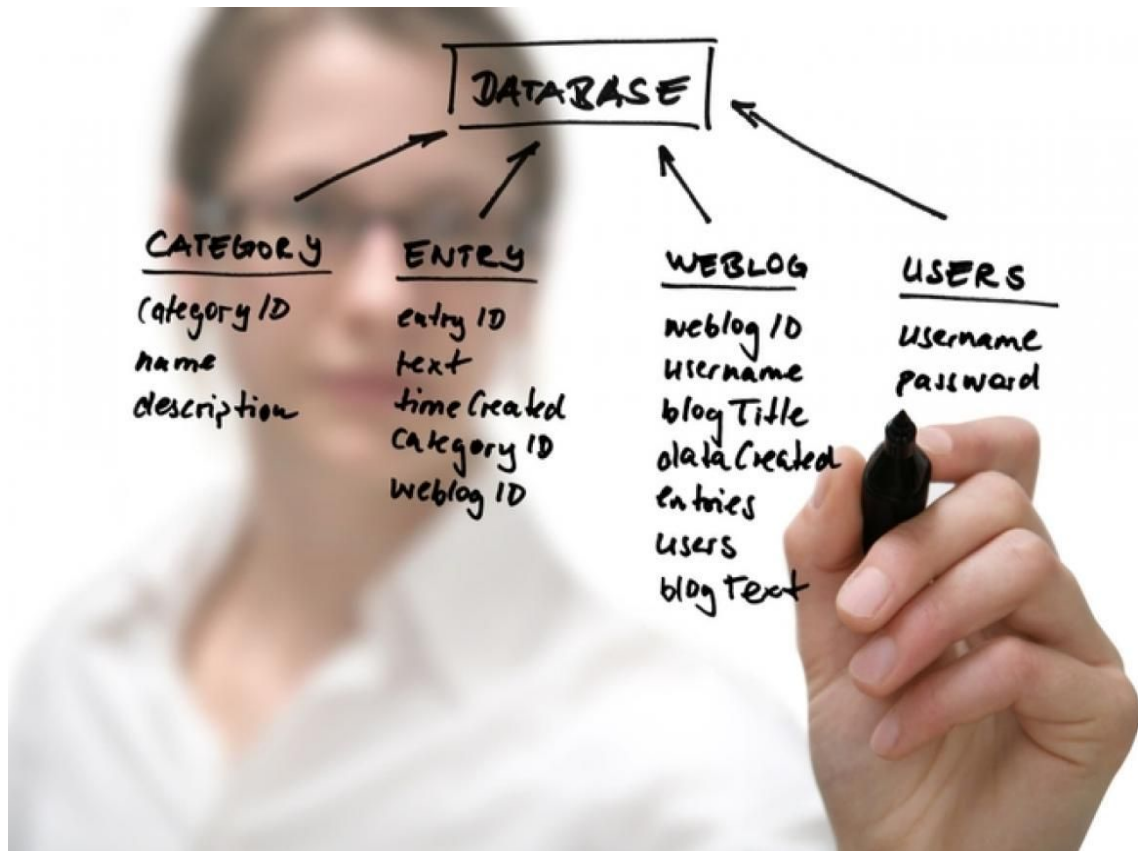


JDBC



Team members

Mazen Elmesery (50)

Mohamed Sharaf (54)

Ahmed hesham (10)

Abdelrahman Omran (37)

