cin.ufpe.br

Centro de Informática

U · F · P · E

Universidade Federal de Pernambuco

Fundamentos de PL/SQL

"Procedural Language/Structured Query Language"

Aula1

Apresentado por:

Robson do Nascimento Fidalgo

rdnf@cin.ufpe.br

Introdução

- É uma linguagem procedural da Oracle
  - Exige reimplementarão quando usar outro SGBD
- Estende a SQL-DML com comandos que permitem a criação de blocos de procedimentos de programação
  - Não aceita SQL-DDL!

```
LOOP
IF ... THEN
...
ELSE
...
END IF
END LOOP;
```

+

```
SELECT ...

INSERT ...

UPDATE ...

DELETE ...
```

= **PL/SQL**

Estrutura básica de um Bloco PL/SQL

[{DECLARE|PROCEDURE nome IS|FUNCTION nome RETURN tipo IS}]

Variáveis, Cursores, Exceções definidas pelos usuários, ...

BEGIN

Comandos SQL e/ou PL/SQL

[RETURN valor]

[EXCEPTION]

Ações para executar quando ocorrer erros

END;

Tipos de Bloco PL/SQL

Anônimo

```
[DECLARE]
BEGIN
 [EXCEPTION]
END;
```

Procedimento

```
PROCEDURE nome IS
BEGIN
 [EXCEPTION]
END;
```

Função

```
FUNCTION nome RETURN tipo IS
BEGIN
RETURN valor
 [EXCEPTION]
END;
```

Variáveis em PL/SQL

- Variáveis = identificadores
- Identificadores
  - Devem iniciar com uma letra
  - Podem incluir números ou caracteres especiais como $,#,_
  - Tamanho limitado até 30 caracteres
  - Não deve ser uma palavra reservada
  - Evite usar nome de colunas

Declarando/Inicializando Variáveis

- Sintaxe:

```
Identificador [CONSTANT] tipo [NOT NULL][:= | DEFAULT expr];
```

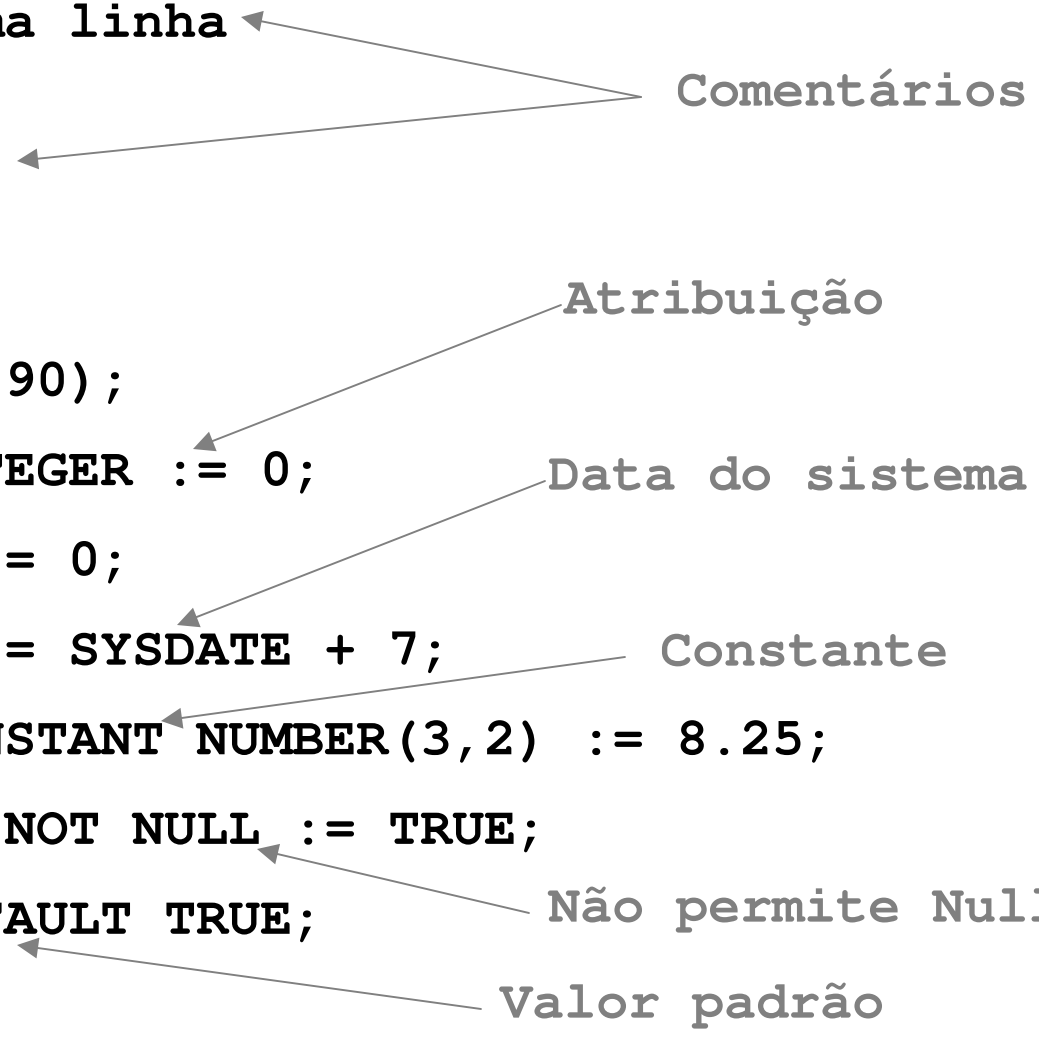
- Tipos de variáveis
  - Mesmos usados pelo Oracle

Numérico	**BINARY_INTEGER inteiro de -231 - 1 to 231 -1.** **NATURAL inteiro de 0 to 231** **POSITIVE inteiro de 1 to 231** **NUMBER(p,e) já visto, onde P é a precisão e E a escala**
Charactere	**CHAR(N) já visto, onde N é o tamanho fixo da string.** **VARCHAR2(N) já visto, onde N é o tamanho máximo da string**
Booleano	**BOOLEAN já visto, onde os valores lógicos são TRUE ou FALSE**
Data-Tempo	**DATE já visto, não esquecer de usar apóstrofe ‘ ’**

Declarando/Inicializando Variáveis

- Exemplos

```
-- um comentário de uma linha
/* um comentário de
mais de uma linha */
DECLARE
    descricao VARCHAR2(90);
    contador BINARY_INTEGER := 0;
    total NUMBER(9,2) := 0;
    data_entrega DATE := SYSDATE + 7;
    desconto_padrao CONSTANT NUMBER(3,2) := 8.25;
    brasileiro BOOLEAN NOT NULL := TRUE;
    fumante BOOLEAN DEFAULT TRUE;
    ...
```



Declarando/Inicializando Variáveis

- Mais exemplos (com saída de dados)

```
SET SERVEROUTPUT ON
DECLARE
    meuNome VARCHAR2(20);
BEGIN
    DBMS_OUTPUT.PUT_LINE('meu nome eh: '||meuNome);
    meuNome := 'Rita';
    DBMS_OUTPUT.PUT_LINE('meu nome eh: '||meuNome);
END;
/
```

ativa a saída de dados

imprime uma linha

Concatena string

Declarando/Inicializando Variáveis

- Mais exemplos (inicializando com apóstrofe)

```
DECLARE
   evento VARCHAR2(15);
BEGIN
   evento := q'!Father's day!';
   DBMS_OUTPUT.PUT_LINE('Dias dos pais ou :'||evento);
   evento := q'[Mother's day]';
   DBMS_OUTPUT.PUT_LINE('Dias das maes ou :'||evento);
END;
/
```

