Using SQL Statements Within a PL/SQL Block

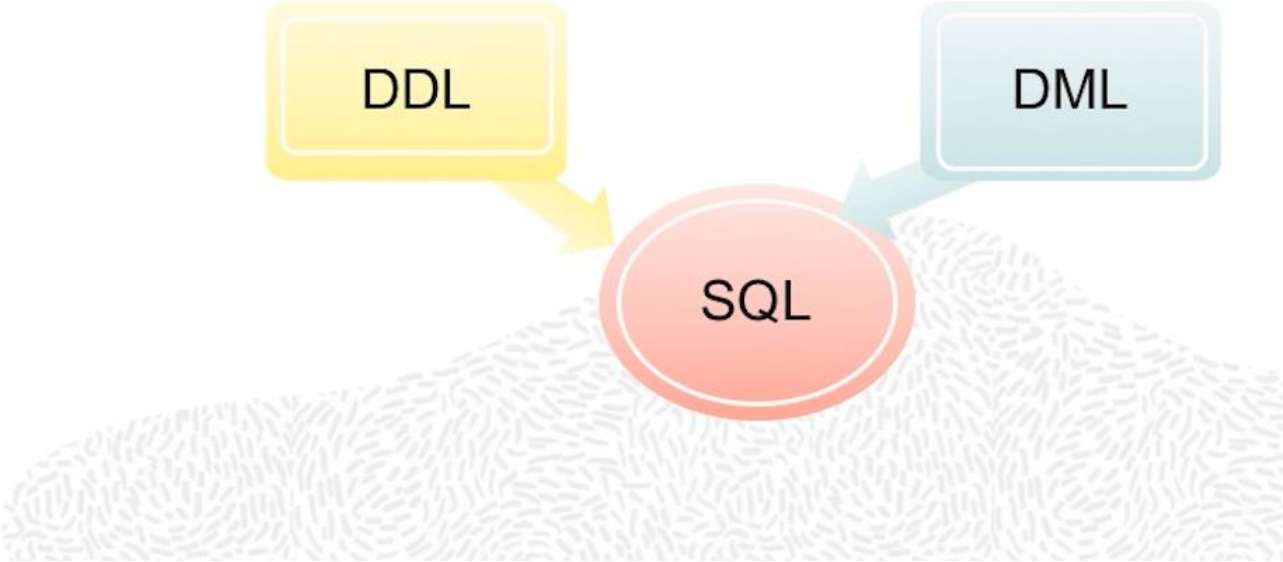
Objectives

After completing this lesson, you should be able to do the following:

- Determine the SQL statements that can be directly included in a PL/SQL executable block
- Make use of the `INTO` clause to hold the values returned by a SQL statement
- Manipulate data with DML statements in PL/SQL
- Use transaction control statements in PL/SQL
- Differentiate between implicit cursors and explicit cursors
- Use SQL cursor attributes

SQL Statements in PL/SQL

- SQL consists of three sub-languages:
  - DDL
  - DML
- DDL statements are generally not included in PL/SQL blocks.
- DML statements are included in PL/SQL blocks.



SELECT Statements in PL/SQL

- The INTO clause is required.
- Queries must return only one row.

```
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;
/
```

```
PL/SQL procedure successfully completed.

First name is :Jennifer
```

Retrieving Data in PL/SQL: Example

Retrieve `hire_date` and `salary` for the specified employee.

```
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;
/
```

```
PL/SQL procedure successfully completed.

Hire date is :17-JUN-11
Salary is :24000
```

Retrieving Data in PL/SQL

Return the sum of salaries for all the employees in the specified department.

Example:

```
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;
```

```
PL/SQL procedure successfully completed.

The sum of salary is :28800
```

Naming Ambiguities

```
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
```

Avoiding Naming Ambiguities

- Use a naming convention to avoid ambiguity in the `WHERE` clause.
- Avoid using database column names as identifiers.
- The names of local variables and formal parameters take precedence over the names of database tables.
- The names of database table columns take precedence over the names of local variables.
- The names of variables take precedence over the function names.

