELECT statements, for instance if you need to know something about the defined triggers, or the structure of a table to which you have access. **Firebird** doesn't have it, but you can retrieve pretty much everything you need from the system tables. These are metadata tables, their names start with "**RDB$**". Let's see how we can retrieve some useful informations from them.

Test data

We need a few sample tables, indices and views to test the following queries, so let's create them. We also create a sample TRIGGER to emulate the autoincrement feature of mysql, and a simple stored procedure.

```
-- sample data to test Firebird system tables

-- TABLE TEST
CREATE TABLE TEST (
  TEST_NAME CHAR(30) CHARACTER SET NONE NOT NULL COLLATE NONE,
  TEST_ID INTEGER DEFAULT '0' NOT NULL,
  TEST_DATE TIMESTAMP NOT NULL
);
ALTER TABLE TEST ADD CONSTRAINT PK_TEST PRIMARY KEY (TEST_ID);

-- TABLE TEST2 with some CONSTRAINTs and an INDEX
CREATE TABLE TEST2 (
  ID INTEGER NOT NULL,
  FIELD1 INTEGER,
  FIELD2 CHAR(15) CHARACTER SET NONE COLLATE NONE,
  FIELD3 VARCHAR(50) CHARACTER SET NONE COLLATE NONE,
  FIELD4 INTEGER,
  FIELD5 INTEGER,
  ID2 INTEGER NOT NULL
);
ALTER TABLE TEST2 ADD CONSTRAINT PRIMARY KEY (ID2);
CREATE UNIQUE INDEX TEST2_FIELD1ID_IDX ON TEST2(ID, FIELD1);
CREATE UNIQUE INDEX TEST2_FIELD4_IDX ON TEST2(FIELD4);
CREATE INDEX TEST2_FIELD5_IDX ON TEST2(FIELD5);

-- TABLE NUMBERS
CREATE TABLE NUMBERS (
  NUMBER INTEGER DEFAULT '0' NOT NULL,
  EN CHAR(100) CHARACTER SET ISO8859_1 NOT NULL COLLATE ISO8859_1,
  FR CHAR(100) CHARACTER SET ISO8859_1 NOT NULL COLLATE ISO8859_1
);

-- TABLE NEWTABLE
CREATE TABLE NEWTABLE (
  ID INT DEFAULT 0 NOT NULL,
  SOMENAME VARCHAR (12),
  SOMEDESCRIPTION VARCHAR (12)
);
ALTER TABLE NEWTABLE ADD CONSTRAINT PKINDEX_IDX PRIMARY KEY (ID);
```

```
CREATE GENERATOR NEWTABLE_SEQ;

-- VIEW on TEST
CREATE VIEW "testview"(
  TEST_NAME,
  TEST_ID,
  TEST_DATE
) AS
SELECT *
FROM TEST
WHERE TEST_NAME LIKE 't%';

-- VIEW on NUMBERS
CREATE VIEW "numbersview"(
  NUM,
  EN,
  FR
) AS
SELECT *
FROM NUMBERS
WHERE NUMBER > 100;

-- TRIGGER on NEWTABLE (emulate autoincrement)
SET TERM ^ ;

CREATE TRIGGER AUTOINCREMENTPK FOR NEWTABLE
ACTIVE BEFORE INSERT POSITION 0
AS BEGIN
  IF (NEW.ID IS NULL OR NEW.ID = 0) THEN
     NEW.ID = GEN_ID(NEWTABLE_SEQ, 1);
END^

SET TERM ; ^

-- SAMPLE STORED PROCEDURE
SET TERM ^ ;

CREATE PROCEDURE getEnglishNumber(N INTEGER)
RETURNS (
  english_number CHAR(100)
)
AS
BEGIN
  FOR
    SELECT EN
    FROM NUMBERS
    WHERE NUMBER = :N
    INTO :english_number
  DO
    BEGIN
      SUSPEND;
    END
END ^

SET TERM ; ^
```

