



Panduan Praktis Database MySQL di Linux

*“... saya persembahkan untuk Negara dan Bangsa yang saya cintai ini
Indonesia Raya, pengguna dan pecinta linux di seluruh negeri, serta untuk
kemajuan pendidikan di Indonesia.. “*

Lisensi Dokumen

Seluruh isi dalam dokumen ini dapat digunakan, dimanfaatkan dan disebarluaskan secara bebas untuk tujuan pendidikan, pembelajaran dan bukan komersial (non profit), dengan syarat tidak menghilangkan, menghapus atau mengubah atribut penulis dokumen ini dan pernyataan dalam lisensi dokumen yang disertakan di setiap dokumen.

Kata Pengantar

Saat ini dunia komputer Indonesia sedang kencang menyoroti bagaimana menggunakan software / sistem operasi yang legal. Menghargai hak cipta sebagai mana yang tertuang dalam Undang-Undang, membuat beberapa orang berpikiran untuk menggunakan software yang legal. Software tersebut diharapkan mampu menekan arus pembajakan yang saat ini menjadi - jadi. Beberapa software alternatif pun mulai bermunculan baik yang berasal dari luar negeri sampai produk lokal. Nah, salah satu software legal (untuk saat ini) yang sangat fenomenal adalah MySQL Database.

MySQL menjadi fenomenal karena legal untuk digunakan secara gratis serta kemampuannya menghandle data yang luar biasa besar. Dan yang lebih fenomenal lagi adalah kemampuannya untuk dijalankan pada lintas platform baik linux maupun windows serta OS lainnya.

Terdorong dari rasa "frustasi" saya mencari buku yang secara khusus membahas beberapa software yang berjalan pada OS linux , coba anda bisa perhatikan sendiri. Kebanyakan buku-buku yang ada ditoko buku hampir semuanya berjalan pada salah satu OS tertentu. Nah, hal inilah yang mencari sumber inspirasi saya untuk membuat buku yang berjudul "Panduan Praktis MySQL di Linux untuk Pemula".

Buku panduan praktis yang saya harapkan dapat memberikan semangat generasi tua maupun muda Indonesia untuk lebih bisa nyaman menggunakan OS Linux. Tanpa ada rasa bingung dan khawatir tidak dapat mengoprasionalkan / menggunakan OS Linux.

Dan pada buku saya kali ini, saya menggunakan Blankon Linux. Kenapa Blankon? Karena saya pikir inilah produk lokal yang saat ini paling populer dan mempunyai kemampuan yang powerfull. Serta memiliki dokumentasi serta tim pengembang yang komplit.

Pada buku saya kali ini saya bagi kedalam 10 Chapter, yaitu :

Chapter 1 : Tentang MySQL, membahas tentang MySQL serta beberapa keunggulan MySQL.

Chapter 2 : Instalasi dan Konfigurasi, membahas tentang bagaimana melakukan instalasi MySQL dengan menggunakan XAMPP for Linux dan mengkonfigurasi pada sistem Linux.

Chapter 3 : Memulai MySQL, membahas tentang bagaimana memulai menggunakan database MySQL. Membuat dan mengelola database. Serta mengenal tipe data pada MySQL.

Chapter 4 : Bergaul dengan Tabel, Membahas bagaimana membuat serta mengelola tabel.

Chapter 5 : Mengolah data pada MySQL, membahas tentang cara mengelola data pada database MySQL.

Chapter 6 : Operator Aritmatika, Operator Logika dan Operator Pembandingan, membahas beberapa operator yang dapat digunakan untuk melakukan pengolahan data pada MySQL.

Chapter 7 : Fungsi-fungsi dalam MySQL, membahas tentang fungsi-fungsi apa saja yang dapat digunakan untuk melakukan pengolahan data.

Chapter 8 : Relasi Antar Tabel, membahas mengenai relasi antar tabel serta mengolah data pada tabel 2 tabel atau lebih.

Chapter 9 : Management User MySQL, membahas mengenai pengelolaan pengguna MySQL. Menambah, menghapus serta memberikan ijin akses kepada pengguna.

Chapter 10 : Penutup. Merupakan penutup dari buku ini.

Semoga apa yang saya tulis ini menjadi penyemangat bagi rekan-rekan untuk aktif dalam menulis buku terutama sekali buku-buku yang berhubungan dengan operasi Linux. Karena masih banyak sekali buku-buku tentang Linux yang lebih cenderung Linux sebagai Server bukan kepada penggunaan secara operator.

Penulis menyadari masih sangat banyak kekurangan disana sini dalam buku ini. Segala bentuk kritik dan saran akan saya terima dengan senang hati. Saya harapkan kritik dan saran dari sobat semua terus menerus disampaikan kepada saya.

Agar kedepannya dapat menghasilkan karya yang lebih baik lagi. Dan juga saya berharap ada sobat yang dapat dijadikan media diskusi untuk menambah dan mengasah pengetahuan saya.

Tak lupa saya haturkan banyak terima kasih dari beberapa pihak yang baik blog maupun ebooknya saya jadikan bahan referensi bagi buku saya. Sekian dan selamat menikmati buku ini.

Tegal, Oktober 2012

Penulis

Dimas Edu Prasada

Daftar Isi

Lisensi Dokumen	iii
Kata Pengantar	iv
Daftar Isi	vii
Chapter 1 : Mengenal Database MySQL	1
Chapter 2 : Instalasi dan Konfigurasi MySQL	4
Chapter 3 : Memulai MySQL	8
Chapter 4 : Bergaul dengan Tabel	14
Chapter 5 : Mengolah data pada MySQL	23
Chapter 6 : Operator Aritmatika, Operator Logika dan Operator Perbandingan	39
Chapter 7 : Fungsi - Fungsi dalam MySQL	49
Chapter 8 : Akses Data antar Tabel	62
Chapter 9 : Management User MySQL	74
Chapter 10 : Penutup	80
Refrensi	81
Tentang Penulis	82

Chapter 1

Mengenal Database MySQL

MySQL. Ya, sapa yang tidak tahu MySQL tunjuk jari.... Bagi anda seorang application developer pasti sangat familiar atau bahkan pernah dengar tentang salah satu Database yang mumpuni di dekade terkahir. Saya pribadi mengenal MySQL dari tahun 2007 dan sampai sekarang masih menggunakan database ini, yang digabung dengan bahasa kesukaan saya yaitu PHP.

Tapi, ketika itu masih di OS Windows, tentunya banyak kemudahan dalam menggunakan database tersebut. Banyak tool GUI yang bisa anda gunakan dengan mudah untuk bermain dengan MySQL. Pada tutorial kali ini saya tidak akan membahas bagaimana menggunakan MySQL pada OS tetangga baik saya.

Namun, sebelum membahas MySQL secara lebih detail, saya akan berbagi pengalaman menggunakan Linux. Ketika terjadi sebuah serangan virus yang mematikan di OS tetangga, kalo tidak salah akhir 2008. Dan saya selaku administrator server yang menggunakan OS tetangga dibuat pusing untuk menangani serangan virus tersebut. Nah, dari situlah saya mulai mencari tentang OS alternatif yang lebih handal terhadap virus-virus nakal.

Nah, ketika itu OS yang pertama kali saya gunakan adalah Blankon Lontara dan saya install pada PC jadul saya. Kecepatan PC tersebut hanya sekitar 300-an Mhz dan alhamdulillah bisa berjalan lancar, malah saya gunakan untuk internetan menggunakan modem hape china plus umtsmon. Saya coba untuk upload file menggunakan filezilla juga OK, dan untuk mengetik dokumen ketika itu menggunakan abiword juga sangat bagus dan powerfull. Tapi, sayang PC tersebut sudah di jual ke pengepul barang rongsok dan hanya tinggal kenangan belaka. Hehehe... ^_^

Proses migrasi ke Linux yang saya alami juga bukan perkara yang gampang. Kurang lebih baru 1 tahun terakhir baru nyaman menggunakan linux. Apa pasal? Karena dalam kurun waktu 2008 sampe 2011 saya berusaha mencari software pengganti yang ada di OS tetangga agar bisa digunakan di Linux. Karena pekerjaan saya adalah seorang penulis kode PHP jadi berusaha

hunting aplikasi yang nyaman saya gunakan di OS Linux. Dan Alhamdulillah ternyata MySQL dan PHP dapat berjalan sangat lancar di OS Linux. Pekerjaan saya tidak terhambat dan tentunya gak perlu repot untuk menelanjangi satu per satu file yang terkena virus.

Oke, itulah kisah saya tentang Linux. Mari kita kembali ke pokok bahasan utama kita yaitu MySQL yang paling populer sejagad raya dan banyak sekali penggunaannya.

Selayang Pandang MySQL

Digarap oleh perusahaan yang bermarkas di Finlandia yaitu AB serta oleh 3 orang jenius yaitu David Axmark, Allan Larson, serta si abang Monthy (Michale Monthy Widenius). MySQL AB memiliki hak cipta penuh terhadap MySQL sebelum diakuisisi oleh Oracle.

MySQL adalah sebuah database yang bersifat opensource dan free, sehingga MySQL dapat dengan mudah dikembangkan oleh pihak ketiga baik individu maupun skala organisasi. Kemampuan MySQL yang dapat berjalan hampir di semua platform OS yang ada, menjadikan MySQL sebuah database idola di kalangan pengembang aplikasi.

Namun dalam perkembangan berikutnya dan apalagi ketika diakuisisi oleh Oracle, MySQL seolah - olah *dibunuh pelan - pelan* dengan lambatnya pengembangan MySQL. Sehingga akhirnya si Abang Mounthy berinisiatif membuat database yang mirip dengan MySQL yaitu si manis Maria DB. MySQL yang dikabarkan tidak akan gratis lagi ditahun 2015 menjadikan MySQL mulai hilang pamor.

Tapi tidak ada salahnya kita belajar tentang MySQL karena si manis Maria DB dibuat dari kakaknya yaitu MySQL serta oleh pengembang yang sama. Jadi, sangat besar kemungkinan apa yang ada di MySQL sama persis di si manis Maria DB.

Kenapa Memilih MySQL?

Kenapa memilih MySQL ? Dari pengalaman saya pribadi, MySQL sangat asik sekali digunakan terutama dapat *menghandle* data yang besar.

Namun, kadang juga bikin repot ketika trafik yang terlalu besar. Tapi secara keseluruhan MySQL merupakan database yang asik digunakan. Berikut beberapa keistimewaan MySQL :

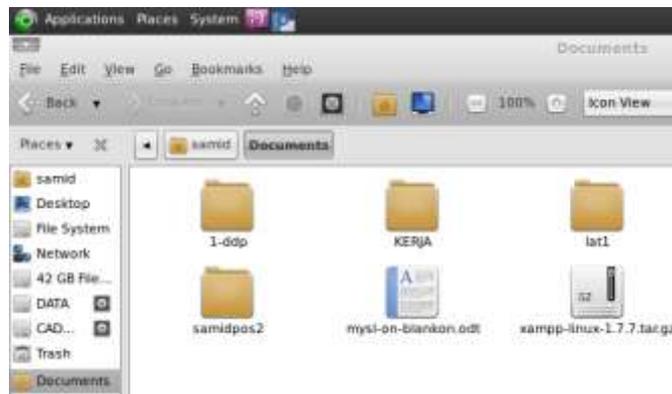
- ✓ **Multiplatform** : mysql adalah salah satu database yang dapat berjalan di hampir semua jenis OS.
- ✓ **Gratis** : saat ini untuk mendapatkan MySQL masih belum dipungut biaya alias gratis.
- ✓ **Open Source** : dengan kode yang dapat kita utak-atik memungkinkan MySQL dapat kita kembangkan sesuka kita.
- ✓ **Client-Server** : Klien dan server MySQL dapat berjalan di komputer yang sama atau terpisah.
- ✓ Dan masih banyak lagi kelebihan MySQL yang membuatnya mendunia.

Chapter 2

Instalasi dan Konfigurasi MySQL

Instalasi MySQL di Blankon

Dalam tutorial ini saya menggunakan instalasi paket milik *apachefriends.org*, yaitu LAMPP (Xampp for Linux). Sehingga kita menginstall secara paket. Dan versi yang saya gunakan adalah 1.7.7, untuk mendapatkan paket instalasi tersebut silahkan kunjungi situs <http://apachefriends.org/> kemudian carilah yang dalam bentuk versi linux. Kalo yang ada di PC saya :



Gambar 1. File Xampp Linux yang sudah di download.

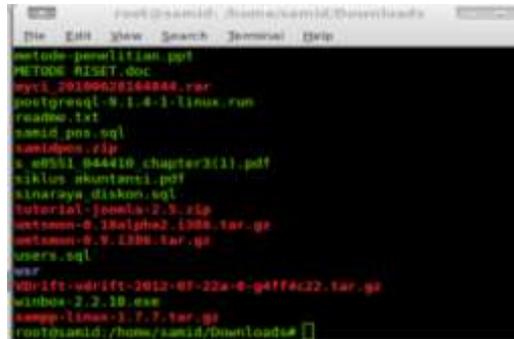
Nah, jika anda sudah mempunyai file seperti yang ada pada gambar diatas. Langkah berikutnya adalah kita akan melakukan instalasi file paket tersebut, kita gunakan terminal untuk melakukan instalasi. Tekan **Ctrl + Alt + T** secara bersamaan maka secara otomatis akan muncul penampakan jendela terminal. Login menggunakan user root anda dan setelah itu lakukan perintah instalasinya.

Perlu diingat bahwa bagi anda yang belum mahir menggunakan terminal terutama untuk mengakses direktori, maka arahkan direktori instalasi pada folder Downloads karena hasil download file biasanya ada di folder tersebut.

Ketikan perintah berikut :

```
root@samid:/home/samid# cd /Downloads
```

Diikuti dengan anda ketikkan `ls` yang fungsinya untuk menampilkan daftar file apa saja yang ada pada folder Downloads sehingga akan muncul penampakan seperti berikut :



Gambar 2. Tampilan daftar file pada direktori Downloads

Kemudian kita tinggal melanjutkan ke langkah selanjutnya yaitu proses ekstraksi file lampp-nya. Untuk melakukan ekstraksi file lampp ketikkan perintah berikut pada terminal :

```
root@samid:/home/samid/Downloads#tar -xvzf xampp-linux-1.7.7.tar.gz -C /opt
```

Perintah `-C /opt` dimaksudkan kita mengkopikan file - file hasil ekstraksi tadi ke folder `/opt`. Semua file konfigurasi lampp ada di `/opt/lampp`.

Konfigurasi MySQL

Nah, selanjutnya kita akan memberikan password untuk mysqlnya. Karena secara default atau bawaan mysql tidak mempunyai password alias password kosong. Oh, iya untuk username default mysql gunakan **root**. Sebelum menkonfigurasi security mysql terlebih dahulu jalankan mysql dengan mengetikkan perintah :

```
root@samid:/home/samid# /opt/lampp/lampp start
```

Selanjutnya kita setting konfigurasi security-nya dengan mengetikkan perintah :

```
root@samid:/home/samid# /opt/lampp/lampp security
```

Ikuti saja perintah yang ada hingga selesai. Dan selamat anda sudah berhasil melakukan instalasi MySQL plus dengan paket-paket yang lainnya. Dan ada satu lagi yang perlu anda lakukan

sebelum lebih jauh mengotak atik MySQL. Ada beberapa hal yang perlu anda lakukan, yaitu :

Pertama, anda harus melakukan link dari direktori `/opt/lampp/var/mysql/mysql.sock` ke direktori `/var/run/mysqld/mysqld.sock`. Biasanya kita akan mengalami error ketika membuka mysql seperti ini :

```
ERROR 2002 (HY000): Can't connect to local MySQL server through socket
`/var/run/mysqld/mysqld.sock'  (2)
```

Solusi untuk error tersebut adalah buat direktori baru dengan nama **mysqld** di `/var/run`. Kemudian jangan lupa berikan full akses pada direktori **mysqld**. Ketikkan perintah berikut :

```
root@samid:/home/samid#mkdir /var/run/mysqld
```

Nah, kalo anda sudah membuat direktori tersebut langkah berikutnya adalah kita tinggal membuat link dengan mengetikkan perintah :

```
root@samid:/home/samid#ln -s /opt/lampp/var/mysql/mysql.sock /var/run/mysqld/mysqld.sock
```

Maksudnya adalah kita membuat link dengan shortcut dari direktori `mysql` yang ada di `/opt/lampp` ke direktori `/var/run/mysqld`. Jika berhasil maka anda bisa menjalankan MySQL pada sistem anda.

