

# Aula 10

## Introdução

### Programação PL/SQL

Banco de Dados – Computação - UNISUL



## PL/SQL

### DEFINIÇÃO:

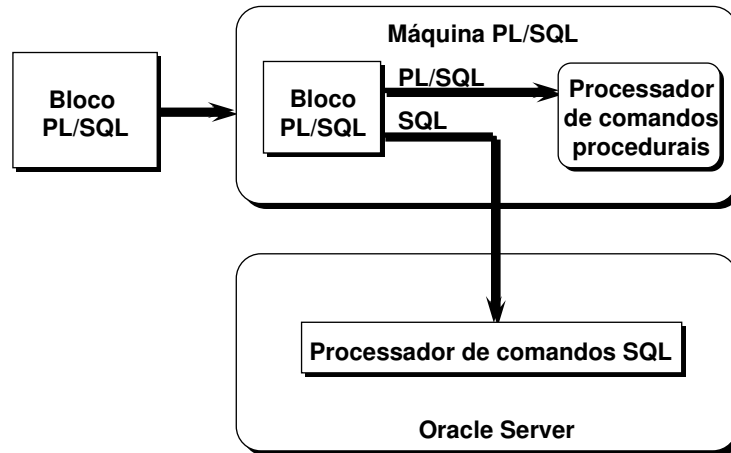
- Linguagem de Programação procedural para uso do RDBMS ORACLE, utilizando como sub-linguagem o SQL

### USO:

- Procedimentos
- Funções
- Triggers



## Ambiente PL/SQL

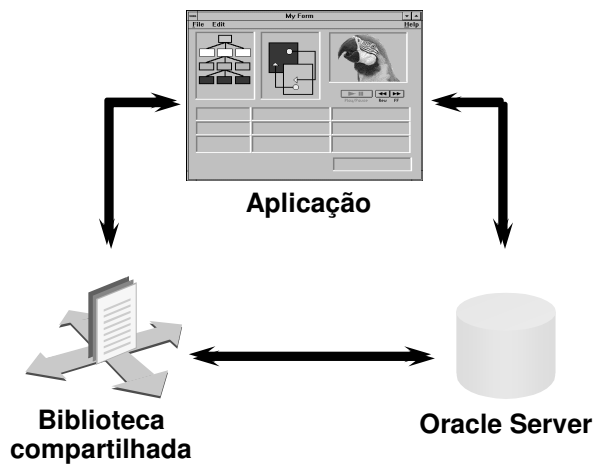


3

Banco de Dados— Computação - UNISUL

## Vantagens do PL/SQL

**Integração**



4

Banco de Dados— Computação - UNISUL

## Vantagens do PL/SQL

- Portabilidade
- Podem ser declarados identificadores.
- Programar com estruturas de controle procedurais.
- Erros podem ser manipulados.

## Tabelas Utilizadas no Curso

**EMP**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	1500		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
			98	22-FEB-81	1250	500	30
			66	03-DEC-81	3000		30
			02	17-DEC-80	800		30
			66	09-DEC-82	3000		30
			88	12-JAN-83	1100		30
			82	23-JAN-82	1300		30

**DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

**SALGRADE**

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

# Blocos PL/SQL

## Estrutura do Bloco PL/SQL

- **DECLARE – Opcional**
  - Variáveis, cursores, exceções definidas pelo usuário
- **BEGIN – Obrigatório**
  - Comandos SQL
  - Comandos PL/SQL
- **EXCEPTION – Opcional**
  - Ações para serem executadas quando ocorrerem erros
- **END; – Obrigatório**

```
DECLARE
  ...
BEGIN
  ...
EXCEPTION
  ...
END;
```

## Estrutura do Bloco PL/SQL

```
DECLARE
  v_variavel VARCHAR2(5);
BEGIN
  SELECT      nome_coluna
  INTO v_variavel
  FROM        nome_tabela;
EXCEPTION
  WHEN nome_exception THEN
  ...
END;
```

```
DECLARE
  ...
BEGIN
  ...
EXCEPTION
  ...
END;
```

