Table triggers

Table trigger types

Trigger types refer to the trigger status (ACTIVE or INACTIVE), the trigger position (BEFORE or AFTER) and the operation type (INSERT, UPDATE or DELETE).

They are specified following the definition of the table or view name, and before the trigger body.

ACTIVE or INACTIVE

ACTIVE or INACTIVE is specified at the time a trigger is created. ACTIVE is the default if neither of these keywords is specified. An inactive trigger does not execute.

BEFORE or AFTER

A trigger needs to be defined to fire either BEFORE or AFTER an operation. A BEFORE INSERT trigger fires before a new row is actually inserted into the table; an AFTER INSERT trigger fires after the row has been inserted.

BEFORE triggers are generally used for two purposes:

1. They can be used to determine whether the operation should proceed, i.e. certain parameters can be tested to determine whether the row should be inserted, updated or deleted or not. If not, an exception can be raised and the transaction rolled back.
2. BEFORE triggers can also be used to determine whether there are linked rows that might be affected by the operation. For example, a trigger might be used to automatically reassign sales before deleting a sales employee.

AFTER triggers are generally used to update columns in linked tables that depend on the row being inserted, updated or deleted for their values. For example, the PERCENT_CHANGE column in the SALARY_HISTORY table is maintained using an AFTER UPDATE trigger on the EMPLOYEE table.

To summarize: Use BEFORE until all data manipulation operations have been completed. The EMPLOYEE database trigger SET_CUST_NO is an example of a BEFORE INSERT, as a new customer number is generated before the data set has been inserted.

When manipulation of the table data should have been concluded before checking or altering other data, then use an AFTER trigger. The EMPLOYEE database trigger SAVE_SALARY_CHANGE is an example of AFTER UPDATE trigger, as the changes to the data have already been completed, before the trigger fires.

INSERT, UPDATE, DELETE

A trigger must be defined to fire on one of the keywords INSERT, UPDATE or DELETE.

1. An INSERT trigger fires before or after a row is inserted into the table.
2. An UPDATE trigger fires when a row is modified in the table.
3. A DELETE trigger fires when a row is deleted from the table.

If the same trigger needs to fire on more than one operation, a universal trigger needs to be defined. Before Firebird 1.5 triggers were restricted to either insert or update or delete actions, but now only one trigger needs to be created for all of these. For example:

```
AS
   BEGIN
      if (new.bez<>'')
      then new.bez=upper(new.bez);
   END
```

The '' UPPER applies to INSERT and UPDATE operations.

Please note that special characters, such as German umlauts, are not recognized and altered to upper case, as the character is treated technically as a special character, and not an alphabetical letter.

For further information regarding NEW variables, please refer to NEW and OLD context variables.

NEW and OLD context variables

In triggers (but not in stored procedures), Firebird/InterBase® provides two context variables that maintain information about the row being inserted, updated or deleted:

1. OLD.columnName refers to the current or previous values in a row being updated or deleted. It is not relevant for INSERT triggers.
2. NEW.columnName refers to the new values in a row being inserted or updated. It is not relevant for DELETE triggers.

Using the OLD. and NEW. values you can easily create history records, calculate the amount or percentage of change in a numeric value, find records in another table that match either the OLD. or NEW. value or do pretty well anything else you can think of. Please note that NEW. variables can be modified in a BEFORE trigger; since the introduction of Firebird 2.0 it is not so easy to alter them in an AFTER trigger. OLD. variables cannot be modified.

It is possible to read to or write from these trigger variables.

New to Firebird 2.0: Restrictions on assignment to context variables in triggers

- Assignments to the OLD context variables are now prohibited for every kind of trigger.
- Assignments to NEW context variables in AFTER-triggers are also prohibited.

Tip: If you receive an unexpected error Cannot update a read-only column then violation of one of these restrictions will be the source of the exception.