Java Database Connectivity (JDBC) provides Java developers with a standard API that is used to access databases, regardless of the driver and database product. JDBC presents a uniform interface to databases - change vendors and your applications only need to change their driver.

```

classDiagram
    class Driver {
        +Driver()
        +acceptsURL(String): boolean
        +connect(String, Properties) Connection
        +getPropertyInfo(String, Properties) DriverPropertyInfo[]
        +getMajorVersion(): int
        +getMinorVersion(): int
        +getParentLogger(): Logger
        +jdbcCompliant(): boolean
    }
    class DataBaseMeta {
        +dbPath: File
        +DataBaseMeta(File)
        +getData(): void
    }
    class MyConnection {
        +dbPath: File
        +MyConnection(Database, File)
        +isWrapperFor(Class<T>): boolean
        +unwrap(Class<T>)
    }
    class MyResultSet {
        +resultSet: MyResultSet
        +tableName: String
        +result: Object[]
        +MyResultSetMeta(Data, Object[][], String, DataBaseMeta)
        +getColumnCount(): int
        +getColumnLabel(int): String
        +getColumnNames(int): String
        +nextColumnIndex(int): int
    }
    class MyResultSetMeta {
        +resultSet: MyResultSet
        +tableName: String
        +result: Object[]
    }
    class MyStatement {
        +listOfCommands: LinkedList<String>
        +con: Connection
        +queryTimeOutDefaultValue: int
        +queryTimeOut: int
        +tableName: String
