APPL-R01104-R

Firebird clients and system tables

Björn Reimer, Dirk Baumeister

Who we are?

Björn Reimer

- Working as DBA at the Friedrich-Alexander-Universität Erlangen Nürnberg
- Independent software developer

Dirk Baumeister

- Working as Computer Scientist at the Language Centre of the Friedrich-Alexander-Universität Erlangen Nürnberg
- Independent software developer

motivation

- real 2-tier applications
  How are the duties distributed?
- A real db is more than a replacement for dBase files
- Access from Web and Windows
- simple clients
- easy administration

content

- infos out of db
  - data type of a field
- jobs for db
  - Handling permissions
    - Am I member of a role?
    - Am I allowed to do something?
    - Generate random pws
  - Handling primary keys
- The missing information
  - Who is logged in?
  - config settings for users
- Gadgets
  - Care of primary keys
  - Gbak bug

Tables in a db

```
SELECT RDB$RELATION_NAME,
 RDB$DESCRIPTION,
 RDB$FIELD_ID AS "ColCount"
FROM RDB$RELATIONS R
WHERE RDB$SYSTEM_FLAG = 0
 AND RDB$VIEW_BLR IS NULL
ORDER BY 1
```

Fields of a table

Fields of a table, e. g. of table 'ABC':

```
SELECT RDB$FIELD_NAME,
RDB$FIELD_SOURCE, RDB$DESCRIPTION
FROM RDB$RELATION_FIELDS
WHERE RDB$RELATION_NAME = 'ABC'
ORDER BY RDB$FIELD_POSITION;
```

Data type of a field

- Infos in Table RDB$FIELDS for each column and each domain:

```
SELECT RF.RDB$FIELD_NAME, RF.RDB$FIELD_SOURCE,
RF.RDB$DESCRIPTION,
F.RDB$FIELD_LENGTH, F.RDB$FIELD_SCALE,
F.RDB$FIELD_TYPE, F.RDB$FIELD_SUB_TYPE,
F.RDB$NULL_FLAG, F.RDB$DEFAULT_SOURCE,
F.RDB$VALIDATION_SOURCE,
F.RDB$COMPUTED_SOURCE,
RDB$CHARACTER_LENGTH
FROM RDB$RELATION_FIELDS RF
JOIN RDB$FIELDS F ON F.RDB$FIELD_NAME =
RF.RDB$FIELD_SOURCE
WHERE RF.RDB$RELATION_NAME = 'ABC'
ORDER BY RF.RDB$FIELD_POSITION;
```

F.RDB$FIELD_TYPE

- Type of a field:
  - 7: SMALLINT
  - 8: INTEGER
  - (9: QUAD)
  - 10: FLOAT
  - (11: D_FLOAT)
  - 12: DATE (dialect 3)
  - 13: TIME
  - 14: CHAR
  - 16: INT64
  - 27: DOUBLE
  - 35: TIMESTAMP (DATE in older versions)
  - 37: VARCHAR
  - (40: CSTRING)
  - 261: BLOB

F.RDB$FIELD_SUB_TYPE

- BLOBs:
  - 0: unspecified
  - 1: text
  - 2: BLR
  - 3: access control list

  >= 0: reserved for firebird

  < 0: user defined types

- NUMBERS (SMALLINT, INTEGER, INT64)
  - 0 or NULL: the field type
  - 1: numeric
  - 2: decimal

F.RDB$FIELD_LENGTH

- Field length important for size of index
  - CHAR, VARCHAR, NCHAR - maximum length of text in bytes
    Max. char length in RDB$CHARACTER_LENGTH
  - 4: FLOAT
  - (8: D_FLOAT)
  - 8: DOUBLE
  - 2: SHORT
  - 4: LONG
  - 8: QUAD
  - 8: INT64
  - 4: DATE
  - 4: TIME
  - 8: TIMESTAMP
  - 8: BLOB

Where to find what?

- Precision of numeric and decimal types:

```
F.RDB$FIELD_PRECISION
```

- Scale for numeric and decimal types:

```
F.RDB$FIELD_SCALE
```

Where to find what?

- Is a field NOT NULL:
  `F.RDB$NULL_FLAG`

Values:

  - 0 or NULL: don't care
  - 1: NOT NULL

- Default value
  `F.RDB$DEFAULT_SOURCE`
  with the word
  “DEFAULT” in front

Where to find what?

- Checks for fields
  `F.RDB$VALIDATION_SOURCE`
- COMPUTED BY
  `F.RDB$COMPUTED_SOURCE`

Where to find what?

