

# Interacting with the Oracle Server



# SQL Statements in PL/SQL

- Extract a row of data from the database by using the SELECT command. Only a single set of values can be returned.
- 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 cursors.



# SELECT Statements in PL/SQL

- Retrieve data from the database with SELECT.
- 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.
- Example

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



# Retrieving Data in PL/SQL

- Retrieve the order date and the ship date for the specified order.
- Example

```
DECLARE
  v_orderdate    ord.orderdate%TYPE;
  v_shipdate     ord.shipdate%TYPE;
BEGIN
  SELECT    orderdate, shipdate
  INTO      v_orderdate, v_shipdate
  FROM      ord
  WHERE     id = 620;
          ...
END;
```



# Retrieving Data in PL/SQL

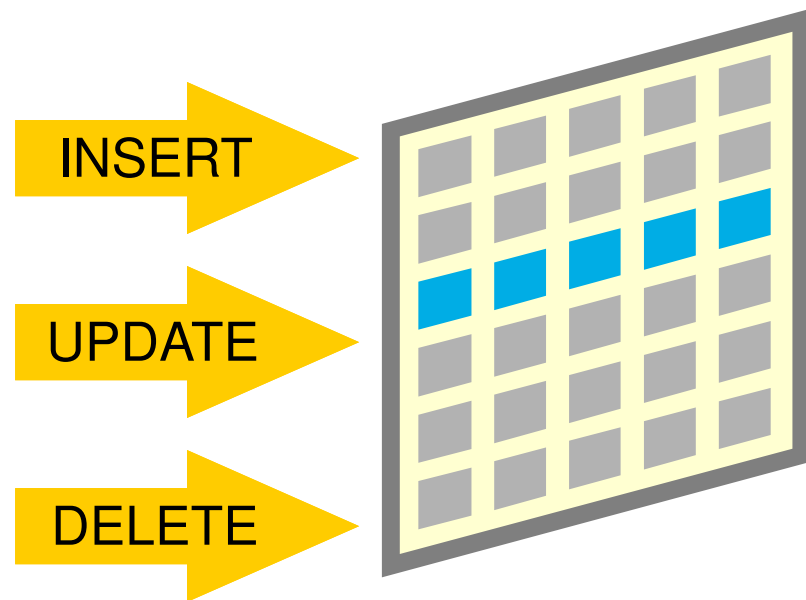
- Return the sum of the salaries for all employees in the specified department.
- Example

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



# Manipulating Data Using PL/SQL

- Make changes to database tables by using DML commands:
  - INSERT
  - UPDATE
  - DELETE



# Inserting Data

- Add new employee information to the EMP table.
- Example

```
BEGIN
  INSERT INTO emp(empno, ename, job, deptno)
  VALUES      (empno_sequence.NEXTVAL, 'HARDING',
               'CLERK', 10);
END;
```





# Updating Data

- Increase the salary of all employees in the EMP table who are Analysts.
- Example

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



# Deleting Data

- Delete rows that belong to department 10 from the EMP table.
- Example

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



# 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.



# Naming Conventions

```
• DECLARE
•   orderdate      ord.orderdate%TYPE;
•   shipdate       ord.shipdate%TYPE;
•   ordid          ord.ordid%TYPE := 601;
• BEGIN
•   SELECT  orderdate, shipdate
•   INTO    orderdate, shipdate
•   FROM    ord
•   WHERE   ordid = ordid;
• END;
• SQL> /
• DECLARE
• *
• ERROR at line 1:
• ORA-01422: exact fetch returns more than requested
• number of rows
• ORA-06512: at line 6
```



# COMMIT and ROLLBACK 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.



# 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.



# 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 order number from the ITEM table. Print the number of rows deleted.
- Example

```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
  v_ordid  NUMBER := 605;
BEGIN
  DELETE FROM  item
  WHERE      ordid = v_ordid;
  :rows_deleted := (SQL%ROWCOUNT ||
                   ' rows deleted.');
```

```
END;
/
PRINT rows_deleted
```

