Manipulating Data

ORACLE

Objectives

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

- **Describe each data manipulation language (DML) statement**
- **Insert rows into a table**
- **Update rows in a table**
- **Delete rows from a table**
- **Control transactions**

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
- **A *transaction* consists of a collection of DML statements that form a logical unit of work.**

ORACLE

Adding a New Row to a Table

70	Public Relations	100	1700

New row (label)

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

Insert new row into the DEPARTMENTS table

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
70	Public Relations	100	1700

ORACLE

`INSERT` Statement Syntax

- **Add new rows to a table by using the `INSERT` statement:**

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

- **With this syntax, only one row is inserted at a time.**

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.**

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 row created.
```

- **Enclose character and date values in single quotation marks.**

ORACLE

Inserting Rows with Null Values

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

```
INSERT INTO   departments (department_id,
                          department_name□□ )
VALUES        (30, 'Purchasing');
1 row created.
```

- **Explicit method: Specify the NULL keyword in the VALUES clause.**

```
INSERT INTO   departments
VALUES        (100, 'Finance', NULL, NULL);
1 row created.
```

Inserting Special Values

The `SYSDATE` function records the current date and time.

```
INSERT INTO employees (employee_id,
                first_name, last_name,
                email, phone_number,
                hire_date, job_id, salary,
                commission_pct, manager_id,
                department_id)
VALUES          (113,
                'Louis', 'Popp',
                'LPOPP', '515.124.4567',
                SYSDATE, 'AC_ACCOUNT', 6900,
                NULL, 205, 100);
1 row created.
```

Inserting Specific Date Values

- **Add a new employee.**

```
INSERT INTO employees
VALUES       (114,
              'Den', 'Raphealy',
              'DRAPHEAL', '515.127.4561',
              TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
              'AC_ACCOUNT', 11000, NULL, 100, 30);
1 row created.
```

- **Verify your addition.**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

ORACLE

Copying Rows
from Another Table

- **Write your `INSERT` statement with a subquery:**

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM    employees
  WHERE   job_id LIKE '%REP%';

4 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

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_P
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Update rows in the **EMPLOYEES** table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

UPDATE Statement Syntax

- **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 are modified if you specify the `WHERE` clause:**

```
UPDATE  employees
SET     department_id = 70
WHERE   employee_id = 113;
1 row updated.
```

- **All rows in the table are modified if you omit the `WHERE` clause:**

```
UPDATE    copy_emp
SET       department_id = 110;
22 rows updated.
```

Updating Two Columns with a Subquery

Update employee 114's job and salary to match that of employee 205.

```
UPDATE     employees
SET        job_id  = (SELECT   job_id
                       FROM      employees
                       WHERE     employee_id = 205),
           salary  = (SELECT   salary
                       FROM      employees
                       WHERE     employee_id = 205)
WHERE      employee_id    =   114;
1 row updated.
```

Updating Rows Based on Another Table

Use subqueries in `UPDATE` statements to update rows in a table based on values from another table:

```
UPDATE   copy_emp
SET      department_id  =   (SELECT department_id
                            FROM employees
                            WHERE employee_id = 100)
WHERE    job_id         =   (SELECT job_id
                            FROM employees
                            WHERE employee_id = 200);
1 row updated.
```

Removing a Row from a Table

`DEPARTMENTS`

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

Delete a row from the `DEPARTMENTS` table:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

ORACLE

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 if you specify the `WHERE` clause:**

```
 DELETE FROM departments
 WHERE   department_name = 'Finance';
1 row deleted.
```

- **All rows in the table are deleted if you omit the `WHERE` clause:**

```
DELETE FROM  copy_emp;
22 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:

```
DELETE FROM employees
WHERE   department_id =
                (SELECT department_id
                 FROM   departments
                 WHERE  department_name
                        LIKE '%Public%');
1 row deleted.
```

TRUNCATE Statement

- **Removes all rows from a table, leaving the table empty and the table structure intact**
- **Is a data definition language (DDL) statement rather than a DML statement; cannot easily be undone**
- **Syntax:**

```
TRUNCATE TABLE table_name;
```

- **Example:**

```
TRUNCATE TABLE copy_emp;
```

Using a Subquery in an `INSERT` Statement

```
INSERT INTO
        (SELECT employee_id, last_name,
                email, hire_date, job_id, salary,
                department_id
         FROM    employees
         WHERE   department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000, 50);


1 row created.
```

Using a Subquery in an `INSERT` Statement

Verify the results:

```
SELECT  employee_id, last_name, email, hire_date,
        job_id, salary, department_id
FROM    employees
WHERE   department_id = 50;
```

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
124	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50
141	Rajs	TRAJS	17-OCT-95	ST_CLERK	3500	50
142	Davies	CDAVIES	29-JAN-97	ST_CLERK	3100	50
143	Matos	RMATOS	15-MAR-98	ST_CLERK	2600	50
144	Vargas	PVARGAS	09-JUL-98	ST_CLERK	2500	50
99999	Taylor	DTAYLOR	07-JUN-99	ST_CLERK	5000	50

6 rows selected.

ORACLE