Modul 10 - Java Database Connectivity (JDBC)

Durasi Modul = 2-5 menit

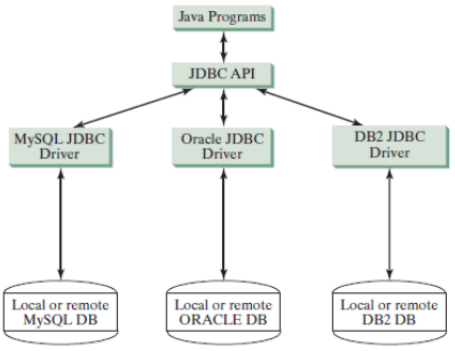
Capaian pembelajaran:

1. Mahasiswa dapat menjelaskan fungsi dan peran JDBC dalam integrasi aplikasi Java dengan basis data.
2. Mahasiswa mampu melakukan koneksi ke database dan mengeksekusi berbagai jenis query SQL menggunakan teknik JDBC.
3. Mahasiswa dapat mengelola hasil query menggunakan ResultSet dan menerapkan konsep CRUD (Create, Read, Update, Delete) dalam konteks objek.
4. Mahasiswa memahami konsep transaksi database, termasuk pengelolaan commit dan rollback, serta penanganan exception yang efektif.

Deskripsi Pembelajaran

Mata kuliah JDBC *Java Database Connectivity* (JDBC) dirancang untuk memberikan pemahaman mendalam tentang cara menghubungkan aplikasi Java dengan berbagai sistem basis data. Dalam perkuliahan ini, mahasiswa akan mempelajari konsep dasar JDBC, termasuk konfigurasi lingkungan pengembangan dan pengaturan driver database. Materi yang diajarkan mencakup cara melakukan koneksi ke berbagai jenis database, mengeksekusi query SQL, serta mengelola hasil query menggunakan objek ResultSet. Dengan mengintegrasikan JDBC ke dalam aplikasi berbasis pemrograman berorientasi objek, mahasiswa diharapkan dapat mengembangkan aplikasi yang efisien dan aman, serta melakukan pengujian dan debugging secara efektif.

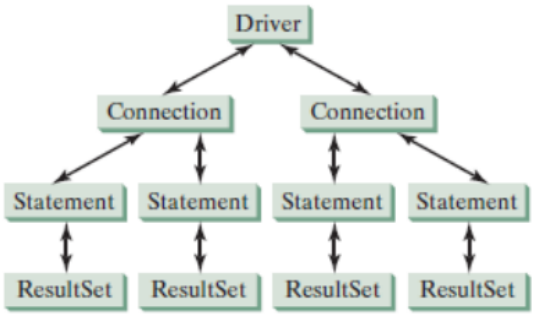
10.1 JDBC(Java Database Connectivity)



Gambar 10.1 Program Java yang mengakses dan memanipulasi database menggunakan JDBC

JDBC (Java Database Connectivity) merupakan API java untuk membuat aplikasi database. Dengan menggunakan API. JDBC suatu aplikasi dapat mengeksekusi query SQL, menampilkan hasil, dan menyajikan data ke dalam bentuk yang *user-friendly.*

1. **Membuat aplikasi database menggunakan JDBC**
   Kunci utama membuat aplikasi database adalah
   Connection,Statement, dan Resultset.



Gambar 10.2 Class-class JDBC membuat program dapat terkoneksi dengan database,
mengirim statement SQL dan memproses result/hasil.

2. **Koneksi ke database**
   Untuk membuat koneksi ke database menggunakan method getConnection pada class
   Connetion.

```
Connection cn=DriverManager.getConnection(url,user,password);
```

Pada method **getConnection** harus ditentukan url yang berupa nama jdbc drivier, lokasi

server,dan nama database serta user dan password. Contoh:

```
Connection cn =DriverManager.getConnection("jdbc:mysql://localhost/db
```

3. **Membuat statement**
   Statement digunakan untuk memanipulasi data dengan membuat perintah SQL, kemudian
   melakukan eksekusi. Ada
   dua interface statement yaitu, ***Statement dan PreparedStatement.***

   Interface Statement digunakan untuk eksekusi terhadap statement sql yang bersifat static, sementara ***Preparestatement*** merupakan *extends* dari *Statement,* yang digunakan untuk
   eksekusi statement sql secara dinamis
   (precompile) yaitu dengan parameter (atau tanpa parameter), penggunaan ***PrepareStatement*** lebih efisien utk eksekusi yang berulang.

