Database and SQL Fundamental

Press Space for next page →

Agenda

Data VS Information

DATA	INFORMATION
Data refers to raw facts that have no specific meaning	Information refers to processed data that has a purpose and meaning
The data is independent of the information	Information is dependent on data
Data or raw data is not enough to make a decision	The information is sufficient to help make a decision in the respective context

Database

> A database is a collection of organized data, information, and records that can be easily accessed and managed. It is a structured repository that stores data in a way that allows for efficient retrieval and manipulation.

Key Characteristics:

Organized Data: Data is stored in a structured format, making it easy to access and manage.

Collection of Records: A database can contain multiple records, each with its own set of attributes.

Easy Access and Management: Data can be easily retrieved, updated, and manipulated using database management systems (DBMS).

Examples of Databases

- **Personal Database::** A list of phone numbers, addresses, or personal contacts.
- **Business Database:** A collection of customer information, sales records, or inventory data.
- **Educational Database:** A repository of student records, course information, or exam results.

Common Database Types

- **Relational Databases:** Organizes data into tables with rows and columns.

  Example: MySQL, PostgreSQL, Oracle.

- **NoSQL Databases:** Stores data in a non-tabular format, such as key-value pairs or document stores.

  Example: MongoDB, Cassandra, Redis.

- **Key-Value Database:** Stores data as key-value pairs.

  Example: Redis, DynamoDB.

Components of a Databases

- **Tables:** A collection of related data, organized into rows and columns.
- **Record: :** A single entry in a table, representing a single instance of data.
- **Fields:** A single column in a table, containing a specific piece of data for each record.

ROLL NO	NAME	AGE	GPA
101	Alice	20	3.5
102	Bob	21	3.2

Common types of keys in a database

Primary Key: A unique identifier for each record in a table.

- Uniquely identifies each record.
- No duplicate values or NULL values allowed.
- Typically indexed for faster search and retrieval.
- Example: Employee ID in an Employees table.

Foreign Key: A field in a table that refers to the primary key of another table.

- Establishes relationships between tables.
- Can contain NULL values and duplicate values.
- Used to link data across tables.
- Example: Order ID in an Orders table referencing the Customer ID in a Customers table.

Download and Install MySQL

- Download XAMPP from: apachefriends.org
- Install XAMPP on your computer.
- Start the Apache and MySQL services.
- Open phpMyAdmin in your browser: http://localhost/phpmyadmin/

Execute SQL Queries

- Open phpMyAdmin in your browser.
- Click on the SQL tab.
- `SHOW DATABASES;` to list all databases.
- `CREATE DATABASE mydatabase;` to create a new database.
- `USE mydatabase;` to switch to the new database.
- `DROP DATABASE mydatabase;` to delete the database.

Data Types in MySQL

Numeric Data Types: INT, FLOAT, DOUBLE, DECIMAL.

- *INT:* Integer values.

```
CREATE TABLE employees (id INT PRIMARY KEY, salary INT);
```

- *DECIMAL:* Fixed-point numbers.

```
CREATE TABLE products (price DECIMAL(10, 2));
```

- *FLOAT:* An approximate floating-point number.

```
CREATE TABLE products (price FLOAT);
```

Date and Time Data Types: DATE, TIME, DATETIME.

- *DATE:* Date values. Example format 'YYYY-MM-DD'

```
CREATE TABLE employees (dob DATE);
```

- *DATETIME:* Date and time values. Example format 'YYYY-MM-DD HH:MM:SS'

```
CREATE TABLE employees (created_at DATETIME);
```

String Data Types: CHAR, VARCHAR, TEXT.

- *VARCHAR:* Variable-length string.

```
CREATE TABLE users (username VARCHAR(50), email VARCHAR(100));
```

SQL Commands

DDL (Data Definition Language)

- *CREATE TABLE:* Creates a new database, table, index, or view
- *ALTER TABLE:* Modifies an existing database, table, or view
- *DROP TABLE:* Deletes a database, table, index, or view
- *TRUNCATE TABLE:* Removes all records from a table, but keeps the table structure

DML (Data Manipulation Language):

Example: INSERT, UPDATE, DELETE

DQL (Data Query Language):

Example: SELECT, SELECT DISTINCT, SELECT INTO

Create a Table in MySQL

```sql
CREATE TABLE students (
  id INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  class INT,
  section CHAR(1),
  fees INT,
  house VARCHAR(20)
);
```

```sql
INSERT INTO students (id, name, class, section, fees, house) VALUES
 (101, 'Alice', 10, 'A', 5000, 'Red'),
 (102, 'Bob', 11, 'B', 6000, 'Blue'),
 (103, 'Charlie', 12, 'C', 7000, 'Green'),
 (104, 'David', 10, 'A', 5500, 'Yellow'),
 (105, 'Emma', 11, 'B', 5000, 'Red'),
 (106, 'Fiona', 10, 'C', 6200, 'Green'),
 (107, 'George', 12, 'A', 6500, 'Blue'),
 (108, 'Hannah', 11, 'C', 7000, 'Red'),
 (109, 'Ivy', 12, 'B', 6000, 'Yellow'),
 (110, 'Jack', 10, 'B', 5800, 'Green'),
 (111, 'Kara', 11, 'A', 5500, 'Blue'),
 (112, 'Liam', 12, 'C', 7500, 'Red'),
 (113, 'Mason', 10, 'A', 5000, 'Yellow'),
 (114, 'Nora', 11, 'B', 5300, 'Green'),
 (115, 'Oscar', 12, 'C', 6400, 'Blue'),
 (116, 'Paula', 10, 'B', 5100, 'Red'),
 (117, 'Quinn', 11, 'A', 7000, 'Yellow'),
 (118, 'Rachel', 12, 'B', 6900, 'Green'),
 (119, 'Steve', 10, 'C', 4800, 'Blue'),
 (120, 'Tina', 11, 'A', 5600, 'Red');
```

```sql
RENAME TABLE students TO new_students;
DROP TABLE new_students;
```

Select Data from a Table

Example 1: Select Specific Columns

```sql
SELECT name, class, fees FROM students;
```

This query selects the name, class, and fees columns from the students table.

