Bab
7

Akses Database Menggunakan JDBC

Dalam bab ini kita akan membicarakan tentang akses database menggunakan JDBC, cara penggunannya mulai dari membuka koneksi sampai melakukan operasi baca, tulis, dan hapus data dari database. Bahkan menggunakan interface dari yang sederhana sampai yang kompleks.

Design sederhana bertujuan untuk memudahkan pemahaman, sedangkan design interface yang kompleks agar bisa menampilkan data dalam bentuk yang lebih menarik dan bisa seperti pada aplikasi yang profesional.

Pemahaman JDBC ini adalah mutlak diperlukan bagi seorang programmer database, jika ingin bekerja dengan Java. Sebagai gambaran saja, jika programmer membuat aplikasi database dengan semisal visual basic/PHP dengan database SQL Server atau MySQL, apakah program tersebut bisa diganti databasenya dengan yang lain semisal Postgree atau Oracle. Dalam hal ini JDBC mampu membuat aplikasi bisa berjalan di atas semua sistem database, hanya cukup dengan memasang driver JDBC dari database tersebut.

7.1 Mengenal JDBC

JDBC adalah sebuah nama dari sebuah produk yang ada dalam Java, atau yang dikenal sebagian orang sebagai **Java Database Connectivity**, seperti dokumentasi yang dikeluarkan oleh **Sun Microsystem** sendiri dalam situs resminya. Sebenarnya lengkapnya adalah **JDBC API** yaitu *Application Programming Interface*, yang menyediakan akses data secara universal dan independent. Bekerja dengan JDBC secara umum adalah sebagai berikut:

- Me-load JDBC Driver ke dalam JVM.
- Membuat koneksi ke database yang direpresentasikan sebagai objek **java.sql.Connection**.
- Membuat objek **java.sql.Statement** yang akan digunakan untuk mengirimkan perintah SQL ke database.
- Menjalankan metode yang bersesuaian dari objek **java.sql.Statement**, seperti **executeQuery()** untuk membaca atau **executeUpdate**() untuk menulis ke tabel.

7.2 Kelas untuk Mengakses Database

Java menyiapkan paket khusus untuk menyimpan kelas-kelas yang terlibat dalam operasi akses database, yaitu dalam paket **java.sql**. Seperti yang Anda ketahui sebelumnya bahwa perlu ada beberapa tahapan agar bisa melakukan operasi ke dalam database, antara lain pengenalan dirver, pembuatan koneksi/hubungan, statement/ perintah SQL baru, kemudian operasi baca dengan excecuteQuery ataupun operasi tulis dengan executeUpdate(). Setiap tahap dan pekerjaan yang berbeda-beda tersebut dilakukan oleh kelas yang berbeda, mulai dari DriverManager, Connection, Statement, PreparedStatement, Resulset.

7.2.1 DriverManager

DriverManager adalah kelas untuk manajemen driver, seperti yang Anda ketahui bahwa untuk bisa mengakses database diperlukan sebuah driver JDBC. Untuk itu file driver tersebut harus ada, dan

terpasang dalam kelas path dalam program kerjanya. Untuk memastikan sebuah driver sudah terpasang atau belum, bisa digunakan objek **Class** untuk mengidentifikasinya, seperti berikut.

```
try {
    Class.forName("com.mysql.jdbc.Driver");
    System.out.println("jdbc.Driver : OK");
}
    catch(ClassNotFoundException e) {
    System.out.println("Kelas: jdbc.Driver tidak ada");
}
```

Jika program tersebut dijalankan, akan menghasilkan pesan "`jdbc.Driver : OK`", bila dalam project Anda sudah terpasang *classpath* untuk driver MySQL. Akan tetapi, program akan menampilkan pesan "`Kelas: jdbc.Driver tidak ada`" bila direktori kerja Anda tidak mengenali path dari Driver tersebut, walaupun dalam komputer Anda sudah ada filenya, ataupun dalam editor sudah terpasang dalam lingkungan library.

Dalam aplikasinya, Anda tidak mesti memberikan operasi pengecekan driver ini, karena hanya untuk melakukan *trace* jika terjadi kesalahan, agar diketahui penyebabnya saja.

Objek **DriverManager** ini bisa menghasilkan sebuah koneksi melalui metode **getConnction()**, dengan menyertakan URL sebagai parameternya. Adapun penulisannya menggunakan titik dua (:), seperti contoh:

```
jdbc:mysql://localhost/db_akademik?user=root&password=admin
```

Berikut ini penjelasannya.

- **jdbc:mysql** adalah nama drivernya.
- **localhost** adalah nama hostname (komputer address).
- **db_akademik** adalah nama databasenya.
- **root** adalah nama account user MySQL.
- **admin** adalah password dari account MySQL tersebut.

7.2.2 Connection

Kelas **Connection** adalah kelas yang menyimpan sebuah koneksi atau hubungan yang telah dilakukan oleh **DriverManager**. Untuk selanjutnya kelas ini sendiri bisa dipakai untuk menentukan jenis operasi baca atau operasi tulis pada tabel. Seperti contoh potongan program berikut.

```
.....
try {
  Connection conn=DriverManager.getConnection(
    "jdbc:mysql://localhost/db_akademik?" +
    "user=root&password=admin");
}
```

Instance **conn** tersebut menyimpan sebuah koneksi ke database "db_akademik", dengan komputer localhost (komputer lokal).

7.2.3 Statement

Kelas ini digunakan untuk mengirimkan statement SQL ke database, tanpa penggunaan parameter. Seperti contoh berikut:

```
try {
  Connection conn=DriverManager.getConnection(
    "jdbc:mysql://localhost/db_akademik?" +
    "user=root&password=admin");
  Statement st = conn.createStatement();
  ResultSet rs = st.executeQuery("Select * from user");
}
```

Instance **st** tersebut membuat sebuah statement baca, yaitu "`Select * from user`", melalui metode `executeQuery` yang dimilikinya. Selain itu, Statement tersebut juga bisa melakukan operasi baca dengan metode `executeUpdate()` yang juga dimiliki oleh kelas ini.

