

FIAP GRADUAÇÃO

# Tecnologia em Análise e Desenvolvimento de Sistemas

Application Development For Databases

PROF. MILTON

## Declarando Variáveis PL/SQL

## Objetivos

Ao concluir esta lição, você será capaz de:

- Reconhecer identificadores válidos e inválidos
- Listar os usos de variáveis
- Declarar e inicializar variáveis
- Listar e descrever vários tipos de dados
- Identificar as vantagens da utilização do atributo `%TYPE`
- Declarar, usar e exibir variáveis de bind

### Objetivos

Você já aprendeu sobre blocos PL/SQL básicos e suas seções. Nesta lição, você conhecerá os identificadores válidos e inválidos. Você aprenderá a declarar e inicializar variáveis na seção declarativa de um bloco PL/SQL. A lição descreve os vários tipos de dados. Você conhecerá também o atributo `%TYPE` e suas vantagens.

# Agenda

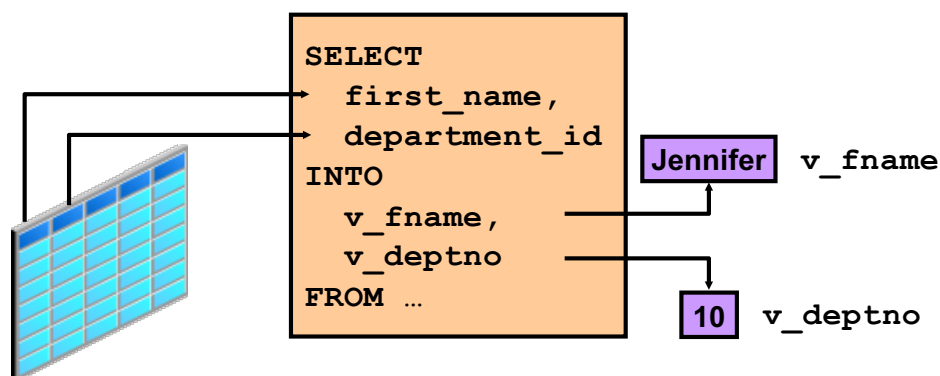
FIAP

- Apresentando variáveis
- Examinando tipos de dados de variável e o atributo `%TYPE`
- Examinando variáveis de bind

## Uso de Variáveis

As variáveis podem ser usadas para:

- Armazenamento temporário de dados
- Manipulação dos valores armazenados
- Reutilização



6

### Uso de Variáveis

Com o código PL/SQL, você pode declarar variáveis e depois usá-las em instruções procedurais e códigos SQL.

As variáveis são usadas principalmente para o armazenamento de dados e a manipulação de valores armazenados. Considere a instrução PL/SQL no slide. A instrução recupera `first_name` e `department_id` da tabela. Se precisar manipular `first_name` ou `department_id`, você precisará armazenar o valor recuperado. As variáveis são usadas para armazenar temporariamente o valor. Os valores armazenados nessas variáveis podem ser usados para processar e manipular os dados. As variáveis podem armazenar qualquer objeto PL/SQL, como variáveis, tipos, cursores e subprogramas.

*Reutilização* é outra vantagem da declaração de variáveis. Após as variáveis serem declaradas, você poderá usá-las repetidamente em uma aplicação, fazendo referência a elas diversas vezes em várias instruções.

## Requisitos para Nomes de Variáveis

Um nome de variável:

- Deve começar com uma letra
- Pode conter letras ou números
- Pode conter caracteres especiais (como \$, \_ e #)
- Não pode conter mais de 30 caracteres
- Não deve conter palavras reservadas



### Requisitos para Nomes de Variáveis

As regras para nomeação de uma variável estão listadas no slide.

# Tratando Variáveis no Código PL/SQL

As variáveis são:

- Declaradas e inicializadas (opcionalmente) na seção declarativa
- Valores novos designados e usados na seção executável
- Transmitidas como parâmetros para subprogramas PL/SQL
- Usadas para armazenar a saída de um subprograma PL/SQL

## Tratando Variáveis no Código PL/SQL

Você pode usar variáveis das seguintes formas:

- **Declará-las e inicializá-las na seção declarativa:** Você pode declarar variáveis na parte declarativa de qualquer bloco, subprograma ou pacote PL/SQL. As declarações alocam espaço de armazenamento para um valor, especificam o seu tipo de dados e nomeiam o local de armazenamento para que você possa fazer referência a ele. As declarações também podem designar um valor inicial e impor a constraint NOT NULL na variável. Não são permitidas referências futuras. É necessário declarar uma variável antes de fazer referência a ela em outras instruções, incluindo outras instruções declarativas.
- **Use-as e atribua novos valores a elas na seção executável:** Na seção executável, o valor existente da variável pode ser substituído por um novo valor.
- **Especifique-as como parâmetros para os subprogramas PL/SQL:** Os subprogramas podem utilizar parâmetros. Você pode especificar variáveis como parâmetros para os subprogramas.
- **Use-as para armazenar a saída de um subprograma PL/SQL:** As variáveis podem ser usadas para armazenar o valor que é retornado por uma função.



