

2011

MySQL dan Java Database Connectivity



Eko Kurniawan Khannedy

StripBandunk

9/1/2011

Daftar Isi

Daftar Isi..... 1

1 MySQL..... 3

1.1 Membuat Database..... 3

1.2 Membuat Tabel..... 5

1.3 Menghapus Tabel..... 8

1.4 Menghapus Database..... 9

1.5 Memasukkan Data..... 10

1.6 Menampilkan Data..... 13

1.7 Penyaringan Data..... 14

1.8 Menggubah Data..... 19

1.9 Menghapus Data..... 21

2 Java Database Connectivity..... 25

2.1 MySQL Connector Java..... 25

2.1.1 Netbeans..... 25

2.1.2 Eclipse..... 26

2.2 Driver..... 28

2.3 Connection..... 29

2.4 Statement..... 30

2.4.1 Memasukkan Data..... 31

2.4.2 Mengubah Data..... 33

2.4.3 Menghapus Data..... 34

2.5 ResultSet..... 36

2.6 PreparedStatement..... 43

2.6.1 Memasukkan Data..... 44

2.6.2	Mengubah Data	46
2.6.3	Menghapus Data	47
2.6.4	Mendapatkan Data	49
2.7	Advanced ResultSet.....	52
2.7.1	Menambah Data	52
2.7.2	Mengubah Data	54
2.7.3	Menghapus Data	56
2.8	Pilih Yang Mana?	57
3	Tentang Penulis	59

1 MySQL

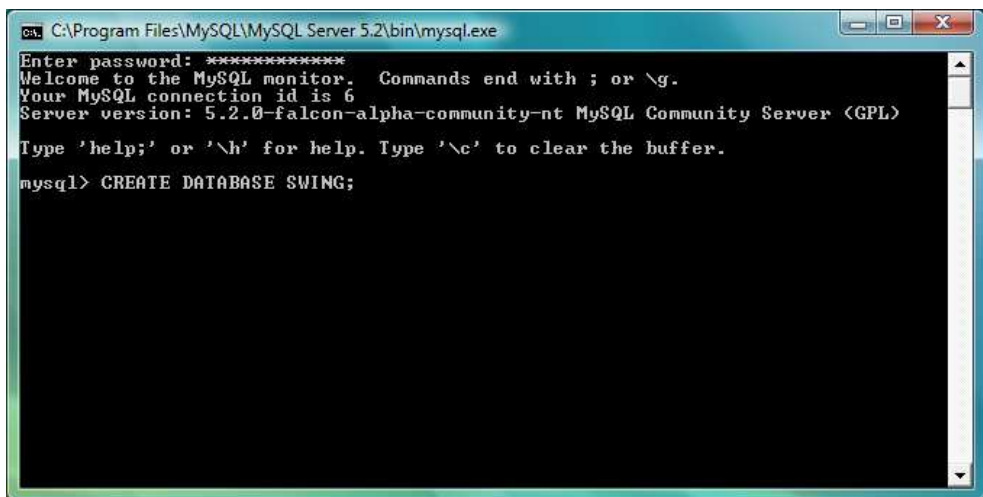
Dalam buku ini saya menggunakan MySQL sebagai DBMS yang akan saya gunakan untuk mengolah data, namun jika anda lebih tertarik untuk menggunakan DBMS selain MySQL anda juga dapat mempraktekan buku ini, karena buku ini sebenarnya tak terlalu tergantung pada DBMS tertentu.

1.1 Membuat Database

Sebelum memanajemen database sudah tentu kita harus membuat databasenya terlebih dahulu. Dan untuk membuat database dalam MySQL kita dapat menggunakan perintah seperti dibawah ini :

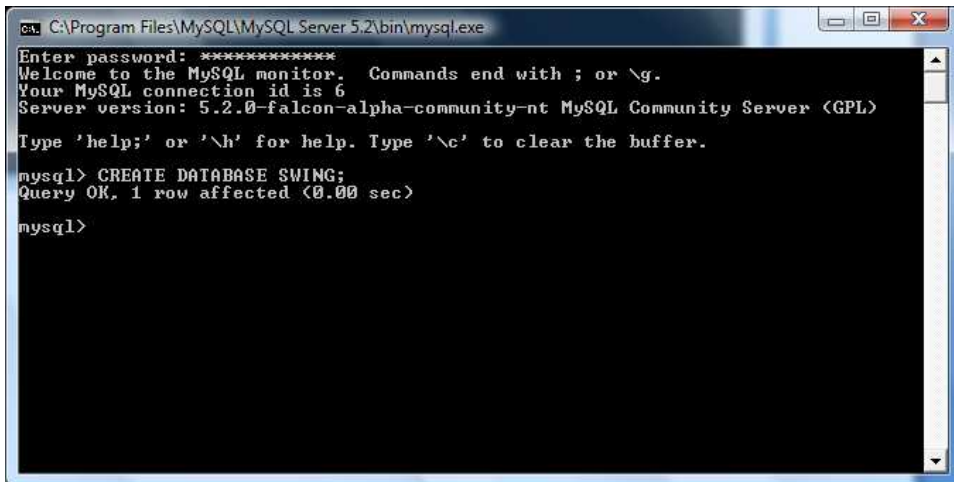
```
CREATE DATABASE NAMA_DATABASE;
```

Misal kita akan membuat database dengan nama "SWING" :



Gambar 1 Membuat Database "SWING"

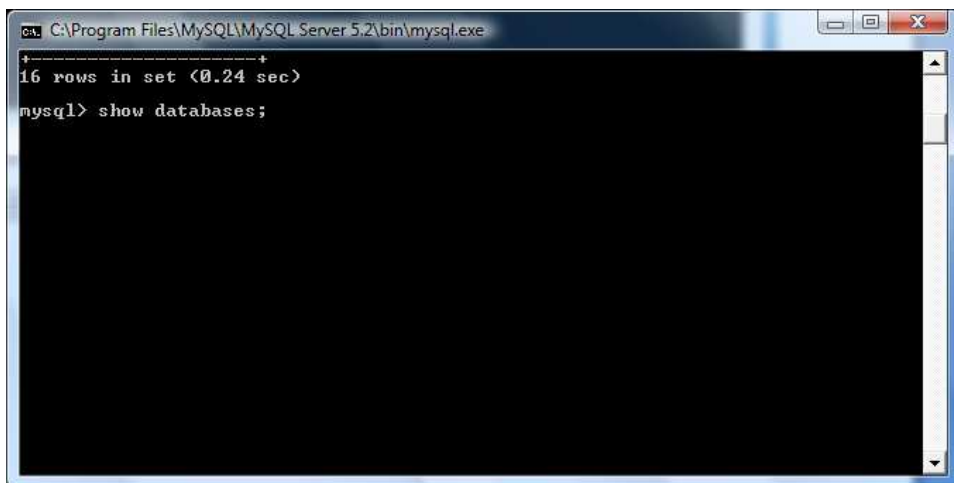
Dan setelah berhasil maka akan tampil seperti ini :



Gambar 2 Tampilan Proses Berhasil Pembuatan Database

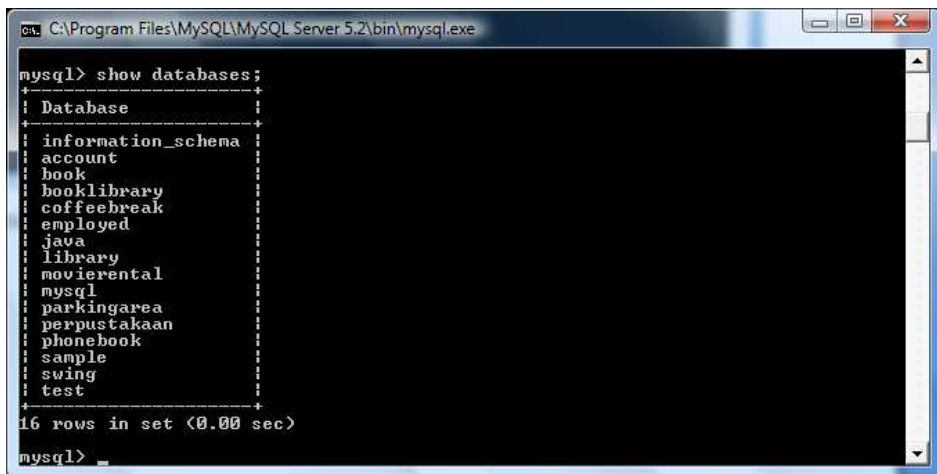
Untuk memastikan kembali bahwa benar-benar database yang anda buat ada gunakan perintah :

```
SHOW DATABASES;
```



Gambar 3 Menampilkan Tabel

Maka MySQL akan memunculkan seluruh nama database yang terdapat dalam DBMS :



Gambar 4 Tampilan Seluruh Tabel

Dan pastikan bahwa database “SWING” atau database yang telah anda buat tadi terdapat dalam DBMS.

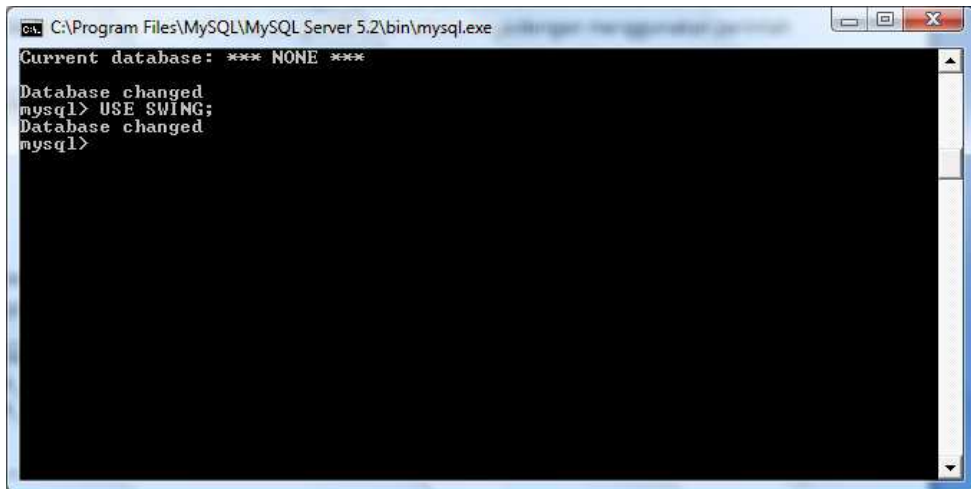
1.2 Membuat Tabel

Tabel merupakan representasi dari entitas atau relasi dalam sebuah database. Untuk lebih jelasnya tentang apa itu entitas dan atau relasi, anda bisa membaca buku tentang “Basis Data” atau tentang “Konsep Database”. Saya merekomendasikan anda untuk membaca buku tentang kedua hal tersebut karena memang sangat berguna ketika kita akan membuat database.

Sebelum kita membuat sebuah tabel kita juga harus menyetakan bahwa kita akan membuat tabel tersebut dalam database “SWING” atau yang anda buat sebelumnya dengan menggunakan perintah :

```
USE NAMA_DATABASE;
```

Misal :

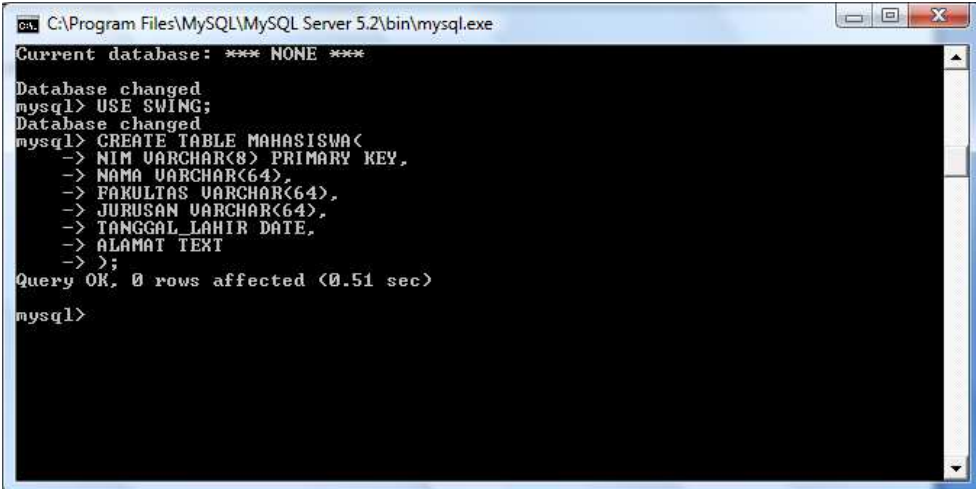


Gambar 5 menggunakan Database

Dengan begitu saat ini kita telah berada dalam database “SWING”. Untuk membuat tabel dalam MySQL kita menggunakan perintah seperti dibawah ini :

```
CREATE TABLE NAMA_TABLE (  
    NAMA_ATRIBUT TIPE_DATA [KETERANGAN],  
    NAMA_ATRIBUT TIPE_DATA [KETERANGAN],  
    ...  
);
```

Misal kita akan membuat tabel mahasiswa :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

Current database: *** NONE ***

Database changed
mysql> USE SWING;
Database changed
mysql> CREATE TABLE MAHASISWA(
-> NIM VARCHAR(8) PRIMARY KEY,
-> NAMA VARCHAR(64),
-> FAKULTAS VARCHAR(64),
-> JURUSAN VARCHAR(64),
-> TANGGAL_LAHIR DATE,
-> ALAMAT TEXT
-> );
Query OK, 0 rows affected (0.51 sec)

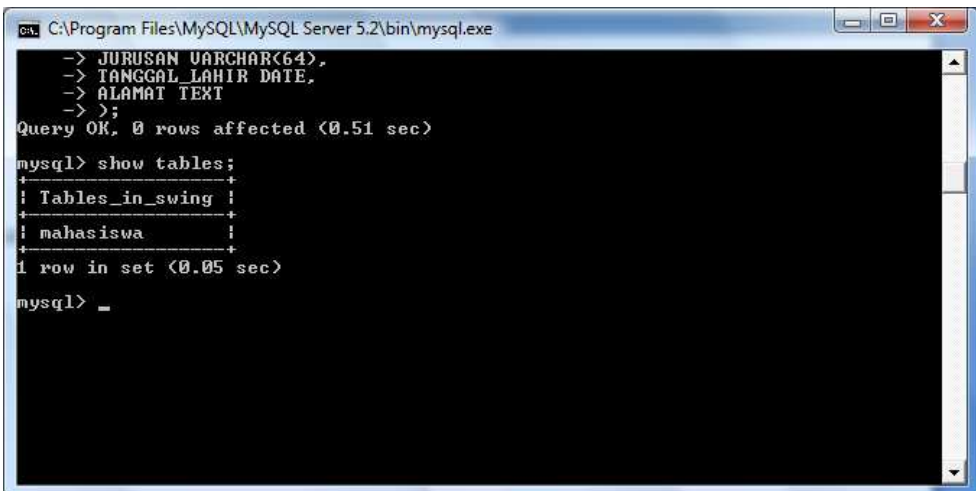
mysql>
```

Gambar 6 Membuat Tabel "MAHASISWA"

Dan saat ini kita telah membuat tabel bernama mahasiswa. Dan untuk meyakinkan bahwa tabel benar-benar telah dibuat, maka gunakan perintah :

```
SHOW TABLES;
```

Perintah diatas digunakan untuk menampilkan tabel yang ada dalam sebuah database.



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

-> JURUSAN VARCHAR(64),
-> TANGGAL_LAHIR DATE,
-> ALAMAT TEXT
-> );
Query OK, 0 rows affected (0.51 sec)

mysql> show tables;
+-----+
| Tables_in_swing |
+-----+
| mahasiswa       |
+-----+
1 row in set (0.05 sec)

mysql> _
```

Gambar 7 Menampilkan Tabel

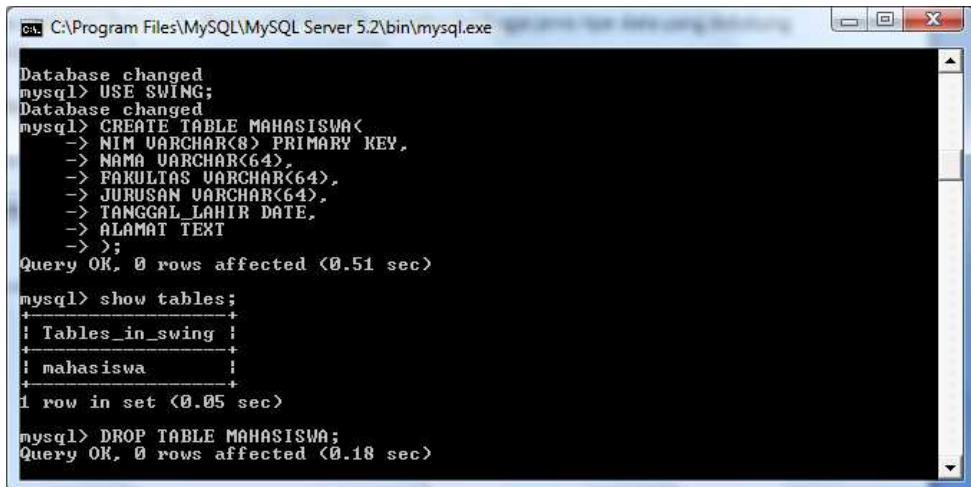
Untuk lebih jelas tentang tipe data dalam MySQL anda bisa melihatnya dalam “MySQL Manual” yang telah saya sediakan dalam CD, disana anda bisa mengetahui berbagai jenis tipe data yang didukung oleh MySQL.