List TABLEs

Here's the query that will return the names of the tables defined in the current database:

```
SELECT DISTINCT RDB$RELATION_NAME
```

```
  FROM RDB$RELATION_FIELDS
 WHERE RDB$SYSTEM_FLAG=0;

-- or

SELECT RDB$RELATION_NAME
  FROM RDB$RELATIONS
 WHERE RDB$SYSTEM_FLAG=0;
```

NB: the above queries will list both the user-defined tables AND views. To exclude the VIEWs from the resultset, you can write one of these queries:

```
SELECT DISTINCT RDB$RELATION_NAME
  FROM RDB$RELATION_FIELDS
 WHERE RDB$SYSTEM_FLAG=0
   AND RDB$VIEW_CONTEXT IS NULL;

-- or

SELECT RDB$RELATION_NAME
  FROM RDB$RELATIONS
 WHERE RDB$SYSTEM_FLAG=0
   AND RDB$VIEW_BLR IS NULL;
```

List VIEWs

Here's the query that will return the names of the VIEWs defined in the current database:

```
SELECT DISTINCT RDB$VIEW_NAME
  FROM RDB$VIEW_RELATIONS;

-- show only the VIEWs referencing a given table

SELECT DISTINCT RDB$VIEW_NAME
  FROM RDB$VIEW_RELATIONS
 WHERE RDB$RELATION_NAME='TEST';
```

List users

```
SELECT DISTINCT RDB$USER
  FROM RDB$USER_PRIVILEGES;
```

List INDICES

Here's the query that will return the names of the INDICES defined in the TEST2 table. NB: the CONSTRAINTs are not listed

```
SELECT RDB$INDEX_NAME
  FROM RDB$INDICES
 WHERE RDB$RELATION_NAME='TEST2'
   AND RDB$UNIQUE_FLAG IS NULL
   AND RDB$FOREIGN_KEY IS NULL;
```

Detailed INDEX info

If you want to know which table columns are referenced by an index, try with this query:

```
SELECT RDB$INDEX_SEGMENTS.RDB$FIELD_NAME AS field_name,
```

```
          RDB$INDICES.RDB$DESCRIPTION AS description,
          (RDB$INDEX_SEGMENTS.RDB$FIELD_POSITION + 1) AS field_position
     FROM RDB$INDEX_SEGMENTS
LEFT JOIN RDB$INDICES ON RDB$INDICES.RDB$INDEX_NAME =
RDB$INDEX_SEGMENTS.RDB$INDEX_NAME
LEFT JOIN RDB$RELATION_CONSTRAINTS ON RDB$RELATION_CONSTRAINTS.RDB$INDEX_NAME =
RDB$INDEX_SEGMENTS.RDB$INDEX_NAME
    WHERE UPPER(RDB$INDICES.RDB$RELATION_NAME)='TEST2'         -- table name
      AND UPPER(RDB$INDICES.RDB$INDEX_NAME)='TEST2_FIELD5_IDX' -- index name
      AND RDB$RELATION_CONSTRAINTS.RDB$CONSTRAINT_TYPE IS NULL
 ORDER BY RDB$INDEX_SEGMENTS.RDB$FIELD_POSITION
```

List CONSTRAINTs

Here's the query that will return the names of the CONSTRAINTs defined in the TEST2 table:

```
SELECT RDB$INDEX_NAME
  FROM RDB$INDICES
 WHERE RDB$RELATION_NAME='TEST2' -- table name
   AND (
       RDB$UNIQUE_FLAG IS NOT NULL
    OR RDB$FOREIGN_KEY IS NOT NULL
   );
```

List table fields

Here's the query that will return the names of the fields of the TEST2 table:

```
SELECT RDB$FIELD_NAME
  FROM RDB$RELATION_FIELDS
 WHERE RDB$RELATION_NAME='TEST2';
```

