

# Database Programming using JDBC

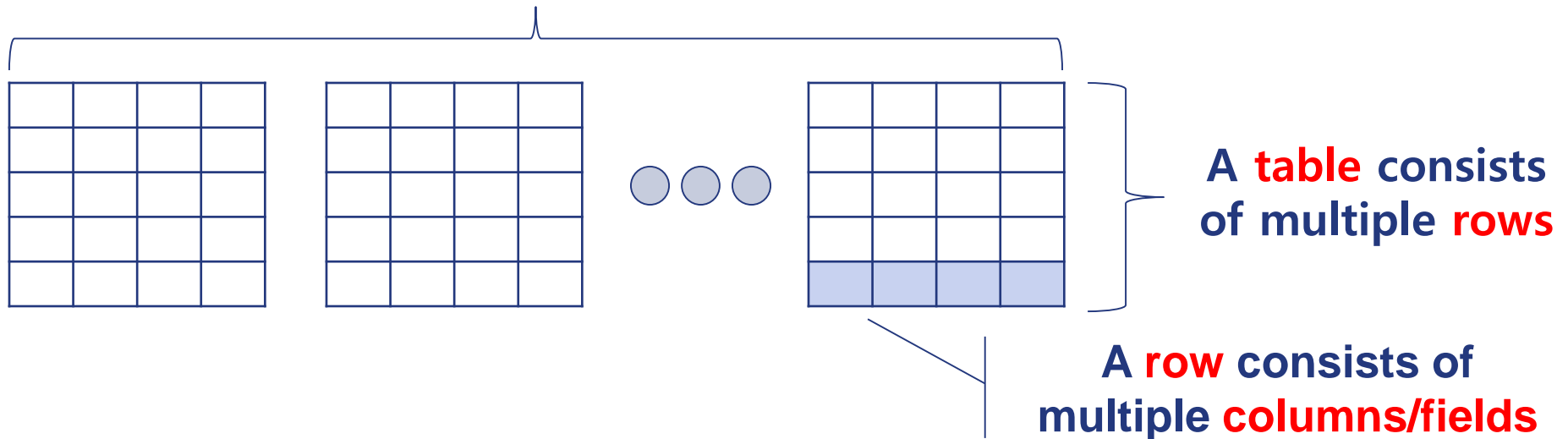
**Database & DBMS**  
**SQL**  
**JDBC**



# Database

- ❖ A **database** is an integrated collection of logically related records or files consolidated into a common pool that provides data for one or more multiple uses.

A **database** consists of multiple tables



# Database: An Example

---

## ❖ Subscription database

*(Patients Table)*

ACCT	LastName	FirstName	DateOfBirth	HomePhone
96709	Smith	John	3/3/1955	703-456-7645
635667	Bush	George	2/4/1934	202-345-8765
78643859	Washington	Edward	5/2/1945	301-567-3412

*(Medications Table)*

ACCT	Med_Name	Instructions	No_Pills	Refills
96709	Atenolol 50mg	take i po qd	60	5
96709	Maxzide 25	take i po qd	90	5
78643859	Zithromycin 250mg	take i po qd	6	0
635667	Zestril 40mg	take i po qd	100	5