    }
    class Database {
        +createDatabase(String, boolean): String
        +executeStructureQuery(String): boolean
        +executeQuery(String): Object[]
        +executeUpdateQuery(String): int
    }
    class ColumnsAndValues {
        +column: String
        +value: String
        +ColumnsAndValues()
        +ColumnsAndValues(String, String)
    }
    class TableData {
        +name: String
        +TableData(String, LinkedList<ColumnsAndValues>)
    }

    Driver --> DataBaseMeta
    Driver --> TableData
    Driver --> MyConnection
    Driver --> MyResultSet
    Driver --> MyStatement
    Driver --> Database
    Driver --> ColumnsAndValues

    DataBaseMeta --> MyConnection
    DataBaseMeta --> MyResultSet
    DataBaseMeta --> MyResultSetMeta
    DataBaseMeta --> MyStatement
    DataBaseMeta --> Database
    DataBaseMeta --> ColumnsAndValues

    MyConnection --> MyResultSet
    MyConnection --> MyResultSetMeta
    MyConnection --> MyStatement
    MyConnection --> Database
    MyConnection --> ColumnsAndValues

    MyResultSet --> MyResultSetMeta
    MyResultSet --> MyStatement
    MyResultSet --> Database
    MyResultSet --> ColumnsAndValues

    MyResultSetMeta --> MyStatement
    MyResultSetMeta --> Database
    MyResultSetMeta --> ColumnsAndValues

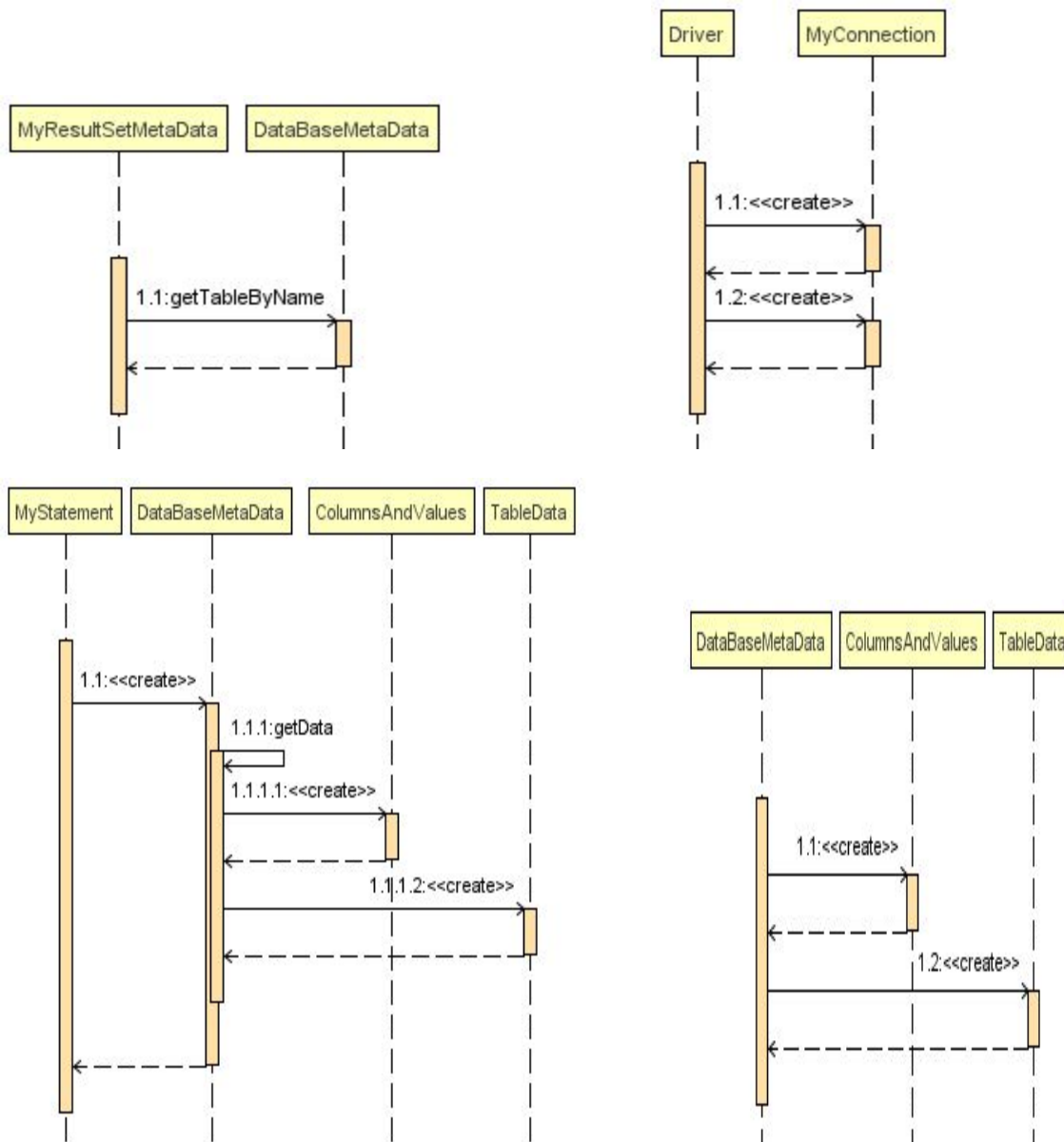
    MyStatement --> Database
    MyStatement --> ColumnsAndValues

    Database --> ColumnsAndValues

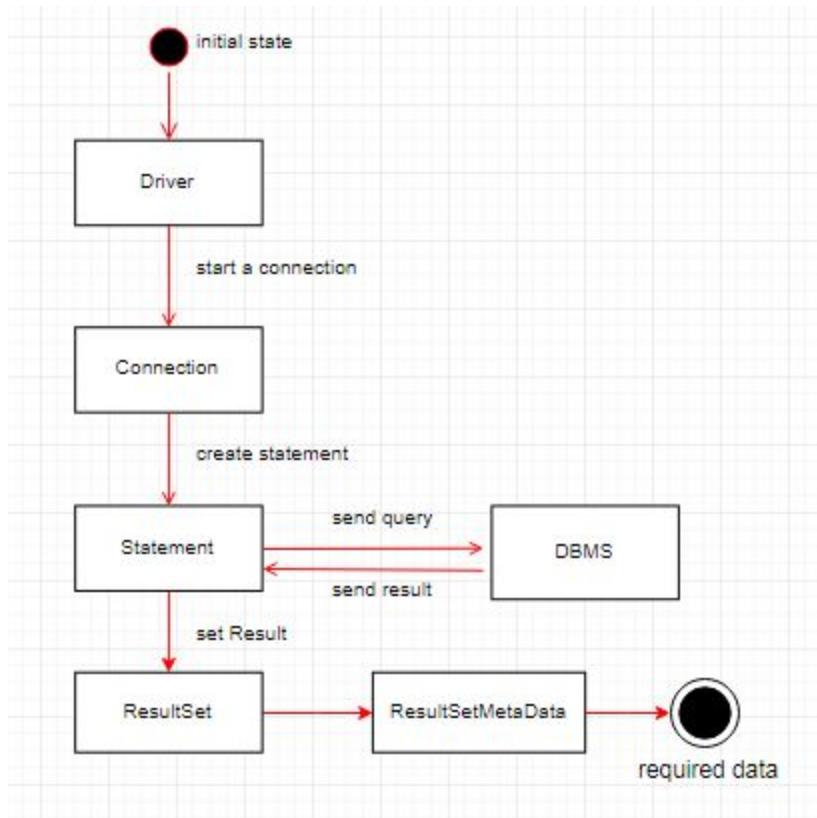
    ColumnsAndValues --> TableData
  
