# The Firebird System Tables Exposed

## Session: APPL-R01102-S

Martijn Tonies

Upscene Productions
Database Tools for Developers
Database Workbench, LogManager Series, Advanced Data Generator,
InterXpress for Firebird
http://www.upscene.com

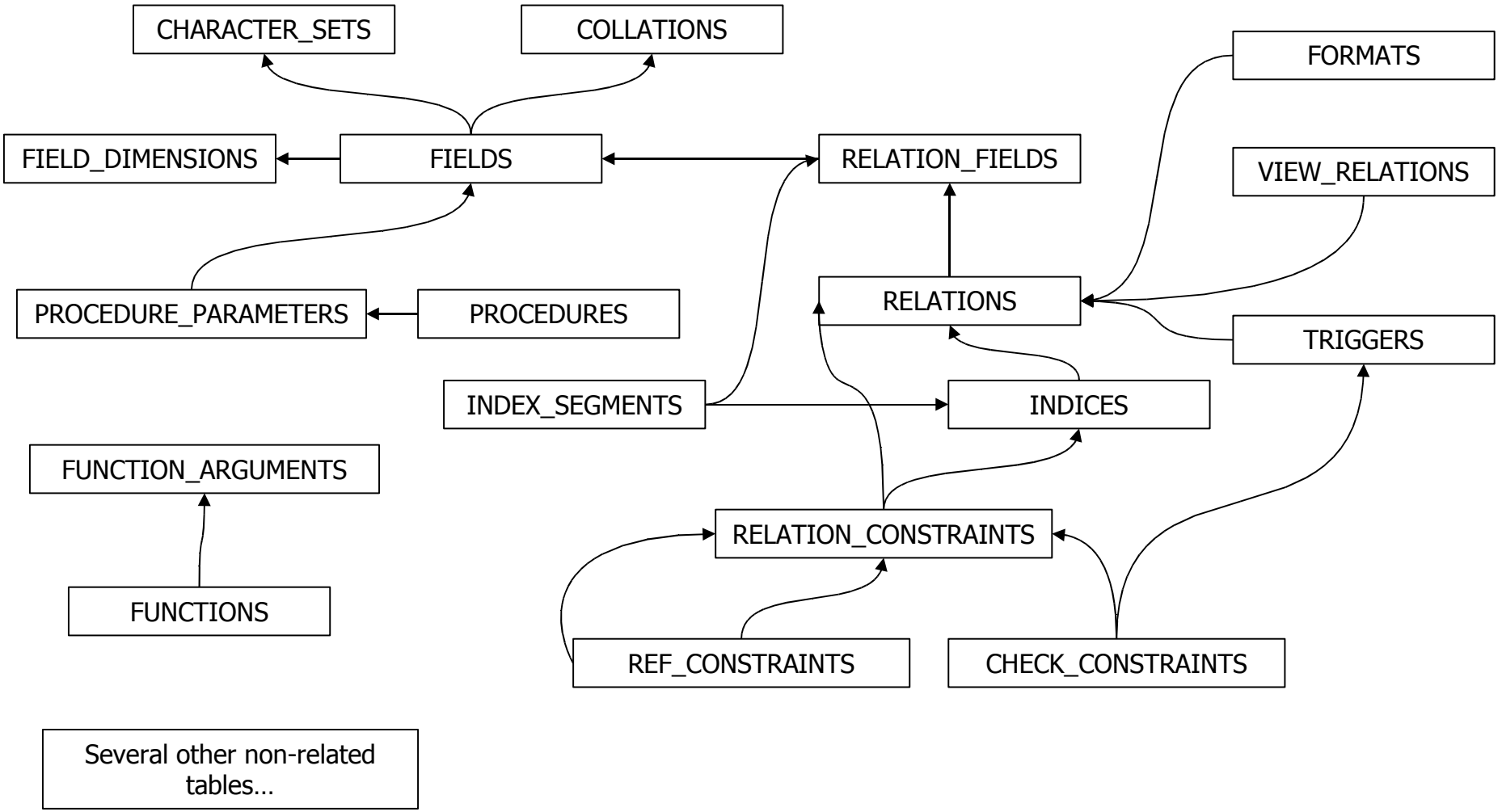
# Overview

- Introduction
- The Untouchables
- Single Object System Tables
- Relations & Domains
- Indices, Triggers, Constraints
- More System Tables: Procedures, External Functions
- Security
- Dependencies
- Protecting Your Source Code

# Introduction

- Where do the system tables come from?
  - Tables are ODS specific
  - Created when the database is created
- "active tables"
  - Changes to these table affect your database

# Schema


CHARACTER_SETS
COLLATIONS
FORMATS
FIELD_DIMENSIONS
FIELDS
RELATION_FIELDS
VIEW_RELATIONS
PROCEDURE_PARAMETERS
PROCEDURES
RELATIONS
TRIGGERS
INDEX_SEGMENTS
INDICES
FUNCTION_ARGUMENTS
RELATION_CONSTRAINTS
FUNCTIONS
REF_CONSTRAINTS
CHECK_CONSTRAINTS
Several other non-related tables…

