

Arquitetura e Administração de Bases de Dados
Oracle10g PL/SQL Programming

João Costa
jcosta@isec.pt

Agenda

- ▶ Introdução ao PL/SQL
- ▶ Conceitos básicos de PL/SQL
- ▶ Estrutura dos blocos
- ▶ Manipular dados em blocos PL/SQL

Modelos de aplicação

- ▶ Três componentes principais
 - ▶ Interface com o utilizador
 - ▶ Lógica do programa (*brains behind the screens*)
 - ▶ Base de dados
- ▶ A maioria dos modelos é baseada em
 - ▶ Estrutura *two-tier*
 - ▶ Estrutura *three-tier*

▶ 3

2021/22 - LEI - AABD - PL/SQL

Aplicações *Two-tier*

- ▶ Normalmente chamado cliente/servidor
- ▶ Processamento repartido entre cliente e servidor
- ▶ *Named or stored program units* são blocos de código PL/SQL armazenados na BD Oracle para efetuarem o processamento do lado do servidor

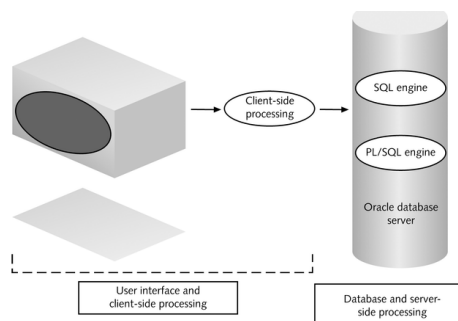


Figure 1-2 The client/server application model

▶ 4

2021/22 - LEI - AABD - PL/SQL

Aplicações *Three-tier*

- ▶ *Thin client* sem código carregado na máquina cliente
 - ▶ ex. acesso via *browser*
- ▶ *Middle tier* é um servidor de aplicações.
 - ▶ Ex. *Forms server for Oracle*
- ▶ O outro *tier* é o servidor de BD
- ▶ Processamento no *middle tier* e no servidor
- ▶ Manutenção simplificada

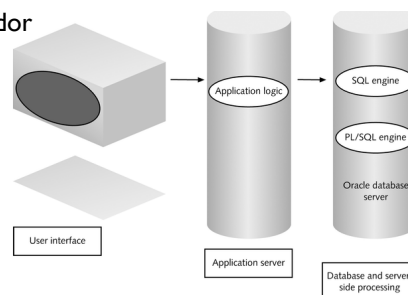


Figure 1-3 Three-tier application model

▶ 5

2021/22 - LEI - AABD - PL/SQL

A Linguagem PL/SQL

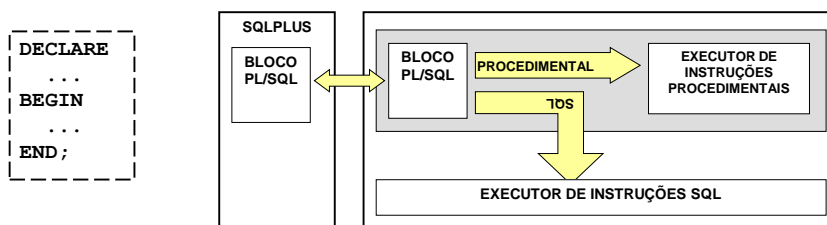
- ▶ PL/SQL é uma linguagem procedimental proprietária da Oracle
 - ▶ Ao contrário do SQL
- ▶ Perfeitamente integrada com SQL
- ▶ Pode melhorar o desempenho
 - ▶ Ao agrupar os comandos em blocos
- ▶ Portável para qualquer plataforma Oracle
- ▶ Usada em muitas ferramentas Oracle
- ▶ Armazenamento dos programas no servidor
 - ▶ Aumenta a segurança

▶ 6

2021/22 - LEI - AABD - PL/SQL

PL/SQL

- ▶ É uma extensão do SQL
- ▶ com características de uma linguagem de programação
- ▶ Nos procedimentos permite incluir instruções de SQL.
- ▶ Programação baseado em blocos



▶ 7

2021/22 - LEI - AABD - PL/SQL

Estrutura de um bloco PL/SQL

- ☐ DECLARE – Opcional
Variáveis, cursores, exceções definidas pelo utilizador
- ☐ BEGIN – Obrigatório
 - Instruções SQL
 - Instruções PL/SQL
- ☐ EXCEPTION – Opcional
Acções a realizar quando ocorre um erro
- ☐ END; – Obrigatório

```

DECLARE
    vvariavel VARCHAR2(5);
BEGIN
    SELECT      coluna
    INTO        vvariavel
    FROM        tabela;
EXCEPTION
    WHEN excepcao THEN
        ...
END;
```

