

FIAP GRADUAÇÃO

# Tecnologia em Análise e Desenvolvimento de Sistemas

Application Development For Databases

PROF. MILTON

## **Interagindo com o Oracle Database Server: Instruções SQL em Programas PL/SQL**

## Objetivos

Ao concluir esta lição, você será capaz de:

- Determinar quais instruções SQL podem ser incluídas diretamente em um bloco executável PL/SQL
- Manipular dados com instruções DML em blocos PL/SQL
- Usar instruções de controle de transação em blocos PL/SQL
- Usar a cláusula `INTO` para armazenar os valores retornados por uma instrução SQL
- Fazer a distinção entre cursores explícitos e implícitos
- Usar atributos de cursores SQL

### Objetivos

Nesta lição, você aprenderá a incorporar instruções-padrão `SELECT`, `INSERT`, `UPDATE`, `DELETE` e `MERGE` em blocos PL/SQL. Aprenderá a incluir códigos DDL (Data Definition Language) e instruções de controle de transação em PL/SQL. Você compreenderá a necessidade dos cursores e as diferenças entre os dois tipos de cursores. A lição também apresentará os vários atributos de cursores SQL que podem ser usados com cursores implícitos.

## Agenda

- Recuperando dados com PL/SQL
- Manipulando dados com PL/SQL
- Apresentando cursores SQL

## Instruções SQL em Blocos PL/SQL

- Recupere uma linha do banco de dados usando o comando `SELECT`.
- Faça alterações em linhas do banco de dados usando comandos DML.
- Controle uma transação com o comando `COMMIT`, `ROLLBACK` ou `SAVEPOINT`.

### Instruções SQL em Blocos PL/SQL

Em um bloco PL/SQL, use instruções SQL para recuperar e modificar dados da tabela do banco de dados. O bloco PL/SQL suporta comandos DML (Data Manipulation Language) e de controle de transações. Você pode usar comandos DML para modificar os dados contidos em uma tabela do banco de dados. No entanto, lembre-se do seguinte ao usar instruções DML e comandos de controle de transação em blocos PL/SQL:

- A palavra-chave `END` sinaliza o fim de um bloco PL/SQL, e não o fim de uma transação. Da mesma forma que um bloco pode abranger várias transações, uma transação pode abranger vários blocos.
- O bloco PL/SQL não suporta diretamente instruções DDL (data definition language), como `CREATE TABLE`, `ALTER TABLE` ou `DROP TABLE`. Há suporte para Early Binding na linguagem PL/SQL, mas esse suporte não ocorrerá se as aplicações precisarem criar objetos de banco de dados passando valores durante o runtime. As instruções DDL não podem ser executadas diretamente. Elas são instruções SQL dinâmicas. As instruções SQL dinâmicas são construídas como strings de caracteres durante o runtime e podem conter placeholders para parâmetros. Em consequência, você pode usar instruções SQL dinâmicas para executar suas instruções DDL no bloco PL/SQL. Os detalhes sobre como trabalhar com SQL dinâmico são abordados no curso *Oracle Database: Desenvolvimento de Unidades de Programa PL/SQL*.
- O bloco PL/SQL não suporta diretamente instruções DCL (data control language), como `GRANT` ou `REVOKE`. Você pode usar SQL dinâmico para executá-las.

# Instruções **SELECT** em Blocos PL/SQL

Recupere dados do banco de dados com uma instrução **SELECT**.

Sintaxe:

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

## Instruções **SELECT** em Blocos PL/SQL

Use a instrução **SELECT** para recuperar dados do banco de dados.