1.3 Menghapus Tabel

Untuk menghapus tabel yang tidak kita perlukan adalah dengan menggunakan perintah :

DROP TABLE NAMA_TABLE

Misal kita akan menghapus tabel mahasiswa tadi :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

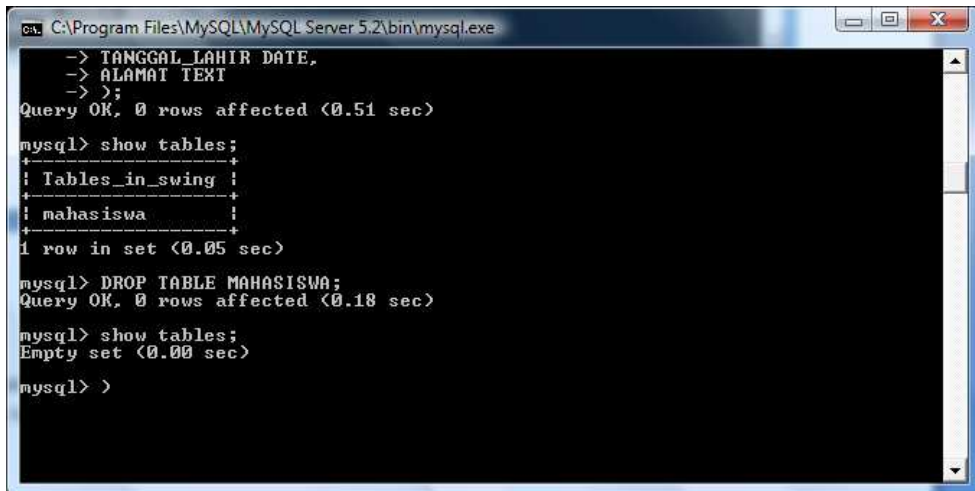
Database changed
mysql> USE SWING;
Database changed
mysql> CREATE TABLE MAHASISWA(
-> NIM VARCHAR(8) PRIMARY KEY,
-> NAMA VARCHAR(64),
-> FAKULTAS VARCHAR(64),
-> JURUSAN VARCHAR(64),
-> TANGGAL LAHIR DATE,
-> ALAMAT TEXT
-> );
Query OK, 0 rows affected (0.51 sec)

mysql> show tables;
+-----+
| Tables_in_swing |
+-----+
| mahasiswa       |
+-----+
1 row in set (0.05 sec)

mysql> DROP TABLE MAHASISWA;
Query OK, 0 rows affected (0.18 sec)
```

Gambar 8 Menaghapus Tabel

Dan untuk memastikan bahwa tabel telah terhapus tampilkanlah seluruh tabel yang ada dalam database tersebut :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
-> TANGGAL_LAHIR DATE,
-> ALAMAT TEXT
-> ;
Query OK, 0 rows affected (0.51 sec)

mysql> show tables;
+-----+
| Tables_in_swing |
+-----+
| mahasiswa       |
+-----+
1 row in set (0.05 sec)

mysql> DROP TABLE MAHASISWA;
Query OK, 0 rows affected (0.18 sec)

mysql> show tables;
Empty set (0.00 sec)

mysql>
```

Gambar 9 Menampilkan Tabel

Empty menyatakan bahwa tak ada table dalam database.

1.4 Menghapus Database

Untuk menghapus database kita bisa menggunakan perintah seperti dibawah ini :

```
DROP DATABASE NAMA_DATABASE;
```

Misal kita akan menghapus database yang tadi kita buat :




```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

mysql> DROP DATABASE SWING;
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

Gambar 10 Menghapus Database

Dan untuk memastikan bahwa database telah terhapus, tampilkanlah seluruh database :



```
Query OK, 0 rows affected (0.00 sec)

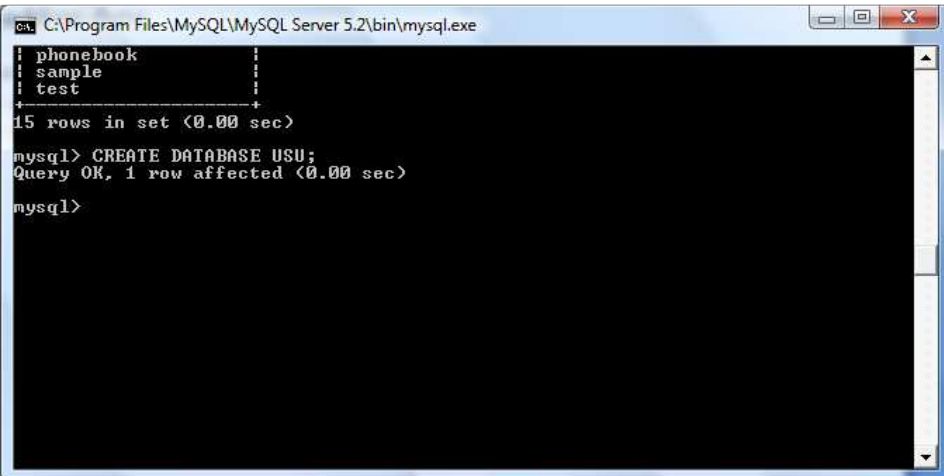
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| account |
| book |
| booklibrary |
| coffeebreak |
| employed |
| java |
| library |
| movierental |
| mysql |
| parkingarea |
| perpustakaan |
| phonebook |
| sample |
| test |
+-----+
15 rows in set (0.00 sec)

mysql>
```

Gambar 11 Menampilkan Database

1.5 Memasukkan Data

Karena untuk memasukkan sebuah data harus ada database dan tabel, jadi sekarang kita buat database dan tabel baru. Misal kita akan membuat database bernama "USU", yang pasti nama database yang akan anda buat harus anda ingat, karena kita akan menggunakannya sampai akhir buku ini:



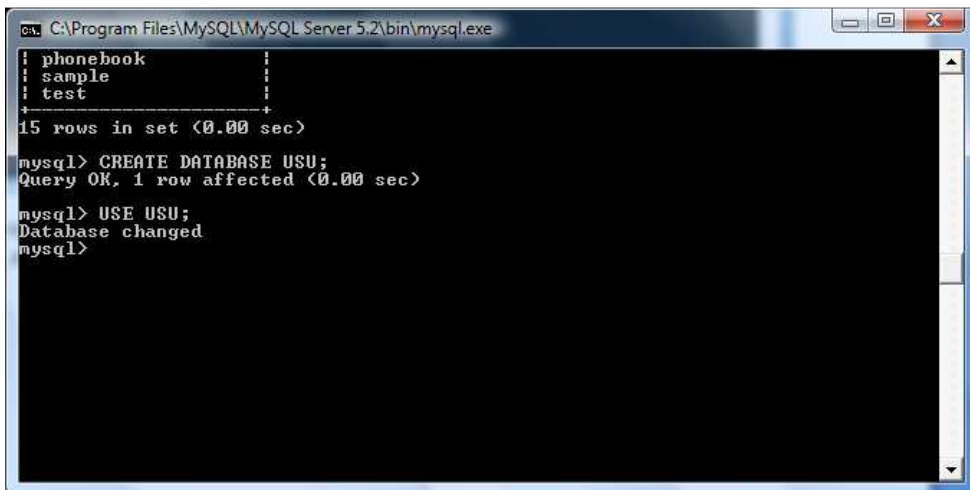
```
phonebook
sample
test
+-----+
15 rows in set (0.00 sec)

mysql> CREATE DATABASE USU;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Gambar 12 Membuat Database "USU"

Dan sekarang kita buat "IDENTITAS" yang didalamnya berisikan atribut ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK. Jangan lupa untuk menggunakan database yang kita buat tadi :



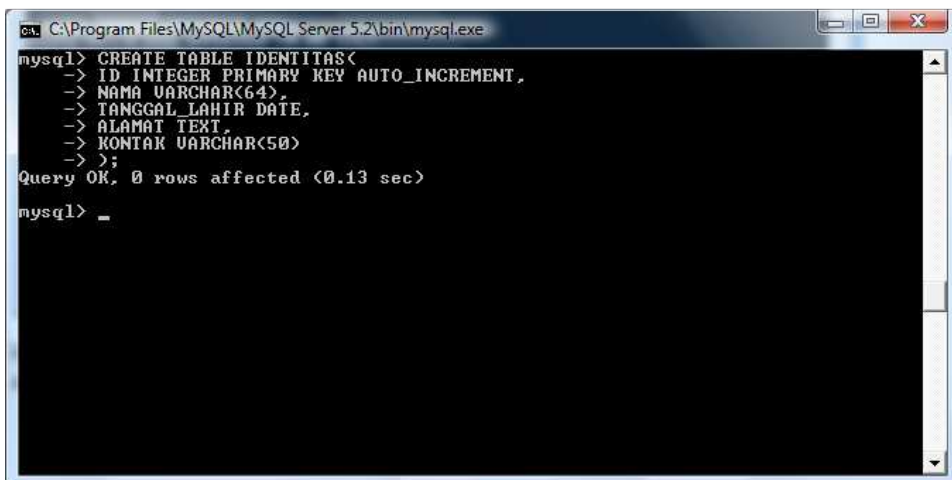
```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
mysql>
+-----+
| phonebook |
| sample    |
| test      |
+-----+
15 rows in set (0.00 sec)

mysql> CREATE DATABASE USU;
Query OK, 1 row affected (0.00 sec)

mysql> USE USU;
Database changed
mysql>
```

Gambar 13 Menggunakan Database "USU"

Sekarang kita buat tabelnya :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
mysql> CREATE TABLE IDENTITAS(
  -> ID INTEGER PRIMARY KEY AUTO_INCREMENT,
  -> NAMA VARCHAR(64),
  -> TANGGAL_LAHIR DATE,
  -> ALAMAT TEXT,
  -> KONTAK VARCHAR(50)
  -> );
Query OK, 0 rows affected (0.13 sec)

mysql> _
```

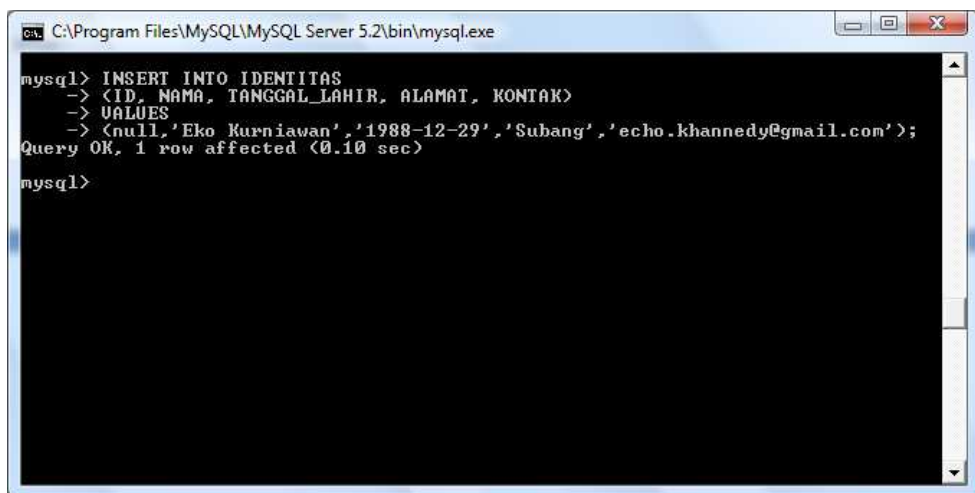
Gambar 14 Membuat Tabel "IDENTITAS"

AUTO_INCREMENT menandakan bahwa data dalam atribut tersebut akan otomatis menaik jika kita memasukkan nilai "null". Sekarang kita akan mencoba

memasukkan data dalam tabel tersebut. Untuk memasukkan data dalam sebuah tabel kita gunakan perintah :

```
INSERT INTO NAMA_TABEL  
(NAMA_ATRIBUT, ... )  
VALUES  
(NILAI_ATRIBUT, ... ),  
(NILAI_ATRIBUT, ... )  
... ;
```

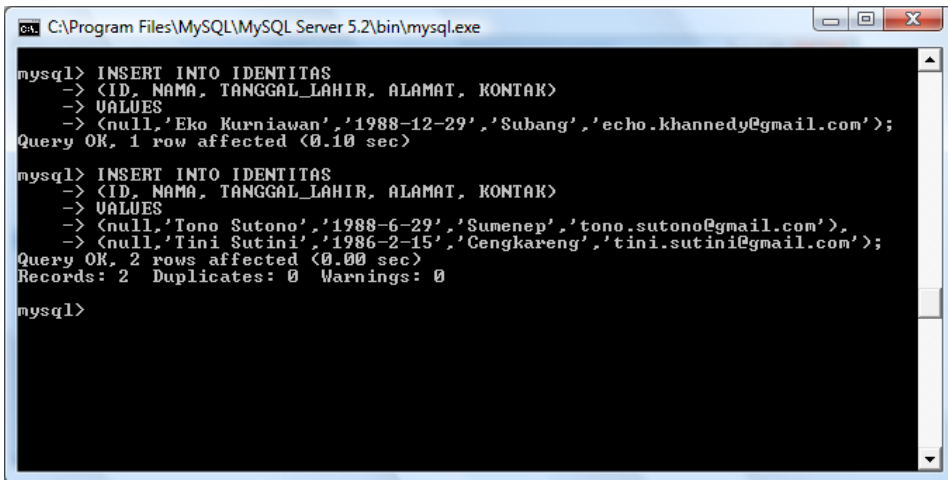
Misal :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe  
mysql> INSERT INTO IDENTITAS  
-> <ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK>  
-> VALUES  
-> <null, 'Eko Kurniawan', '1988-12-29', 'Subang', 'echo.khannedy@gmail.com'>;  
Query OK, 1 row affected (0.10 sec)  
mysql>
```

