

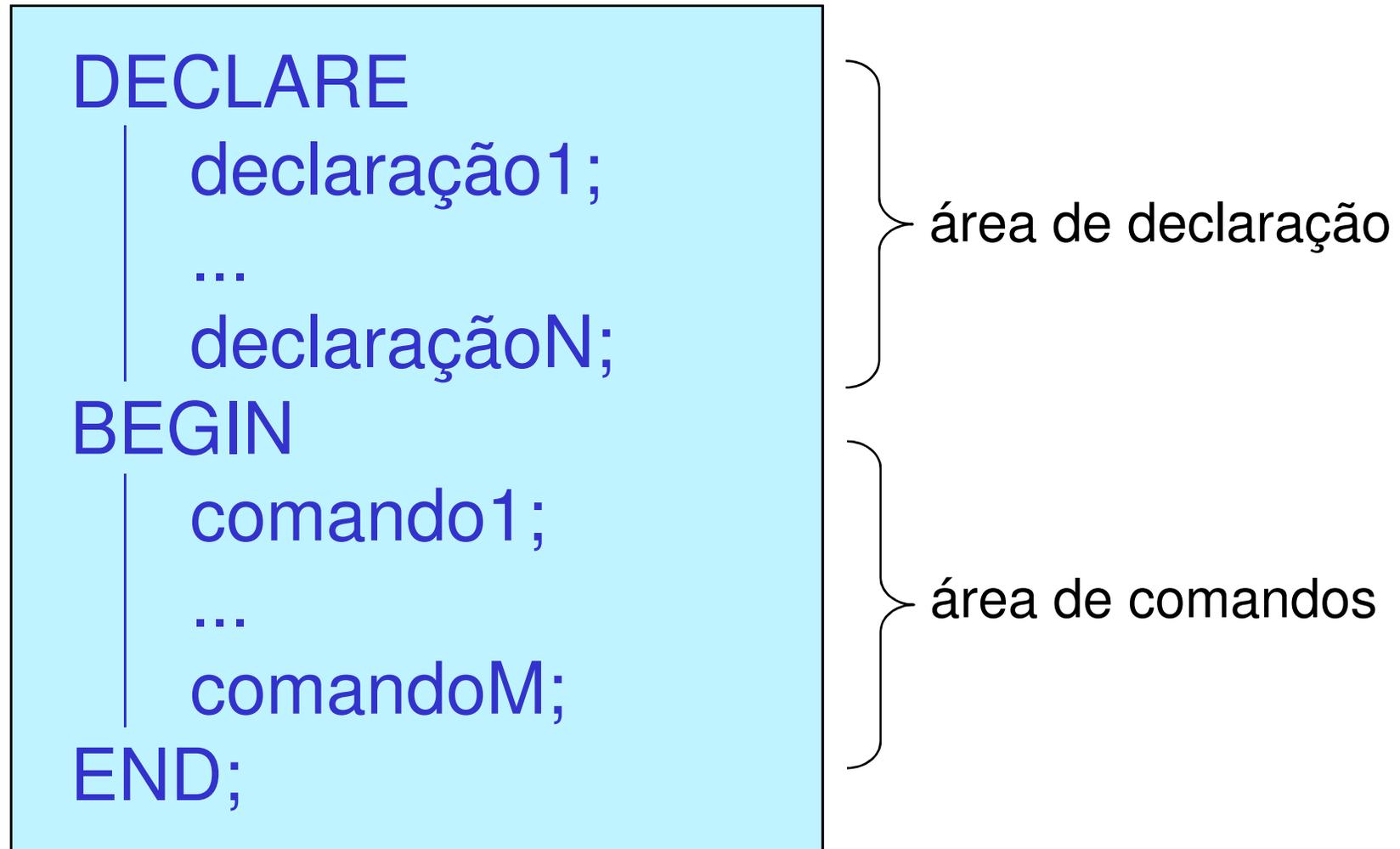
PL/SQL

Laboratório de Bases de Dados
Profa. Dra. Cristina Dutra de Aguiar Ciferri

PL/SQL

- Program Language SQL
 - linguagem de programação da Oracle® que permite processar dados armazenados no banco de dados
- Exemplos de operações permitidas
 - alterar, remover, inserir, selecionar dados
 - criar variáveis, constantes, cursores, registros
 - tratar erros
 - utilizar comandos de repetição, comparação

Estrutura: Bloco PL/SQL



DECLARAÇÃO

- Tipos de declaração
 - constantes
 - variáveis
 - cursores
 - estruturas
 - tabelas

