Database Foundations

6-4

Data Manipulation Language (DML)
Roadmap

- Introduction to Oracle Application Express
- Structured Query Language (SQL)
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Transaction Control Language (TCL)
- Retrieving Data Using SELECT
- Restricting Data Using WHERE
- Sorting Data Using ORDER BY
- Joining Tables Using JOIN

You are here
Objectives

This lesson covers the following objectives:

• Describe the purpose of the data manipulation language (DML)

• Explain the DML operations that are required to manage a database's table data:
  – Insert
  – Update
  – Delete
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.
Adding a New Row to a Table

**DEPARTMENTS**

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Administration</td>
<td>200</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>Marketing</td>
<td>201</td>
<td>1800</td>
</tr>
<tr>
<td>30</td>
<td>Purchasing</td>
<td>114</td>
<td>1700</td>
</tr>
<tr>
<td>40</td>
<td>Human Resources</td>
<td>203</td>
<td>2400</td>
</tr>
<tr>
<td>50</td>
<td>Shipping</td>
<td>121</td>
<td>1500</td>
</tr>
<tr>
<td>60</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
<tr>
<td>80</td>
<td>Sales</td>
<td>145</td>
<td>2500</td>
</tr>
</tbody>
</table>

New row

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>Public Relations</td>
<td>204</td>
<td>2700</td>
</tr>
</tbody>
</table>

Insert new row into the **DEPARTMENTS** table.
INSERT Statement Syntax

• Add rows to a table by using the INSERT statement:

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

• With this syntax, only one row is inserted at a time.
Inserting Rows

• Insert a row containing values for each column.

```
INSERT INTO departments
VALUES (80, 'Marketing', 400, 1900);
```

• List values in the default order of the columns in the table.

• Alternatively, list the columns in the `INSERT` clause.

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

• Enclose character and date values within single quotation marks.
Case Scenario: Inserting Rows

I understand that the `INSERT` statement is used to add rows to a table. Is it possible to insert a row if some columns do not have values?

Absolutely. You can use the `INSERT` statement to specify the columns for which values will not be entered.
Inserting Rows with Null Values

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

```sql
INSERT INTO departments (department_id, department_name) VALUES (30, 'Purchasing');
```

• Explicit method: Specify the `NULL` keyword in the `VALUES` clause.

```sql
INSERT INTO departments VALUES (100, 'Finance', NULL, NULL);
```
Case Scenario: Inserting Rows

1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.

```
INSERT INTO AUTHOR VALUES ('AH0002', 'Oscar Wilde', NULL);
INSERT INTO AUTHOR VALUES ('AH0003', 'George Shaw', NULL);
INSERT INTO AUTHOR VALUES ('AH0004', 'Leo', NULL);
```

Rows inserted successfully
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, 110);
```
Inserting Specific Date and Time Values

• Add an employee.

```sql
INSERT INTO employees
VALUES (114,
'Den', 'Raphealy',
'DRAPHEAL', '515.127.4561',
TO_DATE('Dec 7, 2002', 'MON DD, YYYY'),
'SA_REP', 11000, 0.2, 100, 60);
```

1 rows inserted

• Verify your addition.
Changing Data in a Table

**EMPLOYEES**

Update rows in the EMPLOYEES table:

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>PHONE_NUMBER</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>SALARY</th>
<th>COMMISSION_PCT</th>
<th>MANAGER_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Steven</td>
<td>King</td>
<td>SKING</td>
<td>515.123.4567</td>
<td>17-JUN-03</td>
<td>AD_PRES</td>
<td>24000</td>
<td>-</td>
<td>-</td>
<td>90</td>
</tr>
<tr>
<td>101</td>
<td>Neena</td>
<td>Kochhar</td>
<td>NKCHHAR</td>
<td>515.123.4568</td>
<td>21-SEP-05</td>
<td>AD_VP</td>
<td>17000</td>
<td>-</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>102</td>
<td>Lex</td>
<td>De Haan</td>
<td>LDEHAAN</td>
<td>515.123.4569</td>
<td>13-JAN-01</td>
<td>AD_VP</td>
<td>17000</td>
<td>-</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>103</td>
<td>Alexander</td>
<td>Hulio</td>
<td>AHULIO</td>
<td>590.423.4567</td>
<td>03-JAN-06</td>
<td>IT_PROG</td>
<td>9000</td>
<td>-</td>
<td>102</td>
<td>80</td>
</tr>
<tr>
<td>104</td>
<td>Bruce</td>
<td>Ernst</td>
<td>BERNST</td>
<td>590.423.4568</td>
<td>21-MAY-07</td>
<td>IT_PROG</td>
<td>6000</td>
<td>-</td>
<td>103</td>
<td>80</td>
</tr>
<tr>
<td>105</td>
<td>David</td>
<td>Austin</td>
<td>DAUSTIN</td>
<td>590.423.4569</td>
<td>25-JUN-05</td>
<td>IT_PROG</td>
<td>4800</td>
<td>-</td>
<td>103</td>
<td>80</td>
</tr>
<tr>
<td>106</td>
<td>VALLI</td>
<td>Pataballa</td>
<td>VPATALBA</td>
<td>590.423.4560</td>
<td>05-FEB-06</td>
<td>IT_PROG</td>
<td>4800</td>
<td>-</td>
<td>103</td>
<td>60</td>
</tr>
</tbody>
</table>
UPDATE Statement Syntax

• Modify existing values in a table 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

• Values for a specific row or rows are modified if you specify the \texttt{WHERE} clause:

\begin{verbatim}
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
\end{verbatim}

• Values for all the rows in the table are modified if you omit the \texttt{WHERE} clause:

\begin{verbatim}
UPDATE copy_emp
SET department_id = 110;
\end{verbatim}

• Specify \texttt{SET column\_name= NULL} to update a column value to \texttt{NULL}.
Violating Constraints

UPDATE employees
SET department_id = 55
WHERE department_id = 110;

ORA-02291: integrity constraint (APEXWS2.EMP_DEPT_FK) violated - parent key not found

Department 55 does not exist.
Removing a Row from a Table

DEPARTMENTS

Delete a row from the DEPARTMENTS table:
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 include the **WHERE** clause:

```sql
DELETE FROM departments
WHERE department_name = 'Finance';
```

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

```sql
DELETE FROM copy_emp;
```
Violating Constraints

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

```
DELETE FROM departments
WHERE department_id = 60;
```

ORA-02292: integrity constraint (APEXWS2.JHIST_DEPT_FK) violated - child record found
Case Scenario: Deleting Rows

Is it possible to delete all rows in a table but leave the structure intact?

Absolutely. You can use the TRUNCATE statement to do that.
TRUNCATE Statement

• Removes all rows from a table, leaving the table empty and the table structure intact

• Is a DDL statement rather than a DML statement; cannot easily be undone

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE copy_emp;
```
Summary

In this lesson, you should have learned how to:

• Describe the purpose of DML

• List the DML operations that are required to manage a database's table data:
  – Insert
  – Update
  – Delete