3

Interacting with the Oracle Server

ORACLE

Objectives

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

- Write a successful SELECT statement in PL/SQL
- Write DML statements in PL/SQL
- Control transactions in PL/SQL
- Determine the outcome of SQL data manipulation language (DML) statements

SQL Statements in PL/SQL

- Extract a row of data from the database by using the SELECT command.
- Make changes to rows in the database by using DML commands.
- Control a transaction with the COMMIT, ROLLBACK, or SAVEPOINT command.
- Determine DML outcome with implicit cursor attributes.

SELECT Statements in PL/SQL

Retrieve data from the database with a SELECT statement.

Syntax:

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

SELECT Statements in PL/SQL

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

Example:

```
SET SERVEROUTPUT ON
DECLARE
   v_deptno NUMBER(4);
   v_location_id     NUMBER(4);
BEGIN
   SELECT department_id, location_id
   INTO                v_deptno, v_location_id
   FROM     departments
   WHERE  department_name = 'Sales';
   DBMS_OUTPUT.PUT_LINE (‘Ma phong : ‘ || v_deptno||’ – ‘|| v_location_id);
END;
/
```

Retrieving Data in PL/SQL

Retrieve the hire date and the salary for the specified employee.

Example:

```
SET SERVEROUTPUT ON
DECLARE
   v_hire_date      employees.hire_date%TYPE;
   v_salary  employees.salary%TYPE;
BEGIN
   SELECT hire_date, salary
   INTO      v_hire_date, v_salary
   FROM     employees
   WHERE  employee_id = 100;
  DBMS_OUTPUT.PUT_LINE ('Ngay vao lam : ' || v_hire_date||
                                                      '  va co muc luong :' || v_salary);
END;
/
```

Retrieving Data in PL/SQL

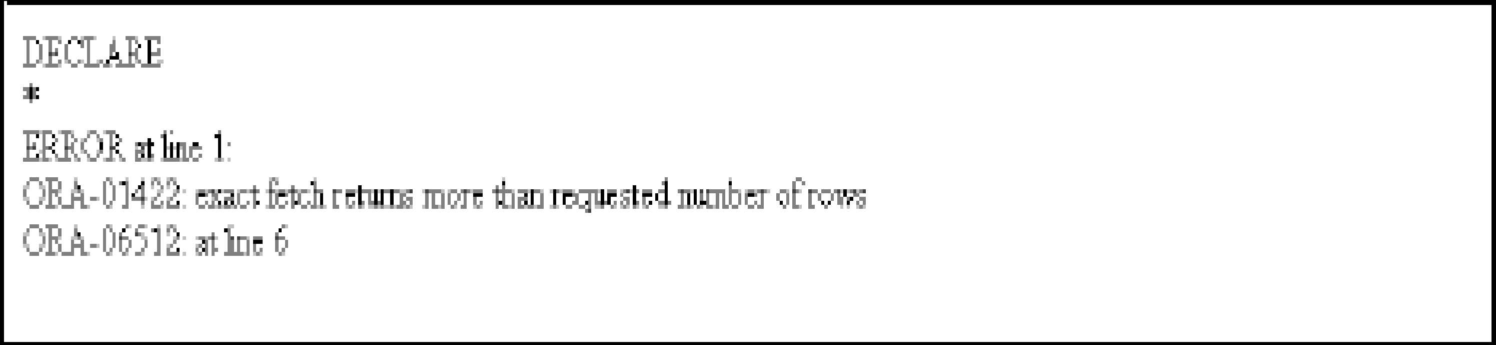
Return the sum of the salaries for all employees in the specified department.

Example:

```
SET SERVEROUTPUT ON
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 salary is ' ||
                                         TO_CHAR(v_sum_sal));
END;
/
```

Naming Conventions

```
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;
/
```

