

## **PERTEMUAN 5 : PENGUNAAN DATABASE PADA APLIKASI**

### **A. TUJUAN PERKULIAHAN:**

Pada pertemuan ini akan dijelaskan mengenai koneksi database dan penggunaannya pada program. Setelah mempelajari materi perkuliahan ini, mahasiswa mampu:

- 5.1 Membuat file database
- 5.2 Memahami cara koneksi database
- 5.3 Menggunakan database dalam program

### **B. URAIAN MATERI:**

<b>Tujuan Pembelajaran 5.1:</b>
Pembuatan File Database

Database merupakan kumpulan data yang disusun sedemikian rupa sehingga menghasilkan informasi yang berguna.

Aplikasi database yang digunakan adalah MySQL, dan untuk pembuatan file database dapat dilakukan dengan beberapa cara, a.l:

- ✓ Menggunakan MySQL console

- a. Membuat database**

- ```
create database nmfileDB
```

- contoh:

- ```
create database MHS
```

- b. Mengaktifkan database**

- ```
use nmfileDB
```

- contoh:

- ```
use MHS
```

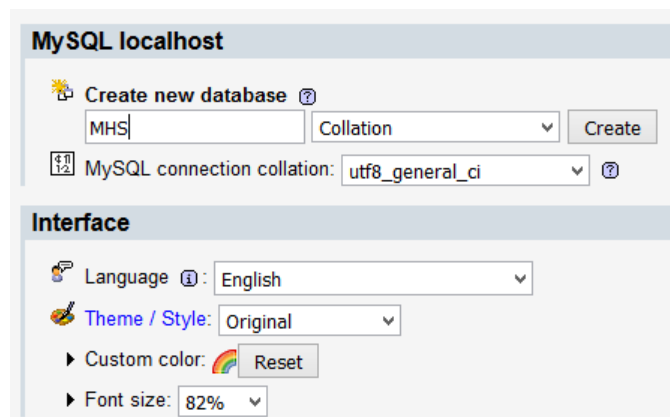
### c. Membuat tabel

```
create table nmtable(  
  nmfield1 tippedata(jml_char) null/not null,  
  nmfield2 tippedata(jml_char),  
  ..... ,  
  .....  
)
```

Contoh:

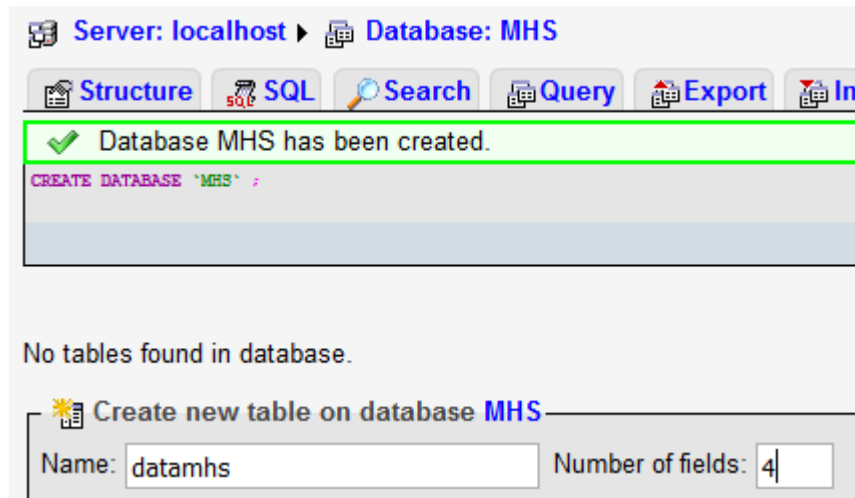
```
create table datamhs(  
  nim varchar(15) not null,  
  nama varchar(30),  
  semester int,  
  kelas varchar(1),  
  primary key ("nim")  
)
```

