Database Systems

Lecture #11

Sang-Wook Kim
Hanyang University

Objectives

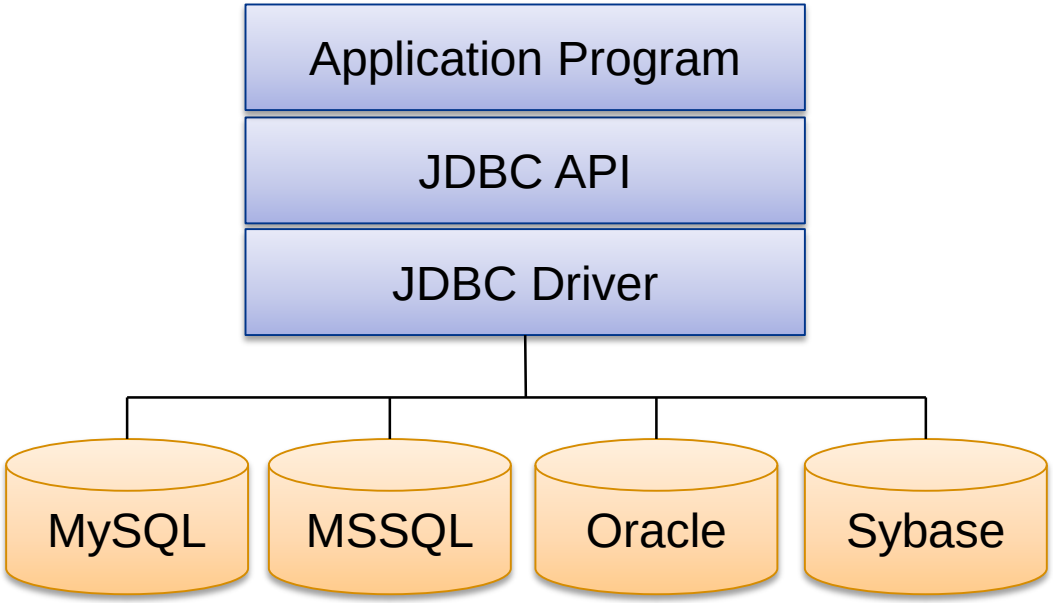
- To learn application programming with a DBMS
  - Java
  - JDBC (MySQL)

Outline

- Application Programming with JDBC
- Querying
- Processing Query Results
- Example

What is JDBC?

- Java Database Connectivity (JDBC)
  - Standard SQL API for Java
  - Provides an environment for developing DB applications
  - Supports any DBMS with a JDBC Driver



Application Programming with JDBC

- Step 1: Load an JDBC Driver
- Step 2: Open a connection to the DBMS
- Step 3: Execute SQL queries
- Step 4: Handle query results
- Step 5: Close a connection to the DBMS

Loading JDBC Driver

- MySQLConnector/J Driver

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

- Sybase jConnect Driver

```
Class.forName("com.sybase.jdbc4.jdbc.SybDriver")
    .newInstance();
```

- Oracle Driver

```
Class.forName("com.oracle.jdbc.OracleDriver")
    .newInstance();
```

Opening Database

- Database URL string (for MySQL)
  - `jdbc:mysql://[hostname]:[port#]/[dbname]`
  - Example
    - Connect to **MySQL**(port#: **3306**) database **test** in **localhost**

```
"jdbc:mysql://localhost:3306/test"
```

Opening Database

- Use the DriverManager class to connect

```
String url = [DBMS Connection URL];

String username = [User name];

String password = [User password];

Connection con = DriverManager.getConnection
                    (url, username, password);
```

Querying

- Create a new Statement instance

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

ResultSet rs = stmt.executeQuery
                    ( "SELECT name FROM student" );
```

Querying

- Query methods in the Statement class
  - executeUpdate()
    - For queries which modify database
      - INSERT, UPDATE, DELETE
    - Returns number of modified records by the query