DECLARAÇÃO

- Declaração de constantes
 - especificação de CONSTANT
 - atribuição de valores
 - constante := valor *ou*
 - definição de um valor DEFAULT
- Declaração de variáveis
 - atribuição de valores
 - variável := valor *ou*
 - DEFAULT valor

valor NULL é atribuído
a variáveis que não
foram inicializadas

DECLARAÇÃO

- Exemplos

constantes

n_alunos CONSTANT NUMBER(4) := 15; *ou*

n_alunos CONSTANT NUMBER(4) DEFAULT 15;

...

variáveis

valor0 NUMBER(4) := 0; *ou*

valor0 NUMBER(4) DEFAULT 0;

...

exemplo1 NUMBER(4) NOT NULL := 1;

exemplo2 NUMBER(4);

Tipos de Dados

- Mesmos vistos na primeira aula!
 - CHAR
 - VARCHAR2
 - CLOB
 - BLOB
 - BFILE
 - NUMBER
 - DATE
 - TIMESTAMP
- BOOLEAN
 - armazena os valores TRUE, FALSE, NULL

Herança de Tipo e Tamanho

- Característica
 - variáveis e constantes podem herdar o tipo de dado
 - de colunas
 - de outras variáveis
 - da linha inteira de uma tabela
- Vantagem
 - diminuição das manutenções oriundas de alterações nas definições das colunas das tabelas

Herança de Tipo e Tamanho

- Coluna de uma tabela
nome_var nome_tabela.nome_coluna%Type
- Linha de uma tabela
nome_var nome_tabela%Rowtype
- Variável previamente declarada
nome_var nome_var_predeclarada%Type
- Variável do tipo cursor
nome_var nome_cursor%Rowtype
- ...

Comandos

- Atribuição
 - IF ... THEN
 - CASE
 - FOR ... LOOP
 - WHILE ... LOOP
 - LOOP
 - EXIT
- } comandos condicionais
- } comandos de repetição

Atribuição

- Atribui um valor a uma variável
- Opções
 - operador de atribuição `:=`
 - a partir de um comando `SELECT ... INTO`
- Exemplo

```
SELECT nome_aluno, sexo_aluno  
INTO nome, sexo  
FROM aluno  
WHERE NUSP = 10;
```

atribuição
com comando
SELECT

```
nome := 'Cristina';
```

atribuição direta

SELECT ... INTO

- Dentro da área de comandos
 - cláusula INTO é obrigatória, retornando um erro caso não seja especificada
 - Deve retornar apenas uma linha
 - 0 linhas → erro `no_data_found`
 - várias linhas → erro `too_many_rows`
- Tratamento de 0 a várias linhas retornadas: CURSORES

Exemplo

```
set serveroutput on;
```

```
DECLARE
```

```
    nome aluno.nome_aluno%Type;
```

```
    sexo aluno.sexo_aluno%Type;
```

```
BEGIN
```

```
    SELECT nome_aluno, sexo_aluno
```

```
        INTO nome, sexo
```

```
        FROM aluno
```

```
        WHERE NUSP = 10;
```

```
    dbms_output.put_line ('Nome do aluno: ' || nome);
```

```
    dbms_output.put_line ('Sexo do aluno: ' || sexo);
```

```
END;
```

IF ... THEN

- Executa um conjunto de ações de acordo com uma ou mais condições

```
IF condição
  THEN relação_de_comandos1
[ELSIF condição
  THEN relação_de_comandos2]
[ELSE
  relação_de_comandos3]
END IF;
```

Exemplo

```
set serveroutput on;
DECLARE
  nome aluno.nome_aluno%Type;
  sexo aluno.sexo_aluno%Type;
BEGIN
  SELECT nome_aluno, sexo_aluno
     INTO nome, sexo
     FROM aluno WHERE NUSP = 10;
  dbms_output.put_line ('Nome do aluno: ' || nome );
  IF sexo = 'f'
     THEN dbms_output.put_line ('Sexo do aluno: feminino' );
     ELSE dbms_output.put_line ('Sexo do aluno: masculino' );
  END IF;
END;
```

CASE

- Retorna um determinado resultado de acordo com o valor do seletor (variável de comparação)

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

Exemplo

```
set serveroutput on;
```

```
DECLARE
```

```
    nome aluno.nome_aluno%Type;
```

```
    sexo aluno.sexo_aluno%Type;
```

```
BEGIN
```

```
    SELECT nome_aluno, sexo_aluno
```

```
        INTO nome, sexo
```

```
        FROM aluno WHERE NUSP = 10;
```

```
    dbms_output.put_line ('Nome do aluno: ' || nome );
```

```
    CASE sexo
```

```
        | WHEN 'f' THEN dbms_output.put_line ('Sexo feminino' );
```

```
        | WHEN 'm' THEN dbms_output.put_line ('Sexo masculino' );
```

```
    END CASE;
```

```
END;
```

FOR ... LOOP

- Executa a relação de comandos várias vezes

```
FOR contador IN [REVERSE] valor inicial..valor final  
LOOP  
    relação de comandos;  
END LOOP;
```

- Variável contador
 - não deve ser declarada previamente
 - deixa de existir após o comando

Exemplo

```
set serveroutput on;
```

```
BEGIN
```

```
  FOR contador IN 10..12
```

```
  LOOP dbms_output.put_line ('Teste: ' ||  
                               contador);
```

```
  END LOOP;
```

```
END;
```

apenas intervalo
com valor
igual a 1

WHILE ... LOOP

- Executa a relação de comandos enquanto a condição for verdadeira

```
WHILE condição  
LOOP  
    relação de comandos;  
END LOOP;
```

Exemplo

```
set serveroutput on;
```

```
DECLARE
```

```
    contador number(2) := 10;
```

```
BEGIN
```

```
    WHILE contador <= 12
```

```
    | LOOP
```

```
        dbms_output.put_line ('Teste: ' || contador);
```

```
        contador := contador + 1;
```

```
    | END LOOP;
```

```
END;
```

LOOP

- Executa a relação de comandos até que uma condição de saída (EXIT) seja encontrada

```
LOOP
```

```
    relação de comandos;
```

```
    IF condição_de_saída
```

```
        THEN EXIT
```

```
END LOOP;
```

EXIT

- Utilizado para interromper a execução de um comando de repetição

```
EXIT [WHEN condição];
```

Exemplo (1/2)

```
set serveroutput on;
```

```
DECLARE
```

```
    valor_maximo number(2);
```

```
    valor_minimo number(2);
```

```
    NUSP_desejado number(2);
```

```
    NUSP aluno.NUSP%Type;
```

```
    nome aluno.nome_aluno%Type;
```

```
    sexo aluno.sexo_aluno%Type;
```

```
BEGIN
```

```
    SELECT max(NUSP)
```

```
        INTO valor_maximo
```

```
        FROM aluno;
```

```
    SELECT min(NUSP)
```

```
        INTO valor_minimo
```

```
        FROM aluno;
```

Exemplo (2/2)

```
NUSP_desejado := valor_minimo;
LOOP
    SELECT NUSP, nome_aluno, sexo_aluno
        INTO NUSP, nome, sexo
        FROM aluno
        WHERE NUSP = NUSP_desejado;
    dbms_output.put_line ('NUSP : ' || NUSP || ', nome: ' ||
                          nome || ', sexo: ' || sexo);
    NUSP_desejado := NUSP_desejado + 1;
    EXIT WHEN NUSP_desejado > valor_maximo;
END LOOP;
END;
```

Cursors

- Funcionalidade
 - áreas compostas de linhas e colunas em memória principal
 - armazenam o resultado de uma seleção que retorna 0 ou mais linhas
- Tipos
 - cursores explícitos
 - cursores implícitos

Cursores Explícitos

- Declaração

```
CURSOR nome_do_cursor (relação_parâmetros)  
  IS SELECT ... FROM ... WHERE ...  
  [FOR UPDATE OF colunas]
```

- Relação de parâmetros

nome tipo_dado {:= / DEFAULT} valor inicial

- parâmetros são sempre de entrada
- não é possível especificar NOT NULL
- não é possível especificar o tamanho

Cursores Explícitos

- Parâmetros
 - permitem a definição de itens dinâmicos de comparação na cláusula WHERE
 - devem ser passados quando o cursor for aberto (comandos OPEN ou FOR)
- Cláusula FOR UPDATE
 - utilizada quando dados forem atualizados
 - especifica um bloqueio que será efetuado nesse momento e somente será liberado quando o cursor for fechado (CLOSE)

Manipulação: Cursores Explícitos

- OPEN
 - abre o cursor
- FETCH
 - disponibiliza a linha corrente e posiciona na próxima linha do cursor
- CLOSE
 - fecha o cursor
- FOR
 - controla de modo completo o acesso ao cursor, substituindo os comandos anteriores

OPEN

- Funcionalidade
 - abre o cursor
 - posiciona o ponteiro do cursor para o primeiro registro resultante da consulta
- Característica
 - o BD não será acessado novamente, pois os registros estão em memória principal
- Comando