Kedua, agar bersifat permanen maksudnya adalah ketika sistem BlankOn anda melakukan startup maka apa yang sudah kita setting dapat secara otomatis akan start. Bagaimana caranya? Anda cukup mengetikkan perintah berikut, buka file `rc.local` di gedit (penulis suka gedit karena bisa pake mouse) heuheuheu :-p.

```
root@samid:/home/samid#gedit /etc/rc.local
```

tambahkan command berikut :

```
ln -s /opt/lampp/var/mysql/mysql.sock /var/run/mysqld/mysqld.sock /opt/lampp/lampp start
```

agar lebih jelas saya sertakan gambarnya :

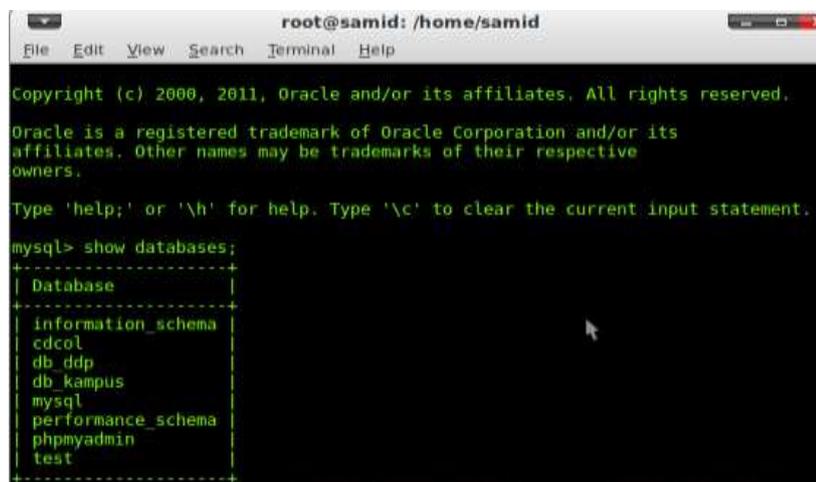
```
#  
# rc.local  
#  
# This script is executed at the end of each multiuser runlevel.  
# Make sure that the script will "exit 0" on success or any other  
# value on error.  
#  
# In order to enable or disable this script just change the execution  
# bits.  
#  
# By default this script does nothing.  
/opt/lampp/lampp start  
ln -s /opt/lampp/var/mysql/mysql.sock /var/run/mysqld/mysqld.sock  
  
exit 0
```

Gambar 3. File rc.local

Demikian beberapa setting yang perlu anda lakukan, memang bagi anda yang belum terbiasa rasanya sangat ribet (rempong cin..). Tapi, dengan seringnya mengotak – atik BlankOn saya jamin anda akan semakin cinta dengan Linux. Nah, coba jalankan MySQL-nya, ketikkan perintah berikut :

```
root@samid:/home/samid#mysql -u root -p
```

Biasanya akan diminta password untuk mengakses MySQL, password yang anda masukkan adalah password yang telah anda buat ketika anda melakukan perubahan security pada sistem Lampp. Kalo berhasil maka akan muncul penampakan berikut :



```
root@samid: /home/samid  
File Edit View Search Terminal Help  
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| cdcol |  
| db_ddp |  
| db_kampus |  
| mysql |  
| performance_schema |  
| phpmyadmin |  
| test |  
+-----+
```

Gambar 4. Tampilan MySQL pada Blankon Pattimura

Chapter 3

Memulai MySQL

Setelah kita melewati bagian yang sangat melelahkan (bagi anda yang baru kenal linux) pada saat melakukan konfigurasi MySQL. Pada bab ini akan kita bahas dan pelajari bersama tentang menggunakan MySQL. Tapi sebelum memulai (menunggu lagi.. lagi lagi menunggu.. -_-"), ada beberapa hal yang perlu anda ketahui tentang MySQL.

Bentuk perintah MySQL

Secara umum perintah MySQL dibagi menjadi 3, yaitu :

- Data Definition Language (DDL), perintah ini merupakan perintah yang berhubungan dengan struktur Database dan Table. Jadi, bukan merupakan perintah yang punya sangkut paut dengan record secara langsung. Beberapa perintah DDL : CREATE, DROP, ALTER, RENAME dll.
- Data Manipulation Language (DML), perintah ini adalah perintah yang secara langsung berhubungan dengan record yang ada dalam database kita. Baik menambah record, mengubah, menghapus dan menampilkan record semua dilakukan dengan perintah jenis ini. Beberapa perintah DML : SELECT, INSERT, DELETE, UPDATE dll.
- Data Control Language (DCL), perintah ini adalah perintah yang berhubungan dengan manipulasi user. Pemberian hak akses dan penambahan user dapat dilakukan dengan DCL. Beberapa perintah DCL : REVOKE, GRANT.

Tipe Data MySQL

Sebelum lebih jauh anda bekerja dengan MySQL ada baiknya anda juga mengenal beberapa tipe data yang ada pada MySQL. Secara garis besar MySQL mengenal tiga kelompok type data, yaitu tipe data numeric atau angka, tipe data tanggal dan waktu serta tipe data string.

Berikut tipe data yang sering digunakan di dalam database MySQL:

- CHAR : digunakan untuk menyimpan data string dengan jumlah karakter 1-255.
- VARCHAR : digunakan untuk menyimpan data string dengan jumlah karakter 1-255. Namun yang membedakan dengan CHAR adalah VARCHAR lebih fleksibel terhadap tipe data lainnya.
- INT : digunakan untuk data integer (bilangan bulat negative dan positif) dengan nilai antara -2.147.483.648 s/d 2.147.483.647.
- FLOAT : untuk menyimpan data yang berbentuk bilangan pecahan positif dan negative presisi tunggal.
- DECIMAL : untuk menyimpan bilangan pecahan positif negative
- DATE : digunakan untuk menyimpan data tanggal dengan format YYYY-MM-DD. Dimana YYYY (year) adalah format tahun misalnya 1999, MM menunjukkan (month) atau bulan misalnya 01, dan DD (day) menunjukkan tanggal misalnya 25.
- DATETIME : tipe data yang digunakan untuk menyimpan data tanggal. Namun, tipe DATETIME lebih detail karena menyertakan waktu sekarang. Dengan format YYYY-MM-DD HH:MM:SS (H untuk jam, M untuk menit dan S untuk detik)
- TEXT dan BLOB : digunakan untuk menyimpan data string dengan jumlah karakter antara 255 sampai dengan 65.535. Perbedaan antara tipe TEXT dan BLOB adalah terletak pada case-sensitive yang dimiliki data BLOB.
- ENUM : enumerasi kumpulan data. Biasanya digunakan untuk menampung data yang berupa kumpulan dan dikriteriakan sesuai dengan kategorinya.

Dan masih banyak lagi beberapa jenis / tipe data dari MySQL, anda dapat melakukan penggalian lebih dalam menggunakan andalan kita yaitu, Si Mbah GOOGLE.... “/^_^\”....

Menggunakan MySQL

Nah, inilah bagian utama dari sekian banyak tulisan yang anda baca, yaitu menggunakan MySQL. Oke, langsung login ke MySQL anda menggunakan terminal ketikan :

```
root@samid:/home/samid#mysql -u root -p
```

Jika, dimintai password tinggal ketikan password yang sudah anda buat tadi.

Melihat Database

Sebelum membuat database, biasanya kita lihat dahulu database yang sudah ada pada sistem kita. Karena kita sering lupa (penulis juga termasuk), terkadang ketika akan membuat database muncul pesan error yang ternyata database yang kita buat sudah ada. Nah, untuk menghindari cara ini adalah cara yang paling mujarab. Ketikkan perintah :

```
SHOW DATABASES;
```

Oh, iya jangan lupa ketika menuliskan perintah akhiri dengan tanda ; tanda tersebut mengisyaratkan akhir dari perintah / command. Jika benar maka akan muncul list database yang ada pada sistem kita. Seperti contoh berikut :

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| cdcol      |
| db_ddp     |
| db_kampus  |
| mysql      |
| performance_schema |
| phpmyadmin |
| test       |
+-----+
```

Membuat Database

Membuat database dapat dilakukan dengan perintah :

```
CREATE DATABASE nama_database;
```

Terkadang ada juga perintah optional yaitu [IF NOT EXISTS], perintah tersebut maksudnya adalah jika database tersebut tidak ada dalam sistem maka sistem akan meng-create database baru. Buat sebuah database baru dengan nama **db_pegawai** .

```
CREATE DATABASE db_pegawai;
```

Jika command yang anda masukkan benar maka akan nampak pesan berikut :

```
mysql> create database db_pegawai;  
Query OK, 1 row affected (1.21 sec)  
mysql> █
```

Periksa kembali dengan perintah **SHOW DATABASES;** untuk memastikan apakah database yang anda buat benar – benar ada dalam sistem.

Membuka database

Untuk membuat / menggunakan / memilih salah satu database yang ada dalam sistem kita, dapat menggunakan perintah :

```
USE nama_database;
```

Kita akan menggunakan database yang sudah kita buat yaitu **db_pegawai** maka ketikkan saja perintah berikut :

```
USE db_pegawai;
```

Jika perintah yang anda masukkan benar maka akan muncul komen berikut :

```
mysql> use db_pegawai;  
Database changed  
mysql> █
```

Menghapus Database

Menghapus database yang ada dalam sistem dapat dilakukan dengan mengetikkan perintah :

```
DROP DATABASE nama_database;
```

Kita hapus database yang sudah kita buat tadi yaitu **db_pegawai**, ketikkan perintah berikut :

```
DROP DATABASE db_pegawai;
```

Jika perintah yang anda ketikkan benar maka akan tampak pesan berikut dari sistem :

```
mysql> drop database db_pegawai;  
Query OK, 0 rows affected (1.06 sec)  
  
mysql> █
```

Kemudian cek kembali dengan perintah **SHOW DATABASES;** apakah database db_pegawai benar - benar telah lenyap dari sistem. Oke, untuk lebih memudahkan anda memahami tentang perintah - perintah diatas cobalah contoh latihan di bawah ini. Semangat..!!!

Latihan Soal A

1. Buatlah database dengan nama siswa.

```
CREATE DATABASE siswa;
```

2. Periksalah apakah database siswa sudah ada dalam sistem.

```
SHOW DATABASES;
```

3. Gunakan database siswa yang sudah dibuat.

```
USE DATABASE siswa;
```

4. Hapuslah database siswa yang sudah dibuat.

```
DROP DATABASE siswa;
```

5. Periksa kembali apakah database siswa sudah benar-benar tidak ada dalam sistem.

```
SHOW DATABASES;
```

Latihan Soal B.

1. Buatlah database baru dengan nama : **db_retail**.
2. Periksa apakah database **db_retail** tersebut sudah ada dalam sistem.
3. Gunakan database **db_retail** yang sudah dibuat.
4. Hapuslah database **db_retail** yang sudah dibuat.
5. Periksa kembali apakah **db_retail** sudah tidak ada dalam sistem.

Bagaimana apakah sudah paham dengan mengolah database di MySQL??? kalo belum coba anda bereksplorasi dengan membuat database sesuai dengan keinginan anda. Semangatttttt!!!!!!!!!!!!!! ^_^

Chapter 4

Bergaul dengan Tabel MySQL

Oke, setelah sobat mempelajari dan memahami bagaimana cara mengawali database MySQL. Selanjutnya adalah saya akan ajak sobat semua menjajal lebih lanjut database MySQL, yaitu bagaimana kita akan merancang dan mengolah sebuah table sesuai dengan kebutuhan. Seperti yang sudah dituliskan di awal, “bergaul” dengan database dan tabel masih dalam kategori DDL atau Data Definition Language. Jadi, belum mengolah data secara mandiri alias masih berkuat dengan “wadah” data yang akan kita gunakan. Okelah, langsung saja kita awali dengan :

Membuat Table

Table merupakan bagian yang penting dari sebuah database, karena fungsi dari table adalah untuk menampung data yang akan kita gunakan. Tapi disini saya tidak akan menjelaskan secara detail tentang relasi antar table dan tetek bengeknya. (Sobat bisa pelajari di buku-buku tentang database). Untuk membuat table sintak umumnya adalah :

```
CREATE TABLE nama_tabel (  
  field1 tipe(panjang),  
  field2 tipe(panjang),  
  ...  
  fieldn tipe(panjang),  
  PRIMARY KEY (field_key)  
);
```

Oke, perintah di atas adalah perintah secara umum dari MySQL untuk membuat sebuah table. Untuk lebih memperjelas perintah di atas kita akan bersama - sama mencoba membuat sebuah table dengan ketentuan sebagai berikut :

Nama Table : t_pegawai

Nama Field	Tipe Data	Panjang	Null	Primary
Nip	Varchar	20		Ya
Nama	Varchar	100		
Alamat	Text	-		
Telp	Varchar	20	Null	
Email	Varchar	50	Null	

Untuk membuat table seperti bentuk di atas sobat dapat mengetikkan sintak seperti berikut :

```
CREATE TABLE t_pegawai (  
  nip VARCHAR(20) NOT NULL,  
  nama VARCHAR(100) NOT NULL,  
  alamat TEXT NOT NULL,  
  telp VARCHAR(20) NULL,  
  email VARCHAR(50) NULL,  
  PRIMARY KEY (nip)  
);
```

Jika sobat mendapati seperti gambar di bawah ini berarti sobat telah berhasil membuat sebuah table pada database MySQL.

```
mysql> CREATE TABLE t_pegawai (  
  -> nip VARCHAR(20) NOT NULL,  
  -> nama VARCHAR(100) NOT NULL,  
  -> alamat TEXT NOT NULL,  
  -> telp VARCHAR(20) NULL,  
  -> email VARCHAR(50) NULL,  
  -> PRIMARY KEY (nip)  
  -> );  
Query OK, 0 rows affected (0.24 sec)
```

Memunculkan daftar table

Untuk melihat table apa saja yang ada dalam database, kita dapat menggunakan command :

```
SHOW TABLES;
```

jika penulisan perintah tersebut benar maka akan muncul seperti gambar di bawah ini :

```
mysql>
mysql>
mysql> show tables;
+-----+
| Tables_in_db_simpeg |
+-----+
| t_pegawai           |
+-----+
1 row in set (0.00 sec)
```

Menampilkan struktur tabel

Sobat dapat melihat struktur table yang telah kita buat, perintah yang digunakan adalah :

```
DESC nama_tabel;
```

Untuk melihat struktur dari tabel "t_pegawai" sobat cukup mengetikkan perintah pada command line:

```
DESC t_pegawai;
```

Sehingga jika perintah tersebut benar maka akan muncul seperti gambar dibawah ini :

```
mysql> desc t_pegawai;
+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+
| nip   | varchar(20)   | NO   | PRI | NULL    |       |
| nama  | varchar(100)  | NO   |     | NULL    |       |
| alamat | text          | NO   |     | NULL    |       |
| telp  | varchar(20)   | YES  |     | NULL    |       |
| email | varchar(50)   | YES  |     | NULL    |       |
+-----+
5 rows in set (0.34 sec)
```

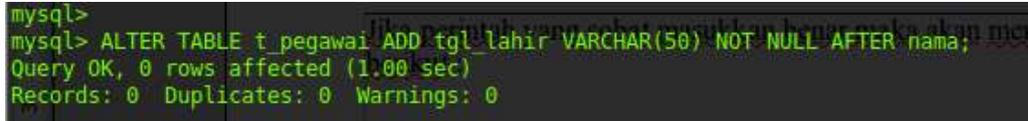
Memanipulasi Tabel dengan ALTER

1) Menambahkan Field ke dalam tabel

Kita akan mencoba untuk menambahkan field tgl_lahir setelah field nama.

```
ALTER TABLE t_pegawai ADD tgl_lahir VARCHAR(50) NOT NULL AFTER nama;
```

Jika perintah yang sobat masukkan benar maka akan menemui penampakan seperti pada gambar berikut :



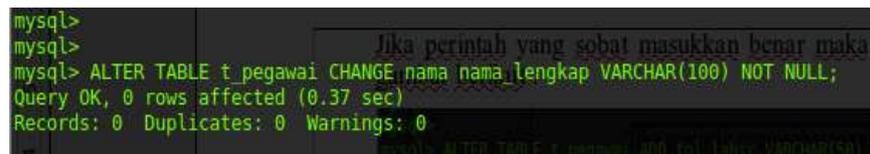
```
mysql>
mysql> ALTER TABLE t_pegawai ADD tgl_lahir VARCHAR(50) NOT NULL AFTER nama;
Query OK, 0 rows affected (1.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

2) Mengubah / mengganti nama field

Mengganti nama field **“nama”** menjadi **“nama_lengkap”** pada tabel t_pegawai, dapat dilakukan dengan perintah :

```
ALTER TABLE t_pegawai CHANGE nama nama_lengkap VARCHAR(100) NOT NULL;
```

Jika perintah tersebut berhasil dijalankan maka akan tampak pada gambar berikut :



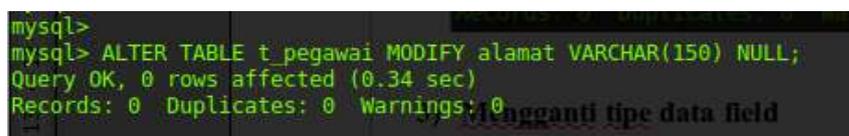
```
mysql>
mysql>
mysql> ALTER TABLE t_pegawai CHANGE nama nama_lengkap VARCHAR(100) NOT NULL;
Query OK, 0 rows affected (0.37 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

3) Mengganti tipe data field

Merubah tipe data pada field **“alamat”** dengan tipe data menjadi **“VARCHAR”** dapat dilakukan dengan perintah :

```
ALTER TABLE t_pegawai MODIFY alamat VARCHAR(150) NULL;
```

Jika perintah tersebut berhasil dijalankan maka akan tampak pada gambar berikut :



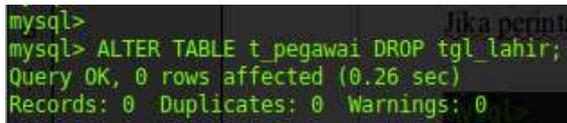
```
mysql>
mysql> ALTER TABLE t_pegawai MODIFY alamat VARCHAR(150) NULL;
Query OK, 0 rows affected (0.34 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

4) Menghapus field

Menghapus field "tgl_lahir" dari tabel "t_pegawai" dapat dilakukan dengan perintah :

```
ALTER TABLE t_pegawai DROP tgl_lahir;
```

Jika perintah tersebut diatas berhasil dijalankan maka akan tampak pada gambar berikut :



```
mysql>  
mysql> ALTER TABLE t_pegawai DROP tgl_lahir;  
Query OK, 0 rows affected (0.26 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

5) Mengganti nama tabel

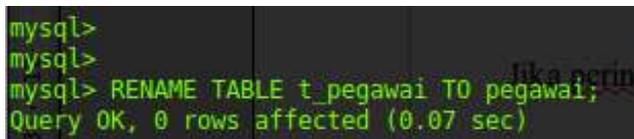
Mengganti nama tabel "t_pegawai" menjadi "pegawai" dapat dilakukan dengan perintah :

```
RENAME TABLE t_pegawai TO pegawai;
```

atau dapat juga dengan menggunakan perintah :

```
ALTER TABLE t_pegawai RENAME TO pegawai;
```

Jika perintah tersebut berhasil dijalankan maka akan tampak pada gambar berikut :



```
mysql>  
mysql>  
mysql> RENAME TABLE t_pegawai TO pegawai;  
Query OK, 0 rows affected (0.07 sec)
```

Menghapus Table

Menghapus tabel "pegawai" dapat dilakukan dengan perintah :

```
DROP TABLE pegawai;
```

Jika perintah tersebut berhasil dijalankan maka akan tampak pada gambar berikut :

```
mysql>  
mysql>  
mysql> DROP TABLE pegawai;  
Query OK, 0 rows affected (0.08 sec)
```

Bagaimana, mudahkan? Untuk mengasah kemampuan dan mengingat kembali apa yang sudah sobat pelajari saya akan berikan beberapa soal latihan sehingga semakin mengasah kemampuan sobat semua. Dan berikut adalah soal latihannya.

SOAL LATIHAN A

1. Buatlah sebuah database dengan nama **db_stok**.

