

---

# **Lab Week – 6**

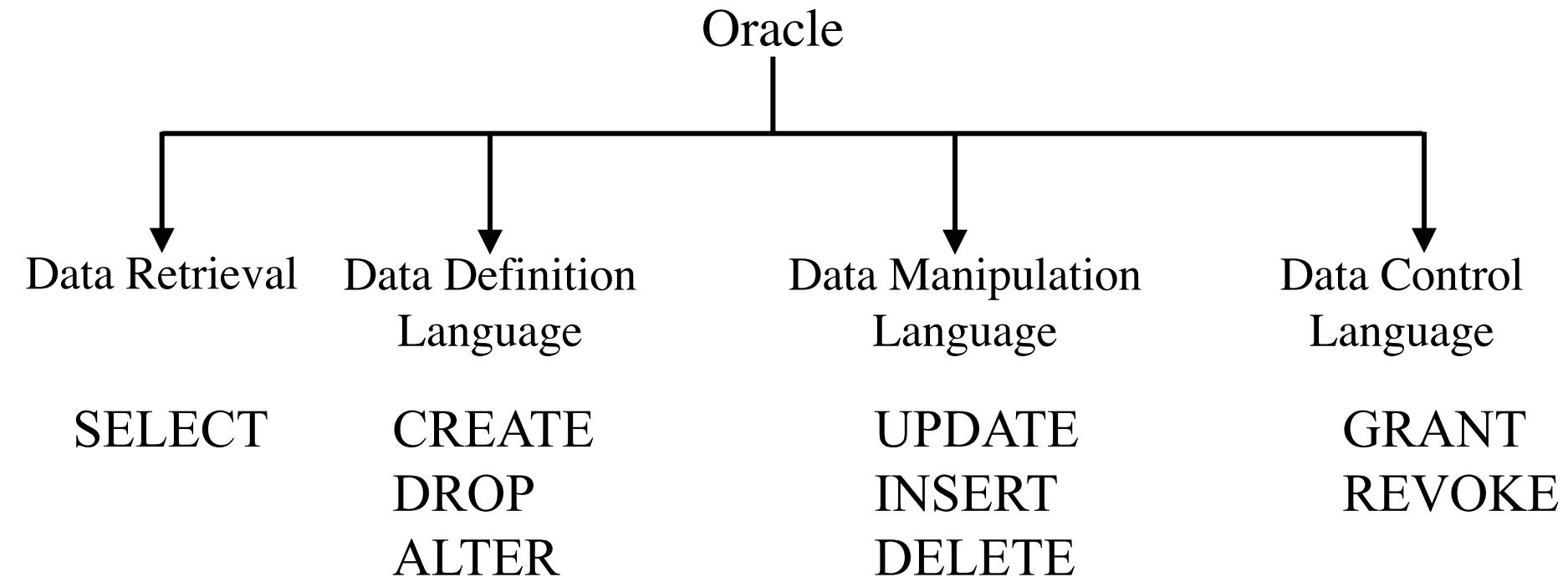
# **Data Manipulation Language**

# Manipulating Data

---

- In this session:
  - Describe each DML statement
  - Insert rows into a table
  - Update rows in a table
  - Delete rows from a table

# Data Access Language



- DML - Commands used to maintain and query data in a DB

# Data Manipulation Language

- A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table

# Adding a New Row to a Table

|    |             |         |
|----|-------------|---------|
| 50 | DEVELOPMENT | DETROIT |
|----|-------------|---------|

New row

DEPT

| DEPTNO | DNAME      | LOC      |
|--------|------------|----------|
| 10     | ACCOUNTING | NEW YORK |
| 20     | RESEARCH   | DALLAS   |
| 30     | SALES      | CHICAGO  |
| 40     | OPERATIONS | BOSTON   |

“...insert a new row  
into DEPT table...”



DEPT

| DEPTNO | DNAME       | LOC      |
|--------|-------------|----------|
| 10     | ACCOUNTING  | NEW YORK |
| 20     | RESEARCH    | DALLAS   |
| 30     | SALES       | CHICAGO  |
| 40     | OPERATIONS  | BOSTON   |
| 50     | DEVELOPMENT | DETROIT  |

# The INSERT Statement

- Add new rows to a table by using the INSERT statement.

```
INSERT INTO    table [(column [, column...])]  
VALUES          (value [, value...]);
```

- Only one row is inserted at a time with this syntax.

# Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally list the columns in the INSERT clause.

```
SQL> INSERT INTO      dept (deptno, dname, loc)
   2 VALUES            (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- Enclose character and date values within single quotation marks.

# Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
SQL> INSERT INTO      dept (deptno, dname)
   2  VALUES            (60, 'MIS');
1 row created.
```

- Explicit method: Specify the NULL keyword.

```
SQL> INSERT INTO      dept
   2  VALUES            (70, 'FINANCE', NULL);
1 row created.
```

# Inserting Special Values

- The SYSDATE function records the current date and time.

```
SQL> INSERT INTO emp (empno, ename, job,
2                      mgr, hiredate, sal, comm,
3                      deptno)
4  VALUES
5        (7196, 'GREEN', 'SALESMAN',
6         7782, SYSDATE, 2000, NULL,
7          10);
8
9  1 row created.
```

# Inserting Values by Using Substitution Variables

- Create an interactive script by using SQL\*Plus substitution parameters. & is a placeholder for the variable value.

```
SQL> INSERT INTO dept (deptno, dname, loc)
  2 VALUES (&department_id,
  3           '&department_name', '&location');
```

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

# Copying Rows from Another Table

- Write your INSERT statement with a subquery.

