

Stored Procedures and Triggers

Content

Trigger

Creating a Trigger Function

Testing the Trigger Function

Trigger

- A *trigger* defines an operation that is performed **when a specific event occurs on a table/view**:
 - inserts a new record / updates an existing record, or deletes a record.
- The function executed as a result of a trigger is called a *trigger function*.
 - Trigger function **format looks similar to the stored procedure function** (same CREATE OR REPLACE FUNCTION command)
 - 2 two things:
 - Trigger functions **do not use input arguments** in the function, but rather are **passed arguments from a trigger event**
 - Trigger functions have access to **special variables** from the database engine

CREATE TRIGGER command

```
CREATE TRIGGER name  
{ BEFORE | AFTER | INSTEAD OF } { event [OR ... ] } ON table/view  
[ FOR [ EACH ] { ROW | STATEMENT } ]  
[ WHEN ( condition ) ]  
EXECUTE PROCEDURE function (arguments)
```

WHEN (condition) that determines whether the trigger function will actually be executed

BEFORE, AFTER can be used for tables and views

INSTEAD OF can be **only used for views at row-level**

<https://www.postgresql.org/docs/13/sql-createtrigger.html>

CREATE TRIGGER command - Explain

- Can occur either before or after the *event* occurs (**INSERT, UPDATE, DELETE, TRUNCATE** on the *table*)
 - multiple events can be specified using **OR**
 - UPDATE events, it is possible to specify a list of columns using this syntax: **UPDATE OF** *column_name1* [, *column_name2* ...]
 - **INSTEAD OF UPDATE** events do not support lists of columns.

(The **TRUNCATE TABLE** command deletes the data inside a table, but not the table itself)

- Fire triggers:
 - **ROW**: for each row that is affected by the event
 - **STATEMENT**: for each statement that triggers the event, no matter how many rows are returned (even if no rows are returned)

CREATE TRIGGER command - Explain

- CREATE TRIGGER check_update
BEFORE UPDATE ON accounts
FOR EACH ROW
WHEN (OLD.balance IS DISTINCT FROM NEW.balance)
EXECUTE PROCEDURE check_account_update();

<file:///C:/Program%20Files/PostgreSQL/9.4/doc/postgresql/html/sql-createtrigger.html>

CREATE TRIGGER command

- *function*: execute when the trigger is fired
 - the arguments in the CREATE TRIGGER command are passed using the `TG_ARGV` special variable
- When a trigger function is called, the database engine *passes a group of special variables to the trigger function* → define the environment
 - how the function was called
 - what data is present when the trigger was fired
 - when the trigger was fired
 - ...

Special Variable	Description
NEW	The record column data values present in the INSERT or UPDATE command
OLD	The record column data values present in the table before an UPDATE or DELETE command is executed
TG_NAME	The name of the called trigger
TG_WHEN	When the trigger was fired, either BEFORE or AFTER the SQL command
TG_LEVEL	The trigger definition, either ROW or STATEMENT
TG_OP	The event that fired the trigger, either INSERT, UPDATE, or DELETE
TG_RELID	The OID of the table that fired the trigger
TG_RELNAME	The name of the table that fired the trigger
TG_NARGS	The number of arguments in the CREATE TRIGGER command
TG_ARGV[]	An array containing the arguments used in the CREATE TRIGGER command

2. Creating a Trigger Function

- Can use the pgAdmin III program to create trigger functions:
 - right-click the Trigger Functions object →select New Trigger Function
 - Set the **Language** textbox to **plpgsql**
 - A trigger function updates table records → VOLATILE function
 - Parameter tab, NOT allowed to define arguments.
 - Definition textbox →enter the function code

Example (previous class)

-- 2. Create a view from eduDB, named student_class_shortinfos, this view contains: student_id, firstname, lastname, gender, class name.

```
create or replace view student_class_shortinfos
AS
    select student_id, last_name, first_name ,
gender, dob, name
    from student s left join clazz c on
s.clazz_id = c.clazz_id;
```

Example (previous class)

```
-- define a INSTEAD OF INSERT trigger
CREATE TRIGGER insert_student_view
INSTEAD OF INSERT ON student_class_shortinfos
FOR EACH ROW
EXECUTE PROCEDURE
insert_view_student_class_shortinfos();
```

Example (previous class)

```
-- define a trigger function
CREATE OR REPLACE FUNCTION
insert_view_student_class_shortinfos() RETURNS
trigger AS
$$
BEGIN
    -- insert student
    insert into student (student_id, last_name,
first_name , gender, dob) values (NEW.student_id,
NEW.last_name, NEW.first_name, NEW.gender, NEW.dob) ;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql ;
```

Remarks

- RETURN in a trigger function
 - **NULL**
 - One record having the **same structure** as table record on which the trigger is defined
- Trigger « **AFTER** »:
 - RETURN NULL; -- or RETURN NEW; RETURN OLD;
- Trigger « **BEFORE** »
 - **RETURN NULL;** : subsequent triggers are not fired, and the INSERT/UPDATE/DELETE does not occur for this row
 - Trigger BEFORE DELETE : **RETURN OLD;**
 - Trigger BEFORE UPDATE OR INSERT: **RETURN NEW;**

Example

Given EduBD:

student(student id, first_name, last_name, dob, gender, address, note, *class_id*)

subject(subject id, name, credit, percentage_final_exam)

lecturer(lecturer id, first_name, last_name, dob, gender, address, email)

teaching(subject id, lecturer id)

grade(code, fromScore, toScore)

clazz(clazz id, name, *lecturer_id*, *monitor_id*, number_students)

enrollment(student id, subject id, semester, midterm_score, final_score)

Exercise No 2 from last week

2) Add a new attribute (named: `number_students`, data type: integer) on clazz table to store number of students in class.

- Define a function (named `update_number_students()`) that computes the number of students for each class and update the correct value for `number_students` attribute.
- Check values in this attribute before and after calling the function `update_number_students()`

Have you done it?

Example

When a new student arrives (a new record is inserted into student table), the number of students in her/his class must be automatically updated

```
-- define a trigger
CREATE TRIGGER af_insert
AFTER INSERT ON student
FOR EACH ROW
WHEN (NEW.clazz_id IS NOT NULL)
EXECUTE PROCEDURE tf_af_insert();

-- define a trigger function
CREATE OR REPLACE FUNCTION tf_af_insert() RETURNS TRIGGER AS $$
BEGIN
    update clazz
    set number_students = number_students+1
    where clazz_id = NEW.clazz_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```


Exercise

Given EduBD, write triggers to ensure the following requirement:

- If data on student table is changed, the number of students in clazz table is always correct.

(delete a student, change student class)

- Assuming that the number of students enrolled in a subject per semester does not exceed 200, write a trigger that guarantees this constraint.

(insert, update event on enrolment table)