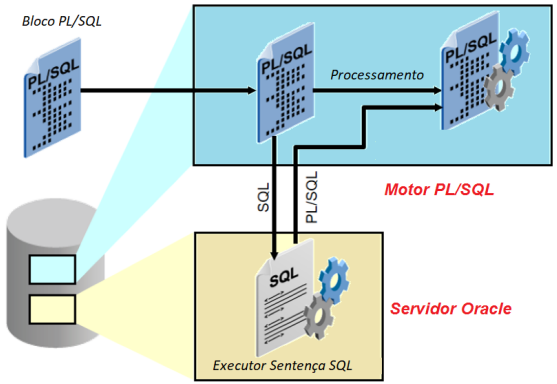
BANCO DE DADOS
PL/SQL: Introdução

Ma. Simone Maria Viana Romano

INTRODUÇÃO

- PL/SQL: *Procedure Language SQL (extensão);*
- Linguagem de programação para bancos de dados relacionais;
- Uma linguagem de programação proprietária da Oracle;
- Combina lógica de programa e fluxo de controle com SQL;
- Linguagem de programação 3ª Geração: Convertido
- Linguagem de máquina por um compilador; mais amigável.
- Linguagens: Visual Basic, C, C++, COBOL, FORTRAN, Java, Pascal, PL/SQL.

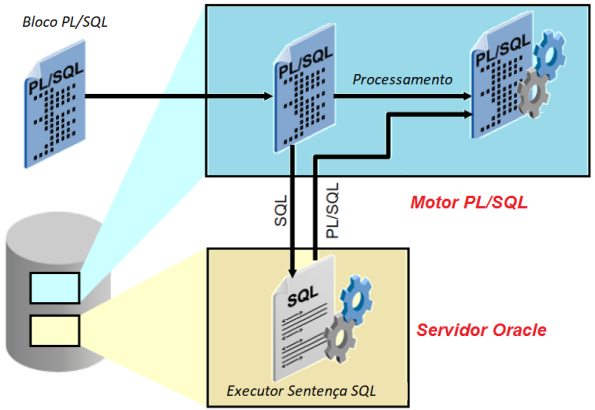
ARQUITETURA DA EXECUÇÃO DO PL/SQL



INTRODUÇÃO

- É uma linguagem altamente estruturada, legível e acessível;
- É uma linguagem padrão e portátil para desenvolvimento Oracle;
- É uma linguagem embarcada e funciona com SQL;
- É uma linguagem de banco de dados de alto desempenho e altamente integrada;
- É baseado na linguagem de programação Ada e tem muitos semelhanças na sintaxe:

ARQUITETURA DA EXECUÇÃO DO PL/SQL

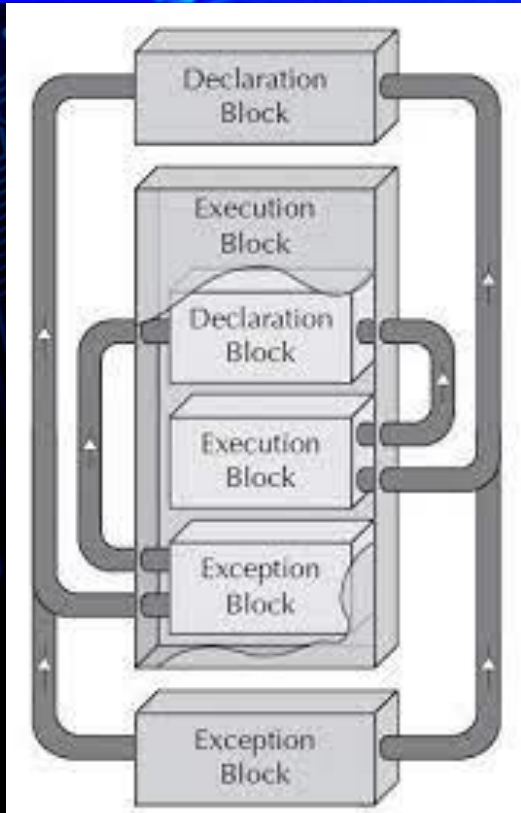


PL/SQL: Exemplo

- Permite escrever uma declaração para promover os representantes comerciais, os representantes de marketing e escriturários:

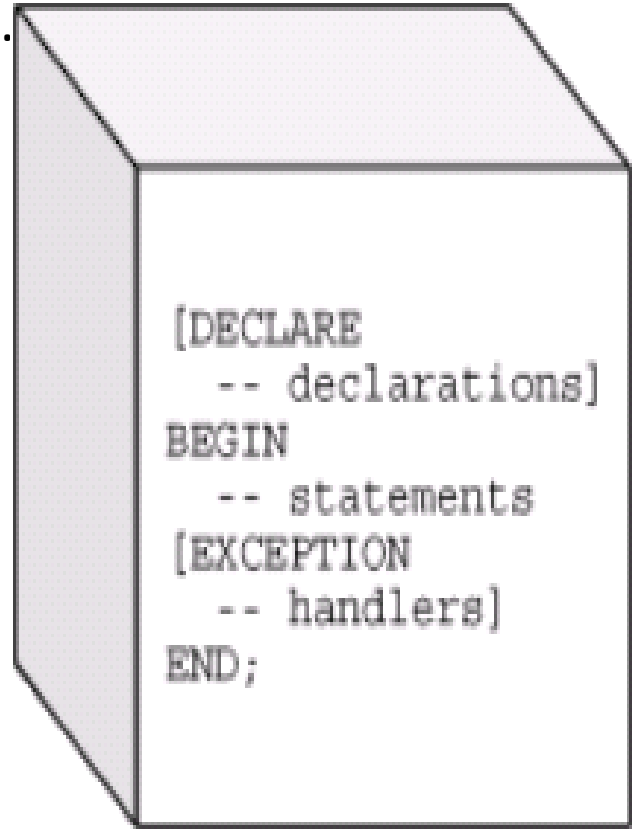
```
DECLARE
CURSOR c_employees IS SELECT * FROM employees;
BEGIN
FOR c_emp in c_employees
LOOP
IF c_emp.job_id = 'SA_REP' AND c_emp.hire_date <= '05-Feb-2005' THEN
UPDATE employees
SET job_id = 'SR_SA_REP' WHERE employee_id = c_emp.employee_id;
ELSIF c_emp.job_id = 'MK_REP' AND c_emp.hire_date <= '05-Feb-2005' THEN
UPDATE employees
SET job_id = 'SR_MK_REP' WHERE employee_id = c_emp.employee_id;
ELSIF c_emp.job_id = 'ST_CLERK' AND c_emp.hire_date <= '05-Feb-2005’ THEN
UPDATE employees
SET job_id = 'SR_ST_CLRK' WHERE employee_id = c_emp.employee_id;
END IF;
END LOOP;
END;
```

BLOCOS ANÔNIMOS



PL/SQL: Blocos Anônimos

- Estrutura básica formada por três partes:



DECLARE

- Seção opcional, em que todos os objetos são declarados.

BEGIN

- Em que os comandos PL/SQL são colocados. Pode conter outros blocos.

EXCEPTION

- Seção opcional, onde os erros são tratados. (pode conter outros blocos)

END

- Finaliza o bloco

Blocos Anônimos:

DECLARAR CONSTANTES E IDENTIFIFICADORES

CONSTANTES

- Especificar com a palavra reservada CONSTANT;
- Atribuir um valor através dos dois pontos e igual (:=);
- Definir um valor com a palavra reservada DEFAULT.

IDENTIFICADOR

- Nomes atribuídos aos elementos do PL/SQL;
- Exemplo: tabelas, cursores ou variáveis.
- Podem ter até 30 caracteres de comprimento, devem começar com uma letra, e podem ter qualquer combinação de letras, números, $, #, _ .

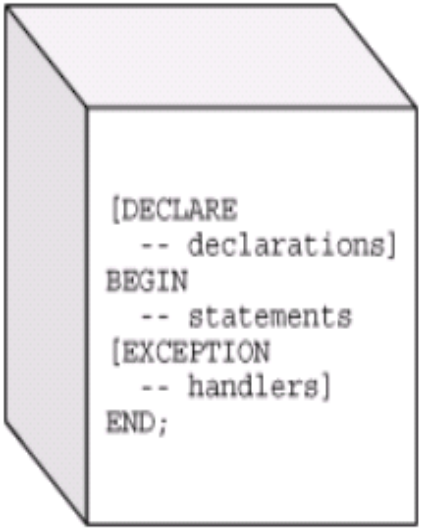
Blocos Anônimos:

%TYPE: Declaração por Referência

Variáveis ancoradas referem-se ao uso deste atributo.

O PL/SQL ancora o tipo de dado de uma variável a uma estrutura de dados, geralmente de uma coluna de uma tabela.

Sintaxe: VARIAVEL TABELA.COLUNA_TABELA**%TYPE**;

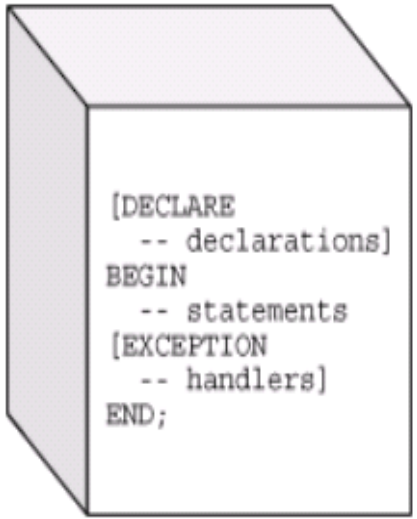


Blocos Anônimos:
DBMS_OUTPUT.PUT_LINE

Um procedimento de pacote fornecido pela Oracle.