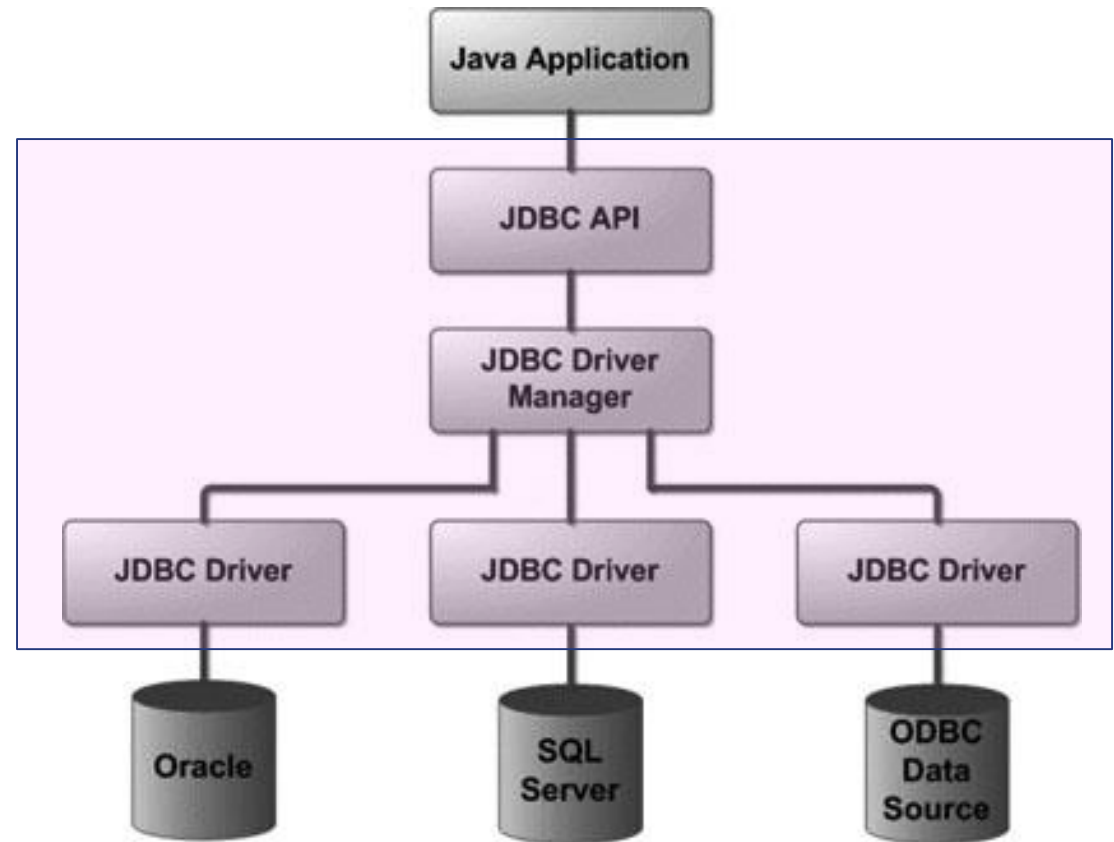
# DBMS

---

- ❖ A **Database Management System (DBMS)** is a set of software that controls the creation, maintenance, and the use of the database.
- ❖ Major DBMS products include Oracle, MS SQL, and MySQL
- ❖ DBMS supports
  - Concurrent processing for a huge number of users
  - High performance with a huge number of data
  - Security
  - Load balancing
  - Availability

# JDBC

- ❖ JDBC stands for **J**ava **D**atabase **C**onnectivity
- ❖ A standard Java API for communication between the Java applications and databases.
- ❖ Supports a wide range of DBMSs



# A glance at JDBC API

```
Connection conn = DriverManager.getConnection(  
    jdbc:mariadb://localhost/Subscription", "root", "admin");
```

```
Statement stmt = conn.createStatement();
```

```
String sql = "SELECT firstName, lastName FROM Patients";
```

```
ResultSet rs = stmt.executeQuery(sql);
```

```
while ( rs.next() ) {
```

```
    //Retrieve by column name
```

```
    String first = rs.getString("firstName"); String last = rs.getString("lastName");
```

```
    //Display values
```

```
    System.out.print(", First: " + first); System.out.println(", Last: " + last);
```

```
}
```

```
//STEP 6: Clean-up environment
```

```
rs.close();
```

```
stmt.close();
```

```
conn.close();
```

SQL

# JDBC

---

- ❖ JDBC can be used to write different types of executables, such as:
  - Java Applications
  - Java Applets
  - Java Servlets
  - Java ServerPages (JSPs)
  - Enterprise JavaBeans (EJBs)
  
- ❖ In other words, all of these different executables above are able to access a database with JDBC APIs

# Preparation for MySQL

---

- ❖ MySQL: <https://www.mysql.com/>
  - You can download it from [MySQL Official Site](https://www.mysql.com/downloads/).
  - <https://www.mysql.com/downloads/>
- ❖ MySQL is an open source database, but not free for commercial application development



- ❖ Instead, consider MariaDB
  - <https://mariadb.org/>
  - **MariaDB** is a community-developed fork of the MySQL
  - intended to remain free under the GNU GPL





# SQL (Structured Query Language)

❖ A standard language for manipulating database.

