

# BAB XVII

## MENYIMPAN RECORD, DELETE RECORD, UPDATE RECORD MELALUI FORM TERKONEKSI JAVA DATABASE

---

### A. CAPAIAN PEMBELAJARAN

1. Mahasiswa dapat memahami perintah Menyimpan, Delete Record, Update dari Form terkoneksi database
2. Mahasiswa dapat mengaplikasikan kode program Menyimpan, Delete Record, Update pada Form terkoneksi Database dengan Kode Java
3. Mahasiswa dapat membuat program dengan Menyimpan, Delete Record, Update pada Form yang terkoneksi Database

### B. MATERI

#### 1. Menyimpan Record

Pada pertemuan sebelumnya Kita sudah membuat tombol **Add New Record**, Tombol tersebut digunakan untuk menyiapkan record baru, dan mengosongkan semua kotak isian textfield, tetapi belum tersimpan

kedalam tabel. Untuk menyimpannya kita sudah siapkan tombol **Save New Record** pada Form.

Sebelum kita menyimpan Record baru, kita harus memindahkan kursor ke sesuatu yang disebut menyisipkan baris. Ini membuat catatan kosong di ResultSet. kemudian menambahkan data ke ResultSet kode programnya seperti dibawah ini :

```
rs.moveToInsertRow( );  
rs.updateInt("ID", newID);  
rs.updateString("First_Name", first);  
rs.updateString("Last_Name", last);  
rs.updateString("Job_Title", job);  
rs.insertRow( );
```

Setelah menambahkan data ke ResultSet, baris terakhir menyisipkan baris baru. Namun, untuk melakukan perubahan apa pun ke database yang akan kita lakukan adalah menutup Statement objek dan ResultSet objek kita. kemudian dapat memuat ulang semuanya. Jika kita tidak melakukan ini, maka record baru tidak akan ditambahkan, baik ke ResultSet atau database. (Ini karena jenis Driver yang digunakan) Untuk menutup Statement atau ResultSet, kita cukup mengeluarkan perintah close:

```
stmt.close( );  
rs.close( );
```

Kode untuk memuat ulang semuanya sama dengan kode yang Anda tulis saat form pertama kali dipanggil :

```
stmt =  
con.createStatement(ResultSet.TYPE_SCROLL  
L_SENSITIVE,  
ResultSet.CONCUR_UPDATABLE);  
String sql = "SELECT * FROM Workers";  
rs = stmt.executeQuery(sql);  
rs.next( );  
  
int id_col = rs.getInt("ID");  
String id = Integer.toString(id_col);  
String first2 = rs.getString("First_Name");  
String last2 = rs.getString("Last_Name");  
String job2 = rs.getString("Job_Title");  
  
textID.setText(id);  
textFirstName.setText(first2);  
textLastName.setText(last2);  
textJobTitle.setText(job2);
```

Kita tidak melakukan sesuatu yang berbeda, di sini: hanya memilih semua Record lagi dan meletakkan record yang pertama di Bidang TeksField.

Berikut ini semua kode yang menyimpan Record baru ke database (Jelas, banyak dari kode ini bisa menjadi metodenya sendiri):

```
String first = textFirstName.getText();
String last = textLastName.getText();
String job = textJobTitle.getText();
String ID = textID.getText();

int newID = Integer.parseInt(ID);

try {
    rs.moveToInsertRow();

    rs.updateInt( "ID", newID );
    rs.updateString( "First_Name", first );
    rs.updateString( "last_Name", last );
    rs.updateString( "Job_Title", job );

    rs.insertRow();

    stmt.close();
    rs.close();

    stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                               ResultSet.CONCUR_UPDATABLE );

    String SQL = "SELECT * FROM Workers";
    rs = stmt.executeQuery( SQL );

    rs.next();

    int id_col = rs.getInt("ID");
    String id = Integer.toString(id_col);
    String first2 = rs.getString("First_Name");
    String last2 = rs.getString("Last_Name");
    String job2 = rs.getString("Job_Title");

    textID.setText(id);
    textFirstName.setText(first2);
    textLastName.setText(last2);
    textJobTitle.setText(job2);

    btnFirst.setEnabled( false );
    btnPrevious.setEnabled( false );
    btnNext.setEnabled( false );
    btnLast.setEnabled( false );
    btnUpdateRecord.setEnabled( false );
    btnDeleteRecord.setEnabled( false );
    btnNewRecord.setEnabled( false );

    btnSaveRecord.setEnabled( true );
    btnCancelNewRecord.setEnabled( true );

    JOptionPane.showMessageDialog(this, "Record Saved");
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(this, err.getMessage());
}
```

Kodenya agak panjang, tetapi kita dapat menyalin dan menempel banyak dari metode DoConnect.

Satu masalah lainnya adalah kolom ID harus unik. Idealnya, kita akan menulis rutin untuk mendapatkan nomor ID terakhir, lalu menambahkan satu ke dalamnya. Basis data lain, seperti MySQL, memiliki nilai AutoIncrement untuk menanganinya. Pastikan bahwa nilai ID bukan yang telah di gunakan sebelumnya, jika tidak, akan mendapatkan pesan kesalahan.

jalankan program dan ujilah. Jika tidak terdapat kesalahan kita sudah mempunyai record baru tersimpan ke tabel database kita.