<i>select_list</i>	Lista de pelo menos uma coluna; pode incluir expressões SQL, funções de linha ou funções de grupo
<i>variable_name</i>	Variável escalar que contém o valor recuperado
<i>record_name</i>	Registro PL/SQL que contém os valores recuperados
<i>table</i>	Especifica o nome da tabela do banco de dados
<i>condition</i>	É composto de nomes de colunas, expressões, constantes e operadores de comparação, incluindo constantes e variáveis PL/SQL

### Diretrizes para Recuperar Dados em Blocos PL/SQL

- Encerre cada instrução SQL com um ponto-e-vírgula (;).
- Todo valor recuperado deve ser armazenado em uma variável por meio da cláusula **INTO**.
- A cláusula **WHERE** é opcional e pode ser usada para especificar variáveis, constantes, literais ou expressões PL/SQL de entrada. Entretanto, ao usar a cláusula **INTO**, você deverá extrair com o comando **fetch** apenas uma linha, e o uso da cláusula **WHERE** é necessário nesses casos.

# Instruções SELECT em Blocos PL/SQL

- A cláusula INTO é obrigatória.
- As consultas devem retornar apenas uma linha.

```
DECLARE
  v_fname VARCHAR2(25);
BEGIN
  SELECT first_name INTO v_fname
  FROM employees WHERE employee_id=200;
  DBMS_OUTPUT.PUT_LINE(' First Name is : '||v_fname);
END;
/
```

```
anonymous block completed
First Name is : Jennifer
```

## Instruções SELECT em Blocos PL/SQL (continuação)

### Cláusula INTO

A cláusula INTO é obrigatória e ocorre entre as cláusulas SELECT e FROM. Ela é usada para especificar os nomes das variáveis que armazenam os valores que o código SQL retorna da cláusula SELECT. Especifique uma variável para cada item selecionado. A ordem das variáveis deve corresponder aos itens selecionados.

Use a cláusula INTO para preencher as variáveis PL/SQL ou as variáveis de host.

### As Consultas Devem Retornar Apenas uma Linha

As instruções SELECT contidas em um bloco PL/SQL recaem na classificação ANSI de SQL incorporada, para a qual se aplica a seguinte regra: As consultas devem retornar apenas uma linha. Uma consulta que retorne mais de uma linha ou nenhuma linha gera um erro.

O bloco PL/SQL gerencia esses erros gerando exceções-padrão, que podem ser tratadas na seção de exceções do bloco com as exceções NO\_DATA\_FOUND e TOO\_MANY\_ROWS. Inclua uma condição WHERE na instrução SQL para que a instrução retorne uma única linha. Você aprenderá sobre o tratamento de exceções na lição “Tratando Exceções”.

**Observação:** Em todos os casos em que DBMS\_OUTPUT.PUT\_LINE é usada nos códigos de exemplo, a instrução SET SERVEROUTPUT ON antecede o bloco.



## Recuperando Dados em Blocos PL/SQL: Exemplo

Recupere `hire_date` e `salary` para o funcionário especificado.

```
DECLARE
  v_emp_hiredate    employees.hire_date%TYPE;
  v_emp_salary      employees.salary%TYPE;
BEGIN
  SELECT    hire_date, salary
  INTO      v_emp_hiredate, v_emp_salary
  FROM employees
  WHERE employee_id = 100;
  DBMS_OUTPUT.PUT_LINE ('Hire date is : ' || v_emp_hiredate);
  DBMS_OUTPUT.PUT_LINE ('Salary is : ' || v_emp_salary);
END;
/
```

```
anonymous block completed
Hire date is : 17-JUN-87
Salary is : 24000
```

11

### Recuperando Dados em Blocos PL/SQL

No exemplo do slide, as variáveis `v_emp_hiredate` e `v_emp_salary` são declaradas na seção declarativa do bloco PL/SQL. Na seção executável, os valores das colunas `hire_date` e `salary` referentes ao funcionário cujo `employee_id` é 100 são recuperados da tabela `employees`. Em seguida, eles são armazenados nas variáveis `emp_hiredate` e `emp_salary`, respectivamente. Observe como a cláusula `INTO`, juntamente com a instrução `SELECT`, recupera os valores das colunas do banco de dados e armazena-os nas variáveis PL/SQL.