Uma alternativa para exibir dados a partir de um bloco PL/SQL.

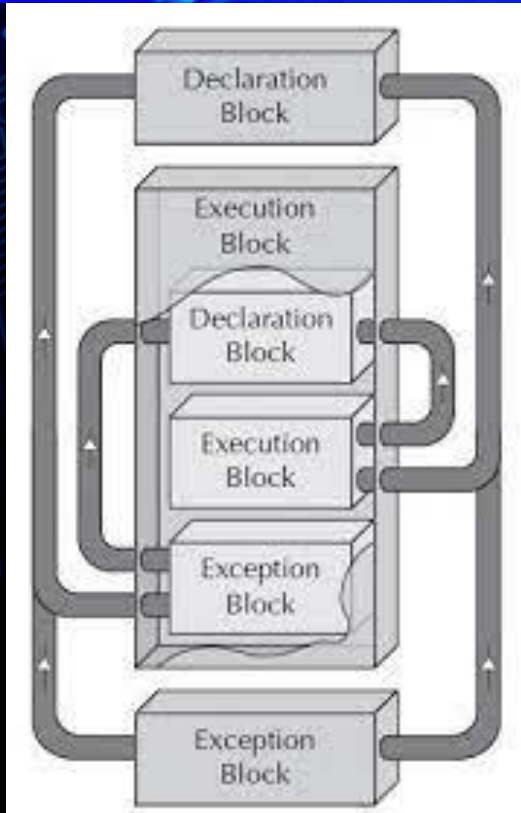
Precisa ser ativado em através do comando: **SET SERVEROUTPUT ON** antes de utilizar o pacote DBMS.



Blocos Anônimos: OPERADORES

OPERADOR	*DESCRIÇÃO*
**, ^, NOT	Exponenciação
NOT	Negação
:=	Atribuição
+,-,*,/	Adição, subtração, multiplicação e divisão
||	Concatenação
=, !=,<,>,>=,<=, IS NULL, LIKE, BETWEEN, IN	Comparação
AND	Conjunção
OR	Inclusão
@	Acesso remoto
;	Finalizar instrução
/	Executar bloco

Instruções SQL



Blocos Anônimos: SELECT...INTO...

Exibe dados do banco de dados atribuindo colunas que DEVEM ser especificadas para cada variável declarada.

Sintaxe:

```
SELECT coluna, coluna2,.., colunan
    INTO {nome_variavel | nome_registro, variavel2,..,variaveln}
      FROM tabela WHERE condição;
```

Obs. Estas variáveis utilizadas na cláusula INTO deverão estar declaradas (DECLARE ou IS)

Blocos Anônimos: COMANDOS DML

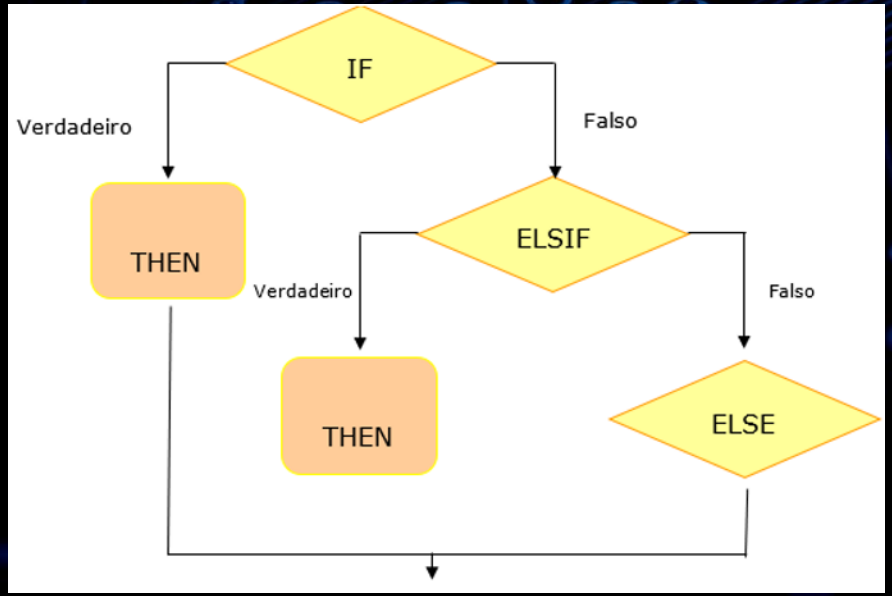
Os comandos de manipulação de dados são iguais aos que já vimos anteriormente:

- **INSERT** (insere dados);
- **UPDATE** (atualiza dados);
- **DELETE** (apaga dados).