- Character set
  `F.RDB$CHARACTER_SET_ID`
- Colation
  `F.RDB$COLLATION_ID`

The real info

- Some infos (same field names) are also in RDB$RELATION_FIELDS
  This infos are winning!

Permissions: Member of a role

- Which roles are valid?

```
SELECT RDB$ROLE_NAME FROM RDB$ROLES ORDER BY 1;
```

- Am I member of a role?

```
SELECT RDB$USER, RDB$RELATION_NAME
FROM  RDB$USER_PRIVILEGES
WHERE RDB$PRIVILEGE = 'M'
```

More information:
RDB$GRANTOR: Who has granted the role
RDB$GRANT_OPTION: 0=no; 2=yes

Permissions: Am I allowed?

- How to get permissions in a db:
  - Direct permission for user
  - Permission for role of an user
  - Permission for special user “public”
- Saved in

```
CREATE TABLE RDB$USER_PRIVILEGES (
    RDB$USER          CHAR(31) CHARACTER SET UNICODE_FSS,
    RDB$GRANTOR       CHAR(31) CHARACTER SET UNICODE_FSS,
    RDB$PRIVILEGE     CHAR(6) CHARACTER SET NONE,
    RDB$GRANT_OPTION   SMALLINT,
    RDB$RELATION_NAME  CHAR(31) CHARACTER SET UNICODE_FSS,
    RDB$FIELD_NAME     CHAR(31) CHARACTER SET UNICODE_FSS,
    RDB$USER_TYPE      SMALLINT,
    RDB$OBJECT_TYPE    SMALLINT);
```

Valid privileges

- M: Member (for Roles)
- S: Select
- I: Insert
- U: Update
- D: Delete
- A: Select AND Insert AND Update AND Delete
- E: Execute (Stored Procedures)

The procedure AmIAllowed (1)

```
CREATE PROCEDURE "PROC_AmIAllowed" (
   "Right" CHAR(1),
   "Relation" VARCHAR(62))
RETURNS (
   "Status" SMALLINT)
AS
DECLARE VARIABLE "IStatus" BIGINT;
begin
  "Status" =-1;
...
```

The procedure AmIAllowed (2)

```
SELECT count(RDB$RELATION_NAME) FROM
  RDB$USER_PRIVILEGES
  WHERE ((RDB$USER = current_user) or
  (RDB$USER = current_role) or
  (RDB$USER = 'PUBLIC')) AND
  RDB$RELATION_NAME = :"Relation" AND
  RDB$PRIVILEGE = UPPER(:"Right")
  INTO "IStatus";

  -- Check for A!

  if ("IStatus" >= 1) then "Status" = 1;

  else "Status" = 0;

  suspend;

end;
```

Performance: allowed for all procs

```
CREATE PROCEDURE "PRC_AmIAllowedProc" (
    "ProcNamePart" VARCHAR(31))
RETURNS (
    "Name" VARCHAR(84),
    "Status" SMALLINT)
AS
begin
  "ProcNamePart" = RTRIM(LTRIM("ProcNamePart"));
...
```

Performance: for all procs (2)

```
if ("ProcNamePart" <> '') then begin

   FOR SELECT distinct(P.Rdb$Relation_Name)
    FROM RDB$USER_PRIVILEGES  P
    WHERE P.Rdb$Relation_Name CONTAINING :"ProcNamePart"
    AND P.RDB$OBJECT_TYPE = 5  -- procedures
    INTO :"Name"

   DO BEGIN

     "Name"=RTRIM(LTRIM("Name"));

     SELECT "Status" FROM "PRC_AmIAllowed"('X', :"Name")
     INTO :"Status";

  suspend;

   END

 end else ...
```

Permissions: Generate PWs

```
CREATE PROCEDURE "PROCi_GetRandPassword"
   RETURNS (

"AutoPasswort" VARCHAR(20)

)
   AS

DECLARE VARIABLE "SollLang" INTEGER;
   DECLARE VARIABLE "Wert" INTEGER;
   DECLARE VARIABLE "Last" INTEGER;

begin

   "SollLang" = 20;

   "AutoPasswort" = '';

   "Last" = 0;

   "Wert" = 0;

   ...
```

Permissions: Generate Pws (2)

```
...
while (STRLEN("AutoPasswort") < "SollLang") do begin
    while ((not "Wert" between 33 and 122) or
     ("Wert" = "Last")) do begin
      "Wert" = RAND()*89+33;
    end
    "AutoPasswort" = "AutoPasswort" ||ASCII_CHAR("Wert");
    "Last" = "Wert";
  end
  suspend;
end
```