**Observação:** A instrução `SELECT` recupera `hire_date` e, em seguida, `salary`. As variáveis da cláusula `INTO` devem, portanto, estar na mesma ordem. Por exemplo, se você trocar `v_emp_hiredate` e `v_emp_salary` na instrução do slide, a instrução resultará em um erro.

## Recuperando Dados em Blocos PL/SQL FIAP

Retorne a soma dos salários de todos os funcionários do departamento especificado.

Exemplo:

```
DECLARE
    v_sum_sal    NUMBER(10,2);
    v_deptno     NUMBER NOT NULL := 60;
BEGIN
    SELECT SUM(salary) -- group function
    INTO v_sum_sal FROM employees
    WHERE department_id = v_deptno;
    DBMS_OUTPUT.PUT_LINE ('The sum of salary is ' || v_sum_sal);
END;
```

```
anonymous block completed
The sum of salary is 28800
```

12

### Recuperando Dados em Blocos PL/SQL (continuação)

No exemplo do slide, as variáveis `v_sum_sal` e `v_deptno` são declaradas na seção declarativa do bloco PL/SQL. Na seção executável, o salário total dos funcionários do departamento com o `department_id` 60 é calculado com base na função agregada SQL `SUM`. O salário total calculado é designado à variável `v_sum_sal`.

**Observação:** As funções de grupo não podem ser usadas na sintaxe PL/SQL. Elas devem ser usadas em instruções SQL dentro de um bloco PL/SQL, como mostrado no exemplo do slide.

Por exemplo, você *não pode* usar as funções de grupo com a seguinte sintaxe:

```
v_sum_sal := SUM(employees.salary);
```

## Ambiguidades de Nomeação

```

DECLARE
    hire_date      employees.hire_date%TYPE;
    sysdate        hire_date%TYPE;
    employee_id    employees.employee_id%TYPE := 176;
BEGIN
    SELECT          hire_date, sysdate
    INTO            hire_date, sysdate
    FROM            employees
    WHERE           employee_id = employee_id;
END;
/

```

```

Error report:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
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

```

13

### Ambiguidades de Nomeação

Em instruções SQL potencialmente ambíguas, os nomes das colunas dos bancos de dados têm precedência sobre os nomes das variáveis locais.

O exemplo mostrado no slide é definido assim: Recupere a data de admissão e a data de hoje da tabela `employees` para o funcionário cujo `employee_id` é 176. Esse exemplo gera uma exceção durante o runtime não tratada porque, na cláusula `WHERE`, os nomes das variáveis PL/SQL são iguais aos nomes das colunas do banco de dados na tabela `employees`.

A instrução `DELETE` a seguir remove todos os funcionários da tabela `employees`, cujo sobrenome não seja nulo (não apenas “King”), porque o Oracle Server pressupõe que ambas as ocorrências de `last_name` na cláusula `WHERE` se referem à coluna do banco de dados:

```

DECLARE
    last_name VARCHAR2(25) := 'King';
BEGIN
    DELETE FROM employees WHERE last_name = last_name;
    . . .

```

## Convenções de Nomeação

- Use uma convenção de nomeação para evitar ambiguidade na cláusula `WHERE`.
- Evite usar nomes de colunas do banco de dados como identificadores.
- Podem ocorrer erros de sintaxe porque o código PL/SQL verifica primeiro se há uma coluna na tabela do banco de dados.
- Os nomes de variáveis locais e parâmetros formais têm precedência sobre os nomes de *tabelas* do banco de dados.
- Os nomes de *colunas* de tabelas do banco de dados têm precedência sobre os nomes de variáveis locais.

14

### Convenções de Nomeação

Evite ambiguidade na cláusula `WHERE` seguindo uma convenção de nomeação que diferencie os nomes de colunas do banco de dados dos nomes de variáveis PL/SQL.