If you want some more info about the field definitions, you can retrieve a larger subset of the fields available in the RDB$RELATIONS_FIELDS table:

```
SELECT RDB$FIELD_NAME AS field_name,
       RDB$FIELD_POSITION AS field_position,
       RDB$DESCRIPTION AS field_description,
       RDB$DEFAULT_VALUE AS field_default_value,
       RDB$NULL_FLAG AS field_not_null_constraint
  FROM RDB$RELATION_FIELDS
 WHERE RDB$RELATION_NAME='TEST2';  -- table name
```

Detailed table field info

If you want a really detailed description of the field, you have to join the RDB$RELATIONS_FIELDS table with RDB$FIELDS:

```
SELECT r.RDB$FIELD_NAME AS field_name,
        r.RDB$DESCRIPTION AS field_description,
        r.RDB$DEFAULT_VALUE AS field_default_value,
        r.RDB$NULL_FLAG AS field_not_null_constraint,
        f.RDB$FIELD_LENGTH AS field_length,
        f.RDB$FIELD_PRECISION AS field_precision,
        f.RDB$FIELD_SCALE AS field_scale,
        CASE f.RDB$FIELD_TYPE
          WHEN 261 THEN 'BLOB'
          WHEN 14 THEN 'CHAR'
          WHEN 40 THEN 'CSTRING'
```

```
          WHEN 11 THEN 'D_FLOAT'
          WHEN 27 THEN 'DOUBLE'
          WHEN 10 THEN 'FLOAT'
          WHEN 16 THEN 'INT64'
          WHEN 8 THEN 'INTEGER'
          WHEN 9 THEN 'QUAD'
          WHEN 7 THEN 'SMALLINT'
          WHEN 12 THEN 'DATE'
          WHEN 13 THEN 'TIME'
          WHEN 35 THEN 'TIMESTAMP'
          WHEN 37 THEN 'VARCHAR'
          ELSE 'UNKNOWN'
        END AS field_type,
        f.RDB$FIELD_SUB_TYPE AS field_subtype,
        coll.RDB$COLLATION_NAME AS field_collation,
        cset.RDB$CHARACTER_SET_NAME AS field_charset
   FROM RDB$RELATION_FIELDS r
   LEFT JOIN RDB$FIELDS f ON r.RDB$FIELD_SOURCE = f.RDB$FIELD_NAME
   LEFT JOIN RDB$COLLATIONS coll ON f.RDB$COLLATION_ID = coll.RDB$COLLATION_ID
   LEFT JOIN RDB$CHARACTER_SETS cset ON f.RDB$CHARACTER_SET_ID =
cset.RDB$CHARACTER_SET_ID
  WHERE r.RDB$RELATION_NAME='TEST2'  -- table name
ORDER BY r.RDB$FIELD_POSITION;
```

List GENERATORs (sequences)

A **GENERATOR** is a sequential number that can be automatically inserted in a column with the `GEN_ID()` function. A `GENERATOR` is often used to ensure a unique value in a `PRIMARY KEY` that must uniquely identify the associated row.

```
SELECT RDB$GENERATOR_NAME
  FROM RDB$GENERATORS
 WHERE RDB$SYSTEM_FLAG IS NULL;
```

List TRIGGERs

```
SELECT * FROM RDB$TRIGGERS
 WHERE RDB$SYSTEM_FLAG IS NULL;

-- list only the triggers for a given table

SELECT * FROM RDB$TRIGGERS
 WHERE RDB$SYSTEM_FLAG IS NULL
   AND RDB$RELATION_NAME='NEWTABLE';  -- table name
```

List FUNCTIONs (UDF)

```
SELECT * FROM RDB$FUNCTIONS
 WHERE RDB$SYSTEM_FLAG IS NULL;
```

List Stored Procedures

```
SELECT * FROM RDB$PROCEDURES;
```

List FOREIGN KEY constraints

As a bonus, I'm going to show how to get more information for the FOREIGN KEY constraints on a table, i.e.

