MODUL 6
KONEKSI DATABASE MYSQL 5.0 VIA SOCKET
(Menggunakan Pemrograman Java)

A. Tujuan

1. Memberikan gambaran tentang konsep koneksi ke database MySQL
2. Mampu membangun aplikasi object terdistribusi dengan menggunakan socket programming yang terkonkesi dengan database.

B. Pendahuluan

Dalama membangun aplikasi yang memanfaatkan database dalam java ada beberapa hal yang perlu dilakukan instlasi yaitu :

1. Memasang Java dan JDBC.
2. Memasang sebuah driver pada mesin Anda
3. Memasang jika DBMS jika perlu

Setelah tahapan instalasi tersebut dilakukan pengaturan database, buat sebuah database di MySQL dengan nama missal : test. Tahapan selanjutnya adalah membangun koneksi. Dalam membangun koneksi perlu dilakukan dua langkah :

- Memuat driver, dalam memuat driver yang ingin anda gunakan caranya sangat sederhana dan hanya perlu satu baris code missal untuk database MySQL :
  Class.*forName*("com.mysql.jdbc.Driver");
- Membuat koneksi, dalam membuat koneksi kodingnya adalah
  String data = "jdbc:mysql://"+server+":3306/"+db;
  Connection con = DriverManager.*getConnection*(data,user,pswd);

Langkah berikutnya membuat table, untuk membuat table dapat digunakan tools yang lain suapaya tidak selalu bermain dalam koding, missal menggunakan Navicat, phpmyadmin. Setelah kita membuat table, kita dapat melakukan beberapa operasi seperti select, insert update atau delete. Untuk dapat melakukan operasi tersebut maka diperlukan beberapa koding sebelumnya yaitu :

Membuat Statement
Gunakan object Connection, yang dalam contoh

```
Connection con = DriverManager.getConnection(data,user,pswd);
Statement stm = con.createStatement();
```

Untuk menjalankan Statement dapat menggunakan 3 perintah berikut, tergantung dari apa yang mau dieksekusi

- Untuk perintah SELECT, maka menggunakan method executeQuery(String sql).
- Untuk perintah INSERT, UPDATE, DELETE, CREATE, DROP menggunakan method executeUpdate(String sql).

- Untuk stored procedure dengan hasil multiple dengan menggunakan method execute(String sql).

ResultSet

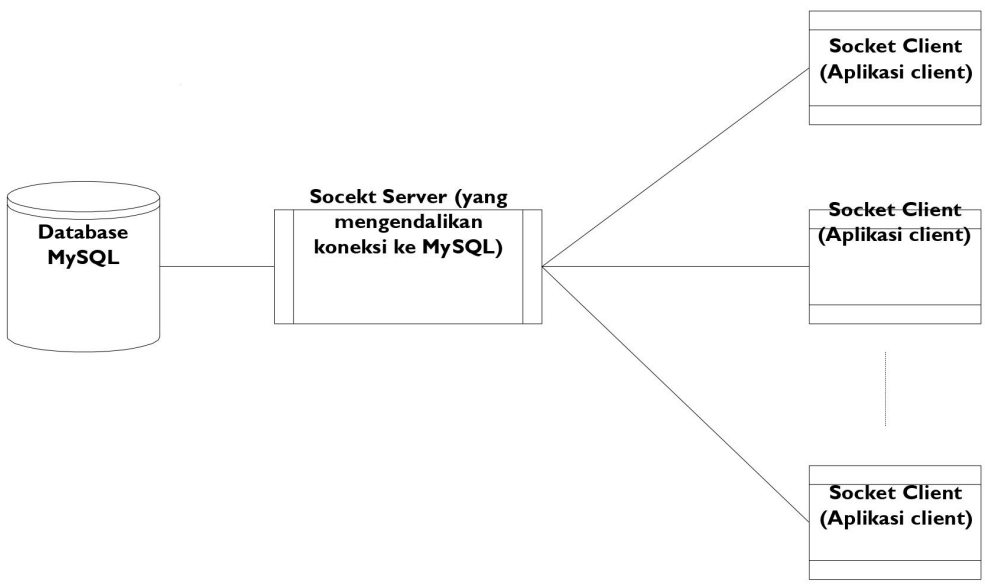
Sebuah objek ResultSet adalah objek java yang berisi hasil-hasil sewaktu menjalankan sebuah query SQL, hasilnya berupa baris-baris yang memenuhi kondisi query tersebut. Data yang disimpan dalam ResulsSet diambil melalui serangkain method get yang mengijinkan akses ke berbagai kolom dari baris. Contohnya :

```
ResultSet rs = stmt.executeQuery("SELECT * FROM mahasiswa");
while(rs.next()){
        System.out.println("Nim : " + rs.getInt(1));
        System.out.println("Nama : "+ rs.getString(2));
}
```

Perintah untuk mengambil data dari table mahasiswa pada kolom pertama menggunakan method getInt(1), Int menunjukkan type data dari data yang disimpan dalam table kolom pertama, sedangkan 1 menunjukkan kolom pertama. Kemudian getString(2), berarti untuk mengambil data pada kolom kedua, String menunjukkan type data yang tersimpan dalam kolom kedua.

C. Praktikum

Praktikum 1 : Kita akan membangu aplikasi tree-tier berbasis socket programming, dengan model aplikasinya sebagai berikut :



Membuat class mahasiswa

```
package test;
import java.io.Serializable;
public class mahasiswa implements Serializable {
        private int nim;
        public int getNim() {
                return nim;
        }
        public void setNim(int nim) {
                this.nim = nim;
        }
        public String getNama() {
                return nama;
        }
        public void setNama(String nama) {
                this.nama = nama;
        }
        private String nama;

}
```

Kemudian membuat class kuliah

```
package test;
import java.io.Serializable;
import java.util.ArrayList;
public class kuliah implements Serializable {
        private ArrayList<mahasiswa> isi = null;
        public ArrayList<mahasiswa> getIsi() {
                return isi;
        }
        public void setIsi(ArrayList<mahasiswa> isi) {
                this.isi = isi;
        }
}
```

Langkah selanjutnya kita buat class yang bertuga untuk melakukan koneksi ke database MySQL 5.0, database yang akan dikoneksi kita beri nama **test** dan didalamnya ada table mahasiswa. Sedangkan user =root dan paswordnya = “”.

```
package test;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.util.ArrayList;
public class conDatabase {
	private Connection con=null;
	private Statement stm = null;
	public conDatabase(String server, String db, String user, String pswd) {
		super();
		try {
			Class.forName("com.mysql.jdbc.Driver");
			String data = "jdbc:mysql://"+server+":3306/"+db;
			con = DriverManager.getConnection(data,user,pswd);
			stm = con.createStatement();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
	public ResultSet query(int nim) throws SQLException{
		String sql = "SELECT * FROM mahasiswa WHERE nim="+nim;
		return this.stm.executeQuery(sql);
	}
	public ArrayList<mahasiswa> getMahasiswa(int nim){
		ArrayList<mahasiswa> isi = new ArrayList<mahasiswa>();
		try {
			ResultSet rs = this.query(nim);
			while(rs.next()){
				mahasiswa aa = new mahasiswa();
				aa.setNim(rs.getInt(1));
				aa.setNama(rs.getString(2));
				isi.add(aa);
			}
		} catch (Exception e){
			e.printStackTrace();
		}
		return isi;
	}
	public void inputData(int a, String b) throws SQLException{
		String sql ="INSERT INTO mahasiswa values("+a+",'"+b+"')";
		stm.executeUpdate(sql);
	}
}
```

Langkah selanjutnya membuat class yang merupakan turunan dari class Thread untuk memanfaatkan class conDatabase, yang nantinya juga bertugas untuk malakukan proses atas permintaan dari client socket. Socket ini berbasis multithreading sehinggga dapat melayani banyak client.

```
package test;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream; import
java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
public class connect extends Thread {
        Socket client;
        mahasiswa x = null;
        Connection con=null;
        Statement stm = null;
        siswa z= null;
        public connect(Socket client) {
                super();
                this.client = client;
                try {
                        InputStream is = client.getInputStream();
                        ObjectInputStream ois = new ObjectInputStream(is);
                        OutputStream os = client.getOutputStream();
                        ObjectOutputStream oos = new ObjectOutputStream(os);
                        x = (mahasiswa) ois.readObject();
                        conDatabase  test =
                        new conDatabase("localhost","test","root","");
                        test.inputData(x.getNim(), x.getNama());
                        ArrayList<mahasiswa> isi = test.getMahasiswa(x.getNim());
                        z = new kuliah();
                        z.setIsi(isi);
                        oos.writeObject(z);
                        oos.flush();
                        ois.close();
                        oos.close();
                } catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

Berikutnya membuat server socket :

```
package test;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream; import
java.net.ServerSocket; import
java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class matServer extends Thread {
        public ServerSocket server = null;
        public matServer(){
                try {
                        System.out.print("Server is Running.....");
                        server = new ServerSocket(4343);
                        this.start();
                } catch(Exception e){
                        e.printStackTrace();
                }
        }
        public static void main(String[] args) {
                new matServer();
        }
        public void run(){
                try {
                        while(true){
                                Socket client = server.accept();
                                connect cc = new connect(client);
                        }
                } catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

Langkah yang terakhir membuat socket client :

```
package test;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream;
```

```
import java.net.Socket;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
public class matClient {
	public static void main(String[] args) {
		mahasiswa data = null;
		siswa  asil = null;
		try {
			Socket soket = new Socket("127.0.0.1",4343);
			OutputStream os = soket.getOutputStream();
			ObjectOutputStream oos = new ObjectOutputStream(os);
			InputStream is = soket.getInputStream();
			ObjectInputStream ois = new ObjectInputStream(is);
			data = new mahasiswa();
			Scanner sc  = new Scanner(System.in);
			System.out.print("Inputkan nim : ");
			data.setNim(sc.nextInt());
			System.out.print("Inputkan nama : ");
			data.setNama(sc.next());
			oos.writeObject(data);
			oos.flush();
			asil = (kuliah) ois.readObject();
			ArrayList<mahasiswa> oke = asil.getIsi();
			Iterator it = oke.iterator();
			while(it.hasNext()){
				mahasiswa ace = (mahasiswa) it.next();
				System.out.println("Nim : "+ ace.getNim());
				System.out.println("Nama : "+ ace.getNama());
			}

			oos.close();
			ois.close();
		}catch(Exception e){
			e.printStackTrace();
		}
	}

}
```

Kemudian Anda compile dan coba running, dengan running dulu untuk client socket baru kemudian server socket.

D. Latihan

1. Modifikasi pada praktikum 1 sehingga dapat digunakan untuk proses DELETE dan UPDATE data pada table mahasiswa.