## Declarando e Inicializando Variáveis PL/SQL

Sintaxe:

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

Exemplos:

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

9

## Declarando e Inicializando Variáveis PL/SQL

Você deve declarar todos os identificadores PL/SQL na seção declarativa para poder fazer referência a eles no bloco PL/SQL. Você tem a opção de designar um valor inicial a uma variável (conforme mostrado no slide). Não é necessário designar um valor a uma variável para declará-la. Se você fizer referência a outras variáveis em uma declaração, certifique-se de que elas já tenham sido declaradas separadamente em uma instrução anterior.

Na sintaxe:

<i>identifier</i>	É o nome da variável
CONSTANT	Restringe a variável para que seu valor não possa mudar (constantes devem ser inicializadas.)
<i>data type</i> apenas	É um tipo de dados escalar, composto, de referência ou LOB (Este curso aborda tipos de dados escalares, compostos e LOB).
NOT NULL	Restringe a variável de modo que ela contenha um valor (variáveis NOT NULL devem ser inicializadas).
<i>expr</i>	É uma expressão PL/SQL que pode ser uma expressão literal, outra variável ou uma expressão contendo operadores e funções.

**Observação:** Além das variáveis, também é possível declarar cursores e exceções na seção declarativa. Você será ensinado a declarar cursores na lição “Usando Cursores Explícitos” e aprenderá sobre as exceções na lição “Tratamento de Exceções”.

## Declarando e Inicializando Variáveis PL/SQL

1

```
DECLARE
    v_myName VARCHAR2(20);
BEGIN
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
    v_myName := 'John';
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

2

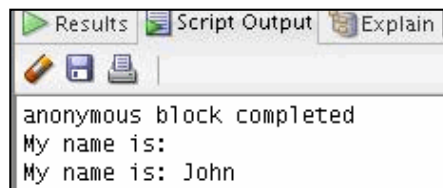
```
DECLARE
    v_myName VARCHAR2(20) := 'John';
BEGIN
    v_myName := 'Steven';
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

10

### Declarando e Inicializando Variáveis PL/SQL (continuação)

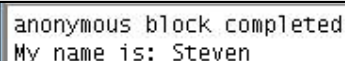
Examine os dois blocos de código no slide.

- No primeiro bloco, a variável `v_myName` é declarada, mas não inicializada. Um valor `John` é designado à variável na seção executável.
  - Os literais de string devem ser delimitados por aspas simples. Caso a sua string tenha aspas como em “Today’s Date”, então ela deverá ser `'Today' 's Date'`.
  - O operador de designação é: `:=`.
  - O procedure `PUT_LINE` é chamado pela especificação da variável `v_myName`. O valor da variável é concatenado com a string `'My name is: '`.
  - A saída desse bloco anônimo é:



```
anonymous block completed
My name is:
My name is: John
```

- No segundo bloco, a variável `v_myName` é declarada e inicializada na seção declarativa. `v_myName` armazena o valor `John` após a inicialização. Esse valor é manipulado na seção executável do bloco. A saída desse bloco anônimo é:



```
anonymous block completed
My name is: Steven
```

## Delimitadores em Literais de String

```
DECLARE
    v_event VARCHAR2(15);
BEGIN
    v_event := q'!Father's day!';
    DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :
    || v_event );
    v_event := q'[Mother's day]';
    DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is :
    || v_event );
END;
/
```

Saída  
resultante

```
anonymous block completed
3rd Sunday in June is : Father's day
2nd Sunday in May is : Mother's day
```

11

## Delimitadores em Literais de String

Se a sua string contiver um apóstrofo (idêntico a uma aspa simples), será necessário usar aspas duplas, como neste exemplo:

```
v_event VARCHAR2(15):='Father''s day';
```

A primeira aspa simples funciona como o caractere de escape. Isso tornará a sua string complicada, especialmente se houver instruções SQL como strings. Você pode especificar como delimitador qualquer caractere que não esteja presente na string. O slide mostra como usar a notação `q'` para especificar o delimitador. O exemplo usa `!` e `[` como delimitadores. Considere o seguinte exemplo:

```
v_event := q'!Father's day!';
```

Compare-o com o primeiro exemplo mostrado nesta página. Inicie a string com `q'` se desejar usar um delimitador. O caractere seguinte à notação é o delimitador usado. Informe a string após especificar o delimitador, feche o delimitador e feche a notação com uma aspa simples. O exemplo a seguir mostra como usar `[` como um delimitador:

```
v_event := q'[Mother's day]';
```

## Agenda

- Apresentando variáveis
- Examinando tipos de dados de variável e o atributo `%TYPE`
- Examinando variáveis de bind

## Tipos de Variáveis

- Variáveis PL/SQL:
  - Escalar
  - Referência
  - LOBs (large objects)
  - Composta
- Variáveis não PL/SQL: Variáveis de bind

### Tipos de Variáveis

Toda variável PL/SQL tem um tipo de dados, que especifica um formato de armazenamento, constraints e uma faixa válida de valores. A linguagem PL/SQL suporta várias categorias de tipos de dados, incluindo os tipos escalares, de referência, LOBs (large objects) e compostos.

