
24-25 OCTOBER
PRAGUE
Firebird
Conference 2014
Firebird
Conference 2014

Tips for using system tables

Firebird Conference 2014

Ivan Přenosil

Thanks to sponsors

MOSCOW EXCHANGE Platinum

SITA SOFTWARE Platinum

IBPhoenix Platinum

IBSurgeon Platinum

REDSOFT

IBObjects

CopyCat

Upscene

Tables without PRIMARY KEY

```
SELECT RDB$RELATION_NAME AS "Table"
  FROM RDB$RELATIONS
 WHERE RDB$RELATION_TYPE IN (0, 4, 5)
       AND (RDB$SYSTEM_FLAG = 0 OR RDB$SYSTEM_FLAG IS NULL)
       AND RDB$RELATION_NAME NOT IN
   (SELECT RDB$RELATION_NAME FROM RDB$RELATION_CONSTRAINTS
    WHERE RDB$CONSTRAINT_TYPE = 'PRIMARY KEY')
 ORDER BY RDB$RELATION_NAME;
```

```
SELECT RDB$FIELD_NAME, RDB$TYPE, RDB$TYPE_NAME FROM RDB$TYPES WHERE RDB$FIELD_NAME = 'RDB$RELATION_TYPE';

RDB$FIELD_NAME                  RDB$TYPE RDB$TYPE_NAME
=============================== ======== ===============================
RDB$RELATION_TYPE                       0 PERSISTENT
RDB$RELATION_TYPE                       1 VIEW
RDB$RELATION_TYPE                       2 EXTERNAL
RDB$RELATION_TYPE                       3 VIRTUAL
RDB$RELATION_TYPE                       4 GLOBAL_TEMPORARY_PRESERVE
RDB$RELATION_TYPE                       5 GLOBAL_TEMPORARY_DELETE


SELECT RDB$FIELD_NAME, RDB$TYPE, RDB$TYPE_NAME FROM RDB$TYPES WHERE RDB$FIELD_NAME = 'RDB$SYSTEM_FLAG';

RDB$FIELD_NAME                  RDB$TYPE RDB$TYPE_NAME
=============================== ======== ===============================
RDB$SYSTEM_FLAG                         0 USER
RDB$SYSTEM_FLAG                         1 SYSTEM
```

Search system tables

```
SELECT *
  FROM RDB$RELATION_FIELDS rf
  JOIN RDB$FIELDS f ON f.RDB$FIELD_NAME=rf.RDB$FIELD_SOURCE
  WHERE (rf.RDB$NULL_FLAG = 1 OR f.RDB$NULL_FLAG = 1)
        AND rf.RDB$SYSTEM_FLAG = 1;
Records affected: 0

-> there are no NOT NULL constraints on system tables

-> (RDB$SYSTEM_FLAG = 0 OR RDB$SYSTEM_FLAG IS NULL)
-> COALESCE(RDB$SYSTEM_FLAG, 0) = 0
```

Example - after backup/restore, values in these columns changed 0 <--> NULL:

- RDB$INDICES.RDB$UNIQUE_FLAG
- RDB$INDICES.RDB$INDEX_TYPE
- RDB$FIELDS.RDB$FIELD_SCALE
- RDB$FIELDS.RDB$FIELD_SUB_TYPE
- RDB$INDICES.RDB$INDEX_INACTIVE
- RDB$PROCEDURES.RDB$PROCEDURE_OUTPUTS
- RDB$PROCEDURE_PARAMETERS.RDB$NULL_FLAG
- RDB$PROCEDURE_PARAMETERS.RDB$PARAMETER_MECHANISM
- RDB$USER_PRIVILEGES.RDB$GRANT_OPTION

Search system tables

```
SELECT CASE RDB$FIELD_TYPE WHEN 37 THEN 'VARCHAR'
  WHEN 14 THEN 'CHAR' END AS TYPE, COUNT(*)
  FROM RDB$RELATION_FIELDS rf
  JOIN RDB$FIELDS f ON f.RDB$FIELD_NAME=rf.RDB$FIELD_SOURCE
  WHERE RDB$FIELD_TYPE IN (37, 14)  -- 37=VARCHAR, 14=CHAR
        AND rf.RDB$SYSTEM_FLAG = 1
  GROUP BY RDB$FIELD_TYPE;
```

```
TYPE           COUNT
======= ===========
CHAR              80
VARCHAR           17
```

->

```
SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE RDB$RELATION_NAME LIKE '%RELATIONS';
Records affected: 0

SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE TRIM(RDB$RELATION_NAME) LIKE '%RELATIONS';
RDB$RELATION_NAME
==============================
RDB$RELATIONS
RDB$VIEW_RELATIONS
Records affected: 2
```

Tables without PRIMARY KEY 2

```
SELECT RDB$RELATION_NAME AS "Table",

IIF(NOT EXISTS(SELECT * FROM RDB$RELATION_CONSTRAINTS
  WHERE RDB$CONSTRAINT_TYPE = 'UNIQUE' AND RDB$RELATION_NAME=r.RDB$RELATION_NAME),
  '',
  IIF(EXISTS(
    SELECT *
    FROM RDB$RELATION_CONSTRAINTS rc
         JOIN RDB$INDEX_SEGMENTS ixs ON ixs.RDB$INDEX_NAME=rc. RDB$INDEX_NAME
         JOIN RDB$RELATION_FIELDS rf ON rf .RDB$FIELD_NAME=ixs.RDB$FIELD_NAME AND
                                        Rf .RDB$RELATION_NAME=r.RDB$RELATION_NAME
         JOIN RDB$FIELDS            f ON f  .RDB$FIELD_NAME=rf. RDB$FIELD_SOURCE
    WHERE rc.RDB$CONSTRAINT_TYPE='UNIQUE'
          AND rc.RDB$RELATION_NAME=r.RDB$RELATION_NAME
          AND (rf.RDB$NULL_FLAG=1 OR f.RDB$NULL_FLAG=1)
  ), 'Unique', 'Unique-nullable')) AS "Unique constraint"

  FROM RDB$RELATIONS r
 WHERE RDB$RELATION_TYPE IN (0, 4, 5)
       AND (RDB$SYSTEM_FLAG = 0 OR RDB$SYSTEM_FLAG IS NULL)
       AND RDB$RELATION_NAME NOT IN ('ACTION_LOG', 'ERROR_LOG')  -- tables intentionally without PK
       AND RDB$RELATION_NAME NOT IN (SELECT RDB$RELATION_NAME
             FROM RDB$RELATION_CONSTRAINTS WHERE RDB$CONSTRAINT_TYPE = 'PRIMARY KEY')
 ORDER BY RDB$RELATION_NAME;
```