- ✓ Menggunakan PHPMyAdmin
  - Aktifkan PHPMyAdmin
  - Buat file database baru

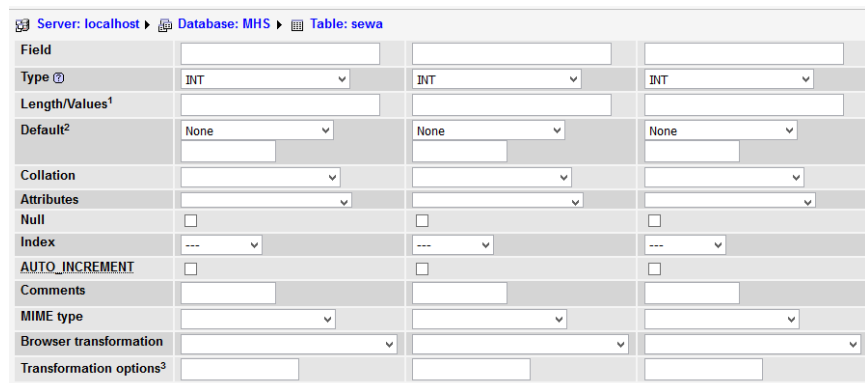


Gb. 5.1 File Database

- Buat table



- Buat struktur table sesuai keperluan



Gb. 5.2 Tabel Database

### Tujuan Pembelajaran 5.2:

Melakukan koneksi database

Koneksi ke database pada Java ditangani oleh JDBC (Java Database Connectivity). JDBC merupakan salah satu API (Application Programming Interface) Java yang khusus menangani koneksi database. Antarmuka (interface) ini memungkinkan untuk mengakses database, seperti Oracle, Access, MySQL

Cara koneksi

Ada beberapa langkah yang harus dilakukan saat akan melakukan koneksi database.

**Langkah-langkah melakukan koneksi:**

- a. Me-load/memanggil JDBC Driver
- b. Membuat koneksi ke database
- c. Membuat dan Mengirimkan perintah ke SQL (menggunakan package java.sql)
- d. Melakukan eksekusi dari perintah SQL

**a. Memanggil Driver JDBC**

Langkah pertama untuk melakukan koneksi dengan database server adalah dengan memanggil JDBC Driver dari database server yang kita gunakan. Driver adalah library yang digunakan untuk berkomunikasi dengan database server. Driver dari setiap database server berbeda-beda, sehingga kita harus menyesuaikan Driver JDBC sesuai dengan database server yang akan digunakan. Sebelum memanggil driver JDBC, cek terlebih dahulu pada library apakah sudah terinstal/blm. Jika belum maka kita harus menginstal driver tersebut.

Untuk menginstall connector JDBC dapat dilakukan dengan beberapa cara, antara lain\_:

- ✓ menyalin file jar ke folder ext dari java, misalnya berada di C:\Program Files\Java\jre7\lib\ext.
- ✓ Meng-extract file jar kedalam suatu folder kemudian mensetting CLASSPATH ke folder tersebut.

File jar untuk MySQL : mysql-connector-java-5.1.14-bin.jar

File jar untuk SQL Server : sqljdbc4.jar

Berikut ini adalah penulisan perintah untuk memanggil driver JDBC.

**Class.forName** (namaDriver);

atau

**Class.forName** (namaDriver).newInstance();

Contoh:

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

**b. Membuat koneksi**

Setelah melakukan pemanggilan terhadap driver JDBC, langkah selanjutnya adalah membangun koneksi dengan menggunakan interface *Connection*. *Object Connection* yang dibuat untuk membangun koneksi dengan database server tidak dengan cara membuat object baru dari interface *Connection* melainkan dari class *DriverManager* dengan menggunakan metode **getConnection()**.

*Connection namaVar* = **DriverManager.getConnection(<argumen>)**

**<argumen>** dapat dituliskan:

- jdbc:<DBServer>://[Host][:Port]/<namafileDBase>?<user=User>&<password=Password>

contoh:

```
"jdbc:mysql://localhost:3306/DbTokoABC?user=root&password=root"
```

Atau

```
"jdbc:mysql://localhost/DbTokoABC?user=root&password=root"
```

- getConnection(*String url, Properties info*)

contoh:

```
String url =
"jdbc:mysql://localhost:3306/DbTokoABC";
Properties prop = new java.util.Properties();
//tidak mengimpor kelas
prop.put("user", "root");
prop.put("password", "root");
Connection cn = DriverManager.getConnection(url,
prop);
```

- getConnection(*String url, String user, String password*)

contoh:

```
String url =  
"jdbc:mysql://localhost:3306/DbTokoABC";  
String user = "root"  
String password = "root"  
Connection cn = DriverManager.getConnection(url,  
user, password);
```

Untuk menangani error yang mungkin terjadi pada proses melakukan koneksi dengan database maka ditambahkan **try-catch** yang digunakan untuk memerangkap kesalahan program. Exception yang akan dihasilkan pada proses ini adalah berupa SQLException.