Function	SQL Syntax
Create Database	CREATE DATABASE database_name
Delete Database	DROP DATABASE database_name
Create Table	CREATE TABLE table_name ( column_name column_data_type, column_name column_data_type, ... )
Delete Table	DROP TABLE table_name
Insert Data	INSERT INTO table_name VALUES (column1, column2, ...)
Select Data	SELECT column_name, column_name, ... FROM table_name WHERE conditions
Update Data	UPDATE table_name SET column_name = value, column_name = value, ... WHERE conditions
Delete Data	DELETE FROM table_name WHERE conditions

# MySQL Client Interface: Command Line

---

```
% C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql
```

```
mysql> create database emp;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> show databases;
```

Database
information_schema
emp
mysql
test

```
4 rows in set (0.00 sec)
```

```
mysql> use emp;
```

```
Database changed
```

```
mysql> CREATE TABLE Employees
```

```
-> (  
->   id INT NOT NULL,  
->   age INT NOT NULL,  
->   first VARCHAR(255),  
->   last VARCHAR(255), PRIMARY KEY ( id )  
-> );
```

```
Query OK, 0 rows affected (0.01 sec)
```

# Create Database

---

- ❖ The CREATE DATABASE statement is used for creating a new database
- ❖ The following SQL statement creates a Database named EMP:

```
mysql> create database EMP;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> show databases;  
+-----+  
| Database  
+-----+  
| information_schema  
| emp  
| mysql  
| test  
+-----+  
4 rows in set (0.00 sec)
```

# Drop Database

---

- ❖ The DROP DATABASE statement is used for deleting an existing database

```
mysql> drop database EMP;  
Query OK, 1 row affected (0.00 sec)
```

- ❖ Be careful, deleting a database would loss all the data stored in database.

# Create Table

---

- ❖ The CREATE TABLE statement is used for creating a new table.
- ❖ The following SQL creates a table named Employees with four columns:

```
mysql> use Emp;
Database changed

mysql> CREATE TABLE Employees
(
  id INT NOT NULL,
  age INT NOT NULL,
  first VARCHAR(255),
  last VARCHAR(255), PRIMARY KEY ( id )
);
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+-----+
| Tables_in_emp |
+-----+
| employees      |
+-----+
1 row in set (0.00 sec)
```

# Drop Table

---

- ❖ The DROP TABLE statement is used for deleting an existing table.
- ❖ The following SQL statement deletes a table named Employees:

```
mysql> DROP TABLE Employees;  
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> show tables;  
Empty set (0.00 sec)
```

# Insert Data

- ❖ INSERT INTO table\_name VALUES (column1, column2, ...);
- ❖ The following SQL INSERT statement inserts 4 rows in the Employees.

```
mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
Query OK, 1 row affected (0.05 sec)
mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');
Query OK, 1 row affected (0.00 sec)
mysql> select * from employees;
```

id	age	first	last
100	18	Zara	Ali
101	25	Mahnaz	Fatma
102	30	Zaid	Khan
103	28	Sumit	Mittal

```
4 rows in set (0.00 sec)
```

# Select Data

---

- ❖ The SELECT statement is used to retrieve data from a database.

```
mysql> SELECT column_name, column_name, ...  
      FROM table_name  
      WHERE conditions;
```

- ❖ The following SQL statement selects the age, first and last columns from the Employees table where id column is 100

```
mysql> SELECT first, last, age  
      FROM Employees  
      WHERE id = 100;
```

```
+-----+-----+-----+  
| first | last | age |  
+-----+-----+-----+  
| Zara  | Ali  | 18  |  
+-----+-----+-----+  
1 row in set (0.02 sec)
```



# Select Data

---

- ❖ Not all fields can be retrieved.

```
SQL> SELECT first, last
      FROM Employees
      WHERE id = 100;
+-----+-----+
| first | last |
+-----+-----+
| Zara  | Ali  |
+-----+-----+
1 row in set (0.00 sec)
```

- ❖ '\*' is used to indicate all the fields

```
SQL> SELECT first, last, age
      FROM Employees
      WHERE id = 100;
```

```
SQL> SELECT *
      FROM Employees
      WHERE id = 100;
```

# Select Data

---

❖ The WHERE clause can use the comparison operators

- =, !=, <, >, <=, and >=

```
SELECT first, last  
FROM Employees  
WHERE id >= 101
```

- BETWEEN

```
SELECT first, last  
FROM Employees  
WHERE id BETWEEN 100 AND 102
```

- LIKE operators.

```
SELECT first, last  
FROM Employees  
WHERE first LIKE '%a%'
```