## Tipos de Bloco

### Anonymous

```
[DECLARE]

BEGIN
  --comandos
[EXCEPTION]

END;
```

### Procedure

```
PROCEDURE nome
IS
BEGIN
  --comandos
[EXCEPTION]

END;
```

### Function

```
FUNCTION nome
RETURN tipo_dado
IS
BEGIN
  --comandos
  RETURN valor;
[EXCEPTION]

END;
```

## **Uso de Variáveis**

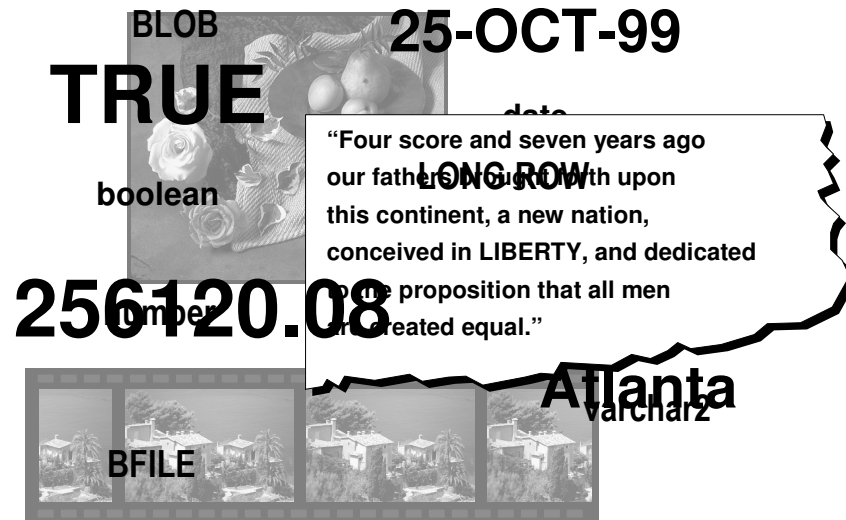
**Variáveis são usadas para:**

- **Armazenar dados temporariamente**
- **Manipular valores armazenados**
- **Reusabilidade**
- **Utilizar o mesmo tipo de declaração que uma coluna definida no Banco de Dados**

## **Manipulando Variáveis no PL/SQL**

- **Declaração e inicialização de variáveis dentro da seção de declaração.**
- **Atribuir novos valores para variáveis dentro da seção de execução.**
- **Passar valores para blocos PL/SQL através de parâmetros.**
- **Visualizar resultados através de variáveis de saída.**

## Tipos de Variáveis



13

Banco de Dados– Computação - UNISUL

## Declaração de Variáveis PL/SQL

### Sintaxe

```
identificador [CONSTANT] tipo_dado [NOT NULL]  
[:= | DEFAULT expr];
```

### Exemplos

```
Declare  
v_dt_nasc      DATE;  
v_deptno      NUMBER(2) NOT NULL := 10;  
v_location     VARCHAR2(13) := 'Atlanta';  
c_comm        CONSTANT NUMBER := 1400;
```

14

Banco de Dados– Computação - UNISUL

## Declaração de Variáveis PL/SQL


### Regras:

- Seguir convenção de nomes.
- Inicializar variáveis identificadas como NOT NULL.
- Inicializar identificadores pelo uso do operador de atribuição (:=).
- Declarar apenas um identificador por linha.

## Regras de Nomeação

- Duas variáveis podem ter o mesmo nome, desde que estejam em blocos diferentes.
- O nome da variável (identificador) não deve ter o mesmo nome de colunas de tabelas usadas no mesmo bloco.

```
DECLARE
empno  NUMBER(4)
BEGIN
SELECT empno
INTO empno
FROM emp
WHERE ename = 'SMITH';
END;
```





## Atribuindo Valores às Variáveis

### Sintaxe

```
identificador := expr;
```

### Exemplos

Atribuir uma data de admissão pré-definida para um novo empregado.

```
v_dt_adm := '31-DEC-98';
```

