3-Data Manipulation Language (DML)

Author: Vincent Lau

Learning Objectives

- Understand Insert, Update, Delete and Select operations in SQL
- Understand different operators in SELECT statements, such as WHERE, ORDER, DISTINCT, LIKE, etc Operators.

Introduction

Data Manipulation Language (DML) is a subset of SQL (Structured Query Language) that focuses on the manipulation of data stored in a database. DML statements are used to insert, update, retrieve, and delete data from database tables.

Here's an introduction to some common DML statements along with code examples in MySQL:

Inserting Data

Insert Data: The `INSERT INTO` statement adds new rows to a table. You provide the column names and values to be inserted.

1: With column name specified - All columns

- The sequence of provided values refers to the column sequence provided in the insert statement, but can be **different** to the sequence of columns in table definition.

```sql
INSERT INTO employees (employee_id, first_name, last_name, hire_date)
VALUES (1, 'John', 'Doe', '2023-01-15');
```

2: With column name specified - Some column(s)

- The sequence of provided values refers to the column sequence provided in the insert statement, but can be **different** to the sequence of columns in table definition.
- Only the column values with column definition - NON NULL, Primary Key, etc are required to provide values.

```sql
INSERT INTO employees (hire_date, employee_id)
VALUES ('2023-01-15', 1);
```

3: Without column name specified

- **All column values must be provided**
- The column values must be provided according to the column sequences specified in table

```sql
INSERT INTO employees
VALUES (1, 'John', 'Doe', '2023-01-15');
```

Updating Data

Update Data: The `UPDATE` statement modifies existing data in a table. You specify the column(s) to be updated and set new values using the `SET` clause. The `WHERE` clause filters which rows to update.

- Support Subquery (Refer to the Subquery Chapter)

```sql
UPDATE employees
```

```
SET salary = salary * 1.1
WHERE department_id = 2;
```

Retrieving Data

Retrieve Data: The `SELECT` statement retrieves data from one or more tables. You can specify columns, use conditions in the `WHERE` clause, and join multiple tables to retrieve desired information.

Simple Query

```
SELECT first_name, last_name, hire_date
FROM employees
WHERE department_id = 1;
```

Join Query

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

Deleting Data

Delete Data: The `DELETE FROM` statement removes rows from a table based on a specified condition in the `WHERE` clause.

```
DELETE FROM employees
WHERE department_id = 3;
```

Truncate Table

- Remove all data from the table
- Non-rollbackable
- MySQL & Oracle: rollback & commit (Most of the SQL statement, not including truncate)

```sql
truncate table employees;

-- Similar to delete from without conidtion
delete from employees;
```

Query Data (SELECT)

The `SELECT` statement is a fundamental part of the Data Manipulation Language (DML) in SQL. It is used to retrieve data from one or more tables and provides various options for filtering, sorting, and manipulating the data.

Here's an introduction to the `SELECT` statement along with code examples:

Retrieving All Columns

```sql
SELECT *
FROM employees;
```

Using DISTINCT to Remove Duplicates

```sql
-- DISTINCT all 3 columns
SELECT DISTINCT department_id, last_name, first_name
FROM employees;
```

Applying Conditions with WHERE

```sql
SELECT first_name, last_name
FROM employees
WHERE salary > 50000;
```

Using BETWEEN Operators

- Use Case: Integer

```sql
SELECT product_name, unit_price
FROM products
WHERE unit_price BETWEEN 10 AND 50;
```

- Use Case: Date and Datetime

```sql
-- Date
SELECT order_id, order_date
FROM orders
WHERE order_date BETWEEN '2023-01-01' AND '2023-06-30';

-- Datetime
SELECT order_id, order_date
FROM orders
WHERE order_date BETWEEN '2023-01-01 00:00:00' AND '2023-06-30 23:59:59';
```

Using the IN Operator

```sql
SELECT first_name, last_name FROM employees WHERE department_id IN (2, 3, 5);
```

Sorting with ORDER BY

```sql
SELECT customer_id, order_date, total_amount
FROM orders
ORDER BY order_date DESC;
```

Limiting Results with LIMIT

```sql
SELECT first_name, last_name
FROM employees
LIMIT 10;
```

Using LIKE for Pattern Matching

```sql
SELECT product_name
FROM products
WHERE product_name LIKE 'S%'; -- '%XX%'
```

Combining Conditions with AND and OR

```sql
SELECT first_name, last_name
FROM employees
WHERE department_id = 2 AND salary > 60000;
```

Negating Conditions with NOT

```sql
SELECT customer_id, total_amount
FROM orders
WHERE NOT status = 'Shipped';
```

Counting Rows

```sql
-- Or you can COUNT(1)
SELECT COUNT(*) AS total_orders
FROM orders;
```

Using Arithmetic Operations

```sql
SELECT product_name, unit_price * 0.9 AS discounted_price
FROM products;
```

NULL Handling

```sql
-- NULL
SELECT first_name, last_name
FROM employees
WHERE email IS NULL;

-- NOT NULL
SELECT first_name, last_name
FROM employees
WHERE email IS NOT NULL;
```

Summary

DML statements are used to interact with the actual data stored in the database. They allow you to insert, update, retrieve, and delete data as needed to perform various operations within your application. Always use DML statements with caution, and ensure that you understand the impact they can have on your data.