# Select Data

---

- ❖ The WHERE clause can use the logical operators: AND, OR, NOT

```
SELECT first, last  
FROM Employees  
WHERE id != 100 OR last LIKE '%a%'
```

- ❖ The ORDER BY clause can be used to order the result.

```
SELECT first, last  
FROM Employees  
WHERE first LIKE '%a%'  
ORDER BY last DESC
```

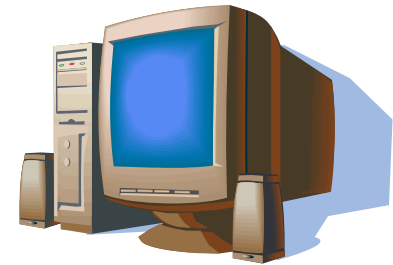
# Java Code using JDBC

---



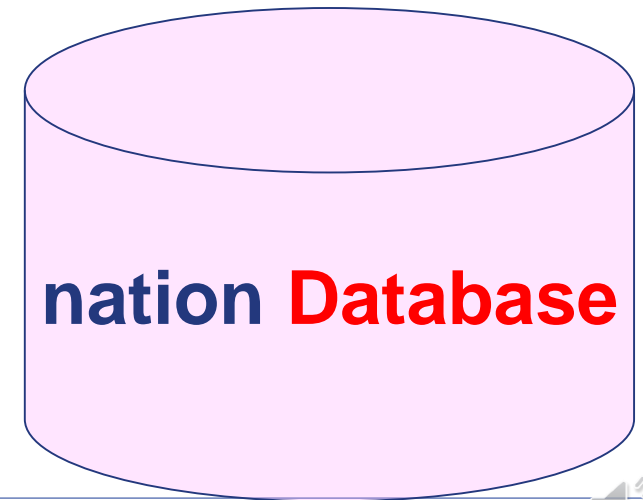
**SelectExample  
MariaDB**

- 
1. **Connect** to the MariaDB Server
  2. Send **SQL Select Query** for countries table