Atribuir um nome para o empregado.

```
v_ename := 'Maduro';
```

## Inicialização de Variáveis e Palavras-chaves

- := Operador de Atribuição
- DEFAULT
- NOT NULL

## Tipo de dado Escalar

- Armazena um dado simples

25-OCT-00  
256120.08  
"Four score and seven years ago our fathers brought forth upon this continent, a new nation, conceived in LIBERTY and dedicated to the proposition that all men are created equal."  
TRUE  
Atlanta

## Principais Tipos de dados Escalares no ORACLE

- VARCHAR2 (*tamanho\_máximo*)
- NUMBER [(*precisão, casas*)]
- DATE
- CHAR [(*tamanho\_máximo*)]
- BOOLEAN

# Declaração de Variáveis Escalares

## Exemplos

```
v_job      VARCHAR2(9);  
v_count    BINARY_INTEGER := 0;  
v_total_sal NUMBER(9,2) := 0;  
v_orderdate DATE := SYSDATE + 7;  
c_tax_rate CONSTANT NUMBER(3,2) := 8.25;  
v_valid    BOOLEAN NOT NULL := TRUE;
```

## Atributo %Type

- Declara uma variável de acordo com:
  - uma definição de uma coluna de uma tabela
  - copia o tipo de declaração de uma variável previamente definida

## Declaração de variáveis com Atributo %TYPE

### Exemplos

```
...  
v_ename          emp.ename%TYPE;  
v_balance        NUMBER(7,2);  
v_min_balance    v_balance%TYPE := 10;  
...
```

## Declaração de variáveis BOOLEAN

- Somente os valores TRUE, FALSE, e NULL podem ser atribuídos a uma variável booleana.
- Podem ser conectadas pelos operadores lógicos AND, OR, e NOT.
- Expressões aritméticas, char ou date podem ter como retorno um valor booleano.

# Variáveis BOOLEAN

## Exemplos

```
v_sal1 := 50000;  
v_sal2 := 60000;
```

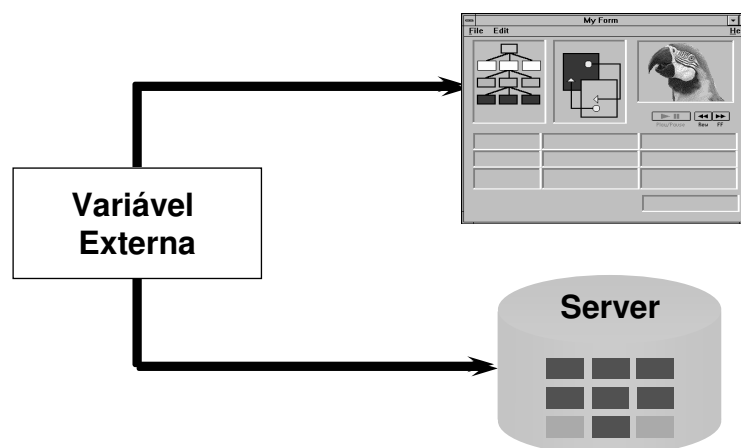
## A seguinte expressão resulta TRUE

```
v_sal1 < v_sal2
```

- Declaração e inicialização de variável boolean.

```
v_comm_sal BOOLEAN := (v_sal1 < v_sal2);
```

# Variáveis Externas (Bind)



## Criando Variáveis Externas

### Exemplo

```
SQL> VAR[|ABLE] nome_variável NUMBER  
  
...  
  
SQL> PRINT nome_variável
```

## Referenciando variáveis Não- PL/SQL

**Armazenando o salário anual em uma variável declarada no ambiente principal SQL\*Plus.**

```
:g_monthly_sal := v_sal / 12;
```

- Referencia uma variável não-PL/SQL como variável externa.
- Utiliza-se o prefixo (:).