▶ 8

2021/22 - LEI - AABD - PL/SQL

Tipos de blocos

□ Geral

```
[DECLARE]

BEGIN
  --instruções

[EXCEPTION]

END;
```

Procedimento

```
PROCEDURE nome
IS
BEGIN
  --instruções

[EXCEPTION]

END;
```

Função

```
FUNCTION nome
RETURN tipodados
IS
BEGIN
  --instruções
  RETURN valor;

[EXCEPTION]

END;
```

- ▶ Cada unidade é constituída por um ou mais blocos

Variáveis

- ▶ Uso de variáveis para:
 - ▶ Armazenamento temporário de dados
 - ▶ Manipulação de valores armazenados
 - ▶ Reutilização
 - ▶ Facilidade de manutenção
- ▶ Declarar e iniciar variáveis na secção da declarativa
 - ▶ Atribuir novos valores a variáveis na secção de execução.
 - ▶ Passagem de valores através de parâmetros aos blocos PL/SQL
 - ▶ Ver os resultados através de variáveis de saída

Variáveis

- ▶ Escalares
- ▶ Variáveis PL/SQL
- ▶ Compostas (registos e grupos de campos)
- ▶ Variáveis Não PL/SQL : variáveis Bind (host)

▶ 11

2021/22 - LEI - AABD - PL/SQL

Nomes das variáveis

- ▶ Começam por um valor alfanumérico
- ▶ Até 30 caracteres
- ▶ Maiúsculas e minúsculas, números, _ , \$, #

- ▶ Ex:

```
x  
t2  
phone#  
credit_limit  
LastName  
oracle$number
```

▶ 12

2021/22 - LEI - AABD - PL/SQL

Tipos de dados

- ▶ Herdados do SQL
 - ▶ **Caracteres** – CHAR(n)
VARCHAR2(n)
 - ▶ **Numéricos** – NUMBER(p,s)
 - ▶ **Data** – DATE
 - ▶ ...
- ▶ Introduzidos pelo PL/SQL
 - ▶ **Booleanos** – BOOLEAN
 - ▶ ...

▶ 13

2021/22 - LEI - AABD - PL/SQL

Declaração de Variáveis em PL/SQL

Sintaxe

```
identifier [CONSTANT] datatype [NOT NULL] [ := | DEFAULT expr ] ;
```

Nota:

No mínimo, exige-se um nome para a variável e o tipo de dados.

▶ 14

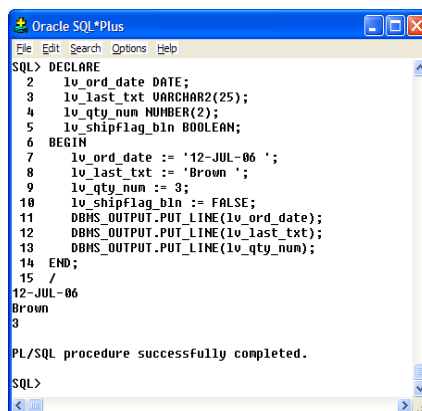
2021/22 - LEI - AABD - PL/SQL

Exemplos de Declarações

```
DECLARE
    lv_ord_date      DATE;
    lv_last_txt      VARCHAR2(25);
    lv_qty_num       NUMBER(2);
    lv_shipflag_bln  BOOLEAN;
BEGIN
    ---- PL/SQL executable
    statements ----
END;
```

Nota:

No mínimo, exige-se um nome para a variável e o tipo de dados.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> DECLARE
2  lv_ord_date DATE;
3  lv_last_txt VARCHAR2(25);
4  lv_qty_num NUMBER(2);
5  lv_shipflag_bln BOOLEAN;
6  BEGIN
7      lv_ord_date := '12-JUL-06';
8      lv_last_txt := 'Brown';
9      lv_qty_num := 3;
10     lv_shipflag_bln := FALSE;
11     DBMS_OUTPUT.PUT_LINE(lv_ord_date);
12     DBMS_OUTPUT.PUT_LINE(lv_last_txt);
13     DBMS_OUTPUT.PUT_LINE(lv_qty_num);
14 END;
15 /
12-JUL-06
Brown
3
PL/SQL procedure successfully completed.
SQL>
```

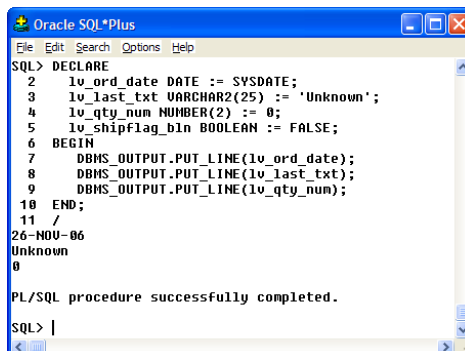
▶ 15

2021/22 - LEI - AABD - PL/SQL

Inicialização de variáveis

