Stored Procedures, triggers and views – why and how

Session: HOWTO-OPL301-R

Martijn Tonies

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

Overview

- What are Stored Procedures (SPs) and Triggers?
- Why use SPs and Triggers?
- SP and Trigger structure
- What can be done by SPs and Triggers?
- New PSQL functionality in Firebird 2.0
- Extending the PSQL language, now and in the future
- Protecting your SP and Trigger source code
- What are views, why use views?
- Pitfalls
- A re-factoring example using SPs and Views

Introduction

- Anyone familiar with SPs and/or Triggers?
  - In Firebird? In other DBMSses?
  - Firebird SPs & Trigger language: PSQL
- Two-tier, multi-tier?
  - Where do SPs fit?

What are Stored Procedures & Triggers?

- SQL-like code (PSQL) stored and inside your database and executed by the DBMS
- SPs are executed by a client application or triggers
- Triggers are "triggered" by DML operations
- Each SP or Trigger can execute multiple statements

Why use SPs and Triggers?

- Security concerns
  - Don't allow direct access to tables
  - Hide tables as source for data
  - Allow 3rd parties access to certain procedures only
- Easier coding
  - The same code runs on all platforms for Firebird server
  - Simpler client code
  - Access the same functionality from different programming languages (any language that can access Firebird)

Why use SPs and Triggers? (continued)

- Add functionality without modifying applications
  - Functionality that should be available for all applications
  - Consistency across applications
    - Eg: auditing (like our IB LogManager product)
    - pseudo "auto-inc" fields:

```
BEGIN
    IF ( (NEW.ID IS NULL) OR (NEW.ID = 0) )
    THEN NEW.ID = GEN_ID(mygenerator, 1);
END
```

- Easier maintenance/refactoring
  - Ability to change database structure without modifying applications

Why use SPs and Triggers? (continued)

- Why use triggers?
  - More complex input validation
  - To ensure operations happen (eg: the autoincrement)
  - Ensure defaults (instead of relying on DEFAULT attribute)
  - Business rules enforcement
  - Consistency across applications

Stored Procedure and Trigger Structure

- A procedure has a header and a body
  - The header declares the parameters, both input and output
  - The body defines local variables and holds the statements to execute
  - A body can be a single statement or a compound statement (BEGIN..END) with multiple statements

```
CREATE PROCEDURE MyProcedureName (MyInput INTEGER)
    RETURNS (MyOutput VARCHAR(10) )
AS
BEGIN
    MyOutput = CAST(MyInput as VARCHAR(10));
END
```

Stored Procedure and Trigger Structure

- A trigger also has a header and body
  - The header defines on what table the trigger should be created, the timing and actions
  - The body is the same as with procedures

```
CREATE TRIGGER CUSTOMERS_ID FOR CUSTOMERS ACTIVE
  BEFORE INSERT
AS
begin
  if ( (new.CUSTID is null) or (new.CUSTID = 0) )
  then new.CUSTID = gen_id(CUSTOMERS_GEN, 1);
end
```

What can be done by SPs & Triggers?

- Short answer: A LOT.

What can be really done by SPs & Triggers?

- Data Manipulation Language (DML) operations
  - INSERT, DELETE, UPDATE

```
INSERT INTO mytable (col1, col2)
VALUES ('test', 4);
DELETE FROM mytable
WHERE ID < :myvariable;
```

  - Singleton SELECT (to fetch data values)

```
SELECT mycolumn, anothercolumn
FROM mytable
[WHERE ... ]
INTO :variable, :variable2;
```

What can be really done by SPs & Triggers?

- Dynamic SQL (EXECUTE STATEMENT)
  - Create a SQL statement by concatenating a string
  - Can also execute DDL, unlike normal PSQL
  - Can be used as a singleton select

```
EXECUTE STATEMENT 'delete from mytable';

sqlstr = 'delete from mytable ';
EXECUTE STATEMENT sqlstr || ' where mypk = 10';
```

What can be really done by SPs & Triggers?

- Cursor Loops
  - FOR SELECT ... DO [BEGIN ... END]
    - Cycle over a resultset, can update current row
  - DECLARE CURSOR .. OPEN, FETCH (Firebird 2)
    - Cycle over a resultset, fetch more than 1 row if needed via explicit "fetch" command
  - Dynamic SQL (FOR EXECUTE STATEMENT ... DO)
    - Same as "FOR SELECT", but with a dynamic statement
  - Often used for SELECT-able procedures

What can be really done by SPs & Triggers?