Gambar 15 Memasukkan Data Ke Tabel IDENTITAS

Diatas kita hanya memasukkan 1 record/baris. Jika kita akan memasukkan lebih dari satu record kita juga bisa menggunakan perintah diatas, misal :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

mysql> INSERT INTO IDENTITAS
  -> <ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK>
  -> VALUES
  -> <null, 'Eko Kurniawan', '1988-12-29', 'Subang', 'echo.khannedy@gmail.com'>;
Query OK, 1 row affected (0.10 sec)

mysql> INSERT INTO IDENTITAS
  -> <ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK>
  -> VALUES
  -> <null, 'Tono Sutono', '1988-6-29', 'Sumenep', 'tono.sutono@gmail.com'>,
  -> <null, 'Tini Sutini', '1986-2-15', 'Cengkareng', 'tini.sutini@gmail.com'>;
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql>
```

Gambar 16 Memasukkan Data Ke Tabel IDENTITAS Lebih Dari Satu

1.6 Menampilkan Data

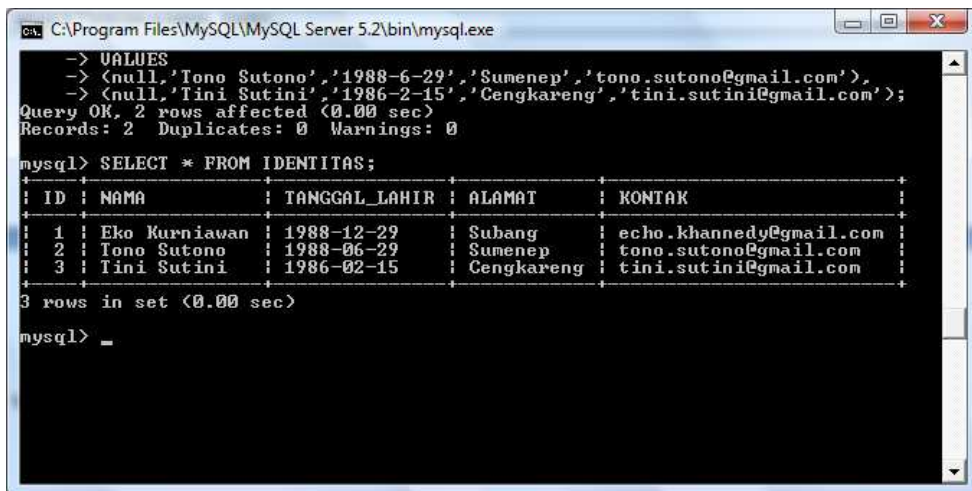
Untuk menampilkan data dalam sebuah database kita bisa menggunakan perintah :

```
SELECT
  [NAMA_ATRIBUTE, ...]
FROM
  [NAMA_TABEL, ...];
```

Namun jika kita akan menampilkan seluruh data yang ada dalam tabel kita bisa menggunakan perintah :

```
SELECT *
FROM
  [NAMA_TABEL, ...];
```

Misal kita akan menampilkan seluruh tabel IDENTITAS :



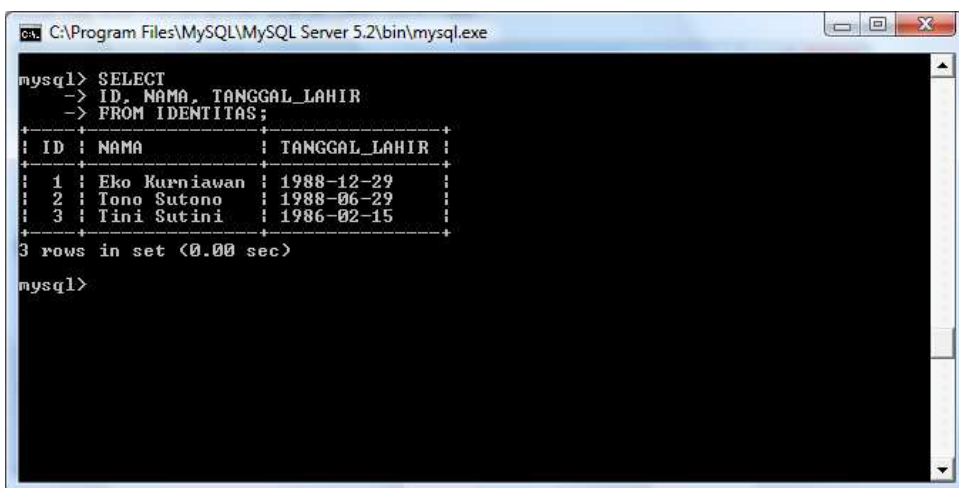
```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
-> VALUES
-> (null,'Tono Sutono','1988-6-29','Sumenep','tono.sutono@gmail.com'),
-> (null,'Tini Sutini','1986-2-15','Cengkareng','tini.sutini@gmail.com');
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM IDENTITAS;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Subang | echo.khannedy@gmail.com |
| 2  | Tono Sutono   | 1988-06-29    | Sumenep | tonosutono@gmail.com |
| 3  | Tini Sutini   | 1986-02-15    | Cengkareng | tini.sutini@gmail.com |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

Gambar 17 Menampilkan Seluruh Data Tabel IDENTITAS

Namun jika kita akan menampilkan atribut-atribut tertentu kita bisa menggunakan perintah pertama, misal kita hanya akan menampilkan atribut ID, NAMA dan TANGGAL_LAHIR :



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
mysql> SELECT
-> ID, NAMA, TANGGAL_LAHIR
-> FROM IDENTITAS;
+----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR |
+----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    |
| 2  | Tono Sutono   | 1988-06-29    |
| 3  | Tini Sutini   | 1986-02-15    |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Gambar 18 Menampilkan Kolom ID, NAMA, TANGGAL_LAHIR Dalam Tabel IDENTITAS

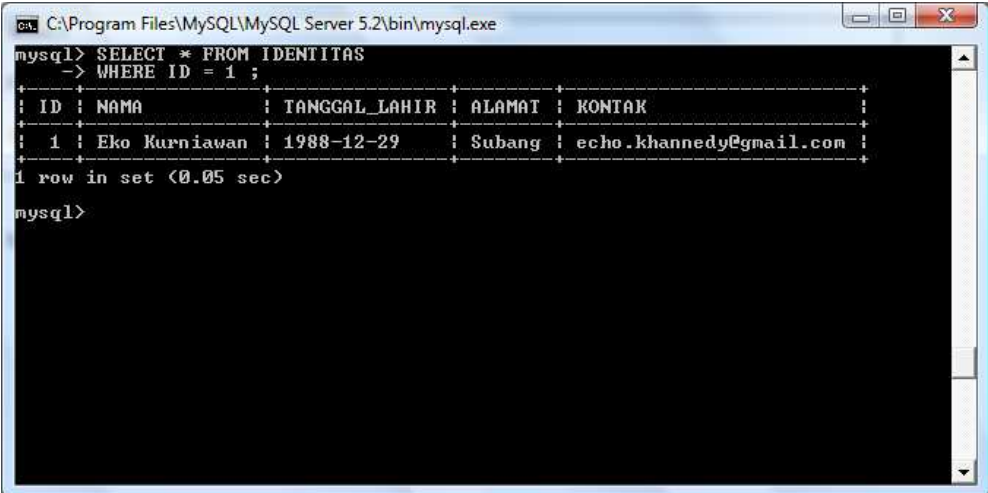
1.7 Penyaringan Data

Salah satu perintah yang sangat penting adalah perintah penyaringan menggunakan “WHERE”. Proses penyaringan sangatlah penting untuk proses UPDATE dan DELETEdelete, karena tanpa penyaringan proses UPDATE dan DELETE akan sangat berbahaya.

Untuk menyaring data kita bisa menggunakan perintah seperti dibawah ini :

```
SELECT
  [NAMA_ATRIBUT,...]
FROM
  [NAMA_TABEL, ...]
WHERE
  [KONDISI...]
```

Misal kita akan menampilkan data identitas yang memiliki ID = 3 :



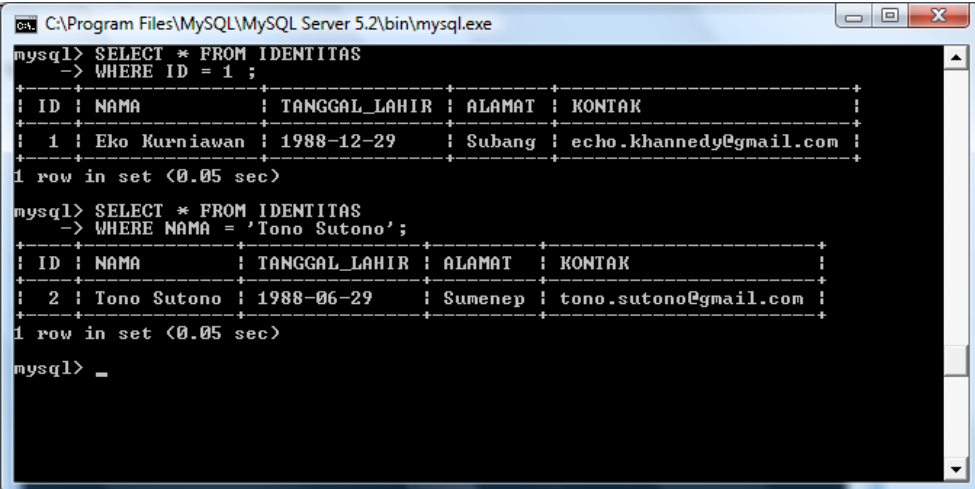
The screenshot shows a MySQL command prompt window with the following text:

```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
mysql> SELECT * FROM IDENTITAS
-> WHERE ID = 1 ;
+----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Subang | echo.khannedy@gmail.com |
+----+-----+-----+-----+
1 row in set (0.05 sec)

mysql>
```

Gambar 19 Menampilkan Data Dalam IDENTITAS Yang Memiliki ID = 1

Misal kita akan menampilkan data identitas yang memiliki NAMA = 'Tono Sutono' :



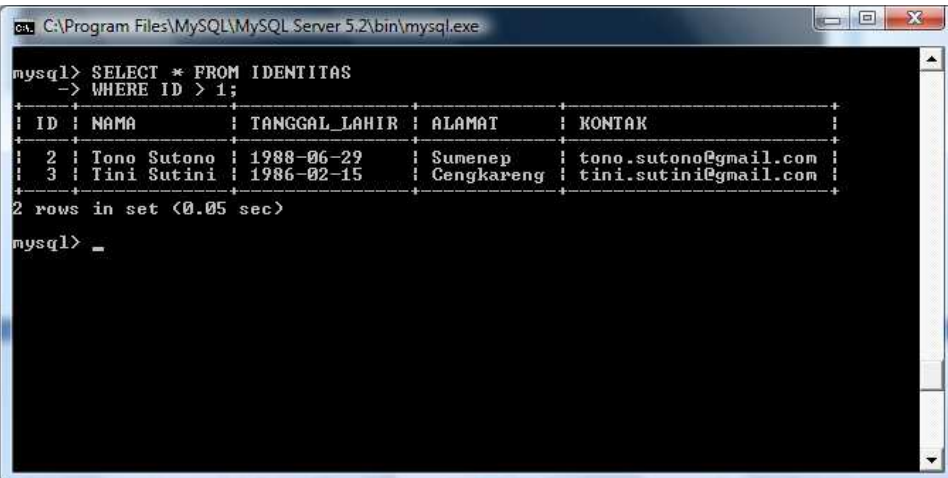
```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
mysql> SELECT * FROM IDENTITAS
-> WHERE ID = 1 ;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Subang | echo.khannedy@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.05 sec)

mysql> SELECT * FROM IDENTITAS
-> WHERE NAMA = 'Tono Sutono';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.05 sec)

mysql> _
```

Gambar 20 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki NAMA = 'Tono Sutono'

Selain menggunakan tanda “=” atau dikenal dengan EQUAL, kita juga bisa menggunakan tanda >, >=, <, != dan <= sebagai operasi dalam penyaringan. Misal kita akan menampilkan data yang memiliki ID > 1 :

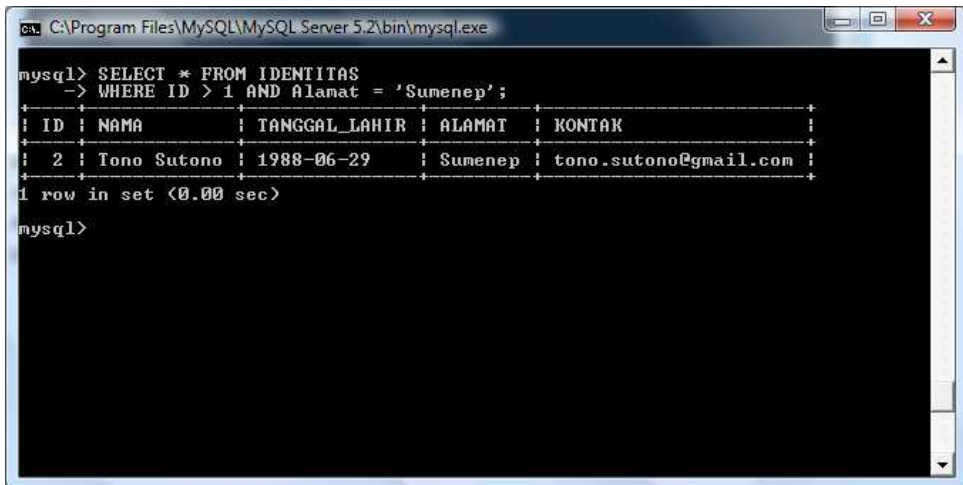


```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe
mysql> SELECT * FROM IDENTITAS
-> WHERE ID > 1;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
| 3  | Tini Sutini | 1986-02-15    | Gengkareng | tini.sutini@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.05 sec)

mysql> _
```

Gambar 21 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki ID > 1

Pada proses penyaringan diatas, kita hanya menggunakan satu kondisi, namun sebenarnya kita bisa menggunakan lebih dari satu kondisi, misal kita akan menampilkan data yang memiliki ID > 1 dan ALAMAT = 'Sumenep' dan untuk menggunakan lebih dari satu kondisi kita bisa menggunakan penghubung AND atau OR:

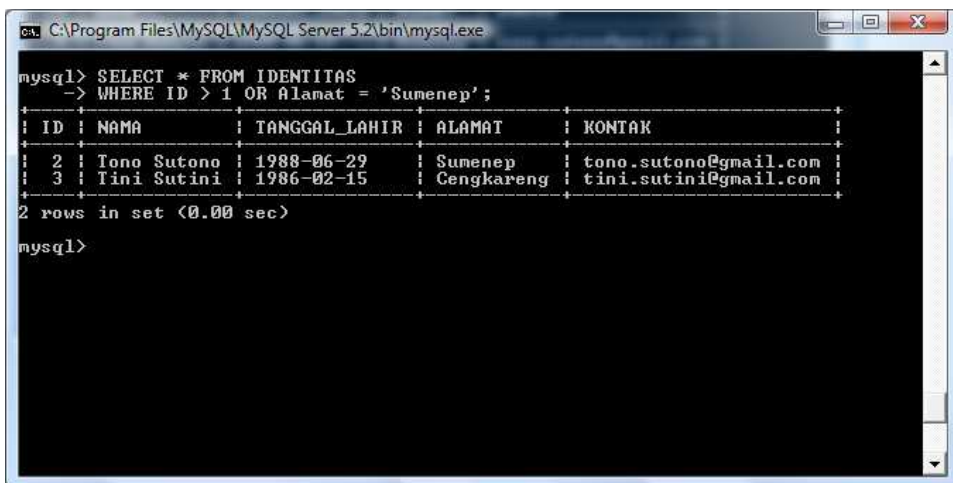


```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

mysql> SELECT * FROM IDENTITAS
-> WHERE ID > 1 AND Alamat = 'Sumenep';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | toni.sutono@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Gambar 22 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki ID > 1 Dan ALAMAT = 'Sumenep'



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

mysql> SELECT * FROM IDENTITAS
-> WHERE ID > 1 OR Alamat = 'Sumenep';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | toni.sutono@gmail.com |
| 3  | Tini Sutini | 1986-02-15    | Cengkareng | tini.sutini@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

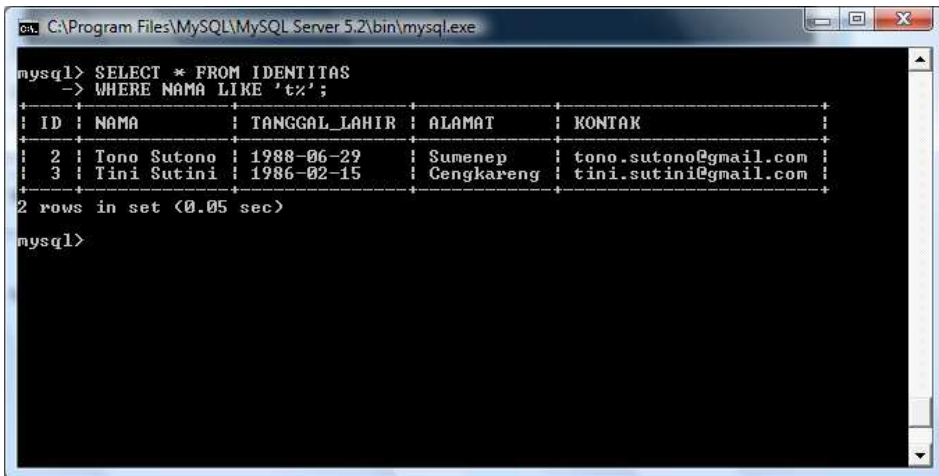
mysql>
```