```
stmt.executeUpdate
    ( "UPDATE student SET age=18
       WHERE name='Kil-Dong Hong'" );
```

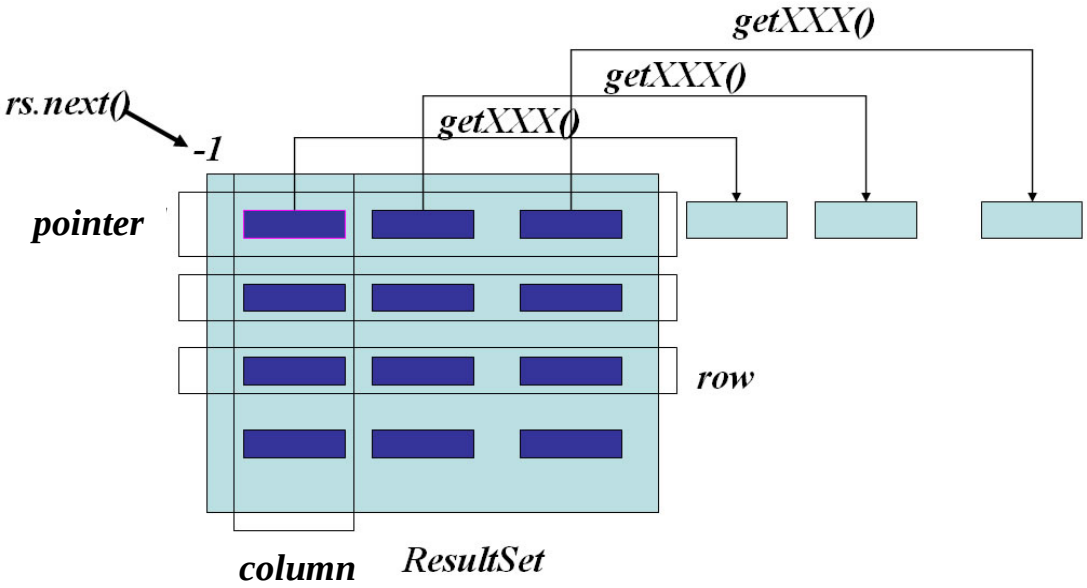
Querying

- Query methods in the Statement class
  - executeQuery()
    - For search queries (SELECT)
    - Return a ResultSet instance which contains the result of search query

```
ResultSet rs = stmt.executeQuery
                ( "SELECT name FROM student" );
```

Processing Query Results

- ResultSet object has rows which match the SELECT query
- ResultSet has a pointer to the current row
  - Starting from -1
  - Run next() method will proceed the pointer
- Use get[*TYPE*] methods to retrieve a value from a column
  - Specify column with parameter: get[TYPE](String columnName) or (int columnIndex)



Types for JDBC

Method	Return Type	Compatible SQL Data Type
getBoolean	boolean	BIT
getByte	byte	TINYINT
getBytes	byte[]	BINARY, VARBINARY
getDate	Date	DATE
getDouble	double	DOUBLE
getFloat	float	FLOAT
getInt	int	INT
getLong	long	BIGINT
getShort	short	SMALLINT
getString	String	CHAR, VARCHAR
getTime	Time	TIME
getTimestamp	Timestamp	TIMESTAMP
getAsciiStream	InputStream	LONGVARCHAR, CLOB
getBinaryStream	InputStream	LONGVARBINARY, BLOB

Processing Query Results

- Example
  - Retrieve each column value from the ResultSet

```
Connection con = DriverManager.getConnection( url, user, psw );
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( "SELECT name, age FROM student" );
while ( rs.next() ) {
            String name = rs.getString( "name" );
            int age = rs.getInt(2);
            System.out.println("(1)name: " + name + " (2)age: " + age);
}
```

Closing DB

- When processing is completed
  - Call the close() method for ResultSet, Statement, Connection instances

```
rs.close();

stmt.close();

con.close();
```

Example

```
import java.sql.*; // Import SQL packages

public class Example
{
    public static void main( String[] args )
    {
        try {
            // 1. Load JDBC Driver (not necessary)
            Class.forName( "com.mysql.jdbc.Driver" ).newInstance();

            // 2. Open DBMS connection
            String url = "jdbc:mysql://localhost:3306/test";
            String user = "user";
            String psw = "password";
            Connection con = DriverManager.getConnection( url, user, psw );
```

Example

```
// 3. Execute SQL query
// -- Using executeUpdate method
Statement stmt = con.createStatement();
stmt.executeUpdate
  ( "INSERT INTO student(name, age) VALUES
    ( 'Kil-Dong Hong', 18 )" );
// -- Using executeQuery method
ResultSet rs = stmt.executeQuery
  ( "SELECT name, age FROM student" );
```

Example

```
// 4. Process search query result
while( rs.next() )
{
        String name = rs.getString( "name" );
        int age = rs.getInt(2);
        System.out.println
           ( "(1)name: " + name + " (2)age: " + age );
}
```

Example

```
                    // 5. Close DBMS connection
                    rs.close();
                    stmt.close();
                    con.close();

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

Summary

- JDBC
  - Step 1: Load an JDBC Driver
  - Step 2: Open a connection to the DBMS
  - Step 3: Execute SQL queries
  - Step 4: Handle query results
  - Step 5: Close a connection to the DBMS

References

- MySQL Installer for Windows (http://dev.mysql.com/downloads/windows/installer/)
- Using JDBC with MySQL, Getting Started (http://www.developer.com/java/data/article.php/3417381/Using-JDBC-with-MySQL-Getting-Started.htm)

Have a nice day!