## 2. Delete Record

Tombol **Delete Recod** digunakan untuk menghapus data/baris record yang tidak digunakan pada tabel record tersimpan. Menghapus baris dapat dilakukan dengan mudah, cukup gunakan metode `deleteRow` dari `ResultSet`:

***rs.deleteRow( );***

Namun, Driver yang kita gunakan, `ClientDriver`, meninggalkan baris kosong di tempat data yang telah dihapus. Jika kita mencoba berpindah ke baris tersebut menggunakan tombol atau ***Previous***, Bidang Teks ID akan memiliki 0 di dalamnya, dan yang lainnya akan kosong.

Untuk mengatasi masalah ini pertama-tama kita akan menghapus satu baris kemudian tutup objek `Statement` dan objek `ResultSet`. kemudian dapat

memuat ulang semua data di Bidang TeksField. Dengan begitu, kita tidak akan memiliki baris kosong.

Berikut kode untuk ditambahkan untuk tombol **Delete Record**:

```
try {
    rs.deleteRow();

    stmt.close();
    rs.close();

    stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                ResultSet.CONCUR_UPDATABLE );

    String SQL = "SELECT * FROM Workers";
    rs = stmt.executeQuery( SQL );

    rs.next();

    int id_col = rs.getInt("ID");
    String id = Integer.toString( id_col );
    String first_name = rs.getString("First_Name");
    String last_name = rs.getString("Last_Name");
    String job = rs.getString("Job_Title");

    textID.setText(id);
    textFirstName.setText(first_name);
    textLastName.setText(last_name);
    textJobTitle.setText(job);

    JOptionPane.showMessageDialog(this, "Record Deleted");
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(this, err.getMessage());
}
```

Jalankan program dan ujilah. Kita sekarang dapat menghapus Reord dari tabel database kita.

### 3. Update Record

Update Record digunakan untuk memperbaharui data yang sudah tersimpan pada tabel, perubahan data ini di sebabkan misalnya adanya kesalahan ketik, atau perubahan data yang disebabkan bukan kesalahan

ketik. Untuk perubahan kita bisa menggunakan ResultSet.

ResultSet memiliki metode Update yang memungkinkan Anda memperbarui catatan tidak hanya di ResultSet itu sendiri, tetapi juga di database yang mendasarinya. Mari kita lihat cara kerjanya.

Buat form kitasedikit lebih panjang. Sekarang tambahkan panel baru ke form. Tambahkan tombol baru ke panel. Ubah nama variabel default menjadi btnUpdateRecord. Ubah teks pada tombol menjadi **Update Record**. kita juga sudah memiliki tombol untuk membuat **Add New Record** untuk menyimpan Record, **Cancel New Record**, dan **Delete Record**. penambaham empat tombol ke panel. Lakukan modifikasi form berikut:

**Button Variable Name: btnNewRecord**

**Button Text: New Record**

**Button Variable Name:**

**btnDeleteRecord**

**Button Text: Delete Record**

**Button Variable Name: btnSaveRecord**

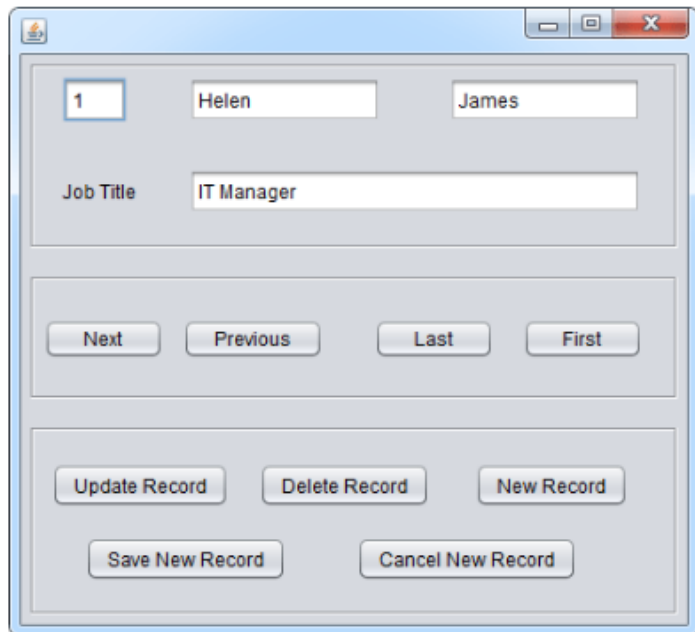
**Button Text: Save New Record**

**Button Variable Name:**

**btnCancelNewRecord**

**Button Text: Cancel New Record**

Setelah selesai, form kita akan terlihat seperti ini (meskipun jangan ragu untuk mengatur ulang tombol):



### Pertemuan 17. 1 Hasil Desain Form

Klik dua kali tombol Perbarui untuk membuat rintisan kode.