# The Untouchables

- RDB$FILES

  holds 1 row for each additional or shadow file

- RDB$PAGES

  holds rows for each database page

- RDB$FORMATS

  holds a row for each relation

- RDB$TRANSACTIONS

  holds a row for each cross-database transaction

# Single Object Tables

- RDB$DATABASE
  - Single row table (like DUAL in Oracle)
  - Database wide (default) characterset
  - Has database description in RDB$DESCRIPTION
  - System generated relation ID#
  - Often used as a "dummy" table

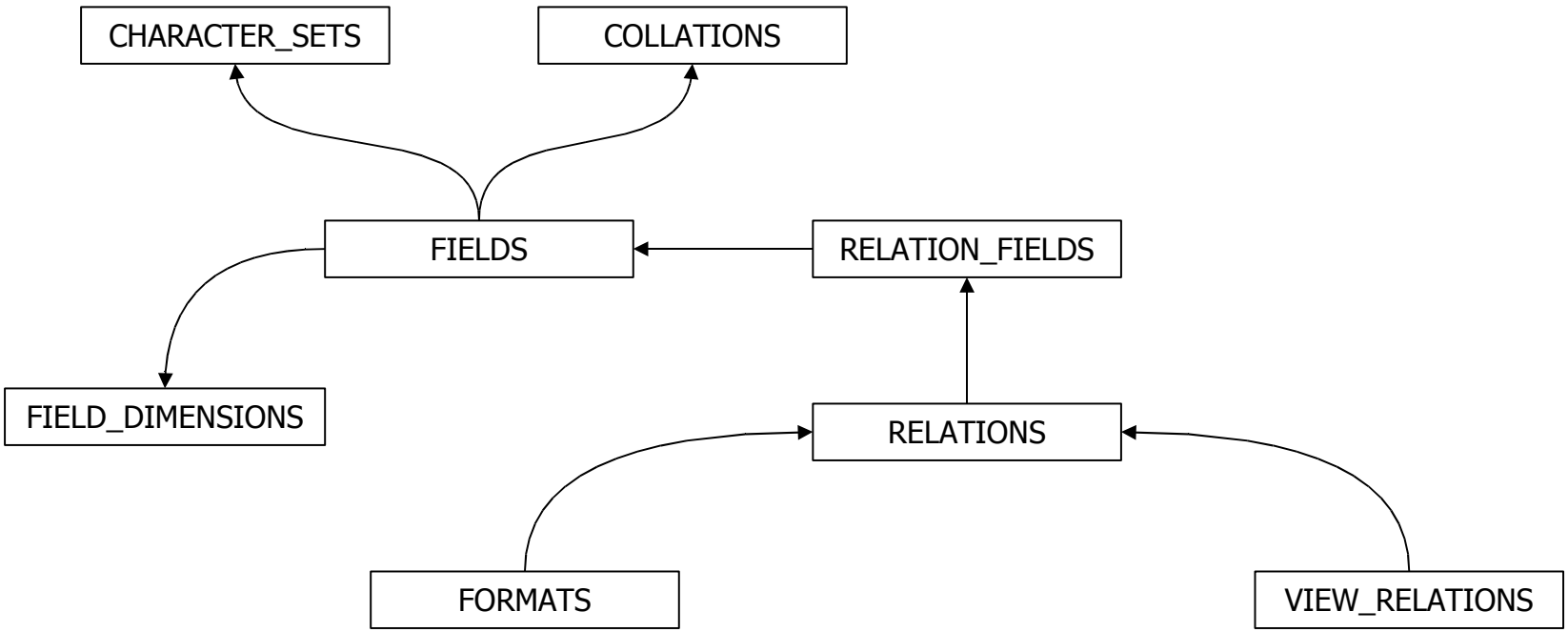
```
select gen_id(mygenerator, 1) as new_value
from rdb$database

select cast('test' as varchar(10))
from rdb$database
```

# Single Object Tables (continued)

- RDB$GENERATORS
  - One row per generator
  - Does not store generator values
- RDB$EXCEPTIONS
  - One row per user defined “exception” object
  - Stores a short message for the exception, larger in Firebird 2
- RDB$FILTERS
  - One row per “BLOB filter” object

# Relations & Domains



# Relations & Domains (continued)

- RDB$FIELDS – Domains
  - One row per domain
    - Datatype info, incl character set and collation
    - Optional check constraint (source & BLR)
    - Domain default
    - "nullable" flag (or "not null" flag)
    - RDB$DESCRIPTION for user written description for the domain
- RDB$FIELDS – Fields
  - One row per non-domain field
    - Datatype info, incl character set and collation
  - RDB$COMPUTED_SOURCE/BLR

# Relations & Domains (continued 2)

- Datatype information

| Datatype | FIELD_LENGTH | FIELD_SCALE | FIELD_PRECISION | CHARACTER_LENGTH |
|---|---|---|---|---|
| 7 (smallint) | 2 | ✓ | ✓ | |
| 8 (integer) | 4 | ✓ | ✓ | |
| 10 (float) | 4 | | | |
| 12 (date) | 4 | | | |
| 13 (time) | 4 | | | |
| 14 (char) | ✓ | | | ✓ |
| 16 (int64) | 8 | ✓ | ✓ | |
| 27 (double) | 8 | | | |
| 35 (timestamp) | 8 | | | |
| 37 (varchar) | ✓ | | | ✓ |
| 40 (cstring) | ✓ | | | ✓ |
| 261 (blob) | 8 | | | |