## Resumo

- **Blocos PL/SQL são compostos pelas seguintes seções:**
  - Declaração (opcional)
  - Execução (obrigatório)
  - Manipulação de Exceções (opcional)
- **Um bloco PL/SQL pode ser um bloco anonymous, procedure, ou function.**

```
DECLARE
  ○ ○ ○
BEGIN
  ○ ○ ○
EXCEPTION
  ○ ○ ○
END;
```

## Resumo

- **Identificadores PL/SQL:**
  - São definidos na seção de declaração
  - Podem ser tipos de dados escalares, compostos, referência ou LOB
  - Podem ser baseados nas estruturas de outra variável ou objetos da base de dados (campos de tabela)
  - Podem ser inicializados

# Comandos Executáveis

## Sintaxe e Regras para Blocos PL/SQL

- Comandos podem ser escritos usando várias linhas.
- Unidades léxicas tem que ser separadas por espaços:
  - Delimitadores
  - Identificadores
  - Literais
  - Comentários





## Sintaxe e Regras para Blocos PL/SQL

### Identificadores

- Podem conter no máximo 30 caracteres
- Palavras reservadas não podem ser utilizadas, a menos que estejam entre aspas(“)
- Tem que começar com um caracter alfabético
- Não pode ter o mesmo nome de uma coluna de uma tabela da base

## Sintaxe e Regras para Blocos PL/SQL

### Literais

- Literais tipo Char e date tem que estar entre apóstrofes (‘)

```
v_ename := 'Henderson';
```

- Numéricos podem ser valores simples ou notação científica. (por exemplo, 2E5, significa  $2 \times 10^5$ )

## Comentário

- Comentário de uma linha usa-se (- -).
- Comentário para várias linhas coloca-se entre os símbolos /\* e \*/.

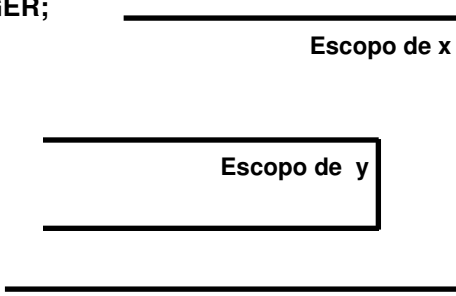
### Exemplo

```
...  
v_sal NUMBER (9,2);  
BEGIN  
/* Calcula o salario anual com base nos salários mensais */  
v_sal := v_sal * 12;  
END; -- Final da transação
```

## Ninhos de Blocos e Escopo de Variável

### Exemplo

```
...  
x BINARY_INTEGER;  
BEGIN  
...  
DECLARE  
y NUMBER;  
BEGIN  
...  
END;  
...  
END;
```



## Operadores no PL/SQL

- Lógico
- Aritmético
- Concatenação
- Parênteses para controlar a ordem das operações
- Operador Exponencial (\*\*)



Idem ao  
SQL

## Operadores no PL/SQL

### Exemplos

- Incrementa o índice para um loop.

```
v_count := v_count + 1;
```

- Atribui um valor para uma variável boolean.

```
v_equal := (v_n1 = v_n2);
```

- Valida o número do empregado se contiver um valor.

```
v_valid := (v_empno IS NOT NULL);
```

## Usando Variáveis Externas(bind)

Para referenciar uma variável externa no PL/SQL, deve-se usar (:) antes do nome da variável.

### Exemplo

```
DECLARE
  v_sal emp.sal%TYPE;
BEGIN
  SELECT      sal
  INTO  v_sal
  FROM emp
  WHERE      empno = 7369;
  :salary    := v_sal;
END;
```

## Padrões de Programação

Para facilitar a manutenção dos códigos:

- Documentar o código com comentários.
- Desenvolver uma convenção(padão) para a codificação
- Desenvolver uma convenção de nomenclatura para identificadores e outros objetos.

## Resumo

- **Estrutura do bloco PL/SQL:**
  - Ninhos de blocos e regras de escopo
- **Programação PL/SQL:**
  - Funções
  - Conversão de tipos de dados
  - Operadores
  - Variáveis Externas (bind)
  - Convenções e regras

```
DECLARE
  ○ ○ ○
BEGIN
  ○ ○ ○
EXCEPTION
  ○ ○ ○
END;
```

## Interagindo com Oracle Server

## Comando SELECT no PL/SQL

Recupera um dado da base com SELECT.

Sintaxe

```
SELECT select_list
INTO    {variable_name[, variable_name]...
        | record_name}
FROM    table
WHERE   condition;
```

## Comando SELECT no PL/SQL

Cláusula INTO é obrigatória.