- Branching
  - IF (<expression>) THEN ... ; [ELSE ... ;]
  - Allows you to create complex logical statements
  - Example:

```
IF (date_column IS NULL)
THEN date_column = CURRENT_DATE;
ELSE IF (date_column > CURRENT_DATE)
THEN EXCEPTION date_error;
```

- Call procedures

```
EXECUTE PROCEDURE CUSTOMERS_I (custid, company_name,
    contact_firstname, contact_lastname, contact_email,
    contact_phone, 'T');
```

What can be really done by SPs & Triggers?

- Controlled exception handling
  - Catch database exceptions when you can handle them
  - Avoid needless exceptions being reported to the client
  - Add logic to retry your action (eg, in case of lock)
  - Exceptions can be re-raised in Firebird 1.5
- Create your own exceptions
  - Allows you to raise exceptions
  - Check exceptions in your client application
- 3 exception types (besides ANY):
  - GDSCODE, SQLCODE, user defined

What can be really done by SPs & Triggers?

- Events
  - A-synchronous server-to-client calls
  - Use EVENTs to signal the client
  - Client has to register interest in a specific named event
  - Examples:
    - Process data and continue the application, signal an event when ready
    - Allow client applications to refresh “browse” screens, avoid polling

What can be really done by SPs & Triggers?

- WHILE loop
  - Logical loop, while <expression> = true, do loop
  - DO <statement> or <compound statement>
- LEAVE <label> to exit a loop (Firebird 2)
- EXIT to exit the procedure/trigger

What can be really done by SPs & Triggers?

- Virtual Tables/SELECT-able Procedures
  - Keyword: SUSPEND
  - Can return 1 result set of the specified structure
  - Output parameters act as "columns" in the result set
  - Can have input parameters

```
CREATE PROCEDURE WHILE_EXAMPLE returns (POUT_VALUE Integer)
AS
DECLARE VARIABLE i integer;
begin
  i = 0;
  while (i < 10)
  do begin
       pout_value = i;
       i = i + 1;
       suspend;
     end
end
```

What can be really done by SPs & Triggers?

- Triggers are different
  - No SUSPEND possible
  - NEW and OLD context variables
  - For multi-action triggers, check which action if firing

```
IF (DELETING) THEN ...
```

  - Raising exceptions will cancel insert/delete/update

What can be really done by SPs & Triggers?

- PSQL (Procedural SQL)
  - Easy-to-learn
  - Powerful
  - Keep your code modular and let procedures call other procedures
- Debugging can be hard
  - Firebird does not provide a debugging interface
  - Several developer tools emulate Firebirds behavior when debugging procedures

Extending the PSQL language

- External Functions
  - Also called “user defined functions”
  - In an external dynamic loadable module (dll/so)
  - Can accept up to 10 parameters
  - Must be declared on a per-database basis
  - Do NOT have a database context (so no transaction)
  - Should not call back into the database
  - Several free ware libraries are available on the internet
  - External libraries have to be compiled for the server platform

New PSQL functionality in Firebird 2.0

- Sequence support according to SQL 99
  - NEXT VALUE FOR <sequence name>
- INSERT INTO ... RETURNING clause
- ROW_COUNT for SELECT statements
- RDB$SET_CONTEXT & RDB$GET_CONTEXT

Extending the PSQL language

- Future ideas and implementations
  - Using Java in Stored Procedures
  - Stored Functions
  - Using .NET stuff
  - Fyracle
  -

Protecting your source code

- System tables have stored source code
- Stored source code is for reference only
- Server uses BLR to execute your code
- Clearing the source code hides your source
  - RDB$PROCEDURES.RDB$PROCEDURE_SOURCE
  - RDB$TRIGGERS.RDB$TRIGGER_SOURCE
  - RDB$FIELDS.RDB$COMPUTED_SOURCE
  - RDB$RELATIONS.RDB$VIEW_SOURCE

What are views?

- "Relations" are Tables and Views
  - Table is a physical relation
  - View is a logical relation
  - Inserts and updates are possible, automatically or by using triggers
- SELECT

Why use views?

- Security concerns
  - Don't allow direct access to tables
  - Hide tables as source for data
  - Allow 3rd parties access to certain views only
- Easier coding
  - Simpler client code
  - Modify the underlying table structure without affecting applications

Pitfalls when using Stored Procedures

- Don't forget SUSPEND for SELECT-able SPs
- When not to use a SELECT-able SP
- GRANTs

Examples

- Procedure to insert or update data
- Business rules enforcement by using triggers/checks
  - No more changes to an invoice once approved
  - Make sure an invoiced order belongs to the
- Data processing on the server
  - Change all product prices according to given percentage
- Re-factoring example
  - From 1 contact per customer, to multiple contacts

Questions?