Sukkur IBA University

CSC-252: Database Management System

Semester IV(CS, SE) Section(A, B) & others (Fall 2020)
Course Instructor(s): Khalid Hussain

Lab 09: DML QUERIES

Objective(s):

1. Introduction to DML Statements
2. INSERT
3. INSERT INTO SELCET
4. INSERT ON DUPLICATE KEY UPDATE
5. UPDATE
6. DELETE

1. DML(Data Manipulation Language)

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

INSERT – is used to insert data into a table.

UPDATE – is used to update existing data within a table.

DELETE – is used to delete records from a database table.

2. INSERT

MySQL INSERT statement is used to insert record(s) or row(s) into a table. The insertion of records or rows in the table can be done in two ways, insert a single row at a time, and insert multiple rows at a time.

The **INSERT INTO** statement is used to insert new records in a table. It is possible to write the INSERT INTO statement in two ways.

The first way specifies both the column names and the values to be inserted:

Syntax:

```
INSERT INTO table_name ( field1, field2,...fieldN )
VALUES( value1, value2,...valueN );
```

Another way is if you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Let's create a new table named tasks and employee for practicing the INSERT statement.

```
CREATE TABLE IF NOT EXISTS tasks (
task_id int auto_increment, title
VARCHAR(255) NOT NULL Unique,
start_date DATE, due_date DATE,
priority TINYINT NOT NULL DEFAULT 3,
description TEXT,
   Constraint Pk_task PRIMARY KEY (task_id)
);
```

```
CREATE TABLE Employee(
empNo varchar(8) not null,
empName varchar(30) not
null, Department varchar(30) not
null, email varchar(30) not null,
phone varchar(30) ,
CONSTRAINT EmployeePK PRIMARY KEY (empNo) );
```

Simple INSERT example

The following statement inserts a new row into the tasks table:

Example: 1

```
INSERT INTO tasks (task_id, title, start_date, due_date, priority, description)
VALUES (123,'Learn MySQL INSERT Statement','2020-06-22','2020-06-27', 1, 'This is the text datatype of MySQL');
```

OR

```
INSERT INTO tasks
VALUES (123,'Learn MySQL INSERT Statement','2020-06-22','2020-06-27', 1, 'This is the text datatype of MySQL');
```

Insert Data Only in Specified Columns Example

It is also possible to only insert data in specific columns.

Example: 2

```
INSERT INTO tasks (title, priority)
VALUES ('Understanding DEFAULT keyword in INSERT statement', DEFAULT);
```

In this example, we specified the values for only title and priority columns. For other columns, MySQL uses the default values.

The task_id column is an AUTO_INCREMENT column. It means that MySQL generates a sequential integer whenever a row is inserted into the table.

The start_date, due_date, and description columns use NULL as the default value, therefore, MySQL uses NULL to insert into these columns if you don't specify their values in the INSERT statement.

Insert data in multiple rows Example

The following statement inserts three rows into the tasks table:

Example 3:

```
INSERT INTO tasks (title, priority)
VALUES
        ('My first task', 1),
        ('It is the second task',2),
        ('This is the third task of the week',3);
```

Example 4:

```
Insert into EMPLOYEE (EMPNO, EMPNAME, DEPARTMENT, EMAIL, PHONE) values
('E100','Chuck Coordinator','Administration','chuck@colorado.edu','3-1111'),
('E101','Mary Manager','Football','mary@colorado.edu','5-1111'),
('E102','Sally Supervisor','Planning','sally@colorado.edu','3-2222');
```

MySQL INSERT IGNORE Statement Example

When you use the INSERT statement to add multiple rows to a table and if an error occurs during the processing, MySQL terminates the statement and returns an error. As the result, no rows are inserted into the table.

However, if you use the INSERT IGNORE statement, the rows with invalid data that cause the error are ignored and the rows with valid data are inserted into the table.

Example 5:

```
INSERT IGNORE INTO tasks (title, priority)
VALUES
        ('Duplicate value', 1),
        ('Duplicate value', 2)
```

MySQL returned a message indicating that one row was inserted and the other row was ignored due to duplicate value.