7.2.4 PreparedStatement

Kelas ini digunakan untuk mengirimkan statement SQL ke database, yang disertai dengan penggunaan parameter, seperti contoh:

```
PreparedStatement pStatement = null;
try {
  Connection conn=DriverManager.getConnection(
```

```
    "jdbc:mysql://localhost/db_akademik?" +
    "user=root&password=admin");
  pStatement = conn.prepareStatement(
    "insert into user (user_id, password, jabatan)"+
    "Values (?,?,?)");
  pStatement.setString(1, "huda");
  pStatement.setString(2, "123");
  pStatement.setString(3, "Kepala Unit SDM");
  pStatement.executeUpdate();
}
```

Instance **pStatement** tersebut membuat sebuah statement tulis, yaitu "`insert into user (user_id, password, jabatan)`", dan dijalankan melalui metode `executeUpdate()` yang dimiliki oleh kelas ini.

Perhatikan metode `setString([nomor kolom], [isi dari parameter])` nomor kolom dimulai dari 1, sehingga contoh di atas dapat disimpulkan user_id="huda", password="123", jabatan="Kepala Unit SDM".

7.2.5 ResultSet

Kelas yang menyimpan dataset (sekumpulan data) dari hasil statement query "SELECT". Seperti contoh berikut ini.

```
try {
  Connection conn=DriverManager.getConnection(
    "jdbc:mysql://localhost/db_akademik?" +
    "user=root&password=admin");
  Statement st = conn.createStatement();
  ResultSet rs = st.executeQuery("Select * from user");
  while (rs.next()) {
     System.out.println("User ID    :"+
      rs.getString("user_id"));
  }
}
```

Instance **rs**, akan menyimpan hasil query yang bisa diakses dengan cara memanggil metode getter untuk setiap jenis datanya. Misalnya getString() adalah untuk mendapat nilai string, diikuti nama fieldnya. Sedangkan untuk membaca record berikutnya bisa dilakukan dengan metode **next()**.

7.3 Latihan Akses Database Sederhana

Tibalah saatnya untuk membuat contoh-contoh aplikasi dari serangkaian teori yang sudah dibicarakan sebelumnya, agar pembaca mendapatkan gambaran lebih jelas dan kongkrit. Perihal teori adalah hal yang gampang, sedangkan tahap implementasi adalah tahapan yang dianggap sulit. Sehingga wajar kalau sering kali kita dengar **praktik tidak semudah teori**. Meskipun sebenarnya malah praktiknya lebih mudah daripada teorinya, karena teori bersifat abstraksi (gambaran imaginatif dalam otak/pikiran), sedangkan praktik menyangkut sistem motorik dalam syaraf yang terakumulasi karena sebuah tindakan yang dilakukan secara kontinyu/kebiasaan. Tentunya melakukan kebiasaan akan lebih ringan/mudah dibandingkan proses berpikir. Untuk itulah dalam buku ini dilakukan latihan-latihan untuk membiasakan dalam memprogram.

7.3.1 Membuat Koneksi

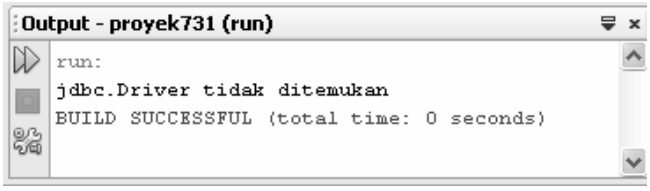
Dalam latihan ini Anda akan belajar bagaimana memasang sebuah koneksi, dan memastikan bahwa koneksi Anda sudah terpasang dengan baik atau tidak. Anda akan dilatih untuk belajar dari sebuah kesalahan (*exeption*), karena banyak orang yang berhasil setelah belajar dari kesalahan. Berikut ini langkah-langkah yang harus diikuti:

