Universidade Tecnológica Federal do Paraná

UTFPR - Campus Cornélio Procópio

AS33C
Banco de Dados 2

Professor: Eduardo Cotrin Teixeira

cotrin@utfpr.edu.br

UTFPR-CP / 2022

SQL – Stored Procedures e Funções

- **Desvio Condicional (IF) - Sintaxe:**

```
CREATE FUNCTION ...(...) RETURNS ... AS $$
DECLARE ...;
BEGIN
      ...
      IF condicao THEN
        ... Comandos ...
      ELSIF condicao THEN --opcional, repetível
        ... Comandos ...
      ELSE ... Comandos ...;
      END IF;
      ...
END; $$ LANGUAGE PLPGSQL;
```

SQL – Stored Procedures e Funções

```
CREATE OR REPLACE FUNCTION ClassificaForn(NomeForn
CHAR(30)) RETURNS TEXT AS $$
DECLARE Categ CHAR;
BEGIN
   Categ := (SELECT FCateg FROM Fornecedor
             WHERE FNome=NomeForn);
   IF Categ='A' OR Categ='B' THEN RETURN('PREMIUM');
   ELSIF Categ='C' THEN RETURN('BOM');
   ELSIF Categ='D' OR Categ='E' THEN RETURN('COMUM');
   ELSE RETURN('ERRO');
   END IF;
END; $$ LANGUAGE PLPGSQL;
```

SELECT ClassificaForn ('Plastec') --> 'PREMIUM'
SELECT ClassificaForn ('C&M') --> 'COMUM'
SELECT ClassificaForn ('Teste') --> 'ERRO'

SQL – Stored Procedures e Funções

```
CREATE OR REPLACE FUNCTION QtdeFornecida(Tipo CHAR(2),
Nome CHAR(30)) RETURNS INT AS $$
BEGIN
    IF Tipo='PE' THEN
      RETURN(SELECT SUM(Quant) FROM Fornece_para
             NATURAL JOIN Peca WHERE PeNome = Nome);
    ELSIF Tipo='PR' THEN
      RETURN(SELECT SUM(Quant) FROM Fornece_para
             NATURAL JOIN Projeto WHERE PNome = Nome);
    ELSIF Tipo='FO' THEN
      RETURN(SELECT SUM(Quant) FROM Fornece_para
             NATURAL JOIN Fornecedor WHERE FNome = Nome);
    END IF;
END; $$ LANGUAGE PLPGSQL;

SELECT PNome, QtdeFornecida('PR',PNome) AS PecasFornecidas
FROM Projeto;
```

SQL – Stored Procedures e Funções

```
CREATE OR REPLACE FUNCTION Dados(opcao CHAR, Peca CHAR(5),
Forn CHAR(5), Proj CHAR(5), Qtde INTEGER) RETURNS CHAR(10)
AS $$
DECLARE retorno CHAR(10);
BEGIN
 IF opcao = 'I' THEN
  INSERT INTO Fornece_para VALUES (Peca,Fornec,Proj,Qtde);
  retorno := 'INSERIDO';
 ELSIF opcao = 'U' THEN
  UPDATE Fornece_para SET Quant = Qtde
         WHERE PeNro=Peca AND FNro=Fornec AND PNro=Proj;
  retorno := 'ATUALIZADO';
 ELSE retorno := 'ERRO';
 END IF;     RETURN retorno;
END; $$ LANGUAGE PLPGSQL;

Select Dados('I','PE1','F5','P1',6); --> INSERIDO
Select Dados('A','PE1','F5','P1',5); --> ERRO
```

SQL – Stored Procedures e Funções

- **Exceção =** Erro. Término anormal da função, devido a um erro detectado pelo SGBD.

* Exemplos de exceção: duplicidade de chave primária, chave estrangeira inexistente.

- No PostgreSQL, quando há uma exceção, além da interrupção forçada da função a variável **SQLSTATE** assume automaticamente um valor de acordo com o erro detectado.