- As colunas de banco de dados e os identificadores devem ter nomes distintos.
- Podem ocorrer erros de sintaxe porque o código PL/SQL verifica primeiro se há uma coluna na tabela do banco de dados.

**Observação:** Não há possibilidade de ambiguidade na cláusula `SELECT` pois qualquer identificador na cláusula `SELECT` deve ser um nome de coluna do banco de dados. Não há possibilidade de ambiguidade na cláusula `INTO` pois os identificadores na cláusula `INTO` devem ser variáveis PL/SQL. Só há possibilidade de confusão na cláusula `WHERE`.

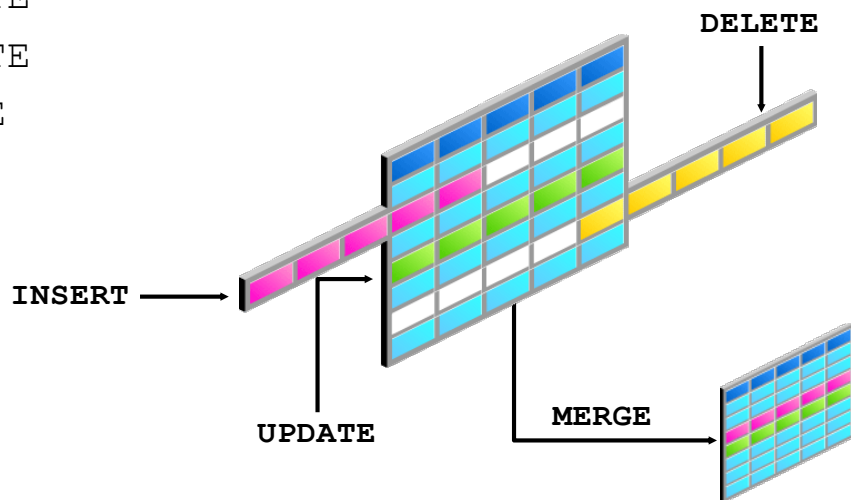
## Agenda

- Recuperando dados com PL/SQL
- **Manipulando dados com PL/SQL**
- Apresentando cursores SQL

## Usando o Código PL/SQL para Manipular Dados

Faça alterações em tabelas do banco de dados usando comandos DML:

- INSERT
- UPDATE
- DELETE
- MERGE



16

### Usando o Código PL/SQL para Manipular Dados

Manipule dados no banco de dados usando comandos DML. Você pode executar comandos DML como INSERT, UPDATE, DELETE e MERGE sem restrição no código PL/SQL. Os bloqueios de linha (e os bloqueios de tabelas) são liberados quando instruções COMMIT ou ROLLBACK são incluídas no código PL/SQL.

- A instrução INSERT adiciona novas linhas à tabela.
- A instrução UPDATE modifica linhas existentes na tabela.
- A instrução DELETE remove linhas da tabela.
- A instrução MERGE seleciona linhas de uma tabela a serem atualizadas ou inseridas em outra tabela. A decisão de efetuar uma atualização ou inserção na tabela de destino se baseia em uma condição na cláusula ON.

**Observação:** MERGE é uma instrução determinante. Quer dizer, não é possível atualizar a mesma linha da tabela de destino várias vezes na mesma instrução MERGE. Você deve ter os privilégios de objeto INSERT e UPDATE na tabela de destino e o privilégio SELECT na tabela de origem.