# Relations & Domains (continued 3)

- RDB$RELATION_FIELDS
  - Overrides domain default
  - Overrides domain “not null” if domain is nullable
  - Overrides domain collation
  - RDB$FIELD_SOURCE holds (user defined) domain name
  - RDB$FIELD_POSITION
  - RDB$DESCRIPTION for field description
  - For views (check also RDB$VIEW_RELATIONS):
    - RDB$BASE_FIELD holds the source column
    - RDB$VIEW_CONTEXT holds the source table for the column

# Relations & Domains (continued 4)

- RDB$FIELD_DIMENSIONS
  - Array dimensions
  - One row per dimension

# Relations & Domains (continued 5)

- RDB$RELATIONS
  - One row per table or view
  - RDB$FIELD_ID holds field number for new fields
  - Views have RDB$VIEW_BLR not NULL
  - RDB$DESCRIPTION for use written description
  - RDB$SYSTEM_FLAG = 1 tells us if it's a system defined table
  - RDB$EXTERNAL_FILE filled in for external tables
  - RDB$OWNER_NAME shows object owner
- RDB$VIEW_RELATIONS
  - One row for each table used in a view with alias

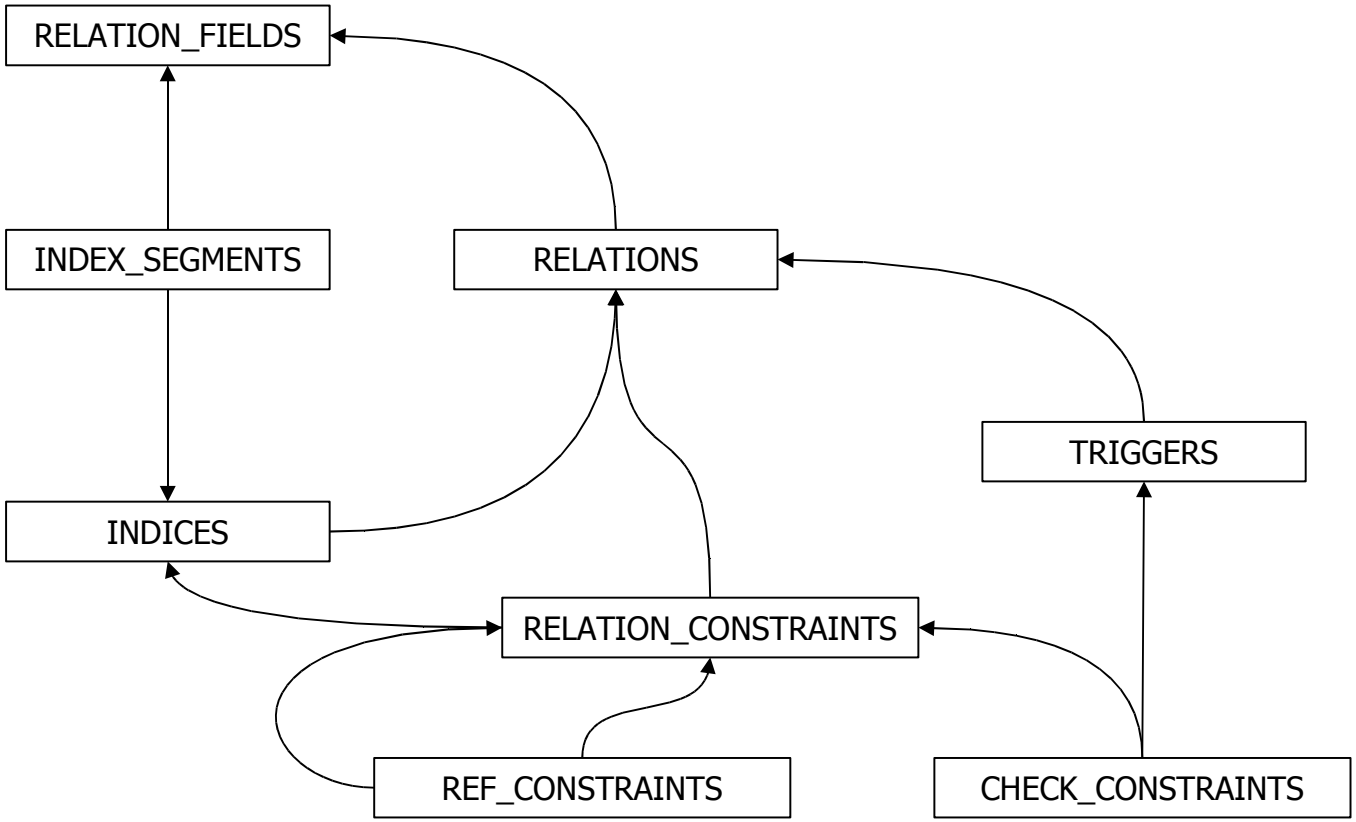
# Indices

- RDB$INDICES
  - One row per index, incl system defined indices
  - RDB$UNIQUE_FLAG
  - RDB$INDEX_TYPE (0 = asc, 1 = desc)
  - RDB$INDEX_INACTIVE
  - RDB$SYSTEM_FLAG
  - Index selectivity
- RDB$INDEX_SEGMENTS
  - One row per index per indexed field
  - RDB$FIELD_POSITION for sorting order
  - Firebird 2 adds selectivity per segment