   Contoh perintah sql menggunakan ***interface Statetement*** :

```
Statement st = cn.createStatement();
String sql = "insert into tbmhs(idmhs, nim, nama, kota_asal) values (3, 'A11
st.executeUpdate(sql);
pst.setString(3,"UMAR");
pst.setString(4,"SEMARANG");
pst.executeUpdate();
System.out.println("Insert berhasil ...");
```

   Contoh perintah sql menggunakan ***Preparedstatetement*** :

```
Connection cn=MySQLDB.getConnection();
String sql="insert into tbmhs(idmhs,nim,nama,kota_asal) values (?,?,?,?)";
PreparedStatement pst = cn.prepareStatement(sql);
pst.setInt(1,7);
pst.setString(2,"A11.2018.90001");
```

4. **Memproses ResultSet**
   Resultset menampung tabel yang sedang diakses, dimana posisi awal baris berada di posisi
   null. Sedangkan untuk berpindah ke baris berikutnya menggunakan method next. Contoh
   memproses resulset:

```
cn=DriverManager.getConnection("jdbc:mysql://localhost/dbjava","nina","
System.out.println("Koneksi Berhasil");
//membuat statement
Statement st = cn.createStatement();
String sql="select * from tbmhs";
ResultSet rs = st.executeQuery(sql);
rs.beforeFirst();
while (rs.next()){
   System.out.println(rs.getString(1));
}
```

▼ **Beberapa method yang penting pada ResultSet adalah :**

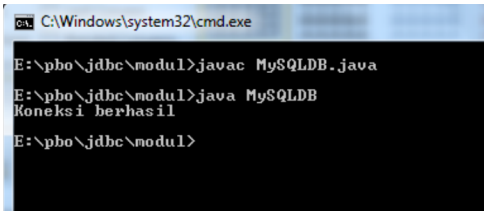
1. public boolean next(): digunakan uintuk memindahkan cursor pada baris berikutnya.
2. public boolean previous(): digunakan untuk memindahkan kursor satu baris sebelumnya
3. public boolean first():digunakan untuk memindahkna cursor ke baris pertama dari satu result set objek
4. public boolean last(): digunakan untuk memindahkna cursor ke baris terakhir dari satu result set objek
5. public boolean absolute(int row): digunakan untuk memindahkan cursor ke baris tertentu dari ResultSet object.
6. public boolean relative (int row): digunakan untuk memindahkan cursor ke baris yagn posisinya relative terhadap result set.ResultSet object, dapat berupa positive or negative.
7. public int getInt(int columnIndex): digunakan untuk mengembalikan data column
   index tertentu dari satu baris

5. **Contoh Aplikasi CRUD dengan JDBC**

Contoh Kode Program untuk Koneksi

```
MySQLDB.java
import java.sql.*;
public class MySQLDB {
 static Connection cn ;
 public static Connection getConnection(){
    try{
      cn = DriverManager.getConnection("jdbc:mysql://localhost/dbjava", "
      System.out.println("Koneksi berhasil");
    }
    catch(SQLException se){
       se.printStackTrace();
    }
       return cn;
    }
   public static void main(String[] args){
         MySQLDB.getConnection();
    }
}
```

Output :



Contoh program dengan statement insert,update,delete

```
MySQLDB.java
import java.sql.*;
public class MySQLDB {
```

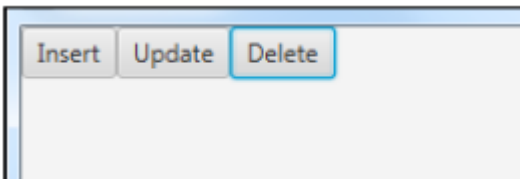
```
    static Connection cn ;
    public static Connection getConnection(){
        if (cn==null){
            try{
                cn = DriverManager.getConnection("jdbc:mysql://localhost/dbjava",
                System.out.println("Koneksi berhasil");
            } catch(SQLException se){
                  se.printStackTrace(); }
            }
            return cn;
        }
}
```