Gambar 23 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki ID > 1 Atau ALAMAT = 'Sumenep'

Selain operator-operator diatas, kita juga bisa menggunakan operator regex seperti LIKE, untuk menggunakannya gunakan perintah :

```
...
WHERE
NAMA_ATRIBUTE LIKE 'REGEX';
```

Misal kita akan menampilkan data dengan NAMA yang memiliki awalan 'T' :

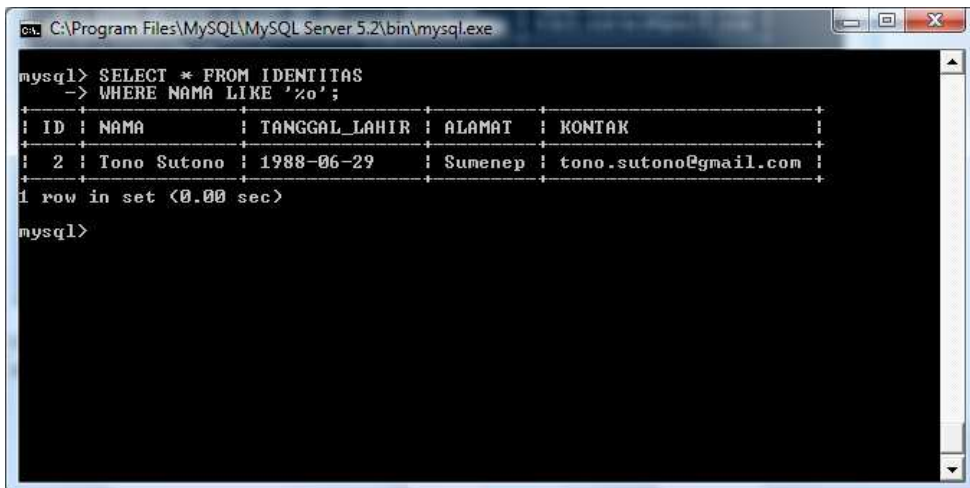


```
mysql> SELECT * FROM IDENTITAS
-> WHERE NAMA LIKE 't%';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT  | KONTAK                |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | toni.sutono@gmail.com |
| 3  | Tini Sutini | 1986-02-15    | Cengkareng | tini.sutini@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.05 sec)

mysql>
```

Gambar 24 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki NAMA Berawalkan 't'

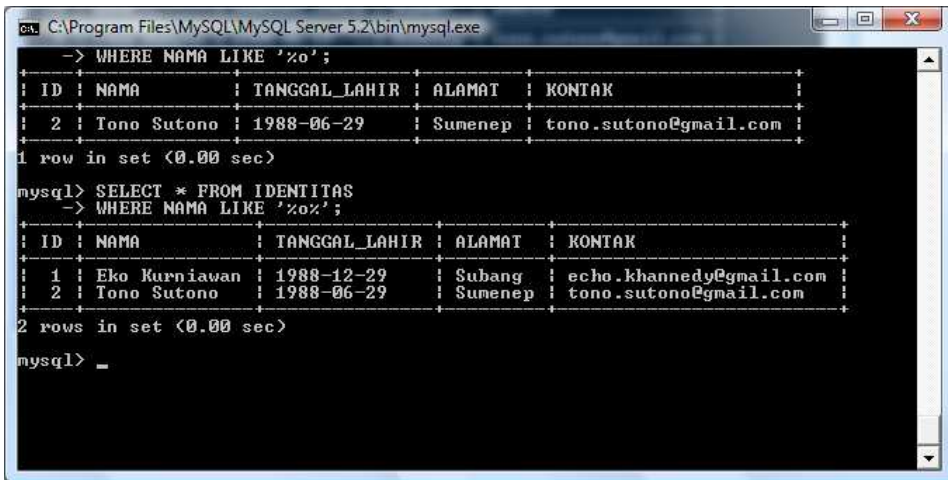
Untuk menampilkan data yang memiliki akhiran 'o' kita bisa menggunakan regex '%o' dan untuk menampilkan data yang mengandung 'o' kita bisa menggunakan regex '%o%' :



```
mysql> SELECT * FROM IDENTITAS
-> WHERE NAMA LIKE '%o';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT  | KONTAK                |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | toni.sutono@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Gambar 25 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki NAMA Berakhiran 'o'



```
C:\Program Files\MySQL\MySQL Server 5.2\bin>mysql.exe

mysql> -> WHERE NAMA LIKE '%o';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM IDENTITAS
mysql> -> WHERE NAMA LIKE '%oz';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Subang | echo.khannedy@gmail.com |
| 2  | Tono Sutono   | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

Gambar 26 Menampilkan Data Dalam Tabel IDENTITAS yang Memiliki NAMA Mengandung 'o'

1.8 Mengubah Data

Selain berdasarkan fakta, sebuah database haruslah memiliki sifat “Up2Date” artinya aktual alias informasi yang terkandung tidak basi. Oleh karena itu proses pengubahan atau lebih dikenal dengan istilah UPDATE sangatlah penting dalam database.

Untuk mengubah data dalam MySQL kita dapat menggunakan perintah :

```
UPDATE NAMA_TABEL
SET
  NAMA_ATRIBUT = “NILAI BARU”,
  ...
[WHERE KONDISI]
```

Misal kita akan mengubah ALAMAT dengan id 1 menjadi ‘Bandung’ :

```

C:\Program Files\MySQL\MySQL Server 5.2\bin\mysql.exe
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK      |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM IDENTITAS
-> WHERE NAMA LIKE '%o%';
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK      |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Subang  | echo.khannedy@gmail.com |
| 2  | Tono Sutono  | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> UPDATE IDENTITAS
-> SET
-> ALAMAT = 'Bandung'
-> WHERE
-> ID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql>

```

Gambar 27 Mengubah ALAMAT Yang memiliki ID = 1

Untuk memastikan bahwa data telah berubah, tampilkanlah data tersebut :

```

C:\Program Files\MySQL\MySQL Server 5.2\bin\mysql.exe
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK      |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Subang  | echo.khannedy@gmail.com |
| 2  | Tono Sutono  | 1988-06-29    | Sumenep | tono.sutono@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> UPDATE IDENTITAS
-> SET
-> ALAMAT = 'Bandung'
-> WHERE
-> ID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM IDENTITAS
-> WHERE ID = 1;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK      |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Bandung | echo.khannedy@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _

```

Gambar 28 Menampilkan Data Dalam Tabel IDENTITAS Yang Memiliki ID = 1

Perlu diingat, jika anda melakukan proses UPDATE tanpa melakukan pengkondisian dengan WHERE, maka anda sama saja mengubah seluruh data dalam tabel tersebut, misal kita akan mengubah seluruh ALAMAT dalam tabel IDENTITAS menjadi 'Jakarta' :

```

C:\Program Files\MySQL\MySQL Server 5.2\bin\mysql.exe
2 rows in set (0.00 sec)

mysql> UPDATE IDENTITAS
-> SET
-> ALAMAT = 'Bandung'
-> WHERE
-> ID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM IDENTITAS
-> WHERE ID = 1;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Bandung | echo.khannedy@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> UPDATE IDENTITAS
-> SET ALAMAT = 'Jakarta';
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql>

```

Gambar 29 Mengubah Seluruh ALAMAT Dalam Tabel IDENTITAS

Untuk membuktikan bahwa seluruh data dalam tabel IDENTITAS berubah, tampilkanlah seluruh data dalam tabel IDENTITAS :

```

C:\Program Files\MySQL\MySQL Server 5.2\bin\mysql.exe

mysql> SELECT * FROM IDENTITAS
-> WHERE ID = 1;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Bandung | echo.khannedy@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> UPDATE IDENTITAS
-> SET ALAMAT = 'Jakarta';
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT * FROM IDENTITAS;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK |
+----+-----+-----+-----+-----+
| 1  | Eko Kurniawan | 1988-12-29    | Jakarta | echo.khannedy@gmail.com |
| 2  | Tono Sutono   | 1988-06-29    | Jakarta | tono.sutono@gmail.com   |
| 3  | Tini Sutini   | 1986-02-15    | Jakarta | tini.sutini@gmail.com   |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

Gambar 30 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

1.9 Menghapus Data

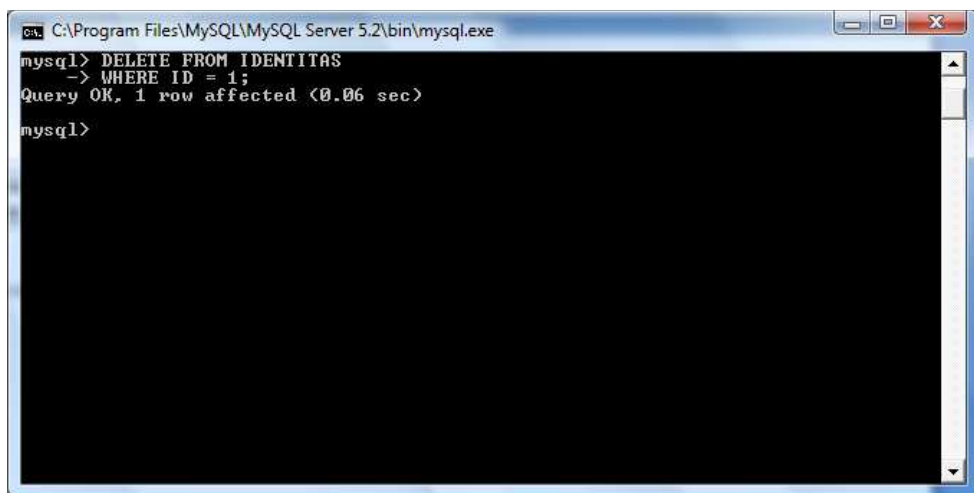
Selain proses INSERT, UPDATE dan SELECT, dalam database kita juga mengenal proses penghapusan data atau lebih dikenal dengan DELETE. Hal ini juga sangat diperlukan mengenal bahwa setiap data yang ada dalam sebuah database tidak

selalu harus ada, sehingga diperlukan proses DELETE untuk menghapus data tersebut.

Untuk melakukan proses penghapusan dalam MySQL, kita bisa menggunakan perintah :

```
DELETE FROM NAMA_TABEL  
[WHERE KONDISI]
```

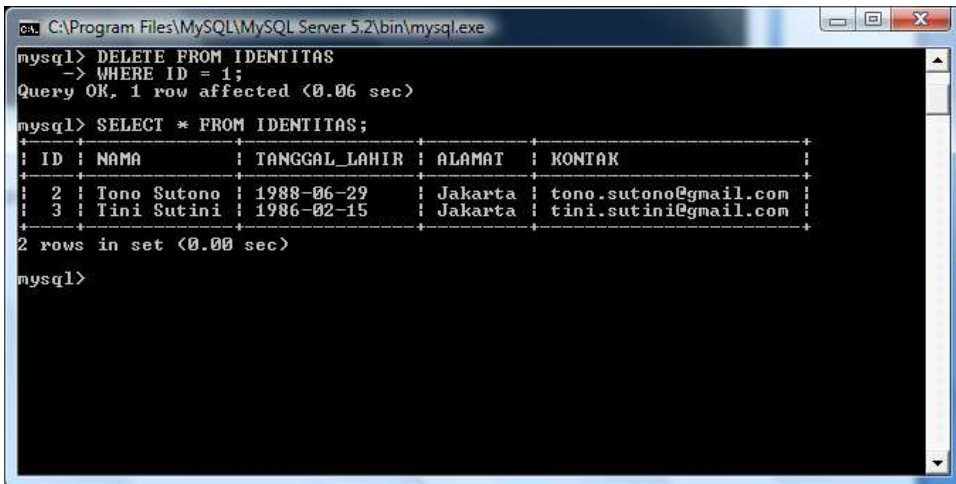
Misal kita akan menghapus data yang memiliki ID = 3 :

A screenshot of a MySQL command prompt window. The title bar shows the path 'C:\Program Files\MySQL\MySQL Server 5.2\bin\mysql.exe'. The command prompt shows the following text: 'mysql> DELETE FROM IDENTITAS', '-> WHERE ID = 1;', 'Query OK, 1 row affected (0.06 sec)', and 'mysql>'. The window has a standard Windows XP-style interface with a blue title bar and a scroll bar on the right.

```
C:\Program Files\MySQL\MySQL Server 5.2\bin\mysql.exe  
mysql> DELETE FROM IDENTITAS  
-> WHERE ID = 1;  
Query OK, 1 row affected (0.06 sec)  
mysql>
```

Gambar 31 Menghapus Data Dalam Tabel IDENTITAS Yang Memiliki ID = 3

Untuk memastikan bahwa data telah terhapus, tampilkan seluruh data dalam tabel IDENTITAS :



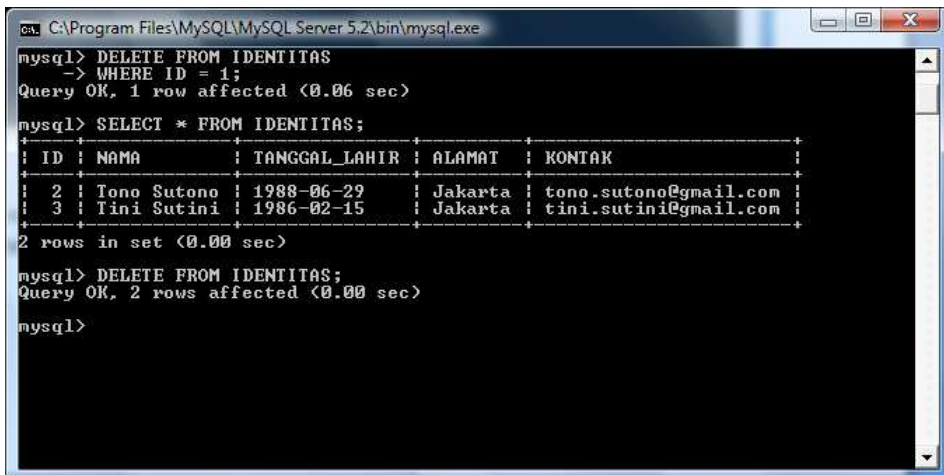
```
mysql> DELETE FROM IDENTITAS
-> WHERE ID = 1;
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM IDENTITAS;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK                |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Jakarta | tonosutono@gmail.com |
| 3  | Tini Sutini | 1986-02-15    | Jakarta | tini.sutini@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Gambar 32 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