# Triggers

- RDB$TRIGGERS
  - One row per trigger
  - RDB$TRIGGER_INACTIVE
  - RDB$TRIGGER_SOURCE/BLR
  - RDB$SYSTEM_FLAG
  - RDB$TRIGGER_TYPE
    - Least significant bit: before/after (0/1)
    - Two bits: insert/update/delete (01/10/11)
    - Repeat the above for Firebird 1.5 for each group of two bits

# Constraints



# Constraints (continued)

- RDB$RELATION_CONSTRAINTS
  - One row per constraint
  - RDB$CONSTRAINT_TYPE (5 types)
  - RDB$INDEX_NAME
- Unique & Primary Key Constraints
  - RDB$RELATION_CONSTRAINTS
  - RDB$INDEX_NAME
    - System defined index name
    - Implementing index in RDB$INDICES

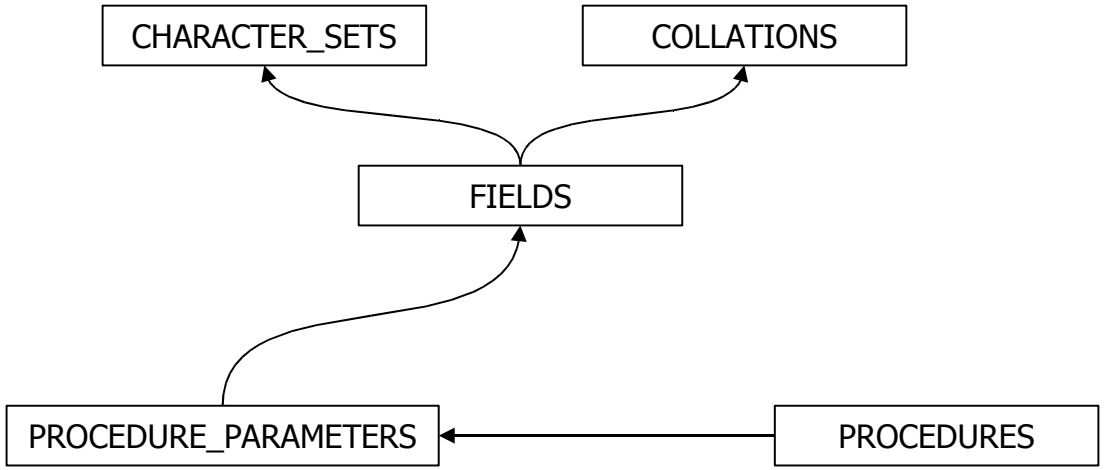
# Constraints (continued 2)

- RDB$CHECK_CONSTRAINTS
  - One row for each NOT NULL field
  - Two rows per user defined CHECK constraint
  - Zero, one or two rows for each FOREIGN KEY constraint
  - RDB$TRIGGER_NAME
    - NOT NULL field
    - Two triggers per CHECK constraint
    - Zero, one or two triggers per FOREIGN KEY constraint, depending on ON UPDATE and ON DELETE actions

# Constraints (continued 3)

- Foreign Key Constraints
  - RDB$RELATION_CONSTRAINTS
    - RDB$INDEX_NAME for auto-created index in “child” table
  - RDB$REF_CONSTRAINTS
    - RDB$CONST_NAME_UQ for “target” UNIQUE or PK
    - RDB$UPDATE_RULE & RDB$DELETE_RULE

# Stored Procedures



# Stored Procedures (continued)

- RDB$PROCEDURES
  - One row per procedure
  - RDB$PROCEDURE_SOURCE/BLR
  - RDB$PROCEDURE_INPUTS/OUTPUTS
  - RDB$DESCRIPTION for user written description
- RDB$PROCEDURE_PARAMETERS
  - One row for each procedure parameter
  - RDB$FIELD_SOURCE -> RDB$FIELDS (datatype info)
  - RDB$PARAMETER_NUMBER for sort order
  - RDB$PARAMETER_TYPE (0 = input, 1 = output)
  - RDB$DESCRIPTION

# External functions

- RDB$FUNCTIONS
  - One row per declared function
  - Function name, module, entry point
  - RDB$RETURN_ARGUMENT
  - RDB$DESCRIPTION
- RDB$FUNCTION_ARGUMENTS
  - One row per function parameter
  - RDB$MECHANISM ( FREE_IT adds * -1 modifier)
    - By value (0), by reference (1), by descriptor (2), by isc descriptor (3), by reference with null (5)
  - Datatype info like in RDB$FIELDS but in this table