TestDB.java

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.control.Button;
import java.sql.*;
public class TestDB extends Application {
    @Override
    public void start(Stage s){
     HBox hb= new HBox();
     Button btInsert = new Button("Insert");
     Button btUpdate = new Button("Update");
     Button btDel = new Button("Delete");
     btInsert.setOnAction(e->{
        try{
            Connection cn=MySQLDB.getConnection();
            Statement st = cn.createStatement();
            String sql="insert into tbmhs(idmhs, nim, nama, kota_asal) values ('6','
            st.executeUpdate(sql);
            System.out.println("Insert berhasil ...");
        }catch(SQLException se){
            se.printStackTrace();
        }
     });
```

```
    btUpdate.setOnAction(e->{
     try{
         Connection cn=MySQLDB.getConnection();
         Statement st = cn.createStatement();
         String sql="Update tbmhs set nama='HERU' where idmhs=1";
         st.executeUpdate(sql);
         System.out.println("UPDATE berhasil ...");
     }catch(SQLException se){
           se.printStackTrace(); }
     });
    btDel.setOnAction(e->{
       try{
          Connection cn=MySQLDB.getConnection();
          Statement st = cn.createStatement();
          String sql="delete from tbmhs where idmhs=7";
          st.executeUpdate(sql);
          System.out.println("Delete berhasil ...");
       }catch(SQLException se){
          se.printStackTrace();
       }
    });
    hb.getChildren().addAll(btInsert,btUpdate,btDel);
    Scene sc = new Scene(hb,300,200);
    s.setTitle("Test Database");
    s.setScene(sc);
    s.show();
     }
}
```

Output :



Contoh program dengan PreparedStatement insert,update,delete

TestDB2.java

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.control.Button;
import java.sql.*;
public class TestDB2 extends Application {
 @Override
 public void start(Stage s){
        HBox hb= new HBox();
        Button btInsert = new Button("Insert");
        Button btUpdate = new Button("Update");
        Button btDel = new Button("Delete");
        btInsert.setOnAction(e->{
         try{
            Connection cn=MySQLDB.getConnection();
            String sql="insert into tbmhs(idmhs,nim,nama,kota_asal) values (?,?,?,?
            PreparedStatement pst = cn.prepareStatement(sql);
            pst.setInt(1,7);
            pst.setString(2,"A11.2018.90001");
            pst.setString(3,"UMAR");
            pst.setString(4,"SEMARANG");
            pst.executeUpdate()
            System.out.println("Insert berhasil ...");
         }catch(SQLException se){
            se.printStackTrace();
         }
        });
        btUpdate.setOnAction(e->{
         try{
            Connection cn=MySQLDB.getConnection();
            String sql="Update tbmhs set nama='HERU' where idmhs=?";
```

```
            PreparedStatement pst = cn.prepareStatement(sql);
            pst.setInt(1,1);
            pst.executeUpdate();
            System.out.println("Update berhasil ...");
         }catch(SQLException se){
            se.printStackTrace();
           }
        });
        btDel.setOnAction(e->{
         try{
            Connection cn=MySQLDB.getConnection();
            String sql="delete from tbmhs where idmhs=?";
            PreparedStatement pst = cn.prepareStatement(sql);
            pst.setInt(1,7);
            pst.executeUpdate();
            System.out.println("Delete berhasil ...");
         }catch(SQLException se){
               se.printStackTrace();
         }
        });
        hb.getChildren().addAll(btInsert,btUpdate,btDel);
        Scene sc = new Scene(hb,300,200);
        s.setTitle("Test Database");
        s.setScene(sc);
        s.show();
        }
    }
```