```
DECLARE
    lv_ord_date      DATE          := SYSDATE;
    lv_last_txt      VARCHAR2(25) := 'Unknown';
    lv_qty_num       NUMBER(2)    := 0;
    lv_shipflag_bln  BOOLEAN      := 'FALSE';
BEGIN
    ---- PL/SQL executable ----
    ---- statements ----
END;
```

Atribuir um valor à variável quando esta é criada



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> DECLARE
2  lv_ord_date DATE := SYSDATE;
3  lv_last_txt VARCHAR2(25) := 'Unknown';
4  lv_qty_num NUMBER(2) := 0;
5  lv_shipflag_bln BOOLEAN := FALSE;
6  BEGIN
7      DBMS_OUTPUT.PUT_LINE(lv_ord_date);
8      DBMS_OUTPUT.PUT_LINE(lv_last_txt);
9      DBMS_OUTPUT.PUT_LINE(lv_qty_num);
10 END;
11 /
26-NOV-06
Unknown
0
PL/SQL procedure successfully completed.
SQL> |
```

▶ 16

2021/22 - LEI - AABD - PL/SQL

Declaração de Variáveis - Opções

- ▶ **NOT NULL** – a variável tem que ter sempre um valor
- ▶ **CONSTANT** – o valor da variável não pode ser alterado dentro do bloco

Uma variável NOT NULL ou CONSTANT tem que ser inicializada

DECLARE

```
lv_shipcntry_txt VARCHAR2(15) NOT NULL := 'US';
lv_taxrate_num CONSTANT NUMBER(2,2) := .06;
```

BEGIN

```
---- PL/SQL executable statements ----
```

END;

▶ 17

2021/22 - LEI - AABD - PL/SQL

Cálculos com variáveis escalares

multiplicação




```
DECLARE
  lv_taxrate_num CONSTANT NUMBER(2,2) := .06;
  lv_total_num NUMBER(6,2) := 50;
  lv_taxamt_num NUMBER(4,2);
BEGIN
  lv_taxamt_num := lv_total_num * lv_taxrate_num;
  DBMS_OUTPUT.PUT_LINE(lv_taxamt_num);
END;
/
```

▶ 18

2021/22 - LEI - AABD - PL/SQL

Utilização de funções SQL

Funções SQL como **MONTHS_BETWEEN** podem ser usadas em programas PL/SQL.

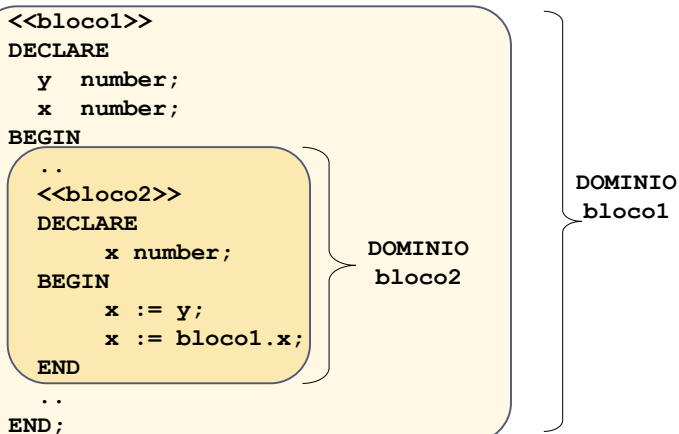


```

DECLARE
  lv_date1 DATE := '20-OCT-07';
  lv_date2 DATE := '20-SEP-05';
  lv_months_num NUMBER(3);
BEGIN
  lv_months_num := MONTHS_BETWEEN( lv_date1, lv_date2);
  DBMS_OUTPUT.PUT_LINE(lv_months_num);
END;
/
  
```

Scope das variáveis

Blocos encadeados



```

<<bloco1>>
DECLARE
  y number;
  x number;
BEGIN
  ..
  <<bloco2>>
  DECLARE
    x number;
  BEGIN
    x := y;
    x := bloco1.x;
  END
  ..
END;
  
```

DOMINIO
bloco1

DOMINIO
bloco2

As variáveis declaradas em cada bloco apenas são válidas nesse bloco, não sendo válidas exteriormente a esse bloco

Scope das variáveis

- ▶ Em blocos encadeados as variáveis são partilhadas?

```

SQL> DECLARE
2  lv_one NUMBER(2) := 10;
3  lv_two NUMBER(2) := 20;
4  BEGIN
5      DECLARE
6          lv_one NUMBER(2) := 30;
7          lv_three NUMBER(2) := 40;
8          BEGIN
9              lv_one := lv_one + 10;
10             lv_two := lv_two + 10;
11             DBMS_OUTPUT.PUT_LINE('Nested lv_one = '||lv_one);
12             DBMS_OUTPUT.PUT_LINE('Nested lv_two = '||lv_two);
13             DBMS_OUTPUT.PUT_LINE('Nested lv_three = '||lv_three);
14         END;
15         lv_one := lv_one + 10;
16         lv_two := lv_two + 10;
17         --lv_three := lv_three + 10;
18         DBMS_OUTPUT.PUT_LINE('Enclosing lv_one = '||lv_one);
19         DBMS_OUTPUT.PUT_LINE('Enclosing lv_two = '||lv_two);
20         --DBMS_OUTPUT.PUT_LINE('Enclosing lv_three = '||lv_three);
21     END;
22 /
Nested lv_one = 40
Nested lv_two = 30
Nested lv_three = 40
Enclosing lv_one = 20
Enclosing lv_two = 40

PL/SQL procedure successfully completed.

SQL> |

```

- ▶ Blocos “interiores” podem usar variáveis de blocos “exteriores”

▶ 21

2021/22 - LEI - AABD - PL/SQL

Variáveis *Host* ou *Bind*

- ▶ As variáveis *host* são referenciadas com “:” em PL/SQL

```
VARIABLE g_shopper NUMBER;
```

Create host variable

```
BEGIN
```

```
:g_shopper := 25;
```

```
END;
```

```
/
```

Reference
host variable

```
PRINT g_shopper
```

Print host variable

▶ 22

2021/22 - LEI - AABD - PL/SQL

Declarar variáveis *Host/Bind*

- ▶ Usar variáveis do ambiente da aplicação para enviar e receber dados de blocos PL/SQL.
- ▶ SQL*Plus é um exemplo de um ambiente de aplicação

```
BEGIN
    :g_state_txt := 'VA';
END;
/
```

Usar variáveis *Host/Bind*

```
DECLARE
    lv_tax_num NUMBER(4,2);
    lv_sub_num NUMBER(6,2) := 100;
BEGIN
    IF :g_state_txt = 'VA' THEN
        lv_tax_num := lv_sub_num * .06;
    ELSIF :g_state_txt = 'CA' THEN
        lv_tax_num := lv_sub_num * .08;
    ELSE
        lv_tax_num := lv_sub_num * .04;
    END IF;
    DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;
/
```

Atributo %TYPE

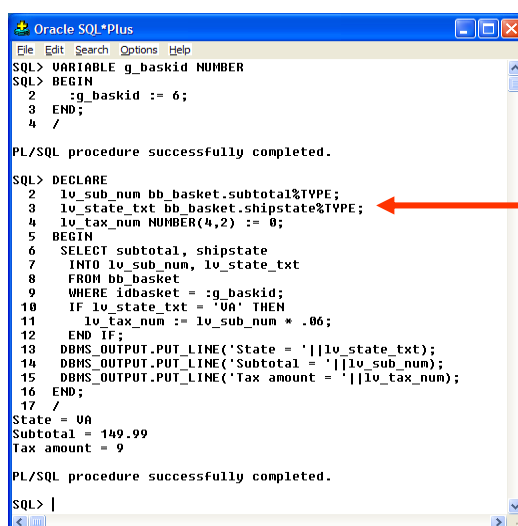
- ▶ Usar na declaração de variáveis
 - ▶ Dar um tipo baseado num atributo de uma tabela
- ▶ Ideal para declarar variáveis que receberão dados de uma tabela
- ▶ Diminui necessidades de manutenção
 - ▶ Não precisa alterar o código quando alterar a BD
- ▶ Chamado tipo de dados *anchored*

```
lv_basket_num bb_basket.idBasket%TYPE;
```

▶ 25

2021/22 - LEI - AABD - PL/SQL

%TYPE Example



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> VARIABLE g_baskid NUMBER
SQL> BEGIN
  2   :g_baskid := 6;
  3   END;
  4   /

PL/SQL procedure successfully completed.

SQL> DECLARE
  2   lv_sub_num bb_basket.subtotal%TYPE;
  3   lv_state_txt bb_basket.shipstate%TYPE;
  4   lv_tax_num NUMBER(4,2) := 0;
  5   BEGIN
  6   SELECT subtotal, shipstate
  7   INTO lv_sub_num, lv_state_txt
  8   FROM bb_basket
  9   WHERE idbasket = :g_baskid;
 10   IF lv_state_txt = 'UA' THEN
 11     lv_tax_num := lv_sub_num * .06;
 12   END IF;
 13   DBMS_OUTPUT.PUT_LINE('State = '||lv_state_txt);
 14   DBMS_OUTPUT.PUT_LINE('Subtotal = '||lv_sub_num);
 15   DBMS_OUTPUT.PUT_LINE('Tax amount = '||lv_tax_num);
 16   END;
 17   /
State = UA
Subtotal = 149.99
Tax amount = 9

PL/SQL procedure successfully completed.

SQL> |

```

▶ 26

2021/22 - LEI - AABD - PL/SQL

Atributo %TYPE

Útil para guardar obtidos através de um SELECT

SELECT *col1, col2,...* **INTO** *var1, var2,..*

```
DECLARE
  vNOME  autores.nome%TYPE;
  vIDADE autores.idade%TYPE;
BEGIN
  SELECT nome, idade
  INTO vNome, vIdade
  FROM autores
  WHERE codigo_autor = 1;

  DBMS_OUTPUT.PUT_LINE(vNome || ' idade ' || vIdade);
END;
/
```

► 27

2021/22 - LEI - AABD - PL/SQL

Atributo %ROWTYPE

- Criar uma estrutura *record* baseada na estrutura de uma tabela

```
DECLARE
  rec_shopper bb_shopper%ROWTYPE;
BEGIN
  SELECT *
  INTO rec_shopper
  FROM bb_shopper
  WHERE idshopper = :g_shopper;

  DBMS_OUTPUT.PUT_LINE(rec_shopper.lastname);
  DBMS_OUTPUT.PUT_LINE(rec_shopper.address);
  DBMS_OUTPUT.PUT_LINE(rec_shopper.email);
END;
/
```

► 28

2021/22 - LEI - AABD - PL/SQL

Tipo de dados *Record*

```

DECLARE
  TYPE type_basket IS RECORD (      -- Definir o tipo de registo
    basket      bb_basket.idBasket%TYPE,
    created      bb_basket.dtcreated%TYPE,
    qty          bb_basket.quantity%TYPE,
    sub          bb_basket.subtotal%TYPE);

  rec_basket type_basket;          -- Declarar um variável desse tipo

  lv_days_num NUMBER(3);
BEGIN
  SELECT idBasket, dtcreated, quantity, subtotal
    INTO rec_basket
  FROM bb_basket
 WHERE idShopper = 25 AND orderplaced = 0;
  lv_days_num := SYSDATE - rec_basket.created;
END;
/

```

▶ 29

2021/22 - LEI - AABD - PL/SQL

Comentar o Código

- ▶ Adicionar comentários ao código para
 - ▶ Identificar o objetivo do código
 - ▶ Descrever os passos do processamento
- ▶ Usar /* */
 - ▶ Para comentários com múltiplas linhas
- ▶ Usar --
 - ▶ Comentários de uma linha ou parte de uma linha

▶ 30

2021/22 - LEI - AABD - PL/SQL

Exemplos de comentários

```

DECLARE
    ex_prod_update EXCEPTION;  --For UPDATE of no rows exception
BEGIN
    /* This block is used to update product descriptions
       Constructed to support the Prod_desc.frm app screen
       Exception raised if no rows updated */
    UPDATE bb_product
        SET description = 'Mill grinder with 5 grind settings!'
        WHERE idProduct = 30;
    --Check if any rows updated
    IF SQL%NOTFOUND THEN
        RAISE ex_prod_update;
    END IF;
EXCEPTION
    WHEN ex_prod_update THEN
        DBMS_OUTPUT.PUT_LINE('Invalid product id entered');
END;
```

▶ 31

2021/22 - LEI - AABD - PL/SQL

Debugging no SQL*Plus

- ▶ Usar `DBMS_OUTPUT.PUT_LINE` para mostrar mensagens durante a execução

```
DBMS_OUTPUT.PUT_LINE('Produto inválido');
```

- ▶ Necessário ativar opção `SERVEROUTPUT`

```
SET SERVEROUTPUT ON
```

- ▶ Colocar mensagens ao longo do bloco

- ▶ Para determinar o fluxo de execução
- ▶ Para verificar o valor de variáveis

▶ 32

2021/22 - LEI - AABD - PL/SQL

Estruturas de decisão

- ▶ Controlar que comandos de um bloco PL/SQL serão executados
- ▶ Permite testar condições para determinar o fluxo de execução
- ▶ A maioria das linguagens de programação oferecem instruções IF e CASE para permitir processamento condicional

▶ 33

2021/22 - LEI - AABD - PL/SQL

Estruturas de decisão - IF

IF SIMPLES

```
IF condição THEN
  -- instruções
END IF;
```

IF ..THEN .. ELSE

```
IF condição THEN
  -- instruções
ELSE
  -- instruções
END IF;
```

▶ 34

2021/22 - LEI - AABD - PL/SQL

Estruturas de decisão - IF

IF SIMPLES

```
IF condição THEN
  -- instruções
END IF;
```

IF ..THEN .. ELSE

```
IF condição1 THEN
  -- instruções
ELSE
  IF condição2 THEN
    -- instruções
  ELSE
    -- instruções
  END IF;
END IF;
```

IF ..THEN .. ELSIF .. ELSE

```
IF condição THEN
  -- instruções
ELSIF condição THEN
  -- instruções
ELSE
  -- instruções
END IF;
```

IF Simples

```
DECLARE
  lv_state_txt CHAR(2) := 'ME';
  lv_sub_num NUMBER (5,2) := 100;
  lv_tax_num NUMBER (4,2);
BEGIN
  IF lv_state_txt = 'VA' THEN
    lv_tax_num := lv_sub_num *.06;
  END IF;
  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;
/
```

IF/THEN/ELSE

```

DECLARE
  lv_state_txt CHAR(2) := 'ME';
  lv_sub_num NUMBER (5,2) := 100;
  lv_tax_num NUMBER (4,2);
BEGIN
  IF lv_state_txt = 'VA' THEN
    lv_tax_num := lv_sub_num *.06;
  ELSE
    lv_tax_num := lv_sub_num *.04;
  END IF;
  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;
/

```

IF/THEN/ELSIF/ELSE

```

DECLARE
  lv_state_txt CHAR(2) := 'ME';
  lv_sub_num NUMBER (5,2) := 100;
  lv_tax_num NUMBER (4,2);
BEGIN
  IF lv_state_txt = 'VA' THEN
    lv_tax_num := lv_sub_num *.06;
  ELSIF lv_state_txt = 'ME' THEN
    lv_tax_num := lv_sub_num *.05;
  ELSIF lv_state_txt = 'NY' THEN
    lv_tax_num := lv_sub_num *.07;
  ELSE
    lv_tax_num := lv_sub_num *.04;
  END IF;
  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;
/

```

Operadores Lógicos no IF

- ▶ Operadores lógicos (**AND** , **OR**) permitem que sejam verificadas várias condições

```
IF lv_state_txt = 'VA' OR lv_state_txt = 'PA' THEN  
    lv_tax_num := lv_sub_num * .06;  
ELSE  
    lv_tax_num := lv_sub_num * .04;  
END IF;
```

Estruturas de decisão - CASE

- ▶ CASE básico
- ▶ *Searched* CASE
- ▶ Expressão CASE

CASE Básico

```
CASE variável
  WHEN valor1 THEN
    -- instruções
  WHEN valor2 THEN
    -- instruções
  ELSE
    -- instruções
END CASE;
```

```
DECLARE
  lv_state_txt CHAR(2) := 'ME';
  lv_sub_num NUMBER (5,2) := 100;
  lv_tax_num NUMBER (4,2);
BEGIN
  CASE lv_state_txt
    WHEN 'VA' THEN
      lv_tax_num := lv_sub_num *.06;
    WHEN 'ME' THEN
      lv_tax_num := lv_sub_num *.05;
    WHEN 'NY' THEN
      lv_tax_num := lv_sub_num *.07;
    ELSE
      lv_tax_num := lv_sub_num *.04;
  END CASE;
  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;
```

▶ 41

2021/22 - LEI - AABD - PL/SQL

Searched CASE

```
CASE
  WHEN condição1 THEN
    -- instruções
  WHEN condição1 THEN
    -- instruções
  ELSE
    -- instruções
END CASE;
```

```
DECLARE
  lv_state_txt CHAR(2) := 'ME';
  lv_zip_txt CHAR(2) := '23321';
  lv_sub_num NUMBER (5,2) := 100;
  lv_tax_num NUMBER (4,2);
BEGIN
  CASE
    WHEN lv_state_txt = 'VA' THEN
      lv_tax_num := lv_sub_num *.06;
    WHEN lv_zip_txt = '23321' THEN
      lv_tax_num := lv_sub_num *.02;
    ELSE
      lv_tax_num := lv_sub_num *.04;
  END CASE;
  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;
```

▶ 42

2021/22 - LEI - AABD - PL/SQL

CASE Expression

```

variavel :=
  CASE outra_variável
    WHEN valor1 THEN
      -- assume valor 4
    WHEN valor2 THEN
      -- assume valor 5
    ELSE
      -- assume valor 7
  END;

```

```

DECLARE
  lv_state_txt CHAR(2) := 'ME';
  lv_sub_num NUMBER (5,2) := 100;
  lv_tax_num NUMBER (4,2);
BEGIN
  lv_tax_num := CASE lv_state_txt
    WHEN 'VA' THEN lv_sub_num * .06;
    WHEN 'ME' THEN lv_sub_num * .05;
    WHEN 'NY' THEN lv_sub_num * .07;
    ELSE lv_sub_num * .04;
  END CASE;
  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
END;

```

▶ 43

2021/22 - LEI - AABD - PL/SQL

Ciclos – LOOP

- ▶ Permite que um comando ou conjunto de comandos seja executado mais do que uma vez
- ▶ Um ciclo deve ter instruções que permitam terminar a repetição para evitar “ciclos infinitos”

▶ 44

2021/22 - LEI - AABD - PL/SQL

LOOP Básico

LOOP

```
-- instruções
EXIT [ WHEN condição];
-- instruções
END LOOP;
```

```
DECLARE
  lv_cnt_num NUMBER(2) := -1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE(lv_tax_num);
    EXIT WHEN lv_cnt_num >= 5;
    lv_cnt_num := lv_cnt_num +1;
  END LOOP;
END;
```

▶ 45

2021/22 - LEI - AABD - PL/SQL

Ciclo WHILE

WHILE condição

LOOP

```
-- instruções
END LOOP;
```

```
DECLARE
  lv_cnt_num NUMBER(2) := -1;
BEGIN
  WHILE lv_cnt_num <= 5
  LOOP
    DBMS_OUTPUT.PUT_LINE(lv_tax_num);
    lv_cnt_num := lv_cnt_num +1;
  END LOOP;
END;
```

▶ 46

2021/22 - LEI - AABD - PL/SQL

Ciclo FOR

```
FOR var IN [REVERSE] menor .. maior
LOOP
    -- instruções
END LOOP;
```

```
BEGIN
  FOR i IN 1 .. 5
  LOOP
    DBMS_OUTPUT.PUT_LINE( i );
  END LOOP;
END;
```

▶ 47

2021/22 - LEI - AABD - PL/SQL

Instrução GOTO

- ▶ Permite “saltar” para outra zona de código para continuar a execução
- ▶ A utilização de GOTO é desaconselhada já que torna o código difícil de perceber e de manter

```
<<label_1>>
A:= A - 1;
IF A > B THEN
    GOTO label_1;
END IF;
```

▶ 48

2021/22 - LEI - AABD - PL/SQL

Incluir SQL num bloco

- ▶ Comandos DML podem ser incluídos dentro da área executável de blocos PL/SQL

DML

- INSERT
- UPDATE
- DELETE

```
DECLARE
  N NUMBER := 1;
BEGIN

  DELETE FROM LIVROS WHERE CODIGO_LIVRO = N;

  UPDATE LIVROS SET PRECO_TABELA = PRECO_TABELA * 1.1;

  INSERT INTO TEMP VALUES (1,100, 'OLA');

END;
```

▶ 49

2021/22 - LEI - AABD - PL/SQL

Incluir SELECT num bloco

- ▶ Comandos **SELECT** usados para obter dados
- ▶ Incluí uma cláusula **INTO** para atribuir os dados obtidos a variáveis

```
DECLARE
  N NUMBER; MEDIA NUMBER;
BEGIN
  SELECT COUNT(*), AVG(PRECO_TABELA) INTO N, MEDIA
  FROM LIVROS;
  IF N > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Existem ' || N || ' livros.');
```

▶ 50

2021/22 - LEI - AABD - PL/SQL

Incluir SELECT num bloco

```

SQL> DECLARE
2  lvBasket_num NUMBER(3);
3  lv_created_date DATE;
4  lv_qty_num NUMBER(2);
5  lv_sub_num NUMBER(5,2);
6  lv_days_num NUMBER(3);
7  lv_shopper_num NUMBER(3) := 25;
8  BEGIN
9  SELECT idBasket, dtcreated, quantity, subtotal
10 INTO lvBasket_num, lv_created_date, lv_qty_num, lv_sub_num
11 FROM bbBasket
12 WHERE idShopper = lv_shopper_num
13 AND orderplaced = 0;
14 lv_days_num := SYSDATE - lv_created_date;
15 DBMS_OUTPUT.PUT_LINE(lvBasket_num||' * '||lv_created_date||' * '||
16 lv_qty_num||' * '||lv_sub_num||' * '||lv_days_num);
17 END;
18 /
12 * 19-FEB-07 * 7 * 72.4 * 11

PL/SQL procedure successfully completed.

SQL> |
  
```

▶ 51

2021/22 - LEI - AABD - PL/SQL

Limitações do SELECT num bloco

O **SELECT** tem que retornar um única linha

- ▶ Se não retornar nada → exceção `NO_DATA_FOUND`

```

DECLARE
  P NUMBER;
BEGIN
  SELECT preco_tabela INTO P
  FROM LIVROS
  WHERE codigo_livro = 0;
END;

Error starting at line : 1 in command -
Error report -
ORA-01403: não foram encontrados dados
ORA-06512: na linha 4
01403. 00000 - "no data found"
*Cause:      No data was found from the objects.
*Action:     There was no data from the objects which may
be due to end of fetch.
  
```

▶ 52

2021/22 - LEI - AABD - PL/SQL

Limitações do SELECT num bloco

O **SELECT** tem que retornar um única linha

- ▶ Se retornar mais que uma linha → exceção **TO_MANY_ROWS**

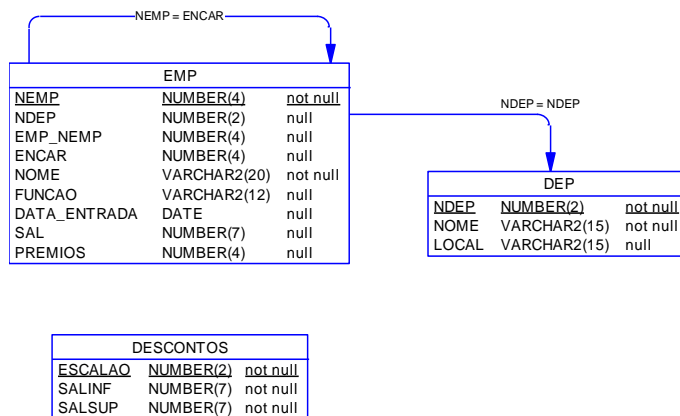
```
DECLARE
  P NUMBER;
BEGIN
  SELECT preco_tabela INTO P
  FROM LIVROS
  WHERE codigo_livro > 0;
END;
```

Error report -
ORA-01422: extração exacta devolve mais que o número pedido de linhas
 ORA-06512: na linha 4
 01422. 00000 - "exact fetch returns more than requested number of rows"
***Cause:** The number specified in exact fetch is less than the rows returned.
***Action:** Rewrite the query or change number of rows requested

▶ 53

2021/22 - LEI - AABD - PL/SQL

Exercícios



▶ 54

2021/22 - LEI - AABD - PL/SQL

Exercícios

I. Atualize o atributo prémios do seguinte modo:

1. se nº empregados > 10, 10% salário médio dos empregados
2. se nº empregados < 3, 20% da diferença entre o salário mais alto e mais baixo
3. Restantes casos, 2% * número de empregados
4. Presidente, tem um aumento de 10% do salário mais baixo * número de empregados que recebem esse salário

► 55

2021/22 - LEI - AABD - PL/SQL

Atualize o atributo prémios do seguinte modo:

1. se nº empregados > 10, 10% salário médio dos empregados
2. se nº empregados < 3, 20% da diferença entre o salário mais alto e mais baixo
3. Restantes casos, 2% * número de empregados
4. Presidente, aumentado em 10% do salário mais baixo * número de empregados que recebem esse salário

DECLARE

END ► 56

2021/22 - LEI - AABD - PL/SQL