Tables without PRIMARY KEY 2

example output:

```
Table                               Unique constraint
=================================== ==================
TABLE_GTT_D
TABLE_GTT_P
TABLE_UNIQUE_CONSTRAINT             Unique-nullable
TABLE_UNIQUE_CONSTRAINT2            Unique-nullable
TABLE_UNIQUE_CONSTRAINT3            Unique
TABLE_UNIQUE_CONSTRAINT_NN          Unique
TABLE_UNIQUE_IDX
TABLE_WITHOUT_PK
```

Tables without PRIMARY KEY 3

Firebird < 2.1 does not have **RDB$RELATION_TYPE** column.

```
SELECT RDB$RELATION_NAME AS "Table"
  FROM RDB$RELATIONS
 WHERE RDB$VIEW_BLR IS NULL
       AND RDB$EXTERNAL_FILE IS NULL
       AND (RDB$SYSTEM_FLAG = 0 OR RDB$SYSTEM_FLAG IS NULL)
       AND RDB$RELATION_NAME NOT IN
 (SELECT RDB$RELATION_NAME FROM RDB$RELATION_CONSTRAINTS
 WHERE RDB$CONSTRAINT_TYPE = 'PRIMARY KEY')
 ORDER BY RDB$RELATION_NAME;
```

Unique indexes not belonging to PRIMARY KEY or UNIQUE constraint.

```
SELECT
  RDB$INDEX_NAME    AS "Index",
  RDB$RELATION_NAME AS "Table"
 FROM RDB$INDICES
 WHERE RDB$UNIQUE_FLAG=1
       AND (RDB$SYSTEM_FLAG=0 OR RDB$SYSTEM_FLAG IS NULL)
       AND NOT EXISTS
    (SELECT *
       FROM RDB$RELATION_CONSTRAINTS
      WHERE RDB$INDEX_NAME=RDB$INDICES.RDB$INDEX_NAME
            AND RDB$CONSTRAINT_TYPE IN ('PRIMARY KEY', 'UNIQUE'))
 ORDER BY RDB$RELATION_NAME, RDB$INDEX_NAME;
```

Working with tables without primary/unique key

1) matching values for all fields, e.g. in IBExpert

```
SELECT COUNT(*) FROM MYTABLE
WHERE (ID = ?  ) AND (A = ?  );
UPDATE MYTABLE set A = ?
WHERE (ID = ?  ) AND (A = ?  );
```

2) using cursor in PSQL

```
syntax 1:
FOR SELECT ... FROM ... WHERE ... INTO  ... AS CURSOR C DO
  BEGIN DELETE FROM ... WHERE CURRENT OF C; ...

syntax 2:
DECLARE MyCursor CURSOR FOR (SELECT Id, A FROM MyTable);
  OPEN MyCursor;
  FETCH MyCursor INTO x, y;
  DELETE FROM MyTable WHERE CURRENT OF MyCursor;
  FETCH MyCursor INTO x, y;
  UPDATE MyTable SET ... WHERE CURRENT OF MyCursor;
  CLOSE MyCursor;
```

Working with tables without primary/unique key

3) using cursor from client application

```
SELECT * FROM MyTable FOR UPDATE;
function isc_dsql_set_cursor_name
    (var status_vector        : TISC_STATUS_VECTOR;
     var stmt_handle          : TISC_STMT_HANDLE;
     cursor_name              : FString;
     cursor_type              : uShort): TISC_STATUS;
    stdcall; external IBASE_DLL;
DELETE FROM MyTable WHERE CURRENT OF MyCursor;
```

Note: Cursors are unidirectional !

4) using DB_KEY

```
EXECUTE BLOCK AS
DECLARE DB_KEY CHAR(8) CHARACTER SET OCTETS;
BEGIN
  FOR SELECT RDB$DB_KEY, ... FROM TABLE_WITHOUT_PK INTO :DB_KEY
  DO  UPDATE TABLE_WITHOUT_PK SET ... WHERE RDB$DB_KEY = :DB_KEY;
END
```

DB_KEYS

- **SELECT first 4 RDB$DB_KEY FROM RDB$PROCEDURES;**

```
Output parameters:   DB_KEY  CHAR     (8)     OCTETS <OCTETS>
DB_KEY
================
0000001A00000004
0000001A00000008
0000001A0000000C
0000001A00000010
```

- DB_KEYs are not stable – they will not survive backup/restore, they will be reused after deleting&garbage collecting row, unless forbidden:

API: isc_attach_database(... isc_dpb_no_garbage_collect ...)
or isc_dpb_dbkey_scope (creates hidden transaction)
FSQL: CONNECT 'db.fdb' NO GARBAGE_COLLECT; or STABLE_DBKEY

- Although 64-bit value, it is retrived as VARCHAR/OCTETS ! i.e. this will not work

