Database trigger

Database triggers were implemented in Firebird 2.1. These are user-defined PSQL modules that can be defined to fire in various connection-level and transaction-level events. This allows you to, for example, set up a protocol relatively quickly and easily.

Database trigger types

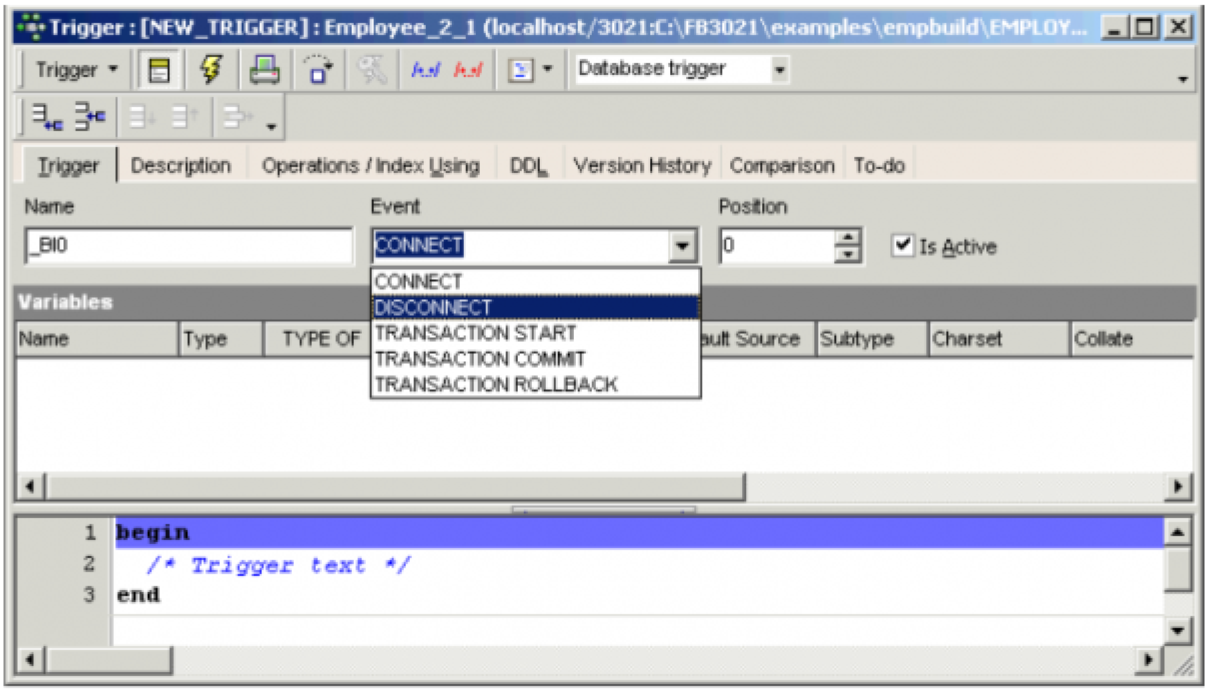
Database-wide triggers can be fired on the following database trigger types:

CONNECT	The database connection is established, a transaction begins, triggers are fired - uncaught exceptions rollback the transaction, disconnect the attachment and are returned to the client. Finally the transaction is committed.
DISCONNECT	A transaction is started, triggers are fired - uncaught exceptions rollback the transaction, disconnect the attachment and are stopped. The transaction is committed and the attachment disconnected.
TRANSACTION START	Triggers are fired in the newly-created user transaction - uncaught exceptions are returned to the client and the transaction is rolled back.
TRANSACTION COMMIT	Triggers are fired in the committing transaction - uncaught exceptions rollback the trigger's savepoint, the commit command is aborted and an exception is returned to the client. For two-phase transactions the triggers are fired in PREPARE and not in COMMIT.
TRANSACTION ROLLBACK	Triggers are fired in the rolling-back transaction - changes made will be rolled back together with the transaction, and exceptions are stopped.

Only the SYSDBA or the database owner can:

- define database triggers
- switch them of for a new connection by:
    - new isc_dpb_no_db_triggers tag
    - new -no_dbtriggers switch in utilities

In IBExpert database triggers can be created, edited and deleted in the same way as table-bound triggers (see New trigger for details). Simply switch to Database trigger in the toolbar, to access the options specific to database triggers:



Name: _BI0
Event: CONNECT
Position: 0
☑ Is Active

```
begin
  /* Trigger text */
end
```

Specify who is allowed to access your application, or raise an exception when certain unwanted applications attempt to access your database. Database triggers are also a really nice feature for protocols, enabling you for example to create your own login mapping with IP addresses an so on.

An example of a database trigger (source Firebird 2.1 What's New, by Vladyslav Khorsum):

Example of an ON CONNECT trigger

```
isql temp.fdb -user SYSDBA -pass masterkey
Database: temp.fdb, User: SYSDBA
SQL> SET TERM ^ ;
SQL> CREATE EXCEPTION EX_CONNECT 'Forbidden !' ^
SQL> CREATE OR ALTER TRIGGER TRG_CONN ON CONNECT
CON> AS
CON> BEGIN
CON> IF (<bad user>)
CON> THEN EXCEPTION EX_CONNECT USER || ' not allowed !';
CON> END ^
SQL> EXIT ^

isql temp.fdb -user BAD_USER -pass ...
Statement failed, SQLCODE = -836
exception 217
-EX_CONNECT
-BAD_USER not allowed !
-At trigger 'TRG_CONN' line: 5, col: 3
Use CONNECT or CREATE DATABASE to specify a database
SQL> EXIT;
```

If you encounter problems with an ON CONNECT trigger, so that noone can connect to the database any more, use the -no_dbtriggers switch in the utilities:

```
isql temp.fdb -user SYSDBA -pass masterkey
-nodbtriggers Database: temp.fdb, User: SYSDBA
SQL> ALTER TRIGGER TRG_CONN INACTIVE;
SQL> EXIT;
```

Database triggers can be quickly and easily defined in IBExpert's Trigger Editor (see below).