Using PL/SQL to Manipulate Data

Make changes to database tables by using DML commands:

- `INSERT`
- `UPDATE`
- `DELETE`

Insert Data: Example

Add new employee information to the `EMPLOYEES` table.

```
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;
/
```

Update Data: Example

Increase the salary of all employees who are stock clerks.

```
SELECT first_name,salary
FROM employees
WHERE job_id = 'ST_CLERK';
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 20 in 0.001 seconds

	FIRST_NAME	SALARY
1	Julia	3200
2	Irene	2700
3	James	2400
4	Steven	2200
5	Laura	3300

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

```
PL/SQL procedure successfully completed.
```

```
SELECT first_name,salary
FROM employees
WHERE job_id = 'ST_CLERK';
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 20 in 0.003 seconds

	FIRST_NAME	SALARY
1	Julia	4000
2	Irene	3500
3	James	3200
4	Steven	3000
5	Laura	4100

Delete Data: Example

Delete rows that belong to department 10 from the `employees` table.

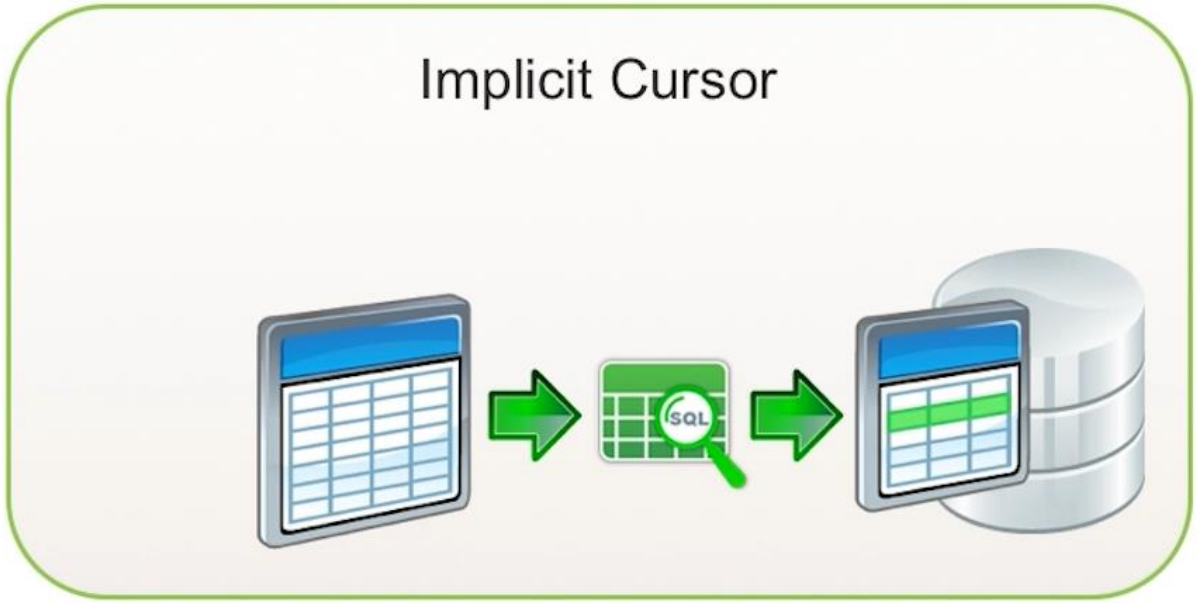
```
DECLARE
  v_emp   employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM   employees
  WHERE  employee_id = v_emp;
END;
/
```