# Security

- Security is a 3 step process:
  - Users (ISC4.GDB, SECURITY.FDB, SECURITY2.FDB), server wide
  - Roles (RDB$ROLES), in each database
  - Grants (RDB$USER_PRIVILEGES), in each database
- Grants in RDB$USER_PRIVILEGES
  - RDB$USER = grantee (User, Role, Procedure, Trigger, View)
  - RDB$RELATION_NAME = grantable (Role, Table, View, Procedure)
  - RDB$USER_TYPE, RDB$OBJECT_TYPE
  - GRANT ALL <> all grants

# Protecting your source code

- Schema information cannot be hidden
- Engine needs BLR to execute your code
- Source code (PSQL) can be removed (hidden)
  - Stored Procedures (RDB$PROCEDURES)
  - Triggers (RDB$TRIGGERS)
  - Check Constraints (RDB$TRIGGERS)
  - Views (RDB$RELATIONS)
  - Domain check constraints (RDB$FIELDS)
  - Computed By field (RDB$FIELDS)
- Keep a copy of your source code!!

# Questions?

# The Firebird System Tables Exposed

## Session: APPL-R01102-S

Martijn Tonies

Upscene Productions
Database Tools for Developers
Database Workbench, LogManager Series, Advanced Data Generator,
InterXpress for Firebird
http://www.upscene.com

# Overview

- Introduction
- The Untouchables
- Single Object System Tables
- Relations & Domains
- Indices, Triggers, Constraints
- More System Tables: Procedures, External Functions
- Security
- Dependencies
- Protecting Your Source Code

## Introduction

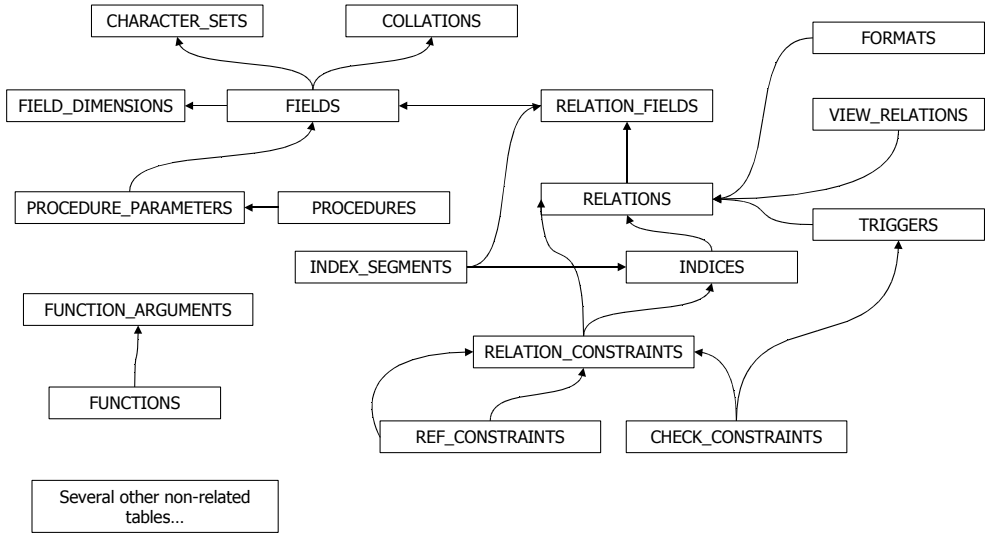
- Where do the system tables come from?
  - Tables are ODS specific
  - Created when the database is created
- "active tables"
  - Changes to these table affect your database


ODS = On Disk Stucture, the structure of the datafiles
Each version of Firebird has it's own ODS version.
Sometimes, older ODS versions can be used with a newer server version. A new ODS version usually introduces new features as well.

The system tables are Active Tables, whenever you change a system table, this affects the database. Some of these changes are not allowed, others are. Using standard DDL is the best way to modify meta data.

# Schema

Click to add an outline






An overview of available system tables. In a real database, all these are prefixed with RDB$...

## The Untouchables

- RDB$FILES

  holds 1 row for each additional or shadow file

- RDB$PAGES

  holds rows for each database page

- RDB$FORMATS

  holds a row for each relation

- RDB$TRANSACTIONS

  holds a row for each cross-database transaction



I named these tables “the Untouchables” - modifying these tables will result into almost instant database corruption.

All these can be read for analysis or administration. Most of the tools that come with Firebird will use these tables to some degree.

## Single Object Tables

- RDB$DATABASE
  - Single row table (like DUAL in Oracle)
  - Database wide (default) characterset
  - Has database description in RDB$DESCRIPTION
  - System generated relation ID#
  - Often used as a "dummy" table

```
select gen_id(mygenerator, 1) as new_value
from rdb$database

select cast('test' as varchar(10))
from rdb$database
```



The only things you should modify in this table is the description, you can change the default character set so that new char columns will take this character set instead. Existing columns will not be changed.