3. INSERT INTO SELECT

To insert data from other tables into a table, you use the following SQL INSERT INTO SELECT statement:

Syntax:

```
INSERT INTO table_name(column_list) SELECT
   select_list
FROM
   another_table
WHERE
   condition;
```

Insert all rows from another table example

To insert data from other tables into a table, you use the following SQL INSERT INTO SELECT statement:

Example 6:

The following statement inserts all department from the employee table into the tasks table:

```
INSERT INTO tasks (title)
SELECT department
FROM employee;
```

Insert some rows from another table example

Sometimes, you just need to insert some rows from another table into a table. In this case, you limit the number of rows returned from the query by using conditions in the WHERE clause.

Example 7:

```
INSERT INTO tasks (title)
SELECT email
FROM employee
WHERE empNo='E101';
```

Using SELECT statement in the VALUES list

We can also use select statement inside the value. Here is an example that first, use the SELECT statements with the COUNT() functions to get the total employees. Second, use the values returned from the SELECT statement in place of values in the VALUES clause of the INSERT statement.

Example 8:

```
INSERT INTO tasks(task_id, title, priority)
VALUES(
        (SELECT COUNT(*) FROM employee),
        (SELECT email FROM employee where empNo='E102'),
        (SELECT COUNT(*) FROM employee)
);
```

4. INSERT ON DUPLICATE KEY UPDATE

The INSERT ON DUPLICATE KEY UPDATE is a MySQL's extension to the SQL standard's INSERT statement.

When you insert a new row into a table if the row causes a duplicate in UNIQUE index or PRIMARY KEY, MySQL will issue an error.

However, if you specify the ON DUPLICATE KEY UPDATE option in the INSERT statement, MySQL will update the existing row with the new values instead.

Syntax:

```
INSERT INTO table (column_list)
VALUES
(value_list)
ON DUPLICATE KEY
UPDATE
  c1    =
v1,

  c2 = v2,
  ...;
```

The only addition to the INSERT statement is the ON DUPLICATE KEY UPDATE clause where you specify a list of column-value-pair assignments in case of duplicate.

Example 9:

```
INSERT INTO tasks(task_id, title)
VALUES (7, 'This will not updated if id duplicates')
ON duplicate key update task_id= values (task_id)+1 ;
```

Basically, the statement first tries to insert a new row into the table. If a duplicate error occurs, it will update the existing row with the value specified in the ON DUPLICATE KEY UPDATE clause i.e. task_id.

5. UPDATE

The UPDATE statement modifies existing data in a table. You can also use the UPDATE statement change values in one or more columns of a single row or multiple rows.

The following illustrates the basic syntax of the UPDATE statement:

Syntax:

```
UPDATE table_name
SET field1 = new-value1, field2 = new-value2
[WHERE Clause]
```

- You can update one or more field altogether.
- You can specify any condition using the WHERE clause.
- You can update the values in a single table at a time.

- The WHERE clause is very useful when you want to update the selected rows in a table.

UPDATE to modify values in a single column example

The following example use employee table tto update single column name

Example 10:

```
Update employee set
empname= 'BlackSmith'
where empNo='E100';
```

UPDATE to modify values in multiple columns

To update values in the multiple columns, you need to specify the assignments in the SET clause. For example, the following statement updates both *empName* and *Department* columns of *empNo* E100:

Example 11:

```
Update employee set empname= 'BlackSmith',
Department= 'Iron' where empNo='E100';
```

UPDATE to replace string example

The following example updates the domain parts of emails of *empNo* E100 **Example 12:**

```
UPDATE employee
SET email = REPLACE(email,'@colorado.edu','@iba-suk.edu.pk')
WHERE empNo='E100';
```

Using MySQL UPDATE with SELECT statement

You can supply the values for the SET clause from a SELECT statement that queries data from other tables.

In this example the query update the description column of tasks table with department column value of customer table, we place the query above in the SET clause of the UPDATE statement as follows:

Example 13:

```
Update tasks
```

```
SET description= (Select department from employee where empNo='E100') WHERE
description IS NULL;
```

MySQL UPDATE on multiple tables

Here we have used two table tasks and employee for the following example as sample table.