```
CREATE DATABASE db_stok;
```

2. Pilih / gunakan database tersebut (**db_stok**)

```
USE db_stok;
```

3. Buatlah tabel **barang** pada database **db_stok** dengan struktur tabel sebagai berikut :

Nama Field	Tipe Data	Panjang	Null	Primary
KODE	Varchar	20		Ya
NAMA	Varchar	100		
HARGA	Int	11		
STOK	Int	11		
TOTAL	Int	11		

```
CREATE TABLE barang(  
KODE VARCHAR(20) NOT NULL PRIMARY KEY,  
NAMA VARCHAR(100) NOT NULL,  
HARGA INT(11) NOT NULL,  
STOK INT(11) NOT NULL,  
TOTAL INT(11) NOT NULL  
);
```

4. Tampilkan struktur tabel **barang** yang sudah dibuat.

```
DESC barang;
```

5. Tambahkan field **KADALUARSA** dengan tipe data **DATE** setelah field **STOK**.

```
ALTER TABLE barang ADD KADALUARSA DATE AFTER STOK;
```

6. Periksa kembali struktur tabel **barang** untuk melihat perubahannya.

```
DESC barang;
```

7. Ubahlah field **NAMA** menjadi **NAMA_BARANG**. Kemudian periksa struktur tabel **barang** untuk melihat perubahan yang terjadi.

```
ALTER TABLE barang CHANGE NAMA NAMA_BARANG VARCHAR(100) NOT NULL;
```

8. Ubahlah tipe data dari field "**KADALUARSA**" menjadi **VARCHAR**. Kemudian lihat struktur tabel **barang** untuk melihat perubahan yang terjadi.

```
ALTER TABLE barang MODIFY KADALUARSA VARCHAR(30) NOT NULL;
```

9. Hapus field **TOTAL** pada tabel **barang**. Kemudian lihat struktur tabel untuk melihat perubahan yang terjadi.

```
ALTER TABLE barang DROP TOTAL;
```

10. Ubah nama tabel **barang** menjadi **t_barang**.

```
ALTER TABLE barang RENAME TO t_barang;
```

11. Hapus tabel **t_barang**.

```
DROP TABLE t_barang;
```

12. Hapus database **db_stok**.

```
DROP DATABASE db_stok;
```

SOAL LATIHAN B

1. Buatlah sebuah database **db_siswa**.
2. Pilih / gunakan database **db_siswa**.
3. Buatlah sebuah tabel **t_siswa** dengan struktur sebagai berikut :

Nama Field	Tipe Data	Panjang	Null	Primary
NIS	Varchar	20		Ya
NAMASISWA	Varchar	150		
ALAMAT	Text	-	Null	
KELAS	Varchar	15	Null	
HAPE	Varchar	20	Null	

4. Tambahkan field **TAHUNMASUK** (tipe data terserah) setelah **KELAS**.
5. Ubah field **HAPE** menjadi field **TELPON**.
6. Hapus field **KELAS** pada tabel **t_siswa**.
7. Ubah nama tabel **t_siswa** menjadi **tbl_siswa**.
8. Hapus tabel **tbl_siswa** dan **db_siswa**.

Chapter 5

Mengolah Data pada MySQL

Oke kita sudah belajar tentang mengolah “wadah” yang akan kita gunakan untuk menampung “isian” dari wadah tersebut. Mulai dari bagaimana membuat wadahnya, sampai menghapus wadah yang sudah kita buat. Kali ini saya akan ajak sobat semua untuk coba mengutak-atik dan bersenang-senang dengan “isian” dari “wadah” yang sudah kita buat alias kita akan bermain dengan data pada MySQL.

Tapi sebelum kita mengolah data lebih baik kita kembali membuat wadah untuk melakukan pengolahan data tersebut. Oke, buatlah sebuah database dan table dengan ketentuan di bawah ini

Database : db_retail

Tabel : tbl_barang

Nama Field	Tipe Data	Panjang	Null	Primary
kode_barang	VARCHAR	20		Ya
nama_barang	VARCHAR	150		
harga_beli	INT	11		
harga_jual	INT	11		
stok	INT	5		

Menambah / memasukan data ke dalam tabel

Yang pertama adalah kita akan belajar bagaimana memasukan data ke dalam sebuah tabel. Untuk menambah data ke dalam tabel perintah yang digunakan sejatinya ada tiga jenis :

```
INSERT INTO nama_tabel VALUES('nilai1', 'nilai2', 'dst....')
```

atau dapat juga dengan bentuk perintah berikut :

```
INSERT INTO nama_tabel (nama_field1, nama_field2, ...) VALUES('nilai1',  
'nilai2', 'dst....')
```

dan bentuk yang terakhir adalah :

```
INSERT INTO nama_tabel SET field1='nilai1', field2='nilai2'....
```

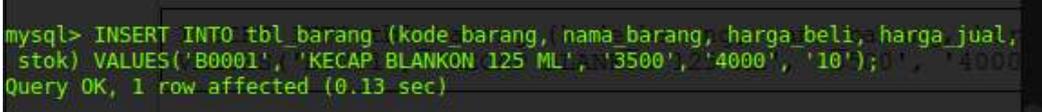
Dari ketiga jenis perintah di atas, saya merekomendasikan kepada sobat semua sebaiknya gunakan perintah yang pertama dan kedua. Tapi akan lebih baik lagi gunakan yang kedua, kenapa? Karena dari pengalaman saya, jika kita mengisi / memasukan data baru ke dalam MySQL dengan bentuk yang pertama hal yang sering terjadi adalah ketika jumlah VALUE yang sobat input tidak sesuai dengan jumlah COLUMN / FIELD yang ada pada tabel.

Nah, hal ini kadang bagi sobat yang belum mahir dengan MySQL kadang dibikin pusing oleh nya. Maka dari itu lebih baik gunakan bentuk perintah yang kedua, dengan mendeklarasikan terlebih dahulu field/kolom yang akan kita isi.

Baiklah kita langsung saja mencoba menginput / mengisi tabel yang sudah kita buat tadi dengan mencoba perintah INPUT data ke dalam table. Coba sobat ketikkan perintah berikut :

```
INSERT INTO tbl_barang (kode_barang, nama_barang, harga_beli, harga_jual,  
stok)  
VALUES('B0001', 'KECAP BLANKON 125 ML', '3500', '4000', '10');
```

Jika perintah tersebut berhasil dijalankan sobat akan mendapati informasi pada terminal seperti gambar berikut :



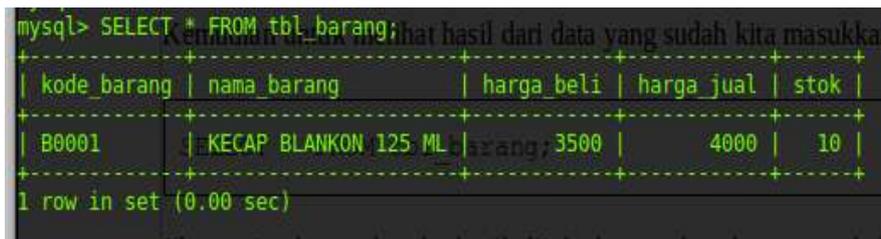
```
mysql> INSERT INTO tbl_barang (kode_barang, nama_barang, harga_beli, harga_jual,  
stok) VALUES('B0001', 'KECAP BLANKON 125 ML', '3500', '4000', '10');  
Query OK, 1 row affected (0.13 sec)
```

Perhatikan gambar di atas, informasi “Query OK, 1 row affected (0.13 sec)” menandakan / menginformasikan bahwa data berhasil anda masukan. Jadi, berhasil tidaknya kita memasukan data kedalam tabel dapat dilihat dari info tersebut.

Kemudian untuk melihat hasil dari data yang sudah kita masukkan sobat dapat mengetikkan perintah :

```
SELECT * FROM tbl_barang;
```

Jika perintah tersebut berhasil dijalankan maka akan tampak baris data yang akan muncul seperti pada gambar berikut :



```
mysql> SELECT * FROM tbl_barang;
+-----+-----+-----+-----+-----+
| kode_barang | nama_barang      | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+
| B0001      | KECAP BLANKON 125 ML | 3500      | 4000      | 10  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Mengedit / mengubah data pada tabel

Ketika terjadi kesalahan dalam proses input data, tentunya kita ingin sekali melakukan perubahan pada data tersebut. Nah, MySQL juga seperti halnya database lainnya menyediakan fitur untuk melakukan edit / ubah data. Untuk mengubah data dapat dilakukan dengan perintah :

```
UPDATE nama_table SET field1 = 'nilai1', field2 = 'nilai2'
WHERE parameter = 'nilai_parameter';
```

Hal yang sangat penting ketika sobat akan melakukan editing data pada MySQL adalah harus adanya atribut WHERE. Kenapa? Karena dengan attribut WHERE kita bisa lebih spesifik untuk melakukan edit data dan apabila atribut WHERE tidak diberikan maka sistem akan melakukan edit pada semua data yang ada pada tabel. OKI (Oleh Karena Itu), ada baiknya sobat memperhatikan hal yang satu ini yaitu atribut WHERE.

Okelah kita akan mencoba saja bagaimana fitur edit ini bekerja pada MySQL. Ketikkan perintah ini pada command-line MySQL :

```
UPDATE tbl_barang SET harga_beli = '4500', harga_jual = '5000'
WHERE kode_barang = 'B0001';
```

Jika perintah tersebut berhasil dijalankan maka sobat akan mendapati sebuah informasi seperti pada gambar di bawah ini :

```
mysql>
mysql> UPDATE tbl_barang SET harga_beli = '4500', harga_jual = '5000' WHERE kode_barang = 'B0001';
Query OK, 1 row affected (0.18 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Dan lihat kembali data yang sudah sobat edit tadi kemudian perhatian perubahan yang terjadi. Ketikkan perintah berikut :

```
SELECT * FROM tbl_barang;
```

Maka sobat akan menemukan perubahan dari data yang sudah sobat edit. Ketika sebelum di-edit, nilai pada harga_beli adalah 3500 dan nilai pada harga_jual adalah 4000. Dan setelah dilakukan edit data pada tabel tbl_barang maka nilai pada harga_beli adalah 4500 dan nilai pada harga_jual adalah 5000.

```
mysql> SELECT * FROM tbl_barang;
+----+-----+-----+-----+-----+
| kode_barang | nama_barang | harga_beli | harga_jual | stok |
+----+-----+-----+-----+-----+
| B0001      | KECAP BLANKON 125 ML | 4500      | 5000      | 10   |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Menghapus data pada tabel

Ketika sobat menemukan data yang tidak perlu ada pada tabel, sobat dapat melakukan penghapusan data record tersebut. Dan yang perlu diingat adalah data yang terhapus bersifat permanen alias ajeg alias tidak dapat dikembalikan seperti semula. Tapi jangan takut dan khawatir, MySQL sudah menyediakan fitur DELETE data yang mudah. Berikut adalah perintah DELETE pada MySQL :

```
DELETE FROM nama_tabel WHERE parameter = 'nilai_parameter';
```

Kembali seperti halnya dengan EDIT data record pada pembahasan sebelumnya, perintah

DELETE pun terdapat atribut WHERE. Karena seperti yang sudah saya tuliskan bahwa DELETE data record pada tabel bersifat permanen alias ajeg. Oleh karena itu, tetap waspada dan berhati-hati ketika akan melakukan perintah DELETE data. Tapi, kalo sobat sudah terbiasa tidak jadi soal karena dari pengalaman saya akan secara otomatis perintah WHERE selalu sobat ketikkan.

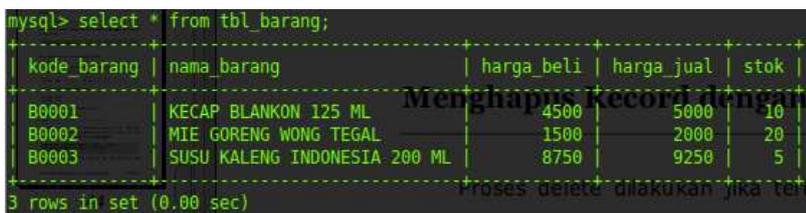
Lebih jelasnya kita akan mempraktikkan bagaimana fitur DELETE ini bekerja pada sistem MySQL. Dan sebelumnya karena kita baru memasukan 1 data record dengan kode_barang adalah B0001 maka kita akan menambahkan 2 record data lagi. Ketikkan perintah berikut:

```
INSERT INTO tbl_barang (kode_barang, nama_barang, harga_beli, harga_jual,
stok)
VALUES
('B0002', 'MIE GORENG WONG TEGAL', '1500', '2000', '20'),
('B0003', 'SUSU KALENG INDONESIA 200 ML', '8750', '9250', '5');
```

Oke kita sudah punya tiga data pada tabel **tbl_barang**. Coba periksa dahulu data record yang ada dengan perintah :

```
SELECT * FROM tbl_barang;
```

Maka akan didapati data sejumlah 3 (tiga) baris data, seperti tampak pada gambar berikut,



```
mysql> select * from tbl_barang;
+-----+-----+-----+-----+-----+
| kode_barang | nama_barang | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+
| B0001 | KECAP BLANKON 125 ML | 4500 | 5000 | 10 |
| B0002 | MIE GORENG WONG TEGAL | 1500 | 2000 | 20 |
| B0003 | SUSU KALENG INDONESIA 200 ML | 8750 | 9250 | 5 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Kemudian hapuslah data dengan kode barang B0003 pada tabel **tbl_barang** dengan mengetikkan perintah berikut :

```
DELETE FROM tbl_barang WHERE kode_barang = 'B0003';
```

Jika perintah tersebut berhasil dijalankan maka akan tampak seperti pada gambar berikut, dan jangan lupa periksa kembali data yang ada pada tabel **tbl_barang**. Jika anda berhasil melakukan delete data maka yang akan tampak adalah data dengan kode B0001 dan B0002.

```
mysql>
mysql>
mysql> DELETE FROM tbl_barang WHERE kode_barang = 'B0003';
Query OK, 1 row affected (0.14 sec)
```

Jumlah data setelah dilakukan DELETE data. Dapat dilihat jumlah record yang semula sejumlah 3 baris data kini hanya ada 2 baris data. Karena kita sudah melakukan penghapusan data pada baris data dengan parameter kode_barang dengan nilai parameter-nya adalah B0003.