Handling primary keys

Via trigger

```
CREATE TRIGGER KM_4PERSON_KATEGORIE_BI0 FOR KM_4PERSON_KATEGORIE

ACTIVE BEFORE INSERT OR UPDATE POSITION 0

AS

BEGIN

  IF ((NEW."Id" IS NULL) OR (NEW."Id" <= 0)) THEN

    NEW."Id" = GEN_ID(GEN_KM_4PERSON_KATEGORIE_ID,1);

 ...
```

Trigger Part 2 – care about logging

```
 NEW."LastModifiedAt" = current_timestamp;
 NEW."LastModifiedFrom" = current_user;
 if (inserting) then begin
    NEW."CreatedAt" = current_timestamp;
    NEW."CreatedFrom" = current_user;
  end else if (updating) then begin
    NEW."CreatedAt" = OLD."CreatedAt";
    NEW."CreatedFrom" = OLD."CreatedFrom";
  end
END
```

Who is logged in?

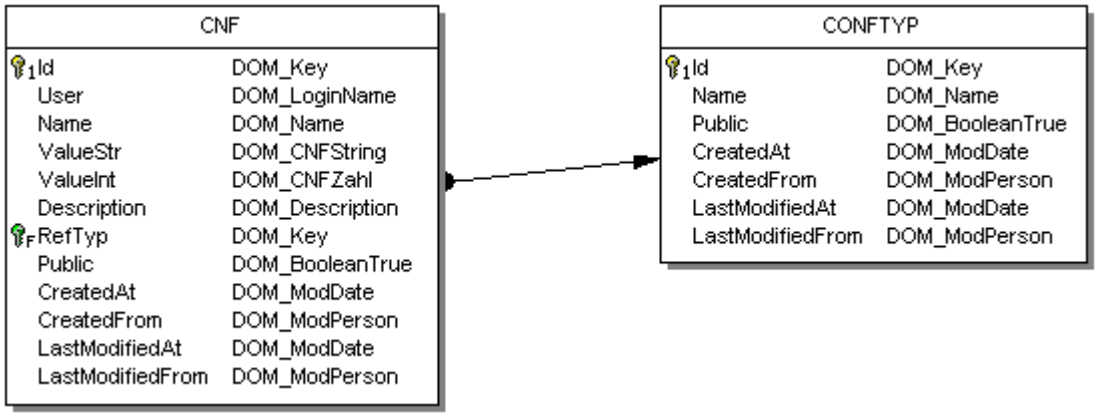
Sorry, no system table for that (not yet!)

Solution

- Insert record in table when logging in with at least user name and client machine
- Update that record, when logging off
- Regularly mark records of crashed clients

Config settings for a user

- Table CNF for config settings
- Table CONFTYP for grouping config settings



Access config settings

- API with five Stored Procedures
  - `PROC_CNFAddValue` for adding new values
  - `PROC_CNFGetInteger` for getting numbers
  - `PROC_CNFSetInteger` for setting numbers
  - `PROC_CNFGetString` for getting strings
  - `PROC_CNFSetString` for setting strings

Special user

- Username is login name
  ```
  SELECT current_user FROM RDB$DATABASE;
  ```
- User: <GLOBAL> for global settings
- When reading infos and nothing for current user is found, then search for settings of user <GLOBAL>
- No access for normal user to table CNF! Access only via SPs!

Care of primary keys

- Not for regularly usage!
- for use after cleaning the db or to produce a shipping version

- Problem: No meta data link between generator and table
- Solution: naming convention for generator,
  e. g.
  GEN_<table name>_ID

Care of primary keys (2)

- Guess matching names for PK
- Get max value for PK
- Get value of generator
- Compare and increase or decrease generator
- Combine all duties in a procedure

gbak-bug: too long identifier

- Identifier mustn't be longer than 27 chars. Otherwise granting may fail when restoring data :-(
- Registered as bug CORE-82. http://tracker.firebirdsql.org/browse/CORE-82
- Result of gbak:

```
...
gbak: restoring privilege for user PROC_KennungGetId
gbak: ERROR: action cancelled by trigger (0) to
      preserve data integrity
gbak: ERROR: could not find table/procedure for GRANT
gbak: Exiting before completion due to errors
```

Gbak-bug: avoid it

- Take care of too long role identifiers
- Take care of too long procedure names
- ...

It's not practicable in real live.

Better check your db regularly with a procedure.

Sources and references

- FirebirdRefGuide.pdf from ibphoenix
- UsingFirebird.pdf from ibphoenix
- Helens book
- Google :-)

The END

- Contact for questions and improvements:

  Björn Reimer (reimer@softbaer.de)

  Dirk Baumeister (baumeister@softbaer.de)