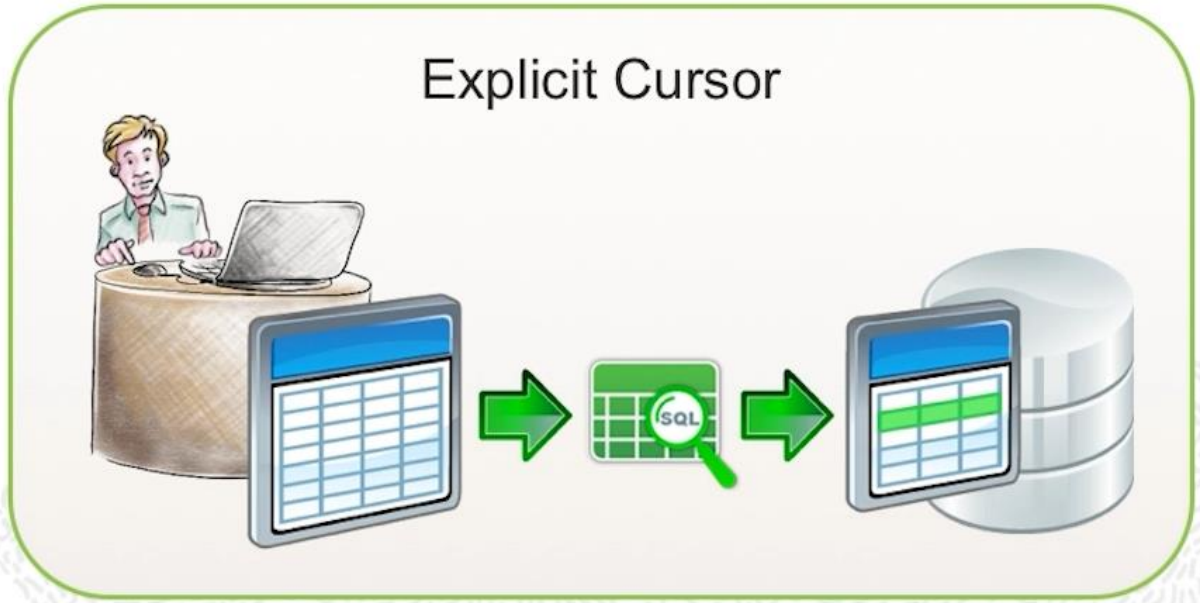
Agenda

- SQL statements in PL/SQL blocks
- Manipulating data with PL/SQL
- Introducing SQL cursors

SQL Cursor

- A cursor is a pointer to the private memory area that stores information about processing a specific `SELECT` or `DML` statement.
- There are two types of cursors:
  - **Implicit:** Created and managed by PL/SQL
  - **Explicit:** Created and managed explicitly by the programmer





SQL Cursor Attributes for Implicit Cursors

Using SQL cursor attributes, you can test the outcome of your SQL statements.

SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement affected at least one row
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement did not affect even one row
SQL%ROWCOUNT	An integer value that represents the number of rows affected by the most recent SQL statement

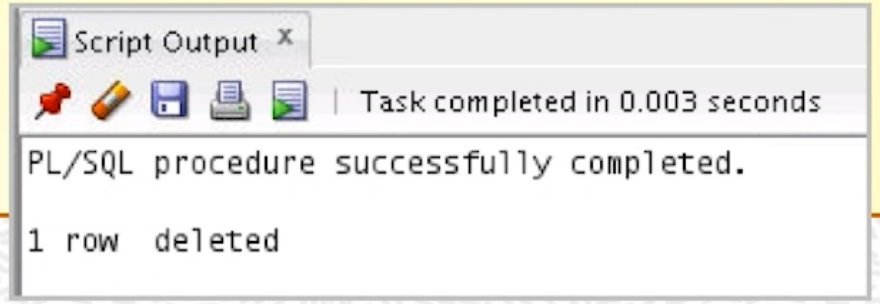
SQL Cursor Attributes for Implicit Cursors

Delete rows that have the specified employee ID from the `employees` table. Print the number of rows deleted.

Example:

```
DECLARE
  v_rows_deleted VARCHAR2(30);
  v_empno employees.employee_id%TYPE := 165;
BEGIN
  DELETE FROM  employees
  WHERE employee_id = v_empno;
  v_rows_deleted := (SQL%ROWCOUNT ||
                     ' row deleted.');
  DBMS_OUTPUT.PUT_LINE (v_rows_deleted);

END;
```



```oracle_plsql
declare
v_empid number:=99;
begin
delete from employees where employee_id=v_empid;
dbms_output.put_line(SQL%ROWCOUNT||' rows deleted' );
update employees set salary=1 where employee_id=100;
dbms_output.put_line(SQL%ROWCOUNT||' rows affected' );
end;
/
rollback
```

```
0 rows deleted
1 rows affected


PL/SQL procedure successfully completed.
```

Quiz

When using the `SELECT` statement in PL/SQL, the `INTO` clause is required and queries can return one or more rows.

a. True
b. False