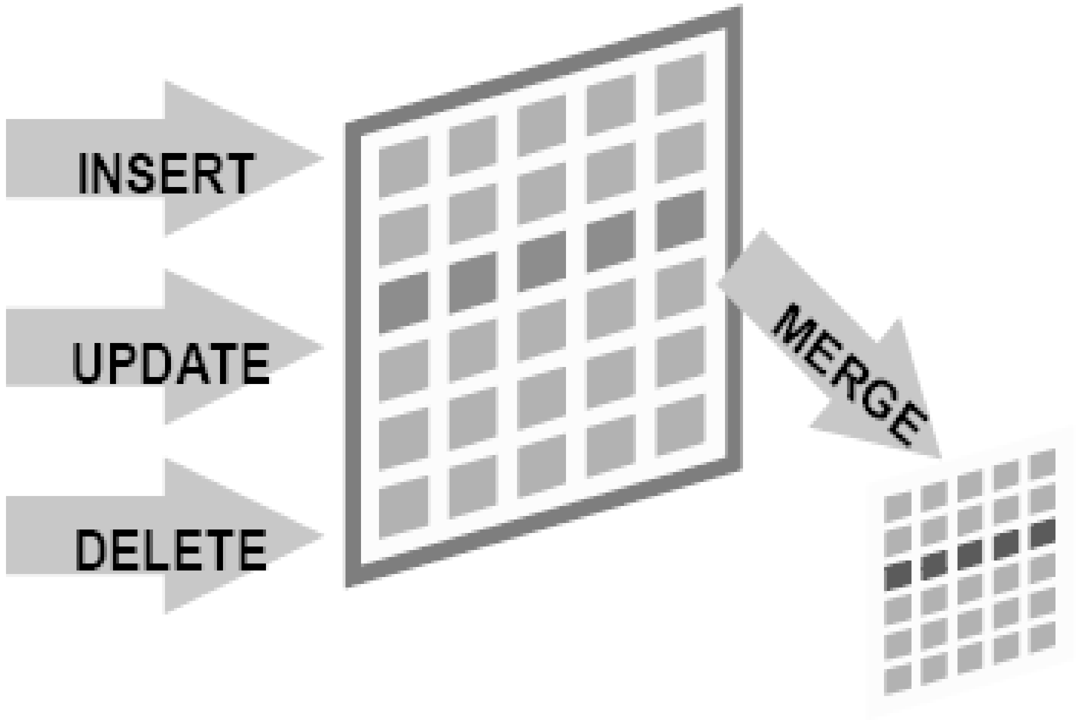


```
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
```

Manipulating Data Using PL/SQL

Make changes to database tables by using DML commands:

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



ORACLE

Inserting Data

Add new employee information to the EMPLOYEES table.

Example:

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

Updating Data

Increase the salary of all employees who are stock clerks.

Example:

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

Deleting Data

Delete rows that belong to department 10 from the EMPLOYEES table.

Example:

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

Merging Rows

Insert or update rows in the COPY_EMP table to match the EMPLOYEES table.

```
DECLARE
   v_empno EMPLOYEES.EMPLOYEE_ID%TYPE := 100;
BEGIN
  MERGE INTO copy_emp c
  USING employees e
  ON (e.employee_id = v_empno)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name = e.first_name,
      c.last_name = e.last_name,
      c.email = e.email,
      c.phone_number = e.phone_number,
      c.hire_date = e.hire_date,
      c.job_id = e.job_id,
      c.salary = e.salary,
      c.commission_pct = e.commission_pct,
      c.manager_id = e.manager_id,
      c.department_id = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES (e.employee_id, e.first_name, e.last_name, e.email, e.phone_number, e.hire_date,
                              e.job_id, e.salary, e.commission_pct, e.manager_id,e.department_id);
END;
/
```

ORACLE

Naming Conventions

- Use a naming convention to avoid ambiguity in the WHERE clause.
- Database columns and identifiers should have distinct names.
- Syntax errors can arise because PL/SQL checks the database first for a column in the table.
- 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.

Naming Conventions

Identifier	Naming Convention	Example
Variable	`v_name`	`v_sal`
Constant	`c_name`	`c_company_name`
Cursor	`name_cursor`	`emp_cursor`
Exception	`e_name`	`e_too_many`
Table type	`name_table_type`	`amount_table_type`
Table	`name_table`	`countries`
Record type	`name_record_type`	`emp_record_type`
Record	`name_record`	`customer_record`
*i*SQL*Plus substitution variable (also referred to as substitution parameter)	`p_name`	`p_sal`
*i*SQL*Plus host or bind variable	`g_name`	`g_year_sal`