1. Buat proyek baru (Java Application), beri nama **Proyek731**, cukup dengan kelas Main saja.
2. Edit kelas **Main.java**, sehingga menjadi seperti berikut.

```
package proyek731;

import java.sql.*;

/**
 *
 * @author Administrator
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
```

```
        // TODO code application logic here

        String user="root";
        String pwd ="admin";
        String host="localhost";
        String db="latihan_bab6";
        String urlValue="";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            urlValue="jdbc:mysql://"+ host +"/"+
                db +"?user=" + user +
                "&password="+ pwd;

            Connection conn=
              DriverManager.getConnection(urlValue);

            System.out.println("koneksi sukses");
            conn.close();
       }

       catch (SQLException e){
         System.out.println("koneksi gagal " +
         e.toString());
       }

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

3. Lakukan kompilasi (Clean and Build), kemudian jalankan program. Dan lihat apa yang muncul dalam jendela output?



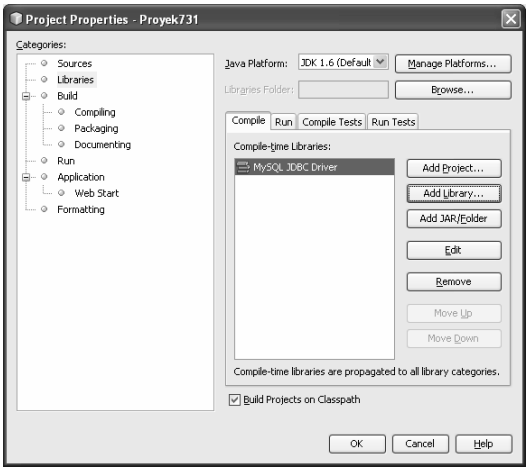
Gambar 7.1 Output Proyek731 Awal

Terlihat bahwa program di-compile sukses tanpa error, tetapi program menunjukkan kegagalan/*runt time error*. Tentunya hal ini akan membuat pusing bagi pemula, karena sudah merasa

program sudah free error, tetapi kenapa saat dijalankan program memberikan hasil yang error? Inilah persoalan mendasar yang sering membuat programmer baru berputus asa dan mengurungkan niatnya untuk melanjutkan mempelajari program.

Baiklah, mari dikupas persoalan tersebut dengan saksama, diawali dengan pesan error yang muncul, yaitu "*jdbc.Driver tidak ditemukan*". Hal ini terlihat di dalam program bahwa perintah untuk menampilkan pesan ini ada pada langkah eksepsi kedua (**ClassNotFoundException e**). Dari sini dapat disimpulkan 100% bahwa ini terjadi akibat kelas tersebut tidak ada, kelas apakah itu? Kita bisa melihat bahwa dalam program kita menuliskan **Class.forname** ("com.mysql.jdbc.Driver"). Berarti kelas driver MySQL-lah yang bermasalah. Untuk itu, lakukan pemasangan driver, lanjutkan ke langkah berikutnya.

4. Untuk memasang driver MySQL tersebut, klik kanan project → **Properties**, sehingga muncul jendela **Project Properties – Proyek731**.
5. Dalam jendela **Project Properties – Proyek731**, pilih **Libraries** pada kotak **Categories**. Selanjutnya di sebelah kanan tekan tombol **Add Library...** (pada tab Compile), sehingga muncul jendela **Add Library**. Pilih **MySQL JDBC Driver**, tekan **Add Library**, kemudian tekan **OK** pada jendela **Properties**.



Gambar 7.2 Project Properties Proyek731

6. Kemudian compile ulang dan jalankan program, lihat hasilnya. Anda akan mendapati pesan "*koneksi sukses*". Jika Anda masih belum berhasil, periksa langkah Anda.

7.4 Membuka Isi Tabel

Pada tahap sebelumnya Anda tentu ingat, telah membuat tabel **user** di dalam database MySQL (Sub-subbab 6.4.2). Sekarang tabel tersebut akan kita baca dengan program untuk menguji program koneksi kita sebelumnya. Untuk itu ikuti langkah berikut ini:

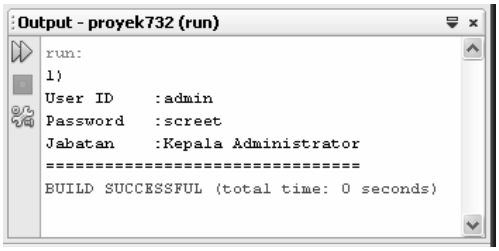
1. Buat proyek baru (Java Application), beri nama **Proyek732**, cukup dengan kelas Main saja.
2. Edit kelas **Main.java**, sehingga menjadi seperti berikut:

```
package proyek732;

import java.sql.*;

/**
 *
 * @author Administrator
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        String user="root";
        String pwd ="admin";
        String host="localhost";
        String db="latihan_bab6";
        String urlValue="";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            urlValue="jdbc:mysql://"+ host +"/"+
                db +"?user=" + user +
                "&password="+ pwd;

            Connection conn=
              DriverManager.getConnection(urlValue);
```

```
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(
                   "Select * from user");

            while (rs.next()) {
                  no=no+1;
                System.out.println(no + ")");
                System.out.println("User ID    :" +
                   rs.getString("user_id"));
                System.out.println("Password   :" +
                   rs.getString("password"));
                System.out.println("Jabatan    :" +
                   rs.getString(3));
               System.out.println("============" +
                   "====================");
             }

              st.close();
            conn.close();
        }

        catch (SQLException e){
          System.out.println("koneksi gagal " +
          e.toString());
        }

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

3. Jangan lupa untuk memasang driver JDBC MySQL-nya, perhatikan contoh sebelumnya **Proyek731**.

4. Lakukan kompilasi (Clean and Build), kemudian jalankan program. Dan lihat apa yang muncul dalam jendela output?



Gambar 7.3 Output Proyek732

7.4.1 Memasukkan Data ke Tabel

Berikut ini teknik untuk melakukan operasi tulis (*write*) ke database. Dalam hal ini program akan menambahkan record baru ke tabel **user**. Untuk itu lakukan beberapa langkah berikut:

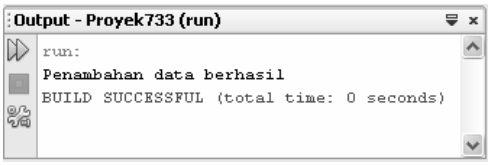
1. Buat proyek baru (Java Application), beri nama **Proyek733**, cukup dengan kelas Main saja.
2. Edit kelas **Main.java**, sehingga menjadi seperti berikut.

```
package proyek733;

import java.sql.*;

/**
 *
 * @author Administrator
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        String user="root";
        String pwd ="admin";
        String host="localhost";
        String db="latihan_bab6";
        String urlValue="";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            urlValue="jdbc:mysql://"+ host +"/"+
                db +"?user=" + user +
                "&password="+ pwd;

            Connection conn=
              DriverManager.getConnection(urlValue);

            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(
                   "Select * from user");

            while (rs.next()) {
                  no=no+1;
                System.out.println(no + ")");
```

```
                System.out.println("User ID    :" +
                   rs.getString("user_id"));
                System.out.println("Password   :" +
                   rs.getString("password"));
                System.out.println("Jabatan    :" +
                   rs.getString(3));
               System.out.println("============" +
                   "====================");
             }

             st.close();
            conn.close();
        }

        catch (SQLException e){
          System.out.println("koneksi gagal " +
          e.toString());
        }

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

3. Jangan lupa untuk memasang driver JDBC MySQL-nya, perhatikan contoh sebelumnya **Proyek731**.

4. Lakukan kompilasi (Clean and Build), kemudian jalankan program. Dan lihat apa yang muncul dalam jendela output?



Gambar 7.4 Output Proyek733

5. Untuk memastikan lagi bahwa operasi penambahan data tersebut berhasil, bukalah PhpMyAdmin Anda, kemudian lakukan **Browse** pada tabel user, pasti datanya sudah ada. Cara lain, Anda dapat menjalankan program **Proyek732**.

   Tapi ingat, jangan jalankan program **Proyek733** ini lebih dari sekali, karena akan melakukan perintah yang sama. Dalam artian menambahkan user yang sama sehingga akan ada data

ganda dalam database, yang tentunya itu akan dihalangi oleh MySQL akibat dari atribut tabel yang tidak mengizinkan, yakni field **user_id** sebgai **Primary key** (ingat dalam proses pembuatan pada bab sebelumnya). Dan Anda akan memperoleh pesan dalam jendela Output:

```
koneksi gagal
com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolatio
nException: Duplicate entry 'Roshdiana' for key 1
```

7.4.2 Melakukan Edit/Update Record

Melakukan operasi edit atau perbaikan data, hampir dilakukan di setiap aplikasi. Untuk itu materi ini sangat penting dikuasi secara benar. Berikut ini contoh membuat aplikasi perbaikan data, ikuti langkah-langkah di bawah ini:

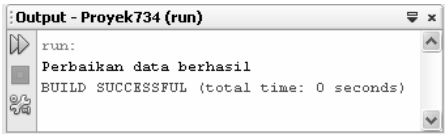
1. Buat proyek baru (Java Application), beri nama **Proyek734**, cukup dengan kelas Main saja.
2. Edit kelas **Main.java**, sehingga menjadi seperti berikut.

```
package proyek734;

import java.sql.*;

/**
 *
 * @author Administrator
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        String user="root";
        String pwd ="admin";
        String host="localhost";
        String db="latihan_bab6";
        String urlValue="";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            urlValue="jdbc:mysql://"+ host +"/"+
                db +"?user=" + user +
```

```
                "&password="+ pwd;

            Connection conn=
              DriverManager.getConnection(urlValue);

            PreparedStatement pStatement = null;
             String sql ="update user set password=?,"+
                    "jabatan=? where user_id=? ";

             pStatement = conn.prepareStatement(sql);
            pStatement.setString(1, "rahasia bgt");
            pStatement.setString(2,
                    "Kepala Unit Marketing");

            pStatement.setString(3, "Roshdiana");
             int intBaris= pStatement.executeUpdate();
            if (intBaris>0)
                System.out.println(
                  "Perbaikan data berhasil");
            else
                System.out.println(
                   "Perbaikan data gagal");
            pStatement.close();
            conn.close();
        }

        catch (SQLException e){
          System.out.println("koneksi gagal " +
          e.toString());
        }

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

3. Jangan lupa untuk memasang driver JDBC MySQL-nya, perhatikan contoh sebelumnya **Proyek731**.

4. Lakukan kompilasi (Clean and Build), kemudian jalankan program. Dan lihat apa yang muncul dalam jendela output?



Gambar 7.5 Output Proyek734

7.4.3 Melakukan Penghapusan Record

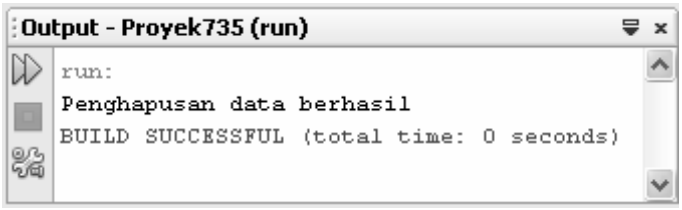
Setelah Anda belajar membuka, membaca dan memperbaiki data, kurang sempurna jika belum mempelajari penghapusan data (delete). Untuk itu lakukan beberapa langkah sederhana berikut:

1. Buat proyek baru (Java Application), beri nama **Proyek735**, cukup dengan kelas Main saja.
2. Edit kelas **Main.java**, sehingga menjadi seperti berikut:

```
package proyek735;

import java.sql.*;

/**
 *
 * @author Administrator
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        String user="root";
        String pwd ="admin";
        String host="localhost";
        String db="latihan_bab6";
        String urlValue="";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            urlValue="jdbc:mysql://"+ host +"/"+
                db +"?user=" + user +
                "&password="+ pwd;

            Connection conn=
              DriverManager.getConnection(urlValue);

            PreparedStatement pStatement = null;
            String sql ="delete from user " +
                    " where user_id=?";
             pStatement = conn.prepareStatement(sql);
            pStatement.setString(1, "Roshdiana");
             int intBaris= pStatement.executeUpdate();
            if (intBaris>0)
                System.out.println(
```

```
                    "Penghapusan data berhasil");
            else
                System.out.println(
                  "Penghapusan data gagal");

            pStatement.close();
            conn.close();
        }

        catch (SQLException e){
          System.out.println("koneksi gagal " +
          e.toString());
        }

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

3. Jangan lupa untuk memasang driver JDBC MySQL-nya, perhatikan contoh sebelumnya **Proyek731**.
4. Lakukan kompilasi (Clean and Build), kemudian jalankan program. Dan lihat apa yang muncul dalam jendela output?



Gambar 7.6 Output Proyek735

Proses-proses tersebut adalah proses dasar dalam aplikasi database. Apa pun bentuk formnya, secara teknis adalah sama sehingga sebelum melanjutkan ke tahap berikutnya, sebaiknya Anda menguasai betul teknik ini dengan melakukan improvisasi sendiri. Misalkan menjadikan satu semua aplikasi tersebut, atau menambahkan sebuah fungsional dari contoh lain. Bisa juga melakukan operasi baca tanpa menggunakan parameter (Statement).

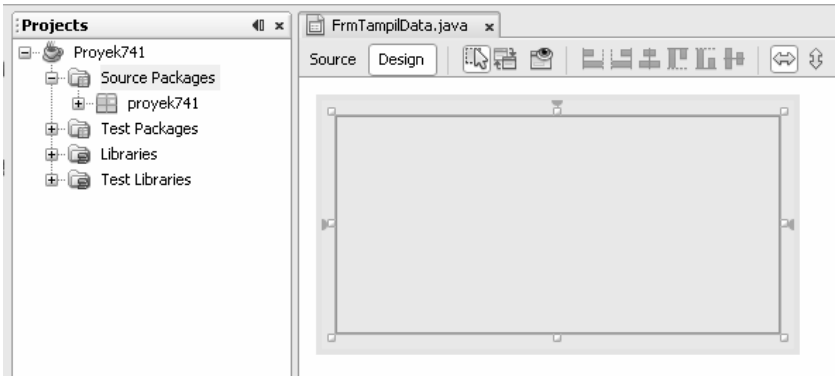
7.5 Latihan Akses Database dengan Form

Setelah pembaca menguasai teknik dasar dalam operasi database, berikutnya adalah perlu melakukan dengan tampilan form, agar lebih menarik dan mampu menyaingi aplikasi profesional. Karena hampir semua aplikasi masa kini menggunakan form (template tampilan). Hal ini karena faktor kemudahan dalam design dan perawatan.

7.5.1 Menampilkan Tabel dalam JTable

Menampilkan data dalam bentuk tabel, yaitu baris dan kolom seperti halnya tabel-tabel dalam MS Excel, sering kali diperlukan dalam sebuah aplikasi. Java menyediakan objek tabel tersebut yang dikenal dengan Jtable. Berikut ini teknik untuk menampilkan data dari database ke dalam sebuah JTabel pada form. Berikut langkah-langkahnya:

1. Buat proyek baru (Java Application), beri nama **Proyek741**, tanpa kelas Main, buat proyeknya saja.
2. Tambahkan sebuah paket, beri nama **proyek741**, dengan cara klik kanan proyek → **New → Java Package…**
3. Tambahkan sebuah JFrame, dengan cara klik kanan paket **proyek741 → New → JFrame Form…**
4. Tambahkan sebuah objek **JScroolPane** ke dalam form, atur melebar di bagian tengah form.



Gambar 7.7 Posisi JScrollPane

5. Masuk dalam editor source program, kemudian edit programnya, dengan cara menambahkan import seperti berikut ini (tulis yang dicetak tebal saja):