Declarando/Inicializando Variáveis

- Capturando o tipo de uma coluna (%TYPE)
  - Sintaxe:

```
identificador {tabela.coluna|variável}%TYPE;
```

  - Exemplo:

```
DECLARE
    ...
    descricao produto.desc%TYPE;
    balanco NUMBER(7,2);
    resumo_balanco balanco%TYPE := 1000;
    ...
```

Aninhando blocos

- Estrutura de aninhamento

```
DECLARE
    ...
BEGIN
    DECLARE
        ...
    BEGIN
        ...
    END;
    ...
END;
/
```

Aninhando blocos

- Estrutura de aninhamento – exemplo global X local

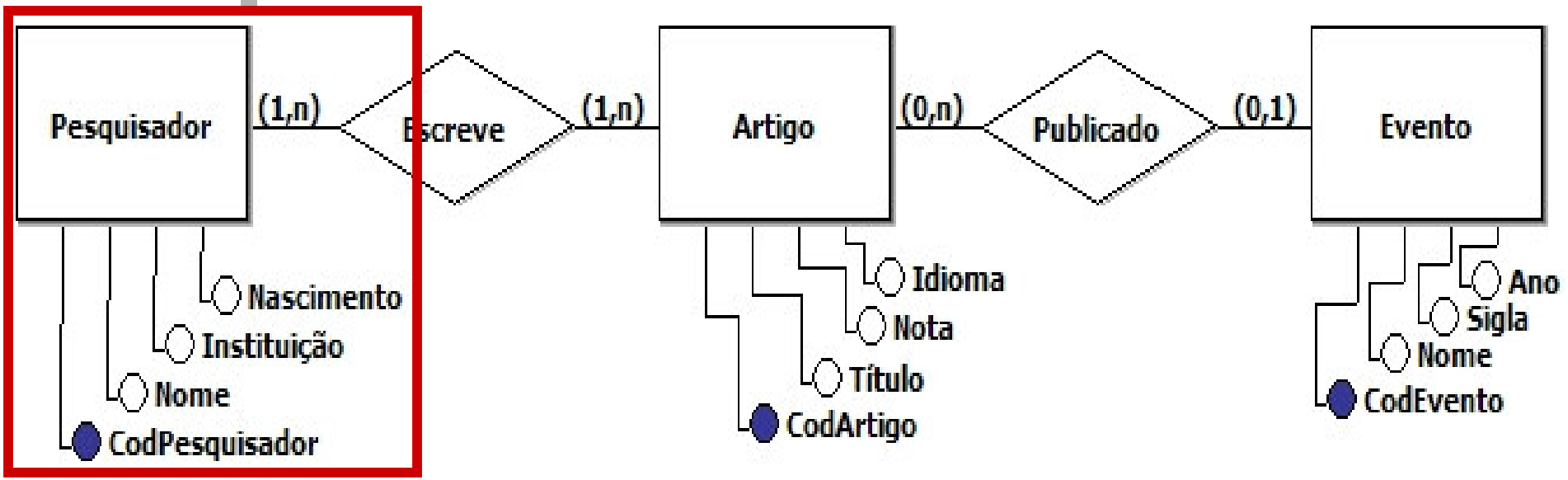
```
DECLARE
    nome_chefe VARCHAR2(20):='Rita';
    sexo CHAR:='F';
BEGIN
    DECLARE
        nome_empregado VARCHAR2(20):='Pedro';
        sexo CHAR:='M';
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Chefe: '||nome_chefe);
        DBMS_OUTPUT.PUT_LINE('Empregado : '||nome_empregado);
        DBMS_OUTPUT.PUT_LINE('Sexo empregado: '||sexo);
    END;
    DBMS_OUTPUT.PUT_LINE('Sexo Chefe: '||sexo);
END;
/
```

Aninhando blocos

- Estrutura de aninhamento – exemplo qualificado

```
<<global>>
DECLARE
    nome_chefe VARCHAR2(20):='Rita';
    sexo CHAR:='F';
BEGIN
    DECLARE
        nome_empregado VARCHAR2(20):='Pedro';
        sexo CHAR:='M';
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Chefe: '||nome_chefe);
        DBMS_OUTPUT.PUT_LINE('Sexo Chefe: '||global.sexo);
        DBMS_OUTPUT.PUT_LINE('Empregado : '||nome_empregado);
        DBMS_OUTPUT.PUT_LINE('Sexo empregado: '||sexo);
    END;
END;
```

Esquema ER & Relacional (exemplo)




```
Create Table Pesquisador(
CodPesquisador Varchar2(4),
Nome Varchar2(80) NOT NULL,
Instituicao Varchar(40) NOT NULL,
Nascimento Date,
Primary Key (CodPesquisador),
Unique (Nome, Nascimento));
```

Recuperando dados com SELECT

- Sintaxe:

```
SELECT coluna1[,coluna2, ... colunaN]

INTO variável1[,variável2, ... variávelN]

FROM tabela1 [,tabela2, ... tabelaN]

[WHERE condição1 [AND condição2 AND ... condiçãoN]];
```

Atenção: Consultas devem retornar apenas 1 linha!

Recuperando dados com SELECT

- Exemplo:

```
DECLARE
    qtdP number;
BEGIN
    SELECT COUNT(codPesquisador) INTO qtdP
    FROM pesquisador;
    DBMS_OUTPUT.PUT_LINE('Quantidade : '||qtdP);
END;
/
```

Recuperando dados com SELECT

- Exemplo:

```
DECLARE
    nomeP pesquisador.nome%TYPE;
    instituicaoP pesquisador.instituicao%TYPE;
BEGIN
    SELECT nome,instituicao INTO nomeP,instituicaoP
    FROM pesquisador WHERE codPesquisador = 'p001';
    DBMS_OUTPUT.PUT_LINE('Nome,Inst: '||nomeP);
    DBMS_OUTPUT.PUT_LINE('Nome,Inst: '||instituicaoP);
END;
/
```

Recuperando dados com SELECT

- Contra-Exemplo:

```
DECLARE
    nomeP pesquisador.nome%TYPE;
    instituicaoP pesquisador.instituicao%TYPE;
    codPesquisador pesquisador.codPesquisador%TYPE := 'p002';
BEGIN
    SELECT nome,instituicao INTO nomeP,instituicaoP
    FROM pesquisador WHERE codPesquisador = codPesquisador;
    DBMS_OUTPUT.PUT_LINE('Nome,Inst: '||nomeP);
    DBMS_OUTPUT.PUT_LINE('Nome,Inst: '||instituicaoP);
END;
/
```

Atenção: Não use nome de colunas para identificadores!
(nomes de colunas têm preferência sobre identificadores)

Inserindo dados com INSERT

- Exemplo:

```
BEGIN
    INSERT INTO pesquisador
    (codPesquisador, nome, instituicao, nascimento)
    VALUES('p003', 'Ana', 'UPE', '01/01/1970');
END;
/       BEGIN
           INSERT INTO pesquisador
           (codPesquisador, nome, instituicao, nascimento)
           VALUES('p004', 'Jose', 'UFRPE', '22/01/1976');
        END;
        /
```