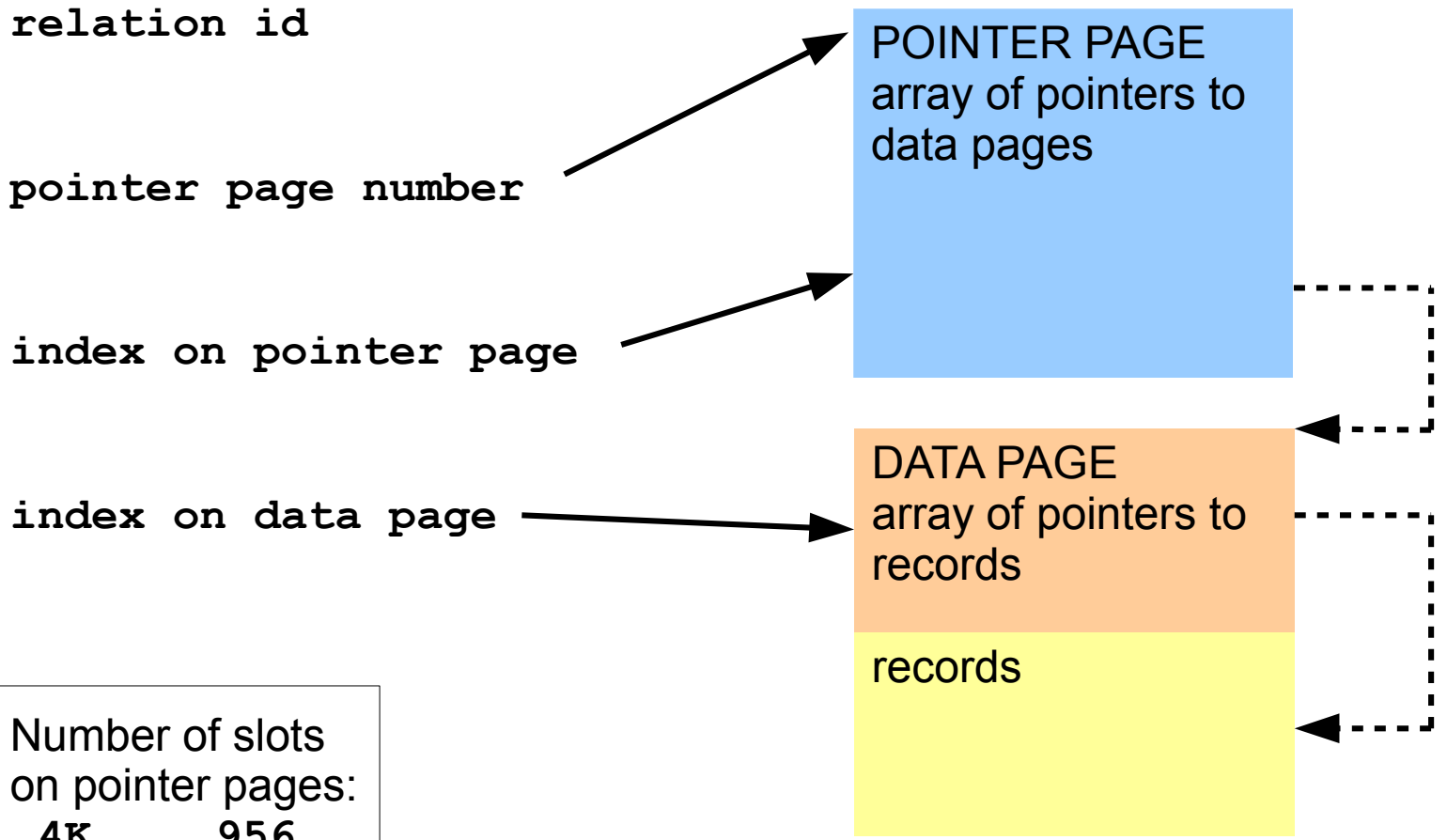
```
SELECT * FROM RDB$PROCEDURES WHERE 0x0000001A00000004 = RDB$DB_KEY
```

this will work

```
SELECT * FROM RDB$PROCEDURES WHERE RDB$DB_KEY = ASCII_CHAR(0x1A)||
ASCII_CHAR(0x00)||ASCII_CHAR(0x00)||ASCII_CHAR(0x00)||ASCII_CHAR(0x04)||
ASCII_CHAR(0x00)||ASCII_CHAR(0x00)||ASCII_CHAR(0x00)
```

- DB_KEY value is structured: 16 bit relation id, 32 + 8 bit record number

DB_KEYS structure



Number of slots on pointer pages:

4K	956
8K	1920
16K	3846

Pointer pages are listed in RDB$PAGES.

List of big tables

```
SELECT (SELECT rdb$relation_name FROM rdb$relations
        WHERE rdb$relation_id=X.rdb$relation_id) AS "Table",
  CAST(PPP * (Pages-1)+1 AS INTEGER) AS "Size From",
  CAST(PPP *  Pages AS INTEGER)      AS "Size To"
FROM (
    SELECT RDB$RELATION_ID, Count(*) AS Pages,
        (SELECT (MON$PAGE_SIZE - MON$PAGE_SIZE/1024*60-32)/4
         FROM MON$DATABASE) AS PPP
      FROM RDB$PAGES
      WHERE RDB$PAGE_TYPE=4 /* pointer page */
      GROUP BY RDB$RELATION_ID
      HAVING COUNT(*) > 10
    ) X ORDER BY 1;
```

Table	Size From	Size To
MAXPRICE	66921	67876
NORM	67877	68832
ORDER	40153	41108

Note: Pointer pages in RDB$PAGES are NOT freed !
(not even after sweep, only backup/restore)

Delimited identifiers

```
CREATE TABLE "mytab" (I INTEGER);  -- lowercase
CREATE TABLE "t a b" (I INTEGER);  -- spaces
CREATE TABLE "_TAB"  (I INTEGER);  -- starting with underscore
CREATE TABLE "$TAB"  (I INTEGER);  -- starting with dollar
CREATE TABLE "4TAB"  (I INTEGER);  -- starting with number
CREATE TABLE ":::::" (I INTEGER);  -- special characters
CREATE TABLE "ÁÁÁÁÁ" (I INTEGER);  -- national characters
CREATE TABLE "TABLE" (I INTEGER);  -- reserved word
CREATE TABLE TTTTT   ("i" INTEGER);
CREATE PROCEDURE "p" ("x" INTEGER) AS BEGIN END;
```

```
SELECT Object AS "Object", Name AS "Name", Tab AS "Table/Procedure"
  FROM (
SELECT 'Table:      ' AS Object, TRIM(TRAILING FROM RDB$RELATION_NAME) AS Name, '' AS Tab FROM RDB$RELATIONS   UNION ALL
SELECT 'Field:      ',TRIM(TRAILING FROM RDB$FIELD_NAME),      RDB$RELATION_NAME FROM RDB$RELATION_FIELDS      UNION ALL
SELECT 'Trigger:    ',TRIM(TRAILING FROM RDB$TRIGGER_NAME),    RDB$RELATION_NAME FROM RDB$TRIGGERS             UNION ALL
SELECT 'Index:      ',TRIM(TRAILING FROM RDB$INDEX_NAME),      RDB$RELATION_NAME FROM RDB$INDICES              UNION ALL
SELECT 'Constraint:',TRIM(TRAILING FROM RDB$CONSTRAINT_NAME), RDB$RELATION_NAME FROM RDB$RELATION_CONSTRAINTS UNION ALL
SELECT 'Constraint:',TRIM(TRAILING FROM RDB$CONSTRAINT_NAME), ''                FROM RDB$REF_CONSTRAINTS      UNION ALL
SELECT 'Constraint:',TRIM(TRAILING FROM RDB$CONST_NAME_UQ),   ''                FROM RDB$REF_CONSTRAINTS      UNION ALL
SELECT 'Procedure: ',TRIM(TRAILING FROM RDB$PROCEDURE_NAME),  ''                FROM RDB$PROCEDURES           UNION ALL
SELECT 'Proc.param:',TRIM(TRAILING FROM RDB$PARAMETER_NAME), RDB$PROCEDURE_NAME FROM RDB$PROCEDURE_PARAMETERS UNION ALL
SELECT 'Domain:     ',TRIM(TRAILING FROM RDB$FIELD_NAME),      ''                FROM RDB$FIELDS               UNION ALL
SELECT 'Generator:  ',TRIM(TRAILING FROM RDB$GENERATOR_NAME),  ''                FROM RDB$GENERATORS           UNION ALL
SELECT 'Exception:  ',TRIM(TRAILING FROM RDB$EXCEPTION_NAME),  ''                FROM RDB$EXCEPTIONS           UNION ALL
SELECT 'Function:   ',TRIM(TRAILING FROM RDB$FUNCTION_NAME),   ''                FROM RDB$FUNCTIONS            UNION ALL
SELECT 'Role:       ',TRIM(TRAILING FROM RDB$ROLE_NAME),       ''                FROM RDB$ROLES                UNION ALL
SELECT 'Char.set:   ',TRIM(TRAILING FROM RDB$CHARACTER_SET_NAME), ''             FROM RDB$CHARACTER_SETS       UNION ALL
SELECT 'Collation:  ',TRIM(TRAILING FROM RDB$COLLATION_NAME),  ''                FROM RDB$COLLATIONS
)
  WHERE Name NOT SIMILAR TO '[A-Z][A-Z0-9$_]*'
```

