
ORACLE
Academy

Database Programming with PL/SQL

3-1

Review of SQL DML

ORACLE Academy

Objectives

- This lesson covers the following objectives:
  - Insert data into a database table
  - Update data in a database table
  - Delete data from a database table
  - Merge data into a database table

Purpose

- When you create, change, or delete an object in a database, the language you use is referred to as data definition language (DDL)
- When you change data within an object (inserting rows, deleting rows, changing column values within a row), it is called data manipulation language (DML)
- This lesson reviews basic SQL DML statements
- Later, you will use DML statements in your PL/SQL code to modify data

Data Manipulation Language (DML)

- You can use DML commands to add rows, delete rows and modify the data within a row
- The DML commands are INSERT, UPDATE, DELETE, and MERGE

ORACLE Academy

Data Manipulation Language (DML)

- In online/hosted versions of Application Express at iAcademy, the default setting for the APEX SQL command processor is that AUTOCOMMIT is turned on
- This means each statement executed will be automatically committed as it is executed
- There is no user control to disable AUTOCOMMIT

Data Manipulation Language (DML)

- If you have a locally installed version of APEX, there is an AUTOCOMMIT checkbox that can be unchecked to disable AUTOCOMMIT
- When AUTOCOMMIT is disabled, DML commands produce permanent changes only when the COMMIT command is issued either within a PL/SQL block or by itself
- If the user logs off normally or closes the browser window before executing a COMMIT command, the changes will be rolled back

INSERT

- You use the INSERT statement to add new rows to a table
- It requires at least two items:
  - The name of the table
  - Values for each of the columns in the table
  - (Optional, but recommended) The names of the columns that will receive a value when the row is inserted
  - If this option is used, then the list of values must match the list of columns
- At the very least, you must list and initialize with a value each column that can not be NULL

INSERT Explicit Syntax

- The syntax shown explicitly lists each column in the table that can not be NULL plus the column for the employee's first name
- The values for each column must be listed in the same order as the columns are listed

```
INSERT INTO employees (employee_id, first_name,
   last_name, email, hire_date, job_id)
  VALUES (305, 'Kareem', 'Naser',
   'naserk@oracle.com', SYSDATE, 'SR_SA_REP');
```

ORACLE Academy

When inserting a row into a table, you must provide a value for each column that can not be NULL. You also could insert the new employee by listing all the columns in the table and including NULL in the list of values that are not known at the moment and can be updated later. For example:

INSERT INTO employees (employee_id, first_name, last_name, email, phone_number,

hire_date, job_id, salary, commission_pct, manager_id, department_head)

VALUES (305, 'Kareem', 'Naser', 'naserk@oracle.com', '111-222-3333', SYSDATE, 'SR_SA_REP', 7000, NULL, NULL, NULL);

Notice the hire_date column is receiving its value from the SYSDATE function. A column's value can be an explicit value or the result of a function. It also can be the result of a subquery or the result of an expression.

INSERT Explicit Syntax

- When inserting a row into a table, you must provide a value for each column that can not be NULL
- You also could insert the new employee by listing all the columns in the table and including NULL in the list of values that are not known at the moment and can be updated later

```
INSERT INTO employees (employee_id, first_name, last_name,
   email, phone_number, hire_date, job_id, salary,
   commission_pct, manager_id, department_head)
 VALUES (305, 'Kareem', 'Naser', 'naserk@oracle.com',
  '111-222-3333',SYSDATE,'SR_SA_REP',7000,NULL,NULL,NULL);
```


Notice the hire_date column is receiving its value from the SYSDATE function. A column's value can be an explicit value or the result of a function. It also can be the result of a subquery or the result of an expression.

INSERT Implicit Syntax

- Another way to insert values in a table is to implicitly add them without listing the column names
- The values must match the order in which the columns appear in the table and a value must be provided for each column

```
INSERT INTO employees
 VALUES (305, 'Kareem', 'Naser',
  'naserk@oracle.com', '111-222-3333', SYSDATE,
  'SR_SA_REP', 7000, NULL, NULL, NULL, NULL);
```


Although this syntax works, it is not recommended. In this example, the EMPLOYEES table has the same eleven columns as listed in the explicit example. But what if the DBA later adds a twelfth column using a DDL ALTER TABLE statement? The INSERT statement above would no longer work. It is good programming practice to explicitly identify the columns when inserting a new row as done on the previous slides.