Atualizando dados com UPDATE

- Exemplo:

```
DECLARE
nova_inst pesquisador.instituicao%TYPE := 'UFPE';
BEGIN
    UPDATE pesquisador
    SET instituicao = nova_inst
    WHERE codPesquisador = 'p003';
END;
/
```

Excluindo dados com DELETE

- Exemplo:

```
DECLARE
codP pesquisador.codPesquisador%TYPE := 'p003';
BEGIN
    DELETE FROM pesquisador
    WHERE codPesquisador = codP;
END;
/
```

Controlando o fluxo com IF

- Sintaxe:

```
IF condição1 THEN
   Bloco de comandos;
[ELSIF condição2 THEN
   Bloco de comandos;]
   ...
   [ELSIF condiçãoN THEN
      Bloco de comandos;]
      [ELSE
         Bloco de comandos;]
END IF;
```

Controlando o fluxo com IF

- Exemplo:

```
DECLARE
    x number:=0;
BEGIN
    IF (x>0) OR (x<0) THEN
       DBMS_OUTPUT.PUT_LINE(' X <> 0');
    ELSE
       DBMS_OUTPUT.PUT_LINE(' X = 0');
    END IF;
END;
/
```

Controlando o fluxo com IF

- Exemplo:

```
DECLARE
    x number:=10;
BEGIN
    IF x < 0 THEN
       DBMS_OUTPUT.PUT_LINE(' X < 0 ');
    ELSIF x > 0 THEN
       DBMS_OUTPUT.PUT_LINE(' X > 0 ');
       ELSE
          DBMS_OUTPUT.PUT_LINE(' X = 0 ');
    END IF;
END;
/
```

Controlando o fluxo com IF

- Contra-Exemplo:

```
DECLARE
   x number; -- não inicializou. Seu valor é NULL!!!
BEGIN
   IF x < 0 THEN
      DBMS_OUTPUT.PUT_LINE(' X < 0 ');
   ELSIF x > 0 THEN
      DBMS_OUTPUT.PUT_LINE(' X > 0 ');
      ELSE
         DBMS_OUTPUT.PUT_LINE(' X = 0 ');
   END IF;
END;
/
```

Atenção: Não esqueça de inicializar!
(como x = NULL, o resultado será o último ELSE 'X = 0')

Controlando o fluxo com IF

- Mais Exemplo:

```
DECLARE
    x number:=10;
BEGIN
    IF x <> 0 THEN
        IF x < 0 THEN
            DBMS_OUTPUT.PUT_LINE(' X < 0 ');
        ELSE
            DBMS_OUTPUT.PUT_LINE(' X > 0 ');
        END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE(' X = 0 ');
    END IF;
END;
/
```

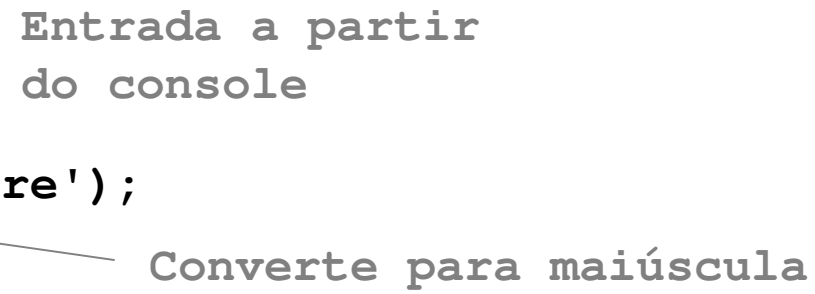
Controlando o fluxo com CASE

- Sintaxe:

```
CASE seletor
   WHEN expressão1 THEN resultado1
   WHEN expressão2 THEN resultado2
   ...
   WHEN expressãoN THEN resultadoN
   [ELSE resultadoN+1]
END;
/
```

Controlando o fluxo com CASE

- Exemplo:




```
DECLARE
  x CHAR(1) := UPPER('&caractere');
  descricao VARCHAR2(20);
BEGIN
  descricao :=
  CASE
    WHEN x = '+' THEN 'adicao'
    WHEN x = '-' THEN 'subtracao'
    WHEN x IN ('A','E','I','O','U') THEN 'vogal'
    ELSE 'DESCONHECIDO'
  END;
    DBMS_OUTPUT.PUT_LINE ('Caractere:  '|| x ||
                          ' é um(a) ' || descricao);
END;
/
```

(Funções para manipulação de strings)

- Exemplo:

```
DECLARE
   STR VARCHAR2(20) := UPPER('exemplo    ');
   STR2 VARCHAR2(20) := UPPER('emp');
BEGIN
   DBMS_OUTPUT.PUT_LINE ('ORIGINAL ['|| STR|| ']');
   DBMS_OUTPUT.PUT_LINE ('SEM ESPACOS A DIREITA['|| RTRIM(STR)|| ']');
   DBMS_OUTPUT.PUT_LINE ('TAMANHO : '|| LENGTH(STR));
   DBMS_OUTPUT.PUT_LINE ('MINÍSCULO : '|| LOWER(STR));
   DBMS_OUTPUT.PUT_LINE ('POSICAO DA STR2 EM STR :'|| INSTR(STR,STR2));
   DBMS_OUTPUT.PUT_LINE ('SUB-STRING DE 2 A 4:'|| SUBSTR(STR,2,4));
END;
/
```

```
RESULTADO:  ORIGINAL [EXEMPLO    ]
            SEM ESPACOS A DIREITA[EXEMPLO]
            TAMANHO : 10
            MINISCULO : exemplo
            DIZ A POSICAO INICIAL DA STR2 EM STR :3
            SUB-STRING DE 2 A 4:XEMP
```

Repetição com LOOP

- Sintaxe:

```
LOOP
    comandos;
EXIT [WHEN condição];
END LOOP;
```

Repetição com LOOP

- Exemplo sem WHEN condição:

```
DECLARE
   I NUMBER(6);
BEGIN
   I := 1;
   LOOP
      DBMS_OUTPUT.PUT_LINE('aI: ' || I);
      I := I + 1;
      IF I > 5 THEN
         EXIT;
      END IF;
      DBMS_OUTPUT.PUT_LINE('bI: ' || I);
   END LOOP;
END;
/
```

```
Resultado:
aI: 1
bI: 2
aI: 2
bI: 3
aI: 3
bI: 4
aI: 4
bI: 5
aI: 5
```

Repetição com LOOP

- Exemplo com WHEN condição:

```
DECLARE
    I NUMBER(6);
BEGIN
    I := 1;
    LOOP
        DBMS_OUTPUT.PUT_LINE('aI: ' || I);
        I := I + 1;
        EXIT WHEN I > 5;
        DBMS_OUTPUT.PUT_LINE('bI: ' || I);
    END LOOP;
END;
/
```

```
Resultado:
aI: 1
bI: 2
aI: 2
bI: 3
aI: 3
bI: 4
aI: 4
bI: 5
aI: 5
```

Repetição com LOOP

- Exemplo com WHEN condição2 :

```
DECLARE
    I NUMBER(6);
BEGIN
    I := 1;
    LOOP
        DBMS_OUTPUT.PUT_LINE('aI: ' || I);
        I := I + 1;
        DBMS_OUTPUT.PUT_LINE('bI: ' || I);
        EXIT WHEN I > 5;
    END LOOP;
END;
/
```

```
Resultado:
aI: 1
bI: 2
aI: 2
bI: 3
aI: 3
bI: 4
aI: 4
bI: 5
aI: 5
bI: 6
```