Jika kita ingin menghapus seluruh data dalam tabel IDENTITAS kita bisa menggunakan perintah diatas tanpa menggunakan kondisi :



```
mysql> DELETE FROM IDENTITAS
-> WHERE ID = 1;
Query OK, 1 row affected (0.06 sec)

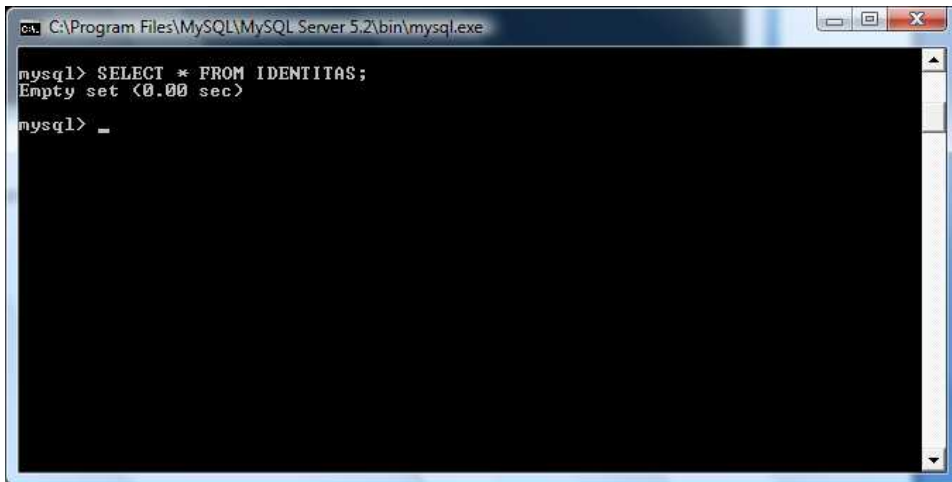
mysql> SELECT * FROM IDENTITAS;
+----+-----+-----+-----+-----+
| ID | NAMA      | TANGGAL_LAHIR | ALAMAT | KONTAK                |
+----+-----+-----+-----+-----+
| 2  | Tono Sutono | 1988-06-29    | Jakarta | tonosutono@gmail.com |
| 3  | Tini Sutini | 1986-02-15    | Jakarta | tini.sutini@gmail.com |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> DELETE FROM IDENTITAS;
Query OK, 2 rows affected (0.00 sec)

mysql>
```

Gambar 33 Menghapus Seluruh Data Dalam Tabel IDENTITAS

Untuk memastikan seluruh data telah terhapus, tampilkan data dalam tabel IDENTITAS :



```
mysql> SELECT * FROM IDENTITAS;  
Empty set (0.00 sec)  
  
mysql> _
```

Gambar 34 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

Empty menyatakan bahwa dalam tabel tersebut tak memiliki data / baris.

2 Java Database Connectivity

JDBC (Java Database Connectivity) merupakan library yang digunakan untuk mengkoneksikan DBMS dengan Java, baik itu MySQL, Oracle, Microsoft ODBC dan DBMS lainnya. Dan dalam buku ini kita menggunakan MySQL sebagai DBMSnya.

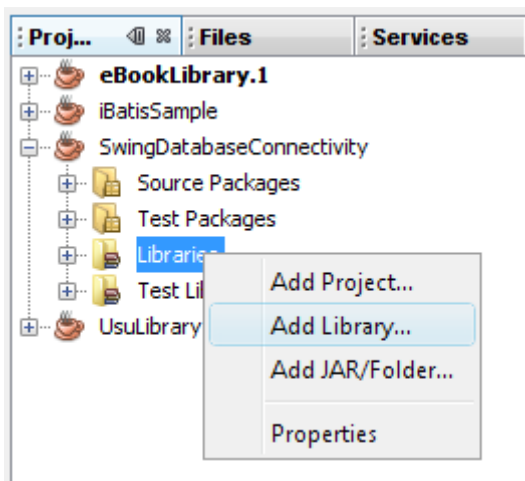
2.1 MySQL Connector Java

Untuk menghubungkan antara Java dan MySQL kita membutuhkan sebuah library yang bernama MySQLConnectorJava yang telah saya sediakan dalam CD. Jika anda akan membutuhkan connector yang baru anda bisa mendownloadnya di situs resmi Sun Java, <http://java.sun.com/>.

Untuk memudahkan proses pembuatan aplikasi-aplikasi java, sebaiknya kita menggunakan IDE untuk Java seperti NetBeans atau Eclipse yang sampai saat ini bersifat OpenSource dan Gratis. Dan sekarang kita akan mencoba menggunakan MySQLconnectorJava dalam dua IDE tersebut.

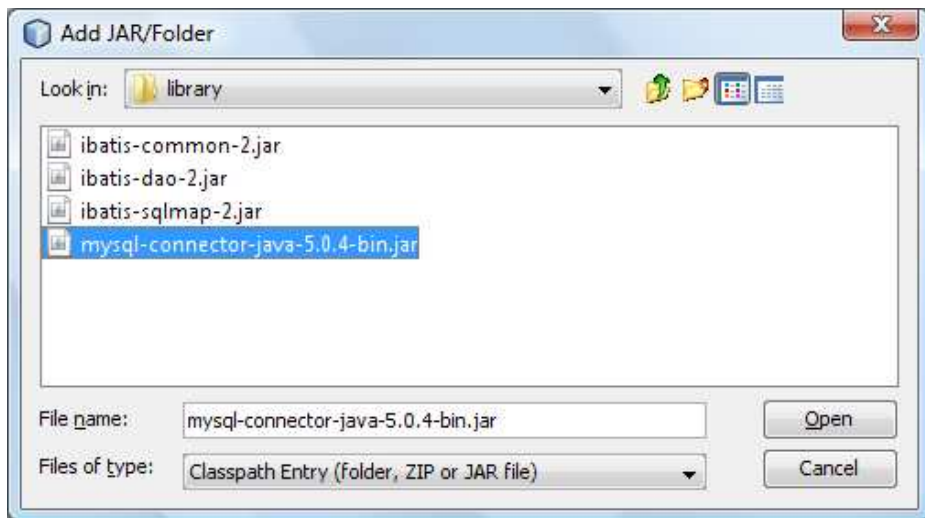
2.1.1 Netbeans

Untuk menggunakan / memasukkan library MySQLConnectorJava ke sebuah project, kita bisa memasukkannya lewat library dengan cara mengklik kanan paket library, lalu pilih Add Jar/Folder :



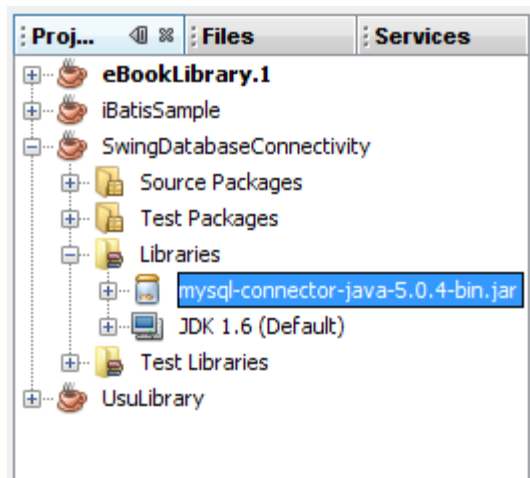
Gambar 35 Menambah Library MySQLConnectorJava Dalam NetBeans

Cari file MySQLConnectorJava :



Gambar 36 Add Jar/Folder

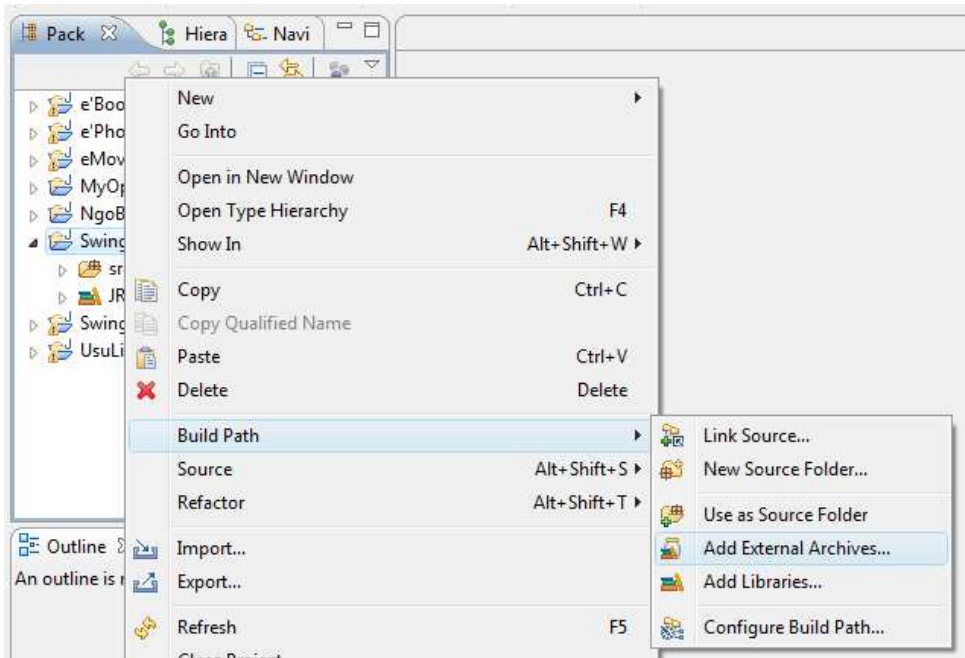
Setelah itu klik OPEN dan secara otomatis library MySQLConnectorJava akan terinclude dalam Project Netbeans :



Gambar 37 Libraries MySQLConnectorJava

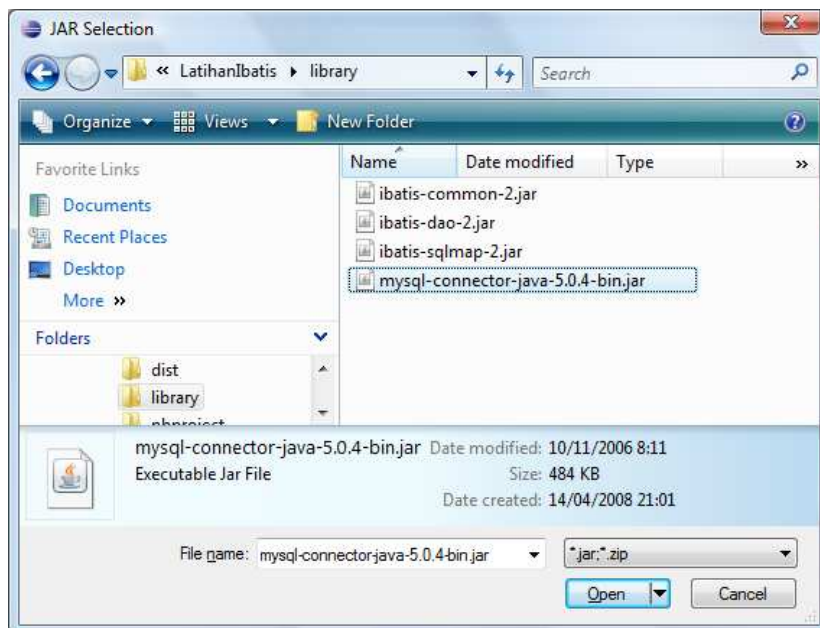
2.1.2 Eclipse

Untuk memasukkan MySQLConnectorJava dalam sebuah project Eclipse kita bisa mengklik kanan project tersebut lalu pilih Build Path > Add External Archives...



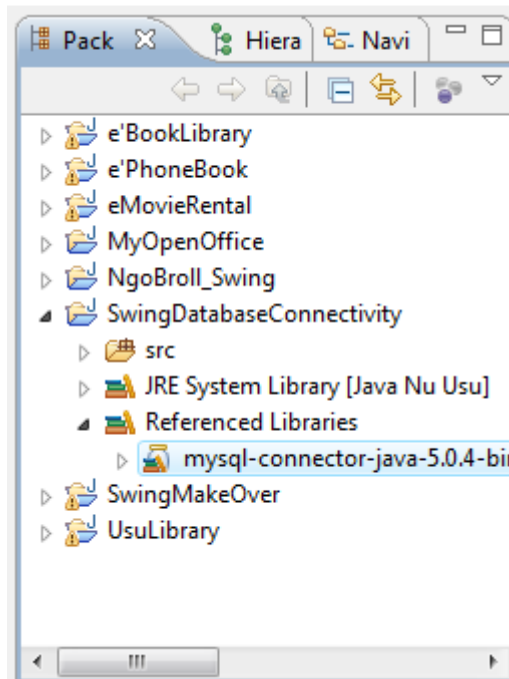
Gambar 38 Memasukkan Library MySQLConnectorJava dalam Eclipse

Pilih library MySQLConnectorJava :



Gambar 39 JAR Selection

Setelah itu klik OPEN untuk menambahkan file MySQLConnectorJava ke dalam project yang tadi anda pilih dan secara otomatis Eclipse akan memasukkan file tersebut kedalam Libraries :



Gambar 40 Libraries MySQLConnectorJava

2.2 Driver

Untuk membuat koneksi dari Java ke MySQL kita memerlukan class yang bernama Driver. Dan setiap connector (yang kita gunakan MySQLConnectorJava) memiliki Driver masing-masing yang selalu mengimplementasi class Driver yang ada dalam paket "java.sql". dan untuk mengecek keberadaan Driver dalam MySQLConnectorJava gunakan perintah seperti ini :

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

Atau lengkapnya seperti ini :

```
package usu.jdbc;  
  
import java.util.logging.Level;  
import java.util.logging.Logger;
```

```
/**
 * @author usu
 */
public class DriverTest {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();
            System.out.println("Proses Deteksi Driver
            Berhasil");
        } catch (final InstantiationException ex) {
        } catch (final IllegalAccessException ex) {
        } catch (final ClassNotFoundException ex) {
        }
    }
}
```

Java 1 DriverTest.java

Jika berhasil proses run akan menampilkan tulisan “Proses Deteksi Driver Berhasil”, namun jika terjadi kesalahan, dimungkinkan kesalahan terjadi jika class Driver tak ditemukan atau library MySQLConnectorJava belum terdapat dalam project yang anda buat.

2.3 Connection

Setelah mengenal Driver untuk menghubungkan Java dan MySQL kita harus membuat Connection yang terdapat dalam paket “java.sql”. Dan untuk membuat Connection, kita harus mengetahui host, port, dan nama database yang akan kita gunakan. Selain itu kita juga harus mengetahui username dan password yang digunakan untuk koneksi MySQL tersebut.

Untuk membuat Connection kita membutuhkan class `java.sql.DriverManager` yang digunakan sebagai class library yang berguna untuk membuat koneksi :

```
Connection koneksi = DriverManager.getConnection(
    "jdbc:mysql://HOST:PORT/DATABASE ", "username", "password");
```

Sekarang kita coba membuat class untuk mengetes koneksi database yang telah kita buat pada bab sebelumnya :

```
package usu.jdbc;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class ConnectionTest {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", "username", "password");
            System.out.println("Koneksi Berhasil");
        } catch (final SQLException ex) {
        } catch (final InstantiationException ex) {
        } catch (final IllegalAccessException ex) {
        } catch (final ClassNotFoundException ex) {
        }

    }
}
```

Java 2 ConnectionTest.java

Jika proses run berhasil, maka akan tampil tulisan “Koneksi Berhasil”, dan jika gagal dimungkinkan database yang dibuat tidak ditemukan atau username dan password yang digunakan salah.