```
mysql> select * from tbl_barang;
+-----+-----+-----+-----+-----+
| kode_barang | nama_barang          | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+
| B0001      | KECAP BLANKON 125 ML | 4500      | 5000      | 10  |
| B0002      | MIE GORENG WONG TEGAL | 1500      | 2000      | 20  |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec) Query OK, 1 row affected (0.14 sec)
```

Menampilkan data dengan perintah SELECT

Pada pembahasan sebelumnya sobat sudah dikenalkan dengan perintah SELECT, tapi mungkin bagi yang belum paham sebenarnya apa maksud dari perintah SELECT ini. Kali ini saya akan ajak sobat semua belajar mengenal, mengerti dan memahami perintah SELECT dan nantinya akan ada beberapa variasi dari perintah SELECT.

Secara umum perintah SELECT digunakan untuk menampilkan data yang ada pada tabel. Perintah umumnya adalah :

```
SELECT [field | *] FROM nama_table WHERE parameter = 'nilai_paremeter';
```

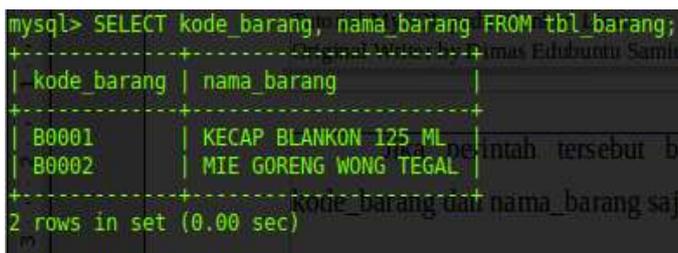
Dapat dilihat dari bentuk perintah diatas, ada [field | *] apa artinya?... Artinya bahwa kita dapat memanggil beberapa field yang kita inginkan untuk ditampilkan pada layar sistem MySQL. Dan perintah * diartikan kita akan mengambil semua field yang terdapat pada tabel tanpa terkecuali. Perhatikan contoh berikut,

```
SELECT * FROM tbl_barang;
```

Perintah diatas menggunakan karakter * ketika akan mengambil data pada tabel tbl_barang dan apa yang terjadi?. Maka semua field akan dimunculkan pada layar sistem MySQL. Dan cobalah contoh berikut,

```
SELECT kode_barang, nama_barang FROM tbl_barang;
```

Jika perintah tersebut berhasil dijalankan maka yang akan muncul adalah data pada field kode_barang dan nama_barang saja, seperti tampak pada gambar berikut,



```
mysql> SELECT kode_barang, nama_barang FROM tbl_barang;
+-----+-----+
| kode_barang | nama_barang |
+-----+-----+
| B0001      | KECAP BLANKON 125 ML |
| B0002      | MIE GORENG WONG TEGAL |
+-----+-----+
2 rows in set (0.00 sec)
```

Filter / Menyaring data

Biasanya ketika sobat mempunyai jumlah data yang sangat besar atau juga ketika sobat membuat sebuah aplikasi tentunya kita butuh data yang lebih spesifik hasil penyaringan dari sejumlah data yang ada pada tabel. MySQL memiliki fitur tersebut yaitu penyaringan data dari sejumlah data berdasarkan kriteria tertentu.

Nah sebelum mencoba lebih lanjut tentang penyaringan data, kita coba modifikasi tabel yang sudah ada yaitu tbl_barang. Coba tambahkan field **supplier** setelah field **nama_barang**,

```
ALTER TABLE tbl_barang ADD supplier VARCHAR(100) AFTER nama_barang;
```

Kemudian kita update record data yang ada pada tabel dengan perintah berikut,

```
UPDATE tbl_barang SET supplier = 'BLANKON FOOD' WHERE kode_barang = 'B0001';
```

dan lagi,

```
UPDATE tbl_barang SET supplier = 'SAMID FOOD' WHERE kode_barang = 'B0002';
```

Dan tambahkan beberapa data lagi pada tabel tbl_barang,

```
INSERT INTO tbl_barang VALUES('B0003', 'COLA GEDE',  
'BLANKON FOOD', '2750', '3000', '30');
```

```
INSERT INTO tbl_barang VALUES('B0004', 'BISKUIT COKLAT TEGAL BAHARI',  
'BLANKON FOOD', '5750', '6500', '50');
```

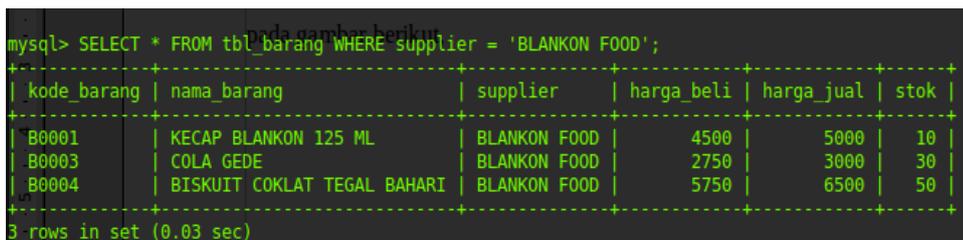
```
INSERT INTO tbl_barang VALUES('B0005', 'MINYAK GORENG BTL 1 KG',  
'SAMID FOOD', '15750', '17000', '10');
```

Oke kita sudah punya 5 baris data yang ada pada tabel. Selanjutnya sobat akan melakukan penyaringan data yang ada pada tabel. Kita akan menampilkan data yang supplier-nya hanya BLANKON FOOD saja. Maka ketikkan perintah berikut,

```
SELECT * FROM tbl_barang WHERE supplier = 'BLAKON FOOD';
```

Dan jika perintah tersebut berhasil dijalankan maka data yang akan muncul pada layar adalah seperti pada gambar berikut,

pada gambar berikut



```
mysql> SELECT * FROM tbl_barang WHERE supplier = 'BLANKON FOOD';
```

kode_barang	nama_barang	supplier	harga_beli	harga_jual	stok
B0001	KECAP BLANKON 125 ML	BLANKON FOOD	4500	5000	10
B0003	COLA GEDE	BLANKON FOOD	2750	3000	30
B0004	BISKUIT COKLAT TEGAL BAHARI	BLANKON FOOD	5750	6500	50

3 rows in set (0.03 sec)

Filter / menyaring data yang mengandung karakter tertentu.

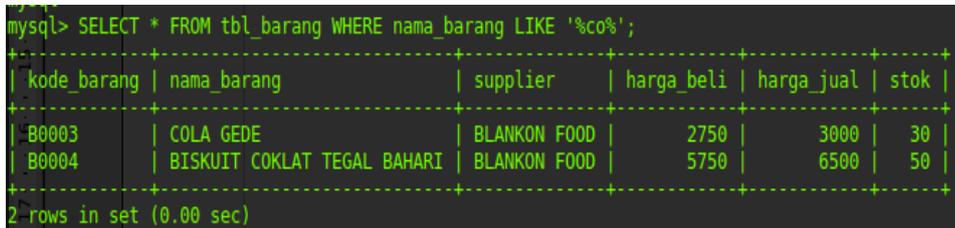
Maksudnya adalah seumpama kita akan menyaring sebuah data yang hanya mengandung karakter tertentu atau hanya beberapa karakter. Misalkan kita akan mengambil data pada sebuah tabel yang berisi data siswa dan hanya siswa yang namanya mengandung kata "DIMAS" mungkin dari beberapa siswa nama tersebut akan muncul 2 atau 3 bahkan lebih nama siswa.

Nah, kali ini sobat akan akan belajar bagaimana mengambil / menyaring data dengan kriteria mengandung karakter tertentu. Sebagai contoh kita akan mengambil data pada tabel tbl_barang yang mengandung kata "CO" pada field nama_barang.

Jadi, nama barangnya mengandung kata “CO” pada tabel tbl_barang. Ketikkan perintah berikut,

```
SELECT * FROM tbl_barang WHERE nama_barang LIKE '%CO%';
```

Jika perintah tersebut berhasil dijalankan, maka akan tampak pada layar seperti gambar berikut,



```
mysql> SELECT * FROM tbl_barang WHERE nama_barang LIKE '%co%';
+-----+-----+-----+-----+-----+-----+
| kode_barang | nama_barang          | supplier   | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+-----+
| B0003       | COLA GEDE            | BLANKON FOOD | 2750       | 3000       | 30   |
| B0004       | BISKUIT COKLAT TEGAL BAHARI | BLANKON FOOD | 5750       | 6500       | 50   |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Terlihat pada gambar data yang akan muncul adalah COLA GEDE dan BISKUIT COKLAT TEGAL BAHARI, karena keduanya terdapat kata CO pada COLA dan COKLAT.

Sortir / Pengurutan Data

Seperti halnya ketika sobat menggunakan aplikasi SpreadSheet pasti sering melakukan pengurutan data untuk beberapa baris data untuk tujuan tertentu. Nah, MySQL juga memiliki fitur untuk pengurutan data tersebut. Dan tentunya dapat membantu sobat semua untuk memudahkan pengurutan data baik secara A ke Z atau ASCENDING (kecil ke besar) ataupun secara Z ke A atau DESCENDING (besar ke kecil). Secara umum bentuk perintahnya adalah,

```
SELECT * FROM nama_table ORDER BY field [DESC/ASC]
```

Pengurutan data yang ada pada MySQL menggunakan atribut ORDER BY (bisa diartikan diurutkan berdasarkan) tentunya berasarkan field yang akan dijadikan acuan pengurutan data dan DESC/ASC menjadi bentuk pengurutan dari data tersebut. Cobalah contoh berikut,

```
SELECT * FROM tbl_barang ORDER BY nama_barang ASC;
```

Hasil dari perintah tersebut adalah,

```
mysql> SELECT * FROM tbl_barang ORDER BY nama_barang ASC;
```

kode_barang	nama_barang	supplier	harga_beli	harga_jual	stok
B0004	BISKUIT COKLAT TEGAL BAHARI	BLANKON FOOD	5750	6500	50
B0003	COLA GEDE	BLANKON FOOD	2750	3000	30
B0001	KECAP BLANKON 125 ML	BLANKON FOOD	4500	5000	10
B0002	MIE GORENG WONG TEGAL	SAMID FOOD	1500	2000	20
B0005	MINYAK GORENG BTL 1 KG	SAMID FOOD	15750	17000	10

```
5 rows in set (0.04 sec)
```

dan cobalah contoh berikut ini,

```
SELECT kode_barang, nama_barang FROM tbl_barang ORDER BY nama_barang DESC;
```

Hasil perintah tersebut adalah :

```
mysql> SELECT kode_barang, nama_barang FROM tbl_barang ORDER BY nama_barang DESC;
```

kode_barang	nama_barang
B0005	MINYAK GORENG BTL 1 KG
B0002	MIE GORENG WONG TEGAL
B0001	KECAP BLANKON 125 ML
B0003	COLA GEDE
B0004	BISKUIT COKLAT TEGAL BAHARI

```
5 rows in set (0.00 sec)
```

Sobat, tahu apa bedanya kedua gambar tersebut?? Coba perhatikan pada field nama_barang, saya sengaja hanya menampilkan kode_barang dan nama_barang agar sobat tau bedanya. Ya, benar sekali. Perintah yang pertama, meminta MySQL untuk mengurutkan data secara ASCENDING jadi akan diurutkan berdasarkan abjad. Sedangkan perintah kedua meminta MySQL mengurutkan data secara DESCENDING atau kebalikan dari abjad. Bagaimana mudah bukan?

Membatasi jumlah tampilan data

Untuk membatasi jumlah yang akan ditampilkan pada layar ada sebuah fitur yang disediakan oleh MySQL untuk melakukannya, yaitu dengan menggunakan atribut LIMIT. Jadi, sobat dapat membatasi jumlah maksimal yang nantinya akan ditampilkan baik pada layar sistem MySQL atau mungkin pada aplikasi yang sobat kerjakan. Dan biasanya juga digunakan untuk fitur pagination atau halaman pada software atau aplikasi tertentu.

Bentuk perintah umum untuk melakukan pembatasan jumlah yang akan ditampilkan adalah,

```
SELECT * FROM nama_table LIMIT awal, jumlah_data;
```

Awal diartikan dari baris ke berapa kita akan menampilkan data, tapi jika awal tidak diberikan maka sistem akan memberikan nilai 0 alias kita akan memulai menampilkan data dari baris pertama. Biasanya *awal* ini disebut dengan offset. Sedangkan *jumlah_data* diartikan sebagai berapa banyak data yang akan ditampilkan. Agar sobat tidak bingung memahami dua hal diatas baiklah sekarang coba contoh berikut,

```
SELECT * FROM tbl_barang LIMIT 2;
```

Dan hasil dari perintah tersebut adalah

```
mysql> SELECT * FROM tbl_barang LIMIT 2;
+-----+-----+-----+-----+-----+-----+
| kode_barang | nama_barang | supplier | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+-----+
| B0001 | KECAP BLANKON 125 ML | BLANKON FOOD | 4500 | 5000 | 10 |
| B0002 | MIE GORENG WONG TEGAL | SAMID FOOD | 1500 | 2000 | 20 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Terlihat bahwa data yang akan dimunculkan adalah sebanyak 2 baris dan karena kita tidak memberikan nilai offset atau nilai awal maka sistem MySQL akan langsung menganggap bahwa data yang tampil dimulai dari baris pertama. Kemudian cobalah contoh berikut,

```
SELECT * FROM tbl_barang LIMIT 2, 3;
```

Maka sobat akan mendapati seperti dibawah ini :

```
mysql> SELECT * FROM tbl_barang LIMIT 2,3;
+-----+-----+-----+-----+-----+-----+
| kode_barang | nama_barang | supplier | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+-----+
| B0003 | COLA GEDE | BLANKON FOOD | 2750 | 3000 | 30 |
| B0004 | BISKUIT COKLAT TEGAL BAHARI | BLANKON FOOD | 5750 | 6500 | 50 |
| B0005 | MINYAK GORENG BTL 1 KG | SAMID FOOD | 15750 | 17000 | 10 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Bagaimana sudah menemukan perbedaannya??. Coba perhatikan, perintah pertama

memunculkan data sebanyak 2 baris dengan dimulai dari kode_barang B0001 karena tidak memberikan nilai awal, sedangkan pada contoh kali ini data yang muncul pertama kali adalah kode_barang B0003 karena kita memberikan nilai awal 2 dan jumlah data yang dimunculkan sebanyak 3.

Sehingga secara otomatis sistem MySQL akan melewati dua baris dan langsung memunculkan data pada baris ketiga. Agar semakin memahami bagaimana mengelola data saya akan berikan latihan soal dibawah ini. Let's Cekidod....

SOAL LATIHAN A

1. Buatlah sebuah database dengan nama **db_kampus**.

```
CREATE DATABASE db_sekolah;
```

2. Buatlah sebuah tabel dengan nama **t_mahasiswa**. Dengan struktur sebagai berikut,

Nama Field	Tipe Data	Panjang	Null	Primary
nim	INT	15		Ya
nama_siswa	VARCHAR	100		
tempat_lahir	VARCHAR	50	NULL	
tanggal_lahir	DATE		NULL	
alamat	TEXT		NULL	

```
CREATE TABLE t_mahasiswa(  
nim INT(15) NOT NULL PRIMARY KEY,  
nama_siswa VARCHAR(100) NOT NULL,  
tempat_lahir VARCHAR(50) NULL,  
tanggal_lahir DATE NULL,  
alamat TEXT  
);
```

3. Masukkan data pada tabel **t_mahasiswa** dengan perintah berikut,

```
INSERT INTO t_mahasiswa VALUES('120500101', 'Dimas Edu Prasada',  
'Tegal','1988-01-27', 'Jl. Sangir No. 18 RT 09 RW 11 Kota Tegal');
```

```
INSERT INTO t_mahasiswa VALUES('120500102', 'Arkana Safa',  
'Tegal','1990-11-28', 'Jl. Mangkukusuman Raya No. 18 Slawi');
```

```
INSERT INTO t_mahasiswa VALUES('120500103', 'Regina Syifa M',  
'Banjarmasin','1991-05-05', 'Ds. Pagedangan Kabupaten Tegal');
```

```
INSERT INTO t_mahasiswa VALUES('120500104', 'Andhika Bayangkara',  
'Brebek','1992-08-08', 'Jl. Nanas No. 19 RT 10 RW 12 Tegal');
```

```
INSERT INTO t_mahasiswa VALUES('120500105', 'Arrie Boediono',  
'Brebek', '1992-02-11', 'Jl. Kapten Samadikun No. 109 Tegal');
```

```
INSERT INTO t_mahasiswa VALUES('120500106', 'Mafrikha',  
'Tegal', '1992-02-19', 'Jl. Kol. Soegiono No. 100 Slawi');
```

4. Tampilkan data yang sudah diinput ke tabel **t_mahasiswa**.

```
SELECT * FROM t_mahasiswa;
```

5. Ubahlah nama mahasiswa menjadi Sodik Palapa untuk nim 120500104.

```
UPDATE t_mahasiswa SET nama = 'Sodik Palapa' WHERE nim = '120500104';
```

6. Hapuslah data dengan nim 120500103.

```
DELETE FROM t_mahasiswa WHERE nim = '120500103';
```

7. Tampilkan data mahasiswa dimana kota lahir adalah Brebes.

```
SELECT * FROM t_mahasiswa WHERE tempat_lahir = 'Brebek';
```

8. Tampilkan field nim, nama serta tempat_lahir pada tabel t_mahasiswa dan urutkan berdasarkan nama secara ASCENDING.

```
SELECT nim, nama_siswa, tempat_lahir FROM t_mahasiswa ORDER BY nama ASC;
```

9. Tampilkan data dimana alamat mengandung kata "Tegal"

```
SELECT * FROM t_mahasiswa WHERE alamat LIKE '%Tegal%'
```

10. Tampilkan data dari tabel t_mahasiswa sebanyak 3 baris dan diurutkan berdasarkan nama_siswa secara DESCENDING.

```
SELECT * FROM t_mahasiswa ORDER BY nama_siswa DESC LIMIT 3;
```

SOAL LATIHAN B

1. Buatlah sebuah database dengan nama **db_sobat**.
2. Buatlah tabel dengan nama **tbl_teman**, dengan ketentuan sebagai berikut,

Nama Field	Tipe Data	Panjang	Null	Primary	Extra
id_teman	INT	11		Ya	Autoincrement
nama_teman	VARCHAR	100			
alamat	TEXT		NULL		
kota	VARCHAR	50	NULL		
telp	VARCHAR	30	NULL		

3. Masukkan data berikut ke dalam tabel **tbl_teman**,

id	nama_teman	alamat	kota	telp
1	Duljoni	Jl. Melati No 18	Jakarta	08567891011
2	Vansya M	Jl. Anggur No. 80	Surabaya	08122334455
3	Toni G	Jl. Sangir No. 88 RT 13 RW 11	Tegal	085742007070
4	Soimah	Ds. Pabean No. 99 RT 10 RW12	Slawi	088825566642
5	Salip Arip	Ketapang City No 45 RT 17 RW 08	Tegal	081112255667
6	Seger Sutowo	Perum. Sumbodro Indah Blok M3	Tegal	08199995599
7	Moh. Danjuro	Tegalsari RT 3 RW 4	Brebes	08155555665
8	Soni Suryani	Jl. Merak No. 12	Pekalongan	08555555663

4. Tampilkan semua data teman dan urutkan berdasarkan nama teman secara ASCENDING.
5. Ubah kota dengan Banten untuk id_teman adalah 5.
6. Ubah alamat dan telp untuk id_teman 7.
7. Hapus data teman dengan id_teman 1.