one query that returns the FK names, the names of the table and field they're tied to, the names of the table and field they reference and the action on update/delete. These are the two test tables and the FK constraint:

```
CREATE TABLE "a" (

  "id" INTEGER NOT NULL,
  "name" VARCHAR(20) DEFAULT '' NOT NULL
);
ALTER TABLE "a" ADD PRIMARY KEY ("id");

CREATE TABLE "b" (
  "id" INTEGER NOT NULL,
  "a_id" INTEGER DEFAULT 0 NOT NULL
);
ALTER TABLE "b" ADD PRIMARY KEY ("id");
ALTER TABLE "b" ADD CONSTRAINT "FK_b" FOREIGN KEY ("a_id")
REFERENCES "a"("id") ON DELETE CASCADE ON UPDATE CASCADE;

And this is the query that will return the above mentioned values (replace the
table name in the last line to get the FK definitions relative to another
table):

SELECT DISTINCT

          rc.RDB$CONSTRAINT_NAME AS "constraint_name",
          rc.RDB$RELATION_NAME AS "on table",
          d1.RDB$FIELD_NAME AS "on field",
          d2.RDB$DEPENDED_ON_NAME AS "references table",
          d2.RDB$FIELD_NAME AS "references field",
          refc.RDB$UPDATE_RULE AS "on update",
          refc.RDB$DELETE_RULE AS "on delete"
     FROM RDB$RELATION_CONSTRAINTS AS rc
LEFT JOIN RDB$REF_CONSTRAINTS refc ON rc.RDB$CONSTRAINT_NAME =
refc.RDB$CONSTRAINT_NAME
LEFT JOIN RDB$DEPENDENCIES d1 ON d1.RDB$DEPENDED_ON_NAME = rc.RDB$RELATION_NAME
LEFT JOIN RDB$DEPENDENCIES d2 ON d1.RDB$DEPENDENT_NAME = d2.RDB$DEPENDENT_NAME
    WHERE rc.RDB$CONSTRAINT_TYPE = 'FOREIGN KEY'
      AND d1.RDB$DEPENDED_ON_NAME <> d2.RDB$DEPENDED_ON_NAME
      AND d1.RDB$FIELD_NAME <> d2.RDB$FIELD_NAME
      AND rc.RDB$RELATION_NAME = 'b'  -- table name
```

Detailed CONSTRAINT info

If you want to retrieve detailed info from any constraint (fields, type, rules, referenced table and fields for FOREIGN KEYs, etc.) given its name and table, here's the query to do so:

```
SELECT rc.RDB$CONSTRAINT_NAME,

          s.RDB$FIELD_NAME AS field_name,
          rc.RDB$CONSTRAINT_TYPE AS constraint_type,
          i.RDB$DESCRIPTION AS description,
          rc.RDB$DEFERRABLE AS is_deferrable,
          rc.RDB$INITIALLY_DEFERRED AS is_deferred,
          refc.RDB$UPDATE_RULE AS on_update,
          refc.RDB$DELETE_RULE AS on_delete,
          refc.RDB$MATCH_OPTION AS match_type,
          i2.RDB$RELATION_NAME AS references_table,
          s2.RDB$FIELD_NAME AS references_field,
          (s.RDB$FIELD_POSITION + 1) AS field_position
     FROM RDB$INDEX_SEGMENTS s
LEFT JOIN RDB$INDICES i ON i.RDB$INDEX_NAME = s.RDB$INDEX_NAME
```

```
LEFT JOIN RDB$RELATION_CONSTRAINTS rc ON rc.RDB$INDEX_NAME = s.RDB$INDEX_NAME
LEFT JOIN RDB$REF_CONSTRAINTS refc ON rc.RDB$CONSTRAINT_NAME =
refc.RDB$CONSTRAINT_NAME
LEFT JOIN RDB$RELATION_CONSTRAINTS rc2 ON rc2.RDB$CONSTRAINT_NAME =
refc.RDB$CONST_NAME_UQ
LEFT JOIN RDB$INDICES i2 ON i2.RDB$INDEX_NAME = rc2.RDB$INDEX_NAME
LEFT JOIN RDB$INDEX_SEGMENTS s2 ON i2.RDB$INDEX_NAME = s2.RDB$INDEX_NAME
    WHERE i.RDB$RELATION_NAME='b'       -- table name
      AND rc.RDB$CONSTRAINT_NAME='FK_b' -- constraint name
      AND rc.RDB$CONSTRAINT_TYPE IS NOT NULL
 ORDER BY s.RDB$FIELD_POSITION
```