- **Tipos de dados escalares:** Os tipos de dados escalares armazenam um único valor. O valor depende do tipo de dados da variável. Por exemplo, a variável `v_myName` do exemplo da seção “Declarando e Inicializando Variáveis PL/SQL” (desta lição) é do tipo `VARCHAR2`. Portanto, `v_myName` pode armazenar um valor de string. O código PL/SQL também suporta variáveis booleanas.
- **Tipos de dados de referência:** Os tipos de dados de referência armazenam valores, chamados *ponteiros*, que apontam para um local de armazenamento.
- **Tipos de dados LOB:** Os tipos de dados LOBs armazenam valores, chamados *localizadores*, que especificam a localização de objetos grandes (como imagens gráficas) que são armazenados fora da tabela.
- **Tipos de dados compostos:** Os tipos de dados compostos estão disponíveis quando você utiliza variáveis PL/SQL de *conjunto* e de *registro*. Conjuntos e registros PL/SQL contêm elementos internos que você pode tratar como variáveis individuais.

As variáveis que não são PL/SQL contêm variáveis de linguagem host declaradas em programas pré-compiladores, campos de tela em aplicações Forms e variáveis de host. Você conhecerá as variáveis de host posteriormente nesta lição.

Para obter mais informações sobre LOBs, consulte *PL/SQL User's Guide and Reference*.

## Tipos de Variáveis

TRUE



15-JAN-09

Branca de Neve  
Há muito tempo,  
em um reino muito distante,  
vivia uma princesa chamada  
Branca de Neve. . .



256120.08

Atlanta

14

### Tipos de Variáveis (continuação)

O slide ilustra os seguintes tipos de dados:

- TRUE representa um valor booleano.
- 15-JAN-09 representa uma DATE.
- A imagem representa um BLOB.
- O texto do callout pode representar um tipo de dados VARCHAR2 ou um CLOB.
- 256120.08 representa um tipo de dados NUMBER com precisão e escala.
- A bobina do filme representa um BFILE.
- O nome da cidade *Atlanta* representa um tipo de dados VARCHAR2.

## Diretrizes para Declarar e Inicializar Variáveis PL/SQL

- Siga convenções de nomeação consistentes.
- Use identificadores significativos para as variáveis.
- Inicialize variáveis que sejam designadas como `NOT NULL` e `CONSTANT`.
- Inicialize as variáveis com o operador de designação (`:=`) ou a palavra-chave `DEFAULT`:

```
v_myName VARCHAR2(20) := 'John';
```

```
v_myName VARCHAR2(20) DEFAULT 'John';
```

- Declare um identificador por linha para obter melhor legibilidade e manutenção de código.

15

### Diretrizes para Declarar e Inicializar Variáveis PL/SQL

Estas são algumas diretrizes a serem seguidas quando você declara variáveis PL/SQL.


- Siga convenções de nomeação consistentes — por exemplo, use `name` para representar uma variável e `c_name` para representar uma constante. De modo similar, para nomear uma variável, você pode usar `v_fname`. O segredo é aplicar a convenção de nomeação de modo consistente para facilitar a identificação.
- Use identificadores significativos e adequados para as variáveis. Por exemplo, considere o uso de `salary` e `sal_with_commission` em vez de `salary1` e `salary2`.
- Se você usar a constraint `NOT NULL`, deverá atribuir um valor quando declarar a variável.
- Na declaração de constantes, a palavra-chave `CONSTANT` deve preceder o especificador de tipo. A declaração a seguir nomeia uma constante do tipo `NUMBER` e designa o valor de 50.000 à constante. Uma constante deve ser inicializada na sua declaração; caso contrário haverá um erro de compilação. Após inicializar uma constante, não é possível alterar o seu valor.

```
sal CONSTANT NUMBER := 50000.00;
```

## Diretrizes para Declarar Variáveis PL/SQL

- Evite usar nomes de colunas como identificadores.

```
DECLARE
  employee_id NUMBER(6);
BEGIN
  SELECT  employee_id
  INTO    employee_id
  FROM    employees
  WHERE   last_name = 'Kochhar';
END;
/
```



- Use a constraint NOT NULL se a variável precisar armazenar um valor.

16

## Diretrizes para Declarar Variáveis PL/SQL

- Inicialize a variável em uma expressão com o operador de designação (:=) ou com a palavra reservada DEFAULT. Se você não designar um valor inicial, a nova variável conterá NULL por default até que você designe um valor. Para designar ou redesignar um valor a uma variável, crie uma instrução de atribuição PL/SQL. No entanto, é recomendável inicializar todas as variáveis.
- Dois objetos podem ter o mesmo nome somente se estiverem definidos em blocos diferentes. Se eles coexistirem, será possível qualificá-los com labels e usá-los.
- Evite usar nomes de colunas como identificadores. Se as variáveis PL/SQL ocorrerem em instruções SQL e tiverem o mesmo nome que uma coluna, o Oracle Server pressuporá que é a coluna que está sendo referenciada. Embora o código de exemplo do slide funcione, um código criado com o mesmo nome em uma tabela do banco de dados e em uma variável não é de fácil leitura ou manutenção.
- Imponha a constraint NOT NULL se a variável precisar conter um valor. Você não pode designar nulos a uma variável definida como NOT NULL. A constraint NOT NULL deve ser seguida por uma cláusula de inicialização.  
`pincode VARCHAR2(15) NOT NULL := 'Oxford';`