Repetição com WHILE

- Sintaxe:

```
WHILE condição LOOP
    comandos;
END LOOP;
```

Repetição com WHILE

- Exemplo:

```
DECLARE
    I NUMBER(6);
BEGIN
    I := 1;
    WHILE I <= 5 LOOP
        DBMS_OUTPUT.PUT_LINE('aI: ' || I);
        I := I + 1;
        DBMS_OUTPUT.PUT_LINE('bI: ' || I);
    END LOOP;
END;
/
```

```
Resultado:
aI: 1
bI: 2
aI: 2
bI: 3
aI: 3
bI: 4
aI: 4
bI: 5
aI: 5
bI: 6
```

Repetição com FOR

- Sintaxe:

```
FOR contador IN [REVERSE]lim_inf..lim_sup LOOP
    comandos
END LOOP;
```

Repetição com FOR

- Exemplo:

```
DECLARE
    J NUMBER(6);
    K NUMBER(6);
BEGIN
    J := 7;
    K := 2;
    FOR I IN K..J LOOP
        DBMS_OUTPUT.PUT_LINE('I: ' || I);
    END LOOP;
END;
/
```

Não se declara o contador, pois este é declarado implicitamente

```
Resultado:
I: 2
I: 3
I: 4
I: 5
I: 6
I: 7
```

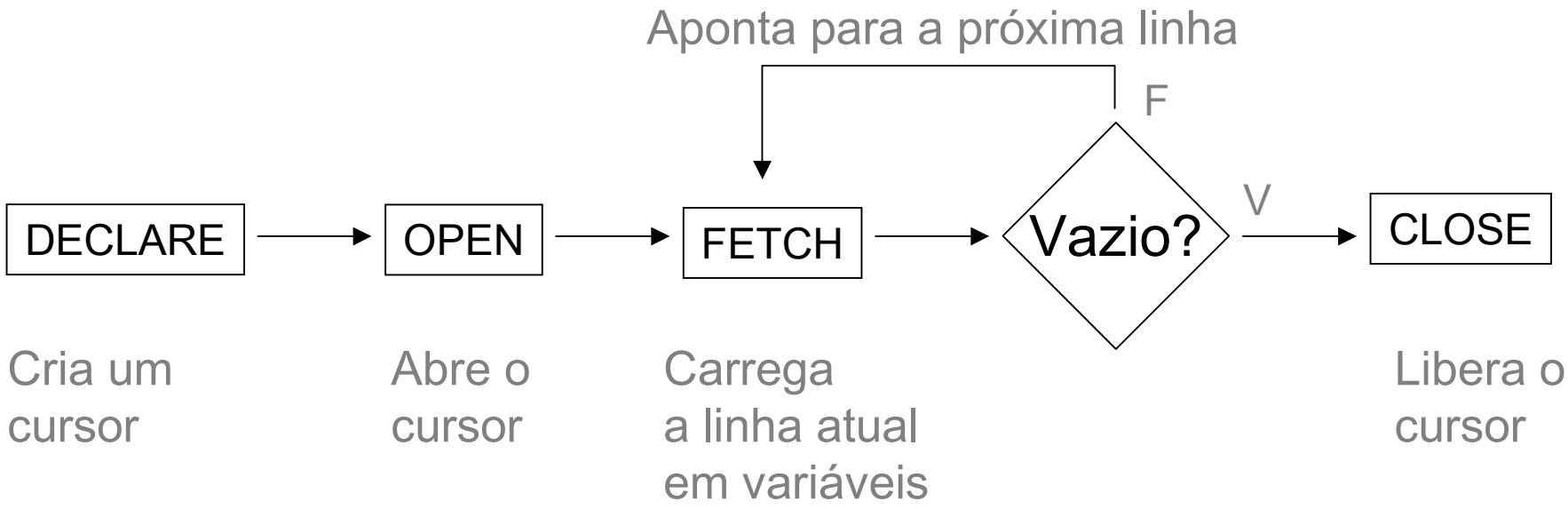
Cursor

- Semelhante a um ponteiro para manipular linhas de uma tabela temporária

- Podem ser de dois tipos:
  - Implícito: Declarado e gerenciado pelo servidor Oracle para todo comando DML e PL/SQL SELECT
  - Explicito: Declarado e gerenciado pelo programador. Gerado apenas pelo SELECT

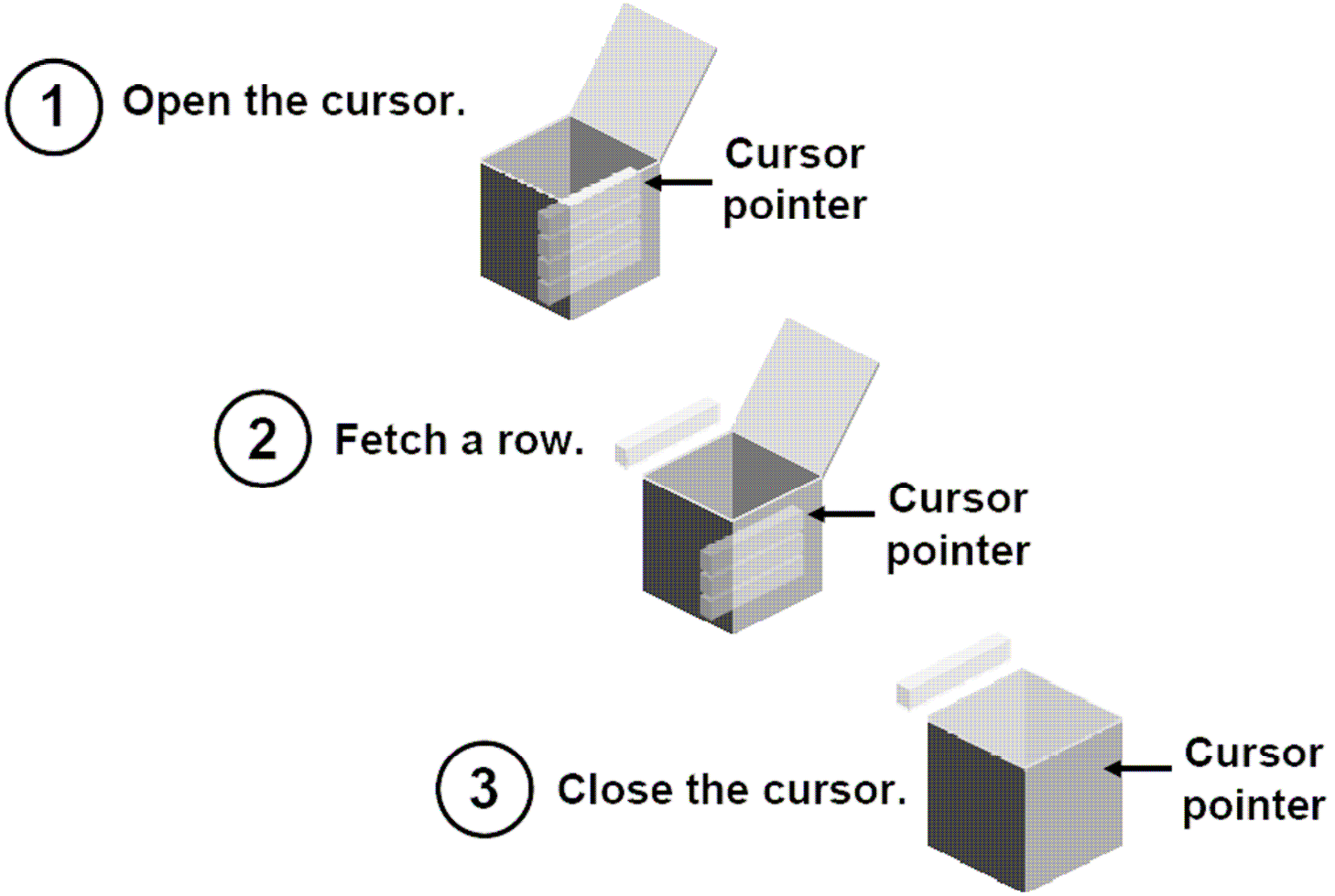
Cursor

- Fluxo de controle de cursores explícitos



Cursor

- Fluxo de controle de cursores explícitos



Cursor

- Sintaxe:

```
CURSOR nome_cursor IS
    comando_select;
```

- Exemplo:

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    ...
```

Abrindo um Cursor

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    ...
BEGIN
    OPEN pesq_cursor;
    ...
END;
/
```

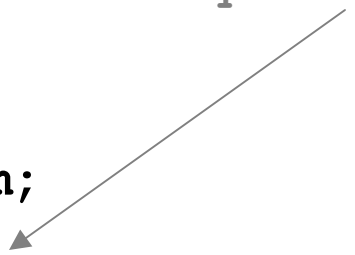
Lendo um Cursor

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    codP pesquisador.codPesquisador%TYPE;
    nm pesquisador.nome%TYPE;
    ...
BEGIN
    OPEN pesq_cursor;
    FETCH pesq_cursor INTO codp, nm;
    DBMS_OUTPUT.PUT_LINE( codp ||' '||nm);
    ...
END;
/
```

Lendo um Cursor

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    codP pesquisador.codPesquisador%TYPE;
    nm pesquisador.nome%TYPE;
    ...
BEGIN
    OPEN pesq_cursor;
    LOOP
        FETCH pesq_cursor INTO codp, nm;
        EXIT WHEN pesq_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( codp ||' '||nm);
    END LOOP;
    ...
END;
/
```

True quando não achar



(Atributos de Cursores Explícitos)

%rowcount	Quantidade de linhas afetadas pelo comando que gerou o cursor.
%found	**True** caso alguma linha tenha sido afetada.
%notfound	**True** caso alguma linha não tenha sido afetada.
%isopen	**True** caso o cursor esteja aberto (cursores explícitos).

Fechando um Cursor

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    codP pesquisador.codPesquisador%TYPE;
    nm pesquisador.nome%TYPE;
    ...
BEGIN
    OPEN pesq_cursor;
    LOOP
        FETCH pesq_cursor INTO codp, nm;
        EXIT WHEN pesq_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( codp ||' '||nm);
    END LOOP;
    CLOSE pesq_cursor;
    ...
END;
/
```

Tipo Registro em Cursor

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
        SELECT codPesquisador, nome FROM pesquisador
        WHERE instituicao = inst;
    pesq_reg pesq_cursor%ROWTYPE;
    ...
BEGIN
    OPEN pesq_cursor;
    LOOP
        FETCH pesq_cursor INTO pesq_reg;
        EXIT WHEN pesq_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( pesq_reg.codPesquisador ||
                              ' '||pesq_reg.nome);
    END LOOP;
    CLOSE pesq_cursor;
    ...
END;
/
```

Cursor com Laços FOR

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    ...
BEGIN
    FOR pesq_reg IN pesq_cursor LOOP
       DBMS_OUTPUT.PUT_LINE( pesq_reg.codPesquisador ||
                             ' '||pesq_reg.nome);
    END LOOP;
    ...
END;
/
```

Forma mais reduzida de manipular cursores, pois:

- Faz uso implícito dos comandos OPEN, FETCH, EXIT e CLOSE
- Faz a declaração implícita da variável de tipo registo

Cursor com Laços FOR (sem declaração)

```
DECLARE
    inst VARCHAR2(8):= 'UFPE';
    ...
BEGIN
    FOR pesq_reg IN (SELECT codPesquisador, nome FROM
    pesquisador WHERE instituicao = inst) LOOP
        DBMS_OUTPUT.PUT_LINE( pesq_reg.codPesquisador ||
                              ' '||pesq_reg.nome);
    END LOOP;
    ...
END;
/
```

Cursor com Parâmetros

```
DECLARE                                          Não informar o tamanho
   CURSOR pesq_cursor (inst VARCHAR2) IS
   SELECT codPesquisador, nome FROM pesquisador
   WHERE instituicao = inst;
   pesq_reg pesq_cursor%ROWTYPE;
   ...
BEGIN
   OPEN pesq_cursor('UFPE');
   LOOP
      FETCH pesq_cursor INTO pesq_reg;
      EXIT WHEN pesq_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE( pesq_reg.codPesquisador ||
                            ' '||pesq_reg.nome);
   END LOOP;
   CLOSE pesq_cursor;
   ...
END;
/
```

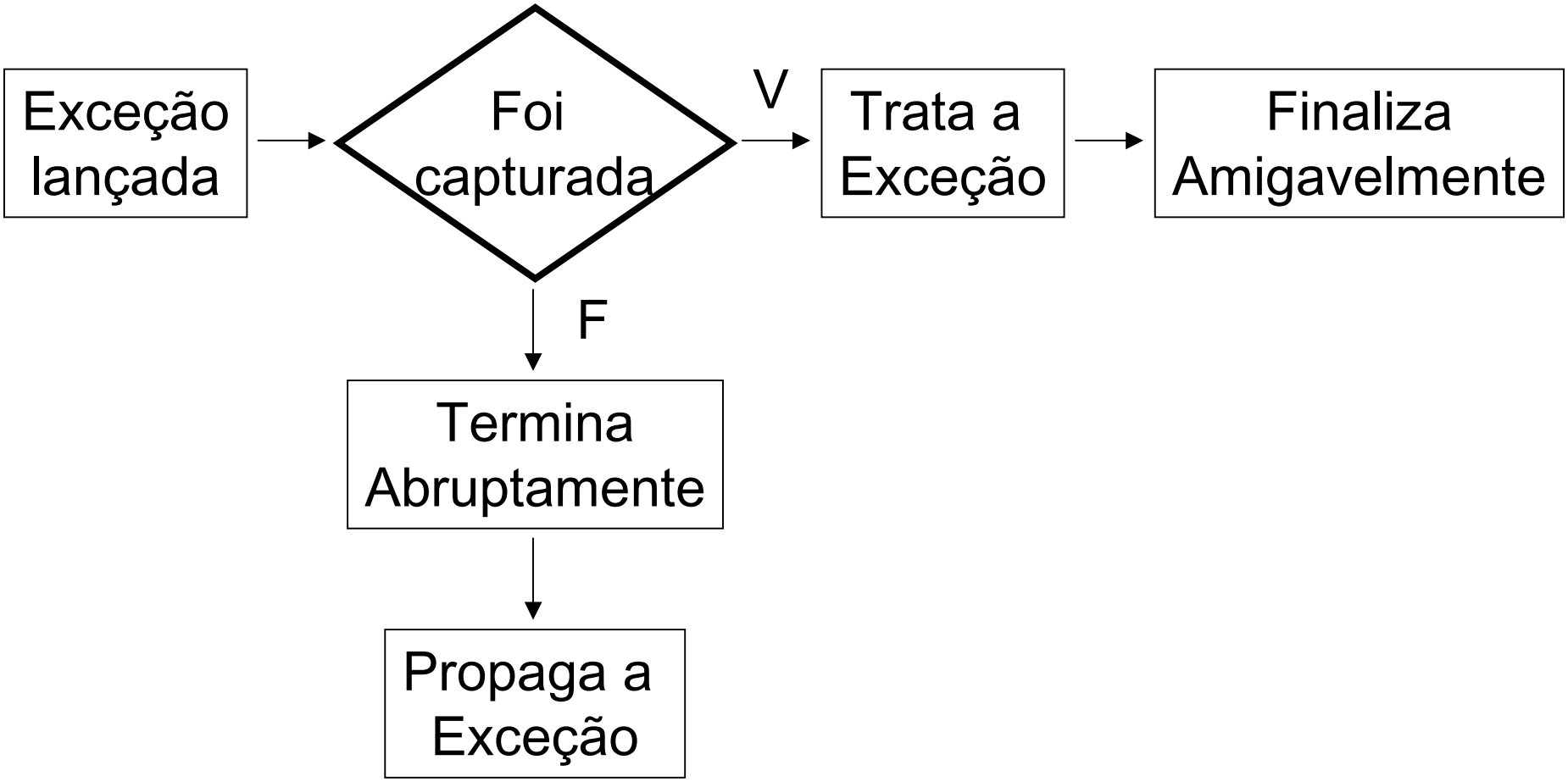
Cursor com Parâmetros

