netlink
digital solutions

DATABASE

www.netlink.com

PostgreSQL Triggers

A PostgreSQL trigger is a function invoked automatically whenever an event such as insert, update, or delete occurs.

What are PostgreSQL Triggers?

A trigger is a special user-defined function connected with a table. if we want to generate a new trigger:

- **Firstly, we can specify a trigger function.**
- **Secondly, bind the same trigger function to a table.**

- A PostgreSQL Trigger is a function, which involved automatically whenever an event linked with a table.
- The event can be described as any of the following **INSERT, UPDATE, DELETE or TRUNCATE.**

Type of Triggers In PostgreSQL

- **Row-level trigger**
- **Statement-level trigger**

For example, if we issue an **UPDATE** command, which affects 10 rows, the **row-level trigger** will be invoked **10 times**, on the other hand, the **statement level trigger** will be invoked **1 time**.

Note: The major variance between a trigger and a user-defined function is that, when any triggering event occurs, a trigger is automatically raised.

Triggers

What is the Trigger function?

A trigger function is parallel to the consistent user-defined function. But a trigger function can return a value with the type trigger and does not take any parameters.

Syntax of Create trigger function

```
CREATE FUNCTION trigger_function()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
AS $$
BEGIN
  -- trigger logic goes here?
END;
$$
```

- A trigger function receives data about its calling environment through a special structure called TriggerData which contains a set of local variables.
- For example, **OLD and NEW** represent the states of the row in the table before or after the triggering event.
- PostgreSQL also provides other local variables preceded by TG_ such as TG_WHEN, and TG_TABLE_NAME.
- Once you define a trigger function, you can bind it to one or more trigger events such as INSERT, UPDATE, and DELETE.

Triggers

How to Create a New Trigger

Step1: Firstly, we will create a trigger function with the help of the **CREATE FUNCTION** command.

Step2: Then, we will fix the trigger function to a table with the help of the **CREATE TRIGGER** command.

The syntax of the PostgreSQL **CREATE TRIGGER** command is as follows:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} { event }
ON table_name
[FOR [EACH] { ROW | STATEMENT }]
EXECUTE PROCEDURE trigger_function ()
```

Steps:

1. First, specify the name of the trigger after the TRIGGER keywords.
2. Second, specify the timing that cause the trigger to fire. It can be BEFORE or AFTER an event occurs.
3. Third, specify the event that invokes the trigger. The event can be INSERT , DELETE, UPDATE or TRUNCATE.
4. Fourth, specify the name of the table associated with the trigger after the ON keyword.
5. Fifth, specify the type of triggers which can be.
6. Finally, specify the name of the trigger function after the EXECUTE PROCEDURE keywords.

TRIGGER

To delete a trigger from a table, you use the DROP TRIGGER statement with the following syntax:

DROP TRIGGER [IF EXISTS] trigger_name

ON table_name [CASCADE | RESTRICT];

To disable a trigger, you use the ALTER TABLE DISABLE TRIGGER statement:

ALTER TABLE table_name
DISABLE TRIGGER trigger_name | ALL

the ALL keyword to disable all triggers associated with the table.

The ALTER TRIGGER statement allows you to rename a trigger. The following shows the syntax of the ALTER TRIGGER statement:

ALTER TRIGGER trigger_name
ON table_name
RENAME TO new_trigger_name;

To enable a trigger or all triggers associated with a table, you use the ALTER TABLE ENABLE TRIGGER statement:

ALTER TABLE table_name
ENABLE TRIGGER trigger_name | ALL;