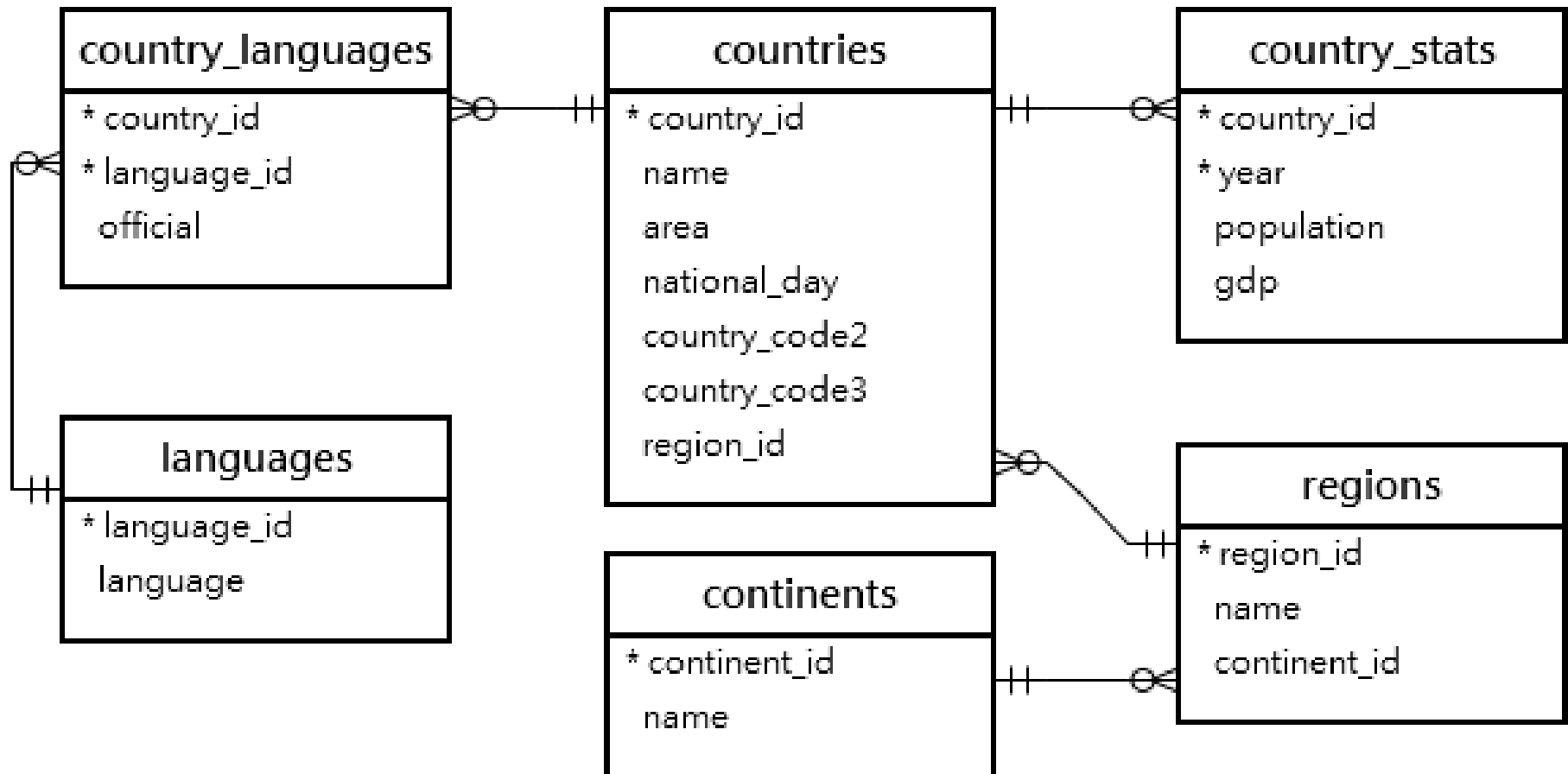


**MariaDB Server**

- 
3. Return the selected rows of countries table



# nation DB Schema



<https://www.mariadbtutorial.com/getting-started/mariadb-sample-database/>

# Creating nation DB

```
MariaDB [none]> source c:\\nation.sql;
```

```
...
```

```
MariaDB [nation]> show tables;
```

```
+-----+  
| Tables_in_nation |  
+-----+  
| continents        |  
| countries          |  
| country_languages |  
| country_stats     |  
| guests            |  
| languages          |  
| region_areas      |  
| regions           |  
| vips              |  
+-----+
```

```
9 rows in set (0.001 sec)
```

# Table: continents

```
MariaDB [nation]> select * from continents;
```

```
+-----+-----+
| continent_id | name          |
+-----+-----+
|           1 | North America |
|           2 | Asia          |
|           3 | Africa        |
|           4 | Europe        |
|           5 | South America |
|           6 | Oceania       |
|           7 | Antarctica    |
+-----+-----+
7 rows in set (0.000 sec)
```

continents
* continent_id
name

# Table: countries

```
MariaDB [nation]> select country_id, name, area from countries;
```

+-----+-----+-----+		
country_id	name	area
+-----+-----+-----+		
1	Aruba	193.00
2	Afghanistan	652090.00
3	Angola	1246700.00
4	Anguilla	96.00
5	Albania	28748.00
6	Andorra	468.00
7	Netherlands Antilles	800.00
8	United Arab Emirates	83600.00
9	Argentina	2780400.00



## // STEP 1. Import required packages

```
import java.sql.*;
public class SelectExampleMariaDB {
    private static final String JDBC_DRIVER = "org.mariadb.jdbc.Driver";
    private static final String DB_URL = "jdbc:mariadb://localhost/nation";
    private final String USER = "root";
    private static final String PASS = "root";
```

- URL indicates the database to be accessed
- URL Format depends on DBMS

```
public static void main(String[] args) {
    try {
```

## // STEP 2: Register JDBC driver

```
        Class.forName(JDBC_DRIVER);
```

Driver name depends on DBMS

```
    } catch (ClassNotFoundException e) { e.printStackTrace(); }
```

## //STEP 3: Open a connection

```
try ( Connection conn = DriverManager.getConnection(DB_URL,USER,PASS) ) {
```

## //STEP 4: Execute a query

```
try ( Statement stmt = conn.createStatement() ) {
```

```
    String sql = "SELECT country_id, name, area FROM countries"
```

```
        + " WHERE name LIKE '%K%' AND area BETWEEN 50000 AND 200000"
```

```
        + " ORDER BY area";
```

```
try ( ResultSet rs = stmt.executeQuery(sql) ) {
```



### **// STEP 5: Extract data from result set**

```
while ( rs.next() ) {  
    //Retrieve by column name  
    int id  = rs.getInt("country_id");  
    String name = rs.getString("name");  
    double area = rs.getDouble("area");  
  
    //Display values  
    System.out.println("ID: " + id + ", Name: " + name + ", Area: " + area);  
}  
}  
}  
conn.close();  
} catch ( SQLException se) { se.printStackTrace(); }  
System.out.println("Goodbye!");  
}  
}
```

Connecting to database...