2.4 Statement

Statement merupakan class yang terdapat dalam paket “java.sql” yang dapat digunakan sebagai pengeksekusi perintah-perintah SQL. Misal tadi kita mengenal proses INSERT, UPDATE, SELECT dan DELETE dalam MySQL, dan kita juga bisa menggunakan perintah-perintah tersebut lewat Statement. Untuk membuat Statement kita dapat menggunakan Connection :

```
Connection koneksi = DriverManager.getConnection(.....);
Statement statement = koneksi.createStatement();
```

Misal kita akan membuat Statement untuk database yang kita buat tadi :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class StatementTest {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", username, password);
            final Statement statement = koneksi.createStatement();

            System.out.println("Statement Berhasil");
        } catch (final SQLException ex) {
        } catch (final InstantiationException ex) {
        } catch (final IllegalAccessException ex) {
        } catch (final ClassNotFoundException ex) {
        }
    }
}
```

Java 3 StatementTest.java

Jika proses run berhasil maka akan menampilkan tulisan "Statement Berhasil".

2.4.1 Memasukkan Data

Seperti yang saya bilang tadi, lewat Statement kita bisa melakukan proses INSERT, UPDATE dan DELETE. Dan sekarang kita akan melakukan proses INSERT :

```
Connection koneksi = DriverManager.getConnection(...);
Statement statement = koneksi.createStatement();
statement.executeUpdate("INSERT, UPDATE, DELETE ...");
```


Misal kita akan menambah data ke dalam tabel IDENTITAS yang ada dalam database USU tadi yang telah kita buat :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class StatementInsert {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

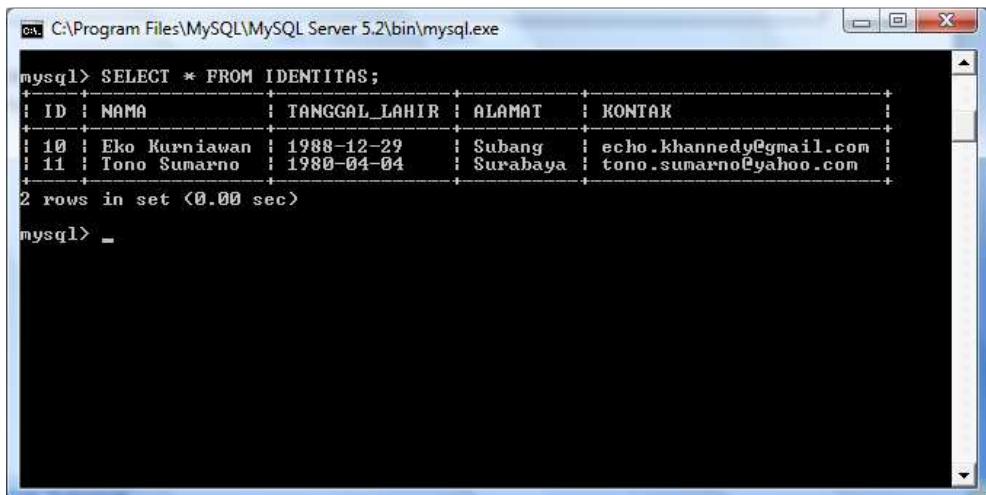
            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", username, password);
            final Statement statement = koneksi.createStatement();
            statement
                .executeUpdate(" INSERT INTO IDENTITAS "
                    + " (ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK) "
                    + " VALUES "
                    + " (10, 'Eko Kurniawan', '1988-12-29', 'Subang', 'echo.khannedy@gmail.com'), "
                    + " (11, 'Tono Sumarno', '1980-4-4', 'Surabaya', 'tono.sumarno@yahoo.com'); ");
            } catch (final SQLException ex) {
            } catch (final InstantiationException ex) {
            } catch (final IllegalAccessException ex) {
            } catch (final ClassNotFoundException ex) {
            }
        }
    }
}
```

Java 4 StatementInsert.java

StripBandunk | MySQL dan Java Database Connectivity

Setelah anda jalankan maka otomatis perintah tersebut akan dieksekusi ke dalam MySQL, dan untuk melihat hasil eksekusi, tampilkanlah isi data dalam tabel IDENTITAS :



Gambar 41 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.4.2 Mengubah Data

Setelah tadi kita memasukkan data lewat Statement, sekarang kita akan mencoba mengubah data yang ada dalam database lewat perintah UPDATE.

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class StatementUpdate {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
```

```

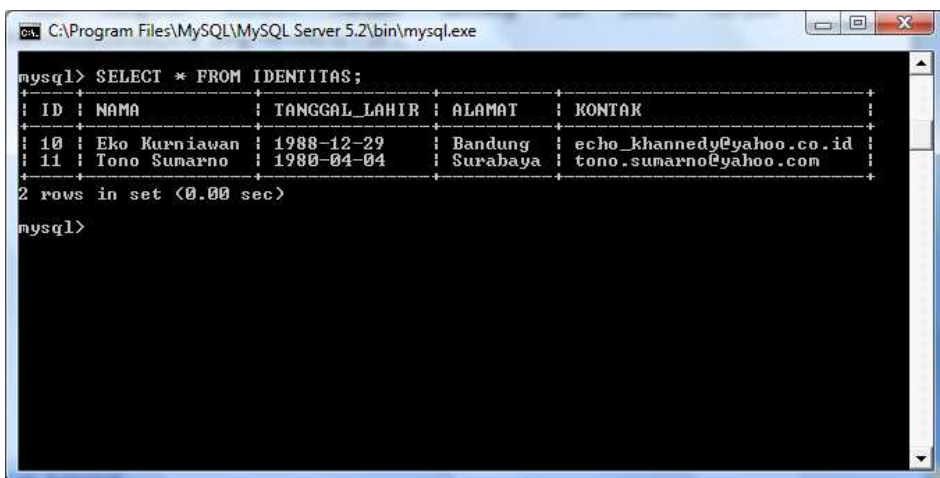
final String password = "rahasia";

final Connection koneksi = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/usu", username, password);
final Statement statement = koneksi.createStatement();
    statement
        .executeUpdate(" UPDATE IDENTITAS "
            + " SET "
            + " ALAMAT = 'Bandung', "
            + " KONTAK = 'echo_khannedy@yahoo.co.id' "
            + " WHERE ID = '10';");
} catch (final SQLException ex) {
} catch (final InstantiationException ex) {
} catch (final IllegalAccessException ex) {
} catch (final ClassNotFoundException ex) {
}
}
}

```

Java 5 StatementUpdate.java

Diatas saya mengubah ALAMAT menjadi 'Bandung' dan EMAIL menjadi 'echo_khannedy@yahoo.co.id' yang memiliki ID = 10 . Dan setelah kode diatas di run, maka anda bisa melihat perubahannya dalam database :



Gambar 42 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.4.3 Menghapus Data

Sekarang kita akan mengeksekusi perintah DELETE menggunakan Statement :

```
package usu.jdbc;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class StatementDelete {

    public static void main(final String[] args) {

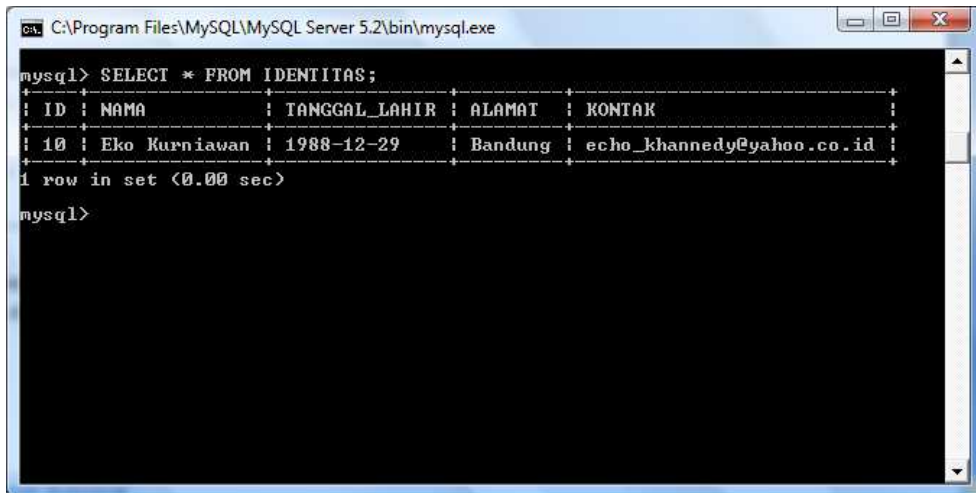
        Class.forName("com.mysql.jdbc.Driver").newInstance();

        final String username = "root";
        final String password = "rahasia";

        final Connection koneksi = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/usu", username, password);
        final Statement statement = koneksi.createStatement();
        statement.executeUpdate(" DELETE FROM IDENTITAS "
            + " WHERE ID = 11");
        } catch (final SQLException ex) {
        } catch (final InstantiationException ex) {
        } catch (final IllegalAccessException ex) {
        } catch (final ClassNotFoundException ex) {
        }
    }
}
```

Java 6 StatementDelete.java

Diatas saya menghapus data yang memiliki ID = 11, dan setelah di run, maka data yang memiliki ID = 11 akan dihapus dalam tabel IDENTITAS :



Gambar 43 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.5 ResultSet

Mungkin anda bertanya kenapa dalam proses Statement diatas saya tidak membahas perintah SELECT, yang notabenenya untuk menampilkan data, saya membutuhkan perintah SELECT. Hal ini karena SELECT merupakan perintah yang menghasilkan kumpulan data dalam tabel yang ada dalam database, sehingga untuk mendapatkannya kita membutuhkan bantuan class yang bernama ResultSet yang ada dalam paket java.sql.

Untuk membuat class ResultSet kita bisa menggunakan Statement :

```
Connection koneksi = DriverManager.getConnection(...);
Statement statement = koneksi.createStatement();
ResultSet result = statement.executeQuery("SELECT ...");
```

Sebelum kita menggunakan ResultSet, alangkan baiknya kita isi tabel IDENTITAS yang tadi kita buat dengan beberapa data, misal kita masukkan data ini :

ID	NAMA	TANGGAL LAHIR	ALAMAT	KONTAK
100	Budi	1982-5-17	Bandung	budi@gmail.com
101	Sumarno	1986-9-10	Surabaya	sumarno@gmail.com
102	Tukiyem	1978-10-10	Semarang	tukiyem@yahoo.com
103	Sumarni	1988-3-17	Surabaya	sumarni@hotmail.com
104	Rudi	1986-7-10	Bandung	rudi@gmail.com

105	Friko	1987-9-9	Bogor	friko@yahoo.com
106	Purwangga	1988-8-9	Subang	purwangga@yahoo.com
107	Joko	1987-10-10	Cirebon	joko@yahoo.com
108	Hisyam	1988-1-1	Depok	hisyam@gmail.com

Dan sekarang kita buat kode untuk memasukkan data diatas :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class ResultSetBefore {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", username, password);
            final Statement statement = koneksi.createStatement();
            statement.executeUpdate(" INSERT INTO IDENTITAS "
                + " (ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK) "
                + " VALUES "
                + " (100, 'Budi', '1982-5-17', 'Bandung', 'budi@gmail.com'), "
                + " (101, 'Sumarno', '1986-9-10', 'Surabaya', 'sumarno@gmail.com'), "
                + " (102, 'Tukiyem', '1978-10-10', 'Semarang', 'tukiyem@yahoo.com'), "
                + " (103, 'Sumarni', '1988-3-17', 'Surabaya', 'sumarni@hotmail.com'), "
                + " (104, 'Rudi', '1986-7-10', 'Bandung', 'rudi@gmail.com'), "
                + " (105, 'Friko', '1987-9-9', 'Bogor', 'friko@yahoo.com'), "
            );
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

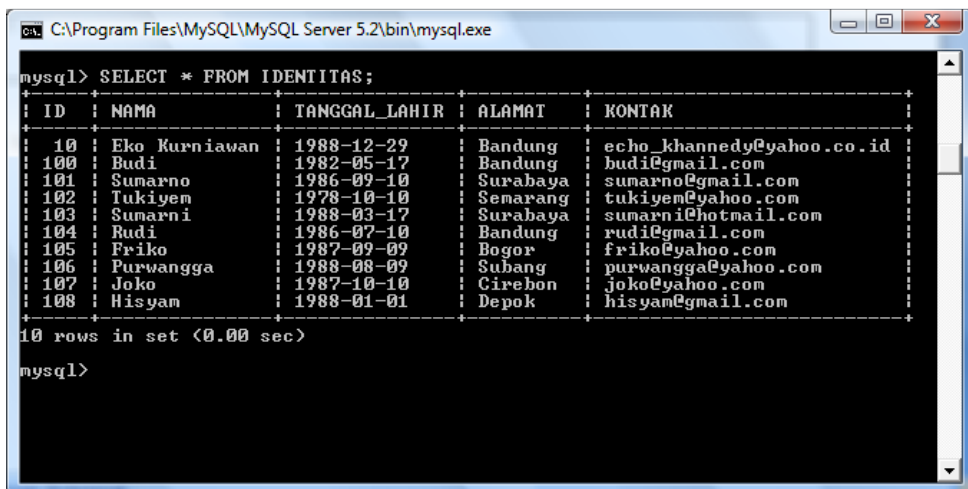
```

        + " (106,'Purwangga','1988-8-9','Subang','purwangga@yahoo.com'), "
        + " (107,'Joko','1987-10-10','Cirebon','joko@yahoo.com'), "
        + " (108,'Hisyam','1988-1-1','Depok','hisyam@gmail.com'); ";
    } catch (final SQLException ex) {
    } catch (final InstantiationException ex) {
    } catch (final IllegalAccessException ex) {
    } catch (final ClassNotFoundException ex) {
    }
}
}

```

Java 7 ResultSetBefore.java

Setelah kode diatas di run, maka anda akan mendapatkan tabel IDENTITAS berisikan dalam tabel diatas :



ID	NAMA	TANGGAL_LAHIR	ALAMAT	KONTAK
10	Eko Kurniawan	1988-12-29	Bandung	echo_khannedy@yahoo.co.id
100	Budi	1982-05-17	Bandung	budi@gmail.com
101	Sumarno	1986-09-10	Surabaya	sumarno@gmail.com
102	Tukiyem	1978-10-10	Semarang	tukiyem@yahoo.com
103	Sumarni	1988-03-17	Surabaya	sumarni@hotmail.com
104	Rudi	1986-07-10	Bandung	rudi@gmail.com
105	Friko	1987-09-09	Bogor	friko@yahoo.com
106	Purwangga	1988-08-09	Subang	purwangga@yahoo.com
107	Joko	1987-10-10	Cirebon	joko@yahoo.com
108	Hisyam	1988-01-01	Depok	hisyam@gmail.com

Gambar 44 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

Perlu anda ketahui untuk mendapatkan record dalam ResultSet kita bisa menggunakan metode next() untuk mengecek apakah ada record lagi setelah record ini dan prev() untuk mengecek apakah ada record sebelum record yang saat ini terseleksi.