```

The diagram illustrates the relationships between various database-related classes and interfaces. The **Driver** class is the central component, interacting with **DataBaseMeta**, **TableData**, **MyConnection**, **MyResultSet**, **MyStatement**, **Database**, and **ColumnsAndValues**. **DataBaseMeta** acts as a bridge, connecting the **Driver** to the **MyConnection** and **MyResultSet** classes, as well as the **Database** interface and **ColumnsAndValues** class. **MyConnection** is responsible for managing the database connection and provides methods to create **MyResultSet** objects and execute queries. **MyResultSet** and **MyResultSetMeta** classes are used to represent the results of database queries. **MyStatement** is used to execute SQL statements and queries. The **Database** interface defines the methods for creating databases, executing queries, and updating data. **ColumnsAndValues** is a class used to represent the columns and values of a table. **TableData** is a class used to represent the data of a table.

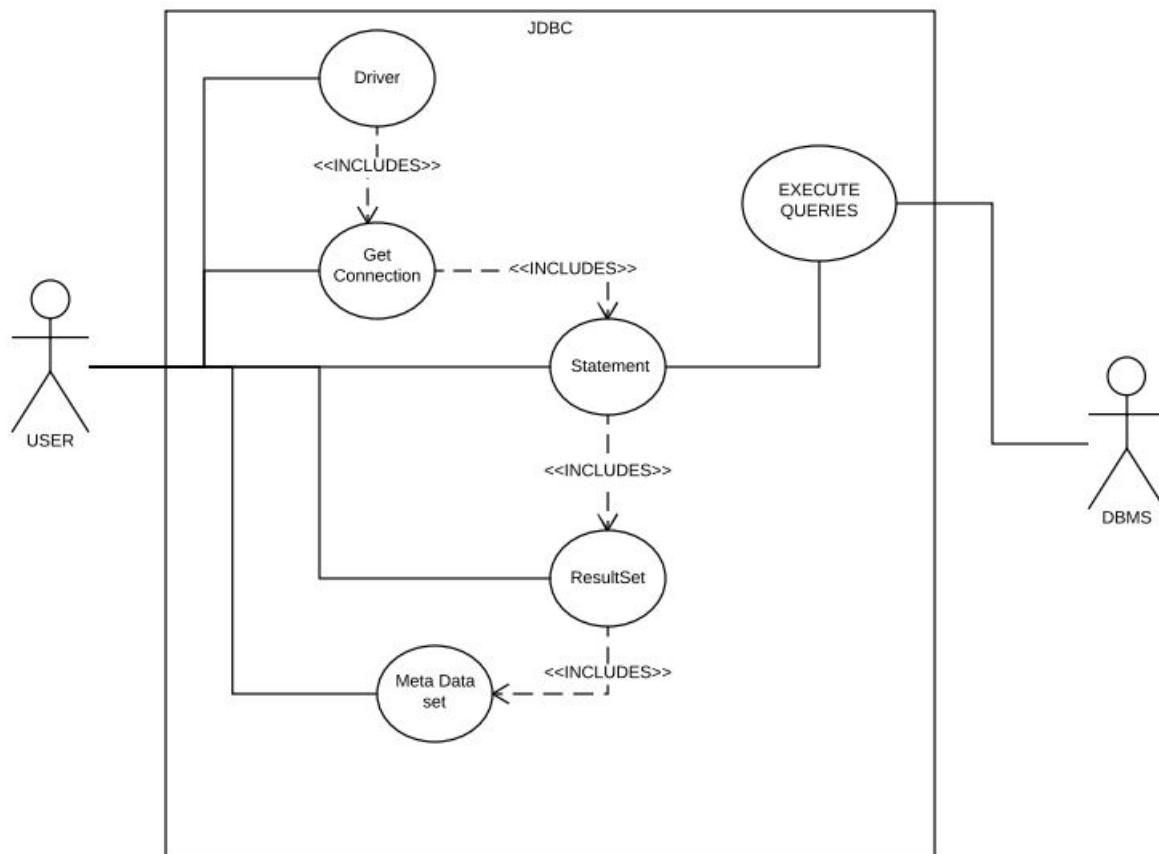
Sequence diagram



State diagram



Use case



Division of labor

Mohamed sharaf & Ahmed hesham

Testing the DBMS and make sure it is working properly and fixing the old bugs

Abdelrahman & Mazen Elmesery

Implementing the JDBC and connect it to the DBMS following piazza discussions
Testing the JDBC and make sure it is working fine .

Design description

Driver : checks the url and make the path in the info list and establishes a connection with the DBMS

MyConnection: creates a statement with an instance from Database class and the path

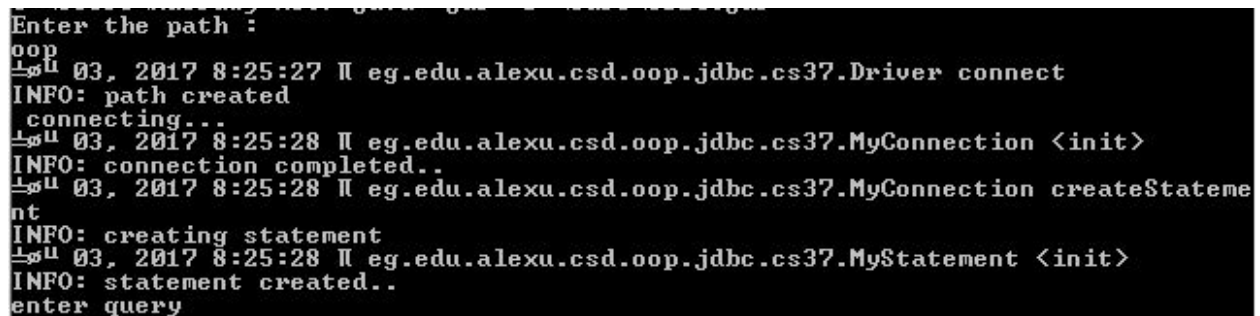
MyStatement : it is responsible for validating the coming queries then executing them and set the result in a ResultSet (The SELECT query).

MyResultSet : it is responsible for applying different operations on the result data like getting the index of column by label , iterating over the rows , etc..

MyResultSetMetaData : it is used for getting some data from the result such as getting the number of columns, the columns labels and names , the columns types, the table name,

User interface

Establishing a connection between the user and DBMS



```
Enter the path :
oop
03, 2017 8:25:27 II eg.edu.alexu.csd.oop.jdbc.cs37.Driver connect
INFO: path created
connecting...
03, 2017 8:25:28 II eg.edu.alexu.csd.oop.jdbc.cs37.MyConnection <init>
INFO: connection completed..
03, 2017 8:25:28 II eg.edu.alexu.csd.oop.jdbc.cs37.MyConnection createStateMe
nt
INFO: creating statement
03, 2017 8:25:28 II eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement <init>
INFO: statement created..
enter query
```

Creating database

```
enter query
CREATE DATABASE db1;
15:03, 2017 8:27:18 I eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement execute
INFO: Query is valid
15:03, 2017 8:27:18 I eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement execute
INFO: create query applied
enter query
```

Creating table

```
CREATE TABLE table1 (column1 int, column2 varchar);
15:03, 2017 8:29:27 I eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement execute
INFO: Query is valid
15:03, 2017 8:29:27 I eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement execute
INFO: create query applied
```

Update table

```
enter query
insert into table1 (column1,column2) values (9, grade);
15:04, 2017 6:56:36 F eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement execute
INFO: Query is valid
15:04, 2017 6:56:36 F eg.edu.alexu.csd.oop.jdbc.cs37.MyStatement executeUpdate
INFO: executing update..
enter query
```