```
DECLARE
    CURSOR pesq_cursor (inst VARCHAR2) IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    pesq_reg pesq_cursor%ROWTYPE;
    ...
BEGIN
    OPEN pesq_cursor('UFPE');
    ...
    CLOSE pesq_cursor;
    ...
    OPEN pesq_cursor('UPE');
    ...
    CLOSE pesq_cursor;

END;
/
```

Cada nova passagem de parâmetro exige um OPEN(parâmetro)/CLOSE

Tratando exceções



Tratando exceções

- Sintaxe:

```
EXCEPTION
WHEN exceção1 [OR exceção2 . . .] THEN
    comando1;
    comando2;
    . . .
[WHEN exceção3 [OR exceção4 . . .] THEN
    comando1;
    comando2;
    . . .]
[WHEN OTHERS THEN
    comando1;
    comando2;
    . . .]
```

Exceções pre-definidas pelo Oracle

- NO_DATA_FOUND
- TOO_MANY_ROWS
- INVALID_CURSOR
- ZERO_DIVIDE
- DUP_VAL_ON_INDEX

Tratando exceções

- Exemplo:

```
DECLARE
    inst VARCHAR2(8) := UPPER('&instituicao');
    codP pesquisador.codPesquisador%TYPE;
    nm pesquisador.nome%TYPE;
BEGIN
    SELECT codPesquisador, nome INTO codP, nm FROM pesquisador
    WHERE instituicao = inst;
    DBMS_OUTPUT.PUT_LINE( codp ||' '||nm);
EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE ('Sua consulta deve usar cursor');
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE ('Sua consulta retornou null');
        insert into t_erro (descricao,valor)
                    values ('retorno null', inst);
     WHEN OTHERS  THEN
       NULL;
END;
```

Só fazer isso em último caso!
Oculta qualquer erro!

Subprogramas

- São blocos nomeados (não usar DECLARE)
- Podem ser:
  - Procedimentos (PROCEDURE)
    - Por padrão não retornam valor (exceção: modo OUT ou IN OUT)
  - Funções (FUNCTION)
    - Por padrão, necessariamente, retornam um único valor
- Blocos Anônimos X Subprogramas

Subprogramas	Blocos Anônimos
São compilados só uma vez	Sempre são compilados
São armazenados no BD	Não são armazenados no BD
Nomeados, por isso podem ser chamados por outras aplicações	Sem nome, por isso não podem ser chamados por outras aplicações
FUNCTION deve retornar um valor	Nunca retornam valor
Pode ser parametrizado	Não pode ser parametrizado

Subprogramas

- Sintaxe:
  - Procedimento

```
CREATE [OR REPLACE] PROCEDURE nome_procedimento [(parâmetro1
[modo1] tipo1, parâmetro2 [modo2] tipo2,. . .)] IS|AS
corpo_do_procedimento;
```

  - Função

```
CREATE [OR REPLACE] FUNCTION nome_função [(parâmetro1 [modo1]
tipo1, parâmetro2 [modo2] tipo2,. . .)] RETURN tipo_retorno IS|AS
corpo_da_função;
```

Subprogramas

- Exemplo procedimento:

```
CREATE OR REPLACE PROCEDURE pesquisadores_ufpe IS
    inst VARCHAR2(8):= 'UFPE';
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    codP pesquisador.codPesquisador%TYPE;
    nm pesquisador.nome%TYPE;
BEGIN
    OPEN pesq_cursor;
    LOOP
        FETCH pesq_cursor INTO codp, nm;
        EXIT WHEN pesq_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( codp ||' '||nm);
    END LOOP;
    CLOSE pesq_cursor;
END;
/
```

Subprogramas

- Exemplo função:

```
CREATE OR REPLACE FUNCTION qtd_pesquisadores_ufpe RETURN NUMBER IS
    qtdP number;
    inst VARCHAR2(8):= 'UFPE';
BEGIN
    SELECT count(codPesquisador) INTO qtdP
    FROM pesquisador
    where instituicao = inst;
    RETURN qtdP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN NULL;
END;
/
```

Chamando Subprogramas

- Procedimento:

```
BEGIN
    pesquisadores_ufpe;
END;
```

- Função:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Quantidade : '||qtd_pesquisadores_ufpe);
END;
```

Subprogramas Parametrizados

- Exemplo procedimento:

```
CREATE OR REPLACE PROCEDURE pesq_instituicao (inst Varchar) IS
    CURSOR pesq_cursor IS
    SELECT codPesquisador, nome FROM pesquisador
    WHERE instituicao = inst;
    codP pesquisador.codPesquisador%TYPE;
    nm pesquisador.nome%TYPE;
BEGIN
    OPEN pesq_cursor;
    LOOP
        FETCH pesq_cursor INTO codp, nm;
        EXIT WHEN pesq_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( codp ||' '||nm);
    END LOOP;
    CLOSE pesq_cursor;
END;
/
```

```
BEGIN
    pesq_instituicao('UFPE');
END;
```

Subprogramas Parametrizados

- Exemplo função:

```
CREATE OR REPLACE FUNCTION qtd_pesq_instituicao (inst Varchar)
RETURN NUMBER IS
    qtdP number;
BEGIN
    SELECT count(codPesquisador) INTO qtdP
    FROM pesquisador
    where instituicao = inst;
    RETURN qtdP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN NULL;
END;
/
```

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Quantidade :
    '||qtd_pesq_instituicao('UFPE'));
END;
```

Modos de Passagem de Parâmetro

- Pode ser:
  - IN (entrada - default)
    - Pode ser um número, string ou variável inicializada
  - OUT (saída – deve ser especificado)
    - Deve ser uma variável não inicializada
  - IN OUT (entrada/saída – deve ser especificado)
    - Deve ser uma variável inicializada

Subprogramas Parametrizados

- Exemplos :