```
package proyek741;

import java.sql.*;
import javax.swing.JTable;
```

6. Buat metode dalam kelas ini, untuk mengambil data dari tabel **user**, tulis di bagian paling bawah sebelum tanda akhir kelas/kurung kurawal "}". Berikut ini metodenya:

```
private Object[][] getData(){

   String user="root";
   String pwd ="admin";
   String host="localhost";
   String db="latihan_bab6";
   String urlValue="";

   Object[][] data1= null;

   try {
      Class.forName("com.mysql.jdbc.Driver");
      urlValue="jdbc:mysql://"+ host +"/"+
        db +"?user=" + user +
        "&password="+ pwd;

      Connection conn=
         DriverManager.getConnection(urlValue);

      Statement st = conn.createStatement();
      ResultSet rs = st.executeQuery(
        "Select * from user");

      rs.last();
      int rowCount= rs.getRow();
      rs.beforeFirst();

      data1= new Object[rowCount][3];
      int no=-1;
      while (rs.next()) {
          no=no+1;
          data1[no][0]=rs.getString("user_id");
          data1[no][1]=rs.getString("password");
          data1[no][2]=rs.getString("jabatan");
```

```
      }

      st.close();
      conn.close();

   }

   catch(ClassNotFoundException e) {
      System.out.println("jdbc.Driver tidak ditemukan");
   }

   catch (SQLException e){
      System.out.println("koneksi gagal " +
         e.toString());
   }

   return data1;
}
```

Perhatikan penjelasan berikut ini:

- `rs.last();` adalah menunjuk record paling terakhir.
- `int rowCount= rs.getRow();` adalah untuk mendapatkan nomor record sekarang/aktif. Hal ini untuk memperoleh jumlah recordnya karena ini akan dibutuhkan untuk melakukan inisialisasi ukuran array pada objek JTable yang akan menampilkannya.
- `rs.beforeFirst();` adalah untuk mengarahkan kursor database yang tadinya pada posisi terakhir, kembali ke awal lagi, untuk ditampilkan semua datanya.
- `data1[no][0]` adalah menentukan letaknya dalam array, variabel **no** adalah menyatakan baris (record), sedangkan 0 adalah kolomnya.

7. Tambahkan lagi sebuah metode untuk inisialisasi JTable, dan letakkan di bawah metode sebelumnya seperti berikut ini.

```
private void tampilTabel() {

   String[] columnNames =
       {"User ID", "Password","Jabatan"};

   JTable table = new JTable(
```

```
        getData(), columnNames);

    jScrollPane1.setViewportView(table);
}
```

Parameter aktual inisialisasi JTable() adalah getData (bertipe array objek dua dimensi) dan columnNames (berisi array dari judul tabel).

Metode **`setViewportView()`** adalah untuk mengeset objek yang akan ditampilkan dalam **ScrollPane**. Objek ini memiliki kemampuan untuk menampilkan objek yang dilengkapi dengan layar gulung/scroll.

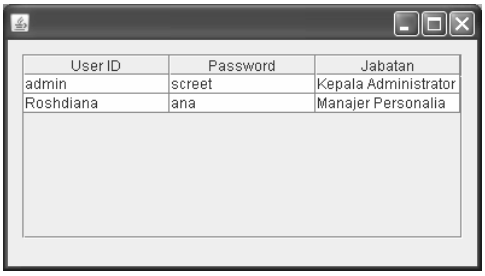
8. Selanjutnya, tambahkan perintah untuk memanggil metode `tampilData()` tersebut. Pada metode konstruktor `FrmTampilData()`, di bawah initComponents() seperti berikut:

```
public FrmTampilData() {
    initComponents();

    tampilTabel();
}
```

Tulis yang dicetak tebal saja. Pemanggilan **initComponents()** dan **tampilData()** dijalankan ketika form ini diciptakan/diinisialisasi.

9. Selanjutnya compile dan jalankan program Anda. Jika belum dipilih kelas utamanya, masuk dalam jendela **Project – Properties**, tentukan kelas mainnya dalam Categories **Run.**

10. Agar lebih menarik dan terlihat isi datanya banyak, Anda bisa menambahkan record lagi melalui PhpMyAdmin atau menjalankan **proyek733**.

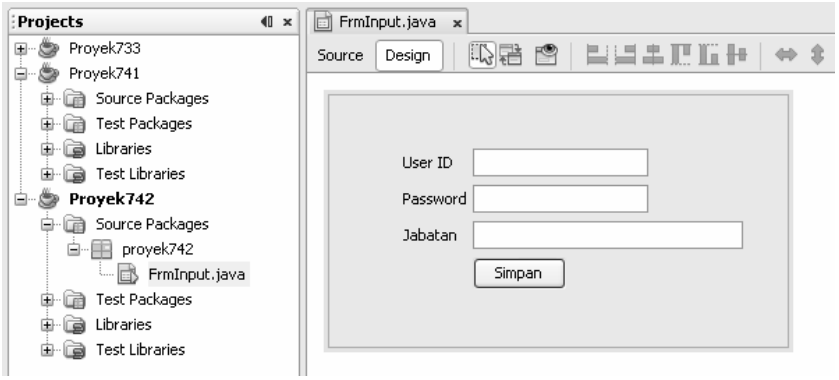


Gambar 7.8 Tampilan JTable

7.5.2 Membuat Form Input

Hal yang tak kalah menariknya adalah waktu membuat input data dari sebuah form yang akan disimpan ke dalam database, secara konsep sama saja dengan contoh sederhana sebelumnya. Akan tetapi, dalam kasus ini memasukkannya secara dinamis dari keybord. Untuk itu lakukan langkah-langkah berikut ini:

1. Buat proyek baru (Java Application), beri nama **Proyek742**, tanpa kelas Main, buat proyeknya saja.
2. Tambahkan sebuah paket, beri nama **proyek742**, dengan cara klik kanan proyek → **New → Java Package…**
3. Tambahkan sebuah JFrame, dengan cara klik kanan paket **proyek742 → New → JFrame Form…**
4. Tambahkan tiga buah objek JLable, berikan nilai properti Textnya masing-masing "User ID", "Password", "Jabatan".
5. Tambahkan tiga buah JTextField, kosongkan properti Text-nya, dan atur sedemikian rupa sehingga menjadi seperti berikut.



Gambar 7.9 Design Form Input User

6. Tambahkan dalam bagian import menjadi seperti berikut ini (tulis yang dicetak tebal saja):

```
package proyek743;

import java.sql.*;
import javax.swing.JOptionPane;
```

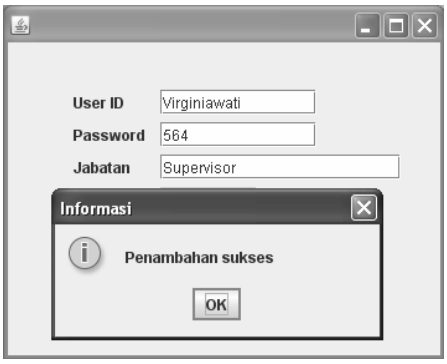
7. Selanjutnya masukkan event klik (actionPerformed) pada jButton1, dan masukkan kode program sehingga menjadi seperti berikut:

```
private void jButton1ActionPerformed(
                  java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
   String user="root";
   String pwd ="admin";
   String host="localhost";
   String db="latihan_bab6";
   String urlValue="";
   try {
      Class.forName("com.mysql.jdbc.Driver");
      urlValue="jdbc:mysql://"+ host +"/"+
         db +"?user=" + user +
         "&password="+ pwd;

      Connection conn=
           DriverManager.getConnection(urlValue);

      PreparedStatement pStatement = null;
      String sql ="insert into user " +
        "(user_id, password, jabatan)" +
        "Values (?,?,?)";
      pStatement = conn.prepareStatement(sql);
      pStatement.setString(1, jTextField1.getText());
      pStatement.setString(2, jTextField2.getText());
      pStatement.setString(3, jTextField3.getText());

      int intTambah= pStatement.executeUpdate();
      if (intTambah>0)
        JOptionPane.showMessageDialog(this,
           "Penambahan sukses", "Informasi",
           JOptionPane.INFORMATION_MESSAGE);
      else
         JOptionPane.showMessageDialog(this,
           "Penambahan gagal", "Informasi",
           JOptionPane.INFORMATION_MESSAGE);

      pStatement.close();
      conn.close();

      jTextField1.setText("");
      jTextField2.setText("");
      jTextField3.setText("");
   }

   catch(ClassNotFoundException e) {
     System.out.println("jdbc.Driver tidak ditemukan");
```

```
    }

    catch (SQLException e){
      System.out.println("koneksi gagal " +
        e.toString());
    }
}
```

Secara umum penjelasan kode program tersebut sudah kita bicarakan sebelumnya. Pada pemasukan nilai dari keybord, user id didapatkan dari jTextField1. Sedangkan password dari jTextField2, dan jabatan dari jTextField3. Untuk memperolehnya cukup dengan metode getText().

8. Jangan lupa untuk memasang driver JDBC MySQL-nya.
9. Lakukan kompilasi dan jalankan program. Perhatikan respon yang muncul. Untuk memastikan operasinya berhasil, Anda bisa membuka PhpMyAdmin atau menjalankan **proyek741**.



Gambar 7.10 Tampilan Proyek742

7.5.3 Membuat Form Edit

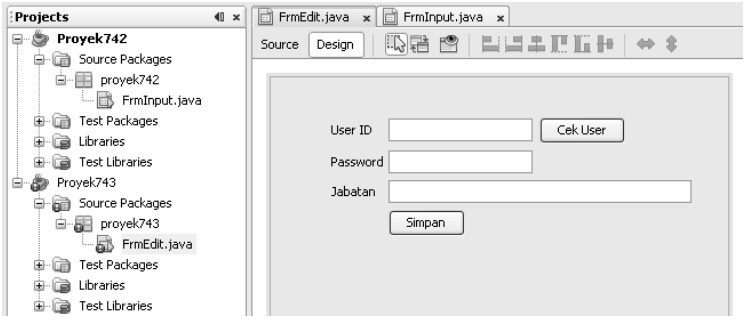
Dalam aplikasi database, kebutuhan akan perubahan data hampir tidak dapat dihindari, entah karena perubahan waktu atau alasan lain. Database dituntut untuk bisa dinamis mengikuti perkembangan zaman. Berikut ini kita akan membuat sebuah contoh program untuk melakukan perubahan data/edit data, ikuti langkah-langkah berikut:

1. Buat proyek baru (Java Application), beri nama **Proyek743**, tanpa kelas Main, buat proyeknya saja.

2. Tambahkan sebuah paket, beri nama **proyek743**, dengan cara klik kanan proyek → **New → Java Package…**.

3. Tambahkan sebuah JFrame, dengan cara klik kanan paket **proyek743 → New → JFrame Form…**.

4. Buat form seperti pada latihan **Proyek742** sebelumnya, dan tambahkan sebuah JButton pada sebelah kanan isian **User ID**. Beri nama variabelnya menjadi bCekUser, dengan cara klik kanan tombol tersebut, pilih → **Change Variable Name…**, sehingga design form menjadi seperti berikut.



Gambar 7.11 Design Proyek743

5. Tambahkan dalam bagian import menjadi seperti berikut ini.