8. Tampilkan data teman dimana kota adalah Tegal.
9. Tampilkan data teman dimana nama teman mengandung kata "SO".
10. Tampilkan data dari baris ke 4 sebanyak 3 baris dan diurutkan berdasarkan nama teman secara DESCENDING.

Chapter 7 :

Operator Aritmatika, Operator Logika dan Operator Perbandingan

Setelah pada pembahasan sebelumnya sobat telah saya ajak bermain dengan data, kali ini saya akan ajak sobat semua mengenal lebih dalam tentang Operator Aritmatika, Operator Logika dan Operator Perbandingan. Ketiganya sangat penting, apa pasal? Karena banyak sekali digunakan ketika kita mengolah data yang begitu besar. Okelah dari pada terlalu lama saya membual langsung saja kita awali dengan...

Operator Aritmatika

Operator Aritmatika digunakan untuk melakukan penghitungan pada data dan tentunya tipe dari data field tersebut harus bertipe NUMERIC. Karena hanya tipe NUMERIC saja yang mampu melakukan perhitungan ARITMATIKA. Berikut adalah tabel daftar Operator Aritmatika :

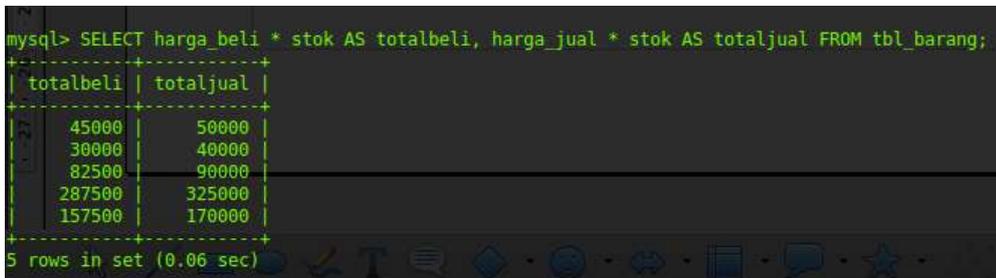
Operator	Deskripsi
+ (Tambah)	Untuk menambahkan dua data atau lebih
- (Kurang)	Untuk mengurangi dua data atau lebih
* (Kali)	Untuk mengalikan dua data atau lebih
/ (Bagi)	Untuk membagi dua data atau lebih
% (Modulus)	Sisa pembagian.

Oke kita sekarang mulai saja untuk mencoba bagaimana operator Aritmatika ini bekerja. Masih ada kan database db_retail dan tbl_barang?? kalo sudah tiada alias database dan tabelnya hilang buatlah terlebih dahulu.

Ada pada pembahasan sebelumnya, silahkan sobat kembali ke halaman sebelumnya. Kita akan coba memunculkan jumlah total dari harga jual dan harga beli pada table tbl_barang. Cobalah contoh berikut,

```
SELECT harga_beli * stok AS totalbeli, harga_jual * stok AS totaljual  
FROM tbl_barang;
```

Dan jika perintah tersebut dijalankan maka akan menghasilkan tampilan seperti tampak pada gambar berikut,



```
mysql> SELECT harga_beli * stok AS totalbeli, harga_jual * stok AS totaljual FROM tbl_barang;
```

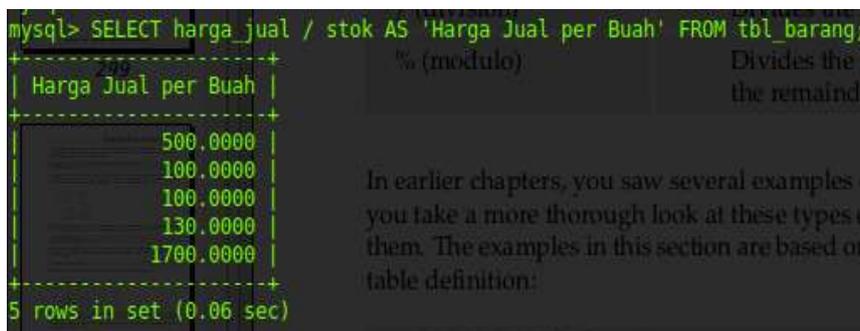
totalbeli	totaljual
45000	50000
30000	40000
82500	90000
287500	325000
157500	170000

5 rows in set (0.06 sec)

Masih belum puas dengan contoh diatas, baiklah saya akan kasih contoh lain. Coba ketikkan perintah dibawah ini pada command-line MySQL sobat,

```
SELECT harga_jual / stok AS 'Harga Jual per Buah' FROM tbl_barang;
```

Jika perintah tersebut benar maka sobat akan menjumpai tampilan seperti gambar dibawah ini,



```
mysql> SELECT harga_jual / stok AS 'Harga Jual per Buah' FROM tbl_barang;
```

Harga Jual per Buah
500.0000
100.0000
100.0000
130.0000
1700.0000

5 rows in set (0.06 sec)

Dapat juga sobat melakukan perhitungan seperti dibawah ini,

```
SELECT 12 % 3;
```

```
SELECT 100 + 4500;
```

```
SELECT 45 - 30;
```

Bagaimana mengasikkan bukan?? Ya, itulah cara penggunaan dari Operator Aritmatika pada MySQL. Dan tentunya jika akan mengolah data menggunakan Operator Aritmatika, tipe datanya harus NUMERIC. Dan akan kita lanjutkan dengan.....

Operator Logika

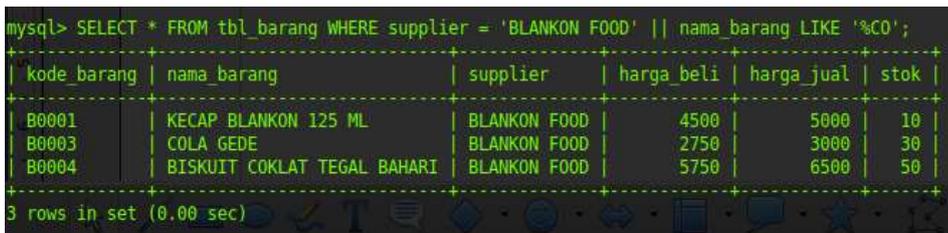
Berbeda dengan Operator Aritmatika, Operator Logika biasanya digunakan ketika pada proses penyaringan data. Dan tipe data tidak harus bersifat NUMERIC, karena memang berhubungan dengan penyaringan data. Berikut adalah tabel Operator Logika,

Operator	Deskripsi
AND atau &&	Dan / menggabungkan dua parameter atau lebih
OR atau	Atau / memilih salah satu parameter.
NOT atau !	Bukan atau tidak termasuk.

Baiklah untuk memudahkan sobat memahami apa itu Operator Logika, cobalah contoh berikut ini,

```
SELECT * FROM tbl_barang WHERE supplier = 'BLANKON FOOD' || nama_barang  
LIKE '%CO';
```

Dan jika berhasil dijalankan maka akan menghasilkan sebuah tampilan seperti gambar dibawah ini,



```
mysql> SELECT * FROM tbl_barang WHERE supplier = 'BLANKON FOOD' || nama_barang LIKE '%CO';
```

kode_barang	nama_barang	supplier	harga_beli	harga_jual	stok
B0001	KECAP BLANKON 125 ML	BLANKON FOOD	4500	5000	10
B0003	COLA GEDE	BLANKON FOOD	2750	3000	30
B0004	BISKUIT COKLAT TEGAL BAHARI	BLANKON FOOD	5750	6500	50

3 rows in set (0.00 sec)

Perintah diatas mempunyai arti bahwa kita meminta sistem untuk memunculkan data dimana pada field supplier bernilai BLANKON FOOD atau nama_barang mengandung kata CO.

Jadi, baik yang mempunyai supplier BLANKON FOOD atau pun tidak asal mengandung kata CO pada nama_barang maka akan tetap dimunculkan. Begitu juga sebaliknya jika tidak mengandung kata CO tetapi supplier adalah BLANKON FOOD maka tetap ditampilkan. Biar tidak bingung cobalah contoh ini,

```
SELECT * FROM tbl_barang WHERE supplier = 'BLANKON FOOD' && nama_barang  
LIKE '%CO%';
```

Maka hasil yang akan tampak adalah sebagai berikut,

```
mysql> 56  
mysql> karyawan pun ikut diurut. Kita coba dengan perintah di  
mysql> SELECT * FROM tbl_barang WHERE supplier = 'BLANKON FOOD' && nama_barang LIKE '%CO%';  
+-----+-----+-----+-----+-----+-----+  
| kode_barang | nama_barang | supplier | harga_beli | harga_jual | stok |  
+-----+-----+-----+-----+-----+-----+  
| B0003 | COLA GEDE | BLANKON FOOD | 2750 | 3000 | 30 |  
| B0004 | BISKUIT COKLAT TEGAL BAHARI | BLANKON FOOD | 5750 | 6500 | 50 |  
+-----+-----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

Lah, kenapa hasilnya cuman 2? kenapa juga yang tadi pake OR itu 3??? Trus gue harus bilang WOW gitu?... Karena perintah AND && akan mensyaratkan dua kriteria harus dipenuhi. Jadi, perintah diatas diartikan bahwa kita meminta sistem untuk menampilkan data dimana supplier adalah BLANKON FOOD dan nama_barang mengandung kata CO. Sehingga data yang memenuhi syarat itu hanya ada dua baris.

Dan bagi saya pribadi untuk membedakan keduanya si AND dan si OR adalah bahwa si AND ini orangnya sedikit saklek (kalo orang Tegal bilang) maksudnya sedikit tegas ketika meminta syarat. Jika ada 2 syarat maka keduanya harus dipenuhi. Berbeda dengan si OR, dia sedikit agak fleksibel tidak keras kaya si AND. Jadi kalo ada 2 syarat maka si OR akan memilih salah satunya, sehingga walaupun hanya 1 syarat saja yang terpenuhi maka akan tetap dimunculkan. Oke itu tadi mengenai Operator Logika dan selanjutnya akan kita tampilnya...

Operator Perbandingan

Operator Perbandingan merupakan operator lain diantara yang sudah sobat pelajari pada pembahasan sebelumnya. Operator Perbandingan mempunyai fungsi untuk membandingkan dua buah nilai pada MySQL. Berikut adalah tabel Operator Perbandingan,

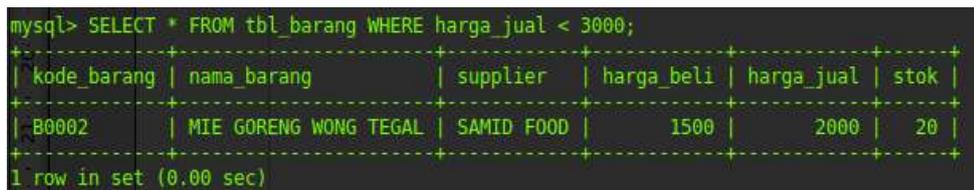
Operator	Deskripsi
=	Sama dengan
!= atau <>	Tidak sama dengan
>	Lebih besar dari
<	Kurang dari
>=	Lebih besar atau sama dengan
<=	Kurang dari atau sama dengan

Ya, dengan melihat beberapa operator yang tertera pada tabel. Mungkin sebagian pembaca ada yang merasa sudah pernah menggunakannya. Yups, betul sekali yaitu ketika melakukan penyaringan data dengan menggunakan perintah WHERE.

Dan implementasi atau penggunaan operator perbandingan memang akan lebih banyak *bersanding mesra* dengan WHERE. Okelah, cobalah contoh berikut, (masih dengan tbl_barang loh ya...)

```
SELECT * FROM tbl_barang WHERE harga_jual < 3000;
```

dan hasil query tersebut kurang lebih seperti ini,



```
mysql> SELECT * FROM tbl_barang WHERE harga_jual < 3000;
+-----+-----+-----+-----+-----+-----+
| kode_barang | nama_barang          | supplier | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+-----+
| B0002       | MIE GORENG WONG TEGAL | SAMID FOOD | 1500       | 2000       | 20   |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Perintah diatas mempunyai arti bahwa kita meminta kepada sistem MySQL agar memunculkan data dengan syarat harga_jual bernilai kurang dari (<) 3000. Sehingga MySQL akan memunculkan data pada layar seperti pada gambar diatas. Lalu bagaimana kalo syaratnya lebih dari satu??? Mau tau?? cobalah contoh berikut,

```
SELECT * FROM tbl_barang WHERE harga_beli > 5000 && stok < 100;
```

Dan jika query tersebut dijalankan maka akan menghasilkan data pada layar seperti berikut ini,

```
mysql> SELECT * FROM tbl_barang WHERE harga_beli > 5000 && stok < 100;
+-----+-----+-----+-----+-----+-----+
| kode_barang | nama_barang          | supplier   | harga_beli | harga_jual | stok |
+-----+-----+-----+-----+-----+-----+
| B0004      | BISKUIT COKLAT TEGAL BAHARI | BLANKON FOOD | 5750      | 6500      | 50  |
| B0005      | MINYAK GORENG BTL 1 KG      | SAMID FOOD  | 15750     | 17000     | 10  |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)
```

Perintah diatas mempunyai arti kita meminta sistem MySQL untuk menampilkan data pada layar, dengan dua syarat / kondisi. Dimana syarat / kondisi yang pertama adalah nilai dari harga_beli adalah lebih dari (>) 5000 dan syarat / kondisi berikutnya nilai dari stok kurang dari (<) 100. Karena menggunakan operator Logika AND (&&) maka kedua syarat / kondisi tersebut harus terpenuhi. Dan sobat pasti sudah paham kan bagaimana sifat si AND ini.

Nah, demikianlah pembahasan mengenai beberapa Operator yang mungkin akan sering sobat temui ketika sedang bermain atau mengerjakan sebuah project dengan MySQL. Seringlah berlatih dan bermain dengan MySQL, terus bereksplorasi agar kemampuan sobat semua semakin meningkat.

SOAL LATIHAN A

1. Buatlah database dengan nama **db_gaji**.

```
CREATE DATABASE db_gaji;
```

2. Buatlah tabel **tbl_karyawan** dengan ketentuan sebagai berikut,

Nama Field	Tipe Data	Panjang	Null	Primary	Extra
nik	INT	11		Ya	Autoincrement
nama_lengkap	VARCHAR	100			
jenkel	ENUM	'L', 'P'			
tanggal_lahir	DATE				
gapok	INT	11			

```
CREATE TABLE tbl_karyawan(  
nik INT(11) NOT NULL PRIMARI KEY,  
nama_lengkap VARCHAR(100) NOT NULL,  
jenkel ENUM ('L', 'P') NOT NULL,  
tanggal_lahir DATE NOT NULL,  
gapok INT(11) NOT NULL  
);
```

3. Masukkan data ke **tbl_karyawan** dengan mengetikkan perintah berikut,

```
INSERT INTO tbl_karyawan VALUES('1000', 'Brodien',  
'L', '1985-02-19', '500000');
```

```
INSERT INTO tbl_karyawan VALUES('1001', 'Lilin Herlina',  
'P', '1988-03-23', '420000');
```

```
INSERT INTO tbl_karyawan VALUES('1002', 'Agung',  
'L', '1986-01-20', '530000');
```

```
INSERT INTO tbl_karyawan VALUES('1003', 'Dwi Ratna',  
'P', '1987-11-11', '50000');
```

```
INSERT INTO tbl_karyawan VALUES('1004', 'Via Vallent',  
'P', '1990-12-25', '700000');
```

```
INSERT INTO tbl_karyawan VALUES('1005', 'Sodiq',  
'L', '1975-08-17', '900000');
```

```
INSERT INTO tbl_karyawan VALUES('1006', 'Gerry Mahesa',  
'L', '1985-08-27', '900000');
```

```
INSERT INTO tbl_karyawan VALUES('1007', 'Mella Barbie',  
'P', '1988-09-27', '800000');
```

4. Tampilkan semua karyawan dimana Jenis Kelamin adalah Lelaki (L) dan Tanggal Lahir dibawah tahun 1990.

```
SELECT * FROM tbl_karyawan WHERE jenkel = 'L' &&  
tanggal_lahir < "1990-01-01";
```

5. Tampilkan semua karyawan dimana Jenis Kelamin bukan Lelaki (L) atau Gaji Pokok dibawah 500.000

```
SELECT * FROM tbl_karyawan WHERE jenkel <> 'L' OR gapok < '500000';
```

6. Tampilkan semua karyawan dimana Tahun lahir lebih dari atau sama dengan 1984 dan Nama Lengkap tidak mengandung kata "di".

```
SELECT * FROM tbl_karyawan WHERE tanggal_lahir >= '1984-01-01'  
AND nama_lengkap NOT LIKE '%di%';
```

7. Tampilkan semua karyawan dimana Tanggal Lahir antara 1 Januari 1980 sampai dengan 31 Desember 1990 dan urutkan berdasarkan Nama Lengkap secara DESCENDING.

```
SELECT * FROM tbl_karyawan WHERE tanggal_lahir >= '1980-01-01'  
AND tanggal_lahir <= '1990-12-31' ORDER BY nama_lengkap DESC;
```

SOAL LATIHAN B

1. Buatlah database **db_kota**.
2. Buatlah tabel **t_penduduk** dengan struktur sebagai berikut,

Nama Field	Type Data	Panjang	Null	Primary	Extra
id_kota	INT	11		Ya	Autoincrement
nama_kota	VARCHAR	50			
jumlah_warga	INT	11			
status_sensus	VARCHAR	20			

3. Inputlah dengan data berikut,

id_kota	nama_kota	jumlah_warga	status
1	Tegal	1500000	Baru
2	Brebes	750000	Lama
3	Pemalang	850000	Baru
4	Slawi	450000	Baru
5	Pekalongan	1750000	Baru
6	Batang	850000	Lama
7	Banyumas	1800000	Lama
8	Purwokerto	2500000	Baru
9	Cilacap	1550000	Baru
10	Cirebon	1505000	Lama

4. Tampilkan kota dimana Jumlah warga kurang dari 1000000 (Satu juta) dan urutkan berdasarkan Nama Kota secara ASCENDING.