Exemplo

```
DECLARE
  v_deptno  NUMBER(2);
  v_loc     VARCHAR2(15);
BEGIN
  SELECT     deptno, loc
  INTO v_deptno, v_loc
  FROM       dept
  WHERE      dname = 'SALES';
  ...
END;
```

# Recuperando dados no PL/SQL

## Exemplo

```
DECLARE
  v_orderdate ord.orderdate%TYPE;
  v_shipdate  ord.shipdate%TYPE;
BEGIN
  SELECT orderdate, shipdate
  INTO v_orderdate, v_shipdate
  FROM   ord
  WHERE  id = 157;
  ...
END;
```

# Recuperando dados no PL/SQL

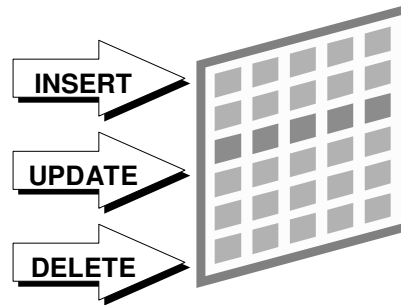
## Exemplo

```
DECLARE
  v_sum_sal emp.sal%TYPE;
  v_deptno  NUMBER NOT NULL := 10;
BEGIN
  SELECT SUM(sal) -- função de grupo
  INTO v_sum_sal
  FROM emp
  WHERE deptno = v_deptno;
END;
```

# Manipulação de dados usando PL/SQL

Fazendo alterações nas tabelas usando comando DML:

- INSERT
- UPDATE
- DELETE



## Inserindo Dados

Inserindo informações de um novo empregado na tabela EMP.

**Exemplo**

```
DECLARE
v_empno          emp.empno%TYPE;
BEGIN
  SELECT          empno_sequence.NEXTVAL
  INTO            v_empno
  FROM            dual;
  INSERT INTO     emp(empno, ename, job, deptno)
  VALUES(v_empno, 'HARDING', 'CLERK', 10);
END;
```



## Atualizando Dados

**Incrementa o valor do salário de todos os empregados que são 'ANALYST'**

### Exemplo

```
DECLARE
  v_sal_increase emp.sal%TYPE := 2000;
BEGIN
  UPDATE      emp
  SET    sal = sal + v_sal_increase
  WHERE   job = 'ANALYST';
END;
```

## Eliminando Dados

**Elimina os empregados que estão no departamento 10.**

### Exemplo

```
DECLARE
  v_deptno emp.deptno%TYPE := 10;
BEGIN
  DELETE FROM emp
  WHERE deptno = v_deptno;
END;
```

## **Comandos COMMIT e ROLLBACK**

- O início de uma transação se dá com um comando DML e termina com um comando COMMIT ou ROLLBACK.
- Usa-se os comandos SQL COMMIT e ROLLBACK para terminar uma transação explícita.

## **Cursor SQL**

- Um cursor é uma área de trabalho fechada do SQL.
- Existem dois tipos de cursores:
  - Cursor Implícito
  - Cursor Explícito
- O Oracle Server usa cursores implícitos para executar seus comandos SQL.
- Cursores Explícitos são declarados pelo programador.

## Atributos do Cursor SQL

Usando atributos do cursor SQL, podemos testar resultados dos comandos SQL.

SQL%ROWCOUNT	número de linhas afetadas pelo comando SQL mais recente(retorna um valor inteiro).
SQL%FOUND	retorna TRUE se uma ou mais linhas foram afetadas pelo comando SQL.
SQL%NOTFOUND	retorna TRUE se nenhuma linha foi afetada pelo comando SQL.

## Atributos do Cursor SQL

### Exemplo

```
VARIABLE rows_deleted char(30)
DECLARE
  v_ordid NUMBER := 605;
BEGIN
  DELETE FROM item
  WHERE   ordid = v_ordid;
         :rows_deleted := SQL%ROWCOUNT
         || ' rows deleted.';
END;
PRINT rows_deleted
```