## Single Object Tables (continued)

- RDB$GENERATORS
  - One row per generator
  - Does not store generator values
- RDB$EXCEPTIONS
  - One row per user defined "exception" object
  - Stores a short message for the exception, larger in Firebird 2
- RDB$FILTERS
  - One row per "BLOB filter" object



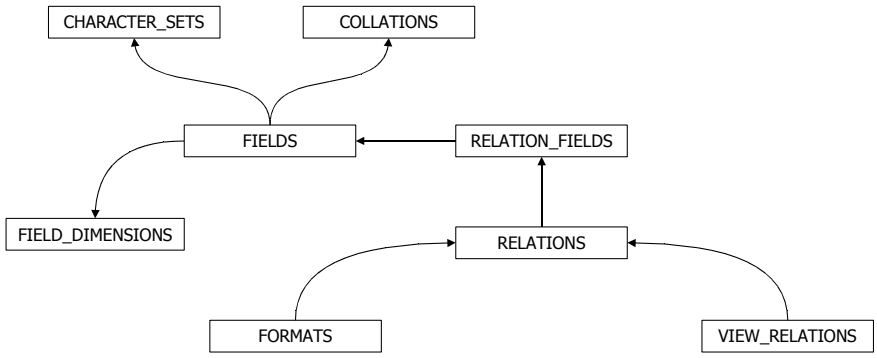
RDB$Generators holds names and ID, not the value. The actual values are stored on a special database page. RDB$SYSTEM_FLAG is 1 for system defined generators.

An "exception" message can be 78 characters long, Firebird 2 increases this limit. There's also a RDB$SYSTEM_FLAG column, but I've never seen any system exceptions.

A BLOB filter is a special type of external function, one that transforms a blob from one subtype to another. You can register blob filters and they will end up in the RDB$FILTER table.

## Relations & Domains

- Click to add an outline






Relations, which are Tables and Views, have a lot in common. Both are stored in RDB$RELATIONS. They share several tables as can be seen in this diagram.

## Relations & Domains (continued)

- RDB$FIELDS – Domains
  - One row per domain
    - Datatype info, incl character set and collation
    - Optional check constraint (source & BLR)
    - Domain default
    - "nullable" flag (or "not null" flag)
    - RDB$DESCRIPTION for user written description for the domain
- RDB$FIELDS – Fields
  - One row per non-domain field
    - Datatype info, incl character set and collation
  - RDB$COMPUTED_SOURCE/BLR



RDB$FIELDS holds two items: user generated Domains and auto-generated domains.

For each field not specifically assigned a domain, Firebird will auto-generate a domain in the RDB$FIELDS table.

# Relations & Domains (continued 2)

- Datatype information

| Datatype | FIELD_LENGTH | FIELD_SCALE | FIELD_PRECISION | CHARACTER_LENGTH |
|---|---|---|---|---|
| 7 (smallint) | 2 | ✓ | ✓ | |
| 8 (integer) | 4 | ✓ | ✓ | |
| 10 (float) | 4 | | | |
| 12 (date) | 4 | | | |
| 13 (time) | 4 | | | |
| 14 (char) | ✓ | | | ✓ |
| 16 (int64) | 8 | ✓ | ✓ | |
| 27 (double) | 8 | | | |
| 35 (timestamp) | 8 | | | |
| 37 (varchar) | ✓ | | | ✓ |
| 40 (cstring) | ✓ | | | ✓ |
| 261 (blob) | 8 | | | |



An overview of the available data types in Firebird and relevant columns. FIELD_LENGTH <> CHARACTER_LENGTH for multi-byte character sets. Always use CHARACTER_LENGTH to determine which value was used in the DDL statement.

## Relations & Domains (continued 3)

- RDB$RELATION_FIELDS
  - Overrides domain default
  - Overrides domain "not null" if domain is nullable
  - Overrides domain collation
  - RDB$FIELD_SOURCE holds (user defined) domain name
  - RDB$FIELD_POSITION
  - RDB$DESCRIPTION for field description
  - For views (check also RDB$VIEW_RELATIONS):
    - RDB$BASE_FIELD holds the source column
    - RDB$VIEW_CONTEXT holds the source table for the column



RDB$RELATION_FIELDS holds the default value if there's no domain, else RDB$FIELDS will hold the default. The default can be overridden, as several other attributes.

Views use an additional table RDB$VIEW_RELATIONS in which the tables can be found that are used by the view. Together with RDB$BASE_FIELD and RDB$VIEW_CONTEXT, you can find out where a specific view column is originating from. If a column is the result of no source or is unioned, RDB$BASE_FIELD is empty.

## Relations & Domains (continued 4)

- RDB$FIELD_DIMENSIONS
  - Array dimensions
  - One row per dimension



IMO, arrays should not be used because they are hardly supported in Firebird SQL, but – here's where you get them from the system tables :-)

## Relations & Domains (continued 5)

- RDB$RELATIONS
  - One row per table or view
  - RDB$FIELD_ID holds field number for new fields
  - Views have RDB$VIEW_BLR not NULL
  - RDB$DESCRIPTION for use written description
  - RDB$SYSTEM_FLAG = 1 tells us if it's a system defined table
  - RDB$EXTERNAL_FILE filled in for external tables
  - RDB$OWNER_NAME shows object owner
- RDB$VIEW_RELATIONS
  - One row for each table used in a view with alias



Besides RDB$VIEW_BLR, there's the RDB$VIEW_SOURCE column to holds the view definition.