5. Tampilkan kota dimana Status Sensusnya adalah bukan Lama.
6. Tampilkan Nama Kota dan Jumlah Warga dimana Jumlah warga antara 500000 (Lima Ratus Ribuan) sampai 1350000 (Satu juta tiga ratus lima puluh ribu) dan Status Sensus adalah Baru.
7. Tampilkan Nama Kota dimana mengandung kata "BA" dan Jumlah Warganya kurang dari 1000000 (Satu juta).
8. Tampilkan data Kota dimana Nama Kota tidak mengandung kata "CI" atau Jumlah Warganya diatas atau sama dengan 1500000 (satu juta lima ratus ribu).

Chapter 7

Fungsi - Fungsi dalam MySQL

Pada chapter 7 ini saya akan ajak sobat semua untuk mengenal, mempelajari serta memahami beberapa fungsi - fungsi yang mungkin akan sangat berguna bagi sobat semua dalam mengolah data MySQL. Fungsi - fungsi yang nantinya akan sobat pelajari, hampir semuanya ada kemiripan dengan beberapa software - software pengolah data lainnya.

Seperti SpreadSheet, Excel, Access serta database lainnya, karena ya memang tujuannya adalah mengolah data agar lebih mudah untuk diolah ^_^.

Dan sebelum ke bagian klimaksnya terlebih dahulu saya akan kelompokkan fungsi-fungsi MySQL agar mudah sobat pelajari. Berikut adalah kelompok fungsi-fungsi nya :

- ✓ Fungsi String : berhubungan dengan operasi string.
- ✓ Fungsi Numeric : dari namanya sudah jelas ada hubungannya dengan bilangan.
- ✓ Fungsi tanggal dan waktu : berhubungan dengan data tanggal dan waktu.
- ✓ Serta fungsi-fungsi lainnya

Ke-empatnya akan sobat pelajari, dan tentunya saya harapkan sobat banyak belajar dan bereksplorasi agar lebih mendalami fungsi-fungsi yang ada pada MySQL. Okelah kita sambut dengan yang pertama, ada.....

Fungsi String

MySQL memiliki banyak sekali fungsi yang berhubungan untuk mengolah string. Berikut adalah beberapa fungsi yang masuk dalam kategori fungsi string :

- ✓ **CONCAT(string1, string2,....)** : Menggabungkan dua kolom / field (string) atau lebih. Misalkan contoh berikut,

```
SELECT CONCAT(nama_lengkap, ' ',gapok) FROM tbl_karyawan;
```

Hasilnya :

```
CONCAT(nama_lengkap, ' ',gapok)
Brodien 500000
Lilin Herlina 420000
Agung 530000
Dwi Ratna 50000
Via Vallent 700000
Sodiq 900000
Gerry Mahesa 900000
Mella Barbie 800000
```

- ✓ **CONCAT_WS(separator, String1, String2, String3,...)** : Menggabungkan dua kolom atau lebih dengan ada “imbuhan” separator sebagai penghubung dari masing-masing string.

```
SELECT CONCAT_WS('-',nama_lengkap,gapok) FROM tbl_karyawan;
```

Hasilnya :

```
CONCAT_WS('-',nama_lengkap,gapok)
Brodien-500000
Lilin Herlina-420000
Agung-530000
Dwi Ratna-50000
Via Vallent-700000
Sodiq-900000
Gerry Mahesa-900000
Mella Barbie-800000
```

- ✓ **SUBSTRING(string, awal, panjang)** : untuk mengambil sebuah dengan memotong string dari nilai awal dan panjang string yang dimunculkan sesuai keinginan. Contohnya :

```
SELECT SUBSTRING('Dimas Edubuntu Samid',1, 15);
```

Hasilnya :

```
Dimas Edubuntu
```

- ✓ **LENGTH(string)** : menghitung panjang string. Contohnya :

```
SELECT LENGTH('Vivi Rosalita');
```

Hasilnya :

```
13
```

- ✓ **LEFT(string, panjang)** : untuk memotong string dimulai dari sebelah kiri string. Contohnya :

```
SELECT LEFT('Ega Azhari', 3) AS Penyanyi;
```

Hasilnya :

```
Ega
```

- ✓ **RIGHT(awal, panjang)** : kebalikan dari LEFT. Memotong string dimulai dari sebelah kanan string. Contohnya :

```
SELECT RIGHT('Lilin Herlina', 4) AS 'Dari Kanan';
```

Hasilnya :

```
Lina
```

- ✓ **LTRIM(String)** : menghilangkan spasi di sebelah kiri / sebelum string. Contohnya :

```
SELECT LTRIM(' New Palapa');
```

Hasilnya :

```
New Palapa
```

- ✓ **RTRIM(String)** : menghilangkan spasi di sebelah kanan / sesudah string. Contohnya :

```
SELECT RTRIM('New Palapa ');
```

Hasilnya :

```
New Palapa
```

- ✓ **TRIM(String)** : menghilangkan spasi di awal dan di akhir string. Contohnya :

```
SELECT TRIM(' New Palapa ');
```

Hasilnya :

```
New Palapa
```

- ✓ **REPLACE(String, 'karakter_yang_diganti', 'karakter_penggantinya')** : mengganti sebuah atau beberapa karakter dengan karakter lain. Contohnya :

```
SELECT REPLACE('Aja Ngono', 'Aja', 'Ojo');
```

Hasilnya :

```
Ojo Ngono
```

- ✓ **REPEAT(String, jumlah_repeat)** : untuk mengulang sebuah string sesuai dengan jumlah yang diinginkan. Contohnya :

```
SELECT REPEAT('Blankon', 3);
```

Hasilnya :

```
BlankonBlankonBlankon
```

- ✓ **REVERSE(String)** : untuk membalik sebuah string. Contohnya :

```
SELECT REVERSE('Dimas Edubuntu Samid');
```

Hasilnya :

```
dimas utnubudE samid
```

- ✓ **LCASE(String), LOWER(String)** : keduanya memiliki fungsi yang sama, yaitu menjadikan string ke dalam bentuk lowercase. Contohnya :

```
SELECT LCASE('Gerry Mahesa');
```

Hasilnya :

```
Gerry mahesa
```

- ✓ **UCASE(String), UPPER(String)** : keduanya memiliki fungsi untuk menjadikan string ke dalam bentuk UPPERCASE. Contohnya :

```
SELECT UPPER('bambang pamungkas');
```

Hasilnya :

```
GERRY MAHESA
```

Fungsi Numeric

Sama seperti namanya fungsi ini diperuntukkan untuk mengolah data yang berbentuk numeric / angka. Dan berikut adalah beberapa fungsi Numeric :

- ✓ **ABS(nilainya)** : mengambil nilai absolute dari sebuah bilangan. Contohnya :

```
SELECT ABS(-1500);
```

Hasilnya

```
1500
```

- ✓ **MOD(a, b)** : melakukan operasi modulus atau sisa pembagian dari dua buah bilangan a dan b. Contohnya :

```
SELECT MOD(15, 3);
```

Hasilnya :

```
0
```

- ✓ **FLOOR(x)** : mengambil nilai integer terbesar dan tidak lebih besar nilainya dari x. Contohnya :

```
SELECT FLOOR(12.785);
```

Hasilnya :

```
12
```

- ✓ **CEILING(x)** : mengambil nilai integer terkecil dan tidak lebih kecil nilainya dari x. Contohnya :

```
SELECT FLOOR(12.785);
```

Hasilnya :

```
13
```

- ✓ **ROUND (x, d)** : untuk membulatkan bilangan x dengan sebanyak d tempat presisi (dibelakang koma). Contohnya :

```
SELECT ROUND(25.0583);
```

Hasilnya :

```
25.06
```

- ✓ **POW(x,n), POWER(x,n)** : melakukan pemangkatan dari sebuah bilangan x dengan pangkat n, X^n . Contohnya :

```
SELECT POWER (4, 2) ;
```

Hasilnya :

```
16
```

- ✓ **RAND()** : mengambil nilai random / secara acak dengan range nilai dari 0 sampai dengan 1.0. Hasil dari contoh tidak selamanya bernilai 0.4533713520070271. Intinya, sistem akan menampilkan bilangan secara acak dengan range nilai yang sudah disebutkan tadi. Contohnya:

```
SELECT RAND () ;
```

Hasilnya :

```
0.4533713520070271
```

- ✓ **TRUNCATE(x,d)** : melakukan pembulatan dari bilangan x tetapi membulatkan ke bawah atau mendekati 0 dengan d panjang desimal dibelakang koma. Contohnya :

```
SELECT TRUNCATE (4.27943) ;
```

Hasilnya :

```
4.27
```

- ✓ **SQRT(x)** : mengambil nilai akar dari bilangan x. Contohnya :

```
SELECT SQRT (100) ;
```

Hasilnya :

```
10
```

Fungsi Tanggal dan Waktu

MySQL juga menyediakan fungsi – fungsi yang berhubungan dengan tanggal dan waktu. Berikut adalah fungsi-fungsi Tanggal dan Waktu :

- ✓ **NOW(), SYSDATE()** : untuk mendapatkan Tanggal dan Waktu saat ini. Menyesuaikan dengan Tanggal dan Waktu komputer sistem MySQL. Contohnya :

```
SELECT NOW();
```

Hasilnya

```
2012-10-10 13:56:00
```

- ✓ **DAY(tanggal)** : untuk mengambil tanggal pada sebuah baris penanggalan (YYYY-MM-DD). Nilai dari tanggal ini adalah integer. Contohnya :

```
SELECT DAY('2012-08-17');
```

Hasil perintah tersebut :

```
17
```

- ✓ **MONTH(tanggal)** : untuk mendapatkan bulan dalam sebuah Tanggal (YYYY-MM-DD). Nilai dari bulan ini adalah integer. Contohnya :

```
SELECT MONTH('2012-08-17');
```

Hasilnya :

```
8
```

- ✓ **YEAR(tanggal)** : mendapatkan tahun dalam sebuah penanggalan (YYYY-MM-DD). Nilai dari tanggal ini adalah integer. Contohnya :

```
SELECT YEAR('2012-08-17');
```

Hasilnya :

```
2012
```

- ✓ **WEEK(tanggal)** : mendapatkan urutan minggu dari sebuah tanggal (YYYY-MM-DD).

Contohnya:

```
SELECT WEEK('2012-08-17');
```

Hasilnya :

```
33
```

- ✓ **HOOR(waktu)** : mendapatkan jam dari sebuah waktu (h:i:s). Contohnya :

```
SELECT HOUR('23:45:03');
```

Hasil perintah tersebut :

```
23
```

- ✓ **MINUTE(waktu)** : mendapatkan menit dari sebuah waktu (h:i:s). Contohnya :

```
SELECT MINUTE('23:45:03');
```

Hasilnya :

```
45
```

- ✓ **SECOND(waktu)** : mendapatkan detik dari sebuah waktu(h:i:s). Contohnya :

```
SELECT SECOND('23:45:03');
```

Hasilnya :

```
03
```

- ✓ **DATE_ADD(waktu,interval,tipe), DATE_SUB(waktu,interval,tipe), DATESUB(waktu, interval, tipe), ADDDATE(waktu, interval,tipe)** : mempunyai fungsi untuk menambahkan Tanggal dan waktu dengan interval atau jarak yang diinginkan, dibarengi dengan bentuk atau tipe dari interval tersebut.

Berikut adalah tabel tipe interval :

<type>	<expression> format
MICROSECOND	<microseconds>
SECOND	<seconds>
MINUTE	<minutes>
HOUR	<hours>
DAY	<days>
MONTH	<months>
YEAR	<years>
SECOND_MICROSECOND	'<seconds>.<microseconds>'
MINUTE_MICROSECOND	'<minutes>.<microseconds>'
MINUTE_SECOND	'<minutes>:<seconds>'
HOUR_MICROSECOND	'<hours>.<microseconds>'
HOUR_SECOND	'<hours>:<minutes>:<seconds>'
HOUR_MINUTE	'<hours>:<minutes>'
DAY_MICROSECOND	'<days>.<microseconds>'
DAY_SECOND	'<days> <hours>:<minutes>:<seconds>'
DAY_MINUTE	'<days> <hours>:<minutes>'
DAY_HOUR	'<days> <hours>'
YEAR_MONTH	'<years>-<months>'

Gambar 5. Tabel tipe interval. Ref: Wrox - Beginning MySQL

Contohnya :

```
SELECT ADDDATE(NOW(), INTERVAL '02:30' HOUR_MINUTE);
```

Hasil perintah tersebut :

```
2012-10-10 16:55:34
```

- ✓ **DATE_FORMAT(tanggal, '%tipe tanggal')**, **TIME_FORMAT(waktu, '%tipe waktu')** : keduanya memiliki fungsi untuk melakukan format sebuah bentuk tanggal dan waktu kedalam tipe yang diinginkan. Berikut daftar tipe format tanggal dan waktu :

Tipe	Deskripsi
%Y	Bentuk tipe tahun 4 digit . Ex : 2012
%y	Bentuk tipe tahun 2 digit. Ex : 12
%M	Bentuk tipe bulan (January, February, March,..)
%m	Bentuk tipe bulan Integer, 2 digit. Ex : 02
%D	Bentuk tipe tanggal ke- . Ex : 10 th , 11 th ...
%d	Bentuk tipe tanggal, 2 digit. Ex : 10, 11, 12 ...
%W	Bentuk tipe hari dalam satu minggu. Ex : Sunday, Saturday,..
%a	Bentuk tipe hari dalam satu minggu, 3 karakter. Ex : Sun, Thu, Wed
%H	Bentuk tipe jam, 24 Jam. Ex : 23 (jam 11 malam)
%i	Bentuk tipe menit.
%s	Bentuk tipe detik.

Contohnya :

```
SELECT DATE_FORMAT (now(), '%d-%M-%Y %H:%i:%s');
```

Hasilnya :

```
11-October-2012 21:05:15
```

- ✓ **DATEDIFF('tanggal_awal', 'tanggal_akhir')** : untuk melakukan perhitungan jarak antara tanggal awal dengan tanggal akhir sedangkan **TIMEDIFF('waktu_awal', 'waktu_akhir')** : untuk melakukan perhitungan jarak antara waktu awal dengan waktu akhir. Contohnya :

```
SELECT TIMEDIFF('20:25:45', '20:15:30');
```

hasilnya :

```
00:10:15
```

Contoh lagi:

```
SELECT DATEDIFF('2012-10-11', '2012-10-09');
```

hasilnya :

```
2
```

Fungsi Lainnya

Berikut adalah beberapa fungsi yang sangat berguna bagi sobat semua ketika melakukan pengolahan data pada MySQL. Kita awali dengan ...

- ✓ **GREATEST(x)** : mengambil nilai terbesar dari sebuah kelompok nilai. Contohnya :

```
SELECT GREATEST(4, 5, 8, 100, 25, 89, 2, 15) ;
```

Hasilnya :

```
100
```

- ✓ **COUNT(x)** : mengambil jumlah baris dari suatu query. Contohnya :

```
SELECT COUNT(*) FROM tbl_karyawan;
```

Hasilnya :

```
8
```

- ✓ **MIN(x)** : mengambil nilai terkecil dari suatu query. Contohnya :

```
SELECT MIN(gapok) FROM tbl_karyawan;
```

Hasilnya :

```
50000
```

- ✓ **MAX(x)** : mengambil nilai terbesar dari suatu query. Contohnya :

```
SELECT MAX(gapok) FROM tbl_karyawan;
```

Hasilnya :

```
900000
```

- ✓ **SUM(x)** : menghitung total dari sebuah query. Contohnya :

```
SELECT SUM(gapok) AS 'Total Gaji Karyawan' FROM tbl_karyawan;
```

Hasilnya :

```
4800000
```

- ✓ **AVG(x)** : mendapatkan nilai rata-rata dari suatu query. Contohnya :

```
SELECT AVG(gapok) AS 'Rata2 Gaji per Karyawan' FROM tbl_karyawan;
```

Hasilnya :

```
600000
```

Dan masih banyak lagi fungsi-fungsi yang dapat membantu sobat untuk melakukan manajemen data pada MySQL. Sobat dapat mencari referensi baik melalui blog ataupun banyak situs yang tersebar di dunia maya. Seperti kaya saya jangan pernah berhenti untuk mencari dan mencari. Seperti kata pak Dahlan Iskan “Kerja, Kerja dan Kerja”. Hehehe :) Insya Allah, saya niatnya mau menulis buku tentang MySQL yang lebih detail untuk Pemula sampai Mahir dan masih tetap platform Linux, mohon doanya semua.

Pada chapter ini saya tidak sertakan soal latihan karena saya pikir soal latihan sudah tertera pada beberapa contoh yang sudah ada. Tapi jangan khawatir pada lampiran buku ini akan saya beri beberapa latihan agar sobat semua mudah dan paham mempelajari MySQL ini. Oke, kita akan lanjut ke Chapter berikutnya.

Chapter 8

Akses Data antar Tabel

Chapter ini kita akan mempelajari bagaimana hubungan antar tabel pada MySQL. Hal ini cukup penting karena jika anda membuat sebuah aplikasi yang cukup besar, hubungan antar tabel menjadi hal yang utama. Oleh karena itu, sobat harus memahami benar tentang Relasi pada sebuah database.