## Inserindo Dados: Exemplo

Adicione informações sobre um novo funcionário à tabela EMPLOYEES.

```
BEGIN
  INSERT INTO employees
    (employee_id, first_name, last_name, email,
     hire_date, job_id, salary)
    VALUES (employees_seq.NEXTVAL, 'Ruth', 'Cores',
            'RCORES', CURRENT_DATE, 'AD_ASST', 4000);
END;
/
```

17

### Inserindo Dados

No exemplo do slide, uma instrução `INSERT` é usada dentro de um bloco PL/SQL para inserir um registro na tabela `employees`. Ao usar o comando `INSERT` em um bloco PL/SQL, você pode:

- Usar funções SQL como `USER` e `CURRENT_DATE`
- Gerar valores de chave primária usando sequências existentes de banco de dados
- Derivar valores no bloco PL/SQL

**Observação:** Os dados da tabela `employees` devem permanecer inalterados. Embora a tabela `employees` não seja somente para leitura, não é permitido fazer inserções, atualizações e deleções nesta tabela para garantir a consistência de saída, como mostrado no arquivo `code_04_15_s.sql` do código de exemplo.

## Atualizando Dados: Exemplo

Aumente o salário de todos os funcionários que trabalham no estoque.

```
DECLARE
    sal_increase    employees.salary%TYPE := 800;
BEGIN
    UPDATE          employees
    SET              salary = salary + sal_increase
    WHERE           job_id = 'ST_CLERK';
END;
/
```

```
anonymous block completed
FIRST_NAME      SALARY
-----
Julia           4000
Irene           3500
James           3200
Steven          3000
```

. . .

```
Curtis          3900
Randall         3400
Peter           3300
```

20 rows selected

18

### Atualizando Dados

É possível haver ambiguidade na cláusula SET da instrução UPDATE porque, embora o identificador à esquerda do operador de designação seja sempre uma coluna do banco de dados, o identificador à direita pode ser uma coluna do banco de dados ou uma variável PL/SQL. Lembre-se de que, se os nomes de colunas e de identificadores foram iguais na cláusula WHERE, o Oracle Server primeiro procurará o nome no banco de dados.

Lembre-se de que a cláusula WHERE é usada para determinar as linhas que são afetadas. Se nenhuma linha for modificada, não ocorrerá erro (ao contrário do que acontece com a instrução SELECT no código PL/SQL).

**Observação:** As designações de variáveis PL/SQL sempre usam := e as designações de colunas SQL sempre usam =.



## Deletando Dados: Exemplo

Delete linhas que pertencem ao departamento 10 da tabela `employees`.

```
DECLARE
    deptno    employees.department_id%TYPE := 10;
BEGIN
    DELETE FROM employees
    WHERE     department_id = deptno;
END;
/
```

19

### Deletando Dados

A instrução `DELETE` remove linhas indesejadas da tabela. Se a cláusula `WHERE` não for usada, todas as linhas de uma tabela poderão ser removidas desde que não haja constraints de integridade.

## Intercalando Linhas

Insira ou atualize as linhas na tabela `COPY_EMP3` para que corresponda à tabela `employees`.

```
BEGIN
MERGE INTO copy_emp c
  USING employees e
  ON (e.employee_id = c.empno)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name = e.first_name,
      c.last_name = e.last_name,
      c.email = e.email,
      . . .
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
      . . . ,e.department_id);
END;
/
```

20

### Intercalando Linhas

A instrução `MERGE` insere ou atualiza linhas de uma tabela usando dados de outra tabela. Cada linha é inserida ou atualizada na tabela de destino, de acordo com uma condição equijoin.

O exemplo mostrado faz a correspondência da coluna `empno` da tabela `copy_emp` com a coluna `employee_id` da tabela `employees`. Se for encontrada uma correspondência, a linha será atualizada para corresponder à linha da tabela `employees`. Se a linha não for encontrada, ela será inserida na tabela `copy_emp`.

