SQL (Funções)

Geomar A. Schreiner
geomarschreiner@gmail.com

Funções

- Programação estruturada
  - Procedimentos
  - Funções
- Em BD
  - Geralmente Procedures contém rollback e funções não
- Postgres
  - Tudo a mesma coisa

Funções

CREATE [OR REPLACE] FUNCTION

name([[argname] argtype])

[**RETURNS** tipo | [**TABLE** (cols)]]

AS $$

[**DECLARE** var tipo;]

BEGIN

  operações

END;

$$ LANGUAGE plpgsql;

Funções

Exemplo – condicional

```
CREATE OR REPLACE FUNCTION numero_par (i int)
RETURNS boolean AS $$
DECLARE
  temp int;
BEGIN
  temp := i % 2;
  IF temp = 0 THEN RETURN true;
  ELSE RETURN false;
  END IF;
END;
$$ LANGUAGE plpgsql;

SELECT numero_par(3), numero_par(42);
```

Funções

Exemplo – laço FOR

```
CREATE OR REPLACE FUNCTION fatorial (i numeric)
RETURNS numeric AS $$
DECLARE
 temp numeric; resultado numeric;
BEGIN
  resultado := 1;
  FOR temp IN 1 .. i LOOP
    resultado := resultado * temp;
  END LOOP;
  RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT fatorial(42::numeric);
```

Funções

Exemplo – laço WHILE

```
CREATE OR REPLACE FUNCTION fatorial (i numeric)
RETURNS numeric AS $$
DECLARE temp numeric; resultado numeric;
BEGIN
  resultado := 1; temp := 1;
  WHILE temp <= i LOOP
    resultado := resultado * temp;
    temp := temp + 1;
  END LOOP;
  RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT fatorial(42::numeric);
```

Funções

Exemplo – SQL dinâmico

```
CREATE OR REPLACE FUNCTION recupera_funcionario(id int)
RETURNS funcionario AS $$
DECLARE
  registro RECORD;
BEGIN
  EXECUTE 'SELECT * FROM funcionario WHERE id = ' || id INTO
registro;
  RETURN registro;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM recupera_funcionario(1);
```

Funções

Exemplo – cursor

```
CREATE OR REPLACE FUNCTION total_salarios()
RETURNS numeric AS $$
DECLARE
  registro RECORD; resultado numeric;
BEGIN
  resultado := 0.00;
  FOR registro IN SELECT * FROM funcionario LOOP
    resultado := resultado + registro.salario;
  END LOOP;
  RETURN resultado;
END;
$$ LANGUAGE plpgsql;

SELECT total_salarios();
```

Funções

- E quando eu preciso retornar apenas um valor de uma consulta?

```
SELECT select_expressions INTO [STRICT] target FROM ...;
INSERT ... RETURNING expressions INTO [STRICT] target;
UPDATE ... RETURNING expressions INTO [STRICT] target;
DELETE ... RETURNING expressions INTO [STRICT] target;
```

Funções

- E seu precisar retornar uma tabela?

```
1  CREATE OR REPLACE FUNCTION retornaCPFs()
2  RETURNS TABLE (
3                  cpf char
4                  ) AS $$
5  BEGIN
6      RETURN QUERY
7          SELECT c.cpf FROM clientes c;
8  END;
9  $$ LANGUAGE plpgsql;
```

Funções

- E seu precisar retornar uma tabela?

```
1  CREATE OR REPLACE FUNCTION retornaCPFs()
2  RETURNS TABLE (
3                  cpf char
4                  ) AS $$
5  DECLARE
6      tupla RECORD;
7  BEGIN
8      FOR tupla IN SELECT c.cpf FROM clientes c
9                  WHERE data_nasc = current_date LOOP
10         cpf := tupla.cpf || ' novo';
11         RETURN NEXT;
12     END LOOP;
13 END;
14 $$ LANGUAGE plpgsql;
```

Funções

- E quando eu preciso retornar apenas um valor de uma consulta?

```
BEGIN
    SELECT * INTO STRICT myrec FROM emp WHERE empname = myname;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE EXCEPTION 'employee % not found', myname;
        WHEN TOO_MANY_ROWS THEN
            RAISE EXCEPTION 'employee % not unique', myname;
END;
```

Funções

- E se eu quiser apresentar alguma informação para o usuário?

```
RAISE [ level ] 'format' [, expression [, ... ]] [ USING option = expression [, ... ] ];
RAISE [ level ] condition_name [ USING option = expression [, ... ] ];
RAISE [ level ] SQLSTATE 'sqlstate' [ USING option = expression [, ... ] ];
RAISE [ level ] USING option = expression [, ... ];
RAISE ;
```

- Opções

`DEBUG`, `LOG`, `INFO`, `NOTICE`, `WARNING`, and `EXCEPTION`

Funções

- E se eu quiser apresentar alguma informação para o usuário?

```
RAISE NOTICE 'Calling cs_create_job(%)', v_job_id;
```

```
RAISE EXCEPTION 'Nonexistent ID --> %', user_id
      USING HINT = 'Please check your user ID';
```

```
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = 'unique_violation';
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = '23505';
```

Funções

- E se eu quiser apresentar alguma informação para o usuário?

```
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = 'unique_violation';
RAISE 'Duplicate user ID: %', user_id USING ERRCODE = '23505';
```