MySQL sebagai salah satu Relationship database management System memberikan fitur relasi antar tabel yang sangat mudah anda pelajari dan kuasai serta kemudahan dalam hal penggunaan. Untuk menyingkat waktu baiklah kita mulai sekarang dengan...

Relasi Tabel

Sebelum sobat mempelajari tentang Relasi antar tabel, coba perhatikan pengertian Database berikut, sedikit berteori dulu, heheheh.... . Database adalah sekumpulan tabel yang saling terkait satu sama lain (berelasi), disusun secara logis, guna menghasilkan sebuah informasi. "*Tabel yang saling terkait atau berelasi*" sudah jelaskan?? Yups, itulah Database. Database berisi tabel - tabel dimana mereka saling berkomunikasi satu sama lain atau istilahnya adalah saling memiliki hubungan atau relasi.

Hubungan antar tabel tersebut menjadikan sebuah database akan semakin powerfull kita digunakan. Dari pengalaman saya mengerjakan sebuah project, memang benar, relasi yang tidak bagus alias tidak disusun secara logis sesuai dengan alur logika algoritmanya bisa dipastikan akan berhasil dengan kegagalan. Karena nyawa sebenarnya dari sebuah aplikasi adalah pada Database. Terkadang ketika kita salah dalam merancang sebuah database bukan lagi sebuah bottle-neck saja yang akan terjadi tetapi stack program (program macet) karena relasi yang salah dan tidak efisien.

Jadi, saran saya sebelum sobat mengerjakan sebuah project terlebih dahulu rancang-lah sebuah database yang efektif dan efisien disertai logika yang benar, saya juga masih terus belajar agar semakin mumpuni dalam membuat database yang efektif dan efisien. Tetapi bukan berarti sobat

melupakan script / source code yang akan sobat tulis pada aplikasi, harus sama-sama diperhatikan. Baiklah kita mulai saja dengan Secara singkat relasi pada database ada tiga bentuk :

1. **One to One**, merupakan bentuk relasi antar 2 tabel dengan Primary Key (PK) dan Foreign Key(FK). Relasi bentuk ini jarang sekali digunakan tetapi ada beberapa alasan relasi jenis ini dapat digunakan :
 - Memindahkan data ke tabel lain yang memungkinkan untuk membuat query berjalan lebih cepat.
 - Mengisolasi dan menghindari nilai NULL pada tabel utama.
2. **One to Many**, sebuah relasi dimana Entity A berelasi ke beberapa record pada entity B, sedangkan entity B hanya dapat ber-relasi dengan satu record pada entity A. Sebagai contoh adalah Entity A adalah Dosen, sedangkan Entity B adalah mata kuliah. Dosen dapat mengajar lebih dari satu matakuliah, sedangkan setiap matakuliah hanya diajar oleh seorang dosen. Pahami maksudnya?
Dalam diagram ER (Entity Relationship) one dinyatakan dengan simbol 1 yang artinya adalah satu. Sedangkan Many disimbolkan dengan M atau banyak. One to Many, satu ke banyak.
3. **Many to Many**, sebuah relasi dimana entity A dapat ber-relasi ke beberapa record pada entity B, begitu juga sebaliknya record pada entity B dapat ber-relasi ke beberapa record pada Entity A. Sebagai contohnya kembali ke Dosen lagi, Entity A adalah dosen, sedangkan Entity B adalah mahasiswa. Dosen (A) dapat mengajar beberapa mahasiswa(B) dan satu mahasiswa(B) dapat diajar oleh lebih dari satu dosen (A).
Dalam diagram ER Many disimbolkan dengan M atau N untuk menyatakan banyak.

Bagaimana sobat? sudah jelas tentang relasi antar tabel? Kalo belum bisa cari referensi lainnya atau bisa juga langsung bertanya sama saya. Dan mudah-mudahan dapat menjawab pertanyaan sobat semua. Memang relasi antar tabel butuh secara dipelajari tersendiri. Karena mengingat begitu kompleksnya bahasan tentang relasi antar tabel.

Namun, ini sebagai gambaran saja karena nantinya kita akan belajar mengenai mengakses data dari beberapa tabel. Dan mungkin akan mendapat pencerahan ketika sobat sudah mempelajari beberapa contoh yang ada. Mari kita lanjutkan dengan....

Mengakses data dari 2 Tabel

Disini akan dibahas mengenai akses / mengambil data dari dua tabel yang saling ber-relasi / berhubungan. Tentunya akan semakin kompleks dan butuh kesebaran ekstra untuk memahami, tapi jangan khawatir saya akan berikan beberapa contoh yang saya jamin sobat semua dapat mempelajari dan memahami maksud dari pembahasan ini.

Oke, sebagai contoh kita akan mempelajari sebuah studi kasus agar sobat dapat lebih mudah dalam memahami bagaimana mengakses data dari 2 tabel yang berbeda. Dalam kasus ini adalah database pada sebuah klinik, dimana ada 2 tabel, yaitu Dokter dan Resep yang diberikan ke pasien.

Jadi, ceritanya resep yang diberikan ke pasien akan menunjukkan dokter mana yang memberikan resep dan nantinya kita akan melakukan filtering / penyaringan data sesuai dengan dokter yang tertera pada resep. Lets cekidot.

Buatlah database terlebih dahulu dengan nama db_klinik. Kemudian buat dua tabel yang pertama adalah tabel t_dokter dengan struktur sebagai berikut :

Nama Field	Tipe Data	Panjang	Null	Primary	Extra
id_dokter	VARCHAR	10		Ya	
nama_dokter	VARCHAR	100			
alamat	TEXT		NULL		
telp	VARCHAR	50	NULL		

Dan tabel `t_resep` dengan struktur sebagai berikut :

Nama Field	Tipe Data	Panjang	Null	Primary	Extra
<code>no_resep</code>	VARCHAR	30		Ya	
<code>tanggal_resep</code>	DATE				
<code>nama_pasien</code>	VARCHAR	100			
<code>id_dokter</code>	VARCHAR	10			

Setelah sobat buat tabelnya kemudian isi dengan data berikut pada tabel `t_dokter` :

<code>id_dokter</code>	<code>nama_dokter</code>	<code>alamat</code>	<code>telp</code>
D001	dr. Bambang Priyo, Sp.B	Slawi	081112255667
D002	dr. Tri Kunjana, Sp.THT	Jl. Melati No 18 Tegal	
D003	dr. Endah	Jl. Jawa No. 90 Tegal	0283-333335
D004	dr. Iman Darjito, Sp.PD	Brebes	
D005	dr. Bambang Susilo, Sp.A	Pemalang	
D006	dr. Munim	Slawi	

Kemudian isi tabel `t_resep` dengan data berikut :

<code>no_resep</code>	<code>tanggal</code>	<code>pasien</code>	<code>id_dokter</code>
R201210001	2012-10-10	Tn. Sodiq	D001
R201210002	2012-10-10	An. Irmayanti	D005
R201210003	2012-10-11	Tn. Dimas Edu	D002
R201210004	2012-10-12	An. Dwi Ratna	D005
R201210005	2012-10-12	Ny. Lilin H	D006
R201210006	2012-10-13	Ny. Susi	D003
R201210007	2012-10-13	Tn. Brodien	D004
R201210008	2012-10-14	Tn. Yusup	D001
R201210009	2012-10-15	Ny. Shania	D001
R201210010	2012-10-15	Ny. Via Vallent	D002

Bentuk umum akses data antar 2 tabel

Nah, setelah sobat membuat tabel dan memasukan datanya kedalam tabel. Kali ini kita akan memunculkan data resep sekaligus nama dokter yang diambil dari tabel dokter. Seperti yang sobat lihat, pada tabel t_resep tidak tercantum nama dokter , hanya tertera id_dokter saja. Itulah PR nya permisah.. lalu bagaimana caranya ya?..

Secara umum command untuk memunculkan data dari dua tabel adalah sebagai berikut :

```
SELECT tabel1.*, tabel2.* FROM tabel1, tabel2  
WHERE field_key.tabel1 = field_key.tabel2;
```

Cobalah query berikut pada command-line MySQL sobat semua :

```
SELECT t_resep.no_resep,  
DATE_FORMAT(t_resep.tanggal, '%d %M %Y') AS tanggal,  
t_resep.pasien,  
t_dokter.nama_dokter  
FROM t_resep, t_dokter  
WHERE t_resep.id_dokter = t_dokter.id_dokter  
ORDER BY t_resep.no_resep ASC;
```

Hasil dari query tersebut adalah :



no_resep	tanggal	pasien	nama_dokter
R201210001	10 October 2012	Tn. Sodiq	dr. Bambang Priyo, Sp.B
R201210002	10 October 2012	An. Irmayanti	dr. Bambang Susilo, Sp.A
R201210003	11 October 2012	Tn. Dimas Edu	dr. Tri Kunjana, Sp.THT
R201210004	12 October 2012	An. Dwi Ratna	dr. Bambang Susilo, Sp.A
R201210005	12 October 2012	Ny. Lilin H	dr. Munim
R201210006	13 October 2012	Ny. Susi	dr. Endah
R201210007	13 October 2012	Tn. Brodien	dr. Iman Darjito, Sp. PD
R201210008	14 October 2012	Tn. Yusup	dr. Bambang Priyo, Sp.B
R201210009	15 October 2012	Ny. Shania	dr. Bambang Priyo, Sp.B
R201210010	15 October 2012	Ny. Via Vallent	dr. Tri Kunjana, Sp.THT

Oke, pada gambar tersebut nama_dokter muncul karena kita menghubungkan t_resep dengan t_dokter melalui sebuah relasi antar **id_dokter**.

Itulah bentuk sederhana mengambil data dari dua tabel yang memiliki relasi. Nah, untuk mengolah data yang lebih kompleks, ada beberapa fitur yang disediakan oleh MySQL, yaitu dengan JOIN.

Dibawah ini akan dijelaskan beberapa JOIN yang harus anda ketahui untuk memudahkan dalam melakukan pengolahan data pada MySQL :

CROSS JOIN

Cross Join merupakan bentuk sederhana dari tanpa melibatkan / menambahkan sebuah kondisi pada querynya. Bentuk umum Cross Join adalah :

```
SELECT field FROM tabel1 CROSS JOIN tabel2;
```

INNER JOIN

Inner Join hampir sama dengan Cross Join, tetapi ada penambahan kondisi pada query. Bentuk umum dari Inner Join adalah :

```
SELECT field FROM tabel1 INNER JOIN tabel2 ON kondisi;
```

STRAIGHT JOIN

Straight Join identik dengan Inner Join tetapi tidak mengenal klausa where. Bentuk umum dari Straight Join adalah :

```
SELECT field FROM tabel1 STRAIGHT JOIN ON tabel2;
```

LEFT JOIN

Left Join akan memunculkan record pada tabel sebelah kanan dengan NULL jika tabel sebelah kanan tidak ada hubungan dengan tabel sebelah kiri. Bentuk umum dari Left Join adalah :

```
SELECT field FROM tabel1 LEFT JOIN tabel2 ON kondisi;
```

RIGHT JOIN

Right Join adalah kebalikan dari Left Join. Tabel yang menjadi acuan adalah tabel sebelah kanan, sehingga data akan tetap dimunculkan meski data pada tabel sebelah kanan tidak ada kaitan dengan data pada tabel sebelah kiri. Bentuk umum dari Right Join adalah :

```
SELECT field FROM tabel1 RIGHT JOIN tabel2 ON kondisi;
```

Oke, itulah beberapa macam join yang ada pada MySQL. Semua join tersebut berfungsi untuk memunculkan data yang ada pada beberapa tabel yang memiliki sebuah relasi.

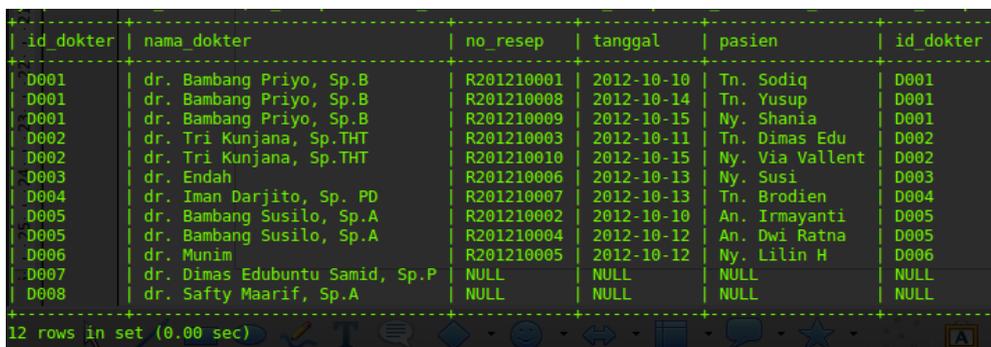
Nah, untuk memudahkan sobat memahami bentuk join tersebut kita coba saja dengan contoh berikut. Coba buka kembali db_klinik yang sudah dibuat, kemudian tambahkan dua record / data pada tabel t_dokter dengan perintah berikut :

```
INSERT INTO t_dokter VALUES  
( 'D007', 'dr. Dimas Edubuntu Samid, Sp.P'),  
( 'D008', 'dr. Safty Maarif, Sp.A');
```

dan coba query berikut pada command-line MySQL sobat :

```
SELECT t_dokter.*, t_resep.* FROM t_dokter  
LEFT JOIN t_resep  
ON t_dokter.id_dokter = t_resep.id_dokter;
```

Hasilnya :



id_dokter	nama_dokter	no_resep	tanggal	pasien	id_dokter
D001	dr. Bambang Priyo, Sp.B	R201210001	2012-10-10	Tn. Sodik	D001
D001	dr. Bambang Priyo, Sp.B	R201210008	2012-10-14	Tn. Yusup	D001
D001	dr. Bambang Priyo, Sp.B	R201210009	2012-10-15	Ny. Shania	D001
D002	dr. Tri Kunjana, Sp.THT	R201210003	2012-10-11	Tn. Dimas Edu	D002
D002	dr. Tri Kunjana, Sp.THT	R201210010	2012-10-15	Ny. Via Vallent	D002
D003	dr. Endah	R201210006	2012-10-13	Ny. Susi	D003
D004	dr. Iman Darjito, Sp. PD	R201210007	2012-10-13	Tn. Brodien	D004
D005	dr. Bambang Susilo, Sp.A	R201210002	2012-10-10	An. Irmayanti	D005
D005	dr. Bambang Susilo, Sp.A	R201210004	2012-10-12	An. Dwi Ratna	D005
D006	dr. Munim	R201210005	2012-10-12	Ny. Lilin H	D006
D007	dr. Dimas Edubuntu Samid, Sp.P	NULL	NULL	NULL	NULL
D008	dr. Safty Maarif, Sp.A	NULL	NULL	NULL	NULL

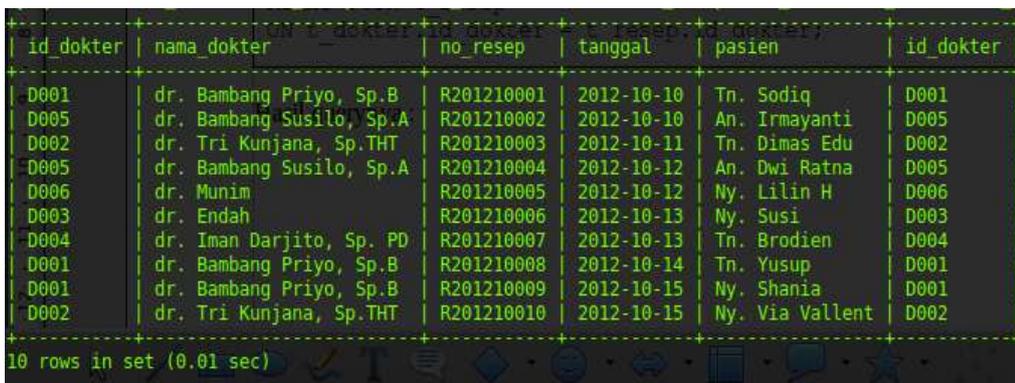
Perhatikan gambar hasil query tersebut, ada field-field yang mempunyai nilai NULL. Seperti yang sudah dijelaskan pada point diatas bahwa LEFT JOIN akan memunculkan data pada tabel kanan dengan nilai NULL jika tidak ada hubungan dengan record pada tabel kanan.

Dengan kata lain dokter tersebut tidak pernah memberikan resep kepada pasien siapapun, karena menggunakan LEFT JOIN kedua dokter tersebut akan tetap dimunculkan walaupun tidak pernah melakukan transaksi yang tersimpan pada tabel resep.

Sekarang coba contoh yang ini,

```
SELECT t_dokter.*, t_resep.* FROM t_dokter  
RIGHT JOIN t_resep  
ON t_dokter.id_dokter = t_resep.id_dokter;
```

Hasil querynya :



id_dokter	nama_dokter	no_resep	tanggal	pasien	id_dokter
D001	dr. Bambang Priyo, Sp.B	R201210001	2012-10-10	Tn. Sodiq	D001
D005	dr. Bambang Susilo, Sp.A	R201210002	2012-10-10	An. Irmayanti	D005
D002	dr. Tri Kunjana, Sp.THT	R201210003	2012-10-11	Tn. Dimas Edu	D002
D005	dr. Bambang Susilo, Sp.A	R201210004	2012-10-12	An. Dwi Ratna	D005
D006	dr. Munim	R201210005	2012-10-12	Ny. Lilin H	D006
D003	dr. Endah	R201210006	2012-10-13	Ny. Susi	D003
D004	dr. Iman Darjito, Sp. PD	R201210007	2012-10-13	Tn. Brodien	D004
D001	dr. Bambang Priyo, Sp.B	R201210008	2012-10-14	Tn. Yusup	D001
D001	dr. Bambang Priyo, Sp.B	R201210009	2012-10-15	Ny. Shania	D001
D002	dr. Tri Kunjana, Sp.THT	R201210010	2012-10-15	Ny. Via Vallent	D002

Tahu bedanya kan? Yang ini tidak terdapat NULL didalamnya karena RIGHT JOIN memfokuskan pada tabel sebelah kanan. Sehingga transaksi yang tidak ada hubungannya dengan tabel sebelah kiri tidak akan dimunculkan.