Note 1: regular expressions (SIMILAR TO) available since FB2.5
Note 2: does not find reserved words

Delimited identifiers 2

```
EXECUTE BLOCK RETURNS("Object" VARCHAR(12), "Name" VARCHAR(31) CHARACTER SET UTF8, "Table" VARCHAR(31) CHARACTER SET UTF8)
AS
DECLARE tmp INTEGER;
BEGIN
  FOR SELECT Object, Name, Tab
    FROM (
    SELECT 'Table:      ' AS Object, TRIM(TRAILING FROM RDB$RELATION_NAME) AS Name, '' AS Tab  FROM RDB$RELATIONS  UNION ALL
    SELECT 'Field:      ',TRIM(TRAILING FROM RDB$FIELD_NAME),      RDB$RELATION_NAME FROM RDB$RELATION_FIELDS      UNION ALL
    SELECT 'Trigger:    ',TRIM(TRAILING FROM RDB$TRIGGER_NAME),    RDB$RELATION_NAME FROM RDB$TRIGGERS             UNION ALL
    SELECT 'Index:      ',TRIM(TRAILING FROM RDB$INDEX_NAME),      RDB$RELATION_NAME FROM RDB$INDICES              UNION ALL
    SELECT 'Constraint:',TRIM(TRAILING FROM RDB$CONSTRAINT_NAME), RDB$RELATION_NAME FROM RDB$RELATION_CONSTRAINTS UNION ALL
    SELECT 'Constraint:',TRIM(TRAILING FROM RDB$CONSTRAINT_NAME), ''                FROM RDB$REF_CONSTRAINTS      UNION ALL
    SELECT 'Constraint:',TRIM(TRAILING FROM RDB$CONST_NAME_UQ),   ''                FROM RDB$REF_CONSTRAINTS      UNION ALL
    SELECT 'Procedure: ',TRIM(TRAILING FROM RDB$PROCEDURE_NAME),  ''                FROM RDB$PROCEDURES           UNION ALL
    SELECT 'Proc.param:',TRIM(TRAILING FROM RDB$PARAMETER_NAME), RDB$PROCEDURE_NAME FROM RDB$PROCEDURE_PARAMETERS UNION ALL
    SELECT 'Domain:     ',TRIM(TRAILING FROM RDB$FIELD_NAME),      ''                FROM RDB$FIELDS               UNION ALL
    SELECT 'Generator: ',TRIM(TRAILING FROM RDB$GENERATOR_NAME),  ''                FROM RDB$GENERATORS           UNION ALL
    SELECT 'Exception: ',TRIM(TRAILING FROM RDB$EXCEPTION_NAME),  ''                FROM RDB$EXCEPTIONS           UNION ALL
    SELECT 'Function:  ',TRIM(TRAILING FROM RDB$FUNCTION_NAME),   ''                FROM RDB$FUNCTIONS            UNION ALL
    SELECT 'Role:       ',TRIM(TRAILING FROM RDB$ROLE_NAME),      ''                FROM RDB$ROLES                UNION ALL
    SELECT 'Char.set:  ',TRIM(TRAILING FROM RDB$CHARACTER_SET_NAME), ''             FROM RDB$CHARACTER_SETS       UNION ALL
    SELECT 'Collation: ',TRIM(TRAILING FROM RDB$COLLATION_NAME),  ''                FROM RDB$COLLATIONS
  ) INTO :"Object", :"Name", :"Table"
  DO
    BEGIN
      IF ("Name" NOT SIMILAR TO '[A-Z][A-Z0-9$_]*') THEN
        SUSPEND;
      ELSE
        EXECUTE STATEMENT 'SELECT 1 AS ' || "Name" || ' FROM RDB$DATABASE' INTO :tmp;
    WHEN ANY DO
      SUSPEND;
    END
END
```

Table/view/procedure identifiers with length > 27

(bug in FB < 2.5)

```
CREATE TABLE    AAAAAAAAAABBBBBBBBBBCCCCCCCCCC1 (i integer);
CREATE TABLE    AAAAAAAAAABBBBBBBBBBCCCCCCCCCC2 (i integer);
GRANT SELECT ON AAAAAAAAAABBBBBBBBBBCCCCCCCCCC1 TO ALPHA;
GRANT SELECT ON AAAAAAAAAABBBBBBBBBBCCCCCCCCCC2 TO BETA;

SELECT RDB$RELATION_NAME, RDB$SECURITY_CLASS FROM rdb$relations WHERE rdb$system_flag=0;
FB2.1 or less:
RDB$RELATION_NAME               RDB$SECURITY_CLASS
=============================== ===============================
AAAAAAAAAABBBBBBBBBBCCCCCCCCCC1 SQL$AAAAAAAAAABBBBBBBBBBCCCCCCC
AAAAAAAAAABBBBBBBBBBCCCCCCCCCC2 SQL$AAAAAAAAAABBBBBBBBBBCCCCCCC
FB2.5:
RDB$RELATION_NAME               RDB$SECURITY_CLASS
=============================== ===============================
AAAAAAAAAABBBBBBBBBBCCCCCCCCCC1 SQL$4
AAAAAAAAAABBBBBBBBBBCCCCCCCCCC2 SQL$5
```

```
SELECT CASE WHEN RDB$VIEW_BLR IS NULL THEN 'Table' ELSE 'View' END AS "Object",
    RDB$RELATION_NAME AS "Name"
  FROM RDB$RELATIONS
  WHERE RDB$RELATION_NAME NOT LIKE '%    '
  -- WHERE SUBSTRING(RDB$RELATION_NAME FROM 28) <> ''
  -- WHERE CHAR_LENGTH(TRIM(TRAILING FROM RDB$RELATION_NAME)) > 27
  -- trim available since FB2.1
UNION ALL
SELECT 'Procedure' AS "Object", RDB$PROCEDURE_NAME AS "Name"
  FROM RDB$PROCEDURES
 WHERE RDB$PROCEDURE_NAME NOT LIKE '%    '
 ORDER BY 1,2;
```