Table A-1. PostgreSQL Error Codes

Error Code	Condition Name
Class 00 — Successful Completion	
00000	successful_completion
Class 01 — Warning	
01000	warning
0100C	dynamic_result_sets_returned
01008	implicit_zero_bit_padding
01003	null_value_eliminated_in_set_function
01007	privilege_not_granted
01006	privilege_not_revoked
01004	string_data_right_truncation

https://www.postgresql.org/docs/9.3/errcodes-appendix.html

Funções

- Como controlo as EXCEPTION?

```
BEGIN
    SELECT * INTO STRICT myrec FROM emp WHERE empname = myname;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE EXCEPTION 'employee % not found', myname;
        WHEN TOO_MANY_ROWS THEN
            RAISE EXCEPTION 'employee % not unique', myname;
END;
```

Funções

- Posso criar sub blocos?

```
[ <<rotulo>> ]
[ DECLARE
  declarações ]
BEGIN
  comandos
END [ rotulo ];
```

```
  RAISE NOTICE 'Quantidade aqui eh %', qtdade;  -- Imprime 30
  qtdade := 50;
  --
  -- Cria um subbloco
  --
  DECLARE
    qtdade integer := 80;
  BEGIN
    RAISE NOTICE ' Quantidade aqui eh %', qtdade;  -- Imprime 80
    RAISE NOTICE 'Quantidade externa eh  %', blocoexterno.qtdade;  -- Imprime 50
  END;
 RAISE NOTICE 'Quantidade aqui eh  %', qtdade;  -- Imprime 50
 RETURN qtdade;
END;
$$ LANGUAGE plpgsql;
```

Funções

- E os arrays?

```
1  CREATE OR REPLACE FUNCTION  recebendoArray( VARIADIC numeros NUMERIC[])
2  RETURNS integer AS $$
3  DECLARE
4      soma int; i int;
5  BEGIN
6      soma :=0;
7      FOREACH i IN ARRAY numeros LOOP
8          soma:= soma+i;
9      END LOOP;
10     RETURN soma;
11 END;
12 $$ LANGUAGE  plpgsql;
13
14
15 SELECT recebendoArray(VARIADIC ARRAY [1,2,3]);
```

Funções

- E os arrays?

```
1  CREATE OR REPLACE FUNCTION  retornaCPFs()
2  RETURNS TEXT[] AS $$
3  DECLARE
4      resultado TEXT[];
5      i RECORD;
6  BEGIN
7      resultado := ARRAY[]::TEXT[];
8      FOR i IN SELECT cpf FROM clientes  LOOP
9          resultado := array_append(resultado, i.cpf::TEXT);
10     END LOOP;
11     RETURN resultado ;
12 END;
13 $$ LANGUAGE  plpgsql;
```

Funções

- E os arrays?

```
CREATE OR REPLACE FUNCTION  retornaCPFs2()
RETURNS TEXT[] AS $$
DECLARE
    resultado TEXT[];
    i RECORD;
BEGIN
    resultado := ARRAY[]::TEXT[];
    FOR i IN SELECT cpf FROM clientes  LOOP
        resultado := resultado|| i.cpf::TEXT;
    END LOOP;
    RETURN resultado ;
END;
$$ LANGUAGE  plpgsql;
```

Funções

- E os arrays?

```
CREATE OR REPLACE FUNCTION  retornaCPFs2()
RETURNS TEXT[] AS $$
DECLARE
    resultado TEXT[];
    i RECORD;
BEGIN
    resultado := ARRAY[]::TEXT[];
    FOR i IN SELECT cpf FROM clientes  LOOP
        resultado := resultado|| i.cpf::TEXT;
    END LOOP;
    RETURN resultado ;
END;
$$ LANGUAGE  plpgsql;
```

Exercícios (part 1)

1) Faça uma função que calcule o fatorial de um número n;

2) Uma prática utilizada durante o desenvolvimento de aplicações que interagem com bancos de dados é a de definir procedimentos ou funções responsáveis pela inclusão, alteração e exclusão de registros. Para as tabelas de Habilitacao e Clientes, crie funções que atendam a essas operações, respeitando as seguintes regras:

   a) no caso de inclusões, a função deverá retornar a chave primária do novo registro como resultado;

```
INSERT [] RETURNING col
```

   b) no caso de alterações, a chave primária cujo registro deverá ser modificado deverá ser passada como parâmetro (juntamente com os dados a serem modificados no registro). O retorno dessa função deverá ser nulo;

   c) no caso de exclusões, a chave primária cujo registro deverá ser removido deverá ser passada como parâmetro. O retorno dessa função deverá ser true se algum registro foi excluido, e false caso contrário.

```
GET DIAGNOSTICS linhasAfetadas = ROW_COUNT;
```

Exercícios (part 2)

3) Crie uma função ‘passaRegua’ que deverá fechar a conta do cliente. A função deve receber como parâmetro o CPF do cliente e retornar o valor a ser pago pelo mesmo. Esta função deve alterar a data *fim* do na tabela locação com a data atual. A função deve somar e retornar o valor gasto com a locaçã (dias locação * valor diária do barco).

4) Crie uma nova tabela chamada de tabela teste. A tabela deve possuir dois campos id (serial primary key) e texto (varchar (100)). Crie uma funão que irá receber um inteiro como parametro, e esse inteiro corresponderá ao número de registros que você deve gerar para essa tabela.

   Para gerar um string aleatória use “`MD5(random()::text)`”

5) Faça uma função que deverá fazer a locação de um barco. Assim, essa função deve receber o funcionário, o cpf do cliente, a matricula do barco, a data de inicio da locação e a provavel devolução. Você deve validar se o barco a ser alugado esta disponível e se o cliente possui a habilitação para o veículo. Caso tudo esteja correto a função deve realizar a inserção dos dados e retornar true.