Database Developmen t and Design

Qinshift Academy

Database Development and Design

Developing and Design of databases using PostgreSQL - Powerful, open-source object-relational database

Agenda

- Session 5
  - Homework discussion
  - Quiz
  - Table valued functions
    - Workshop
  - Variables and constants
    - Row variables
    - Record types
    - Constants
    - Workshop
  - Control structures
    - If statement
    - Workshop
  - Knowledge check (Discussion, Homework)

Table valued functions

Table valued functions

- A table valued function is usually a name for functions that return a table.
- To define a function that returns a table, you use the following form of the create function statement:

```
create or replace function function_name (
   parameter_list
)
returns table (column_list)
language plpgsql
as $$
declare
-- variable declaration
begin
-- body
end; $$
```

Variables and constants

Row variables (row type)

- To store the whole row of a result set returned by the select into statement, you use the row-type variable or row variable.
- You can declare a variable that has the same datatype as the datatype of the row in a table by using the following syntax:

```
row_variable table_name%ROWTYPE;
row_variable view_name%ROWTYPE;
```

- To access an individual field of the row variable, you use the dot notation (.) like this:

```
row_variable.field_name
```

Record types

- PostgreSQL provides a “type” called the record that is similar to the row-type.
- To declare a record variable, you use a variable name followed by the record keyword like this:

```
variable_name record;
```

- A record variable is similar to a row-type variable. It can hold only one row of a result set.
- Unlike a row-type variable, a record variable does not have a predefined structure. The structure of a record variable is determined when the select or for statement assigns an actual row to it.
- To access a field in the record, you use the dot notation (.) syntax like with row types.

Constants

- Unlike a variable, the value of a constant cannot be changed once it initialized.
- Constants make code more readable and maintainable.
- To define a constant in PL/pgSQL, you use the following syntax:

```
constant_name constant data_type := expression;
```

Control structures

If statement

- The if statement determines which statements to execute based on the result of a boolean expression.
- PL/pgSQL provides you with three forms of the if statements.
- if then
- if then else
- if then elsif

if-then statement

- The if statement executes statements if a condition is true. If the condition evaluates to false, the control is passed to the next statement after the END if part.
- The condition is a boolean expression that evaluates to true or false.
- The statements can be one or more statements that will be executed if the condition is true. It can be any valid statement, even another if statement.
- When an if statement is placed inside another if statement, it is called a nested-if statement.

```
if condition then
   statements;
end if;
```

if-then-else statement

- The if then else statement executes the statements in the if branch if the condition evaluates to true; otherwise, it executes the statements in the else branch.

```
if condition then
  statements;
else
  alternative-statements;
END if;
```

if-then-elsif statement

```
if condition_1 then
  statement_1;
elsif condition_2 then
  statement_2
...
elsif condition_n then
  statement_n;
else
  else-statement;
end if;
```

Found variable

- The found is a global variable that is available in PL/pgSQL. If the select into statement sets the found variable if a row is assigned or false if no row is returned.

```
select * from table_name
  into table_record
  where condition;

  if not found then
     -- Raise an error
  end if;
```

Homework 4

User-defined functions

- Get movies by rating (table-valued function)

  -- Usage example:

  SELECT * FROM **get_movies_by_rating**('PG-13');

- Get director's filmography (table-valued function)

  -- Usage example:

  SELECT * FROM get_director_filmography(1);

- Calculate actor's age

  -- Usage example:

  SELECT first_name, last_name, birth_date, **calculate_actor_age**(birth_date) as age

  FROM actors

  WHERE birth_date IS NOT NULL;

- Check if actor has won awards

  -- Usage example:

  SELECT first_name, last_name, **has_won_awards**(actor_id) as has_awards

  FROM actors;

Questions?

Trainer Name

Trainer mail

Assistant Name

Assistant mail

Qinshift Academy