Multifile database 1
secondary / delta / shadow

"all-in-one" example:

```
CREATE DATABASE 'D:\Database\tmpdb1.fdb' LENGTH 1000
          FILE 'E:\Database\tmpdb2.fdb' LENGTH 1000  -- secondary files
          FILE 'F:\Database\tmpdb3.fdb'
          DIFFERENCE FILE 'G:\tmpdb1.fdb.delta';  -- (ALTER DATABASE BEGIN BACKUP)
CREATE SHADOW 10 AUTO                'G:\tmpdb.shadow.10';
CREATE SHADOW 20 AUTO   CONDITIONAL 'G:\tmpdb.shadow.20';
CREATE SHADOW 30 MANUAL              'G:\tmpdb.shadow.30';
CREATE SHADOW 40 MANUAL CONDITIONAL 'G:\tmpdb.shadow.40'
                  LENGTH 1000 FILE 'G:\tmpdb.shadow.50';

SELECT
  IIF(RDB$FILE_FLAGS=0, 'secondary file',
    IIF(BIN_AND(RDB$FILE_FLAGS,  1)<>0, 'SHADOW ' || RDB$SHADOW_NUMBER || ' ' ||
        IIF(BIN_AND(RDB$FILE_FLAGS, 4)=0, 'AUTO ', 'MANUAL ') ||
        IIF(BIN_AND(RDB$FILE_FLAGS,16)=0, '', 'CONDITIONAL ') ||
        IIF(BIN_AND(RDB$FILE_FLAGS, 2)=0, '', '(INACTIVE)'),
      IIF(BIN_AND(RDB$FILE_FLAGS, 32)=0, '', 'DELTA ' ||
        IIF(BIN_AND(RDB$FILE_FLAGS, 64)=0, '(INACTIVE)', '(ACTIVE)')) )
  ) AS "File type",
  RDB$FILE_SEQUENCE AS "Seq.",
  RDB$FILE_NAME AS "File"
  FROM RDB$FILES
  ORDER BY RDB$SHADOW_NUMBER, RDB$FILE_SEQUENCE;
```

Multifile database 2
external tables

```
SELECT RDB$RELATION_NAME AS "Table", RDB$EXTERNAL_FILE AS "Filename"
  FROM RDB$RELATIONS
  WHERE RDB$RELATION_TYPE = 2  -- or WHERE RDB$EXTERNAL_FILE IS NOT NULL
  ORDER BY 1;
```

```
Table                           Filename
=============================== ========================
TABLE_EXTERNAL                  Table_external.txt
```

Multifile database 3
UDFs / blob filters

It is easy to crash FB server by "bad" external library.
There are two kinds - well known UDFs, and Blob Filters.

```
DECLARE EXTERNAL FUNCTION sin
  DOUBLE PRECISION
  RETURNS DOUBLE PRECISION BY VALUE
  ENTRY_POINT 'IB_UDF_sin' MODULE_NAME 'ib_udf';

DECLARE FILTER MyFilter
  INPUT_TYPE -10 OUTPUT_TYPE text
  ENTRY_POINT 'MyFilter' MODULE_NAME 'MyModule';
```

```
SELECT Type As "Type",
  CAST(IIF(CHAR_LENGTH(TRIM(RDB$MODULE_NAME))<=30, RDB$MODULE_NAME,
    LEFT(RDB$MODULE_NAME, 27) || '...') AS VARCHAR(30)) AS "File",
  CAST(LEFT(LIST(TRIM(RDB$FUNCTION_NAME), ', '), 100) AS VARCHAR(100)) AS "Functions"
  FROM (SELECT 'UDF' AS Type, RDB$MODULE_NAME, RDB$FUNCTION_NAME, RDB$SYSTEM_FLAG
          FROM RDB$FUNCTIONS
        UNION ALL
        SELECT 'Filter',      RDB$MODULE_NAME, RDB$FUNCTION_NAME, RDB$SYSTEM_FLAG
          FROM RDB$FILTERS)
  WHERE (RDB$SYSTEM_FLAG = 0) OR (RDB$SYSTEM_FLAG IS NULL)
  GROUP BY Type, RDB$MODULE_NAME;
```

```
Type   File                    Functions
====== ======================= ============
Filter MyModule                MYFILTER
UDF    ib_udf                  SIN, SQRT
```