```
package proyek743;

import java.sql.*;
import javax.swing.JOptionPane;
```

6. Tambahkan beberapa properti pada bagian deklarasi, di bawah penyebutan nama kelas form, seperti berikut ini (tulis yang dicetak tebal saja):

```
public class FrmEdit extends javax.swing.JFrame {

    private String user="root";
    private String pwd ="admin";
    private String host="localhost";
    private String db="latihan_bab6";
    private String urlValue="";
    private Connection Lconnection=null;

    /** Creates new form FrmEdit */
```

7. Tambahkan beberapa perintah untuk inisialisasi koneksi pada metode konstruktornya, sehingga menjadi seperti berikut (tulis yang dicetak tebal saja):

```
public FrmEdit() {
   initComponents();

   jButton1.setEnabled(false);
   try {

    Class.forName("com.mysql.jdbc.Driver");
    urlValue="jdbc:mysql://"+ host +"/"+
      db +"?user=" + user +
      "&password="+ pwd;

    Lconnection=DriverManager.getConnection(urlValue);
  }

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

  catch (SQLException e){
    System.out.println("koneksi gagal:" +
      e.toString());
  }
}
```

8. Selanjutnya masukkan event klik (actionPerformed) pada jButton1, dan masukkan kode program sehingga menjadi seperti berikut:

```
private void jButton1ActionPerformed(
                  java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
   try {

     PreparedStatement pStatement = null;
     String sql ="update user " +
       "set password=?, jabatan=? " +
       " where user_id=? ";
     pStatement = Lconnection.prepareStatement(sql);

     pStatement.setString(1, jTextField2.getText());
     pStatement.setString(2, jTextField3.getText());
     pStatement.setString(3, jTextField1.getText());

     int intTambah= pStatement.executeUpdate();
     if (intTambah>0)
```

```
      JOptionPane.showMessageDialog(this,
        "Edit sukses", "Informasi",
        JOptionPane.INFORMATION_MESSAGE);
    else
      JOptionPane.showMessageDialog(this,
        "Edit gagal", "Informasi",
        JOptionPane.INFORMATION_MESSAGE);

    pStatement.close();

    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
  }

  catch (SQLException e){
    System.out.println("koneksi gagal " +
       e.toString());
  }
}
```

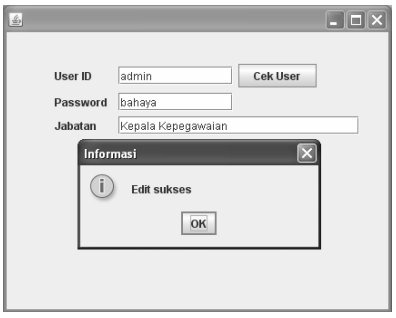
9. Selanjutnya masukkan event klik (actionPerformed) pada bCekUser, dan masukkan kode program sehingga menjadi seperti berikut:

```
private void bCekUserActionPerformed (
                  java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
   try {
     Statement st = Lconnection.createStatement();
     ResultSet rs = st.executeQuery(
        "Select * from user" +
        " where user_id='" +
        jTextField1.getText() + "'");

    if (rs.next()) {
      jButton1.setEnabled(true);
      jTextField2.setText(rs.getString("password"));
      jTextField3.setText(rs.getString("jabatan"));

    } else {
      JOptionPane.showMessageDialog(this,
      "User ID: Salah", "Informasi",
      JOptionPane.INFORMATION_MESSAGE);

      jButton1.setEnabled(false);
      jTextField1.setText("");
      jTextField2.setText("");
      jTextField3.setText("");
```

```
        jTextField1.requestFocus();
      }
    }
    catch (SQLException e){
     System.out.println("koneksi gagal " + e.toString());
    }
}
```

Perhatikan pada perintah `if (rs.next())`, hal ini untuk mengecek apakah dalam *resulset* ditemukan sebuah data/Query dengan User ID yang dimasukkan ada dalam database atau tidak. Jika tidak ada, tentunya kondisi if tersebut akan menghasilkan false.

10. Tambahkan kelas JDBC MySQL.
11. Compile ulang dan jalankan programnya. Masukkan User ID kemudian tekan **Cek User**. Jika data ditampilkan, lakukan perubahan pada password atau jabatan saja (dalam aplikasi ini, tidak diizinkan mengganti User ID), kemudian tekan **Simpan**.



Gambar 7.12 Tampilan Edit Data Sukses

Apabila data User ID yang dimasukkan salah, akan muncul konfirmasi data User ID salah, kemudian isian data akan dibersihkan (form dikosongkan kembali).

Gambar 7.13 Tampilan Ketika Salah Input User

7.5.4 Membuat Form Hapus

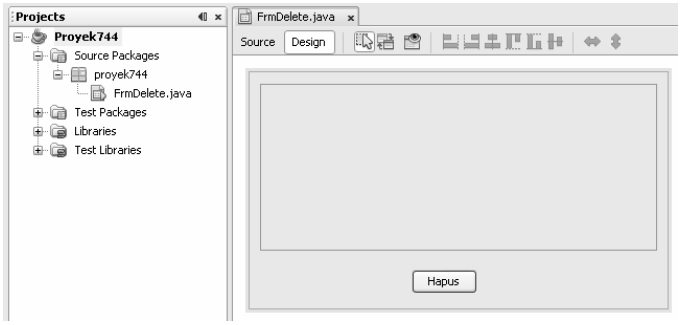
Dalam beberapa kondisi tertentu, sebuah aplikasi membutuhkan fitur hapus atau untuk membersihkan data, karena adanya sebuah alasan terentu, biasanya dilengkapi dengan fitur backup. Namun, dalam kesempatan ini kita akan membicarakan operasi hapus saja. Berikut ini langkah pembuatan form hapus data, ikuti dengan saksama:

1. Buat proyek baru (Java Application), beri nama **Proyek744**, tanpa kelas Main, buat proyeknya saja.
2. Tambahkan sebuah paket, beri nama **proyek744**, dengan cara klik kanan proyek → **New → Java Package…**
3. Tambahkan sebuah JFrame, dengan cara klik kanan paket **proyek744 → New → JFrame Form…**.
4. Tambahkan sebuah JScrollPane, letakkan di tengah melebar, dan tambahkan sebuah JButton sebagai tombol hapus.
5. Tambahkan kode pada bagian import seperti berikut (tulis yang dicetak tebal saja):