Obs. Estas variáveis utilizadas na cláusula INTO deverão estar declaradas (DECLARE ou IS)

Exercícios

Estruturas de Controle



Estrutura Condicional

Há três tipos de instrução IF:
IF-THEN-END IF;
IF-THEN-ELSE-END IF;
IF-THEN-ELSIF-END IF.

Sintaxe:
IF condição THEN Instrução;
[ELSIF condição THEN instrução;]
[ELSE instrução;]
END IF;

```
    FETCH CR BULK COLLECT INTO emp_table LIMIT 200;

    FOR i in 1..emp_table.count()
    LOOP
      IF emp_table(i).JOB_ID = 'AD_PRES' THEN
        UPDATE  HR.EMP
        SET     SALARY = SALARY * 2
        WHERE   EMPLOYEE_ID = emp_table(i).EMPLOYEE_ID;
      ELSIF emp_table(i).JOB_ID = 'AD_VP' THEN
        UPDATE  HR.EMP
        SET     SALARY = SALARY * 1.5
        WHERE   EMPLOYEE_ID = emp_table(i).EMPLOYEE_ID;
      ELSIF emp_table(i).JOB_ID = 'IT_PROG' THEN
        UPDATE  HR.EMP
        SET     SALARY = SALARY * 1.25
        WHERE   EMPLOYEE_ID = emp_table(i).EMPLOYEE_ID;
      ELSIF emp_table(i).JOB_ID = 'SA_REP' THEN
        UPDATE  HR.EMP
        SET     SALARY = SALARY * 1.2
        WHERE   EMPLOYEE_ID = emp_table(i).EMPLOYEE_ID;
      ELSIF emp_table(i).JOB_ID = 'AC_MGR' THEN
        UPDATE  HR.EMP
        SET     SALARY = SALARY * 1.15
        WHERE   EMPLOYEE_ID = emp_table(i).EMPLOYEE_ID;
      END IF;
    END LOOP;

    EXIT WHEN CR%NOTFOUND;
END LOOP;

CLOSE CR;
```

Estruturas de Repetição (Laços)

Os loops repetem uma instrução ou sequência de instruções várias vezes. Existem três tipos de loop: Básico, FOR e WHILE, onde:

- *Básico ou Simples* → fornece ações repetitivas sem condições gerais;
- *For → fornece controle iterativo para ações com base em uma contagem;*
- *While* → fornece controle iterativo para ações com base em uma condição.

Estruturas de Repetição: LOOP SIMPLES

- Permite a execução de sua instrução pelo menos uma vez, mesmo que a condição já esteja atendida no momento em que o loop foi informado;
- Sem a instrução EXIT WHEN, o loop seria infinito.

```
LOOP
    Comandos;
 EXIT WHEN condição;
```

Estruturas de Repetição: FOR

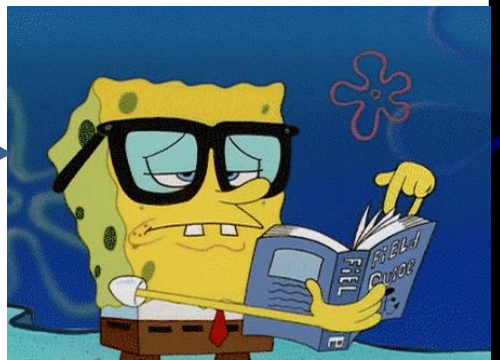
- Mesma estrutura de um loop básico;
- Controle no início da palavra chave LOOP para determinar o numero de iterações que o PL/SQL executa.

FOR contador IN [REVERSE] início.. fim LOOP

comandos;

END LOOP;

REVERSE faz o contador decrescer a cada iteração a partir do limite superior até o limite inferior.



Estruturas de Repetição: WHILE

- A condição é avaliada ao início de cada iteração;
- O loop terminará quando a condição for falsa. Se a condição for falsa no início do loop, nenhuma iteração futura será executada.