```
CREATE OR REPLACE PROCEDURE soma_AB (A INT, B INT, C OUT INT)
IS
BEGIN
    C := A + B;
END;
```

```
DECLARE
    R INT;
BEGIN
    soma_AB(23,29,R);
    DBMS_OUTPUT.PUT_LINE('SOMA : ' || R);
END;
```

```
CREATE OR REPLACE PROCEDURE dobro (N IN OUT INT) IS
BEGIN
N := N * 2;
END;
```

```
DECLARE
R INT;
BEGIN
R := 7;
DBMS_OUTPUT.PUT_LINE('Antes, R : ' || R);
dobro(R);
DBMS_OUTPUT.PUT_LINE('Depois, R : ' || R);
END;
```

Excluindo Subprogramas

- Procedimento:

```
DROP PROCEDURE pesquisadores_ufpe;
DROP PROCEDURE pesq_instituicao;
```

- Função:

```
DROP FUNCTION qtd_pesquisadores_ufpe;
DROP FUNCTION qtd_pesq_instituicao;
```

Organizando Subprogramas (Pacotes)

- Agrupamento de subprogramas logicamente relacionados
- Os subprogramas devem ser especificados de forma que um subprograma ao ser “chamado” já tenha sido especificado antes
- Formado por 2 partes:
  - Especificação (público)
  - Corpo (privado)

Organizando Subprogramas (Pacotes)

- Sintaxe (especificação):

```
CREATE [OR REPLACE] PACKAGE nome_pacote IS|AS
    declarações (ex: tipos, variáveis e cursores) e
    assinaturas dos subprogramas
END [nome_pacote];
```

- Sintaxe (corpo):

```
CREATE [OR REPLACE] PACKAGE BODY nome_pacote IS|AS
    declarações (ex: tipos, variáveis e cursores)
    e corpo dos subprogramas
END [package_name];
```

Organizando Subprogramas (Pacotes)

- Exemplo (especificação):

```
CREATE OR REPLACE PACKAGE teste_pkg IS
    uma_var_qq INT;
    PROCEDURE soma_AB (A INT, B INT, C OUT INT);
    PROCEDURE dobro (N IN OUT INT);
END teste_pkg;
```

- Exemplo (corpo):

```
CREATE OR REPLACE PACKAGE BODY teste_pkg IS
    PROCEDURE soma_AB (A INT, B INT, C OUT INT) IS
    BEGIN
        C := A + B;
    END soma_AB;
    PROCEDURE dobro (N IN OUT INT) IS
    BEGIN
        N := N * 2;
    END dobro;
END teste_pkg;
```

Chamando Pacotes

```
DECLARE
    R INT;
BEGIN
    teste_pkg.soma_AB(23,29,R);
    DBMS_OUTPUT.PUT_LINE('SOMA : ' || R);
    R := 7;
    DBMS_OUTPUT.PUT_LINE('Antes, R : ' || R);
    teste_pkg.dobro(R);
    DBMS_OUTPUT.PUT_LINE('Depois, R : ' || R);
END;
```

Removendo Pacotes

- Sintaxe:

```
DROP PACKAGE package_name;

DROP PACKAGE BODY package_name;
```

exclui todo o pacote, incluindo seu corpo

só exclui o corpo do pacote

- Exemplo:

```
DROP PACKAGE teste_pkg;

DROP PACKAGE BODY teste_pkg;
```

Gatilhos (Triggers)

- São códigos de PL/SQL armazenados no SGBD, associados a um objeto do BD (ex: tabela ou visão)
- São executados implicitamente pelo SGBD na ocorrência de um determinado evento ou combinação deste (ex: logon, shutdown, criação, exclusão ou atualização de objetos)
- Podem ser usados para:
  - Registrar modificações
  - Garantir regras de negócio
  - Gerar valor de coluna
  - Manter tabelas duplicadas
- Recomenda-se não ultrapassar 60 linhas (chamar Procedures)

Gatilhos (Triggers)

- São códigos de PL/SQL armazenados no SGBD, associados a um objeto do BD (ex: tabela ou visão)
- São executados implicitamente pelo SGBD na ocorrência de um determinado evento ou combinação deste
- Podem ser usados para:
  - Registrar modificações
  - Garantir regras de negócio
  - Gerar valor de coluna
  - Manter tabelas duplicadas
- Recomenda-se não ultrapassar 60 linhas (chamar Procedures)
- Não tem como definir a ordem de execução entre triggers

Gatilhos (Triggers)

- Sintaxe:

```
CREATE [OR REPLACE] TRIGGER nome_trigger
momento evento1 [OR evento2 OR evento3]
ON nome_objeto
[[REFERENCING OLD AS apelido1 | NEW AS apelido2]
FOR EACH ROW
[WHEN (condição)]]
corpo_trigger
```

Gatilhos (Triggers)

- Um trigger de banco de dados tem 4 partes:
  - Momento
  - Evento
  - Tipo (só aplicável a eventos de DML)
  - Ação

Gatilhos (Triggers)

- Momento
  - Corresponde ao tempo (BEFORE, AFTER ou INSTEAD OF) em que o trigger deve ser executado
- Evento
  - DBA (ex: CREATE, ALTER, DROP, SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN, GRANT e REVOKE)
  - DML (INSERT, DELETE e UPDATE)
    - São os mais usados!

Gatilhos (Triggers)

- Evento – Exemplo DML:

```
CREATE OR REPLACE TRIGGER evento
BEFORE DELETE OR INSERT OR UPDATE OF instituicao ON pesquisador
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Código :'||:OLD.codPesquisador);
    DBMS_OUTPUT.PUT_LINE('Antiga Instituicao '||:OLD.instituicao||
                         'Nova Instituicao '||:NEW.instituicao);
END;
/
```

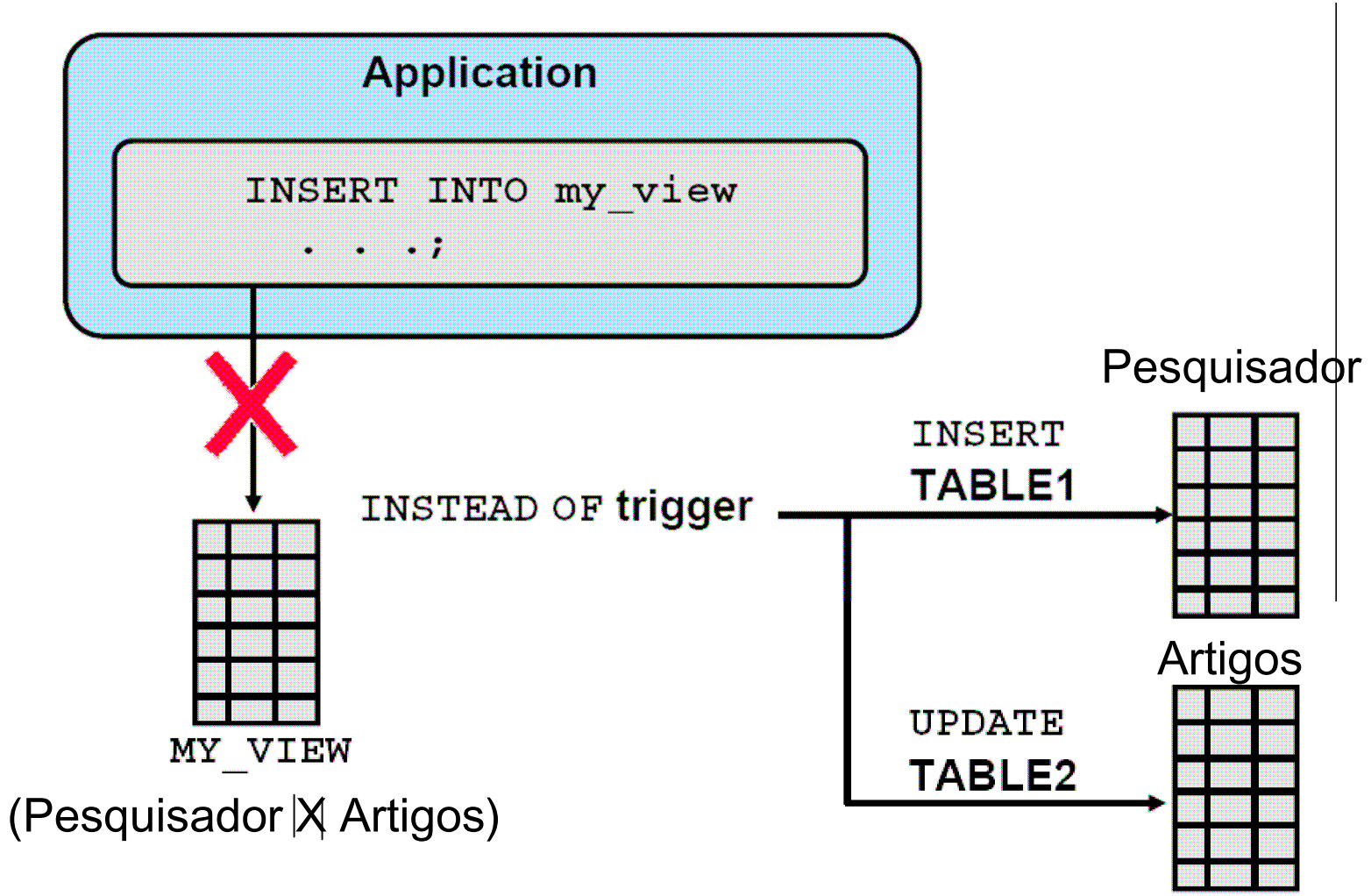
Gatilhos (Triggers)

- Evento – Exemplo VIEW (INSTEAD OF):

```
CREATE OR REPLACE TRIGGER eventoView
INSTEAD OF UPDATE ON visao_pesquisador_ufpe
FOR EACH ROW
BEGIN
    IF :OLD.nome <> :NEW.nome THEN
        DBMS_OUTPUT.PUT_LINE('Antigo Nome '||:OLD.nome||
                             'Novo Nome'||:NEW.nome);
        UPDATE pesquisador SET nome = :NEW.nome
        WHERE codPesquisador = :NEW.codPesquisador;
    END IF;
END;
```

Gatilhos (Triggers)

- Evento – Exemplo VIEW (cenário):



Gatilhos (Triggers)

- Evento – Exemplo VIEW (INSTEAD OF):

```
CREATE OR REPLACE TRIGGER eventoView2
INSTEAD OF INSERT ON visao_pesquisador_artigo
FOR EACH ROW
BEGIN
        INSERT INTO pesquisador VALUES (NEW.codPesquisador,
        NEW.nome, NEW.instituicao, NEW.nascimento);
        INSERT INTO artigo VALUES (NEW.codArtigo, NEW.titulo,
        NEW.nota, NEW.idioma, NEW.codEvento);
        INSERT INTO escreve VALUES (NEW.cod.Pesquisador,
        new.codArtigo);
END;
```

Gatilhos (Triggers)

- Tipo (só aplicável a eventos de DML)
  - Comando:
    - Acionado antes ou depois de um comando, independente deste atualizar ou não uma ou mais linhas
    - Não permite acesso as linhas atualizadas
    - Sintaticamente, basta não usar “FOR EACH ROW”
  - Linha:
    - Acionado para cada linha afetada
    - UPDATE requer a definição do campo (UPDATE OF campo)

Gatilhos (Triggers)

- Comando – Exemplo

```
CREATE OR REPLACE TRIGGER tipo_comando

AFTER UPDATE OF instituicao ON pesquisador

BEGIN

    DBMS_OUTPUT.PUT_LINE('Atualização feita com sucesso!');

END;

/
```

Gatilhos (Triggers)

- Linha – Exemplo

```
CREATE OR REPLACE TRIGGER tipo_linha
BEFORE DELETE OR INSERT OR UPDATE OF instituicao ON pesquisador
REFERENCING OLD AS V NEW as N
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Código :'||:V.codPesquisador);
    DBMS_OUTPUT.PUT_LINE('Antiga Instituicao '||:V.instituicao||
                         'Nova Instituicao '||:N.instituicao);
END;
/
```

Gatilhos (Triggers)

- Linha – Exemplo (com cláusula WHEN)

```
CREATE OR REPLACE TRIGGER tipo_linha
BEFORE DELETE OR INSERT OR UPDATE OF instituicao ON pesquisador
REFERENCING OLD AS V NEW as N
FOR EACH ROW
WHEN (V.instituicao = 'UFPE')
BEGIN
    DBMS_OUTPUT.PUT_LINE('Código :'||:V.codPesquisador);
    DBMS_OUTPUT.PUT_LINE('Antiga Instituicao '||:V.instituicao||
                         'Nova Instituicao '||:N.instituicao);
END;
/
```

ATENÇÃO:
Isto não restringe as linhas a serem atualizadas, apenas aquelas que acionarão o trigger

Gatilhos (Triggers)

- Ação
  - Bloco PL/SQL associado ao evento
  - Requer a definição de predicados lógicos quando o trigger tem mais de um evento DML
    - INSERTING: V quando o trigger foi disparado por um INSERT
    - UPDATING: V quando o trigger foi disparado por um UPDATE
    - DELETING: V quando o trigger foi disparado por um DELETE

Gatilhos (Triggers)

- Ação – Exemplo:

```
CREATE OR REPLACE TRIGGER tipo_linha_acoes
BEFORE DELETE OR INSERT OR UPDATE OF instituicao ON pesquisador
REFERENCING OLD AS V NEW as N
FOR EACH ROW
WHEN (V.instituicao = 'UFPE')
BEGIN
    IF UPDATING THEN -- ou IF UPDATING(campo) THEN
        DBMS_OUTPUT.PUT_LINE('Código :'||:V.codPesquisador);
        DBMS_OUTPUT.PUT_LINE('Antiga Instituicao '||:V.instituicao||
                             'Nova Instituicao '||:N.instituicao);
        ELSIF INSERTING THEN
            DBMS_OUTPUT.PUT_LINE('Código :'||:N.codPesquisador);
            DBMS_OUTPUT.PUT_LINE('Instituicao '||:N.instituicao);
            ELSE
                DBMS_OUTPUT.PUT_LINE('Código :'||:V.codPesquisador);
                DBMS_OUTPUT.PUT_LINE('Instituicao '||:V.instituicao);
    END IF;
END;
```

Gatilhos (Triggers)

- Removendo triggers – Sintaxe:

```
DROP TRIGGER trigger_name;
```

- Removendo triggers – Exemplo:

```
DROP TRIGGER tipo_linha_acoes;
```

cin.ufpe.br

Centro de Informática

U · F · P · E

Universidade Federal de Pernambuco