Instituto Federal

Campus Goiânia

Bacharelado em Sistemas de Informação

Banco de Dados II

Prof. Dory Gonzaga Rodrigues

MySQL®

ORACLE®

PostgreSQL
the world's most advanced open source database

Agenda

- STORED PROCEDURE

Funções

PL/PgSQL: Exercício

1) Faça uma função que retorne a idade de uma pessoa. Deve-se passar como parâmetro a data de nascimento.

```
CREATE OR REPLACE FUNCTION Idade ( IN dt_nasc date, OUT idade int ) AS
    $$ BEGIN
        idade = date_part('year',age(dt_nasc));
    END; $$
LANGUAGE PLPGSQL;

SELECT idade('1976-03-31');
```

Funções

PL/PgSQL: Exercício

2) Faça uma função que retorne a média de preço de venda dos CDs de uma determinada gravadora. Deve-se passar como parâmetro o código da gravadora.

```
CREATE OR REPLACE FUNCTION Med_Preco_CD ( IN cod int , OUT media float ) AS
    $$BEGIN
        SELECT INTO media AVG(cd.Preco_Venda)
        FROM cd
        WHERE idgravadora = cod;
    END; $$
LANGUAGE PLPGSQL;

SELECT Media_Preco_CD(2);
```

Funções

PL/PgSQL: Exercício

3) Faça uma função que retorne o nome do CD e o nome da Música. Deve-se passar como parâmetro o código do CD e o número da faixa do CD.

```
CREATE OR REPLACE FUNCTION Nome_CD_e_Musica ( IN codcd int,
                                              IN nfaixa  int,
                                              OUT nome text,
                                              OUT nome_m text ) AS
    $$ BEGIN
        SELECT cd.nome_cd, m.nomemusica INTO nome, nome_m
        FROM CD cd NATURAL JOIN FAIXA f NATURAL JOIN MUSICA m
        WHERE cd.idCD = codcd
            AND f.idmusica = nfaixa;
    END; $$
LANGUAGE PLPGSQL;

SELECT nome AS CD, nome_m AS Musica FROM Nome_CD_e_Musica(1,1);
```

Funções

PL/PgSQL: Exercício

4) Faça uma função que retorne a quantidade de faixas e o tempo total de músicas de um CD. Deve-se passar como parâmetro o código do código do CD.

```
CREATE OR REPLACE FUNCTION QtdeF_TempoCD (  IN codcd int ,
                                            OUT qtde_f  int,
                                            OUT tempo_cd' time ) AS
    $$BEGIN
        SELECT count(f.idmusica), sum(m.duracao) INTO  qtde_f, tempo_cd
          FROM cd NATURAL JOIN faixa f NATURAL JOIN musica m
        WHERE cd.idcd = codcd;
    END; $$
LANGUAGE PLPGSQL;

SELECT nomeg, qtde_f, tempo_cd FROM QtdeF_TempoCD(1);
```

Funções

PL/PgSQL: Tipo RECORD e SETOF

Para retornar um único registro utilizamos a palavra RECORD, veja o cabeçalho apresentado abaixo:

```
CREATE OR REPLACE FUNCTION retorna_linha() RETURNS record AS
```

Para retornar mais de uma linha continuaremos retornando um tipo RECORD, porém, precisamos definir que retornaremos um SETOF RECORD, veja o cabeçalho da função abaixo:

```
CREATE OR REPLACE FUNCTION retorna_linhas() RETURNS SETOF record AS
```

A utilidade do SETOF é justamente essa: informar o interpretador que o retorno poderá ter várias linhas.

Funções

PL/PgSQL: Instrução RETURN QUERY

Com já foi visto anteriormente, o comando RETURN serve exatamente para retornar um resultado/valor para quem chamar a função.