## Convenções de Nomeação das Estruturas PL/SQL Usadas Neste Curso

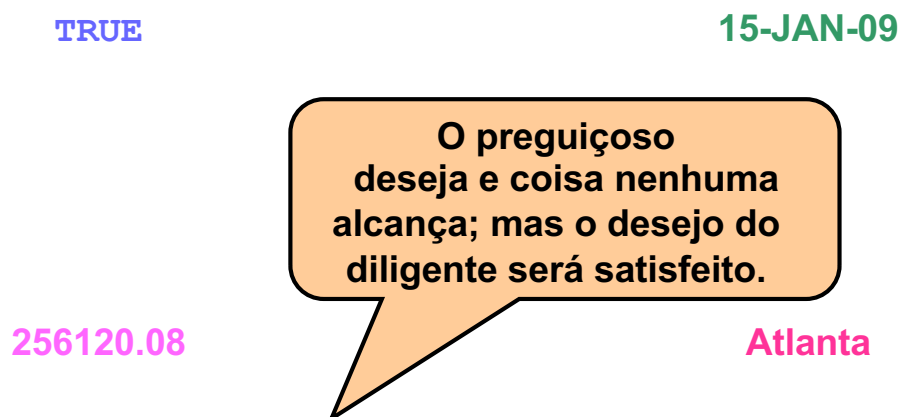
Estrutura PL/SQL	Convenção	Exemplo
Variável	<i>v_variable_name</i>	v_rate
Constante	<i>c_constant_name</i>	c_rate
Parâmetro de subprograma	<i>p_parameter_name</i>	p_id
Variável de bind (host)	<i>b_bind_name</i>	b_salary
Cursor	<i>cur_cursor_name</i>	cur_emp
Registro	<i>rec_record_name</i>	rec_emp
Tipo	<i>type_name_type</i>	ename_table_type
Exceção	<i>e_exception_name</i>	e_products_invalid
Handle de arquivo	<i>f_file_handle_name</i>	f_file

### Convenções de Nomeação das Estruturas PL/SQL Usadas Neste Curso

A tabela do slide exibe alguns exemplos das convenções de nomeação das estruturas PL/SQL usadas neste curso.

## Tipos de Dados Escalares

- Armazenam um único valor
- Não possuem componentes internos



### Tipos de Dados Escalares

O código PL/SQL fornece vários tipos de dados predefinidos. Por exemplo, você pode escolher entre os tipos inteiro, ponto flutuante, caractere, booleano, data, conjunto e LOB. Esta lição aborda os tipos básicos que são usados com frequência em programas PL/SQL.

Um tipo de dados escalar armazena um único valor e não possui componentes internos. Os tipos de dados escalares podem ser classificados em quatro categorias: número, caractere, data e booleano. Os tipos de dados de caractere e número têm subtipos que associam um tipo básico a uma constraint. Por exemplo, `INTEGER` e `POSITIVE` são subtipos do tipo básico `NUMBER`.

Para obter mais informações sobre os tipos de dados escalares (bem como uma lista completa), consulte *PL/SQL User's Guide and Reference*.

## Tipos Básicos de Dados Escalares

- CHAR [(maximum\_length)]
- VARCHAR2 (maximum\_length)
- NUMBER [(precision, scale)]
- BINARY\_INTEGER
- PLS\_INTEGER
- BOOLEAN
- BINARY\_FLOAT
- BINARY\_DOUBLE

### Tipos Básicos de Dados Escalares

Tipo de Dados	Descrição
CHAR [(maximum_length)]	Tipo básico de dados de caracteres de tamanho fixo de até 32.767 bytes. Se você não especificar um <i>tamanho máximo</i> , o tamanho default será definido como 1.
VARCHAR2 (maximum_length)	Tipo básico de dados de caracteres de tamanho variável de até 32.767 bytes. Não existe tamanho default para as variáveis e constantes VARCHAR2.
NUMBER [(precision, scale)]	Número com a precisão <i>p</i> e a escala <i>s</i> . A precisão <i>p</i> pode variar de 1 a 38, enquanto a escala <i>s</i> pode variar de -84 a 127.
BINARY_INTEGER	Tipo básico para inteiros entre -2.147.483.647 e 2.147.483.647

## Tipos Básicos de Dados Escalares

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

### Tipos Básicos de Dados Escalares (continuação)

Tipo de Dados	Descrição
DATE	Tipo básico para datas e horários. Os valores DATE incluem o horário do dia em segundos, desde a meia-noite. A faixa para datas está entre 4712 AC e 9999 DC.
TIMESTAMP	O tipo de dados TIMESTAMP, que expande DATE, armazena ano, mês, dia, hora, minuto, segundo e fração de segundo. A sintaxe é <code>TIMESTAMP[ (precision) ]</code> , onde o parâmetro opcional <code>precision</code> especifica o número de dígitos na parte fracional do campo de segundos. Para especificar a precisão, você deve usar um número inteiro na faixa 0–9. O default é 6.
TIMESTAMP WITH TIME ZONE	O tipo de dados TIMESTAMP WITH TIME ZONE, que expande TIMESTAMP, inclui um deslocamento de fuso horário. O deslocamento de fuso horário é a diferença (em horas e minutos) entre o horário local e o UTC (Coordinated Universal Time), conhecido antigamente como GMT (Greenwich Mean Time). A sintaxe é <code>TIMESTAMP[ (precision) ] WITH TIME ZONE</code> , onde o parâmetro opcional <code>precision</code> especifica o número de dígitos na parte fracional do campo de segundos. Para especificar a precisão, você deve usar um número inteiro na faixa 0–9. O default é 6.