```

Connection koneksi = DriverManager.getConnection(...);
Statement statement = koneksi.createStatement();
ResultSet result = statement.executeQuery("SELECT ...");
while (result.next()) {
    // Proses
}

```

```
}
```

Dan untuk mendapatkan data dalam atribut tersebut kita menggunakan metode `get[TypeData]("NAMA_ATRIBUT")` milik `ResultSet`, misal untuk mendapatkan data tabel IDENTITAS yang kita buat kita menggunakan :

```
int id = result.getInt("ID");
String nama = result.getString("NAMA");
Date tanggalLahir = result.getDate("TANGGAL_LAHIR");
String alamat = result.getString("ALAMAT");
String kontak = result.getString("KONTAK");
```

Dari kode diatas kita bisa tahu kali kita akan mengisi nilai `int id` dengan data yang ada dalam atribut ID, dan kita akan mengisi nilai `String nama` dengan data yang ada dalam atribut NAMA, dan selanjutnya.

Sekarang kita coba untuk membuat kode yang akan menampilkan seluruh data dalam Tabel IDENTITAS :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class ResultSetIdentitas {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
```



```
"jdbc:mysql://localhost:3306/usu", username, password);
final Statement statement = koneksi.createStatement();
final ResultSet result = statement
    .executeQuery("SELECT * FROM IDENTITAS");
while (result.next()) {
    // Proses
    final Integer id = result.getInt("ID");
    final String nama = result.getString("NAMA");
    final Date tanggalLahir = result.getDate("TANGGAL_LAHIR");
    final String alamat = result.getString("ALAMAT");
    final String kontak = result.getString("KONTAK");
    System.out.println("ID : " + id);
    System.out.println("NAMA : " + nama);
    System.out.println("TANGGAL LAHIR : " +
tanggalLahir);
    System.out.println("ALAMAT : " + alamat);
    System.out.println("KONTAK : " + kontak);
    System.out.println("-----
-");
    }
    } catch (final SQLException ex) {
    } catch (final InstantiationException ex) {
    } catch (final IllegalAccessException ex) {
    } catch (final ClassNotFoundException ex) {
    }
    }
}
```

Java 8 ResultSetIdentitas.java

Setelah anda run, anda akan mendapat tampilan seperti dibawah ini :

```
ID : 10
NAMA : Eko Kurniawan
TANGGAL LAHIR : 1988-12-29
ALAMAT : Bandung
KONTAK : echo_khannedy@yahoo.co.id
-----
ID : 100
NAMA : Budi
TANGGAL LAHIR : 1982-05-17
ALAMAT : Bandung
KONTAK : budi@gmail.com
-----
ID : 101
NAMA : Sumarno
TANGGAL LAHIR : 1986-09-10
ALAMAT : Surabaya
KONTAK : sumarno@gmail.com
-----
```

```
ID : 102
NAMA : TukiyeM
TANGGAL LAHIR : 1978-10-10
ALAMAT : Semarang
KONTAK : tukiyeM@yahoo.com
-----
ID : 103
NAMA : Sumarni
TANGGAL LAHIR : 1988-03-17
ALAMAT : Surabaya
KONTAK : sumarni@hotmail.com
-----
ID : 104
NAMA : Rudi
TANGGAL LAHIR : 1986-07-10
ALAMAT : Bandung
KONTAK : rudi@gmail.com
-----
ID : 105
NAMA : Friko
TANGGAL LAHIR : 1987-09-09
ALAMAT : Bogor
KONTAK : friko@yahoo.com
-----
ID : 106
NAMA : Purwangga
TANGGAL LAHIR : 1988-08-09
ALAMAT : Subang
KONTAK : purwangga@yahoo.com
-----
ID : 107
NAMA : Joko
TANGGAL LAHIR : 1987-10-10
ALAMAT : Cirebon
KONTAK : joko@yahoo.com
-----
ID : 108
NAMA : Hisyam
TANGGAL LAHIR : 1988-01-01
ALAMAT : Depok
KONTAK : hisyam@gmail.com
-----
```

Contoh lain? OK sekarang kita akan menampilkan seluruh data yang memiliki email di yahoo.com :

```
package usu.jdbc;
```

```

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class ResultSetIdentitasYahoo {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", username, password);
            final Statement statement = koneksi.createStatement();
            final ResultSet result = statement
                .executeQuery("SELECT * FROM IDENTITAS"
                    + " WHERE KONTAK LIKE '%@yahoo.com'");
            while (result.next()) {
                // Proses
                final Integer id = result.getInt("ID");
                final String nama = result.getString("NAMA");
                final Date tanggalLahir = result.getDate("TANGGAL_LAHIR");
                final String alamat = result.getString("ALAMAT");
                final String kontak = result.getString("KONTAK");
                System.out.println("ID : " + id);
                System.out.println("NAMA : " + nama);
                System.out.println("TANGGAL LAHIR : " +
                    tanggalLahir);
                System.out.println("ALAMAT : " + alamat);
                System.out.println("KONTAK : " + kontak);
                System.out.println("-----");
            }
        } catch (final SQLException ex) {
        } catch (final InstantiationException ex) {
        } catch (final IllegalAccessException ex) {
        } catch (final ClassNotFoundException ex) {
        }
    }
}

```

```
}  
}  
}
```

Java 9 ResultSetYahoo.java

Setelah kode diatas diRun, maka anda akan mendapat tampilan seperti ini :

```
ID : 102  
NAMA : Tukiye  
TANGGAL LAHIR : 1978-10-10  
ALAMAT : Semarang  
KONTAK : tukiye@yahoo.com  
-----  
ID : 105  
NAMA : Friko  
TANGGAL LAHIR : 1987-09-09  
ALAMAT : Bogor  
KONTAK : friko@yahoo.com  
-----  
ID : 106  
NAMA : Purwangga  
TANGGAL LAHIR : 1988-08-09  
ALAMAT : Subang  
KONTAK : purwangga@yahoo.com  
-----  
ID : 107  
NAMA : Joko  
TANGGAL LAHIR : 1987-10-10  
ALAMAT : Cirebon  
KONTAK : joko@yahoo.com  
-----
```

2.6 PreparedStatement

Selain Statement dalam JDBC kita juga mengenal PreparedStatement yang gunanya hampir sama dengan Statement, namun perbedaannya PreparedStatement memiliki fasilitas untuk mempermudah proses INSERT, UPDATE, DELETE. Untuk membuat PreparedStatement kita bisa membuatnya lewat Connection :

```
Connection koneksi = DriverManager.getConnection(...);  
PreparedStatement prepare =  
koneksi.prepareStatement("PERINTAH SQL...");
```

Berbeda dengan Statement, pada PreparedStatement kita harus menuliskan perintah SQL dengan '?' jika kita akan mengubah data tersebut :

```
Connection koneksi = DriverManager.getConnection(...);
PreparedStatement prepare = koneksi
    .prepareStatement(" INSERT INTO IDENTITAS "
        + " (ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK) "
        + " VALUES " + " (?, ?, ?, ?, ?)");
```

Setelah itu kita bisa mengubahnya dengan menset tanda '?' dalam perintah tersebut sesuai dengan tipe datanya menggunakan metode set[TipeData](int index, TipeData nilai). Dan setelah itu gunakan metode executeUpdate() untuk mengeksekusi seluruh perintahnya.

2.6.1 Memasukkan Data

Sekarang kita akan mencoba menambah data kedalam tabel IDENTITAS menggunakan PreparedStatement :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class PreparedStatementInsert {

    public static void main(final String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", username, password);
```

```
final PreparedStatement prepare = koneksi
    .prepareStatement(" INSERT INTO IDENTITAS "
        + " (ID, NAMA, TANGGAL_LAHIR, ALAMAT, KONTAK) "
        + " VALUES " + " (?, ?, ?, ?, ?)");

prepare.setInt(1, 200);
prepare.setString(2, "Usu");
final Calendar cal1 = Calendar.getInstance();
cal1.set(1980, Calendar.DECEMBER, 14);
prepare.setDate(3, new
Date(cal1.getTimeInMillis()));
prepare.setString(4, "Subang");
prepare.setString(5, "usu@gmail.com");
prepare.executeUpdate();

prepare.setInt(1, 201);
prepare.setString(2, "Nesia");
final Calendar cal2 = Calendar.getInstance();
cal2.set(1910, Calendar.OCTOBER, 4);
prepare.setDate(3, new
Date(cal2.getTimeInMillis()));
prepare.setString(4, "Purwakarta");
prepare.setString(5, "nesia@hotmail.com");
prepare.executeUpdate();

} catch (final SQLException ex) {
} catch (final InstantiationException ex) {
} catch (final IllegalAccessException ex) {
} catch (final ClassNotFoundException ex) {
}
}
```

Java 10 PreparedStatementInsert.java

Setelah kode diatas di run, maka anda bisa melihat perubahannya dalam tabel IDENTITAS :

101	Sumarno	1986-09-10	Surabaya	sumarno@gmail.com
102	Tukiye	1978-10-10	Semarang	tukiye@yahoo.com
103	Sumarni	1988-03-17	Surabaya	sumarni@hotmail.com
104	Rudi	1986-07-10	Bandung	rudi@gmail.com
105	Friko	1987-09-09	Bogor	friko@yahoo.com
106	Purwangga	1988-08-09	Subang	purwangga@yahoo.com
107	Joko	1987-10-10	Cirebon	joko@yahoo.com
108	Hisyam	1988-01-01	Depok	hisyam@gmail.com
200	Usu	1980-12-14	Subang	usu@gmail.com
201	Nesia	1910-10-04	Purwakarta	nesia@hotmail.com

Gambar 45 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.6.2 Mengubah Data

Setelah tadi kita mencoba membuat proses INSERT menggunakan PreparedStatement, sekarang kita akan mencoba membuat proses UPDATE menggunakan PreparedStatement :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class PreparedStatementUpdate {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";

            final Connection koneksi = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/usu", username, password);
```

```

final PreparedStatement prepare = koneksi
    .prepareStatement(" UPDATE IDENTITAS " + " SET
ID = ? "
                    + " WHERE ID = ? ");

prepare.setInt(1, 70);
prepare.setInt(2, 200);
prepare.executeUpdate();

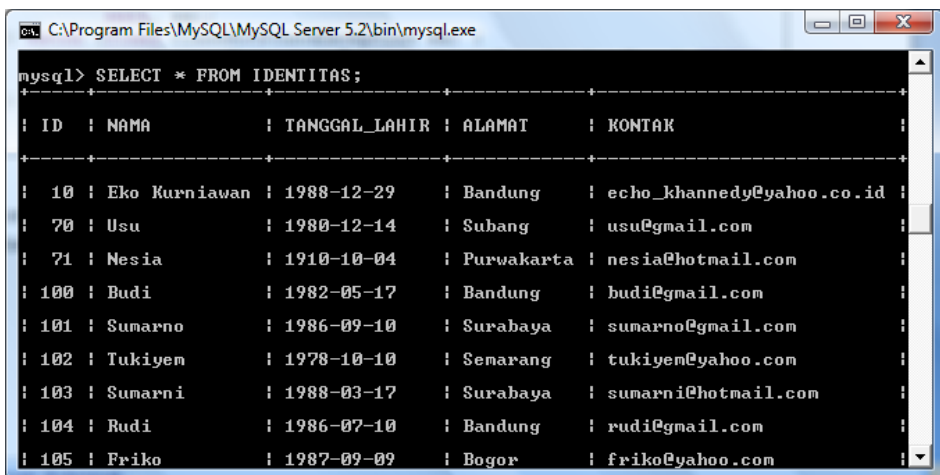
prepare.setInt(1, 71);
prepare.setInt(2, 201);
prepare.executeUpdate();

} catch (final SQLException ex) {
} catch (final InstantiationException ex) {
} catch (final IllegalAccessException ex) {
} catch (final ClassNotFoundException ex) {
}
}
}

```

Java 11 PreparedStatementUpdate.java

Setelah anda menjalankan kode diatas, maka anda akan menemukan bahwa record yang memiliki ID = 200 telah diganti menjadi 70 dan record yang memiliki ID = 201 telah diganti menjadi 71 :



ID	NAMA	TANGGAL_LAHIR	ALAMAT	KONTAK
10	Eko Kurniawan	1988-12-29	Bandung	echo_khannedy@yahoo.co.id
70	Usu	1980-12-14	Subang	usu@gmail.com
71	Nesia	1910-10-04	Purwakarta	nesia@hotmail.com
100	Budi	1982-05-17	Bandung	budi@gmail.com
101	Sumarno	1986-09-10	Surabaya	sumarno@gmail.com
102	Tukiye	1978-10-10	Senarang	tukiye@yahoo.com
103	Sumarni	1988-03-17	Surabaya	sumarni@hotmail.com
104	Rudi	1986-07-10	Bandung	rudi@gmail.com
105	Friko	1987-09-09	Bogor	friko@yahoo.com

Gambar 46 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.6.3 Menghapus Data

Sekarang kita akan mencoba membuat perintah DELETE menggunakan PreparedStatement :


```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class PreparedStatementDelete {

    public static void main(final String[] args) {

        Class.forName("com.mysql.jdbc.Driver").newInstance();

        final String username = "root";
        final String password = "rahasia";

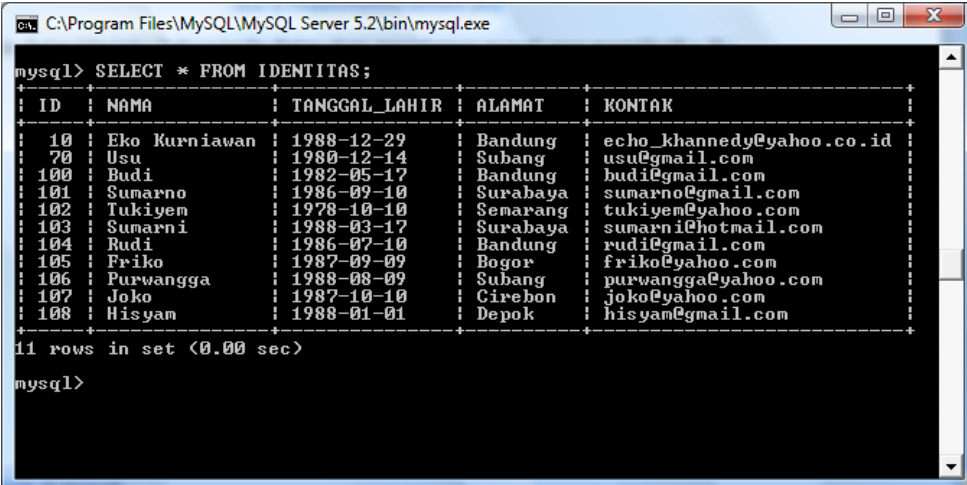
        final Connection koneksi = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/usu", username, password);
        final PreparedStatement prepare = koneksi
            .prepareStatement(" DELETE FROM IDENTITAS "
                + " WHERE ID = ? ");

        prepare.setInt(1, 71);
        prepare.executeUpdate();

        } catch (final SQLException ex) {
        } catch (final InstantiationException ex) {
        } catch (final IllegalAccessException ex) {
        } catch (final ClassNotFoundException ex) {
        }
    }
}
```

Java 12 PreparedStatementDelete.java

Dari kode diatas kita tahu bahwa kode diatas akan menghapus record yang memiliki ID = 71 :



```
mysql> SELECT * FROM IDENTITAS;
```

ID	NAMA	TANGGAL_LAHIR	ALAMAT	KONTAK
10	Eko Kurniawan	1988-12-29	Bandung	echo_khannedy@yahoo.co.id
70	Usu	1980-12-14	Subang	usu@gmail.com
100	Budi	1982-05-17	Bandung	budi@gmail.com
101	Sumarno	1986-09-10	Surabaya	sumarno@gmail.com
102	Tukiyem	1978-10-10	Semarang	tukiyem@yahoo.com
103	Sumarni	1988-03-17	Surabaya	sumarni@hotmail.com
104	Rudi	1986-07-10	Bandung	rudi@gmail.com
105	Friko	1987-09-09	Bogor	friko@yahoo.com
106	Purwangga	1988-08-09	Subang	purwangga@yahoo.com
107	Joko	1987-10-10	Cirebon	joko@yahoo.com
108	Hisyam	1988-01-01	Depok	hisyam@gmail.com

```
11 rows in set (0.00 sec)

mysql>
```

Gambar 47 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.6.4 Mendapatkan Data

Selain proses INSERT, UPDATE, DELETE kita juga bisa dapat menggunakan perintah SELECT yang dapat mengembalikan ResultSet. Misal kita akan mendapatkan seluruh data yang ada dalam tabel IDENTITAS :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class PrepareStatementResult {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";
```

```
final Connection koneksi = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/usu", username, password);
final PreparedStatement prepare = koneksi
    .prepareStatement("SELECT * FROM IDENTITAS");
final ResultSet result = prepare.executeQuery();
while (result.next()) {
    final Integer id = result.getInt("ID");
    final String nama = result.getString("NAMA");
    final Date tanggalLahir = result.getDate("TANGGAL_LAHIR");
    final String alamat = result.getString("ALAMAT");
    final String kontak = result.getString("KONTAK");
    System.out.println("ID : " + id);
    System.out.println("NAMA : " + nama);
    System.out.println("TANGGAL LAHIR : " +
    tanggalLahir);
    System.out.println("ALAMAT : " + alamat);
    System.out.println("KONTAK : " + kontak);
    System.out.println("-----");
}
} catch (final SQLException ex) {
} catch (final InstantiationException ex) {
} catch (final IllegalAccessException ex) {
} catch (final ClassNotFoundException ex) {
}
}
```

Java 13 PreparedStatementResult.java

Setelah anda menjalankan, maka akan muncul tulisan seperti ini :