Quando uma função PL/pgSQL for declarada para retornar várias linhas (utilizando o SETOF), o comando deverá ser RETURN QUERY. Veja o exemplo:

```
CREATE OR REPLACE FUNCTION retorna_linhas() RETURNS SETOF record AS
    $$ BEGIN
        RETURN QUERY SELECT * FROM pessoas ;
    END;$$
LANGUAGE PLPGSQL;
```

Funções

PL/PgSQL: Exercício DESAFIO

5) Faça uma função que retorne os nomes das músicas de um CD. Deve-se passar como parâmetro o código do CD. (Desafio)

```
CREATE OR REPLACE FUNCTION Musicas_CD ( IN codcd int )
  RETURNS SETOF RECORD AS
    $$ BEGIN
        RETURN QUERY SELECT m.nomemusica
                      FROM cd NATURAL JOIN faixa f NATURAL JOIN musica m
                     WHERE idcd = codcd;
    END; $$
LANGUAGE PLPGSQL;

SELECT * FROM Musicas_CD(1) AS (nomeMusica VARCHAR);
```

Funções

PL/PgSQL: Criando um TYPE

- Observe abaixo o comando SQL que utilizamos para chamar a função Musica_CD. Como a função retorna um conjunto de registros, temos que declarar os campos (colunas) e os respectivos tipos que estarão no registro.

```
SELECT * FROM Musicas_CD(1) AS (nomeMusica VARCHAR);
```

Funções

PL/PgSQL: Criando um TYPE

- Podemos **criar um tipo** de dado que corresponda ao **RECORD** que será retornado, não sendo mais necessária a declaração feita no comando **SELECT.** Veja o exemplo:

```
CREATE TYPE musicas AS (nomeMusica VARCHAR, duracao TIME);

CREATE OR REPLACE FUNCTION Musicas_CD ( IN codcd int )
  RETURNS SETOF musicas AS
     $$ BEGIN
          RETURN QUERY SELECT m.nomemusica, m.duracao
                          FROM cd NATURAL JOIN faixa f NATURAL JOIN musica m
                         WHERE idcd = codcd;
     END; $$
LANGUAGE PLPGSQL;


SELECT * FROM Musicas_CD(1);
```

Funções

PL/PgSQL: Sobrecarga de função

- A linguagem PL/PgSQL permite que várias funções tenham o mesmo nome, desde que o número de argumentos seja diferente:

```
1)  CREATE FUNCTION Soma ( IN x int , IN y int, OUT total int ) AS $$
        BEGIN
            total = x + y;
        END; $$
    LANGUAGE PLPGSQL;

2)  CREATE FUNCTION Soma ( VARIADIC x int[] , OUT total int ) AS $$
        BEGIN
        SELECT INTO total sum( x[ i ] ) FROM generate_subscripts( x, 1 ) AS g( i );
        END; $$
    LANGUAGE PLPGSQL;

    SELECT Soma ( 3, 7);
    SELECT Soma ( 3, 7, 10 );
```

Funções

PL/PgSQL: Uso da Condicional

- A linguagem PL/PgSQL oferece a estrutura condicionais IF e CASE com as seguintes sintaxes:

1)
```
IF <condicional> THEN
    <comandos>
ENDIF;
```

2)
```
IF <condicional> THEN
    <comandos>
ELSE
    <comandos>
ENDIF;
```

3)
```
IF <condicional> THEN
    <comandos>
ELSEIF <condicional> THEN
    <comandos>
ELSE
    <comandos>
ENDIF;
```

Funções

PL/PgSQL: Exemplo do uso da Condicional

```
CREATE OR REPLACE FUNCTION categoria_preco_cd (IN codcd int,
                                               OUT categoria Text) AS $$
    DECLARE valor float;
    BEGIN
      SELECT preco_venda INTO valor FROM cd WHERE idcd = codcd;
      IF (valor <= 10) THEN
          categoria := 'Bronze';
      ELSEIF (valor > 10 AND valor <= 13) THEN
          categoria := 'Prata';
      ELSE
          categoria := 'Ouro';
      END IF;
    END; $$
LANGUAGE PLPGSQL;

SELECT categoria_preco_cd(1);
```