## Declarando Variáveis Escalares

Exemplos:

```
DECLARE
  v_emp_job          VARCHAR2(9);
  v_count_loop       BINARY_INTEGER := 0;
  v_dept_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;
  ...
```

23

### Declarando Variáveis Escalares

Os exemplos de declaração de variável mostrados no slide são definidos assim:

- **v\_emp\_job:** Variável para armazenar o cargo de um funcionário
- **v\_count\_loop:** Variável para contar as iterações de um loop; inicializada em 0
- **v\_dept\_total\_sal:** Variável para acumular o salário total de um departamento; inicializada como 0
- **v\_orderdate:** Variável para armazenar a data de entrega de um pedido; inicializada em uma semana a partir de hoje
- **c\_tax\_rate:** Variável constante para a alíquota de imposto (que nunca muda no decorrer do bloco PL/SQL); definida como 8.25
- **v\_valid:** Flag para indicar se um componente dos dados é válido ou não; inicializada como TRUE

## Atributo %TYPE

- É usado para declarar uma variável de acordo com:
  - Uma definição de coluna do banco de dados
  - Outra variável declarada
- Tem como prefixo:
  - A tabela do banco de dados e o nome da coluna
  - O nome da variável declarada

### Atributo %TYPE

Em geral, as variáveis PL/SQL são declaradas para conter e manipular dados armazenados no banco de dados. Ao declarar variáveis PL/SQL para armazenar valores de colunas, assegure-se de que a variável tenha o tipo de dados e a precisão corretos. Caso contrário, ocorrerá um erro PL/SQL durante a execução. Se você precisar projetar subprogramas extensos, isso poderá tomar muito tempo e gerar erros.

Em vez de codificar o tipo de dados e a precisão de uma variável, você pode usar o atributo %TYPE

para declarar uma variável de acordo com outra variável ou coluna do banco de dados declarada anteriormente. O atributo %TYPE é usado com mais frequência quando o valor armazenado na variável tem origem em uma tabela do banco de dados. Se usar o atributo %TYPE para declarar uma variável, você deverá prefixá-lo com a tabela do banco de dados e o nome da coluna. Caso você faça referência a uma variável previamente declarada, use como prefixo o nome da variável declarada anteriormente na variável que está sendo declarada.

## Declarando Variáveis com o Atributo %TYPE

### Sintaxe

```
identifier      table.column_name%TYPE;
```

### Exemplos

```
...  
v_emp_lname      employees.last_name%TYPE;  
...
```

```
...  
v_balance          NUMBER(7,2);  
v_min_balance      v_balance%TYPE := 1000;  
...
```

26

### Declarando Variáveis com o Atributo %TYPE

Declare variáveis para armazenar o sobrenome de um funcionário. A variável `v_emp_lname` é definida para ser do mesmo tipo de dados que a coluna `v_last_name` da tabela `employees`. O atributo `%TYPE` fornece o tipo de dados de uma coluna do banco de dados.

Declare variáveis para armazenar o saldo de uma conta bancária, assim como o saldo mínimo, que é 1.000. A variável `v_min_balance` é definida para ser do mesmo tipo de dados que a variável `v_balance`. O atributo `%TYPE` fornece o tipo de dados de uma variável.

Uma constraint de coluna de banco de dados `NOT NULL` não se aplica a variáveis que são declaradas com `%TYPE`. Portanto, se você declarar uma variável usando o atributo `%TYPE` que usa uma coluna do banco de dados definida como `NOT NULL`, poderá designar o valor `NULL` à variável.

## Declarando Variáveis Booleanas

- Somente os valores `TRUE`, `FALSE` e `NULL` podem ser atribuídos a uma variável booleana.
- As expressões condicionais usam os operadores lógicos `AND` e `OR` e o operador unário `NOT` para verificar os valores da variável.
- As variáveis sempre retornam `TRUE`, `FALSE` ou `NULL`.
- As expressões aritméticas, de caractere ou de data podem ser usadas para retornar um valor booleano.

27

### Declarando Variáveis Booleanas

Com o código PL/SQL, é possível comparar variáveis em instruções SQL e procedurais. Essas comparações, chamadas de expressões booleanas, consistem em expressões simples ou complexas separadas por operadores relacionais. Em uma instrução SQL, é possível usar expressões booleanas para especificar as linhas de uma tabela que são afetadas pela instrução. Em uma instrução procedural, as expressões booleanas são a base do controle condicional. `NULL` significa um valor ausente, não aplicável ou desconhecido.