- Por exemplo, duplicidade de chave primária SQLSTATE = 23505 (demais valores na documentação do PostGreSQL).

https://www.postgresql.org/docs/10/static/errcodes-appendix.html

SQL – Stored Procedures e Funções

- O tratamento de exceções em PL/PGSQL pode ser feito de 2 formas: com **EXCEPTION** ou **RAISE EXCEPTION**.
- **EXCEPTION**: "Captura" a exceção disparada pelo SGBD. O SGBD altera o valor de SQLSTATE mas não há a interrupção da execução, e sim a execução dos comandos na função.
- **Sintaxe**:

```
BEGIN
... código ...
EXCEPTION WHEN SQLSTATE 'valor' THEN ...comandos...;
END;
```

*Funcionamento: Se SQLSTATE assumir o valor definido em **'valor'** durante a execução do bloco definido por **BEGIN**…**END**, então não haverá interrupção da função, e são executados os comandos do **THEN**. O bloco **BEGIN**…**END** pode englobar qualquer parte da função.

SQL – Stored Procedures e Funções

- Uso do EXCEPTION:

```
CREATE OR REPLACE FUNCTION InsereFornecePara(Peca CHAR(5),
Fornec CHAR(5), Proj CHAR(5), Qtde INTEGER) RETURNS
CHAR(10) AS $$
DECLARE retorno CHAR(10);
BEGIN
  BEGIN --exception
   INSERT INTO Fornece_para VALUES (Peca,Fornec,Proj,Qtde);
   retorno := 'INSERIDO';
   EXCEPTION WHEN SQLSTATE '23505' THEN retorno := 'ERRO';
  END; --exception
RETURN retorno;
END; $$ LANGUAGE PLPGSQL;

-Uso:
SELECT InsereFornecePara('PE1','F5','P4',5) --> ERRO
```

SQL – Stored Procedures e Funções

```
CREATE OR REPLACE FUNCTION ...
   EXCEPTION WHEN SQLSTATE '23505' THEN retorno := 'ERRO';
```

* Nesse exemplo, se houver duplicidade de chave primária (SQLSTATE=23505), a função termina normalmente e retorna o texto **'ERRO'**.

* Se houver outro erro detectado pelo SGBD, como chave estrangeira inexistente, o SGBD vai interromper a função e informar a ocorrência de erro.

Para outros casos pode ser usado:

```
EXCEPTION WHEN SQLSTATE '23505' OR SQLSTATE '23503' ...;
ou
EXCEPTION WHEN SQLSTATE '23000' ...;
Ou ainda
EXCEPTION
WHEN SQLSTATE '23505' THEN ...;
WHEN SQLSTATE '23503' THEN ...;
```

SQL – Stored Procedures e Funções

- Tratamento de exceções com **RAISE EXCEPTION**:
  - Dispara uma exceção, causando a interrupção da função pelo próprio comando, e não pelo SGBD. Não altera automaticamente o valor de SQLSTATE.
  - **Sintaxe**: `RAISE EXCEPTION 'mensagem';`

  * Usado para tratamento de **erros de semântica**, que são detectados pelo programador, mas são invisíveis para o SGBD (por exemplo, um valor acima de um limite, ou um retorno não desejado no SELECT).

  * O comando `RAISE` também pode ser usado com `NOTICE` para disparar uma mensagem no sistema, sem interrupção.

SQL – Stored Procedures e Funções

- Exemplo: Nome de um projeto obtido pelo número:

```
CREATE OR REPLACE FUNCTION NomeProj(NroProj CHAR(5))
RETURNS TEXT AS $$
DECLARE
      Nome CHAR(30);
BEGIN
    Nome := (SELECT PNome FROM Projeto WHERE PNro=NroProj);
    IF Nome IS NULL THEN
        RAISE EXCEPTION 'Projeto não existe.';
    END IF;
    RAISE NOTICE 'Projeto encontrado.';
    RETURN(Nome);
END; $$ LANGUAGE PLPGSQL;

SELECT NomeProj('P3');--> Alfa - MSG: Projeto encontrado.
SELECT NomeProj('P13');--> EXCEÇÃO: Projeto não existe.
```

Valor nulo se compara assim!

Não é um erro para o SGBD !!
Mas é para o programador !