Column RDB$FIELD_ID holds a value for incrementing the field ID for each field in the table.

RDB$OWNER_NAME shows who created the table and this is used for security as well. Only the database owner and SYSDBA can access all objects, other users can only access their “own” tables. This is not used as a prefix for tables or anything (like in MS SQL, for example).

# Indices

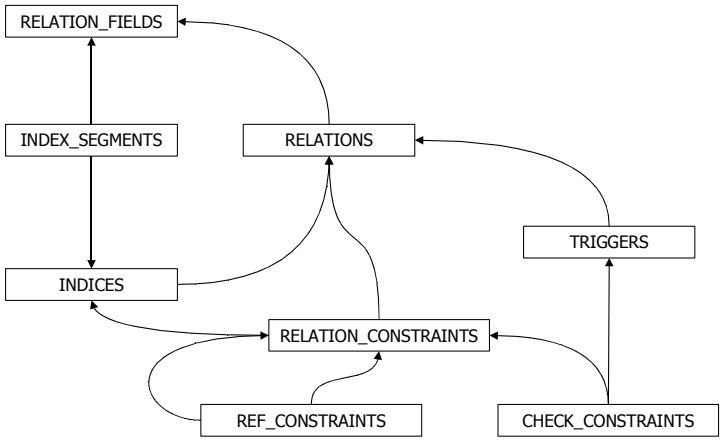
- RDB$INDICES
  - One row per index, incl system defined indices
  - RDB$UNIQUE_FLAG
  - RDB$INDEX_TYPE (0 = asc, 1 = desc)
  - RDB$INDEX_INACTIVE
  - RDB$SYSTEM_FLAG
  - Index selectivity
- RDB$INDEX_SEGMENTS
  - One row per index per indexed field
  - RDB$FIELD_POSITION for sorting order
  - Firebird 2 adds selectivity per segment

# Triggers

- RDB$TRIGGERS
  - One row per trigger
  - RDB$TRIGGER_INACTIVE
  - RDB$TRIGGER_SOURCE/BLR
  - RDB$SYSTEM_FLAG
  - RDB$TRIGGER_TYPE
    - Least significant bit: before/after (0/1)
    - Two bits: insert/update/delete (01/10/11)
    - Repeat the above for Firebird 1.5 for each group of two bits

## Constraints

- Click to add an outline






As you can see, constraints have a lot to do with triggers and indices. This is because Firebird PK/Unique and FK constraints are implemented via indices and Check constraints are implemented as triggers.

## Constraints (continued)

- RDB$RELATION_CONSTRAINTS
  - One row per constraint
  - RDB$CONSTRAINT_TYPE (5 types)
  - RDB$INDEX_NAME
- Unique & Primary Key Constraints
  - RDB$RELATION_CONSTRAINTS
  - RDB$INDEX_NAME
    - System defined index name
    - Implementing index in RDB$INDICES



There are 5 types of constraints: Primary Key, Unique, Foreign Key, Check and NOT NULL. The Not Null seems a bit weird, given the RDB$NULL_FLAG in the fields tables. If the constraint is implemented via an index, RDB$INDEX_NAME holds the system generated index. Firebird 1.5 uses the same name as the constraint.

## Constraints (continued 2)

- RDB$CHECK_CONSTRAINTS
  - One row for each NOT NULL field
  - Two rows per user defined CHECK constraint
  - Zero, one or two rows for each FOREIGN KEY constraint
  - RDB$TRIGGER_NAME
    - NOT NULL field
    - Two triggers per CHECK constraint
    - Zero, one or two triggers per FOREIGN KEY constraint, depending on ON UPDATE and ON DELETE actions



In the CHECK_CONSTRAINTS table, the not null constraints show up again. RDB$TRIGGER_NAME holds the column name – rather confusing.

For user defined check constraints, there's two rows and a system defined trigger name.

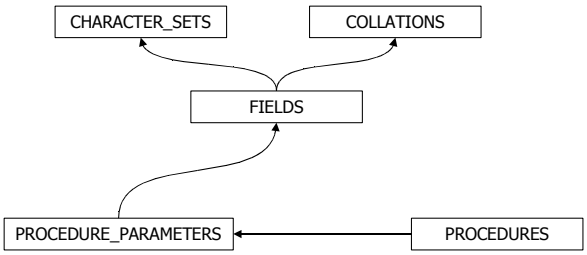
For foreign key constraints, there's one row in this table for any ON DELETE or ON UPDATE actions not equal to “no action”. Firebird will create a system trigger for these actions and store the name in here. So there could be zero, 1 or two triggers for each FK.