#### Exemplos

```
emp_sal1 := 50000;  
emp_sal2 := 60000;
```

A seguinte expressão retorna `TRUE`:

```
emp_sal1 < emp_sal2
```

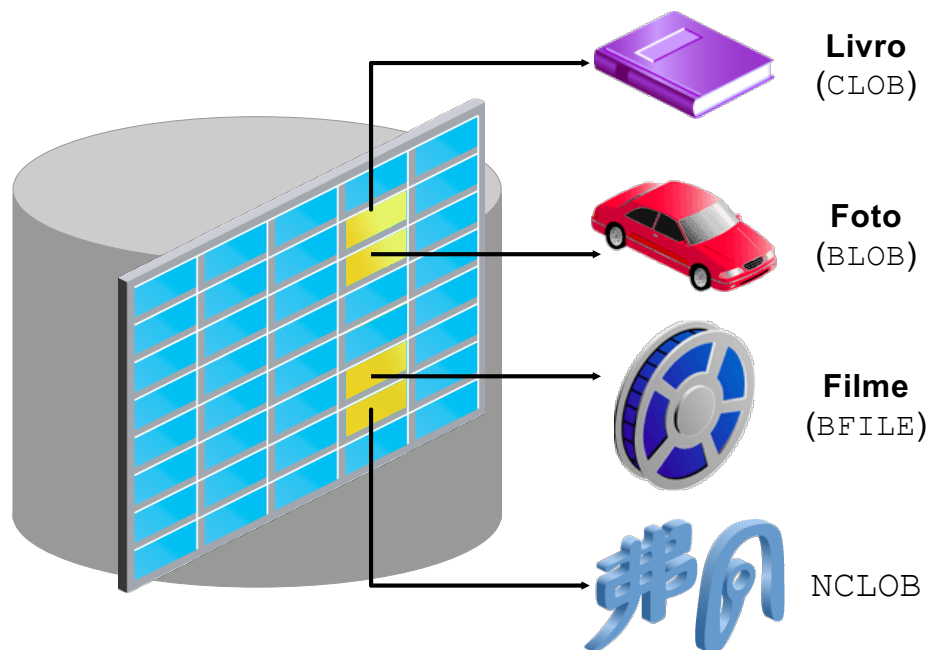
Declare e inicialize uma variável booleana:

```
DECLARE  
    flag BOOLEAN := FALSE;  
BEGIN  
    flag := TRUE;  
END;  
/
```



## Variáveis do Tipo de Dados LOB

FIAP



28


### Variáveis do Tipo de Dados LOB

Os LOBs (large objects) destinam-se a armazenar uma grande quantidade de dados. Uma coluna do banco de dados pode ser da categoria LOB. Com a categoria LOB de tipos de dados (BLOB, CLOB, e assim por diante), você pode armazenar blocos de dados não estruturados (como texto, imagens gráficas, vídeos e formas de ondas sonoras) de até 128 terabytes, dependendo do tamanho do bloco do banco de dados. Os tipos de dados LOB permitem um acesso eficaz, aleatório e em nível de componente aos dados e podem ser atributos de um tipo de objeto.

- O tipo de dados CLOB (character large object) é usado para armazenar blocos grandes de dados<sub>SEP</sub> de caracteres no banco de dados.
- O tipo de dados BLOB (binary large object) é usado para armazenar objetos grandes estruturados ou não no banco de dados. Quando você insere tais dados no banco de dados ou os recupera de lá, o banco de dados não interpreta os dados. As aplicações externas que utilizam esses dados devem interpretá-los.
- O tipo de dados BFILE (binary file) é usado para armazenar arquivos binários grandes. Diferentemente de outros LOBs, BFILES são armazenados fora do banco de dados e não dentro deles. Eles podem ser arquivos do sistema operacional. No banco de dados, será armazenado apenas um ponteiro para o BFILE.
- O tipo de dados NCLOB (national language character large object) é usado para armazenar blocos grandes de dados unicode NCHAR single-byte ou multibyte no banco de dados.

## Tipos de Dados Compostos: Registros e Conjuntos

### Registro PL/SQL:

TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	--

### Conjuntos PL/SQL:

1	SMITH	1	5000
2	JONES	2	2345
3	NANCY	3	12
4	TIM	4	3456

↑ PLS\_INTEGER      ↑ VARCHAR2      ↑ PLS\_INTEGER      ↑ NUMBER

29

## Tipos de Dados Compostos: Registros e Conjuntos

Como mencionado anteriormente, um tipo de dados escalar armazena um único valor e não possui componentes internos. Tipos de dados compostos — chamados Registros PL/SQL e Conjuntos PL/SQL — têm componentes internos que você pode tratar como variáveis individuais.

- Em um registro PL/SQL, os componentes internos podem ser de tipos de dados diferentes e são chamados campos. Você acessa cada campo com esta sintaxe: `record_name.field_name`. Uma variável de registro pode armazenar uma linha de tabela ou algumas colunas de uma linha de tabela. Cada campo de registro corresponde a uma coluna da tabela.
- Em um conjunto PL/SQL, os componentes internos são sempre do mesmo tipo de dados e são chamados elementos. Acesse cada elemento por seu subscript exclusivo. Listas e arrays são exemplos clássicos de conjuntos. Há três tipos de conjuntos PL/SQL: Arrays Associativos, Tabelas Aninhadas e `VARRAY`.

### Observação

- Registros PL/SQL e Arrays Associativos são abordados nesta lição: “Trabalhando com Tipos de Dados Compostos”.
- Os tipos de dados `NESTED TABLE` e `VARRAY` são abordados no curso intitulado *Oracle Database 10g: Advanced PL/SQL* ou *Oracle Database 11g: Advanced PL/SQL*.