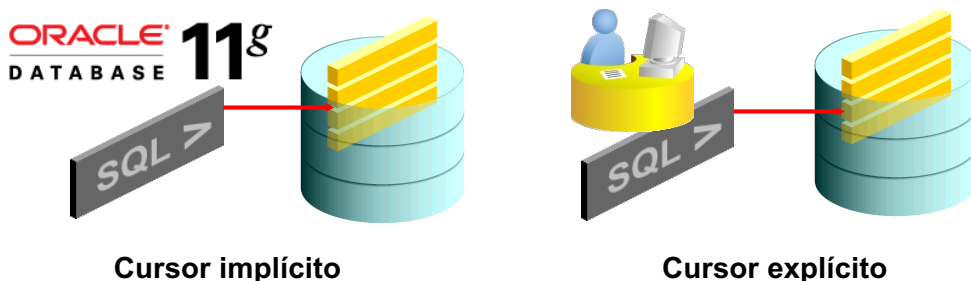
O exemplo completo de uso da instrução `MERGE` em um bloco PL/SQL é mostrado na próxima página.

## Agenda

- Recuperando dados com PL/SQL
- Manipulando dados com PL/SQL
- Apresentando cursores SQL

## Cursor SQL

- Um cursor é um ponteiro para a área de memória particular alocada pelo Oracle Server. Ele é usado para tratar o conjunto de resultados de uma instrução `SELECT`.
- Há dois tipos de cursores: implícitos e explícitos.
  - **Implícitos:** Criados e gerenciados internamente pelo Oracle Server para processar instruções SQL
  - **Explícitos:** Declarados explicitamente pelo programador



### Cursor SQL

Você já aprendeu que é possível incluir instruções SQL em um bloco PL/SQL para retornar uma única linha. Os dados recuperados pela instrução SQL devem ser armazenados em variáveis por meio da cláusula `INTO`.

#### Onde o Oracle Server Processa as Instruções SQL?

O Oracle Server aloca uma área de memória particular chamada *área de contexto* para processar instruções SQL. É efetuado parse na instrução SQL e ela é processada nessa área. Todas as informações necessárias para o processamento e aquelas recuperadas após o processamento são armazenadas nessa área. Você não tem controle sobre essa área porque ela é internamente gerenciada pelo Oracle Server.

Um cursor é um ponteiro para a área de contexto. No entanto, esse cursor é implícito e gerenciado automaticamente pelo Oracle Server. Quando o bloco executável executa uma instrução SQL, o bloco PL/SQL cria um cursor implícito.

#### Tipos de Cursores

Há dois tipos de cursores:

- **Implícitos:** Um *cursor implícito* é criado e gerenciado pelo Oracle Server. Você não tem acesso a ele. O Oracle Server cria esse cursor quando precisa executar uma instrução SQL.

## Atributos de Cursores SQL para Cursores Implícitos

Usando os atributos de cursores SQL, você pode testar o resultado das instruções SQL.

<b>SQL%FOUND</b>	Atributo booleano que será avaliado como <b>TRUE</b> se a instrução SQL mais recente afetar pelo menos uma linha
<b>SQL%NOTFOUND</b>	Atributo booleano que será avaliado como <b>TRUE</b> se a instrução SQL mais recente não afetar nenhuma linha
<b>SQL%ROWCOUNT</b>	Um valor inteiro que representa o número de linhas afetadas pela instrução SQL mais recente

### Atributos de Cursores SQL para Cursores Implícitos

Os atributos de cursores SQL permitem avaliar o que aconteceu quando um cursor implícito foi usado pela última vez. Utilize esses atributos em instruções PL/SQL, mas não em instruções SQL.

Você pode testar os atributos `SQL%ROWCOUNT`, `SQL%FOUND` e `SQL%NOTFOUND` na seção executável de um bloco para reunir informações após o comando DML apropriado ser executado. O código PL/SQL não retornará erro se uma instrução DML não afetar as linhas na tabela subjacente. Entretanto, se uma instrução `SELECT` não recuperar nenhuma linha, o código PL/SQL retornará uma exceção.

Observe que os atributos têm o prefixo `SQL`. Esses atributos de cursores são usados nos cursores implícitos que são criados automaticamente pelo código PL/SQL e cujos nomes você não conhece. Por isso, é usado `SQL` em vez do nome do cursor.