Funções

PL/PgSQL: Uso da Condicional

- A linguagem PL/PgSQL oferece a estrutura condicionais IF e CASE com as seguintes sintaxes:

4)
```
CASE <variável>
    WHEN valor(es) THEN
        <comandos>
    ELSE
        <comandos>
END CASE;
```

5)
```
CASE
    WHEN <expressão_booleana> THEN
        <comandos>
    ELSE
        <comandos>
END CASE;
```

Funções

PL/PgSQL: Exemplo do uso da Condicional

```
CREATE OR REPLACE FUNCTION categoria_preco_cd2 (IN codcd int,
                                                OUT categoria Text) AS $$
    DECLARE valor float;
    BEGIN
      SELECT preco_venda INTO valor FROM cd WHERE idcd = codcd;
      CASE
          WHEN (valor < 10.00)  THEN categoria := 'Bronze';
          WHEN (valor BETWEEN 10.00 AND 13.00)  THEN categoria := 'Prata';
          ELSE categoria :=  'Ouro';
      END CASE;
    END; $$
LANGUAGE PLPGSQL;

SELECT categoria_preco_cd2(1);
```

Funções

PL/PgSQL: Comando EXIT e o CONTINUE

- A instrução EXIT tem como funcionalidade terminar/concluir a execução de um bloco de comando, passando o controle para a instrução seguinte após o final do bloco. Se for utilizado em conjunto com WHEN, a saída do bloco de comandos ocorrerá somente se a expressão booleana for verdadeira. Caso contrário, a instrução que estiver logo após EXIT será executada.

- A instrução CONTINUE tem como funcionalidade retornar a execução do comando para o início do bloco de comando. Se for utilizado em conjunto com WHEN, a próxima instrução será iniciada somente se a expressão booleana for verdadeira. Caso contrário, controle passa para a instrução que estiver após CONTINUE.

Funções

PL/PgSQL: Uso do Laço

- A linguagem PL/PgSQL oferece a estrutura de laço LOOP, WHILE e FOR com as seguintes sintaxes:

1)
```
LOOP
    <comandos>
END LOOP;
```

2)
```
LOOP
    <comandos>
    EXIT WHEN <expressão_booleana>;
END LOOP;
```

3)
```
WHILE <expressão_booleana>  LOOP
    <comandos>
END LOOP;
```

Funções

PL/PgSQL: Uso do Laço

```
CREATE FUNCTION contagem (valor_ini INTEGER,valor_final INTEGER, passo
INTEGER)
    RETURNS SETOF INTEGER AS $$
    DECLARE
        valor INTEGER;    cont INTEGER;
    BEGIN
        cont := 0;
        LOOP
            valor := valor_ini + cont;
            cont := cont + passo;
            RETURN NEXT valor;
            EXIT WHEN (cont >= valor_final);
        END LOOP;
        RETURN;
    END;$$
LANGUAGE PLPGSQL;
```

```
SELECT contagem(1,10,1);
```

Funções

PL/PgSQL: Uso do Laço

- A linguagem PL/PgSQL oferece a estrutura de laço LOOP, WHILE e FOR com as seguintes sintaxes:

4)
```
FOR <variável>  IN [REVERSE] <valor_inicial> .. <valor_final> [BY <passo] LOOP
    <comandos>
END LOOP;
```

5)
```
FOR <variável_RECORD>  IN <consulta SQL> LOOP
    <comandos>
END LOOP;
```

Funções

PL/PgSQL: Uso do Laço

```
CREATE OR REPLACE FUNCTION codigo_cd_preco_acima (valor INTEGER)
  RETURNS SETOF INTEGER AS $$
  DECLARE
     registro RECORD;
  BEGIN
     FOR registro IN SELECT * FROM cd WHERE preco_venda > valor LOOP
          RETURN NEXT registro.idcd;
     END LOOP;
     RETURN;
  END; $$
LANGUAGE PLPGSQL;

SELECT * FROM codigo_cd_preco_acima (10);
```