## Agenda

- Apresentando variáveis
- Examinando tipos de dados de variável e o atributo %TYPE
- Examinando variáveis de bind

## Variáveis de Bind

As variáveis de bind são:

- Criadas no ambiente
- Também chamadas de variáveis de *host*
- Criadas com a palavra-chave\* `VARIABLE`
- Usadas em instruções SQL e blocos PL/SQL
- Acessadas mesmo após a execução do bloco PL/SQL
- Precedidas por dois-pontos quando referenciadas

Os valores podem ser exibidos com o comando `PRINT`.

\* Necessárias na utilização do SQL\*Plus e SQL Developer

### Variáveis de Bind

As variáveis de bind são as criadas em um ambiente de host. Por esse motivo, às vezes elas são chamadas de variáveis de *host*.

#### Usos de Variáveis de Host

As variáveis de bind são criadas no ambiente e não na seção declarativa de um bloco PL/SQL. Portanto, as variáveis de bind são acessíveis mesmo após a execução do bloco. Quando criadas, as variáveis de bind podem ser usadas e manipuladas por vários subprogramas. Elas podem ser usadas em instruções SQL e blocos PL/SQL como qualquer outra variável. Essas variáveis podem ser especificadas como valores de runtime para subprogramas PL/SQL ou podem ser retornadas por eles.

**Observação:** Uma variável de bind é de ambiente, mas não é uma variável global.

#### Criando Variáveis de Bind

Para criar uma variável de bind no SQL Developer, use o comando `VARIABLE`. Por exemplo, declare uma variável do tipo `NUMBER` e `VARCHAR2` assim:

```
VARIABLE return_code NUMBER  
VARIABLE return_msg VARCHAR2(30)
```

#### Exibindo Valores em Variáveis de Bind

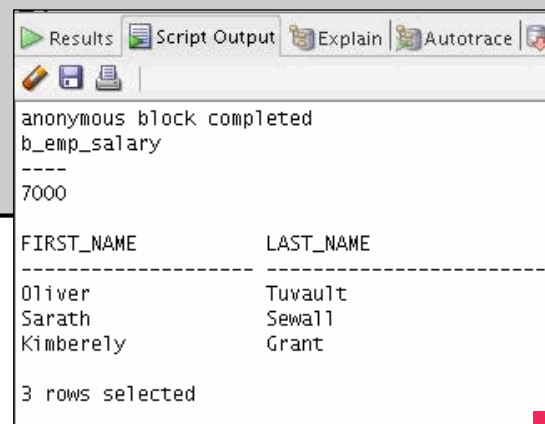
É possível fazer referência à variável de bind usando o SQL Developer e exibir seu valor usando o comando `PRINT`.

## Fazendo Referência a Variáveis de Bind

Exemplo:

```
VARIABLE b_emp_salary NUMBER
BEGIN
    SELECT salary INTO :b_emp_salary
    FROM employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name
FROM employees
WHERE salary=:b_emp_salary;
```

Saída →



FIRST_NAME	LAST_NAME
Oliver	Tuvault
Sarath	Sewall
Kimberely	Grant

3 rows selected

33

### Fazendo Referência a Variáveis de Bind

Como mencionado anteriormente, depois de criar uma variável de bind, você pode fazer referência a ela em qualquer instrução SQL ou programa PL/SQL.

No exemplo, a variável `b_emp_salary` é criada como uma variável de bind no bloco PL/SQL. Em seguida, ela é usada na próxima instrução `SELECT`.

Se você executar o bloco PL/SQL mostrado no slide, obterá a seguinte saída:

- O comando `PRINT` executa:

```
b_emp_salary
-----
7000
```

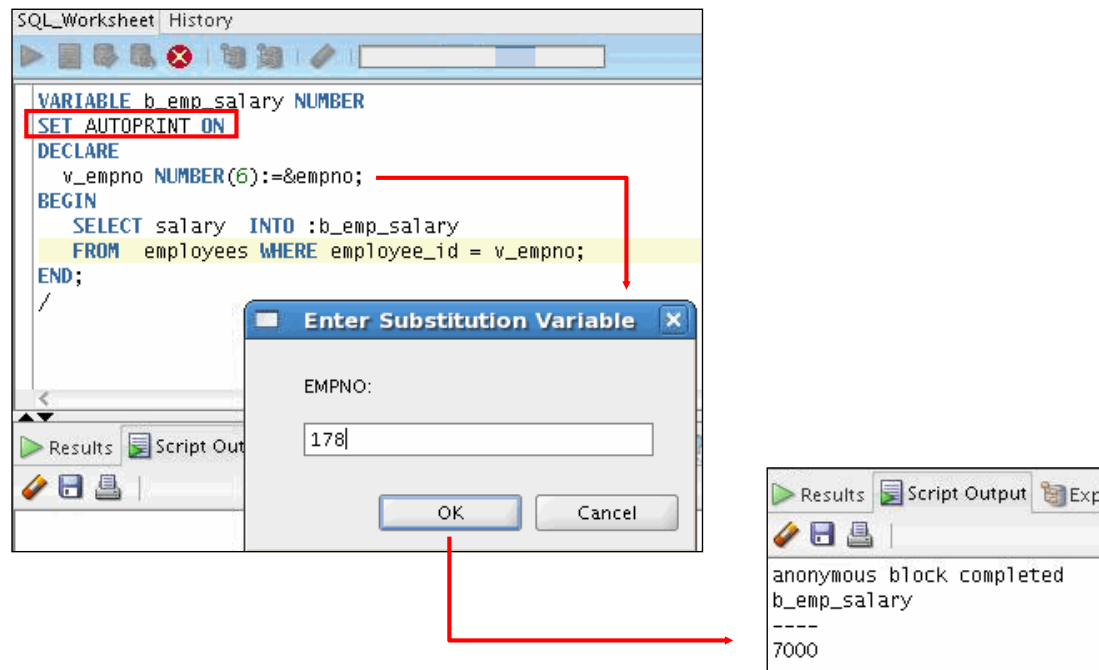
- Em seguida, a saída da instrução SQL é:

```
FIRST_NAME      LAST_NAME
-----
Oliver          Tuvault
Sarath          Sewall
Kimberely       Grant
```