O atributo `SQL%NOTFOUND` é o oposto de `SQL%FOUND`. Esse atributo pode ser usado como a condição de saída de um loop. Isso é útil em instruções `UPDATE` e `DELETE` quando nenhuma linha é alterada, pois não são retornadas exceções nesses casos.

Você aprenderá sobre os atributos de cursores explícitos na lição “Usando Cursores Explícitos”.

## Atributos de Cursores SQL para Cursores Implícitos

Delete linhas que possuem o ID de funcionário especificado da tabela `employees`. Imprima o número de linhas deletadas.

Exemplo:

```
DECLARE
  v_rows_deleted VARCHAR2(30)
  v_empno employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM employees
  WHERE employee_id= v_empno;
  v_rows_deleted := (SQL%ROWCOUNT ||
                    ' row deleted. ');
  DBMS_OUTPUT.PUT_LINE (v_rows_deleted);
END;
```

26

### Atributos de Cursores SQL para Cursores Implícitos (continuação)

O exemplo no slide deleta uma linha com `employee_id 176` da tabela `employees`. Com o atributo `SQL%ROWCOUNT`, você pode imprimir o número de linhas deletadas.

## Questionário

FIAP

Quando você usa a instrução `SELECT` em blocos PL/SQL, a cláusula `INTO` é necessária e as consultas podem retornar uma ou mais linhas.

- a. Verdadeiro
- b. Falso

27

**Resposta: b**

### **Cláusula `INTO`**

A cláusula `INTO` é obrigatória e ocorre entre as cláusulas `SELECT` e `FROM`. Ela é usada para especificar os nomes das variáveis que armazenam os valores que o código SQL retorna da cláusula `SELECT`. Especifique uma variável para cada item selecionado. A ordem das variáveis deve corresponder aos itens selecionados.

Use a cláusula `INTO` para preencher as variáveis PL/SQL ou as variáveis de host.

### **As Consultas Devem Retornar Apenas uma Linha**

**As instruções `SELECT`** contidas em um bloco PL/SQL recaem na classificação ANSI de SQL incorporada, para a qual se aplica a seguinte regra: As consultas devem retornar apenas uma linha. Uma consulta que retorne mais de uma linha ou nenhuma linha gera um erro.

O bloco PL/SQL gerencia esses erros gerando exceções-padrão, que podem ser tratadas na seção de exceções do bloco com as exceções `NO_DATA_FOUND` e `TOO_MANY_ROWS`. Inclua uma condição `WHERE` na instrução SQL para que a instrução retorne uma única linha. Você aprenderá sobre o tratamento de exceções mais adiante no curso.

Nesta lição, você aprendeu a:

- Incorporar instruções DML, instruções de controle de transações e instruções DDL aos blocos PL/SQL
- Usar a cláusula `INTO`, obrigatória para todas as instruções `SELECT`, em blocos PL/SQL
- Fazer a distinção entre cursores explícitos e implícitos
- Usar atributos de cursores SQL para determinar o resultado de instruções SQL

### Sumário

Comandos DML e instruções de controle de transação podem ser usados em programas PL/SQL sem restrição. Entretanto, os comandos DDL não podem ser usados diretamente.

Uma instrução `SELECT` de um bloco PL/SQL pode retornar apenas uma linha. É obrigatório o uso da cláusula `INTO` para armazenar os valores recuperados pela instrução `SELECT`.

Um cursor é um ponteiro para a área da memória. Há dois tipos de cursores. Os cursores implícitos são criados e gerenciados internamente pelo Oracle Server para executar as instruções SQL. É possível usar atributos de cursores SQL com esses cursores para determinar o resultado da instrução SQL. Os cursores explícitos são declarados pelos programadores.



## Exercício 4: Visão Geral

Este exercício aborda os seguintes tópicos:

- Selecionando dados de uma tabela
- Inserindo linhas em uma tabela
- Atualizando linhas de uma tabela
- Deletando um registro de uma tabela