UPDATE

- The UPDATE statement is used to modify existing rows in a table
- It requires at least three items:
  - The name of the table
  - The name of at least one column to modify
  - A value for each column being modified
  - (Optional) a WHERE clause that identifies the row or rows to be modified
  - If the WHERE clause is omitted, ALL rows will be modified
  - Be very careful when running an UPDATE statement without a WHERE clause!

UPDATE Syntax

It is recommended that the UPDATE statement be on a line of its own

- A single column can be modified
- The WHERE clause identifies the row to be modified

```
UPDATE employees
  SET salary = 11000
  WHERE employee_id = 176;
```

- An UPDATE statement can modify multiple columns

```
UPDATE employees
  SET salary = 11000, commission_pct = .3
  WHERE employee_id = 176;
```

DELETE

- You use the DELETE statement to remove existing rows in a table
- The statement requires at least one item:
  - The name of the table
  - (Optional) a WHERE clause that identifies the row or rows to be deleted
- Please note, if the WHERE clause is omitted, ALL rows will be deleted
- Be very careful when running a DELETE statement without a WHERE clause
- Situations requiring that will be rare

DELETE Syntax

- The WHERE clause identifies the row or rows to be deleted

```
DELETE FROM employees
 WHERE employee_id = 149;
```

```
DELETE FROM employees
 WHERE department_id = 80;
```

- Be careful with the DELETE statement
- If the WHERE clause is omitted, ALL rows will be deleted
- Very few situations will require a DELETE statement without a WHERE clause

ORACLE Academy

The first example deletes one row because employee_id is the primary key.

The second example may delete multiple rows. It will delete all rows that have the value 80 in the department_id column.

MERGE

- The MERGE statement will INSERT a new row into a target table or UPDATE existing data in a target table, based on a comparison of the data in the two tables
- The WHEN clause determines the action to be taken
- For example, if a specific row exists in the source table, but there is no matching row in the target table, the row from the source table will be inserted into the target table
- If the matching row does exist in the target table, but some data in the source table is different for that row, the target table may be updated with the different data

MERGE Usage and Syntax

- To set up our MERGE example, consider a situation where we need to calculate annual bonuses for employees earning less than $10,000 USD
- First, we create a table called bonuses

```
CREATE TABLE bonuses
  (employee_id NUMBER(6,0) NOT NULL,
  bonus NUMBER(8,2) DEFAULT 0);
```

ORACLE Academy

MERGE Usage and Syntax

- We then populate the table with the employee ids of all employees with a salary less than $10,000 USD

```
INSERT INTO bonuses(employee_id)
  (SELECT employee_id FROM employees
  WHERE salary < 10000);
```

MERGE Usage and Syntax

- Each employee with a salary less than $10,000 USD is to receive a bonus of 5% of their salary
- To use the salary column from the employees table to calculate the amount of the bonus and update the bonus column in the bonuses table, we use the following MERGE statement

```
MERGE INTO bonuses b
  USING employees e
  ON (b.employee_id = e.employee_id)
  WHEN MATCHED THEN
   UPDATE SET b.bonus = e.salary * .05;
```

MERGE Usage and Syntax

- The resulting bonus table will look something like this:

EMPLOYEE_ID	BONUS
200	220
206	415
176	430
178	350
124	290

- If you attempt to duplicate this example, your results may vary depending on the data in your employees table

Terminology

- Key terms used in this lesson included:
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - DELETE
  - INSERT
  - MERGE
  - UPDATE

ORACLE Academy

- Data Definition Language (DDL) -- When you create, change, or delete an object in a database.
- Data Manipulation Language (DML) -- When you change data in an object (for example, by inserting or deleting rows).
- DELETE -- Statement used to remove existing rows in a table.
- INSERT -- Statement used to add new rows to a table.
- MERGE -- Statement used to INSERT and/or UPDATE a target table, based on matching values in a source table.
- UPDATE -- Statement used to modify existing rows in a table.

Summary

- In this lesson, you should have learned how to:
  - Insert data into a database table
  - Update data in a database table
  - Delete data from a database table
  - Merge data into a database table


ORACLE
Academy