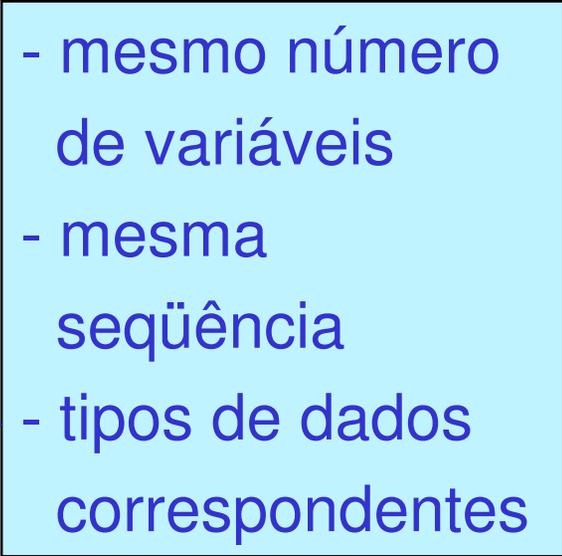
`OPEN nome_do_cursor (relação_parâmetros)`

FETCH

- Funcionalidade
 - transfere o conteúdo do registro corrente para as variáveis correspondentes
 - posiciona o ponteiro do cursor no próximo registro resultante da consulta

- Comando

```
FETCH nome_do_cursor  
    INTO lista_de_variáveis
```



- mesmo número de variáveis
- mesma seqüência
- tipos de dados correspondentes

CLOSE

- Funcionalidade
 - fecha o cursor
 - libera a área de memória principal ocupada pelo resultado da consulta
- Comando

`CLOSE nome_do_cursor`

Atributos de Cursores

- nome_do_cursor%FOUND
 - TRUE: FETCH retorna alguma linha
 - caso contrário, FALSE
 - NULL se nenhum FETCH tiver sido executado
- nome_do_cursor%NOTFOUND
 - FALSE: FETCH retorna alguma linha
 - caso contrário, TRUE
 - NULL se nenhum FETCH tiver sido executado

Atributos de Cursores

- `nome_do_cursor%ROWCOUNT`
 - retorna o número de linhas já processadas pelo cursor
 - 0 se nenhum `FETCH` tiver sido executado
- `nome_do_cursor%ISOPEN`
 - `TRUE`: cursor está aberto
 - caso contrário, `FALSE`

Estrutura Básica

```
DECLARE
```

```
  CURSOR nome_do_cursor
```

```
    IS comando SELECT do cursor;
```

```
  nome_do_registro nome_do_cursor%RowType;
```

```
BEGIN
```

```
  OPEN nome_do_cursor;
```

```
  LOOP
```

```
    FETCH nome_do_cursor
```

```
      INTO nome_do_registro;
```

```
    EXIT WHEN nome_do_cursor%NOTFOUND;
```

```
    processos sobre a linha atual do cursor;
```

```
  END LOOP;
```

```
  CLOSE nome_do_cursor
```

```
END;
```

Exemplo

```
set serveroutput on;
DECLARE
  CURSOR cursor_aluno IS
    SELECT NUSP, nome_aluno, sexo_aluno FROM aluno;
  variavel_cursor cursor_aluno%Rowtype;
BEGIN
  OPEN cursor_aluno;
  LOOP
    FETCH cursor_aluno INTO variavel_cursor;
    EXIT WHEN cursor_aluno%NotFound;
    dbms_output.put_line ('NUSP: ' || variavel_cursor.NUSP ||
      ', nome: ' || variavel_cursor.nome_aluno || ', sexo: ' ||
      variavel_cursor.sexo_aluno);
  END LOOP;
  CLOSE cursor_aluno;
END;
```

Exemplo

```
set serveroutput on;
DECLARE
  CURSOR cursor_aluno (codigo_desejado number) IS
    SELECT NUSP, nome_aluno, sexo_aluno
    FROM aluno WHERE NUSP = codigo_desejado;
  variavel_cursor cursor_aluno%Rowtype;
BEGIN
  OPEN cursor_aluno (2);
  FETCH cursor_aluno INTO variavel_cursor;
  IF cursor_aluno%NotFound
  THEN dbms_output.put_line ('Tupla inexistente.');
```

```
  ELSE dbms_output.put_line ('NUSP: ' || variavel_cursor.NUSP ||
    ', nome: ' || variavel_cursor.nome_aluno || ', sexo: ' ||
    variavel_cursor.sexo_aluno);
  END IF;
  CLOSE cursor_aluno;
END;
```

Cursors Implícitos

- Criados implicitamente pelo Oracle®
 - SELECT (retornando somente uma linha)
 - INSERT, UPDATE, DELETE
- Características
 - não podem ser manipulados por OPEN, FETCH, CLOSE
 - podem ter seus quatro atributos verificados com o significado e o retorno
 - SQL%FOUND SQL%NOTFOUND
 - SQL%ROWCOUNT SQL%ISOPEN