Lab 5

PL/pgSQL

PL/pgSQL is the procedural extension to SQL with features of programming languages

Allows using general programming tools with SQL, for example: loops, conditions, functions, etc.

The basic unit in any PL/pgSQL code is a BLOCK.

Types

- Anonymous blocks (DO)
  • Generally constructed dynamically and executed only once by the user. It is sort of a complex SQL statement.
- Named blocks (Functions and Stored Procedures)
  • Have a name associated with them, are stored in the database, and can be executed repeatably, and can take in parameters

Structure of Anonymous Block

```
DO $$
[ <<label>> ]
DECLARE
/* Declare section (optional). */
BEGIN
/* Executable section (required). */
EXCEPTION
/* Exception handling section (optional). */
END [ label ]
```

Structure of Named Blocks

```
CREATE FUNCTION [ function_name ] ()
RETURNS [return_type] $$
[ <<label>> ]
DECLARE
/* Declare section (optional). */
BEGIN
/* Executable section (required). */
```

```
EXCEPTION
/* Exception handling section (optional). */
END [ label ]
$$ LANGUAGE plpgsql;
```

Triggers

A “trigger” is defined as any event that sets a course of action in a motion.

Invokes the required function on defined events on specific database events, such as INSERT, UPDATE, DELETE, or TRUNCATE.

We can create trigger BEFORE, AFTER or INSTEAD of the events/operation.

Creating a trigger

A trigger is broken into two pieces
• Trigger
• Trigger Function (A function with no parameters that returns TRIGGER)

```
CREATE TRIGGER
```

Syntax

```
CREATE [ OR REPLACE ] [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD
OF } { event [ OR ... ] }
    ON table_name
    [ FROM referenced_table_name ]
    [ NOT DEFERRABLE | [ DEFERRABLE ] [ INITIALLY IMMEDIATE | INITIALLY
DEFERRED ] ]
    [ REFERENCING { { OLD | NEW } TABLE [ AS ] transition_relation_name } [
... ] ]
    [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( condition ) ]
    EXECUTE { FUNCTION | PROCEDURE } function_name ( arguments )

where event can be one of:

    INSERT
    UPDATE [ OF column_name [, ... ] ]
    DELETE
    TRUNCATE
```

```
CREATE FUNCTION trg() RETURNS trigger AS $$
BEGIN
```

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