Detailed TRIGGER info

```
SELECT RDB$TRIGGER_NAME AS trigger_name,

       RDB$RELATION_NAME AS table_name,
       RDB$TRIGGER_SOURCE AS trigger_body,
       CASE RDB$TRIGGER_TYPE
        WHEN 1 THEN 'BEFORE'
        WHEN 2 THEN 'AFTER'
        WHEN 3 THEN 'BEFORE'
        WHEN 4 THEN 'AFTER'
        WHEN 5 THEN 'BEFORE'
        WHEN 6 THEN 'AFTER'
       END AS trigger_type,
       CASE RDB$TRIGGER_TYPE
        WHEN 1 THEN 'INSERT'
        WHEN 2 THEN 'INSERT'
        WHEN 3 THEN 'UPDATE'
        WHEN 4 THEN 'UPDATE'
        WHEN 5 THEN 'DELETE'
        WHEN 6 THEN 'DELETE'
       END AS trigger_event,
       CASE RDB$TRIGGER_INACTIVE
        WHEN 1 THEN 0 ELSE 1
       END AS trigger_enabled,
       RDB$DESCRIPTION AS trigger_comment
  FROM RDB$TRIGGERS
 WHERE UPPER(RDB$TRIGGER_NAME)='AUTOINCREMENTPK'
```

Detailed VIEW info

If, given a VIEW name, you want to retrieve the field aliases and the field names in the original table, you can run this query:

```
SELECT d.RDB$DEPENDENT_NAME AS view_name,

         r.RDB$FIELD_NAME AS field_name,
         d.RDB$DEPENDED_ON_NAME AS depended_on_table,
         d.RDB$FIELD_NAME AS depended_on_field
    FROM RDB$DEPENDENCIES d
    LEFT JOIN RDB$RELATION_FIELDS r ON d.RDB$DEPENDENT_NAME =
r.RDB$RELATION_NAME
         AND d.RDB$FIELD_NAME = r.RDB$BASE_FIELD
   WHERE UPPER(d.RDB$DEPENDENT_NAME)='NUMBERSVIEW'
     AND r.RDB$SYSTEM_FLAG = 0
     AND d.RDB$DEPENDENT_TYPE = 1 --VIEW
ORDER BY r.RDB$FIELD_POSITION
```

If you want to get the VIEW definition, though, the above query is not enough. It will only show the matches between aliases and real fields. Use the following query to get the VIEW body:

```
SELECT RDB$VIEW_SOURCE

  FROM RDB$RELATIONS
 WHERE RDB$VIEW_SOURCE IS NOT NULL
   AND UPPER(RDB$RELATION_NAME) = 'NUMBERSVIEW';
```

One last tip

Since Firebird 2.1, you can query the database for some system information, like the engine version, the network protocol, etc. Have a look at the `doc/sql.extensions/README.context_variables2.txt` file for a listing of all the system variables that are available. Here's an example of what you can do:

```
SELECT RDB$GET_CONTEXT('SYSTEM', 'ENGINE_VERSION') AS engine_version,

       RDB$GET_CONTEXT('SYSTEM', 'NETWORK_PROTOCOL') AS protocol,
       RDB$GET_CONTEXT('SYSTEM', 'CLIENT_ADDRESS') AS address
  FROM RDB$DATABASE;
```

What else?

System tables are a powerful tool in the hands of the Interbase/Firebird db admins. The queries listed in this page are just the top of the iceberg, you can retrieve a lot more from these tables. If you'd like to see some other examples, or have some comments and/or suggestions, just drop me a mail (you can find my address in the footer of this page) and I'll add them to this list.
HTH.