Limits – format versions

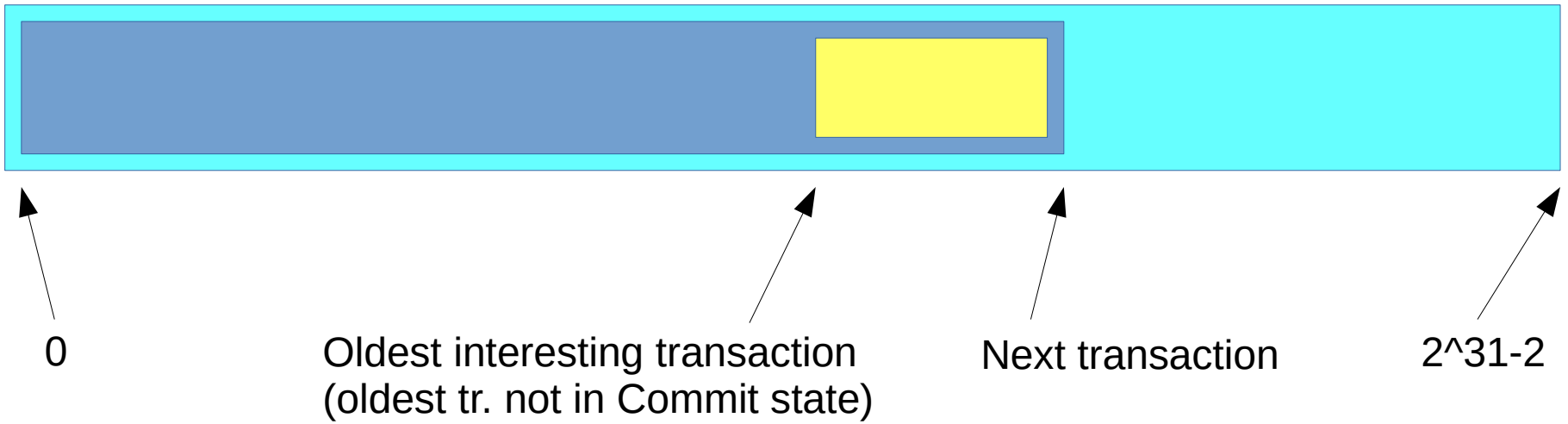
Do not confuse "record versions" (created by DML) (MGA)
with "format versions" (created by DDL).
Each record version contains 1 byte format version number (i.e. max 255).

```
SELECT R.RDB$RELATION_NAME AS "Table",
  MAX(F.RDB$FORMAT) AS "Formats"
  FROM RDB$FORMATS F JOIN RDB$RELATIONS R
       ON F.RDB$RELATION_ID=R.RDB$RELATION_ID
 WHERE (F.RDB$FORMAT >= 100)
 GROUP BY R.RDB$RELATION_NAME;
```

Table	Formats
PARTNER	170
GOODS	230

Limits - transactions

Integral part of MGA is transaction list, holding states of all transactions.
Whenever you start new snapshot transaction, FB server will remember current states of all transactions.
Whenever you insert/update row, new record version is created with transaction id in the header.



Transaction states:

- Active
- Committed
- Rollbacked
- Limbo

Limits - transactions

```
Statement failed, SQLCODE = -904
[335544381] Implementation limit exceeded
[335544864] -Transactions count exceeded. Perform backup and
restore to make database operable again
```

```
SELECT MON$NEXT_TRANSACTION AS "Transactions",
       0x7FFFFFFE - MON$NEXT_TRANSACTION AS "Transactions left"
  FROM MON$DATABASE
  WHERE MON$NEXT_TRANSACTION >= 1500000000;
```

```
Transactions    Transactions left
============ ====================
  2147483646                    0
```

If the "Next transaction" already reached the maximum,
it is necessary to set database to read-only state to be able to backup/restore it.

Limits - transactions

```
SELECT MON$OLDEST_TRANSACTION, MON$NEXT_TRANSACTION
  FROM MON$DATABASE;

MON$OLDEST_TRANSACTION MON$NEXT_TRANSACTION
====================== ====================
             725252504            725252540
```

Difference between Next Transaction and Oldest Transaction.

```
SELECT CAST(MON$NEXT_TRANSACTION - MON$OLDEST_TRANSACTION
       AS INTEGER) AS "Difference",
  IIF(MON$OLDEST_ACTIVE > MON$OLDEST_TRANSACTION+1,
      'Rollback/Limbo', 'Active') AS "Blocked by"
  FROM MON$DATABASE
  WHERE MON$NEXT_TRANSACTION - MON$OLDEST_TRANSACTION > 100;

 Difference Blocked by
========== ==============
        103 Active
```

DB with removed source text

Many fields in DB database are in pairs – original source text and compiled BLR code. e.g.

```
RDB$PROCEDURE_SOURCE & RDB$PROCEDURE_BLR
RDB$VIEW_SOURCE      & RDB$VIEW_BLR
```

```
CREATE DOMAIN DOMAIN_WITH_CHECK INTEGER CHECK (VALUE BETWEEN 1 AND 5);

UPDATE RDB$FIELDS SET RDB$VALIDATION_SOURCE = NULL
 WHERE RDB$FIELD_NAME = 'DOMAIN_WITH_CHECK';

UPDATE RDB$FIELDS SET RDB$VALIDATION_SOURCE = ''
 WHERE RDB$FIELD_NAME = 'DOMAIN_WITH_CHECK';


UPDATE RDB$PROCEDURES SET RDB$PROCEDURE_SOURCE = 'BEGIN EXIT; END'
 WHERE RDB$PROCEDURE_NAME = 'MY_PROCEDURE';
```

DB with removed source text

```
SELECT Object AS "Object", Name AS "Name", Tab AS "Table" FROM (
  SELECT 'Trigger' AS Object, RDB$TRIGGER_NAME AS Name, RDB$RELATION_NAME AS Tab, RDB$TRIGGER_SOURCE AS
Source
    FROM RDB$TRIGGERS  WHERE (RDB$SYSTEM_FLAG=0 OR RDB$SYSTEM_FLAG IS NULL)
  UNION ALL
  SELECT 'Field Default',  RDB$FIELD_NAME, RDB$RELATION_NAME, RDB$DEFAULT_SOURCE
    FROM RDB$RELATION_FIELDS  WHERE RDB$DEFAULT_VALUE IS NOT NULL
  UNION ALL
  SELECT 'Field Computed',  RF.RDB$FIELD_NAME, RF.RDB$RELATION_NAME, F.RDB$COMPUTED_SOURCE
    FROM RDB$RELATION_FIELDS RF JOIN RDB$FIELDS F ON RF.RDB$FIELD_SOURCE = F.RDB$FIELD_NAME
   WHERE F.RDB$COMPUTED_BLR IS NOT NULL AND RF.RDB$VIEW_CONTEXT IS NULL
  UNION ALL
  SELECT 'Field Check',  RDB$CONSTRAINT_NAME, RDB$RELATION_NAME, RDB$TRIGGER_SOURCE
    FROM RDB$CHECK_CONSTRAINTS JOIN RDB$TRIGGERS ON
         RDB$CHECK_CONSTRAINTS.RDB$TRIGGER_NAME=RDB$TRIGGERS.RDB$TRIGGER_NAME
   WHERE RDB$SYSTEM_FLAG=3
  UNION ALL
  SELECT 'View' AS Object, RDB$RELATION_NAME AS Name, '', RDB$VIEW_SOURCE AS Source
    FROM RDB$RELATIONS  WHERE RDB$VIEW_BLR IS NOT NULL
  UNION ALL
  SELECT 'Procedure', RDB$PROCEDURE_NAME, '', RDB$PROCEDURE_SOURCE
    FROM RDB$PROCEDURES
  UNION ALL
  SELECT 'Index', RDB$INDEX_NAME, '', RDB$EXPRESSION_SOURCE
    FROM RDB$INDICES  WHERE RDB$EXPRESSION_BLR IS NOT NULL
  UNION ALL
  SELECT 'Domain Default', RDB$FIELD_NAME, '', RDB$DEFAULT_SOURCE
    FROM RDB$FIELDS
   WHERE RDB$DEFAULT_VALUE IS NOT NULL AND (RDB$SYSTEM_FLAG=0 OR RDB$SYSTEM_FLAG IS NULL)
  UNION ALL
  SELECT 'Domain Check', RDB$FIELD_NAME, '', RDB$VALIDATION_SOURCE
    FROM RDB$FIELDS  WHERE RDB$VALIDATION_BLR IS NOT NULL
) WHERE (Source = '') OR (Source IS NULL);
```

