BANCO DE DADOS II

User Defined
Functions

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Msc. Eliezio Soares
eliezio.soares@ifrn.edu.br | https://docente.ifrn.edu.br/elieziosoares

FUNÇÕES PL/SQL

Uma FUNCTION é um bloco de código PL/SQL que pode ser invocado sempre que necessário.

- Functions SQL devem possuir um nome e um tipo de retorno;
- Functions sempre retornam valores;
- Functions aceitam parâmetros;
- Functions podem ser usadas como parte de uma expressão, em selects e joins;

FUNÇÕES POSTGRESQL

Funções Date/Time

- age(timestamp, timestamp) returns interval
- current_date returns date
- current_time returns time
- current_timestamp returns timestamp
- now() returns timestamp
- date_part(text, timestamp) returns real
- ...

```
select age (now(), timestamp '1989-10-10');

select current_date;

select current_time;

select current_timestamp;

select date_part ('MINUTE', now() );
```

https://www.postgresql.org/docs/9.6/static/functions-datetime.html

FUNÇÕES POSTGRESQL

Funções de formatação

- to_char (timestamp, text) returns text
- to_date (text, text) returns date
- to_timestamp (text, text) returns timestamp

```
select to_char (current_timestamp, 'DD/MM/YYYY HH24:MM:SS');

select to_char (current_timestamp, 'DD, MON YYYY HH24:MM:SS');

select to_date ('07/09/2017 21:09:48','DD/MM/YYYY HH24:MM:SS');

select to_timestamp ('07/09/2017 21:09:48','DD/MM/YYYY HH24:MM:SS');

select to_timestamp (200120400);

select to_char (125, '999');
```

USER DEFINED FUNCTION

- Uma **UDF - User Defined Function** é uma função personalizada, criada sob demanda de um banco de dados por um de seus usuários.

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = }
default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
    [ WITH ( attribute [, ...] ) ]
```

CREATE FUNCTION
CREATE OR REPLACE FUNCTION

- **CREATE FUNCTION:** Define uma nova função.
- **CREATE OR REPLACE FUNCTION:** Define uma nova função, caso ja exista uma de mesmo nome essa será substituida.

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = }
default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
    [ WITH ( attribute [, ...] ) ]
```

NOME (ARGUMENTO)

- name ([[argmode] [argname] argtype [{ DEFAULT | = } default_expr] [, ...]])

- **Nome:** Nome da função criada.
- **argmode:** IN (default), OUT, INOUT, VARIADIC
  - IN: padrão;
  - OUT: Variável utilizada como retorno, podendo ser selecionável (semelhante a passagem de valor por referência);
  - INOUT: Pode ser passado pelo chamador da função e se comporta como OUT;
  - VARIADIC: Possibilita a passagem de arrays variados.

- **Argname:** Identificador da variável;
- **Argtype:** Tipo da variável;
- **DEFAULT:** Define um valor padrão para o argumento.

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = }
default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
    [ WITH ( attribute [, ...] ) ]
```

RETURNS

RETURNS rettype | RETURNS TABLE (column_name column_type [, ...])]

- **Rettype:** Tipo de retorno;
- **Lang_name:** Nome da linguagem (**sql**, c, internal, **plpgsql**, ...).

- IMMUTABLE | STABLE | VOLATILE (padrão) indicam ao otimizador de query sobre o comportamento da função.

- **IMMUTABLE:** Indica que a função não modifica o banco de dados e sempre retornará o mesmo resultado dados os mesmos valores como argumentos. Esse tipo de função não depende de consulta aos dados.

- **STABLE**: Indica que a função não modifica o banco de dados e que dentro de uma leitura simples na tabela ela consistentemente retornará o mesmo resultado, dados os mesmos valores como argumentos. No entanto este resultado pode mudar em diferentes declarações SQL. O resultado depende de parâmetros ou estado da tabela.

- **VOLATILE:** Indica que o valor da função pode mudar dentro de uma única consulta a tabela. Logo não há otimização que se possa fazer.

ARGUMENTOS NULL

CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT

- **CALLED ON NULL INPUT (padrão):** Indica que a função poderá ser chamada normalmente se algum de seus argumentos for null. A checagem passa a ser responsabilidade do autor da função.

- **RETURNS NULL ON NULL INPUT:** Indica que se a função receber algum argumento null ela invariavelmente retornará null.

- **STRICT:** Indica que se a função receber algum argumento null ela invariavelmente retornará null.

FUNÇÕES SQL X FUNÇÕES PLPGSQL

- **Funções SQL** executam uma lista de declarações SQL retornando o resultado da última query da lista.

- **Funções PLPGSQL** são funções que utilizam a **L**inguagem **P**rocedural **PGSQL:**
  - No PostgreSQL as **Stored Procedures** são definidas como functions na linguagem PLPGSQL.

EXEMPLO 1

- Uma função para somar dois inteiros.