```
ID : 10
NAMA : Eko Kurniawan
TANGGAL LAHIR : 1988-12-29
ALAMAT : Bandung
KONTAK : echo_khannedy@yahoo.co.id
-----
ID : 70
NAMA : Usu
TANGGAL LAHIR : 1980-12-14
ALAMAT : Subang
KONTAK : usu@gmail.com
-----
ID : 100
NAMA : Budi
TANGGAL LAHIR : 1982-05-17
ALAMAT : Bandung
KONTAK : budi@gmail.com
-----
```

```
ID : 101
NAMA : Sumarno
TANGGAL LAHIR : 1986-09-10
ALAMAT : Surabaya
KONTAK : sumarno@gmail.com
-----
ID : 102
NAMA : Tukiyem
TANGGAL LAHIR : 1978-10-10
ALAMAT : Semarang
KONTAK : tukiyem@yahoo.com
-----
ID : 103
NAMA : Sumarni
TANGGAL LAHIR : 1988-03-17
ALAMAT : Surabaya
KONTAK : sumarni@hotmail.com
-----
ID : 104
NAMA : Rudi
TANGGAL LAHIR : 1986-07-10
ALAMAT : Bandung
KONTAK : rudi@gmail.com
-----
ID : 105
NAMA : Friko
TANGGAL LAHIR : 1987-09-09
ALAMAT : Bogor
KONTAK : friko@yahoo.com
-----
ID : 106
NAMA : Purwangga
TANGGAL LAHIR : 1988-08-09
ALAMAT : Subang
KONTAK : purwangga@yahoo.com
-----
ID : 107
NAMA : Joko
TANGGAL LAHIR : 1987-10-10
ALAMAT : Cirebon
KONTAK : joko@yahoo.com
-----
ID : 108
NAMA : Hisyam
TANGGAL LAHIR : 1988-01-01
ALAMAT : Depok
KONTAK : hisyam@gmail.com
-----
```

2.7 Advanced ResultSet

Tadi kita telah membuat ResultSet, dan sekarang kita akan mengetahui apa saja yang dapat dilakukan oleh ResultSet selain menampilkan data.

2.7.1 Menambah Data

Selain Statement dan PreparedStatement, ternyata ResultSet juga bisa kita gunakan untuk menambah data, namun untuk membuat sebuah ResultSet yang dapat mengubah data kita harus membuat Statement yang berbeda dari Biasanya :

```
Connection koneksi = DriverManager.getConnection(...);
Statement statement = koneksi.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
ResultSet result = statement.executeQuery("SELECT ...");
```

Dan sebelum menambahkan data kita perlu memanggil metode moveToInsertRow(), dan untuk menambah atau mengubah data kita bisa menggunakan metode update[TipeData](int indexAtribut, TipeData nilai) dan setelah proses berakhir gunakan metode insertRow() untuk memasukkan data yang tadi anda masukkan, misal :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class AdvancedResultSetInsert {

    public static void main(final String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
final String username = "root";
final String password = "rahasia";

final Connection koneksi = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/usu", username, password);
final Statement statement = koneksi.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
final ResultSet result = statement.executeQuery("SELECT "
    + "ID, NAMA, TANGGAL_LAHIR, " + "ALAMAT, "
    + "KONTAK "
    + " FROM IDENTITAS");

    result.moveToInsertRow();

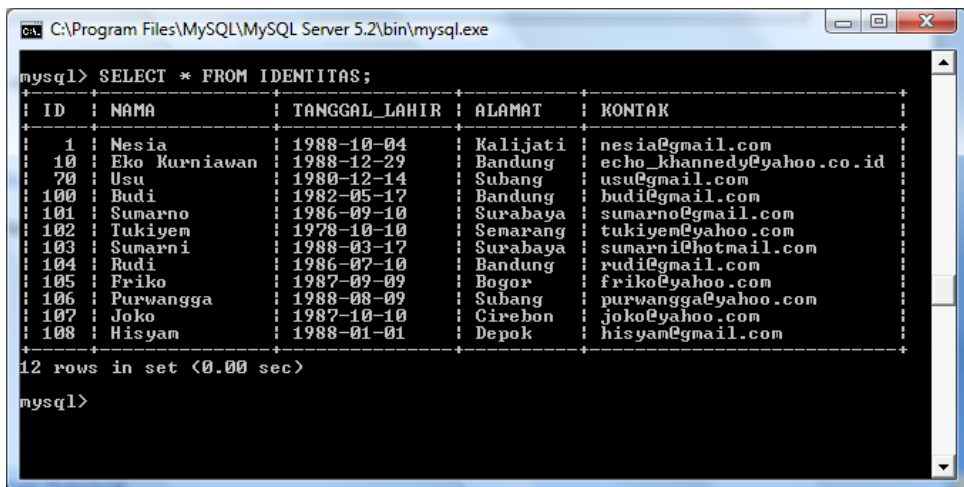
    result.updateInt(1, 1);
    result.updateString(2, "Nesia");
final Calendar cal = Calendar.getInstance();
    cal.set(1988, Calendar.OCTOBER, 4);
    result.updateDate(3, new
    Date(cal.getTimeInMillis()));
    result.updateString(4, "Kalijati");
    result.updateString(5, "nesia@gmail.com");

    result.insertRow();

    } catch (final SQLException ex) {
    } catch (final InstantiationException ex) {
    } catch (final IllegalAccessException ex) {
    } catch (final ClassNotFoundException ex) {
    }
    }
}
```

Java 14 AdvancedResultSetInsert.java

Setelah menjalankan kode diatas, anda bisa melihat perubahannya dalam tabel IDENTITAS :



Gambar 48 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.7.2 Mengubah Data

Sekarang kita akan mengubah data menggunakan ResultSet. Dan berbeda dengan Statement dan PreparedStatement, dalam ResultSet, kita harus tahu lokasi baris / record yang akan kita ubah. Sehingga proses ini agak sulit. Tapi hal ini sangat berguna ketika anda menggunakan TableModel yang nanti akan kita bahas dalam bagian SwingmakeOver.

Untuk mengubah data kita harus menentukan dulu record yang akan kita ubah dengan metode `absolute(int indexRecord)` milik ResultSet. Dan seperti proses INSERT untuk proses UPDATE kita juga menggunakan metode `update[TipeData](int indexAtribut, TipeData nilai)` dan harus diakhiri dengan metode `updateRow()`, misal :

```
package usu.jdbc;

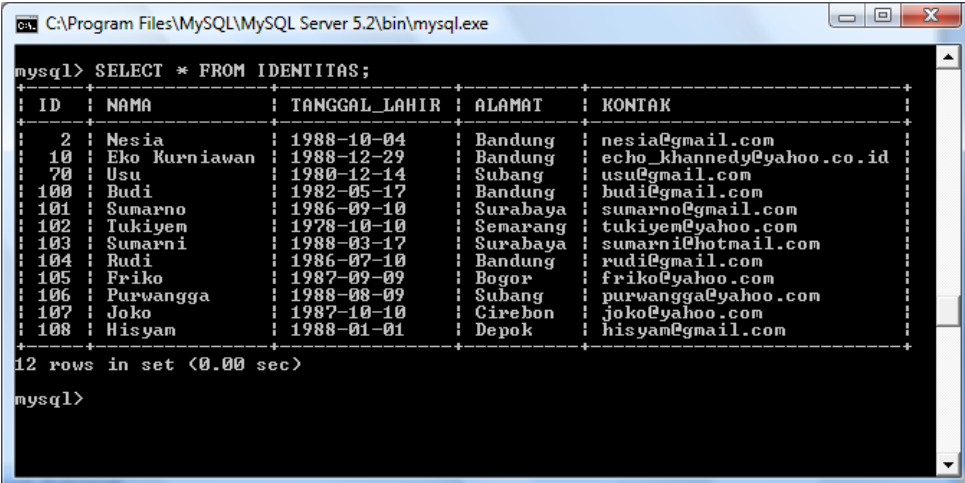
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
```

```
*/  
publicclass AdvancedResultSetUpdate {  
  
publicstaticvoid main(final String[] args) {  
try {  
  
Class.forName("com.mysql.jdbc.Driver").newInstance();  
  
final String username = "root";  
final String password = "rahasia";  
  
final Connection koneksi = DriverManager.getConnection(  
"jdbc:mysql://localhost:3306/usu", username, password);  
final Statement statement = koneksi.createStatement(  
    ResultSet.TYPE_SCROLL_INSENSITIVE,  
    ResultSet.CONCUR_UPDATABLE);  
final ResultSet result = statement.executeQuery("SELECT "  
    + "ID, NAMA, TANGGAL_LAHIR, " + "ALAMAT, "  
    KONTAK "  
    + " FROM IDENTITAS");  
  
    result.absolute(1);  
  
    result.updateInt(1, 2);  
    result.updateString(2, "Nesia");  
final Calendar cal = Calendar.getInstance();  
    cal.set(1988, Calendar.OCTOBER, 4);  
    result.updateDate(3, new  
    Date(cal.getTimeInMillis()));  
    result.updateString(4, "Bandung");  
    result.updateString(5, "nesia@gmail.com");  
  
    result.updateRow();  
  
    } catch (final SQLException ex) {  
    } catch (final InstantiationException ex) {  
    } catch (final IllegalAccessException ex) {  
    } catch (final ClassNotFoundException ex) {  
    }  
    }  
}
```

Java 15 AdvancedResultSetUpdate.java

Dari kode diatas kita telah mengubah record/baris ke -1 :



```
mysql> SELECT * FROM IDENTITAS;
```

ID	NAMA	TANGGAL LAHIR	ALAMAT	KONTAK
2	Nesia	1988-10-04	Bandung	nesia@gmail.com
10	Eko Kurniawan	1988-12-29	Bandung	echo_khannedy@yahoo.co.id
70	Usu	1980-12-14	Subang	usu@gmail.com
100	Budi	1982-05-17	Bandung	budi@gmail.com
101	Sumarno	1986-09-10	Surabaya	sumarno@gmail.com
102	Tukiyem	1978-10-10	Semarang	tukiyem@yahoo.com
103	Sumarni	1988-03-17	Surabaya	sumarni@hotmail.com
104	Rudi	1986-07-10	Bandung	rudi@gmail.com
105	Friko	1987-09-09	Bogor	friko@yahoo.com
106	Purwangga	1988-08-09	Subang	purwangga@yahoo.com
107	Joko	1987-10-10	Cirebon	joko@yahoo.com
108	Hisyam	1988-01-01	Depok	hisyam@gmail.com

```
12 rows in set (0.00 sec)

mysql>
```

Gambar 49 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.7.3 Menghapus Data

Untuk menghapus record/baris menggunakan ResultSet kita bisa menggunakan metode `deleteRow()`, namun sebelumnya kita harus menentukan record yang akan dihapus dengan metode `absolute(int indexRecord)`, misal kita akan menghapus record/baris yang ke-1 :

```
package usu.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author usu
 */
public class AdvancedResultSetDelete {

    public static void main(final String[] args) {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            final String username = "root";
            final String password = "rahasia";
```

```

final Connection koneksi = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/usu", username, password);
final Statement statement = koneksi.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
final ResultSet result = statement.executeQuery("SELECT "
    + "ID, NAMA, TANGGAL_LAHIR, " + "ALAMAT,
KONTAK "
    + " FROM IDENTITAS");

result.absolute(1);

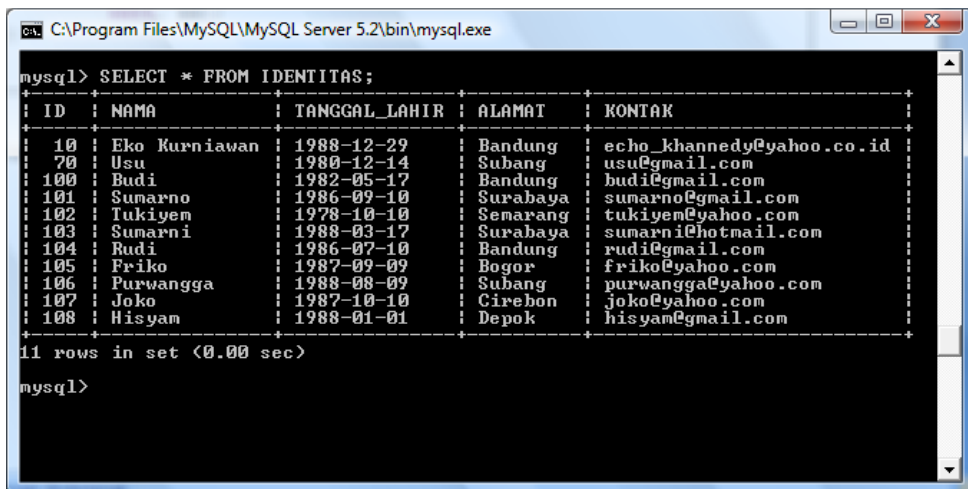
result.deleteRow();

} catch (final SQLException ex) {
} catch (final InstantiationException ex) {
} catch (final IllegalAccessException ex) {
} catch (final ClassNotFoundException ex) {
}
}
}

```

Java 16 AdvancedResultSetDelete.java

Setelah kode diatas dijalankan, maka record ke-1 akan terhapus dalam database :



```

mysql> SELECT * FROM IDENTITAS;

```

ID	NAMA	TANGGAL_LAHIR	ALAMAT	KONTAK
10	Eko Kurniawan	1988-12-29	Bandung	echo_khannedy@yahoo.co.id
70	Usu	1980-12-14	Subang	usu@gmail.com
100	Budi	1982-05-17	Bandung	budi@gmail.com
101	Sunarno	1986-09-10	Surabaya	sunarno@gmail.com
102	Tukiyem	1978-10-10	Semarang	tukiyem@yahoo.com
103	Sunarni	1988-03-17	Surabaya	sunarni@hotmail.com
104	Rudi	1986-07-10	Bandung	rudi@gmail.com
105	Friko	1987-09-09	Bogor	friko@yahoo.com
106	Purwangga	1988-08-09	Subang	purwangga@yahoo.com
107	Joko	1987-10-10	Cirebon	joko@yahoo.com
108	Hisyam	1988-01-01	Depok	hisyam@gmail.com

```

11 rows in set (0.00 sec)

mysql>

```

Gambar 50 Menampilkan Seluruh Data Dalam Tabel IDENTITAS

2.8 Pilih Yang Mana?

Dan sekarang pertanyaannya, apa yang harus kita gunakan dalam mengolah database, Statement, PreparedStatement atau ResultSet. Jawabannya tergantung kebutuhan. Misal saja proses INSERT dan UPDATE sangat cocok menggunakan

PrepareStatment, dan proses DELETE dan SELECT sangat cocok menggunakan Statement. Namun ada kalanya proses SELECT lebih baik menggunakan PreparedStatement, yaitu ketika proses SELECT tersebut memiliki banyak kondisi. Dan untuk ResultSet baik digunakan ketika kita membuat TableModel yang secara otomatis autoupdate ketika kita rubah nilainya.

3 Tentang Penulis



Penulis bernama **Eko Kurniawan Khannedy S.Kom.** Lahir di kota Subang tanggal 29 Desember 1988, dan besar di kota Subang. Penulis merupakan lulusan Universitas Komputer Indonesia.

Saat ini penulis menjabat sebagai **Chief Executive Officer** di **StripBandunk**, yaitu perusahaan yang bergerak di pengembangan teknologi informasi dan komunikasi.

Penulis aktif di berbagai komunitas teknologi dan saat ini penulis adalah **Leader** di komunitas **Java User Group Bandung** dan juga **Moderator** di komunitas **NetBeans Indonesia**.

Penulis dapat dihubungi di :

- echo.khannedy@gmail.com
- <http://twitter.com/khannedy>
- <http://facebook.com/khannedy>

:D