Adapun cara penulisan adalah sebagai berikut :

```
try {  
    //... koneksi database  
} catch (Exception ex){  
    //... penanganan error koneksi  
}
```

- c. Membuat dan Mengirimkan perintah ke SQL (menggunakan package java.sql)

JDBC API menyediakan interface yang berfungsi untuk melakukan proses pengiriman statement SQL yang terdapat pada package java.sql. Di dalam JDBC API disediakan tiga buah interface untuk fungsi tersebut yaitu :

✓ **Statement**

Interface ini dibuat oleh method **Connection.createStatement()**.

Object Statement digunakan untuk pengiriman statement SQL tanpa parameter.

Contoh:

```
Statement sta = Connection.createStatement();
```

✓ **PreparedStatement**

Interface ini dibuat oleh method **Connection.prepareStatement()**.

Object PreparedStatement digunakan untuk pengiriman statement SQL dengan atau tanpa parameter. Dengan object ini, kita dapat menampung satu atau lebih parameter sebagai argumen input (parameter IN). Interface ini memiliki performa lebih baik dibandingkan dengan interface Statement karena dapat menjalankan beberapa proses dalam sekali pengiriman perintah SQL.

Contoh:

```
PreparedStatement preSta=  
Connection.prepareStatement();
```

d. Melakukan eksekusi dari perintah SQL

Setelah kita memiliki object statement, kita dapat menggunakannya untuk melakukan pengiriman perintah SQL dan mengeksekusinya. Metode eksekusi yang digunakan untuk perintah SQL terbagi menjadi dua bagian yaitu untuk perintah SELECT metode eksekusi yang digunakan adalah executeQuery() dengan nilai kembalinya adalah ResultSet, dan untuk perintah INSERT, UPDATE, DELETE metode eksekusi yang digunakan adalah executeUpdate().

<b>Tujuan Pembelajaran 5.3:</b>
Menggunakan database pada program

Database dapat digunakan pada program dengan menggunakan class, method, dan objek yang sudah dipelajari.

**Langkah-langkah untuk menyimpan adalah:**

- Membuat database (sudah dibahas)
- Membuat koneksi ke database (sudah dibahas)
- Mengidentifikasi Perintah ( SQL ) menyimpan data
- Membuat dan Mengirimkan perintah ke SQL untuk menyimpan
- Melakukan eksekusi dari perintah SQL

**Penjelasan:**

**a. Mengidentifikasi SQL yang digunakan untuk menambah/menyimpan data**

Sql yang digunakan untuk menambah/menyimpan data adalah:

**INSERT INTO** *namatabel* **VALUE** (“data1”, “data2”, “data3”,...)

**Contoh:**

INSERT INTO datamhs **VALUE** (“1234”, “Sri”, “Sem 3”)

**b. Mengidentifikasi class yang digunakan untuk menyimpan data**

**Class** yang digunakan:

**PreparedStatement:** kelas untuk mengirimkan perintah SQL ke database  
menggunakan parameter.

**Metode** yang digunakan:

**prepareStatement(“SQL”);**

Penulisan:

**PreparedStatement** *namaVar*=namaVarkoneksi.metode(“SQL”);

Cara I:

**PreparedStatement** *nmVar*=nmVarkoneksi. **prepareStatement**( “insert into nmtabel (field1, field2, field3)”+” value (?,?,?)”);

*nmVar.setString*(no.urut, data); → huruf tebal sesuai dg tipe data dalam file database (1 data yang disimpan diwakili 1 tanda ?)

Contoh:

**PreparedStatement** *pStat*=koneksi. **prepareStatement**( “insert into datamsh (nim, nama, sem)”+” value (?,?,?)”);

*pStat.setString*(1, nim.getText());

*pStat. setString*(2, nama.getText());

*pStat. setString*(3, semester.getText());



Cara II:

```
PreparedStatement namaVar = nmVarkoneksi.prepareStatement
("INSERT INTO nmtabel (nmfield1, nmfield2, ...) " + "VALUES (' " +
data1 + " ',' " + data2 + " ',' " + ..... + " '");
```

Contoh:

```
PreparedStatement pStat = koneksi.prepareStatement ("INSERT
INTO datamhs (nim, nama, sem)" + "VALUES (' " + nim.getText() + " ',' "
+ nama.getText() + " ',' " + semester.getText() + " '");
```

- c. Melakukan eksekusi penyimpanan

Perhatikan perintah untuk eksekusi ini, tanpa menggunakan perintah ini data tidak akan tersimpan ke table pada database

Metode yang digunakan: **executeUpdate()**

Penulisan:

```
nmVarstatement.executeUpdate();
```

contoh:

```
pStat. executeUpdate();
```

## Tampil data

Data pada table database dapat ditampilkan pada komponen-komponen melalui sebuah program dengan membaca dan mengambil data dari table database tersebut.