Pengelompokkan Data

Mungkin sobat semua pernah melihat sebuah laporan / report pada sebuah aplikasi. Ya, paling gampang kalo sobat belanja ke Indomaret atau Alfamart (wah harus dapet fee masangin iklannya .. hehehehe) pasti mendapatkan lembaran kertas alias struk berupa daftar belanjaan yang sekaligus ada jumlah total harga tiap item serta jumlah keseluruhan yang harus sobat bayar. Coba sobat bayangkan (ya saya ajak sobat berpikiri), setiap item memunculkan harga dan kemudian akan dihitung total keseluruhan dari item yang anda beli.

Logikannya kan, mereka (si item belanjaan itu) dikelompokkan berdasarkan mungkin kode barang dan kemudian akan dihitung jumlah keseluruhan sehingga menghasilkan total yang harus

sobat bayar. Nah, itulah yang jadi PR kita nantinya. Sobat akan saya ajak belajar mengenal dan mengetahui bagaimana fitur pengelompokkan data yang ada pada MySQL ini bekerja. Marilah kita langsung ke Te... Kaa.... Pe... Yaa.. eeee.....

GROUP BY

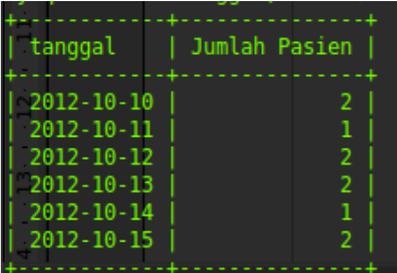
Ini dia sobat fitur yang diberikan oleh MySQL yang berfungsi untuk mengelompokkan data. GROUP BY berfungsi mengelompokkan data yang disesuaikan dengan data record yang sejenis. Bentuk umum dari GROUP BY adalah :

```
SELECT field FROM tabel GROUP BY kondisi;
```

Sebagai contoh kita akan menampilkan berapa jumlah pasien berdasarkan tanggal. Nah tentunya kita harus mengelompokkan Data Resep berdasarkan tanggal. Coba tulis pada command-line MySQL sobat perintah berikut:

```
SELECT tanggal, COUNT(tanggal) AS 'Jumlah Pasien'  
FROM t_resep GROUP BY tanggal;
```

Hasilnya :



tanggal	Jumlah Pasien
2012-10-10	2
2012-10-11	1
2012-10-12	2
2012-10-13	2
2012-10-14	1
2012-10-15	2

Jumlah pasien didapat dari COUNT(tanggal) yang nantinya akan dikelompokkan berdasarkan tanggal. Sehingga masing-masing tanggal memunculkan jumlah pasiennya disesuaikan dengan jumlah transaksi yang ada pada tabel t_resep.

Bagaimana mudah bukan? Bagi sobat yang masih awam dalam dunia programming, tips dengan penggabungan record menggunakan GROUP BY sering dilakukan para programmer yang biasanya digunakan untuk memunculkan data pada fitur report sebuah transaksi.

Oleh karena itu, agar sobat semakin memahami cara kerja relasi antar tabel serta pengelompokkan data mari kita berlatih dengan soal dibawah ini.

SOAL LATIHAN

1. Buatlah sebuah database dengan nama **penjualan**.
2. Buatlah tabel **marketing** dengan struktur sebagai berikut :

Nama Field	Tipe Data	Panjang	Null	Primary	Extra
kd_marketing	VARCHAR	5		Ya	
nama	VARCHAR	150			
no_hape	VARCHAR	25			
area	VARCHAR	5			

3. Buatlah tabel penjualan dengan struktur sebagai berikut :

Nama Field	Tipe Data	Panjang	Null	Primary	Extra
id_jual	VARCHAR	10		Ya	
tgl_jual	DATE				
kd_marketing	VARCHAR	5			
qty	INT	11			

4. Isilah tabel marketing dengan data berikut :

Kode	Nama	No. Hape	Area
M01	DUL JONI	081500000678	TGL
M02	GUFRON M	081888888564	SLW
M03	JAFFAR	081255648557	TGL
M04	SHANIA T	081322556689	CRB
M05	INDAH PUSPITA SARI N	085556655548	TGL

5. Isilah tabel penjualan dengan data berikut :

Kode	Tanggal	Marketing	QTY
F0025	2012-02-25	M02	25
F0067	2012-03-20	M01	10
F0098	2012-04-05	M04	20
F0106	2012-04-10	M02	26
F0135	2012-10-06	M06	510

6. Tampilkan data dari dua tabel tersebut dengan menggunakan bentuk query yang sederhana.
7. Tampilkan data dengan menggunakan LEFT JOIN dan RIGHT JOIN.
8. Tampilkan data dengan menggunakan CROSS JOIN dan INNER JOIN.
9. Tampilkan total penjualan dari masing area marketing.
10. Tampilkan total keseluruhan penjualan.

Chapter 9

Management User MySQL

Pada chapter ini kita akan sama - sama membahas mengenai mengelola pengguna pada database MySQL. Seperti halnya rumah yang memiliki banyak penghuni (bayangkan saja rumah yang saat ini ditinggali oleh sobat semua), biasanya ada beberapa orang yang diberi kunci rumah (biasanya yang sering pulang malem punya kunci cadangan untuk masuk ke rumah). Dan tentunya penghuni rumah tersebut apalagi yang empunya rumah punya akses yang luar biasa terhadap rumahnya.

Tetapi lain halnya bagi tamu, atau mungkin sodara yang tinggal dirumah sobat, tentunya tidak se bebas dan semau gue beraktivitas dirumah sobat. Ada batasan - batasan tertentu dimana kadang jika sodara tersebut diberi kepercayaan menjaga rumah pasti diberi akses yang luar biasa.

Nah, itulah yang akan kita pelajari. Bagaimana membuat penghuni di dalam "rumah" MySQL. Serta siapa saja yang akan diberi hak-hak istimewa dan mana yang hanya sekedar bertamu saja.

Mengenal jenis hak akses pada MySQL

Pada sistem MySQL terdapat sebuah tabel - tabel dimana kita bisa melakukan manajemen user. Setiap tabel memiliki fungsinya masing-masing. Untuk lebih jelasnya silahkan coba buka database mysql kemudian tampilkan tabel yang ada pada database mysql.

```
USE mysql;
```

```
SHOW TABLES;
```

Jika perintah tersebut berhasil dijalankan seharusnya sobat akan mendapati seperti gambar dibawah ini,

```
Tables_in_mysql
|
| columns_priv
| db
| event
| func
| general_log
| help_category
| help_keyword
| help_relation
| help_topic
| host
| ndb_binlog_index
| plugin
| proc
| procs_priv
| proxies_priv
| servers
| slow_log
| tables_priv
| time_zone
| time_zone_leap_second
| time_zone_name
| time_zone_transition
| time_zone_transition_type
| user
```

Terlihat banyak sekali tabel pada database mysql, dari semua tabel tersebut ada beberapa tabel yang sangat penting dalam melakukan manajemen user, diantaranya :

- **user**, tabel ini Berisi data user yang mendapatkan izin akses MySQL, asal koneksi dan izin akses kepada user.
- **db**, tabel ini Mengatur database apa saja yang dapat diakses oleh seorang user dan jenis izin aksesnya.
- **host**, Mengatus asl host yang diperkenankan bagi user untuk mengakses MySQL, jika lebih dari satu host.
- **tables_priv**, Mengatur tabel apa saja yang dapat diakses oleh seorang user dan jenis izin aksesnya.
- **coloumn_priv**, Mengatur kolom (field) apa saja yang dapat diakses oleh seorang user dan jenis izin aksesnya.
- **proc_priv**, menyimpan informasi mengenai hak akses user terhadap procedure.
- **func**, menyimpan fungsi yang difenisikan oleh sistem MySQL.
- **proc**, menyimpan daftar prosedur dalam Sistem MySQL.

Jenis Hak akses (privilage) user

Selain itu ada juga beberapa hak akses / privilage yang perlu sobat ketahui yaitu,

1. Tingkatan akses user biasa

Mencakup izin akses kedalam database atau kolom, antara lain :

- a. ALTER
- b. CRETATE
- c. DELETE
- d. DROP
- e. INDEX
- f. INSERT
- g. SELECT
- h. UPDATE
- i. REFERENCES

2. Tingkatan akses administrator -Global administrative

Hanya digunakan oleh user setingkat root atau administrator dan tidak diberikan kepada user biasa, yaitu :

- a. FILE
- b. PROCESS
- c. RELOAD
- d. SHUTDOWN
- e. CREATE TEMPORARY TABLE
- f. EXCUTE
- g. LOCK TABLES
- h. REPLICATION CLIENT
- i. REPLICATION SLAVE
- j. SHOW DATABASES
- k. SUPER

3. Tingkatan Akses khusus - Special privileges

Dapat diterapkan pada setiap user dengan izin akses sebagai berikut :

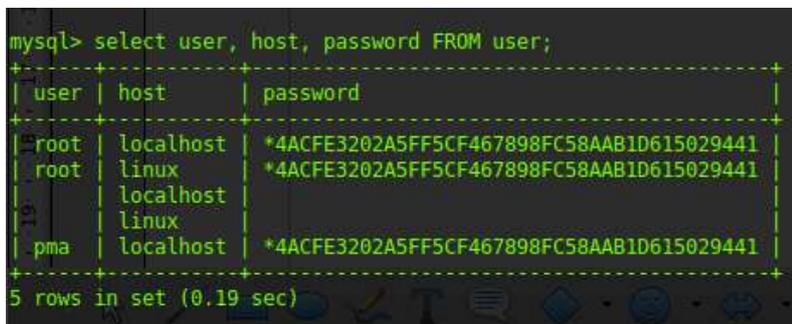
- a. ALL
- b. USAGE

Menghapus user Anonym

Nah, ini dia yang berbahaya. User anonym yang aktif dapat merusak sistem kita, karena setiap orang bisa bebas keluar masuk ke dalam sistem kita. Untuk mengetahui apakah ada user anonym atau tidak, coba periksa dengan menggunakan perintah :

```
SELECT user, host, password FROM user;
```

maka akan tampak seperti berikut :



```
mysql> select user, host, password FROM user;
+-----+-----+-----+
| user | host      | password |
+-----+-----+-----+
| root | localhost | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
| root | linux     | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
|      | localhost |          |
|      | linux     |          |
| pma  | localhost | *4ACFE3202A5FF5CF467898FC58AAB1D615029441 |
+-----+-----+-----+
5 rows in set (0.19 sec)
```

Owwwww.. tidak. Terlihat pada gambar tersebut ada user anonymous. Dicitrakan dengan user yang kosong serta password yang kosong. Untuk mencegah hal yang buruk segera-lah melakukan P3K / Pertolongan Pertama Pada Kejadian. Caranya adalah kita akan menghapus user tersebut dengan :

```
DELETE FROM user WHERE user = '';
```

Coba periksa kembali tabel user, seharusnya tidak ada lagi user dan password yang kosong.

Memberikan Password pada user Root

Biasanya jika kita menginstall MySQL, seperti halnya ketika menginstall Xampp for Linux yang dibahas di awal buku ini. User root tidak memiliki password, tetapi kan Xampp for Linux mempunyai tool dan command tersendiri untuk melakukan / memberikan password untuk user root. (coba dibaca lagi tentang instalasi LAMPP).

Tetapi masalahnya jika kita memberikan password tersebut langsung dari sistem MySQL. Oke akan kita pelajari bersama, sebenarnya cukup mudah untuk melakukan perubahan / memberikan password pada user root, cukup dengan :

```
UPDATE user SET password = PASSWORD('samid') WHERE user = 'root';
```

dan setelah sukses dengan update password-nya, kemudian dilanjutkan dengan :

```
FLUSH PRIVILEGES;
```

Fungsi FLUSH PRIVILEGES bertujuan untuk memerintahkan kepada sistem MySQL, agar membaca ulang grant table tanpa harus melakukan restart ulang database-nya.

Menambah dan menghapus user dengan GRANT dan REVOKE

GRANT mempunyai fungsi untuk membuat user baru beserta dengan hak akses si user tersebut.

Bentuk perintah umumnya adalah :

```
GRANT jenis_akses (kolom)
ON nama_database
TO nama_user
IDENTIFIED BY "Password"
WITH GRANT pilihan_akses
```

- Menambah user dengan nama “**emjhe**” dengan password “**cantik**” dapat mengakses semua database dan dapat login dari komputer mana saja dengan host **localhost**. Dan mendapatkan hak untuk memberikan GRANT kepada user lain.

```
GRANT ALL PRIVILEGES ON *.* TO emjhe@localhost
IDENTIFIED BY 'cantik' WITH GRANT OPTION;
```

- Menambahkan user dengan nama “**arka**” dengan password “**smart**” hanya dapat mengakses database “**test**” serta hanya dapat login dari IP **192.168.123.2**

```
GRANT ALL PRIVILEGES ON test.*  
TO arka@192.168.123.2  
IDENTIFIED BY 'smart';
```

- Menghapus user arka yang sudah dibuat.

```
REVOKE CREATE ON test.* FROM arka@192.168.123.2;
```

Serta jangan lupa setiap melakukan perubahan pada user selalu tambahkan :

```
FLUSH PRIVILEGES;
```

Sebenarnya masih banyak lagi tentang manajemen user pada MySQL, tapi sebagai pengetahuan dasar saja bagi sobat yang baru belajar MySQL. Tetap belajar dan mencari banyak referensi. Semangattttttt...

Chapter 10

Penutup

MySQL adalah sebuah database yang powerfull, sehingga menjadikan alasan yang kuat database ini memiliki banyak penggunanya di dunia. MySQL mempunyai kemampuan yang luar biasa dalam meng-handle banyak data dan sangat mumpuni juga ketika berkomunikasi dengan bahasa program lain.

Semua yang saya tulis pada buku ini semoga menjadi jalan mempermudah sobat semua untuk belajar mengenal dan mempelajari salah satu database yang sangat populer.

Saya sadar masih banyak kekurangan disana – sini dalam buku ini, oleh karena itu segala bentuk kritikan dan saran sangat berharga untuk saya. Jadi, mohon saran dan kritiknya agar nanti Insya Allah akan saya tulis tentang MySQL yang lebih komplit dan lebih detail lagi. Namun, tetap pada platform linux.

Dan saya mohon ini untuk menyebarluaskan ebook ini dengan tetap mencantumkan original writer-nya. Semoga apa yang saya berikan ini dapat mendapatkan imbalan dari Allah SWT.

Demikian, saya ucapkan terima kasih bagi semua yang telah membaca buku saya ini. Saya sangat senang dapat berkenalan dengan anda semua. Semoga perkenalan ini dapat langgeng dunia dan akhirat dan semakin dimudahkan rizkinya baik saya ataupun anda. Dan sampai jumpa pada buku-buku saya lainnya. SALAM LINUX!!!

Referensi

Achmad Solichin, MySQL 5 : Dari pemula hingga Mahir, Achmatim.net, 2010

Abdul Kadir, Mudah Mempelajari Database MySQL, Penerbit Andi, Yogyakarta, 2010

Robert Sheldon & Goeff Moes, Wrox-Beginning MySQL. Wiley Publishin, Inc.

MySQL. Situs Resmi MySQL. <http://mysql.com>.

Tentang Penulis

Dimas Edu Prasada (Dimas Edubuntu Samid) / Edu, lahir di Kota Tegal, 27 Januari 1988. Bujangan yang hobi dengan music koplo ini , saat ini aktif sebagai praktisi IT. Menggemari “mazhab” web based application dan PHP menjadi bahasa yang paling disukainya.

Pernah bekerja untuk situs koranlokal.com(2007 - 2008), dan bekerja untuk nirmalapost.com(2009) serta pernah juga bekerja di PT. Suzuki Gedong Jembar Kota Tegal (2008) menjadi head office IT Dept, dan kemudian meloncat ke dunia website lagi dengan menjadi founder dari panturanews.com(2009-2010). Kemudian menjadi developer untuk CV. Unggul Research(2010) dan berpindah ke Apotik Saras Sehat Slawi (2010 - 2011) menjadi developer untuk system Informasi Apotik.

Dan akhirnya di tahun 2011 pertengahan sampai sekarang (2012) menjadi bagian dari sebuah lembaga pendidikan PUSDIKOM Astagina. Dan sekarang menjadi Founder untuk software house Dimas Edu NET.

Sangat ingin sekali mewujudkan cita - cita yaitu memiliki sekolah gratis untuk anak-anak yang tidak mampu. Namun, saat ini masih belum bisa mewujudkannya semoga dikemudian hari dapat terwujud keinginan tersebut. Mohon doanya ya.... ☺

Penulis dapat dihubungi melalui :

HP : 085742100454 | 081391579795

Email : edudimas1@gmail.com

FB : <http://facebook.com/eduaying>

Twitter : @edu_aying

Blog / URL : mevy.wordpress.com / dimasedu.net

DIMASEDU NET - Software House

Office Jl. Sangir No. 18 Kota Tegal