Formatting of BLR - example

```
CREATE DOMAIN DOMAIN_WITH_CHECK INTEGER CHECK (VALUE BETWEEN 1 AND 5);

SELECT RDB$VALIDATION_SOURCE,                          -- SUB_TYPE 1, UNICODE_FSS
       RDB$VALIDATION_BLR,                             -- SUB_TYPE 2
       CAST(RDB$VALIDATION_BLR AS BLOB SUB_TYPE 1) AS SUB_TYPE_1,
       CAST(RDB$VALIDATION_BLR AS BLOB SUB_TYPE 0) AS SUB_TYPE_0
  FROM RDB$FIELDS
  WHERE RDB$FIELD_NAME = 'DOMAIN_WITH_CHECK';

RDB$VALIDATION_SOURCE RDB$VALIDATION_BLR        SUB_TYPE_1        SUB_TYPE_0
===================== ================== ================== ==================
                 0:27               2:205               0:24               2:205
================================================================================
RDB$VALIDATION_SOURCE:
CHECK (VALUE BETWEEN 1 AND 5)
================================================================================
RDB$VALIDATION_BLR:
blr_version5,
blr_between,
   blr_fid, 0, 0,0,
   blr_literal, blr_long, 0, 1,0,0,0,
   blr_literal, blr_long, 0, 5,0,0,0,
blr_eoc
================================================================================
SUB_TYPE_1:        (different formatting)
blr_version5,blr_between,    blr_fid, 0, 0,0,    blr_literal, blr_long, 0, 1,0,0,0,   blr_literal,
blr_long, 0, 5,0,0,0,blr_eoc
================================================================================
SUB_TYPE_0:
8..........L
================================================================================
```

Database as BLR compiler 1

```
CREATE OR ALTER PROCEDURE BLR AS
DECLARE TS TIMESTAMP;
BEGIN
  TS = CURRENT_TIMESTAMP;          -- Firebird variable
  TS = CURRENT_TIMESTAMP(3);       -- with precision
  TS = CAST('Now' AS TIMESTAMP);   -- explicit cast
  TS = 'Now';                      -- implicit cast
  TS = TIMESTAMP'Now';             -- date literal (typed string)
END

SELECT * FROM RDB$PROCEDURES WHERE RDB$PROCEDURE_NAME='BLR';
    blr_current_timestamp,

    blr_current_timestamp2, 3,

    blr_cast, blr_timestamp,
       blr_literal, blr_text2, 51,0, 3,0, 'N','o','w',

    blr_literal, blr_text2, 51,0, 3,0, 'N','o','w',

    blr_literal, blr_timestamp, 'd',-34,0,0,-118,31,-89,24,
```

Database as BLR compiler 2

Which one is better ?

```
SELECT ... FROM ... WHERE ID=1 OR ID=2 OR ID=3 OR ID=4;
SELECT ... FROM ... WHERE ID IN (1, 2, 3, 4);
```

```
CREATE OR ALTER PROCEDURE BLR2 AS DECLARE I INTEGER;
BEGIN  SELECT 1 FROM Table_PK WHERE ID=1 OR ID=2 OR ID=3 OR ID=4 INTO :I;  END

CREATE OR ALTER PROCEDURE BLR3 AS DECLARE I INTEGER;
BEGIN  SELECT 1 FROM Table_PK WHERE ID IN (1, 2, 3, 4) INTO :I;  END
```

```
SELECT * FROM RDB$PROCEDURES WHERE RDB$PROCEDURE_NAME IN ('BLR2', 'BLR3');
              blr_for,
                 blr_singular,
                    blr_rse, 1,
                       blr_relation, 8, 'T','A','B','L','E','_','P','K', 0,
                       blr_boolean,
                          blr_or,
                             blr_or,
                                blr_or,
                                   blr_eql,
                                      blr_field, 0, 2, 'I','D',
                                      blr_literal, blr_long, 0, 1,0,0,0,
                                   blr_eql,
                                      blr_field, 0, 2, 'I','D',
                                      blr_literal, blr_long, 0, 2,0,0,0,
                                blr_eql,
                                   blr_field, 0, 2, 'I','D',
                                   blr_literal, blr_long, 0, 3,0,0,0,
                             blr_eql,
                                blr_field, 0, 2, 'I','D',
                                blr_literal, blr_long, 0, 4,0,0,0,
                       blr_end,
```

BLR ... just curiosity

The Firebird‘s ancestor – InterBase – missed very useful and demanded function Substring().
But in its header file ibase.h there was deifned blr_substring. How to call it:

1)

```
CREATE PROCEDURE substr (str VARCHAR(20), start INTEGER, len INTEGER)
  RETURNS (ret VARCHAR(20)) AS
BEGIN
  ret = str / start / len;
END
```

2)

In **RDB$PROCEDURES.RDB$PROCEDURE_BLR** replace two **blr_divide** by one **blr_substring**

Note: SUBSTRING function was added at SQL level in Firebird 1.

Restoring corrupted database

Restoring corrupted database/backup may require

```
GBAK -C -INACTIVE ...
GBAK -C -NO_VALIDITY ...
```

-INACTIVE will set all indexes inactive, including PK, FK !!!

-NO_VALIDITY will remove CHECK constraints on all domains,
and will remove NOT NULL flags on all domains and all columns. .