# Constraints (continued 3)

- Foreign Key Constraints
  - RDB$RELATION_CONSTRAINTS
    - RDB$INDEX_NAME for auto-created index in "child" table
  - RDB$REF_CONSTRAINTS
    - RDB$CONST_NAME_UQ for "target" UNIQUE or PK
    - RDB$UPDATE_RULE & RDB$DELETE_RULE

# Stored Procedures

- Click to add an outline

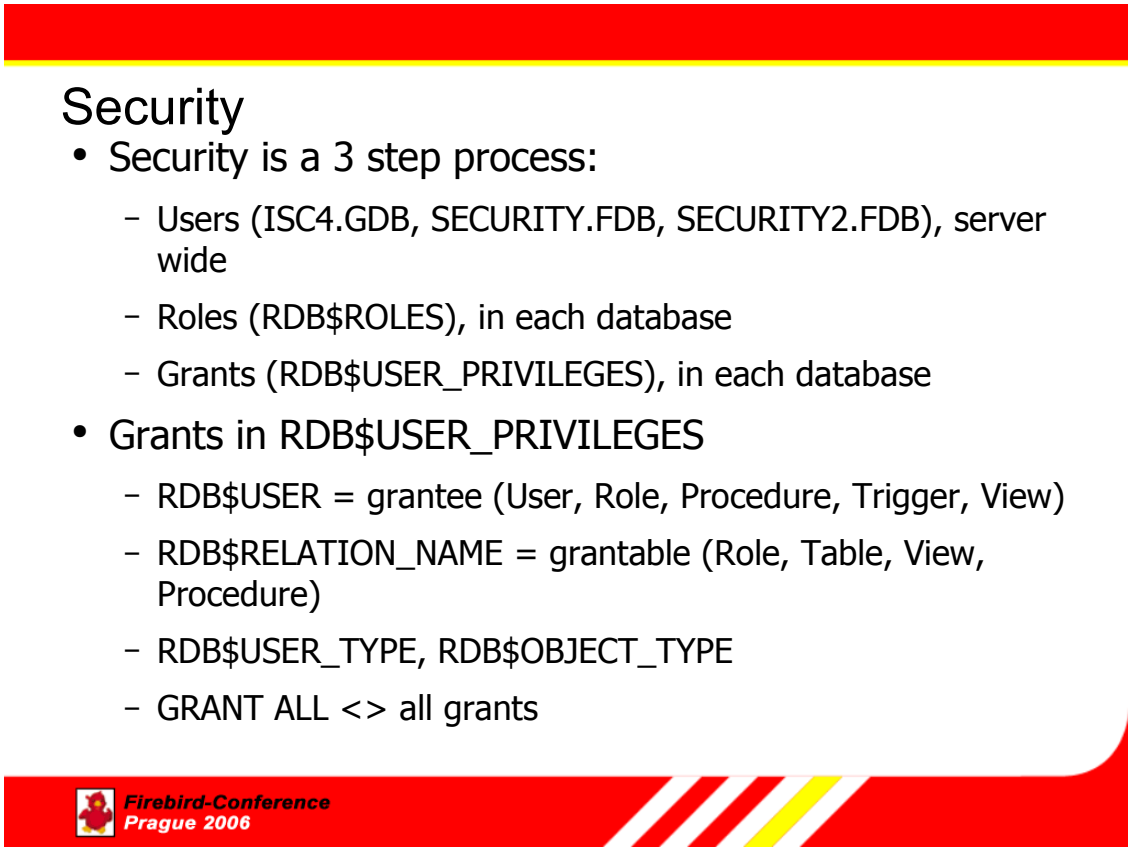


# Stored Procedures (continued)

- RDB$PROCEDURES
  - One row per procedure
  - RDB$PROCEDURE_SOURCE/BLR
  - RDB$PROCEDURE_INPUTS/OUTPUTS
  - RDB$DESCRIPTION for user written description
- RDB$PROCEDURE_PARAMETERS
  - One row for each procedure parameter
  - RDB$FIELD_SOURCE -> RDB$FIELDS (datatype info)
  - RDB$PARAMETER_NUMBER for sort order
  - RDB$PARAMETER_TYPE (0 = input, 1 = output)
  - RDB$DESCRIPTION

# External functions

- RDB$FUNCTIONS
  - One row per declared function
  - Function name, module, entry point
  - RDB$RETURN_ARGUMENT
  - RDB$DESCRIPTION
- RDB$FUNCTION_ARGUMENTS
  - One row per function parameter
  - RDB$MECHANISM ( FREE_IT adds * -1 modifier)
    - By value (0), by reference (1), by descriptor (2), by isc descriptor (3), by reference with null (5)
  - Datatype info like in RDB$FIELDS but in this table




Planning security is quite the complex task. You can grant views rights to tables, procedures rights to tables or views etc etc. All this can result in users not being able to update a table except via a procedure (see my other presentation). It helps to have a GUI tool to set your grants.

The “ALL” privileges is different from giving someone all privileges and is stored differently as well.

The RDB$GRANT_OPTION column tells you if the privilege has been granted with “with grant option”, meaning the user/role can forward this privilege to others.

Removing SYSDBA privileges does not remove access. SYSDBA can always access anything.

# Protecting your source code

- Schema information cannot be hidden
- Engine needs BLR to execute your code
- Source code (PSQL) can be removed (hidden)
  - Stored Procedures (RDB$PROCEDURES)
  - Triggers (RDB$TRIGGERS)
  - Check Constraints (RDB$TRIGGERS)
  - Views (RDB$RELATIONS)
  - Domain check constraints (RDB$FIELDS)
  - Computed By field (RDB$FIELDS)
- Keep a copy of your source code!!

# Questions?

- Click to add an outline