**beberapa Langkah untuk menampilkan data adalah:**

- Melakukan koneksi ke database (sudah dibahas)
- Mengidentifikasi Perintah ( SQL ) untuk membaca data
- Membuat program untuk menampilkan data

### Penjelasan:

**SQL** untuk membaca table database:

- `SELECT * FROM namatabel`
- `SELECT namafield1, namafield2, ...FROM namatabel`
- `SELECT * FROM namatabel WHERE namafield='data' order by namafield ascending/descending`

### Cara menampilkan data ke jTable

Koneksi database

```
try{
    Statement st = (Statement) conn.createStatement();
    ResultSet rs = st.executeQuery("SELECT * FROM
                                   namaTabel");

    while (rs.next()) {
        Object[] nmvardata=new Object[jml.kolom];
        nmvardata [0]=rs.getString("nmfield1");
        nmvardata [1]=rs.getString("nmfield2");
        nmvardata [2]=rs.getString(".....");
        nmvardata [3]=rs.getString(".....");
        nmvardata [4]=rs.getString(".....");

        nmVarmodel.addRow(nmvardata);
    }
    rs.close();
    conn.close();
}

catch(SQLException e){
    System.out.println("Data gagal disimpan" + e.getMessage());
}

catch(ClassNotFoundException e){
    System.out.println("driver tidak ditemukan" );
}
```

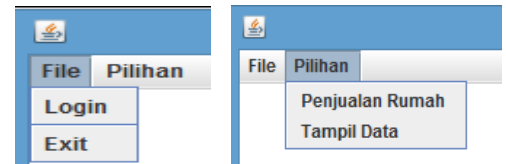
Nama field tabel

Menambah baris pada objek jTable


### C. LATIHAN SOAL

Buat Form Utama yang berisi form-form menu:

**Pilihan : Penjualan Rumah, Tampil data**



#### Menu Penjualan Rumah

Nama Pemesan	<input type="text"/>	Harga Bangunan	<input type="text" value="90000000"/>
Harga Tanah/m2	<input type="text" value="500000"/>	Tipe Rumah	Luas Tanah Asli
Area	Bougenville	<input checked="" type="radio"/> Tipe-36	<input type="text" value="90"/>
		<input type="radio"/> Tipe-45	Luas Tanah Tersedia
		<input type="radio"/> Tipe-54	<input type="text"/>
	Harga	<input type="text"/>	
	DP	<input type="text"/>	Bulan
	Lama Cicilan	<input type="text"/>	<input type="checkbox"/> Setuju
PPN (10%)	<input type="text"/>		
Cicilan/bln	<input type="text"/>		
PROSES		SIMPAN	CLEAR

Klik tombol simpan, maka data akan tersimpan di table database

#### Menu Tampil Data

Tampil (menampilkan data yang sudah disimpan di tabel database)

Nama Peme...	Area	Tipe Rumah	Luas Tanah	Harga	Lama Cicilan	Cicilan/bln

JML. RUMAH TERJUAL

TOTAL PENJUALAN RU...

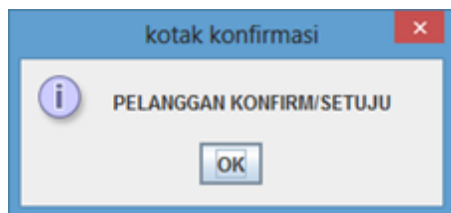
Ketentuan:

AREA RUMAH	Hrg. Tanah /m <sup>2</sup>
Bougenvile	500000
Melati	600000
Flamboyen	700000

Area RUMAH	HARGA BANGUNAN
Bougenvile	90,000,000
Melati	120,000,000
Flamboyen	150,000,000

TIPE RUMAH	LUAS TANAH ASLI
T-36	90
T-45	120
T-54	140

- Harga= LUAS TANAH TERSEDIA \* Hrg. TANAH/M<sup>2</sup> + HARGA BANGUNAN
- PPn =10% X HARGA
- CICILAN/BLN = (HARGA + PPn - DP)/ JUMLAH CICILAN
- Jika klik cek akan muncul kotak dialog konfirmasi



#### D. DAFTAR PUSTAKA

Budiharto, W. (2004). *Pemrograman Web Menggunakan J2EE*. Jakarta: Elexmedia Komputindo.

JENI, T. P. (2007). JENI 1-6.

Wijono, S. H., Suharto, B. H., & Wijono, M. S. (2006). *Pemrograman Java Servlet dan JSP dengan Netbeans*. Yogyakarta: C.V ANDI OFFSET.