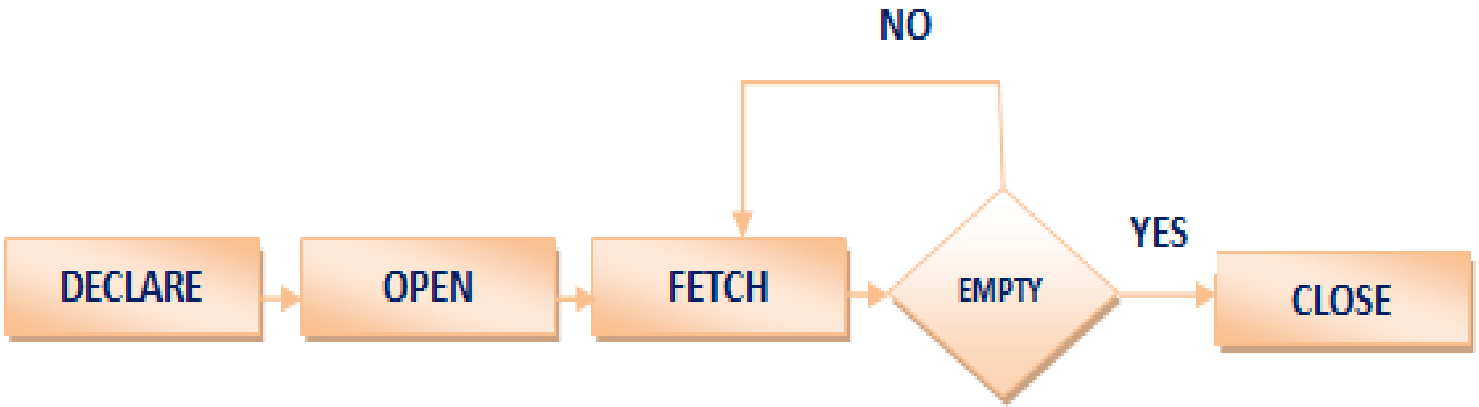
```
LOOP
  Comandos;
  EXIT WHEN condição;
END LOOP;
```

Exercícios

CURSORES EXPLÍCITOS

CURSOR EXPLÍCITO: Introdução

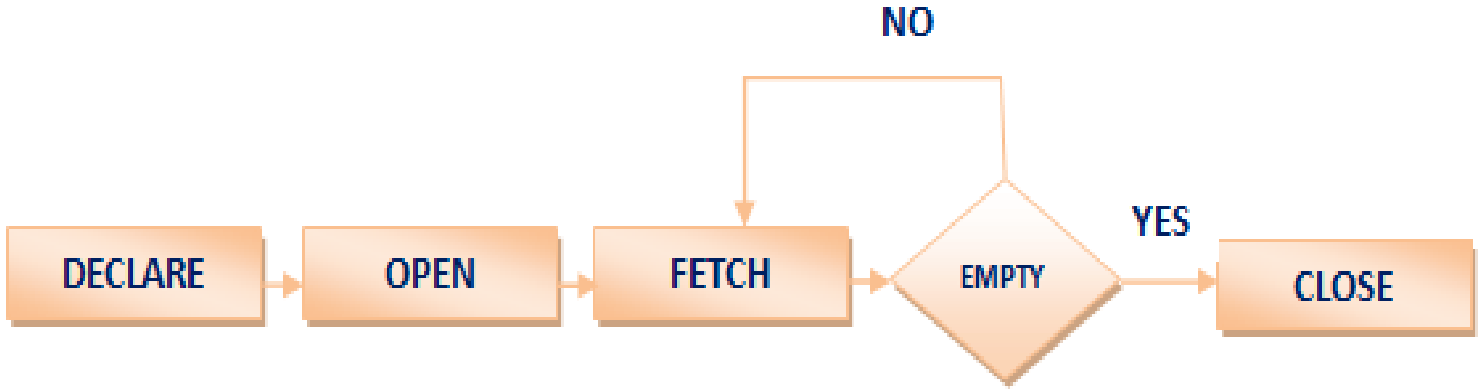
O cursor do PL/SQL: nomear uma área SQL particular e acessar suas informações armazenadas.
O cursor orienta todas as fases do processamento.
Os cursores podem ser de dois tipos: **IMPLÍCITO** e **EXPLÍCITO** consultas que retornam mais de uma linha. Os cursores explícitos são declarados e nomeados pelo programador e manipulados através de instruções específicas nas ações executáveis do bloco.



CURSOR EXPLÍCITO: Introdução

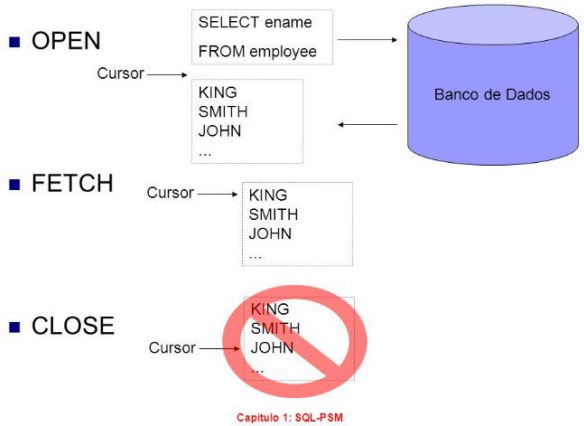
Usado para processar individualmente cada linha (processa uma linha por vez) retorna por uma instrução SQL de várias linhas;
CONJUNTO ATIVO: conjunto de linhas retornado por uma consulta de várias linhas.
Pode ser usado para:

- Além da primeira linha retornada pela consulta, linha por linha;
- Controla que linha está sendo processada no momento;
- Permite que o programador controle as linhas manualmente no bloco PL/SQL.



CURSOR EXPLÍCITO: Passos

1. Declare o cursor nomeando-o e definindo a estrutura da consulta a ser executada dentro dele (DECLARE);
2. Abra o cursor. A instrução OPEN executa a consulta e vincula as variáveis que estiverem referenciadas. As linhas identificadas pela consulta são chamadas de CONJUNTO ATIVO e estão agora disponíveis para extração;
3. Extraia dados do cursor, comando FETH, que busca as linhas do cursor. Após cada extração, testamos o cursor para qualquer linha existente. Se não existirem mais linhas para serem processadas, precisamos fechar o cursor.
4. Feche o cursor. A instrução CLOSE libera o conjunto ativo de linhas. É possível reabrir o cursor e estabelecer um novo conjunto ativo.



CURSOR EXPLÍCITO: Declarar o Cursor

OBJETIVO: Declarar o cursor explícito contendo a consulta SEM INTO;

Sintaxe: **DECLARE**

nome_cursor IS consulta;

```
DECLARE
  CURSOR c1 IS
        SELECT empno, ename, job, sal
        FROM   emp
        WHERE sal > 2000;
BEGIN
...
END;
```

CURSOR EXPLÍCITO: Abrir o Cursor

- OBJETIVO: Abre o cursor para executar uma consulta e identificar o conjunto ativo;
- Sintaxe: OPEN nome_cursor;

```
declare
cursor cur_emp
is
 select ename,sal from emp where deptno=&deptnumb;
vnm emp.ename%type;
vsal emp.sal%type;
begin
  open cur_emp;
 loop
   fetch cur_emp into vnm,vsal;
   exit when cur_emp%notfound;
dbms_output.put_line(vnm||'-'||vsal);
dbms_output.put_line(cur_emp%rowcount);
 end loop;
 close cur_emp;
end;
/
```

CURSOR EXPLÍCITO: Iniciar Laço

- OBJETIVO: Usado para retornar uma ou mais linhas do cursor aberto.
- Sintaxe: LOOP... EXIT WHEN... END LOOP.

```
declare
cursor cur_emp
is
 select ename,sal from emp where deptno=&deptnumb;
vnm emp.ename%type;
vsal emp.sal%type;
begin
  open cur_emp;
 loop
   fetch cur_emp into vnm,vsal;
   exit when cur_emp%notfound;
dbms_output.put_line(vnm||'-'||vsal);
dbms_output.put_line(cur_emp%rowcount);
 end loop;
 close cur_emp;
end;
/
```

CURSOR EXPLÍCITO: Extrair os Dados

- OBJETIVO: Usado para retornar uma ou mais linhas do cursor aberto nas variáveis de saídas declaradas no DECLARE.
- Sintaxe:

FETCH nome_cursor INTO [variável1, variável2,...|nome_registro];

- Observações:
- Inclua o mesmo número de variáveis na cláusula INTO para cada coluna da instrução SELECT e certifique-se de que os tipos de dados são compatíveis;
- Exemplo: **FETCH emp_cursor INTO v_id, v_nome;**

CURSOR EXPLÍCITO: Atributos

- OBJETIVO: Condição de execução para manipular os dados através do cursor.
- Sintaxe: **EXIT WHEN nome_cursor%atributo_cursor;**

Atributo	Tipo	Descrição
%ISOPEN	Booleano	Validado para TRUE se o cursor estiver aberto
%NOTFOUND	Booleano	Validado para TRUE se a extração mais recebendo não retornar uma linha
%FOUND	Booleano	Validado para TRUE se a extração mais recente não retornar uma linha (complemento de %NOTFOUND)
%ROWCOUNT	Número	Validado para o número total de linhas retornadas até o momento.

Exercícios

EXCEPTIONS


1. DECLARE
2. Sample_exception EXCEPTION;
3. PROCEDURE nested_block
4. IS
5. BEGIN
6. Dbms_output.put_line('Inside nested block');
7. Dbms_output.put_line('Raising sample_exception from nested block');
8. RAISE sample_exception;
9. EXCEPTION
10. WHEN sample_exception THEN
11. Dbms_output.put_line ('Exception captured in nested block. Raising to main block');
12. RAISE;
13. END;
14. BEGIN
15. Dbms_output.put_line('Inside main block');
16. Dbms_output.put_line('Calling nested block');
17. Nested_block;
18. EXCEPTION
19. WHEN sample_exception THEN
20. Dbms_output.put_line ('Exception captured in main block');
21. END;
22. /
Declaring user defined exception
Raising the exceptions
Propagating the exception from child to parent block.
Exception from nested block, propagated to parent block, and handled in parent block exceptions

EXCEPTIONS: Introdução

- Objetivo: cria uma exceção, mas especifica um *handler* de exceções para executar ações finais. Podemos criar uma exceção utilizando dois métodos:
- Ocorre um erro do Oracle e a exceção associada é criada automaticamente;
- Crie uma exceção explicitamente emitindo a instrução RAISE dentro do bloco.
- Sintaxe:

```
EXCEPTION
WHEN exceção1 [OR exceção2..] THEN
        Instrução1;
        Instrução2;
WHEN OTHERS THEN
        Instrução1;
        Instrução2;
....
```

EXCEÇÃO	DESCRIÇÃO	ORIENTAÇÃO PARA TRATAMENTO
Erro predefinido pelo Oracle Server	Um dos cerca de 20 erros que ocorrem com mais frequência no código PL/SQL.	Não declare e permita que o Oracle Server crie as exceções implicitamente.
Erro não predefinido pelo Oracle Server.	Qualquer outro erro padrão do Oracle Server.	Declare dentro da seção declarativa e permita que o Oracle Server crie as exceções de forma implícita.
Erro definido pelo usuário.	Uma condição que o desenvolvedor determina seja anormal.	Declare dentro da seção declarativa e crie exceções de forma explícita.

....

Obs.: *OTHERS é uma cláusula de manipulação de exceção opcional que captura exceções não especificadas.*

EXCEPTIONS: Introdução

- Objetivo: cria uma exceção, mas especifica um *handler* de exceções para executar ações finais. Podemos criar uma exceção utilizando dois métodos:
- Ocorre um erro do Oracle e a exceção associada é criada automaticamente;
- Crie uma exceção explicitamente emitindo a instrução RAISE dentro do bloco;
- *OTHERS é uma cláusula de manipulação de exceção opcional que captura exceções não especificadas.* Sintaxe:

```
EXCEPTION
WHEN exceção1 [OR exceção2..] THEN
        Instrução1;
        Instrução2;
WHEN OTHERS THEN
        Instrução1;
        Instrução2;
....
```

Neste link há todos os códigos de erros Oracle
http://www.dba-oracle.com/t_error_code_list.htm

....

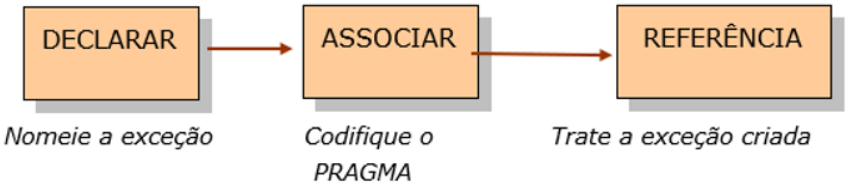
EXCEPTIONS: Erros Definidos

NOME DA EXCEÇÃO	ERRO	DESCRIÇÃO
ACCESS_INTO_NULL	ORA-06530	Tentativa de atribuir valores aos atributos de um objeto não inicializado.
CASE_NOT_FOUND	ORA-06592	Nenhuma das cláusulas WHEN de uma instrução CASE foi selecionada e não há nenhuma cláusula ELSE padrão.
COLLECTION_IS_NULL	ORA-06531	Tentativa de aplicação de métodos de conjunto diferentes de EXISTS para um varray ou tabela aninhada não inicializada.
CURSOR_ALREADY_OPEN	ORA-06511	Tentativa de abertura de um cursor já aberto.
DUP_VAL_ON_INDEX	ORA-00001	Tentativa de inserção de um valor duplicado.
INVALID_CURSOR	ORA-01001	Ocorreu operação ilegal de cursor.
INVALID_NUMBER	ORA-01722	Falha de conversão de string de caracteres para número
LOGIN_DENIED	ORA-01017	Estabelecendo login com o Oracle com um nome de usuário ou senha inválida.
TIMEOUT_ON_RESOURCE	ORA-00051	Ocorreu um timeout enquanto o Oracle está aguardando por um recurso.
TOO_MANY_ROWS	ORA-01422	SELECT de uma única linha retornou mais de uma linha.
VALUE_ERROR	ORA-06502	Ocorreu erro aritmético, de conversão, truncamento ou restrição de tamanho. Quando ocorre em uma instrução SQL (ao invés de PL/SQL) a mensagem é INVALID_NUMBER.
ZERO_DIVIDE	ORA-01476	Tentativa de divisão por zero.

NOME DA EXCEÇÃO	ERRO	DESCRIÇÃO
NO_DATA_FOUND	ORA-01403	SELECT de linha única não retornou dados.
NOT_LOGGED_ON	ORA-01012	O programa PL/SQL emite uma chamada de banco de dados sem estar conectado ao Oracle
PROGRAM_ERROR	ORA-06501	O código PL/SQL tem um problema interno.
ROWTYPE_MISMATCH	ORA-06504	Variável de cursor de host e variável de cursor PL/SQL envolvida em um atributo têm tipos de retorno incompatíveis.
SELF_IS_NULL	ORA-30625	Tentativa de chamar um método MEMBER em um objeto nulo. Isto é, o parâmetro interno SELF (sempre primeiro membro passado para o MEMBER) é nulo.
STORAGE_ERROR	ORA-06500	O PL/SQL esgotou a memória ou a memória está corrompida.
SUBSCRIPT_BEYOND_COUNT	ORA-06533	Feita referência a um elemento de varray ou tabela aninhada usando um número de índice fora da faixa legal (-1, por exemplo)
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	Tentativa de referenciar um elemento de tabela aninhada ou varray usando um número de índice que está fora do intervalo válido (exemplo:-1)
SYS_INVALID_ROWID	ORA-01410	Conversão de uma string de caracteres em um rowid universal falhou porque a string não representa um rowid válido.

EXCEPTIONS: PRAGMA EXCEPTION_INIT

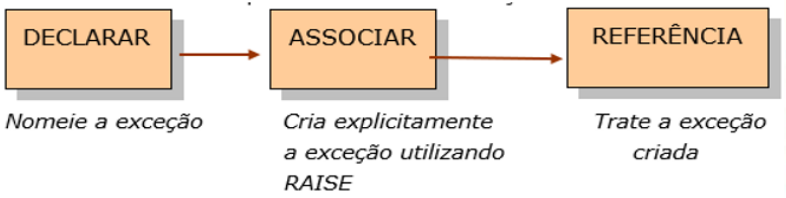
- OBJETIVO: Capturar erros não predefinidos do servidor Oracle;
- Sintaxe: **PRAGMA EXCEPTION_INIT** (erro_declarado, -erro Oracle)
- Passos para capturar uma exceção não predefinida:
  1. Declare o nome para a exceção na seção declarativa;
  2. Associe a exceção declarada ao número de erro padrão do Oracle Server usando a instrução **PRAGMA EXCEPTION_INIT;**
  3. Faça referência à exceção declarada dentro da rotina de tratamento de exceção correspondente.




```
DECLARE
    e_Faculty_Remaining EXCEPTION;
    PRAGMA EXCEPTION_INIT (e_Faculty_Remaining, -6501);
    V_deptNo  dept.deptNo%TYPE := &p_deptno;

    BEGIN
        DELETE
        FROM    dept
        WHERE  deptNo = V_deptNo;
        COMMIT;
    EXCEPTION
        WHEN e_faculty_Remaining THEN
            DBMS_OUTPUT.PUT_LINE('can not remove dept, there are
              faculty in the dept');
    END;
```

EXCEPTIONS: RAISE



OBJETIVO:
Podemos definir suas próprias exceções que são definidas pelo usuário.

PASSOS: Declaradas na seção de declaração de um bloco;
Criadas explicitamente com instrução RAISE.

SINTAXE:

```
DECLARE
        ERRO      EXCEPTION;
BEGIN
        INSTRUÇÃO
        IF SQL%NOTFOUND THEN
        RAISE ERRO;
        END IF;
END;
```

EXCEPTIONS: RAISE_APPLICATION_ERROR

OBJETIVOS:
Comunicar uma exceção predefinida interativamente retornando um código ou uma mensagem de erro não padronizada;
Relatar erros para a aplicação e evitar o retorno de exceções não tratáveis.
OBSERVAÇÕES: Número de erro: número especificado pelo usuário para a exceção entre -20000 e -20999.
SINTAXE:

```
EXCEPTION
        WHEN NO_DATA_FOUND THEN
                RAISE_APPLICATION_ERROR(-ERRO, 'MENSAGEM');
END;
```

Exercícios

SAIBA MAIS

- https://blogs.oracle.com/connect/post/building-with-blocks