Hal pertama yang harus dilakukan adalah mendapatkan teks dari Bidang TextField :

```
String first = textFirstName.getText( );  
String last = textLastName.getText( );  
String job = textJobTitle.getText( );  
String ID = textID.getText( );
```



Tetapi, jika kita ingin memperbarui bidang ID, kita perlu mengonversi String menjadi Integer:

```
int newID = Integer.parseInt( ID );
```

Objek Integer memiliki metode yang disebut `parseInt`. Di antara tanda kurung `parseInt`, kita menetikkan string yang ingin kita konversi.

Sekarang setelah kita memiliki semua data dari Bidang `TextField`, kita dapat memanggil metode `Update` yang relevan dari objek `ResultSet`:

```
rs.updateString( "First_Name", first );
```

Ada beberapa metode ***Update*** yang berbeda untuk dipilih. Yang di atas menggunakan ***updateString***. Tetapi kita memerlukan jenis Kolom dari tabel database di sini. Kita memiliki tiga string (`First_Name`, `Last_Name`, `Job_Title`) dan satu nilai integer (`ID`). Jadi kita membutuhkan tiga metode `updateString` dan satu `updateInt`.

Di antara tanda kurung buka dari metode ***Update***, kita memerlukan nama kolom dari database kita (meskipun ini bisa berupa nilai Indeksnya). Setelah koma kita ketik data pengganti. Jadi, pada contoh di atas, kita ingin memperbarui kolom `First_Name` dan menggantinya dengan nilai yang disimpan dalam variabel yang disebut `first`.

Metode **Update** hanya memperbarui ResultSet. Untuk melakukan perubahan ke database, kita mengeluarkan perintah `updateRow`:

```
rs.updateRow( );
```

Berikut adalah semua baris kode untuk **Update** ResultSet dan tabel database:

```
try {  
    rs.updateInt( "ID", newID );  
    rs.updateString( "First_Name", first );  
    rs.updateString( "last_Name", last );  
    rs.updateString( "Job_Title", job );  
    rs.updateRow( );  
    JOptionPane.showMessageDialog(Wo  
    rkers.this, "Updated");  
    }  
  
    catch (SQLException err) {  
        System.out.println(err.getMessage() );  
    }
```

Sekali lagi, kita perlu membungkus semuanya dalam pernyataan **try ... catch**, untuk berjaga-jaga jika terjadi kesalahan. Perhatikan juga, bahwa kita telah menambahkan kotak pesan untuk update data yang berhasil.

Berikut seluruh kode untuk ditambahkan untuk Tombol **Update** kita :

```

String first = textFirstName.getText();
String last = textLastName.getText();
String job = textJobTitle.getText();
String ID = textID.getText();

int newID = Integer.parseInt(ID);

try {
    rs.updateInt( "ID", newID );
    rs.updateString( "First_Name", first );
    rs.updateString( "last_Name", last );
    rs.updateString( "Job_Title", job );
    rs.updateRow( );
    JOptionPane.showMessageDialog(this, "Updated");
}
catch (SQLException err) {
    System.out.println(err.getMessage());
}

```

Jalankan program dan cobalah, ubah beberapa data di Bidang TeksField (Tommy ke Timmy, misalnya). Kemudian klik tombol **Update** kita. Gulir melewati record lalu kembali. Perubahan itu harus ada. Sekarang tutup program Anda dan jalankan lagi. Jika tidak ada masalah seharusnya menemukan bahwa perubahan itu sudah terjadi permanen.

### C. LATIHAN

1. Jelaskan Potongan kode program dibawah ini setiap barisnya ini digunakan untuk apa? :

```

try {
    rs.updateInt( "ID", newID );
    rs.updateString( "First_Name", first );

```

```

        rs.updateString( "last_Name", last );
        rs.updateString( "Job_Title", job );
        rs.updateRow( );
        JOptionPane.showMessageDialog(Wo
rkers.this, "Updated");
    }

    catch (SQLException err) {
        System.out.println(err.getMessage() );
    }

```

2. Apa arti pada potongan kode dibawah ini yang diberi kotak merah

```

String first = textFirstName.getText();
String last = textLastName.getText();
String job = textJobTitle.getText();
String ID = textID.getText();

int newID = Integer.parseInt(ID);

try {
    rs.updateInt( "ID", newID );
    rs.updateString( "First_Name", first );
    rs.updateString( "last_Name", last );
    rs.updateString( "Job_Title", job );
    rs.updateRow( );
    JOptionPane.showMessageDialog(this, "Updated");
}
catch (SQLException err) {
    System.out.println(err.getMessage());
}

```

#### **D. REFERENSI**

[https://www.homeandlearn.co.uk/java/databases\\_and\\_java\\_frameworks.html](https://www.homeandlearn.co.uk/java/databases_and_java_frameworks.html) ,diakses pada    diakses pada tanggal 10 Juli 2022.

<https://www.w3schools.com/sql/>, diakses pada    diakses pada tanggal 30 Juli 2022

Romi satrio Wahono, Java Gui, 2016