```
package proyek744;

import java.sql.*;
import javax.swing.JOptionPane;
import javax.swing.JTable;
```

6. Tambahkan beberapa properti pada kelas form ini, seperti berikut. Tulis yang dicetak tebal.

```
public class FrmDelete extends javax.swing.JFrame {

   private String user="root";
   private String pwd ="admin";
   private String host="localhost";
   private String db="latihan_bab6";
   private String urlValue="";
   private Connection Lconnection=null;
   private JTable table=null;
```



Gambar 7.14 Design Proyek744

7. Tambahkan sebuah metode untuk mengambil data dari database, letakkan di bagian paling bawah, sebelum ada tanda kurung akhir kelas "}". Berikut ini metodenya:

```
private Object[][] getData(){

    Object[][] data1= null;

    try {

      Statement st = Lconnection.createStatement();
      ResultSet rs = st.executeQuery(
           "Select * from user");

      rs.last();
      int rowCount= rs.getRow();
      rs.beforeFirst();

      data1= new Object[rowCount][3];
      int no=-1;
      while (rs.next()) {
           no=no+1;
           data1[no][0]=rs.getString("user_id");
           data1[no][1]=rs.getString("password");
           data1[no][2]=rs.getString("jabatan");
      }
      st.close();
    }
    catch (SQLException e){
      System.out.println("koneksi gagal " +
        e.toString());
    }

    return data1;
}
```

8. Tambahkan metode untuk melakukan inisialisasi JTable agar menempel pada JscrollPane. Letakkan metode tersebut di bawah metode **getData()** sebelumnya. Berikut ini metodenya:

```
private void tampilTabel() {

         String[] columnNames =
           {"User ID", "Password","Jabatan"};

         table = new JTable(
           getData(), columnNames);

        jScrollPane1.setViewportView(table);
}
```

9. Tambahkan beberapa perintah untuk inisialisasi koneksi pada metode konstruktornya, sehingga menjadi seperti berikut (tulis yang dicetak tebal saja):

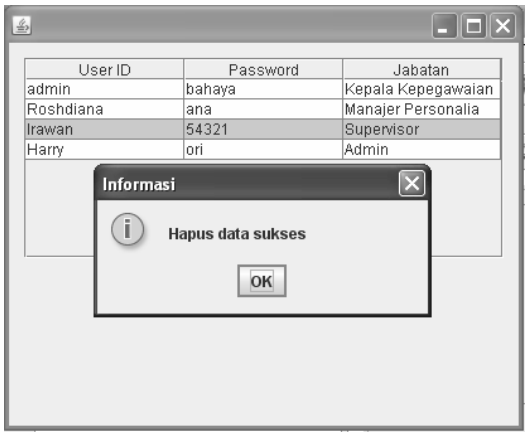
```
public FrmEdit() {
   initComponents();

   try {

    Class.forName("com.mysql.jdbc.Driver");
    urlValue="jdbc:mysql://"+ host +"/"+
      db +"?user=" + user +
      "&password="+ pwd;

    Lconnection=
       DriverManager.getConnection(urlValue);

   }

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

   catch (SQLException e){
      System.out.println("koneksi gagal:" +
        e.toString());
   }

   tampilTabel();
}
```

10. Selanjutnya masukkan event klik (actionPerformed) pada tombol hapus (jButton1), dan masukkan kode program sehingga menjadi seperti berikut (tulis yang dicetak tebal saja):

```
private void jButton1ActionPerformed(
                  java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:

    String usrId = table.getValueAt(
       table.getSelectedRow(),0).toString();

    try {

      PreparedStatement pStatement = null;
      String sql ="delete from user " +
        " where user_id=? ";
      pStatement = Lconnection.prepareStatement(sql);

      pStatement.setString(1, usrId);
      int intTambah= pStatement.executeUpdate();
      if (intTambah>0)
        JOptionPane.showMessageDialog(this,
          "Hapus data sukses", "Informasi",
          JOptionPane.INFORMATION_MESSAGE);
      else
         JOptionPane.showMessageDialog(this,
            "Hapus data gagal", "Informasi",
            JOptionPane.INFORMATION_MESSAGE);

      pStatement.close();

      tampilTabel();
   }

   catch (SQLException e){
     System.out.println("koneksi gagal " +
        e.toString());
   }
}
```

11. Tambahkan kelas JDBC MySQL.

12. Compile ulang dan jalankan programnya. Semua data yang tersimpan dalam tabel user akan ditampilkan dalam Jtable.



Gambar 7.15 Design Proyek744

Untuk melakukan hapus data, pilih usernya dalam grid dan klik tombol **Hapus**, sehingga akan muncul jendela pesan keberhasilan proses hapus. Klik saja tombol **OK**, maka data sudah dihilangkan dari form dan juga dari database.

ooo0ooo