**Observação:** Para exibir todas as variáveis de bind, use o comando `PRINT` sem uma variável.

## Usando AUTOPRINT com Variáveis de Bind

FIAP



34

### Usando AUTOPRINT com Variáveis de Bind

Use o comando `SET AUTOPRINT ON` para exibir automaticamente as variáveis de bind usadas em um bloco PL/SQL bem-sucedido.

#### Exemplo

No código de exemplo:

- Uma variável de bind chamada `b_emp_salary` é criada e `AUTOPRINT` é ativado.
- Uma variável chamada `v_empno` é declarada e uma variável de substituição é usada para receber a entrada do usuário.
- Por último, a variável de bind e as variáveis temporárias são usadas na seção executável do bloco PL/SQL.

Quando um número de funcionário válido é informado — neste caso, 178 — a saída da variável de bind é automaticamente impressa. A variável de bind contém o salário referente ao número do funcionário que o usuário fornece.

## Questionário

O atributo %TYPE:

- a. É usado para declarar uma variável de acordo com uma definição de coluna do banco de dados
- b. É usado para declarar uma variável de acordo com um conjunto de colunas de uma tabela ou view do banco de dados
- c. É usado para declarar uma variável de acordo com a definição de outra variável declarada
- d. Tem como prefixo o nome da tabela do banco de dados e o nome da coluna ou da variável declarada

35

**Resposta: a, c, d**

### O Atributo %TYPE

Em geral, as variáveis PL/SQL são declaradas para conter e manipular dados armazenados no banco de dados. Ao declarar variáveis PL/SQL para armazenar valores de colunas, assegure-se de que a variável tenha o tipo de dados e a precisão corretos. Caso contrário, ocorrerá um erro PL/SQL durante a execução. Se você precisar projetar subprogramas extensos, isso poderá tomar muito tempo e gerar erros.

Em vez de codificar o tipo de dados e a precisão de uma variável, você pode usar o atributo %TYPE para declarar uma variável de acordo com outra variável ou coluna do banco de dados <sup>[1]</sup> declarada anteriormente. O atributo %TYPE é usado com mais frequência quando o valor armazenado na variável tem origem em uma tabela do banco de dados. Se usar o atributo %TYPE para declarar uma variável, você deverá prefixá-lo com a tabela do banco de dados e o nome da coluna. Caso você faça referência a uma variável previamente declarada, use como prefixo o nome da variável declarada anteriormente na variável que está sendo declarada. A vantagem de %TYPE é que não será necessário alterar a variável caso a coluna seja alterada. Além disso, se a variável for usada em um cálculo, você não precisará se preocupar com a sua precisão.

### O Atributo %ROWTYPE

O atributo %ROWTYPE é usado para declarar um registro que possa armazenar uma linha inteira de uma tabela ou view. Você aprenderá sobre esse atributo na lição “Trabalhando com Tipos de Dados Compostos.”

## Sumário

Nesta lição, você aprendeu a:

- Reconhecer identificadores válidos e inválidos
- Declarar variáveis na seção declarativa de um bloco PL/SQL
- Inicializar variáveis e utilizá-las na seção executável
- Diferenciar tipos de dados escalares e compostos
- Usar o atributo `%TYPE`
- Usar variáveis de bind

36

### Sumário

Um bloco PL/SQL anônimo é uma unidade básica não nomeada de um programa PL/SQL. Ele consiste em um conjunto de instruções SQL ou PL/SQL para executar uma função lógica. A parte declarativa é a primeira parte de um bloco PL/SQL e é usada para declarar objetos como variáveis, constantes, cursors e definições de situações de erro chamadas de *exceções*.

Nesta lição, você aprendeu a declarar variáveis na seção declarativa. Você conheceu algumas diretrizes para declaração de variáveis. E aprendeu a inicializar variáveis ao declará-las.

A parte executável de um bloco PL/SQL é a parte obrigatória e contém instruções SQL e PL/SQL para consultar e manipular dados. Você aprendeu a inicializar variáveis na seção executável, a como utilizá-las e a manipular seus valores.



## Exercício 2: Visão Geral

Este exercício aborda os seguintes tópicos:

- Definição de identificadores válidos
- Definição de declarações de variáveis válidas
- Declaração de variáveis dentro de um bloco anônimo
- Uso do atributo `%TYPE` para declarar variáveis
- Declaração e exibição de uma variável de bind
- Execução de um bloco PL/SQL

### Exercício 2: Visão Geral

Os exercícios 1, 2 e 3 são impressos.