```
CREATE FUNCTION add(integer, integer) RETURNS integer AS '
        select $1 + $2;
' LANGUAGE SQL
  IMMUTABLE
  RETURNS NULL ON NULL INPUT;
```

$1, $2, $N podem ser utilizados para se referia ao argumento 1, argumento 2, argumento N e assim sucessivamente, na ordem da declaração.

EXEMPLO 2

- Uma função para somar dois inteiros.

```
CREATE OR REPLACE FUNCTION increment(i integer) RETURNS integer AS $$
        BEGIN
                RETURN i + 1;
        END;
$$ LANGUAGE plpgsql;
```

FUNÇÕES SQL

- Funções SQL executam uma lista de declarações SQL retornando o resultado da última query da lista.
  - A primeira linha da última query será retornada.
  - Se a última query não retorna linhas, NULL será o retorno.

- Para retornar um conjunto de linhas o tipo de retorno deve ser especificado como RETURNS SETOF <TABLE>.
  - Nesse caso todas as linhas da última query serão retornadas.

- O corpo da função deve ser uma lista de declarações SQL separadas por ponto e vírgula.
- Funções SQL devem ter como último comando um SELECT, ou INSERT, UPDATE e DELETE com retorno (RETURNING).

```
CREATE FUNCTION clean_emp() RETURNS void AS '
    DELETE FROM emp
        WHERE salary < 0;
' LANGUAGE SQL;
```

EXEMPLO SQL FUNTION 1

- Uma função que remove alunos com matrícula menor que 17 e não retorna nenhuma linha.
  - Nesses casos deve ser indicada com RETURNS void.

```
CREATE OR REPLACE FUNCTION clean_alunos() RETURNS void AS '

   DELETE FROM alunos
      WHERE mat_alu < 17;

' LANGUAGE SQL;
```

EXEMPLO SQL FUNTION 2

- Uma função que seleciona alunos com base na matrícula e retorna uma linha da tabela Alunos.

```
CREATE OR REPLACE FUNCTION select_alunos(mat integer) RETURNS alunos AS $$

   select * from alunos where mat_alu = $1

$$ LANGUAGE SQL;
```

EXEMPLO SQL FUNTION 3

- Uma função que seleciona alunos com base na matrícula e retorna uma linha da tabela Alunos.

```
CREATE OR REPLACE FUNCTION nome_alunos(mat integer) RETURNS text AS $$
    select nom_alu from alunos where mat_alu = $1
$$ LANGUAGE SQL;
```

EXEMPLO SQL FUNTION 4

- Uma função que insere um novo aluno e retorna a matrícula gerada para o mesmo.

CTE:
Content Table Expressions.

```
CREATE OR REPLACE FUNCTION insert_alunos(cod_curso int, dat_nasc date, tot_cred int,
      mgp numeric, nom_alu text, email text) RETURNS int AS $$

    with inserido as(
      INSERT INTO ALUNOS(cod_curso, dat_nasc, tot_cred, mgp, nom_alu, email)
      VALUES($1,$2,$3,$4,$5,$6)
      RETURNING mat_alu
    )

    select mat_alu from inserido

$$ LANGUAGE SQL;
```

DÚVIDAS?

REFERÊNCIAS BIBLIOGRÁFICAS

PostgreSQL 9.0.22 Documentation. Disponível em: <https://www.postgresql.org/files/documentation/pdf/9.0/postgresql-9.0-US.pdf>. Acesso em 27 Set. 2016.