ID: 125, Name: Sri Lanka, Area: 65610.0

ID: 117, Name: South Korea, Area: 99434.0

ID: 174, Name: North Korea, Area: 120538.0

ID: 209, Name: Tajikistan, Area: 143100.0

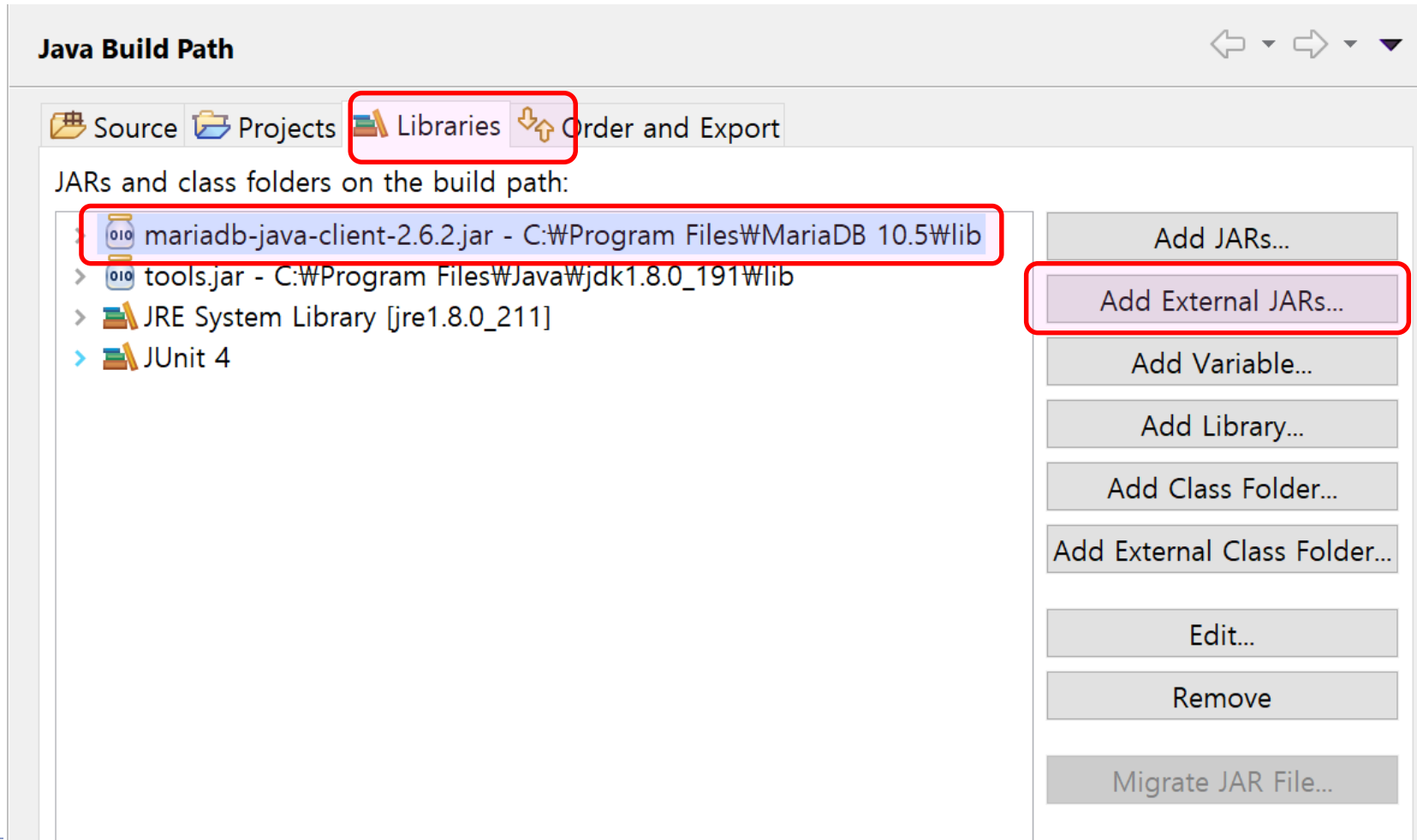
ID: 113, Name: Kyrgyzstan, Area: 199900.0

Goodbye!