- Primary keys with nullable fields
- Inactive indexes

```
SELECT rc.RDB$RELATION_NAME
FROM RDB$RELATION_CONSTRAINTS rc
     JOIN RDB$INDEX_SEGMENTS ixs ON ixs.RDB$INDEX_NAME=rc. RDB$INDEX_NAME
     JOIN RDB$RELATION_FIELDS rf ON rf .RDB$FIELD_NAME=ixs.RDB$FIELD_NAME AND
                                    rf .RDB$RELATION_NAME=rc.RDB$RELATION_NAME
     JOIN RDB$FIELDS             f ON f  .RDB$FIELD_NAME=rf. RDB$FIELD_SOURCE
WHERE rc.RDB$CONSTRAINT_TYPE='PRIMARY KEY'
      AND (COALESCE(rf.RDB$NULL_FLAG, 0)=0 AND COALESCE(f.RDB$NULL_FLAG, 0)=0);
```

```
SELECT RDB$INDEX_NAME AS "Index Name", RDB$RELATION_NAME AS "Table"
  FROM RDB$INDICES
 WHERE RDB$INDEX_INACTIVE = 1
 ORDER BY 2,1;
```

Inactive triggers

```
SELECT RDB$TRIGGER_NAME  AS "Trigger Name",
       RDB$RELATION_NAME AS "Table"
  FROM RDB$TRIGGERS
 WHERE RDB$TRIGGER_INACTIVE = 1
 ORDER BY 2,1;
```

Duplicate indexes

```
SELECT A.RDB$INDEX_NAME AS "Index 1", B.RDB$INDEX_NAME AS "Index 2"
  FROM RDB$INDICES A JOIN RDB$INDICES B ON
         A.RDB$RELATION_NAME=B.RDB$RELATION_NAME AND
         A.RDB$INDEX_NAME<B.RDB$INDEX_NAME
 WHERE A.RDB$SEGMENT_COUNT = B.RDB$SEGMENT_COUNT
   AND COALESCE(A.RDB$UNIQUE_FLAG, 0)=COALESCE(B.RDB$UNIQUE_FLAG, 0)
   AND COALESCE(A.RDB$INDEX_TYPE,  0)=COALESCE(B.RDB$INDEX_TYPE,  0)
   AND (A.RDB$EXPRESSION_BLR=B.RDB$EXPRESSION_BLR OR
        A.RDB$SEGMENT_COUNT = (SELECT COUNT(*)
          FROM RDB$INDEX_SEGMENTS AA JOIN RDB$INDEX_SEGMENTS BB
          ON AA.RDB$FIELD_POSITION=BB.RDB$FIELD_POSITION AND
             AA.RDB$FIELD_NAME=BB.RDB$FIELD_NAME AND
             AA.RDB$INDEX_NAME=A.RDB$INDEX_NAME AND
             BB.RDB$INDEX_NAME=B.RDB$INDEX_NAME));
```

Fields with redefined collation

```
CREATE DOMAIN D VARCHAR(5) CHARACTER SET WIN1250 COLLATE WIN_CZ;
CREATE TABLE  T(F D COLLATE PXW_CSY);


SELECT
  RF.RDB$RELATION_NAME AS "Table",
  RF.RDB$FIELD_NAME    AS "Field",
  RF.RDB$FIELD_SOURCE  AS "Domain",
  (SELECT RDB$COLLATION_NAME FROM RDB$COLLATIONS
     WHERE RDB$CHARACTER_SET_ID=F.RDB$CHARACTER_SET_ID AND
           RDB$COLLATION_ID=RF.RDB$COLLATION_ID) AS "Field collation",
  (SELECT RDB$COLLATION_NAME FROM RDB$COLLATIONS
     WHERE RDB$CHARACTER_SET_ID=F.RDB$CHARACTER_SET_ID AND
           RDB$COLLATION_ID=F.RDB$COLLATION_ID)  AS "Domain collation"

  FROM RDB$RELATION_FIELDS RF JOIN RDB$FIELDS F
       ON F.RDB$FIELD_NAME=RF.RDB$FIELD_SOURCE

  WHERE EXISTS(SELECT * FROM RDB$RELATIONS
                 WHERE RDB$RELATION_NAME=RF.RDB$RELATION_NAME AND
                       RDB$RELATION_TYPE IN (0))
        AND RF.RDB$COLLATION_ID>0 AND F.RDB$COLLATION_ID>0 AND
            RF.RDB$COLLATION_ID<>F.RDB$COLLATION_ID

  ORDER BY RF.RDB$RELATION_NAME, RF.RDB$FIELD_NAME;
```

Unused domains

```
SELECT RDB$FIELDS.RDB$FIELD_NAME AS "Domain"
  FROM RDB$FIELDS
 WHERE COALESCE(RDB$FIELDS.RDB$SYSTEM_FLAG, 0) = 0
       AND NOT (EXISTS(SELECT * FROM RDB$RELATION_FIELDS
                  WHERE RDB$FIELD_SOURCE = RDB$FIELDS.RDB$FIELD_NAME) OR
                EXISTS(SELECT * FROM RDB$PROCEDURE_PARAMETERS
                  WHERE RDB$FIELD_SOURCE = RDB$FIELDS.RDB$FIELD_NAME) )
 ORDER BY RDB$FIELD_NAME;
```

Generators with negative or bigint value

```
SET TERM ^;
EXECUTE BLOCK RETURNS("Generator" VARCHAR(31), "Value" BIGINT) AS
BEGIN
  FOR SELECT RDB$GENERATOR_NAME
      FROM RDB$GENERATORS
      ORDER BY RDB$GENERATOR_NAME
      INTO :"Generator"
  DO
    BEGIN
      EXECUTE STATEMENT 'SELECT GEN_ID("' || :"Generator" || '",0)
                         FROM RDB$DATABASE' INTO :"Value";
      IF ("Value" < 0 OR "Value" >= 2147483648) THEN
        SUSPEND;
    END
END^
SET TERM ;^
```

Generators - QUIZ

In FB3 run this, what will be the generator value at the end ?

```
SET HEAD OFF;
SET PLAN OFF;
SET AUTODDL OFF;
CREATE DATABASE 'testdb.fdb';

CREATE GENERATOR G;
COMMIT;
SELECT Gen_id(G, 0) FROM RDB$DATABASE;
                    0
SELECT Gen_id(G, 1) FROM RDB$DATABASE;
                    1
COMMIT;
SET GENERATOR G TO 2;
SELECT Gen_id(G, 0) FROM RDB$DATABASE;
                    2
SELECT Gen_id(G, 1) FROM RDB$DATABASE;
                    3
ROLLBACK;
SELECT Gen_id(G, 0) FROM RDB$DATABASE;
                   ???
```

The end

Questions ?

Thank you