Example 14:

```
Update tasks t, employee e
Set t.due_date=adddate(current_date(),3),e.phone ='3-111'
Where e.department=t.description;
```

MySQL UPDATE JOIN example

In MySQL, you can use the JOIN clauses in the UPDATE statement to perform the cross-table update.

```
UPDATE employee e
      left JOIN
  tasks t ON e.department = t.description  SET      phone = due_date;
```

For each row in the *employee* table, the query checks the value in the *department* column against the value in the *description* column in the *tasks* table. If it finds a match, it gets the *due_date* value from tasks table and updates the *phone* column in the *employee* table.

The UPDATE LEFT JOIN statement basically updates a row in a table when it does not have a corresponding row in another table so it will replace those non matching column value with NULL.

6. DELETE

If you want to delete a record from any MySQL table, then you can use the SQL command DELETE FROM.

Syntax:

```
DELETE FROM table_name [WHERE Clause]
```

If the WHERE clause is not specified, then all the records will be deleted from the given MySQL table.

The WHERE clause is very useful when you want to delete selected rows in a table.

MySQL DELETE specific rows or records

Suppose you want to delete employees where the *empNo* is E101, you use the DELETE statement with the WHERE clause as shown in the following query:

Example 15:

```
DELETE FROM employees
WHERE empNo = 'E101';
```

To delete all rows from the *employee* table, you use the DELETE statement without the WHERE clause as follows:

```
DELETE FROM employee;
```

Note: The TRUNCATE TABLE statement is a faster way to empty a table than a DELETE statement with no WHERE clause.

MySQL DELETE with LIMIT and ORDER BY clause example:

ORDER BY and LIMIT keyword can be used with MySQL DELETE statement to remove only a given number of rows, where columns are sorted in a specific order.

Example 16:

```
DELETE FROM tasks
ORDER BY title DESC LIMIT 2;
```

MySQL DELETE Multiple tables with JOIN example:

MySQL also allows you to use the JOIN clause in the DELETE statement to delete rows from a multiple table on matching rows in another table.

Example 17:

```
delete tasks, employee  from tasks join employee on
tasks.description=employee.department;
```

LAB Task(s)

Exercise

Run SQL statement to create a table projects and student.

```
CREATE TABLE projects(
      project_id INT AUTO_INCREMENT,
      name VARCHAR(100) NOT NULL,
      start_date DATE,      end_date
DATE,
      PRIMARY KEY(project_id));

CREATE TABLE student( std_NO
      varchar(8) not null, std_Name
      varchar(30) not null, Department
      varchar(30) not null, email
      varchar(30) not null, phone
      varchar(30) , project_id int ,
      FOREIGN KEY (project_id) REFERENCES projects (project_id),
      PRIMARY KEY (std_No) );
```

1. Write a SQL statement to insert 4 rows in project table and 4 rows in student table below by a single insert statement.

project_id	name	start_date	end_date
1	AI for Marketing	2019-08-01	2019-12-31
2	ML for Sales	2019-05-15	2019-11-20
3	CS for IT	2020-01-01	2020-05-20
4	SQL for input	2020-06-13	2020-11-20

std_NO	std_Name	Department	email	phone	project_id
S100	Ali Mehmood	Administration	ali@iba-suk.edu.pk	0333-895311	3
S101	Manisha Kataria	Computer Science	manisha@iba-suk.edu.pk	0345-111333444	2
S102	Sagar Sanjay	Engineering	sagar@iba-suk.edu.pk	0300-22224454	2
S103	Sara Shaikh	IT	sara@iba-suk.edu.pk	0300-111110000	3

Now Create duplicate of projects table named projects_copy with all structure and data

```
CREATE TABLE IF NOT EXISTS projects_copy
```

```
SELECT * FROM projects;
```

2. Write a SQL statement to delete all records from projects table.
3. Write a SQL statement to insert rows from projects_copy table to projects table.
4. Write a SQL statement to update start_date to '2023-02-01' of a project name *CS for IT*.

Now add a column in table project by following command.

```
ALTER TABLE projects
ADD cost INT;

ALTER TABLE student
ADD daysToComplete INT;
```

5. Write a SQL statement to update cost of project to 90000 where cost are null.
6. Write a SQL statement to update daysToComplete column from student by calculating difference from project start_date and end_date.