# Set the JDBC library in Eclipse

<https://mariadb.com/downloads/connector>



# Executing SQL

```
Connection conn = null ; Statement stmt = null ;  
try {  
    conn = DriverManager.getConnection(DB_URL,USER,PASS);  
    stmt = conn.createStatement();  
    String sql = "SELECT country_id, name, area FROM countries";  
    ResultSet rs = stmt.executeQuery(sql);  
}  
finally {  
    conn.close() ;  
}
```



Type	Purpose
executeQuery	SELECT
executeUpdate	INSERT, DELETE, UPDATE, DDL
execute	All the above commands

# PreparedStatement

---

- ❖ Efficiently execute the statement multiple times with different parameters

```
try ( Connection conn = DriverManager.getConnection(DB_URL,USER,PASS) ) {  
    // STEP 4: Execute a query  
    String sql = "UPDATE countries SET name = ? WHERE country_id = ?";  
    try ( PreparedStatement stmt = conn.prepareStatement(sql) ) {  
        stmt.setString(1, "Republic of Korea") ;  
        stmt.setInt(2, 117) ;  
  
        final int rows = stmt.executeUpdate();  
        System.out.println("Rows impacted : " + rows );  
    }  
} catch ( SQLException se) { se.printStackTrace(); }  
System.out.println("Goodbye!");
```

# Preparing JDBC for DBMS

---

- ❖ Install the **driver library** by one of the following ways.
  1. % java **-cp driver.jar** MyProg
  2. Into **CLASSPATH**
  3. Into **jre/lib/ext**
  
- ❖ Register the **driver class** by one of the following ways.
  1. java **-Djdbc.drivers=***driver\_name* MyProg
  2. System.setProperty("jdbc.drivers", "*driver\_name*") ;
  3. Class.forName("*driver\_name*") ;

# JDBC Driver Name for DBMS

---

❖ Driver names vary with DBMSs

RDBMS	JDBC driver name	Connection URL Format
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://hostname:port/DBName
MariaDB	org.mariadb.jdbc.Driver	jdbc:mariadb://hostname:port/DBName
ORACLE	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@hostname:port:DBName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	jdbc:db2:hostname:port/DBName
Sybase	com.sybase.jdbc.SybDriver	jdbc:sybase:Tds:hostname:port/DBName

# SQL Types and Java Types

---

SQL	JDBC/Java	setXXX	updateXXX
VARCHAR	java.lang.String	setString	updateString
CHAR	java.lang.String	setString	updateString
LONGVARCHAR	java.lang.String	setString	updateString
BIT	boolean	setBoolean	updateBoolean
NUMERIC	java.math.BigDecimal	setBigDecimal	updateBigDecimal
TINYINT	byte	setByte	updateByte
SMALLINT	short	setShort	updateShort
INTEGER	int	setInt	updateInt
BIGINT	long	setLong	updateLong
REAL	float	setFloat	updateFloat



# SQL Types and Java Types

SQL	JDBC/Java	setXXX	updateXXX
FLOAT	float	setFloat	updateFloat
DOUBLE	double	setDouble	updateDouble
VARBINARY	byte[ ]	setBytes	updateBytes
BINARY	byte[ ]	setBytes	updateBytes
DATE	java.sql.Date	setDate	updateDate
TIME	java.sql.Time	setTime	updateTime
TIMESTAMP	java.sql.Timestamp	setTimestamp	updateTimestamp
CLOB	java.sql.Clob	setClob	updateClob
BLOB	java.sql.Blob	setBlob	updateBlob
ARRAY	java.sql.Array	setARRAY	updateARRAY
REF	java.sql.Ref	SetRef	updateRef
STRUCT	java.sql.Struct	SetStruct	updateStruct

# References

---

- ❖ JDBC Tutorial, Tutorials Point

<http://www.tutorialspoint.com/jdbc/index.htm>

- ❖ MySQL Tutorial <http://www.tutorialspoint.com/mysql/index.htm>

- ❖ MariaDB Tutorial

<https://www.mariadbtutorial.com/>

<https://www.tutorialspoint.com/mariadb/index.htm>

# Q&A

---