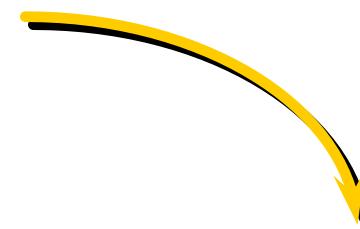
```
SQL> INSERT INTO managers(id, name, salary, hiredate)
  2
  3
  4
  5      SELECT empno, ename, sal, hiredate
  6      FROM   emp
  7      WHERE  job = 'MANAGER';
  8
  9 3 rows created.
```

- Do not use the VALUES clause.
- Match the number of columns in the INSERT clause to those in the subquery.

# Changing Data in a Table

| EMPNO | ENAME | JOB       | ... | DEPTNO |
|-------|-------|-----------|-----|--------|
| 7839  | KING  | PRESIDENT |     | 10     |
| 7698  | BLAKE | MANAGER   |     | 30     |
| 7782  | CLARK | MANAGER   |     | 10     |
| 7566  | JONES | MANAGER   |     | 20     |
| ...   |       |           |     |        |

“...update a row  
in EMP table...”



EMP

| EMPNO | ENAME | JOB       | ... | DEPTNO |
|-------|-------|-----------|-----|--------|
| 7839  | KING  | PRESIDENT |     | 10     |
| 7698  | BLAKE | MANAGER   |     | 30     |
| 7782  | CLARK | MANAGER   |     | 20     |
| 7566  | JONES | MANAGER   |     | 20     |
| ...   |       |           |     |        |

# The UPDATE Statement

- Modify existing rows with the UPDATE statement.

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Update more than one row at a time, if required.

# Updating Rows in a Table

- Specific row or rows to be modified in the WHERE clause.

```
SQL> UPDATE    emp  
  2  SET         deptno = 20  
  3  WHERE      empno = 7782;  
1 row updated.
```

- All rows in the table are modified if you omit the WHERE clause.

```
SQL> UPDATE    employee  
  2  SET         deptno = 20;  
14 rows updated.
```

# Updating with Multiple-Column Sub-query

- Update employee 7698's job and department to match that of employee 7499.

```
SQL> UPDATE emp
  2  SET      (job, deptno) =
  3
  4
  5
  6 WHERE    empno = 7698;
1 row updated.
```

```
(SELECT job, deptno
 FROM   emp
 WHERE  empno = 7499)
```

# Updating Rows Based on Another Table

- Use subqueries in UPDATE statements to update rows in a table based on values from another table.

```
SQL> UPDATE employee
  2  SET deptno = (SELECT deptno
  3                      FROM emp
  4                     WHERE emplno = 7788)
  5  WHERE job      = (SELECT job
  6                      FROM emp
  7                     WHERE emplno = 7788) ;
2 rows updated.
```

# Updating Rows: Integrity Constraint Error

```
SQL> UPDATE emp  
2 SET deptno = 55  
3 WHERE deptno = 10;
```

```
UPDATE emp  
*  
ERROR at line 1:  
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)  
violated - parent key not found
```

Department number 55 does not exist

# Removing a Row from a Table

DEPT

| DEPTNO | DNAME       | LOC      |
|--------|-------------|----------|
| 10     | ACCOUNTING  | NEW YORK |
| 20     | RESEARCH    | DALLAS   |
| 30     | SALES       | CHICAGO  |
| 40     | OPERATIONS  | BOSTON   |
| 50     | DEVELOPMENT | DETROIT  |
| 60     | MIS         |          |
| ...    |             |          |

“...delete a row  
from DEPT table...”

DEPT



| DEPTNO | DNAME      | LOC      |
|--------|------------|----------|
| 10     | ACCOUNTING | NEW YORK |
| 20     | RESEARCH   | DALLAS   |
| 30     | SALES      | CHICAGO  |
| 40     | OPERATIONS | BOSTON   |
| 60     | MIS        |          |
| ...    |            |          |

# The DELETE Statement

- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM]    table
[WHERE          condition] ;
```

# Deleting Rows from a Table

- Specific rows are deleted when you specify the WHERE clause.

```
SQL> DELETE FROM department  
  2 WHERE dname = 'DEVELOPMENT' ;  
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause.

```
SQL> DELETE FROM department;  
4 rows deleted.
```

# Deleting Rows Based on Another Table

- Use subqueries in DELETE statements to remove rows from a table based on values from another table.

```
SQL> DELETE FROM employee
  2 WHERE deptno =
  3
  4
  5
  6 rows deleted.
```

deptno =

```
(SELECT deptno
   FROM dept
  WHERE dname = 'SALES' );
```

# Deleting Rows: Integrity Constraint Error

```
SQL> DELETE FROM dept  
  2 WHERE deptno = 10;
```

```
DELETE FROM dept  
  *  
ERROR at line 1:  
ORA-02292: integrity constraint (USC.EMP.DTNO_FK)  
violated - child record found
```

- You cannot delete a row
- that contains a primary key
- that is used as a foreign key
- in another table

# **COMMIT and ROLLBACK**

---

- Use COMMIT to permanently save the changes made to the database
- Use ROLLBACK to undo the previous operation.
- Note:
  - If you close SQL\*Plus using the x in top right hand corner-your work will be lost unless you have specifically committed your DML
  - If you type ‘Exit’ to close SQL\*Plus, your work will be saved

# Lab Activities

---

- Complete the exercise.