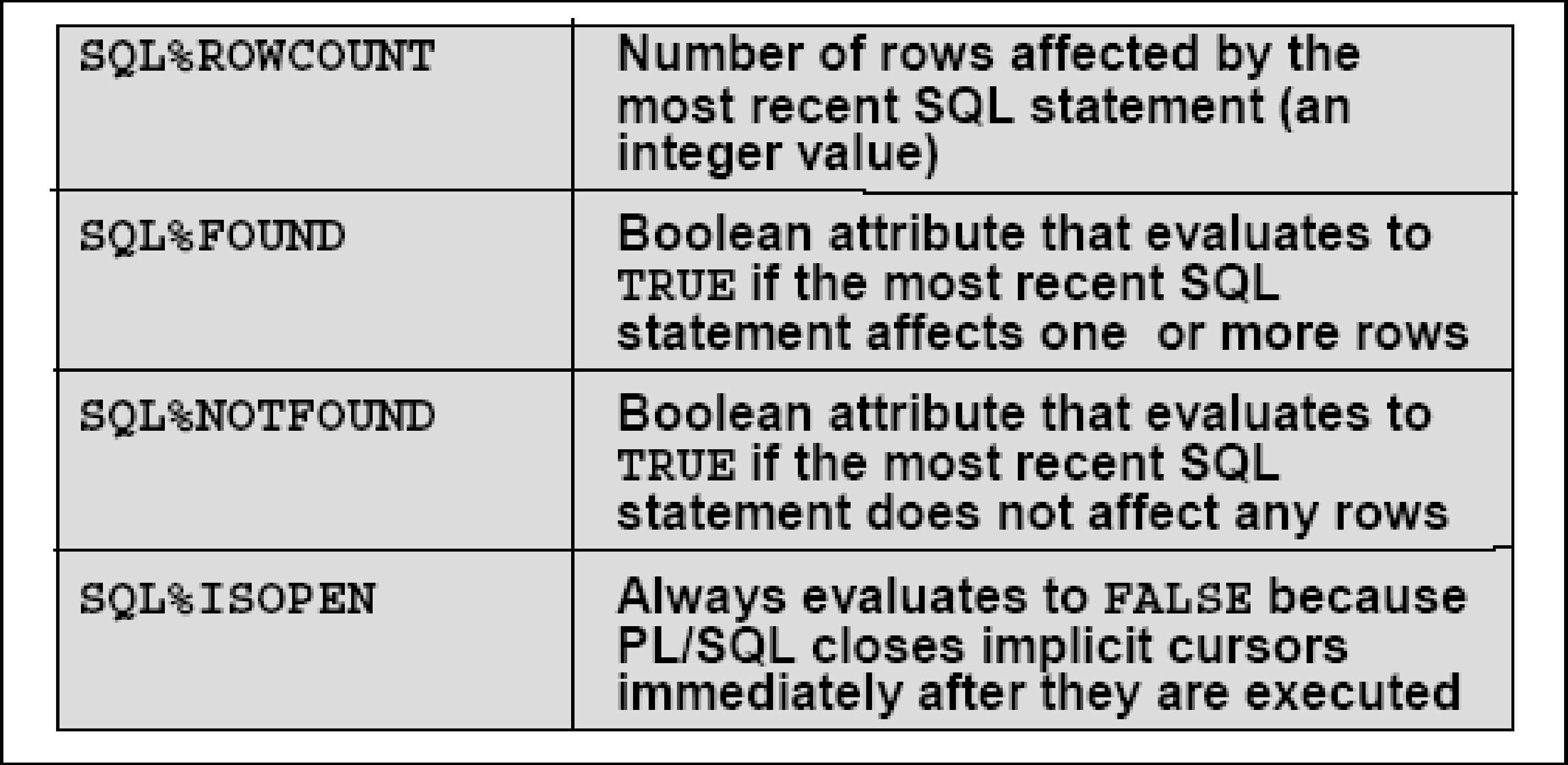
SQL Cursor

- A cursor is a private SQL work area.
- There are two types of cursors:
  - Implicit cursors
  - Explicit cursors
- The Oracle server uses implicit cursors to parse and execute your SQL statements.
- Explicit cursors are explicitly declared by the programmer.

ORACLE

SQL Cursor Attributes

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



`SQL%ROWCOUNT`	Number of rows affected by the most recent SQL statement (an integer value)
`SQL%FOUND`	Boolean attribute that evaluates to `TRUE` if the most recent SQL statement affects one or more rows
`SQL%NOTFOUND`	Boolean attribute that evaluates to `TRUE` if the most recent SQL statement does not affect any rows
`SQL%ISOPEN`	Always evaluates to `FALSE` because PL/SQL closes implicit cursors immediately after they are executed

SQL Cursor Attributes

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

Example:

```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
      v_employee_id     employees.employee_id%TYPE := 176;
BEGIN
      DELETE       FROM    employees
      WHERE      employee_id = v_employee_id;
      :rows_deleted := ( SQL%ROWCOUNT ||  ' row deleted.');
END;
/
PRINT rows_deleted
```

SQL Cursor Attributes

```
CREATE TABLE del_history (
        tenbang       VARCHAR2(20),
        sodong        NUMBER(5),
        ngayxoa       DATE);

VARIABLE rows_deleted VARCHAR2(30)
DECLARE
   v_employee_id employees.employee_id%TYPE := 163;
BEGIN
      DELETE FROM employees
      WHERE employee_id = v_employee_id;
      :rows_deleted := SQL%ROWCOUNT;
      INSERT INTO del_history VALUES ('employees',:rows_deleted, SYSDATE);
--INSERT INTO del_history VALUES ('employees',SQL%ROWCOUNT, SYSDATE);
END;
/
SELECT * FROM del_history;
```

Transaction Control Statements

- Initiate a transaction with the first DML command to follow a COMMIT or ROLLBACK.
- Use COMMIT and ROLLBACK SQL statements to terminate a transaction explicitly.

Summary

In this lesson you should have learned how to:

- Embed SQL in the PL/SQL block using SELECT, INSERT, UPDATE, DELETE, and MERGE
- Embed transaction control statements in a PL/SQL block COMMIT, ROLLBACK, and SAVEPOINT

Summary

In this lesson you should have learned that:

- There are two cursor types: implicit and explicit.
- Implicit cursor attributes are used to verify the outcome of DML statements:
  - SQL%ROWCOUNT
  - SQL%FOUND
  - SQL%NOTFOUND
  - SQL%ISOPEN
- Explicit cursors are defined by the programmer.

Practice 3 Overview

This practice covers creating a PL/SQL block to:

- Select data from a table
- Insert data into a table
- Update data in a table
- Delete a record from a table