Example 2: Select All Columns

```sql
SELECT * FROM students;
```

This query selects all columns from the students table.

Example 3: Select with WHERE Clause

```sql
SELECT * FROM students WHERE class = 10;
```

This query selects all columns from the students table where the class is 10.

Example 4: Select with AND Operator

```sql
SELECT * FROM students WHERE class = 10 AND section = 'A';
```

This query selects all columns from the students table where the class is 10 and the section is 'A'.

Select Query

Example 5: Select with OR Operator

```sql
SELECT * FROM students WHERE class = 12 OR fees > 2500;
```

This query selects all columns from the students table where the class is 12 or the fees are greater than 2500.

Example 6: Select with ORDER BY

```sql
SELECT * FROM students ORDER BY fees DESC;
```

This query selects all columns from the students table and orders the result by the fees column in descending order. *Example 7: Select with DISTINCT Clause*

```sql
SELECT DISTINCT Class FROM students;
```

This query selects all distinct values from the class column in students table.

Example 8: Select with LIMIT Clause

```sql
SELECT * FROM students LIMIT 5;
```

This query selects all columns from the students table and limits the result to the first 5 rows.

More Select Query

Example 9: Select with GROUP BY Clause

```sql
SELECT class, AVG(fees) FROM students GROUP BY class;
```

This query selects the class and the average fees from the students table, grouped by the class column. *Example 10: Select with HAVING Clause*

```sql
SELECT class, AVG(fees) FROM students GROUP BY class HAVING AVG(fees) > 2500;
```

This query selects the class and the average fees from the students table, grouped by the class column, and filters the result to only include groups where the average fees are greater than 2500.

Update and Delete Data

Update Query

```sql
UPDATE students SET fees = 3000 WHERE name = 'Anu Jain';
```

This query updates the fees column in the students table to 3000 where the name is 'Anu Jain'. ***Delete Query***

```sql
DELETE FROM students WHERE name = 'Mohit Sharma';
```

This query deletes the record from the students table where the name is 'Mohit Sharma'.

Database Testing

Purpose of Database Testing:

- To ensure that the database operates correctly and efficiently.
- To verify data integrity, data consistency, and data accuracy.
- To ensure that CRUD operations (Create, Read, Update, Delete) work as expected.
- To check for performance issues like indexing and query optimization.

Key Aspects of Database Testing

- **Schema Testing:** Verifying that the database schema (tables, columns, indexes, etc.) is set up correctly.
- **Data Integrity Testing:** Ensuring that data remains consistent and accurate across operations.
- **Performance Testing:** Checking how the database performs under load.
- **Security Testing:** Ensuring that the database is secure from unauthorized access and vulnerabilities.

Adding Dependency

Link: https://mvnrepository.com/artifact/com.mysql/mysql-connector-j

```xml
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>9.1.0</version>
</dependency>
```

Database Connection

```java
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/nexxvali", "root","");
System.out.println("Connected to MySQL Database");

Statement statement = connection.createStatement();
System.out.println("Statement created");
```

Execute SQL Query

```java
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/nexxvali", "root","");
System.out.println("Connected to MySQL Database");

Statement smt = connection.createStatement();
System.out.println("Statement created");

ResultSet rs=smt.executeQuery("select * from students");
int id;
String name;
while (rs.next()) {
    id = rs.getInt("id");
    name = rs.getString("name").trim();
    System.out.println("ID : " + id + " Name : " + name);
}
rs.close();
smt.close();
connection.close();
```

Update Query

Update Query

```java
String updateQuery = "UPDATE students SET fees = 3000 WHERE name = 'Alice'";
int rowsAffected = smt.executeUpdate(updateQuery);
System.out.println("Rows Affected: " + rowsAffected);
```

Database Testing with TestNG

```java
public class DatabaseTest {
    Connection connection;
    Statement statement;

    @BeforeTest
    public void setup() throws SQLException {
        try {
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/nexxvali", "root", "");
            // Create statement
            statement = connection.createStatement();
        } catch (Exception e) {
            e.printStackTrace();
            Assert.fail("Database connection setup failed!");
        }
    }

    @AfterClass
    public void tearDown() {
        try {
            if (statement ≠ null) statement.close();
            if (connection ≠ null) connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Database Testing with TestNG : Example

```java
@Test(priority = 1)
public void testInsert() {
    try {
        String query = "INSERT INTO students (id, name, class) VALUES (501, 'TestUser', 10)";
        int rowsInserted = statement.executeUpdate(query);
        Assert.assertEquals(rowsInserted, 1, "Insert operation failed!");
    } catch (SQLException e) {
        e.printStackTrace();
        Assert.fail("Insert query execution failed!");
    }
}
@Test(priority = 2, dependsOnMethods = "testInsert")
public void testSelect() {
    try {
        String query = "SELECT * FROM students WHERE id = 501";
        ResultSet resultSet = statement.executeQuery(query);
        Assert.assertTrue(resultSet.next(), "Record not found after insert!");
        Assert.assertEquals(resultSet.getString("name").trim(), "TestUser", "Name mismatch!");
        Assert.assertEquals(resultSet.getInt("class"), 10, "Class mismatch!");
        resultSet.close();
    } catch (SQLException e) {
        e.printStackTrace();
        Assert.fail("Select query execution failed!");
    }
}
```

Thank you ❤️

qa-june-2024-automation-with-java-slides