Memproses ResultSet dengan menampilkan di tabelview

```
MHS.java
public class MHS {
 private int idmhs;
 private String nim;
 private String nama;
 private String kota_asal;
 @Override
```

```
 public String toString() {
    return "MHSM{" + "idmhs=" + idmhs + ", nim=" + nim + ", nama=" + nama
 }
 public MHS(){
 }
 public int getIdmhs() {
    return idmhs;
 }
 public void setIdmhs(int idmhs) {
    this.idmhs = idmhs;
 }
 public String getNim() {
    return nim;
 }
 public void setNim(String nim) {
    this.nim = nim;
 }
 public String getNama() {
    return nama;
 }
 public void setNama(String nama) {
    this.nama = nama;
 }
 public String getKota_asal() {
    return kota_asal;
 }
 public void setKota_asal(String kota_asal) {
    this.kota_asal = kota_asal;
 }
}
```

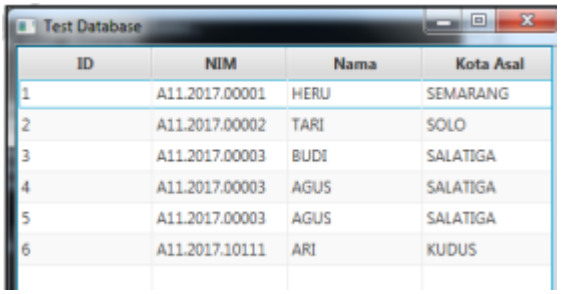
TestDB3.java

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.control.TableView;
import java.sql.*;
```

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Pos;
import javafx.scene.control.TableColumn;
import javafx.scene.control.cell.PropertyValueFact
public class TestDB3 extends Application {
    @Override
    public void start(Stage s){
        Scene sc = new Scene(getHBTabel(),600,400);
        s.setTitle("Test Database");
        s.setScene(sc);
        s.show();
     }
    public HBox getHBTabel(){
     HBox hb= new HBox();
     TableView<MHS> tbMHS = new TableView();
     hb.getChildren().add(tbMHS);
     TableColumn<MHS,Integer> colIdmhs= new TableColumn("ID");
     TableColumn<MHS,String> colNim=new TableColumn("NIM");
     TableColumn<MHS,String> colNama=new TableColumn("Nama");
     TableColumn<MHS,String> colKota_asal=new TableColumn("Kota Asal");
     colIdmhs.setCellValueFactory(new PropertyValueFactory<>("idmhs"));
     colNim.setCellValueFactory(new PropertyValueFactory<>("nim"));
     colNama.setCellValueFactory(new PropertyValueFactory<>("nama"));
     colKota_asal.setCellValueFactory(new PropertyValueFactory<>("kota_asal"
     colIdmhs.prefWidthProperty().bind(tbMHS.widthProperty().divide(4));
     colNim.prefWidthProperty().bind(tbMHS.widthProperty().divide(4));
     colNama.prefWidthProperty().bind(tbMHS.widthProperty().divide(4));
     colKota_asal.prefWidthProperty().bind(tbMHS.widthProperty().divide(4));
     tbMHS.getColumns().addAll(colIdmhs,colNim,colNama,colKota_asal);
     tbMHS.setItems(getAllMhs());
         return hb;
    }
    public ObservableList<MHS> getAllMhs(){
        ObservableList<MHS> lmhs= FXCollections.observableArrayList();
        try {
            Statement st=MySQLDB.getConnection().createStatement();
            String sql="select * from tbmhs";
```

```
            ResultSet rs=st.executeQuery(sql);
            rs.beforeFirst();
                while(rs.next()){
                    MHS m = new MHS();
                    m.setIdmhs(Integer.parseInt(rs.getString(1)));
                    m.setNim(rs.getString(2));
                    m.setNama(rs.getString(3));
                    m.setKota_asal(rs.getString(4));
                    lmhs.add(m);
                    System.out.println(m);
                }
        } catch (SQLException ex) {
                ex.printStackTrace();
        }
        return lmhs;
    }
}
```

Output :



📚REFERENSI

Liang, D.J., 2015, Introduction to JAVA Programming Comprehensive Version Tenth
Edition, Pearson.

Pratama, M.R., 2007, Implementasi MVC Dengan DAO Pada Java Desktop Application, url: