

MODUL MATA KULIAH SISTEM BASIS DATA



ABDUL ROZAQ

**PROGRAM STUDI
SISTEM INFORMASI
JURUSAN ADMINISTRASI BISNIS**

**KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
POLITEKNIK NEGERI BANJARMASIN
2025**

Daftar Isi

KATA PENGANTAR	ii
DAFTAR ISI	iii
DAFTAR TABEL	v
DAFTAR GAMBAR	v
BAB I MENGENAL MySQL	1
MATERI	1
1.1 Pendahuluan	1
1.2 APA ITU MySQL?	2
1.3 MySQL SEBAGAI DATABASE SERVER	2
1.4 MENGAPA MEMILIH MySQL	3
1.5 INSTALASI MySQL	4
SOAL LATIHAN	15
DAFTAR PUSTAKA	15
BAB II DASAR-DASAR MySQL	15
MATERI	16
2.1 Dasar-Dasar MySQL	16
2.2 Tipe Data pada MySQL	18
2.2 Melihat User dan Versi MySQL	22
2.3 Melihat Tanggal dan Waktu	22
2.4 MySQL Sebagai Kalkulator	22
2.5 Membuat Database Baru	23
2.6 Menghapus Database	23
2.7 Memilih dan Membuka Sebuah Database	24
2.8 Melihat Isi Sebuah Database	24
2.9 Membuat Tabel Baru	24
2.10 Melihat Struktur Tabel	25
2.11 Menghapus Tabel	26
2.12 Constraint	27
SOAL LATIHAN	28
DAFTAR PUSTAKA	28
BAB III Struktur Tabel Pada MySQL	29
MATERI	29
3.1 Mengubah Struktur Sebuah Tabel	29
3.2 Mengisi data ke dalam tabel	32
3.3 Melihat Data Pada Tabel	34
3.4 Meng-Update Data Pada Tabel	38
3.5 Pemasukan Data Secara Masal	39
SOAL LATIHAN	41

DAFTAR PUSTAKA	41
BAB IV Operator Pembanding dan Operator Logika	42
MATERI	42
4.1 Operator Pembanding dan Operator Logika	42
4.2 Fungsi Statistik Dasar	55
SOAL LATIHAN	57
DAFTAR PUSTAKA	57
BAB V Operator Precedence, LIKE, NOT LIKE, REGEXP	58
MATERI	58
5.1 Operator Precedence	58
5.2 Operator LIKE, NOT LIKE, REGEXP	59
A. Operator LIKE	59
B. Operator REGEXP	62
SOAL LATIHAN	66
DAFTAR PUSTAKA	66
BAB VI DATABASE RELASI	67
MATERI	67
6.1 Model Database	67
6.2 Model Database Relasi	68
6.3 Tingkatan Data Dalam Database Relasi	68
A. Karakter (Characters)	68
B. Field atau Attribute	68
C. Record atau Tupple	69
D. Table/Entity	69
E. Database	69
6.4 Sifat Yang Melekat Pada Suatu Tabel	69
6.5 Jenis Hubungan Antar Tabel	69
A. Satu Ke Satu (One to One)	70
B. Satu Ke Banyak (One to Many)	70
C. Banyak Ke Banyak (Many to Many)	71
6.6 Relasi Database dengan MySQL	71
SOAL LATIHAN	77
DAFTAR PUSTAKA	77
BAB VII MySQL dan Visual Basic 6.0	78
7.1 Pemrograman Database	78
7.2 Koneksi VB dengan Database MySQL	79
A. Langkah-Langkah Koneksi	79
B. Tambah, Ubah dan Hapus data	80
SOAL LATIHAN	83
DAFTAR PUSTAKA	83

Daftar Tabel

Tabel 4.1 Operator Pembanding	42
Tabel 4.2 Operator Logika.....	43
Tabel 4.3 Operator Precedence	58
Tabel 5.1 Simbol Operator REGEXP	62

Daftar Gambar

Gambar 1.2 MySQL Server 5.1-Setup Wizard (1)	6
Gambar 1.3 MySQL Server 5.1-Setup Wizard (2)	7
Gambar 1.4 MySQL Server 5.1-Setup Wizard (3)	8
Gambar 1.5 MySQL Enterprise (1).....	8
Gambar 1.6 MySQL Enterprise (2).....	9
Gambar 1.7 MySQL Server 5.1 Setup Wizard (4).....	9
Gambar 1.8 MySQL Server Instance Configuration Wizard (1).....	10
Gambar 1.9 MySQL Server Instance Configuration Wizard (2).....	10
Gambar 1.10 MySQL Server Instance Configuration Wizard (3).....	11
Gambar 1.11 MySQL Server Instance Configuration Wizard (4).....	12
Gambar 1.12 MySQL Server Instance Configuration Wizard (5).....	12
Gambar 1.13 MySQL Server Instance Configuration Wizard (6).....	13
Gambar 1.14 MySQL Command Line Client.....	13
Gambar 1.15 Menggunakan perintah SHOW DATABASES	14
Gambar 7.1. Alur Kerja Pemrograman Database dengan Visual Basic.....	79

BAB I

Mengenal MySQL

Standar Kompetensi :

1. Mahasiswa dapat mengetahui, memahami, menguasai dan mampu mengimplementasi teori, konsep dan prinsip pemrograman database MySQL dengan logika pemrograman yang benar, ringkas, dan tepat dalam penerapannya di bidang teknologi informasi

Kompetensi Dasar	Indikator
1.1 Mengetahui perkembangan database MySQL	<ul style="list-style-type: none">▪ Menjelaskan perkembangan database MySQL▪ Mengenal MySQL dan menjelaskan perkembangan versinya
1.2 Mengenal MySQL	<ul style="list-style-type: none">▪ Menjelaskan MySQL sebagai database server▪ Menjelaskan Keunggulan MySQL
1.3 Menjalankan MySQL	<ul style="list-style-type: none">▪ Download MySQL▪ Menginstal MySQL▪ Menjalankan MySQL

Materi

1.1 Pendahuluan

Seiring dengan waktu, banyak perkembangan yang terjadi pada dunia internet. Termasuk pesatnya perkembangan perangkat lunak Apache, MySQL dan PHP yang selalu di-update oleh produsennya masing-masing. Sebagai pertimbangan, versi terbaru dari Apache adalah **Apache 2.2.11** (per **14 Desember 2008**), versi terbaru dari PHP adalah **PHP 5.2.9** (per **26 Pebruari 2009**), dan versi terbaru dari MySQL adalah **MySQL 5.1.32** (per **Maret 2009**).

1.2 APA ITU MySQL?

MySQL (bisa dibaca dengan **mai-es-ki-el** atau bisa juga **mai-se-kuel**) adalah suatu perangkat lunak **database relasi** (**Relational Database Management System** atau **RDBMS**), seperti halnya ORACLE, Postgresql, MS SQL, dan sebagainya. Jangan disalah-artikan MySQL dengan SQL. **SQL** (singkatan dari **Structured Query Language**) sendiri adalah suatu sintaks perintah-perintah tertentu atau bahasa (pemrograman) yang digunakan untuk mengelola suatu database. Jadi, MySQL dan SQL adalah dua '*mahluk*' yang berbeda. Mudah-mudahan, MySQL adalah software-nya, dan SQL adalah bahasa perintahnya.

1.3 MySQL SEBAGAI DATABASE SERVER

Software database mulai bermunculan seiring dengan bertambahnya kebutuhan akan database server. Salah satu dari pendatang baru dalam dunia database ialah MySQL, sebuah server/klien database SQL yang berasal dari Skandinavia. MySQL terdiri atas server SQL, klien program untuk mengakses server, tools untuk administrasi, dan interface program untuk menulis program sendiri.

Pengembangan MySQL dimulai pada tahun 1979 dengan tool database UNIREG yang dibuat oleh Michael "Monty" Widenius untuk perusahaan TcX di Swedia. Kemudian pada tahun 1994, TcX mulai mencari server SQL untuk mengembangkan aplikasi Web. Mereka menguji beberapa server komersial namun semuanya masih terlalu lambat untuk table-tabel TcX yang besar.

Tahun 1995 David Axmark dari Detro HB berusaha menekan TcX untuk merelease MySQL di Internet. Ia juga membuat dokumentasi MySQL yang di-build untuk GNU configure utility. MySQL 3.11.1 dipublikasikan di dunia tahun 1996 dan didistribusikan untuk Linux dan Solaris. Sekarang ini MySQL bekerja untuk banyak platform serta tersedia source codenya.

1.4 MENGAPA MEMILIH MySQL

Jika anda mencari system manajemen database yang murah atau bahkan gratis, ada beberapa pilihan antara lain MySQL, mSQL, PostgresSQL, atau salah satu dari produk vendor komersial yang gratis. Ketika dibandingkan antara MySQL dengan system databae yang lain, maka perlu dipikirkan apa yang paling penting untuk anda. Apakah performa, support, fitur-fitur SQL, kondisi keamanan dalam lisensi, atau masalah harga. Dengan pertimbangan tersebut, MySQL memiliki banyak hal yang bisa ditawarkan, antara lain :

Kecepatan

Banyak ahli berpendapat MySQL merupakan server tercepat.

Kemudahan penggunaan

MySQL punya performa tinggi namun merupakan database yang simple sehingga mudah disetup dan dikonfigurasi

Harga

MySQL cenderung gratis untuk penggunaan tertentu.

Mendukung query language

MySQL mengerti bahasa SQL (Structured Query Language) yang merupakan pilihan system database modern. Anda juga dapat mengakses MySQL lewat protocol ODBC (Open Database Connectivity) buatan Microsoft.

Kapabilitas

Banyak klien dapat mengakses server dalam satu waktu. Mereka dapat menggunakan banyak database secara simultan.

Konektifitas dan sekuritas

Database MySQL dapat diakses dari semua tempat di Internet dengan hak akses tertentu.

Pertabilitas

MySQL dapat berjalan dalam banyak varian UNIX dengan baik, sebaik seperti saat berjalan di system non-UNIX.

Distribusi yang terbuka

MySQL mudah didapatkan dan memiliki source code yang boleh disebarluaskan sehingga bisa dikembangkan lebih lanjut.

Sedangkan pengguna database MySQL ini antara lain adalah :

- ✚ Silicon Graphics (<http://www.sgi.com>)
- ✚ Siemens (<http://www.siemens.com>)
- ✚ Terjemahan Al Quran dalam bahasa Indonesia (<http://netmon.itb.ac.id/~quran/>)
- ✚ ITB digital Library (<http://digital.lib.itb.ac.id>)
- ✚ Game Strategi Online Multiplayer Kurusetra (<http://www.kurusetra.com>)

Bagaimanapun, mungkin yang paling menarik dari semua karakteristik adalah kenyataan bahwa MySQL adalah gratis. Hal ini benar karena T.c.X menawarkan MySQL sebagai produk gratis untuk umum.

1.5 INSTALASI MySQL

Sebelum melanjutkan pembahasannya tentang MySQL, Anda harus memiliki sistem MySQL yang telah terinstall dengan baik di komputer Anda. MySQL dapat diunduh (di-*download*) langsung dari situsnyanya di www.mysql.com. Ada 2 jenis produk yang ditawarkan: **Community Edition** dan **Enterprise Edition**. Perbedaannya, Community Edition menggunakan bendera *Open Source* dengan konsekuensi gratis *diunduh*, gratis digunakan, tapi dengan fasilitas dukungan yang terbatas dari produsernya. Sebaliknya, Enterprise Edition, ada biaya lisensi yang harus dibayarkan kepada produsernya, dan Anda pun akan mendapatkan dukungan penuh dari produsernya. Untuk tutorial ini, kita akan menggunakan jenis Community Edition saja. Itu sudah mencukupi.

Pada tutorial ini, kita akan melakukan proses instalasi program **MySQL versi 5.1.34** basis Windows yang dirilis **per April 2009**. Anda disarankan untuk menggunakan versi MySQL yang terbaru dan telah dinyatakan stabil oleh produsernya. Ada kata *recommended* di samping versi MySQLnya. **MySQL 5.0.77** dirilis pada tanggal 14 Februari 2009, **MySQL 5.1.32** pada tanggal 13 Maret 2009 (*Community Edition*), **MySQL 5.1.34**

pada tanggal 02 April 2009. Nah sekarang silakan Anda download dulu (bisa langsung melalui link <http://dev.mysql.com>) dan kemudian diinstall ke dalam komputer Anda.

Bila Anda sudah masuk kedalam situs MySQL, carilah link untuk platform Windows (atau *Windows Download*). Ada 3 paket yang disediakan:

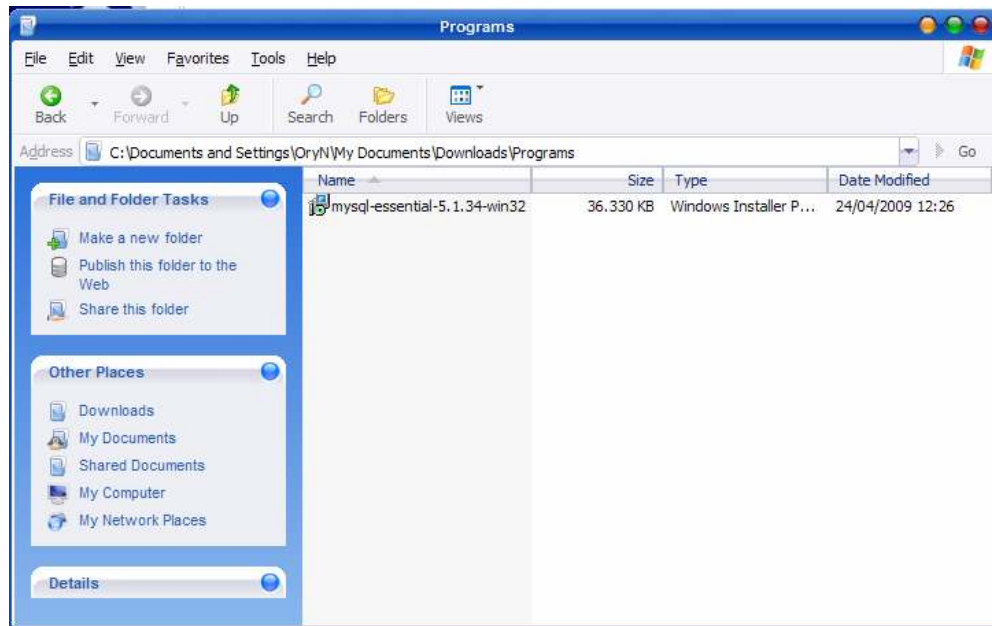
1. **Windows Essential (x86)** dengan ukuran file 22,7 MB
2. **Windows (x86)** dengan ukuran file 44,3 MB
3. **Windows, without installer**, dengan ukuran file 55,8 MB

Pilih yang mana dari 3 paket di atas? Kalo untuk sekedar coba-coba saja dan kapasitas harddisk Anda terbatas, paket *Windows Essential* sudah cukup memadai untuk latihan. Kalo ingin lebih serius lagi dan lebih lengkap lagi, maka pilih saja paket **Windows (x86)**. Paket kedua ini sudah dilengkapi dengan fasilitas instalasi yang cukup mudah diikuti. Lalu apa bedanya antara paket kedua dengan paket ketiga? Pada paket ketiga kelengkapan modulnya sama dengan paket kedua, hanya saja **tidak** dilengkapi dengan fasilitas instalasi. Kedua paket ini dalam format zip, yang harus di-uncompress atau di-unzip ke drive C:\).

Saya merekomendasikan untuk menggunakan paket yang kedua aja (**Windows (x86)**).... :). Bagi mereka yang telah didukung dengan komputer berprosesor inti ganda (*AMD 64 X2*) atau *Intel Core Duo*, bisa menggunakan paket yang **Windows x64**. Bila Anda telah selesai mengunduhnya (*download*), mari kita lanjutkan dengan proses instalasi MySQL.

1. Langkah 1

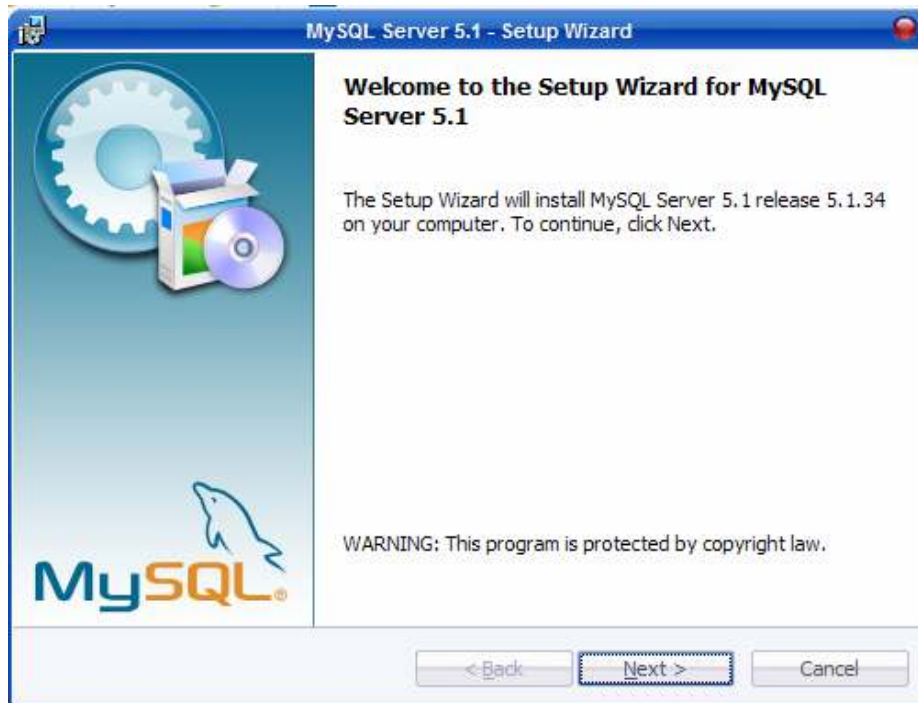
Pilih dan sorotlah file MySQL yang baru Anda download. Pada tutorial ini nama filenya adalah **mysql-essential-5.1.34-win32** dengan ukuran file 36.330 KB.



Gambar 1.1 File MySQL 5.1.34 yang telah didownload

2. Langkah 2

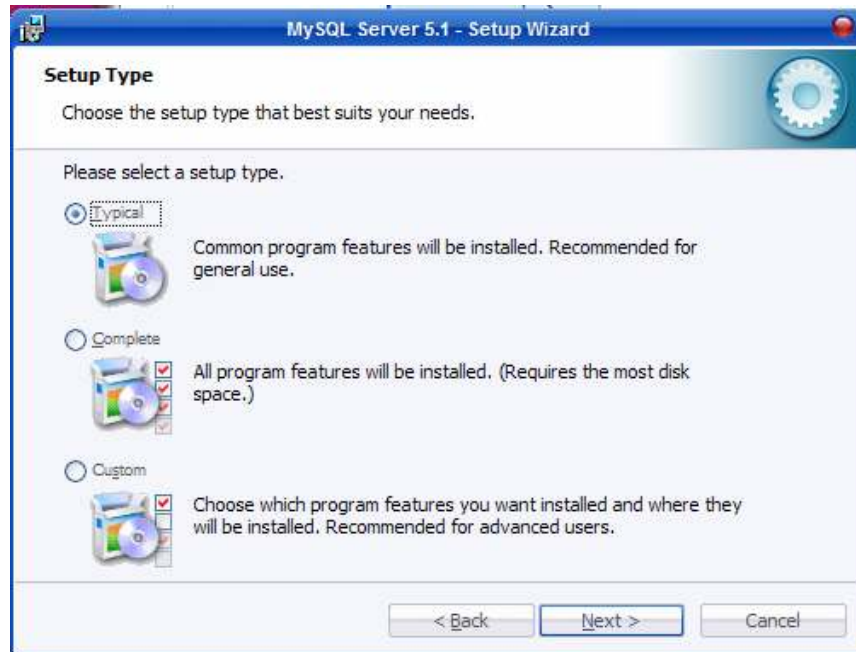
Double Click file tersebut, maka muncul tampilan selamat datang. Untuk melanjutkan proses instalasi klik pada tombol *Next >*.



Gambar 1.2 MySQL Server 5.1-Setup Wizard (1)

3. Langkah 3

Pada tampilan ini Anda harus memilih jenis instalasi yang akan dilaksanakan. Yang paling mudah adalah jenis instalasi **Typical**. Maka saya sarankan kita menggunakan jenis **Typical** saja... :) Untuk melanjutkan, silakan klik pada tombol *Next >*



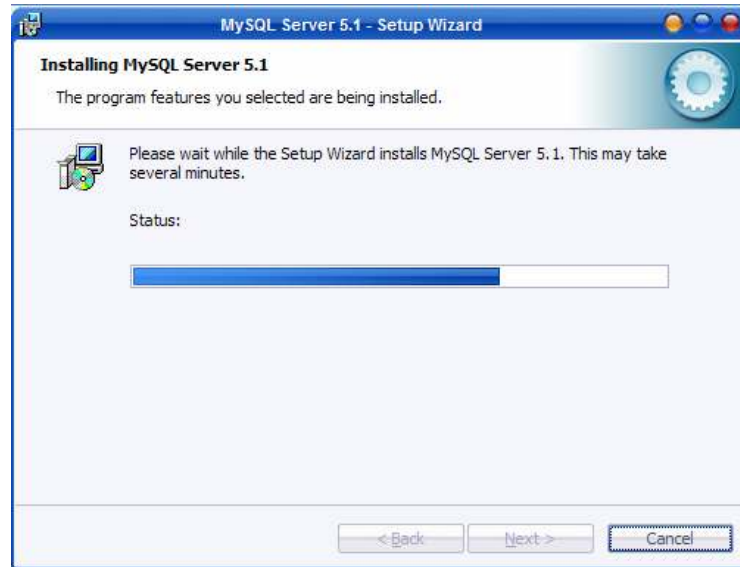
Gambar 1.3 MySQL Server 5.1-Setup Wizard (2)

4. Langkah 4

Tampilan berikutnya konfirmasi bahwa program MySQL akan diinstall ke dalam direktori **C:\Program Files\ MySQL\ MySQL Server 5.1**. Karena kita menggunakan jenis instalasi *Typical*, maka direktori instalasi tidak dapat kita ubah (kecuali Anda menggunakan jenis instalasi *Custom* pada langkah no.3 di atas). Untuk sementara kita gunakan saja apa adanya. Silakan klik tombol **Install** untuk melanjutkan.

5. Langkah 5

Anda bisa santai dulu menunggu proses persiapan instalasi berlangsung... :)



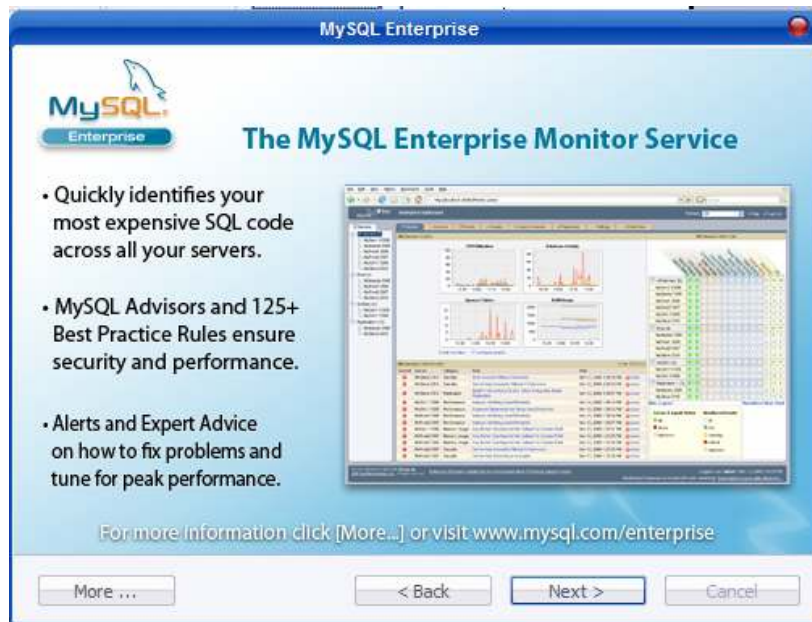
Gambar 1.4 MySQL Server 5.1-Setup Wizard (3)

6. Langkah 6

Tampilan berikutnya, informasi mengenai produk **MySQL Enterprise** (Anda harus membayar sejumlah uang untuk menggunakan produk berlisensi ini). Bila Anda berminat untuk menggunakan produk dengan lisensi ini, silakan kunjungi situsnya. Mari kita lanjutkan proses instalasi ini dengan menekan tombol *Next >* sebanyak dua kali.



Gambar 1.5 MySQL Enterprise (1)



Gambar 1.6 MySQL Enterprise (2)

7. Langkah 7

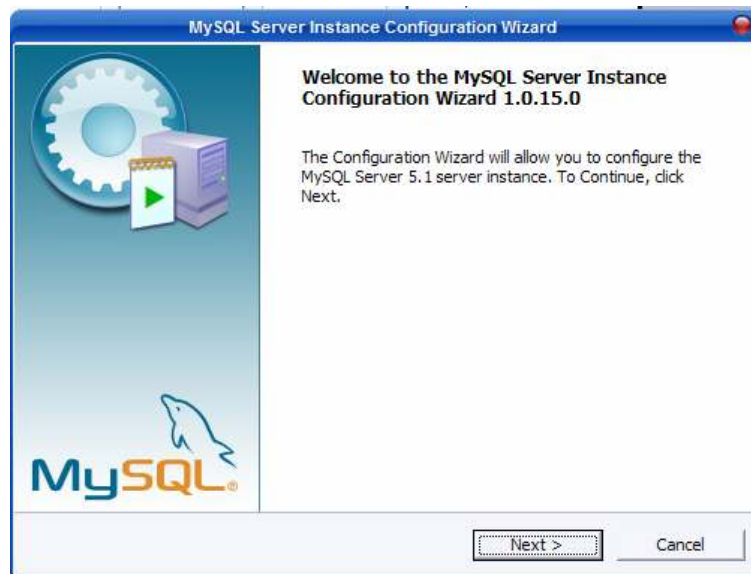
Tunggu beberapa saat hingga proses instalasi selesai. Kemudian (sangat disarankan) untuk melanjutkan ke proses konfigurasi MySQL server (*Configure the MySQL Server now*). Dan Anda bisa menekan tombol *Finish* untuk tahapan ini (yang akan dilanjutkan dengan proses konfigurasi).



Gambar 1.7 MySQL Server 5.1 Setup Wizard (4)

8. Langkah 8

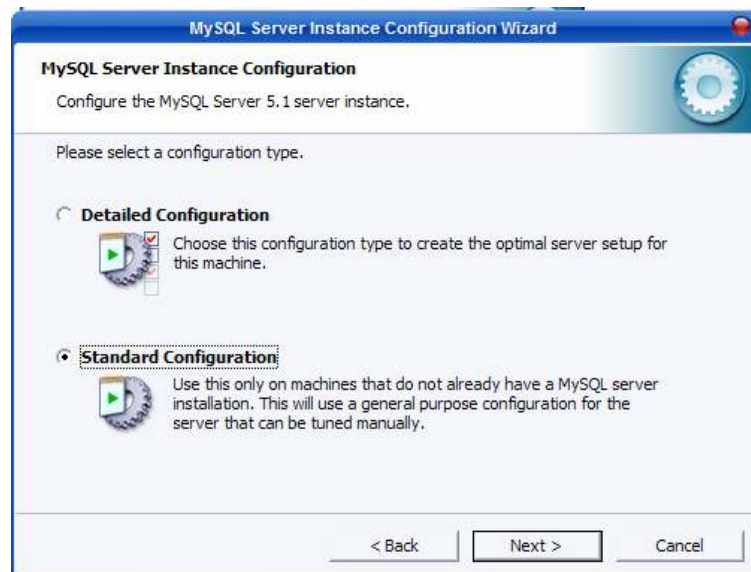
Sudah siap untuk mengkonfigurasi MySQL server? Silakan klik tombol *Next...*



Gambar 1.8 MySQL Server Instance Configuration Wizard (1)

9. Langkah 9

Ada dua pilihan konfigurasi, *Detailed Configuration* dan *Standard Configuration*. Kita pilih saja **Standard Configuration**, karena mudah untuk diselesaikan. Klik pada tombol *Next >* untuk melanjutkan.



Gambar 1.9 MySQL Server Instance Configuration Wizard (2)

10. Langkah 10

Tampilan berikutnya, disarankan untuk **mengaktifkan** pilihan **Install as Windows Service** dan juga **Launch the MySQL Server automatically** . Dengan pilihan ini maka setiap komputer Anda dinyalakan, secara otomatis program MySQL server akan dijalankan. Begitupun sebaiknya **aktifkan** pilihan **Include Bin directory in Windows Path**. Program-program MySQL biasanya disimpan di dalam directory **C:\Program Files\MySQL\MySQL Server 5.0\Bin**. Dengan mengaktifkan pilihan ini, maka Anda dapat menjalankan atau memanggil program MySQL langsung dari DOS/Command Prompt.



Gambar 1.10 MySQL Server Instance Configuration Wizard (3)

11. Langkah 11

Tampilan berikutnya, mengenai sistem keamanan server MySQL. Sebaiknya Anda memberikan password khusus sebagai Root, dan tidak memberikan peluang kepada orang lain untuk memasuki sistem anda tanpa password. Maka **aktifkan** pilihan *Modify Security Setting* dan **masukkan password** Root Anda dengan seksama. Tetapi, **matikan** pilihan **Create An Anonymous Account**. Dengan demikian tidak sembarangan orang dapat masuk menggunakan MySQL server Anda. Satu hal lagi, disarankan **mematikan**

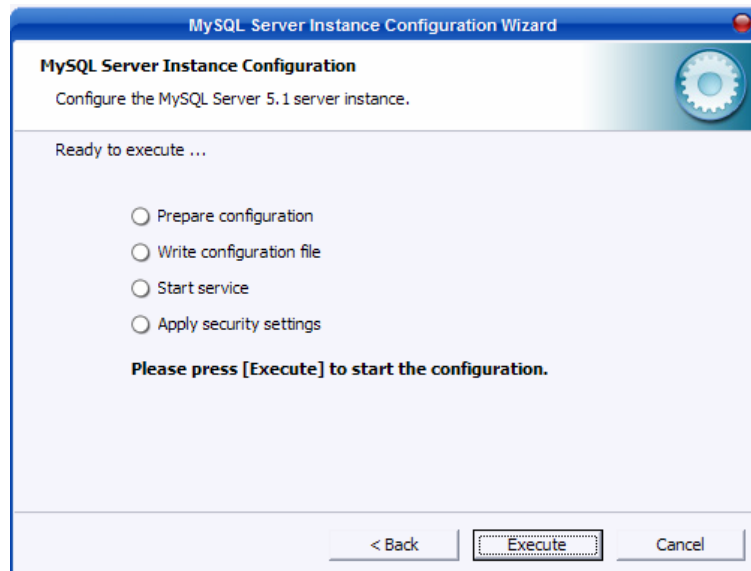
pilihan *Enable Root access from remote machines*. Ini untuk mencegah celah-celah yang bisa dimasuki oleh orang-orang yang tidak bertanggungjawab menyelinap ke dalam sistem kita. Lanjutkan dengan menekan tombol *Next*.



Gambar 1.11 MySQL Server Instance Configuration Wizard (4)

12. Langkah 12

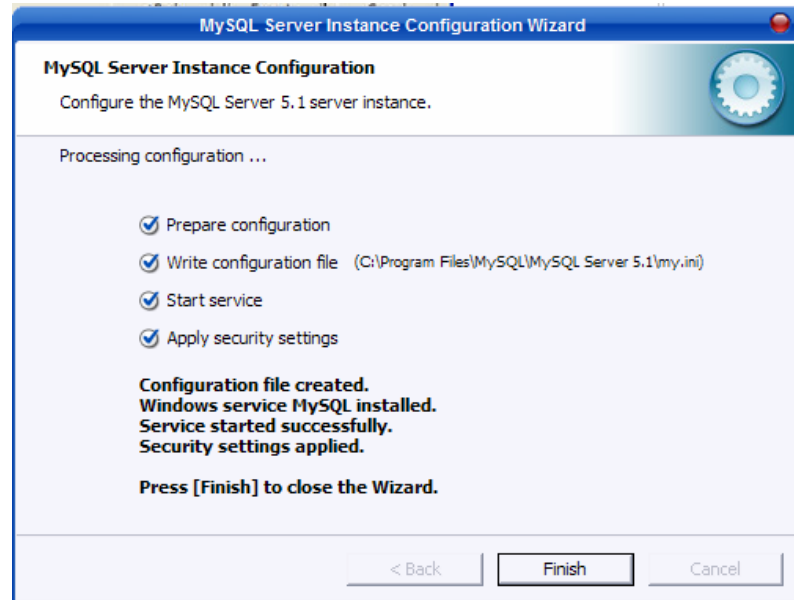
Bila Anda telah yakin untuk melanjutkan, klik pada tombol *Execute*. Dan Anda bisa santai sejenak sambil menunggu proses setting selesai.



Gambar 1.12 MySQL Server Instance Configuration Wizard (5)

13. Langkah 13

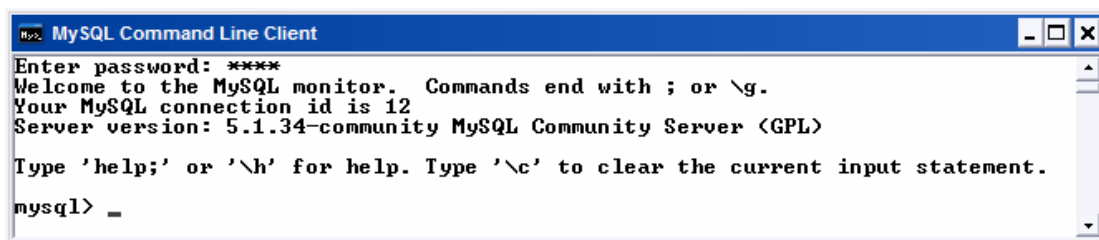
Bila tidak ada kendala apapun, maka selesailah keseluruhan proses instalasi dan setting pada program MySQL 5.1.x ini. Selamat yach... Silakan klik pada tombol *Finish* untuk menuntaskan proses ini.



Gambar 1.13 MySQL Server Instance Configuration Wizard (6)

14. Langkah 14

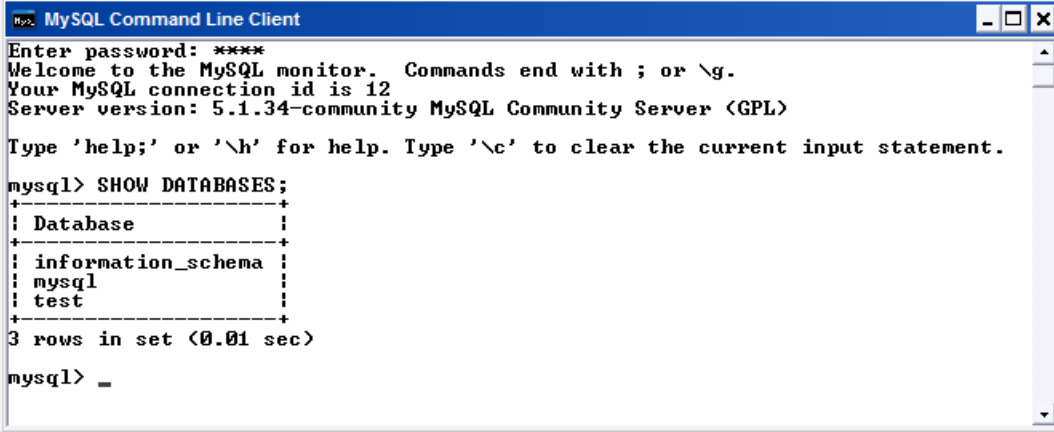
Sekarang silakan anda lakukan uji coba untuk mengakses MySQL dari DOS Prompt/Command Line. Jalankan program MySQL Server melalui menu utama Windows: **Start -> Programs -> MySQL -> MySQL Server 5.0 -> MySQL Command Line Client**. Kemudian ketikkan password yang telah Anda buat pada saat proses instalasi:



Gambar 1.14 MySQL Command Line Client

15. Langkah 15

Cobalah dengan perintah sederhana lainnya seperti **SHOW DATABASES;** untuk menampilkan semua database yang terdapat di MySQL.



```
MySQL Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.1.34-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql       |
| test        |
+-----+
3 rows in set (0.01 sec)

mysql> _
```

Gambar 1.15 Menggunakan perintah SHOW DATABASES

Catatan: Istilah database perlu dipahami dengan baik. **Database di dalam MySQL adalah sekumpulan tabel-tabel.** Jumlah tabel minimal satu buah, dan maksimalnya tidak terbatas. Semakin banyak tabel, maka akan semakin besar ukuran database Anda. Yang membatasi besarnya database adalah kemampuan sistem operasi kita, dan juga jumlah kapasitas ruang dalam harddisk dan memori komputer Anda. Keterangan selengkapnya mengenai hal ini dapat dilihat pada situs MySQL (<http://www.mysql.com>).

16. Langkah 16

Untuk keluar dari sistem MySQL, ketikkan perintah “\q;” atau klik pada tombol close



Soal Latihan

Buat artikel tentang MySQL meliputi sejarah perkembangannya mulai dari awal munculnya MySQL sampai versi terbarunya saat ini.

Daftar Pustaka

<http://dev.mysql.com>

[http://www.rohmat-mimi.com/download/MODUL PRAKTIKUM MY SQL-BASIS DATA](http://www.rohmat-mimi.com/download/MODUL_PRAKTIKUM_MY_SQL-BASIS_DATA)

<http://www.mysql.com>

<http://www.arbiedesign.com/index.php>

Tim Training SMK-TI.Modul MySQL

BAB II

Dasar-Dasar MySQL

Standar Kompetensi :

1. Mahasiswa dapat mengetahui, memahami, menguasai dan mampu mengimplementasi teori, konsep dan prinsip pemrograman database MySQL dengan logika pemrograman yang benar, ringkas, dan tepat dalam penerapannya di bidang teknologi informasi

Kompetensi Dasar	Indikator
1.4 Mengetahui Dasar-Dasar MySQL	<ul style="list-style-type: none">▪ Mengetahui tipe data dalam MySQL▪ Mengenal perintah dasar MySQL dan mengetahui fungsinya▪ Menggunakan MySQL untuk mengetahui tanggal dan waktu▪ Menggunakan MySQL sebagai kalkulator
1.5 Membuat Database	<ul style="list-style-type: none">▪ Mengetahui cara membuat database, menggunakan database dan menghapus database
1.6 Membuat Tabel	<ul style="list-style-type: none">▪ Mengetahui cara membuat tabel, menentukan field beserta tipe datanya▪ Mengetahui cara menghapus tabel
1.7 Mengenal Constraint	<ul style="list-style-type: none">▪ Mengetahui beberapa tipe constraint▪ Mengetahui fungsi masing-masing tipe constraint

Materi

2.1 Dasar-Dasar MySQL

Dalam bahasa SQL pada umumnya informasi tersimpan dalam tabel-tabel yang secara logik merupakan struktur dua dimensi terdiri dari baris (*row* atau *record*) dan kolom (*column* atau *field*). Sedangkan dalam sebuah *database* dapat terdiri dari beberapa *table*.

Prompt **mysql>** menunjukkan bahwa database mysql telah aktif. Jika prompt ini telah aktif dapat langsung mengetikkan perintah-perintah dilingkungan MySQL.

Perintah-perintah MySQL antara lain :

Tabel 2.2 Perintah-perintah MySQL

Perintah	Perintah Singkat	Kegunaan
Help	\h	Menampilkan daftar perintah
Clear	\c	Menghapus (clear)
Connect	\r	Menghubungkan kembali database MySQL
Exit	\q	Keluar dari MySQL
Go	\g	Mengirimkan perintah kepada MySQL
Ego	\G	Mengirimkan perintah kepada MySQL dan menampilkan hasilnya secara vertical
Print	\p	Mencetak perintah saat ini
Use	\u	Membuat/mengganti koneksi kepada database

Ketentuan Memberikan Perintah

- ✚ Perintah dalam MySQL mengenal case insensitive, perintah dapat ditulis dengan huruf besar (uppercase), ataupun dengan huruf kecil (lowercase).
- ✚ Setiap perintah diakhiri dengan ; (tanda titik koma) atau dengan memberikan perintah \g diakhir perintah
- ✚ Perintah dapat berupa perintah SQL atau perintah khusus MySQL
- ✚ Jika Prompt mysql> berganti dengan -> berarti prompt tersebut menunggu kelengkapan perintah dari baris sebelumnya atau menunggu diberikan tanda ; atau \g.
Contoh : Perhatikan perintah dibawah ini ditulis **tanpa tanda titik-koma ";"**.

```
mysql> create database latihan1  
->
```

Sistem MySQL akan menampilkan tanda panah '->' yang menyatakan bahwa perintah MySQL tersebut dianggap belum selesai (karena belum diakhiri dengan tanda titik-koma ';').

Sekarang kita lengkapi perintah sebelumnya dengan tanda titik-koma ';'.

```
mysql> create database latihan1  
-> ;  
Query OK, 1 row affected (0.02 sec)
```

2.2 Tipe Data pada MySQL

Pemilihan tipe data merupakan suatu hal yang cukup penting dalam mengelola server. Salah satu sebabnya adalah berkaitan dengan ruang di harddisk dan memori yang akan “digunakan” oleh data-data tersebut.

Berikut ini akan diberikan tipe-tipe data yang didukung oleh MySQL yang terambil dari dokumentasi MySQL. Tipe - tipe data ini diberikan dalam bentuk yang siap dituliskan pada sintaks-sintaks MySQL, misalnya *Create Table*. Pada tipe-tipe data tersebut terdapat beberapa atribut yang memiliki arti sebagai berikut:

- ✚ M, menunjukkan lebar karakter maksimum. Nilai M maksimum adalah 255.
- ✚ D, menunjukkan jumlah angka di belakang koma. Nilai maksimum D adalah 30 tetapi dibatasi oleh nilai M, yaitu tidak boleh lebih besar daripada M-2.
- ✚ Atribut yang diberi tanda [dan] berarti pemakaiannya adalah optional.
- ✚ Jika atribut ZEROFILL disertakan, MySQL akan otomatis menambahkan atribut UNSIGNED.
- ✚ UNSIGNED adalah bilangan tanpa tanda di depannya (misalnya tanda negatif).

Inilah tipe-tipe data tersebut:

1. **TINYINT[(M)] [UNSIGNED] [ZEROFILL]**

Integer yang sangat kecil jangkauan nilainya, yaitu -128 hingga 127. Jangkauan unsigned adalah 0 hingga 255.

2. **SMALLINT[(M)] [UNSIGNED] [ZEROFILL]**

Integer yang kecil jangkauan nilainya, yaitu -32768 hingga 32767. Jangkauan unsigned adalah 0 hingga 65535.

3. **MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]**

Integer tingkat menengah. Jangkauan nilainya adalah -8388608 hingga 8388607. Jangkauan unsigned adalah 0 hingga 16777215.

4. **INT[(M)] [UNSIGNED] [ZEROFILL]**

Integer yang berukuran normal. Jangkauan nilainya adalah -2147483648 hingga 2147483647. Jangkauan unsigned adalah 0 hingga 4294967295.

5. **INTEGER[(M)] [UNSIGNED] [ZEROFILL]**

Sama dengan INT.

6. **BIGINT[(M)] [UNSIGNED] [ZEROFILL]**

Integer berukuran besar. Jangkauan nilainya adalah -9223372036854775808 hingga 9223372036854775807. Jangkauan unsigned adalah 0 hingga 18446744073709551615.

7. **FLOAT(precision) [ZEROFILL]**

Bilangan floating-point. Tidak dapat bersifat unsigned. Nilai atribut precision adalah ≤ 24 untuk bilangan floating-point presisi tunggal dan di antara 25 dan 53 untuk bilangan floating-point presisi ganda.

8. **FLOAT[(M,D)] [ZEROFILL]**

Bilangan floating-point presisi tunggal. Tidak dapat bersifat unsigned. Nilai yang diijinkan adalah $-3.402823466E+38$ hingga $-1.175494351E-38$ untuk nilai negatif, 0, and $1.175494351E-38$ hingga $3.402823466E+38$ untuk nilai positif.

9. **DOUBLE[(M,D)] [ZEROFILL]**

Bilangan floating-point presisi ganda. Tidak dapat bersifat unsigned. Nilai yang diijinkan adalah $-1.7976931348623157E+308$ hingga $-2.2250738585072014E-308$ untuk nilai negatif, 0, dan $2.2250738585072014E-308$ hingga $1.7976931348623157E+308$ untuk nilai positif.

10. **DOUBLE PRECISION[(M,D)] [ZEROFILL] dan REAL[(M,D)] [ZEROFILL]**

Keduanya sama dengan DOUBLE.

11. **DECIMAL[(M[,D])] [ZEROFILL]**

Bilangan floating-point yang “unpacked”. Tidak dapat bersifat unsigned. Memiliki sifat mirip dengan CHAR. Kata “unpacked” berarti bilangan disimpan sebagai string, menggunakan satu karakter untuk setiap digitnya. Jangkauan nilai dari DECIMAL sama dengan DOUBLE, tetapi juga tergantung dai nilai atribut M dan D yang disertakan. Jika D tidak diisi akan dianggap 0. Jika M tidak diisi maka akan

dianggap 10. Sejak MySQL 3.22 nilai M harus termasuk ruang yang ditempati oleh angka di belakang koma dan tanda + atau -.

12. NUMERIC(M,D) [ZEROFILL]

Sama dengan DECIMAL.

13. DATE

Sebuah tanggal. MySQL menampilkan tanggal dalam format 'YYYY-MM-DD'. Jangkauan nilainya adalah '1000-01-01' hingga '9999-12-31'.

14. DATETIME

Sebuah kombinasi dari waktu (jam) dan tanggal. MySQL menampilkan waktu dan tanggal dalam format 'YYYY-MM-DD HH:MM:SS'. Jangkauan nilainya adalah '1000-01-01 00:00:00' hingga '9999-12-31 23:59:59'.

15. TIMESTAMP[(M)]

Sebuah timestamp. Jangkauannya adalah dari '1970-01-01 00:00:00' hingga suatu waktu di tahun 2037. MySQL menampilkan tipe data TIMESTAMP dalam format YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, atau YYMMDD, tergantung dari nilai M, apakah 14 (atau tidak ditulis), 12, 8, atau 6.

16. TIME

Tipe data waktu. Jangkauannya adalah '-838:59:59' hingga '838:59:59'. MySQL menampilkan TIME dalam format 'HH:MM:SS'.

17. YEAR[(2|4)]

Angka tahun, dalam format 2- atau 4-digit (default adalah 4-digit). Nilai yang mungkin adalah 1901 hingga 2155, 0000 pada format 4-digit, dan 1970-2069 pada format 2-digit (70-69).

18. CHAR(M) [BINARY]

String yang memiliki lebar tetap. Nilai M adalah dari 1 hingga 255 karakter. Jika ada sisa, maka sisa tersebut diisi dengan spasi (misalnya nilai M adalah 10, tapi data yang disimpan hanya memiliki 7 karakter, maka 3 karakter sisanya diisi dengan spasi). Spasi ini akan dihilangkan apabila data dipanggil. Nilai dari CHAR akan

disortir dan diperbandingkan secara case-insensitive menurut default character set yang tersedia, kecuali bila atribut BINARY disertakan.

19. VARCHAR(M) [BINARY]

String dengan lebar bervariasi. Nilai M adalah dari 1 hingga 255 karakter. Jika nilai M adalah 10 sedangkan data yang disimpan hanya terdiri dari 5 karakter, maka lebar data tersebut hanya 5 karakter saja, tidak ada tambahan spasi.

20. TINYBLOB dan TINYTEXT

Sebuah BLOB (semacam catatan) atau TEXT dengan lebar maksimum $2^8 - 1$ karakter.

21. BLOB dan TEXT

Sebuah BLOB atau TEXT dengan lebar maksimum $2^{16} - 1$ karakter.

22. MEDIUMBLOB dan MEDIUMTEXT

Sebuah BLOB atau TEXT dengan lebar maksimum $2^{24} - 1$ karakter.

23. LONGBLOB dan LONGTEXT

Sebuah BLOB atau TEXT dengan lebar maksimum $2^{32} - 1$ karakter.

24. ENUM('value1','value2',...)

Sebuah enumerasi, yaitu objek string yang hanya dapat memiliki sebuah nilai, dipilih dari daftar nilai 'value1', 'value2', ..., NULL atau nilai special "" error. Sebuah ENUM maksimum dapat memiliki 65535 jenis nilai.

25. SET('value1','value2',...)

Sebuah set, yaitu objek string yang dapat memiliki 0 nilai atau lebih, yang harus dipilih dari daftar nilai 'value1', 'value2', Sebuah SET maksimum dapat memiliki 64 anggota.

2.2 Melihat User dan Versi MySQL

Untuk melihat user dan versi MySQL Anda, cukup dengan menggunakan rumus :

Select User (),Version ();

```
mysql> select user(),version();
+-----+
| user()          | version()          |
+-----+
| root@localhost | 5.1.34-community |
+-----+
1 row in set (0.05 sec)
```

2.3 Melihat Tanggal dan Waktu

Untuk melihat tanggal didalam MySQL anda dapat melakukan dengan rumus **curdate();**

```
Mysql>select curdate();
```

Sedangkan untuk melihat waktu dengan rumus **curtime();**

```
Mysql>select curtime();
```

Untuk melihat waktu dan sekaligus tanggal, maka rumus yang dituliskan adalah **now();**

```
Mysql>select now();
```

2.4 MySQL Sebagai Kalkulator

Dengan MySQL, kita tidak usah bingung ketika suatu saat kita harus menggunakan alat bantu kalkulator., karena hal ini dapat ditangani langsung oleh MySQL tanpa harus membuat program terlebih dahulu. Rumus yang dituliskan adalah **select rumus_perhitungan;**

```
mysql> select 5+5;
+-----+
| 5+5 |
+-----+
| 10 |
+-----+
1 row in set (0.03 sec)
```

Operator penghitungan meliputi : penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/). Silakan Anda coba melakukan penghitungan bilangan dengan menggunakan operator tersebut.

2.5 Membuat Database Baru

Sudah ada 3 buah database di dalam sistem MySQL. Sekarang kita akan membuat sebuah database untuk latihan kita. Gunakan perintah "**CREATE DATABASE**" untuk membuat sebuah database.

```
mysql> create database latihan2 ;  
Query OK, 1 row affected (0.02 sec)
```

Kita periksa hasil dari perintah di atas dengan "**SHOW DATABASE**".

```
mysql> show databases ;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| latihan1 |  
| latihan2 |  
| mysql |  
| test |  
+-----+  
5 rows in set (0.00 sec)
```

2.6 Menghapus Database

Jika kita tidak memerlukan database *latihan2*, maka kita dapat menghapusnya dengan perintah **DROP DATABASE**.

*Hati-hati dalam menggunakan perintah **DROP DATABASE** ini, karena database beserta seluruh isinya akan lenyap dari muka bumi tanpa bisa kita kembalikan lagi! Parahnya lagi, sistem MySQL tidak memberikan pertanyaan konfirmasi kepada Anda sebelum melakukan proses penghapusan database ini!*

```
mysql> drop database latihan2 ;  
Query OK, 0 row affected (0.02 sec)
```

Anda bisa memeriksanya lagi hasil dari perintah di atas dengan **"SHOW DATABASE"**.

```
mysql> show databases ;
+-----+
| Database                |
+-----+
| information_schema      |
| latihan1                 |
| mysql                   |
| test                    |
+-----+
4 rows in set (0.00 sec)
```

Anda perhatikan, database *latihan2* sudah menghilang. Sekali lagi, hati-hati dalam menggunakan perintah DROP DATABASE !

2.7 Memilih dan Membuka Sebuah Database

Sekarang kita pilih database **"latihan1"** dan kita buka dengan perintah **"USE"**

```
mysql> use latihan1 ;
Database change
```

2.8 Melihat Isi Sebuah Database

Untuk melihat apa isi dari sebuah database, kita gunakan perintah **"SHOW TABLES"**. Mari kita coba.

```
mysql> show tables ;
Empty set (0.00 sec)
```

Hasil dari perintah SHOW TABLES diatas adalah **"Empty Set"**, yang berarti belum ada tabel apapun di dalam database *latihan1*.

2.9 Membuat Tabel Baru

Kita akan membuat sebuah tabel baru dengan menggunakan perintah **"CREATE TABLE"**.

Contohnya sebagai berikut..

```
mysql> create table karyawan ;
ERROR 1113 (42000): A table must have at least 1 column
```

Ternyata ada kesalahan yang terjadi. Untuk membuat sebuah tabel di MySQL, kita harus menentukan minimal satu buah field/kolom di dalamnya. Sekarang kita ubah perintah di atas menjadi sebagai berikut:

```
mysql> create table karyawan
-> (nopeg INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
-> nama VARCHAR(50) NOT NULL)
-> ;
Query OK, 0 rows affected (0.14 sec)
```

Cukup panjang ya perubahan perintahnya. Mungkin sintaks perintahnya agak membingungkan pada awalnya. Tidak apa-apa, nanti akan kita bahas artinya. Secara umum, kita akan membuat sebuah **tabel Karyawan** dengan 2 buah kolom/field.

🚦 Kolom pertama adalah **NOPEG** dengan jenis data bilangan bulat (**INTeger**), tanpa tanda negatif (**UNSIGNED**), yang akan bertambah nilainya secara otomatis (**AUTO_INCREMENT**), kolom NOPEG adalah kolom utama (**PRIMARY KEY**).

🚦 Pada kolom kedua, **NAMA** akan menampung nama karyawan, dengan jenis data **VARIabel CHARacter**, lebar datanya dapat menampung maksimal 50 karakter, dan tidak boleh dikosongkan (**NOT NULL**). Kurang lebih seperti itulah ceritanya.. :)

Kita lihat kembali apa isi dari database *latihan1*:

```
mysql> show tables ;
+-----+
| Tables_in_latihan1 |
+-----+
| karyawan           |
+-----+
1 row in set (0.00 sec)
```

Dari hasil perintah di atas, kita lihat bahwa database *latihan1* telah memiliki sebuah tabel yang bernama *karyawan*. Selanjutnya kita akan lihat apa struktur dari tabel *karyawan* tersebut.

2.10 Melihat Struktur Tabel

Untuk melihat struktur sebuah tabel dapat menggunakan perintah "**DESCRIBE**" atau bisa juga menggunakan perintah "**SHOW COLUMNS FROM**". Contohnya berikut ini:

```
mysql> describe karyawan ;
+-----+-----+-----+-----+-----+-----+
|Field |Type           | Null |Key |Default|Extra           |
+-----+-----+-----+-----+-----+-----+
|nopeg  |int(10) unsigned| NO   |PRI |NULL   |auto_increment |
|nama   |varchar(50)     | NO   |    |       |               |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)
```

Atau menggunakan perintah "SHOW COLUMNS FROM..."

```
mysql> show columns from karyawan ;
+-----+-----+-----+-----+-----+
|Field|Type          |Null|Key|Default|Extra          |
+-----+-----+-----+-----+-----+
|nopeg|int(10) unsigned|NO  |PRI|NULL   |auto_increment|
|nama |varchar(50)    |NO  |    |        |              |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Tidak ada perbedaan hasil dari dua perintah di atas, bukan? Sekarang kita buat sebuah tabel baru lagi, kita namakan saja tabel *contoh1*.

```
mysql> create table contoh1
-> (noid INT)
-> ;
Query OK, 0 rows affected (0.13 sec)
```

Sekarang kita lihat berapa tabel yang ada di dalam database *latihan1*:

```
mysql> show tables ;
+-----+
| Tables_in_latihan1 |
+-----+
| contoh1            |
| karyawan           |
+-----+
2 rows in set (0.00 sec)
```

2.11 Menghapus Tabel

Tabel *contoh1* yang baru saja kita buat ini akan kita hapus kembali. Perintah untuk menghapus sebuah tabel dalam MySQL adalah "**DROP TABLE**". Cukup mirip dengan perintah menghapus database, bukan? Kita harus menggunakan perintah "DROP" ini dengan **kehati-hatian yang tinggi**. Sistem MySQL tidak akan memberikan peringatan awal atau konfirmasi untuk proses penghapusan tabel. Dan bila sudah dihapus, maka tabel tersebut tidak bisa lagi kita kembalikan. **Maka, berhati-hatilah!**

```
mysql> drop table contoh1 ;
Query OK, 0 rows affected (0.03 sec)
```

Kita lihat lagi tabel yang ada di dalam database *latihan1*:

```
mysql> show tables ;
+-----+
| Tables_in_latihan1 |
+-----+
| karyawan            |
+-----+
1 rows in set (0.00 sec)
```

2.12 Constraint

Constraint adalah batasan atau aturan yang ada pada tabel.

MySQL menyediakan beberapa tipe constraint berikut :

🚦 NOT NULL

Suatu kolom yang didefinisikan dengan constraint NOT NULL tidak boleh berisi nilai NULL. Kolom yang berfungsi sebagai kunci primer (primary key) otomatis tidak boleh NULL.

🚦 UNIQUE

Mendefinisikan suatu kolom menjadi bersifat unik, artinya antara satu data dengan data lainnya namanya tidak boleh sama, misal alamat email.

🚦 PRIMARY KEY

Constraint PRIMARY KEY membentuk key yang unik untuk suatu tabel.

🚦 FOREIGN KEY

FOREIGN KEY constraint didefinisikan pada suatu kolom yang ada pada suatu table, dimana kolom tersebut juga dimiliki oleh table yang lain sebagai suatu PRIMARY KEY, biasa dipakai untuk menghubungkan antara 2 tabel.

Soal Latihan

Buat Database dengan nama dbKursus. Pilih dan buka database tersebut. Buat tabel dengan nama peserta untuk menyimpan data peserta meliputi : nomor, nama, email, alamat, kota.

Sedangkan strukturnya seperti tabel dibawah ini :

Kolom (<i>Field</i>)	Tipe Data (<i>Data Type</i>)
nomor	Char (8) Not Null Primary Key
nama	VarChar (20) Not Null
email	VarChar (30) Null
alamat	VarChar (20) Not Null
kota	VarChar (10) Not Null

Daftar Pustaka

<http://dev.mysql.com>

[http://www.rohmat-mimi.com/download/MODUL PRAKTIKUM MY SQL-BASIS DATA](http://www.rohmat-mimi.com/download/MODUL_PRAKTIKUM_MY_SQL-BASIS_DATA)

<http://www.mysql.com>

<http://www.arbiedesign.com/index.php>

Tim Training SMK-TI.Modul MySQL

BAB III

Struktur Tabel pada MySQL

Standar Kompetensi :

1. Mahasiswa dapat mengetahui, memahami, menguasai dan mampu mengimplementasi teori, konsep dan prinsip pemrograman database MySQL dengan logika pemrograman yang benar, ringkas, dan tepat dalam penerapannya di bidang teknologi informasi

Kompetensi Dasar	Indikator
1.8 Mengubah struktur tabel	<ul style="list-style-type: none">▪ Menambahkan field pada tabel▪ Mengubah lebar dan jenis field▪ Menghapus kolom dan indeks▪ Penggantian nama kolom▪ Penggantian nama tabel
1.9 Memasukkan data pada tabel	<ul style="list-style-type: none">▪ Dapat memasukkan data pada tabel secara satu per satu atau secara masal

Materi

3.1 Mengubah Struktur Sebuah Tabel

Ada saatnya kita perlu mengubah struktur tabel yang pernah kita buat sebelumnya. Pengubahan struktur bisa dalam hal penambahan kolom (**ADD**), pengubahan lebar dan jenis kolom (**MODIFY**), atau bisa saja penghapusan kolom dan indeks (**DROP**), penggantian nama kolom (**CHANGE**), penggantian nama tabel (**RENAME**), dan sebagainya. Apa pun juga yang anda lakukan pada kolom tersebut tentu akan mempunyai dampak langsung pada data-data yang sudah ada. Nah, sekarang kita perlu menambahkan beberapa kolom baru, yaitu kolom jenis kelamin, kota, tanggal lahir dan kodepos pada tabel *karyawan*.

Perintah untuk mengubah struktur tabel adalah "**ALTER TABLE**". Mari kita coba...

```
mysql> ALTER TABLE karyawan
-> ADD jenkelamin CHAR(2) NOT NULL,
-> ADD kota VARCHAR(25) NOT NULL,
-> ADD kodepos CHAR(5) NOT NULL,
-> ADD tgllahir DATE
-> ;
Query OK, 0 rows affected (0.20 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Sekarang kita lihat hasilnya:

```
mysql> describe karyawan ;
+-----+-----+-----+-----+-----+-----+
|Field      |Type      |Null|Key |Default|Extra      |
+-----+-----+-----+-----+-----+-----+
|nopeg      |int(10)   |NO  |PRI |NULL   |auto_increment|
|nama       |varchar(50)|NO  |    |       |              |
|jenkelamin |char(2)   |YES |    |NULL   |              |
|kota       |varchar(25)|NO  |    |       |              |
|kodepos    |char(5)   |NO  |    |       |              |
|tgllahir   |date      |YES |    |NULL   |              |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Jenis kelamin hanya membutuhkan lebar data 1, oleh karena itu kita bisa mengubah lebar data pada jenis kelamin tersebut dengan perintah MODIFY,

```
mysql> ALTER TABLE karyawan MODIFY jenkelamin Char(1);
Query OK, 0 rows affected (0.25 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Sekarang kita lihat hasilnya:

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| nopeg      | int(10) unsigned | NO  | PRI | NULL   | auto_increment |
| nama       | varchar(50)      | NO  |    | NULL   |              |
| jenkelamin | char(1)          | YES |    | NULL   |              |
| kota       | varchar(25)      | NO  |    | NULL   |              |
| kodepos    | char(5)          | NO  |    | NULL   |              |
| tgllahir   | date            | YES |    | NULL   |              |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.30 sec)
```

Mungkin ada baiknya kalau nama kolom *nopeg* kita ubah aja menjadi *noid*. Begitupun dengan nama kolom *jenkelamin*, kita ubah namanya menjadi *jenkel* saja. Dalam pengubahan kolom ini sebaiknya 'sifat-sifat' kolom yang asli tetap ditulis ulang. Misal bila

kolom *nopeg* memiliki sifat 'auto_increment', maka selama sifat itu tetap dipertahankan, maka dia (*auto_increment*) harus ditulis ulang. Begini caranya...
Mengubah kolom *nopeg* menjadi *noid*, tanpa mengubah jenis datanya (tetap INT(10), dan tetap auto_increment):

```
mysql> alter table karyawan
      -> change nopeg noid int(10) auto_increment
      -> ;
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Mengubah kolom *jenkelamin* menjadi *jenkel*,

```
mysql> alter table karyawan
      -> change jenkelamin jenkel char(1) ;
Query OK, 0 rows affected (0.24 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Sekarang kita lihat struktur tabel setelah perubahan:

```
mysql> describe karyawan ;
+-----+-----+-----+-----+-----+-----+
|Field  |Type      |Null|Key  |Default|Extra      |
+-----+-----+-----+-----+-----+-----+
|noid   |int(10)   |NO  |PRI  |NULL   |auto_increment|
|nama   |varchar(50)|NO  |     |       |              |
|jenkel |char(1)   |YES  |     |NULL   |              |
|kota   |varchar(25)|NO  |     |       |              |
|kodepos|char(5)   |NO  |     |       |              |
|tgllahir|date      |YES  |     |NULL   |              |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Bagaimana, sudah sesuai dengan perubahan struktur yang kita inginkan, bukan? Nah, sekarang bagaimana kalau kita ingin mengubah nama tabel *karyawan* menjadi tabel *pegawai*? Silakan dicoba dibawah ini:

```
mysql> alter table karyawan
      -> rename pegawai ;
Query OK, 0 rows affected (0.09 sec)
```

Kita lihat lagi hasilnya:

```
mysql> show tables ;
+-----+
| Tables_in_latihan1 |
+-----+
| pegawai            |
+-----+
1 row in set (0.00 sec)
```

Sekarang kita kembalikan lagi nama tabel *pegawai* menjadi *karyawan*. Tetapi dengan perintah yang berbeda, yaitu **"RENAME TABLE"**.

```
mysql> rename table pegawai
-> to karyawan
-> ;
Query OK, 0 rows affected (0.06 sec)
```

Jangan lupa untuk memeriksa hasilnya:

```
mysql> show tables ;
+-----+
| Tables_in_latihan1 |
+-----+
| karyawan            |
+-----+
1 row in set (0.00 sec)
```

Nah, sampai sejauh ini tidak sulit kan untuk mempelajari MySQL? Sekarang kita lanjutkan dengan cara-cara pengisian data. Yuukk..

3.2 Mengisi data ke dalam tabel

Kita akan mulai mengisi data karyawan ke dalam tabel. Perintah yang digunakan adalah **"INSERT INTO"**. Cara yang pertama sebagai berikut:

```
mysql> insert into karyawan
-> (nama, jenkel, kota, kodepos, tgllahir)
-> values
-> ("Ahmad Zobari", "L", "Bandung", "41011", "1977-10-02")
-> ;
Query OK, 1 row affected (0.17 sec)
```

Anda perhatikan bahwa dalam memasukkan data yang berjenis karakter, **selalu diapit** dengan tanda kutip ganda ("). Bisa juga digunakan tanda kutip tunggal ('). Tetapi **jangan dicampur** dengan tanda kutip ganda dan tanda kutip tunggal, misal: "Ahmad

Zobari'. Perhatikan juga pada penulisan tanggal lahir, menggunakan format "**tahun-bulan-tanggal**". Memang agak janggal. Tapi begitulah memang standar MySQL untuk format penulisan tanggal. Kalau Anda perhatikan lebih teliti, mengapa kita tidak memasukkan data untuk kolom "noid"? Ini karena sifat kolom noid yang *auto_increment*, sehingga dia akan secara otomatis berisi dengan angka 1, dan terus bertambah 1, seiring dengan penambahan data.

Nah, kita akan memasukkan 3 buah record lagi dengan cara:

```
mysql> insert into karyawan
-> (nama, jenkel, kota, kodepos, tgllahir)
-> values
-> ("Sundariwati", "P", "Bandung", "40123", "1978-11-12"),
-> ("Ryan Cakep", "L", "Jakarta", "12111", "1981-03-21"),
-> ("Zukarman", "L", "Bekasi", "17211", "1978-08-10")
-> ;
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Sekarang kita coba memasukkan data dengan cara yang kedua :

```
mysql> insert into karyawan
-> set nama="Yuliawati",
-> jenkel="P",
-> kota="Bogor",
-> kodepos="00000",
-> tgllahir="1982-06-09"
-> ;
Query OK, 1 row affected (0.05 sec)
```

Dan kita akan mencoba cara yang ketiga seperti berikut :

```
mysql> insert into karyawan
-> values ("Mawar", "P", "Bogor", "12345", "1985-07-07")
-> ;
ERROR 1136: Column count doesn't match value count at row 1
```

Lho, kok error? Kenapa nich? Perhatikan pada pesan error yang terjadi. Di sana dinyatakan bahwa jumlah kolom tidak sesuai dengan jumlah masukan data yang ada. OK..OK.. pelan-pelan yach. Kita ingat kalo jumlah kolom ada 6, yaitu noid, nama, jenkel, kota, kodepos dan tgllahir. Sedangkan data yang kita masukkan untuk 5 kolom saja, yaitu nama, jenkel, kota, kodepos dan tgllahir. Jadi, gimana duonk dengan nasibnya kolom noid? Masa dicuekin aja sich? Itu sebabnya jadi error. Walaupun kolom noid ini sifatnya

AUTO_INCREMENT, khusus untuk bentuk ketiga ini dia harus diisi juga dengan nilai DEFAULTNYA yaitu "NULL". Sehingga perintah diatas kita ubah sedikit menjadi :

```
mysql> insert into karyawan
      -> values ("NULL","Mawar","P","Bogor","12345","1985-07-07")
      -> ;
```

Query OK, 1 row affected (0.03 sec)

Kita sudah memasukkan beberapa data. Bagaimana untuk melihat data-data yang sudah kita masukkan tadi?

3.3 Melihat Data Pada Tabel

Kita bisa melihat data yang ada di dalam tabel dengan menggunakan perintah **"SELECT"**. Perintah SELECT adalah perintah yang akan sering kita gunakan nantinya. Kita mulai dengan cara yang paling sederhana dulu yaa..

```
mysql> select * from karyawan ;
+-----+-----+-----+-----+-----+-----+
|noid|nama          |jenkel|kota   |kodepos|tgllahir |
+-----+-----+-----+-----+-----+-----+
| 1|Ahmad Zobari |L      |Bandung|41011  |1977-10-02|
| 2|Sundariwati |P      |Bandung|40123  |1978-11-12|
| 3|Ryan Cakep  |L      |Jakarta|12111  |1981-03-21|
| 4|Zukarman    |L      |Bekasi |17211  |1978-08-10|
| 5|Yuliawati   |P      |Bogor  |00000  |1982-06-09|
| 6|Mawar       |P      |Bogor  |12345  |1985-07-07|
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)
```

Perintah di atas menampilkan seluruh data yang ada di dalam tabel karyawan, karena menggunakan tanda asterik "*" di dalam perintah **SELECT**. Bagaimana kalau kita hanya mau menampilkan kolom *nama* dan *jenis kelamin* saja?

```
mysql> select nama, jenkel from karyawan
      -> ;
```

```
+-----+-----+
| nama          | jenkel |
+-----+-----+
| Ahmad Zobari  | L      |
| Sundariwati   | P      |
| Ryan Cakep    | L      |
| Zukarman      | L      |
| Yuliawati     | P      |
| Mawar         | P      |
+-----+-----+
```

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

Kalau kita hanya mau menampilkan data-data karyawan yang berjenis kelamin perempuan saja, bagaimana caranya? Cukup dengan menambahkan perintah **"WHERE"** pada **"SELECT"**

```
mysql> select nama, jenkel from karyawan
-> where jenkel="P"
-> ;
```

nama	jenkel
Sundariwati	P
Yuliawati	P
Mawar	P

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

Kita tampilkan data berdasarkan **urutan nama karyawan** dengan menambahkan perintah **"ORDER BY"** pada **"SELECT"**:

```
mysql> select * from karyawan
-> order by nama ;
```

noid	nama	jenkel	kota	kodepos	tgllahir
1	Ahmad Zobari	L	Bandung	41011	1977-10-02
6	Mawar	P	Bogor	12345	1985-07-07
3	Ryan Cakep	L	Jakarta	12111	1981-03-21
2	Sundariwati	P	Bandung	40123	1978-11-12
5	Yuliawati	P	Bogor	00000	1982-06-09
4	Zukarman	L	Bekasi	17211	1978-08-10

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

Atau diurut berdasarkan **kota**:

```
mysql> select * from karyawan
-> order by kota ;
```

noid	nama	jenkel	kota	kodepos	tgllahir
1	Ahmad Zobari	L	Bandung	41011	1977-10-02
2	Sundariwati	P	Bandung	40123	1978-11-12
4	Zukarman	L	Bekasi	17211	1978-08-10
5	Yuliawati	P	Bogor	00000	1982-06-09
6	Mawar	P	Bogor	12345	1985-07-07
3	Ryan Cakep	L	Jakarta	12111	1981-03-21

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

Atau diurut berdasarkan **tanggal lahir** :

```
mysql> select * from karyawan
-> order by tgllahir ;
```

noid	nama	jenkel	kota	kodepos	tgllahir
1	Ahmad Zobari	L	Bandung	41011	1977-10-02
4	Zukarman	L	Bekasi	17211	1978-08-10
2	Sundariwati	P	Bandung	40123	1978-11-12
3	Ryan Cakep	L	Jakarta	12111	1981-03-21
5	Yuliawati	P	Bogor	00000	1982-06-09
6	Mawar	P	Bogor	12345	1985-07-07

6 rows in set (0.00 sec)

Nah kalo yang sekarang diurut berdasarkan nama, tetapi dengan **urutan terbalik (descending)**. Cukup dengan menambahkan perintah "**DESC**" pada SELECT:

```
mysql> select * from karyawan
-> order by nama DESC ;
```

noid	nama	jenkel	kota	kodepos	tgllahir
4	Zukarman	L	Bekasi	17211	1978-08-10
5	Yuliawati	P	Bogor	00000	1982-06-09
2	Sundariwati	P	Bandung	40123	1978-11-12
3	Ryan Cakep	L	Jakarta	12111	1981-03-21
6	Mawar	P	Bogor	12345	1985-07-07
1	Ahmad Zobari	L	Bandung	41011	1977-10-02

6 rows in set (0.00 sec)

Bisa juga kalau yang diurutnya adalah **tanggal lahir** secara urutan terbalik (**descending**):

```
mysql> select * from karyawan
-> order by tgllahir DESC ;
```

noid	nama	jenkel	kota	kodepos	tgllahir
6	Mawar	P	Bogor	12345	1985-07-07
5	Yuliawati	P	Bogor	00000	1982-06-09
3	Ryan Cakep	L	Jakarta	12111	1981-03-21
2	Sundariwati	P	Bandung	40123	1978-11-12
4	Zukarman	L	Bekasi	17211	1978-08-10
1	Ahmad Zobari	L	Bandung	41011	1977-10-02

6 rows in set (0.00 sec)

Ternyata kita perlu menambahkan sebuah kolom field lagi, yaitu **kolom gaji**. Kolom Gaji merupakan kolom **numerik** yang menampung data gaji pokok karyawan per bulannya. Jadi, yang kita perlukan adalah jenis data **INTeger** dengan lebar data 12 digit. Penerapannya sebagai berikut dengan menggunakan perintah **ALTER**.

```
mysql> alter table karyawan
      -> ADD gaji INT(12) NOT NULL default 0
      -> ;
Query OK, 6 rows affected (0.25 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Kita periksa struktur tabelnya dulu:

```
mysql> describe karyawan ;
+-----+-----+-----+-----+-----+-----+
|Field  |Type      |Null| Key|Default|Extra      |
+-----+-----+-----+-----+-----+-----+
|noid   |int(10)   |NO  | PRI|NULL   |auto_increment|
|nama   |varchar(50)|NO  |    |       |              |
|jenkel |char(1)   |YES  |    |NULL   |              |
|kota   |varchar(25)|NO  |    |       |              |
|kodepos|char(5)   |NO  |    |       |              |
|tgllahir|date     |YES  |    |NULL   |              |
|gaji   |int(12)   |NO  |    |0      |              |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

Ya, kolom gaji sudah ditambahkan ke dalam tabel *karyawan*. Sekarang kita akan menambahkan data gaji kepada tiap-tiap karyawan yang ada. Untuk memudahkan, kita tampilkan dulu semua data yang ada di tabel *karyawan*:

```
mysql> select * from karyawan ;
+-----+-----+-----+-----+-----+-----+-----+
| noid | nama          | jenkel | kota    | kodepos | tgllahir | gaji |
+-----+-----+-----+-----+-----+-----+-----+
| 1    | Ahmad Zobari  | L      | Bandung | 41011   | 1977-10-02 | 0 |
| 2    | Sundariwati  | P      | Bandung | 40123   | 1978-11-12 | 0 |
| 3    | Ryan Cakep   | L      | Jakarta | 12111   | 1981-03-21 | 0 |
| 4    | Zukarman     | L      | Bekasi  | 17211   | 1978-08-10 | 0 |
| 5    | Yuliawati    | P      | Bogor   | 00000   | 1982-06-09 | 0 |
| 6    | Mawar        | P      | Bogor   | 12345   | 1985-07-07 | 0 |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

3.4 Meng-Update Data Pada Tabel

Sekarang kita masukkan data gaji masing-masing karyawan dengan menggunakan perintah **UPDATE**. Kita mulai dari **Ahmad Zobari**, dengan **noid=1**:

```
mysql> update karyawan
      -> set gaji=1000000
      -> where noid=1 ;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Periksa dulu hasilnya:

```
mysql> select * from karyawan
      -> where noid=1 ;
+-----+-----+-----+-----+-----+-----+
|noid|nama          |jenkel|kota   |kodepos|tgllahir |gaji   |
+-----+-----+-----+-----+-----+-----+
|  1 |Ahmad Zobari |L      |Bandung|41011  |1977-10-02|1000000|
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Kita lanjutkan dengan karyawan lainnya, seperti Sundariwati dengan **noid=2**, Ryan Cakep dengan **noid=3**, dan seterusnya. Sayangnya, perintah ini hanya bisa dilakukan satu per satu. Jadi, Anda harus sabar menjalankan perintah di bawah ini yaa ...:

```
mysql> update karyawan
      -> set gaji=1250000 where noid=2 ;
Query OK, 1 row affected (0.39 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> update karyawan
      -> set gaji=1500000 where noid=3 ;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> update karyawan
      -> set gaji=1750000 where noid=4 ;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> update karyawan
      -> set gaji=2000000 where noid=5 ;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> update karyawan
      -> set gaji=2250000 where noid=6 ;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Kita periksa semua hasilnya:

```
mysql> select * from karyawan ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
1	Ahmad Zobari	L	Bandung	41011	1977-10-02	1000000
2	Sundariwati	P	Bandung	40123	1978-11-12	1250000
3	Ryan Cakep	L	Jakarta	12111	1981-03-21	1500000
4	Zukarman	L	Bekasi	17211	1978-08-10	1750000
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
6	Mawar	P	Bogor	12345	1985-07-07	2250000

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

Cukup mudah kan? Nah, itulah dasar-dasar menggunakan perintah MySQL. Sekarang kita membutuhkan lebih banyak data untuk latihan kita. Ya minimal sekitar 30-an data lagi. Tapi apakah ada cara lain yang lebih mudah dibanding harus mengetikkan datanya satu per satu? Kan kalo kita ketik satu per satu, faktor resiko kesalahan ketik karena faktor kelelahan, dan sebagainya. Untungnya untuk pemasukan data masal kita bisa menggunakan cara yang lebih mudah.

3.5 Pemasukan Data Secara Masal

Untuk pemasukan data secara masal, kita menggunakan data-data yang telah ditulis dalam sebuah file teks biasa. File ini kita namakan **tambahdata.txt**, dan untuk contoh ini kita simpan di dalam **folder C:\Data**. Anda dapat mengunduh (*download*) file tambahdata.txt dari situs ini. Silakan kunjungi <http://dwiaprisetyorini.blogspot.com/2009/09/mysqlstmik-db.html> untuk download file tambahdata.txt

Perintah yang kita gunakan adalah "".

```
mysql> load data local infile 'C:\\data\\tambahdata.txt'
-> into table karyawan
-> fields terminated by ','
-> lines terminated by '\n'
-> ;
```

```
Query OK, 36 rows affected, 36 warnings (0.47 sec)
Records: 36 Deleted: 0 Skipped: 0 Warnings: 0
```

Catatan:

Perhatikan di atas bahwa digunakan 2 garis miring (\\) sebagai pembatas nama direktori, bukannya tunggal (\). Ini karena tanda \" dianggap sebagai karakter khusus oleh MySQL (disebut sebagai Escape Character).

Kelebihan menggunakan cara tersebut di atas adalah kita bisa "mengkawinkan" data-data dari program database apa saja (seperti Microsoft Access, dBASE, FoxPro, dsb) ke dalam MySQL dengan syarat diubah dulu kedalam bentuk file teks.

Ada pesan dari sistem kalo perintah mysql berhasil dilaksanakan. Tapi, tidak ada salahnya kalo kita periksa juga kan...

Sekarang kita lihat hasilnya di tabel *karyawan*:

```
mysql> select * from karyawan ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
1	Ahmad Zobari	L	Bandung	41011	1977-10-02	1000000
2	Sundariwati	P	Bandung	40123	1978-11-12	1250000
3	Ryan Cakep	L	Jakarta	12111	1981-03-21	1500000
4	Zukarman	L	Bekasi	17211	1978-08-10	1750000
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
6	Mawar	P	Bogor	12345	1985-07-07	2250000
7	Zobari	L	Jakarta	41011	1976-10-02	1100000
8	Melia	P	Bandung	40123	1979-11-12	1200000
9	Zanda Cute	L	Jakarta	12111	1980-03-21	1300000
10	Maman	L	Bekasi	17211	1977-08-10	1400000
11	Yenny	P	Bogor	00000	1985-06-09	1150000
12	Rossa	P	Jakarta	12345	1987-07-07	1350000
13	Dadan	L	Bandung	41011	1975-10-02	1450000
14	Wawan	P	Semarang	40123	1971-11-12	1600000
15	The Cute	L	Jakarta	12111	1977-03-21	1700000
16	Marpaung	L	Surabaya	17211	1988-08-10	1800000
17	Yono	P	Bogor	00000	1989-06-09	1900000
18	Dian	P	Jakarta	12345	1980-07-07	1650000
19	Donno	L	Bandung	41011	1971-10-02	1850000
20	Ratu	P	Yogyakarta	40123	1972-11-12	1950000
21	Bambang	L	Jakarta	12111	1982-03-21	2100000
22	Dadang	L	Surabaya	17211	1977-08-10	2200000
23	Yuliawati	P	Bogor	00000	1974-06-09	2300000
24	Miranda	P	Bogor	12345	1980-07-07	2400000
25	Subur	L	Bandung	41011	1977-10-02	2150000
26	Banowati	P	Malang	40123	1978-11-12	2350000
27	Gungun	L	Jakarta	12111	1981-03-21	2450000
28	Gunadi	L	Bekasi	17211	1978-08-10	2125000
29	Yossy	P	Bogor	00000	1982-06-09	2225000
30	Melia	P	Malang	12345	1981-07-07	2325000
31	Anwar	L	Purwakarta	41011	1972-10-02	2425000
32	Susilowati	P	Bandung	40123	1973-11-12	1125000
33	Rahmat	L	Jakarta	12111	1977-03-21	1225000

34	Zamzam	L	Bekasi	17211	1974-08-10	1325000
35	Nenny	P	Medan	00000	1972-06-09	1425000
36	Mardiatun	P	Bogor	12345	1975-07-07	1625000
37	Andika	L	Bandung	41011	1978-10-02	1725000
38	Siti	P	Medan	40123	1988-11-12	1825000
39	Rohimat	L	Jakarta	12111	1980-03-21	1925000
40	Beno	L	Bekasi	17211	1978-08-10	1175000
41	Yanti	P	Jakarta	00000	1981-06-09	1275000
42	Miranti	P	Medan	12345	1975-07-07	1375000

42 rows in set (0.00 sec)

Kita sudah memiliki lebih banyak data. Cukuplah untuk bahan latihan-latihan berikutnya...

Soal Latihan

Buka database dbKursus yang telah Anda buat. Kemudian buka struktur tabel peserta.

- Ganti field nomor dengan idPeserta, tipe data sama
- Ganti lebar data field email menjadi 50
- Tambahkan field tempatlhr dan tglhr
- Isikan data pada table tersebut, minimal 10 data.

Daftar Pustaka

<http://dev.mysql.com>

[http://www.rohmat-mimi.com/download/MODUL PRAKTIKUM MY SQL-BASIS DATA](http://www.rohmat-mimi.com/download/MODUL_PRAKTIKUM_MY_SQL-BASIS_DATA)

<http://www.mysql.com>

<http://www.arbiedesign.com/index.php>

Tim Training SMK-TI.Modul MySQL

BAB IV

Operator Pembandingan dan Operator Logika

Standar Kompetensi :

1. Mahasiswa dapat mengetahui, memahami, menguasai dan mampu mengimplementasi teori, konsep dan prinsip pemrograman database MySQL dengan logika pemrograman yang benar, ringkas, dan tepat dalam penerapannya di bidang teknologi informasi

Kompetensi Dasar	Indikator
1.10 Mengetahui Operator Pembandingan dan Operator Logika	<ul style="list-style-type: none">▪ Menggunakan operator pembandingan▪ Menggunakan operator logika▪ Mengetahui fungsi masing-masing operator pembandingan dan operator logika
1.11 Mengetahui Fungsi Statistik Dasar	<ul style="list-style-type: none">▪ Menggunakan fungsi SUM, AVG, MIN, MAX pada struktur bahasa SQL

Materi

4.1 Operator Pembandingan dan Operator Logika

Sudah saatnya kita melangkah ke permainan data yang lebih mengasyikkan lagi dengan menggunakan dua operator, yaitu **Operator Pembandingan** dan **Operator Logika**. Kedua jenis operator ini akan sering digunakan dalam proses "query" data.

A. Operator Pembandingan

Tabel 4.1 Operator Pembandingan

Operator Pembandingan	Keterangan
Lebih besar	>
Lebih kecil	<
Lebih besar atau sama dengan	>=
Lebih kecil atau sama dengan	<=
Sama dengan	=
Tidak sama dengan	<>

B. Operator Logika

Tabel 4.2 Operator Logika

Operator Logika	Keterangan
Dan	AND atau &&
Atau	OR atau
Lebih besar atau sama dengan	NOT atau !
Lebih kecil atau sama dengan	<=
Tidak sama dengan	<>

Berikut ini adalah penerapan dari kedua operator di atas : kita tampilkan data karyawan yang tanggal lahirnya **sebelum tanggal 1 Januari 1980**, dan tampilan data diurut berdasarkan nama. Cukup kolom nama, jenis kelamin, tanggal lahir yang ditampilkan:

```
mysql> select nama, jenkel, tgllahir  
-> from karyawan  
-> where tgllahir < "1980-01-01"  
-> order by nama ;
```

```
+-----+-----+-----+  
| nama          | jenkel | tgllahir |  
+-----+-----+-----+  
| Ahmad Zobari  | L      | 1977-10-02 |  
| Andika        | L      | 1978-10-02 |  
| Anwar         | L      | 1972-10-02 |  
| Banowati      | P      | 1978-11-12 |  
| Beno          | L      | 1978-08-10 |  
| Dadan         | L      | 1975-10-02 |  
| Dadang        | L      | 1977-08-10 |  
| Donno         | L      | 1971-10-02 |  
| Gunadi        | L      | 1978-08-10 |  
| Maman         | L      | 1977-08-10 |  
| Mardiatun     | P      | 1975-07-07 |  
| Melia         | P      | 1979-11-12 |  
| Miranti       | P      | 1975-07-07 |  
| Nenny         | P      | 1972-06-09 |  
| Rahmat        | L      | 1977-03-21 |  
| Ratu          | P      | 1972-11-12 |  
| Zobari        | L      | 1976-10-02 |  
| Subur         | L      | 1977-10-02 |  
| Sundariwati   | P      | 1978-11-12 |  
| Susilowati    | P      | 1973-11-12 |  
| The Cute      | L      | 1977-03-21 |  
| Wawan         | P      | 1971-11-12 |  
| Yuliawati     | P      | 1974-06-09 |  
| Zamzam        | L      | 1974-08-10 |  
| Zukarman      | L      | 1978-08-10 |  
+-----+-----+-----+
```

25 rows in set (0.00 sec)

MySQL memiliki *kelonggaran* dalam penulisan tanggal selama formatnya mengikuti aturan "**tahun-bulan-tanggal**". Misal "1971-11-12" dapat ditulis 1971-11-12, atau 1971#11#12, atau 19711112, atau 711112.

Kita lihat contohnya dibawah ini dimana tanggal "1980-01-01" ditulis dengan **19800101**

```
mysql> select nama, jenkel, tgllahir
-> from karyawan
-> where tgllahir < 19800101
-> and jenkel="L"
-> order by nama ;
```

nama	jenkel	tgllahir
Ahmad Zobari	L	1977-10-02
Andika	L	1978-10-02
Anwar	L	1972-10-02
Beno	L	1978-08-10
Dadan	L	1975-10-02
Dadang	L	1977-08-10
Donno	L	1971-10-02
Gunadi	L	1978-08-10
Maman	L	1977-08-10
Rahmat	L	1977-03-21
Zobari	L	1976-10-02
Subur	L	1977-10-02
The Cute	L	1977-03-21
Zamzam	L	1974-08-10
Zukarman	L	1978-08-10

15 rows in set (0.00 sec)

Kita lihat contohnya di bawah ini bila tanggal "1980-01-01" ditulis dengan cara **800101**.

```
mysql> select nama, jenkel, tgllahir
-> from karyawan
-> where tgllahir < 800101
-> and jenkel="L"
-> order by nama;
```



```

+-----+-----+-----+
| nama          | jenkel | tgllahir      |
+-----+-----+-----+
| Ahmad Zobari  | L      | 1977-10-02    |
| Andika        | L      | 1978-10-02    |
| Anwar         | L      | 1972-10-02    |
| Beno          | L      | 1978-08-10    |
| Dadan         | L      | 1975-10-02    |
| Dadang        | L      | 1977-08-10    |
| Donno         | L      | 1971-10-02    |
| Gunadi        | L      | 1978-08-10    |
| Maman         | L      | 1977-08-10    |
| Rahmat        | L      | 1977-03-21    |
| Zobari        | L      | 1976-10-02    |
| Subur         | L      | 1977-10-02    |
| The Cute      | L      | 1977-03-21    |
| Zamzam        | L      | 1974-08-10    |
| Zukarman      | L      | 1978-08-10    |
+-----+-----+-----+
15 rows in set (0.00 sec)

```

Kita lihat contohnya di bawah ini bila tanggal "1980-01-01" ditulis dengan cara **"1980#01#01"**.

```

mysql> select nama, jenkel, tgllahir
      -> from karyawan
      -> where tgllahir < "1980#01#01"
      -> and jenkel="L"
      -> order by nama ;

```

```

+-----+-----+-----+
| nama          | jenkel | tgllahir      |
+-----+-----+-----+
| Ahmad Zobari  | L      | 1977-10-02    |
| Andika        | L      | 1978-10-02    |
| Anwar         | L      | 1972-10-02    |
| Beno          | L      | 1978-08-10    |
| Dadan         | L      | 1975-10-02    |
| Dadang        | L      | 1977-08-10    |
| Donno         | L      | 1971-10-02    |
| Gunadi        | L      | 1978-08-10    |
| Maman         | L      | 1977-08-10    |
| Rahmat        | L      | 1977-03-21    |
| Zobari        | L      | 1976-10-02    |
| Subur         | L      | 1977-10-02    |
| The Cute      | L      | 1977-03-21    |
| Zamzam        | L      | 1974-08-10    |
| Zukarman      | L      | 1978-08-10    |
+-----+-----+-----+
15 rows in set (0.00 sec)

```

Kita lihat contohnya di bawah ini bila tanggal "1980-01-01" ditulis dengan cara **"1980.01.01"**.

```
mysql> select nama,jenkel, tgllahir
      -> from karyawan
      -> where tgllahir < "1980.01.01"
      -> and jenkel="L"
      -> order by nama ;
```

```
+-----+-----+-----+
| nama          | jenkel | tgllahir      |
+-----+-----+-----+
| Ahmad Zobari  | L      | 1977-10-02    |
| Andika        | L      | 1978-10-02    |
| Anwar         | L      | 1972-10-02    |
| Beno          | L      | 1978-08-10    |
| Dadan         | L      | 1975-10-02    |
| Dadang        | L      | 1977-08-10    |
| Donno         | L      | 1971-10-02    |
| Gunadi        | L      | 1978-08-10    |
| Maman         | L      | 1977-08-10    |
| Rahmat        | L      | 1977-03-21    |
| Zobari        | L      | 1976-10-02    |
| Subur         | L      | 1977-10-02    |
| The Cute      | L      | 1977-03-21    |
| Zamzam        | L      | 1974-08-10    |
| Zukarman      | L      | 1978-08-10    |
+-----+-----+-----+
15 rows in set (0.00 sec)
```

Perhatikan semua hasil di atas sama walaupun cara penulisan tanggalnya berbeda-beda (tetapi formatnya tetap mengikuti "tahun-bulan-tanggal").

Sekarang kita tampilkan data karyawan yang tanggal lahirnya **antara tanggal 1 Januari 1980 dan 31 Desember 1985**, dan tampilan data diurut berdasarkan nama. Cukup hanya kolom nama, jenis kelamin dan tanggal lahir saja yang ditampilkan :

```
mysql> select nama, jenkel, tgllahir
      -> from karyawan
      -> where tgllahir >= "1980-01-01"
      -> and tgllahir <= "1985-12-31"
      -> order by nama ;
```

```

+-----+-----+-----+
| nama      | jenkel | tgllahir  |
+-----+-----+-----+
| Bambang   | L      | 1982-03-21 |
| Dian      | P      | 1980-07-07 |
| Gungun    | L      | 1981-03-21 |
| Mawar     | P      | 1985-07-07 |
| Melia     | P      | 1981-07-07 |
| Miranda   | P      | 1980-07-07 |
| Rohimat   | L      | 1980-03-21 |
| Ryan Cakep | L      | 1981-03-21 |
| Yanti     | P      | 1981-06-09 |
| Yenny     | P      | 1985-06-09 |
| Yossy     | P      | 1982-06-09 |
| Yuliawati | P      | 1982-06-09 |
| Zanda Cute | L      | 1980-03-21 |
+-----+-----+-----+
13 rows in set (0.00 sec)

```

Sekarang kita tampilkan data karyawan yang tanggal lahirnya **antara tanggal 1 Januari 1980 dan 31 Desember 1985**, dan tampilan data diurut berdasarkan nama. Cukup hanya kolom nama, jenis kelamin dan tanggal lahir saja, serta hanya yang berjenis kelamin laki-laki yang ditampilkan:

```

mysql> select nama, jenkel, tgllahir
-> from karyawan
-> where tgllahir >= "1980-01-01"
-> and tgllahir <= "1985-12-31"
-> and jenkel="L"
-> order by nama ;
+-----+-----+-----+
| nama      | jenkel | tgllahir  |
+-----+-----+-----+
| Bambang   | L      | 1982-03-21 |
| Gungun    | L      | 1981-03-21 |
| Rohimat   | L      | 1980-03-21 |
| Ryan Cakep | L      | 1981-03-21 |
| Zanda Cute | L      | 1980-03-21 |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

Bagaimana, semakin menarik kan? Kita lanjutkan dengan menampilkan semua data karyawan dengan usianya pada saat ini. Untuk masalah ini memang cukup panjang solusinya. Tidak apa-apa, kita coba saja ya. Disini kita memerlukan bantuan beberapa fungsi-fungsi yang sudah disediakan oleh MySQL. Kita lihat dulu ya :

```
mysql> select nama, tgllahir,
-> current_date AS SEKARANG,
-> (year(current_date) - year(tgllahir))
-> - (right(current_date,5) < right(tgllahir,5)) AS USIA
-> from karyawan ;
```

nama	tgllahir	SEKARANG	USIA
Ahmad Zobari	1977-10-02	2007-08-30	29
Sundariwati	1978-11-12	2007-08-30	28
Ryan Cakep	1981-03-21	2007-08-30	26
Zukarman	1978-08-10	2007-08-30	29
Yuliawati	1982-06-09	2007-08-30	25
Mawar	1985-07-07	2007-08-30	22
Zobari	1976-10-02	2007-08-30	30
Melia	1979-11-12	2007-08-30	27
Zanda Cute	1980-03-21	2007-08-30	27
Maman	1977-08-10	2007-08-30	30
Yenny	1985-06-09	2007-08-30	22
Rossa	1987-07-07	2007-08-30	20
Dadan	1975-10-02	2007-08-30	31
Wawan	1971-11-12	2007-08-30	35
The Cute	1977-03-21	2007-08-30	30
Marpaung	1988-08-10	2007-08-30	19
Yono	1989-06-09	2007-08-30	18
Dian	1980-07-07	2007-08-30	27
Donno	1971-10-02	2007-08-30	35
Ratu	1972-11-12	2007-08-30	34
Bambang	1982-03-21	2007-08-30	25
Dadang	1977-08-10	2007-08-30	30
Yuliawati	1974-06-09	2007-08-30	33
Miranda	1980-07-07	2007-08-30	27
Subur	1977-10-02	2007-08-30	29
Banowati	1978-11-12	2007-08-30	28
Gungun	1981-03-21	2007-08-30	26
Gunadi	1978-08-10	2007-08-30	29
Yossy	1982-06-09	2007-08-30	25
Melia	1981-07-07	2007-08-30	26
Anwar	1972-10-02	2007-08-30	34
Susilowati	1973-11-12	2007-08-30	33
Rahmat	1977-03-21	2007-08-30	30
Zamzam	1974-08-10	2007-08-30	33
Nenny	1972-06-09	2007-08-30	35
Mardiatun	1975-07-07	2007-08-30	32
Andika	1978-10-02	2007-08-30	28
Siti	1988-11-12	2007-08-30	18
Rohimat	1980-03-21	2007-08-30	27
Beno	1978-08-10	2007-08-30	29
Yanti	1981-06-09	2007-08-30	26
Miranti	1975-07-07	2007-08-30	32

42 rows in set (0.00 sec)

Kita lanjutkan dengan menampilkan data karyawan yang **usianya sama atau dibawah 25 tahun**. Nah bagaimana caranya?:

```
mysql> select nama, tgllahir,
-> current_date AS SEKARANG,
-> (year(current_date) - year(tgllahir))
-> - (right(current_date,5) < right(tgllahir,5))
-> AS USIA
-> from karyawan
-> where ((year(current_date) - year(tgllahir))
-> - (right(current_date,5) < right(tgllahir,5)))
-> <= 25 ;
```

nama	tgllahir	SEKARANG	USIA
Yuliawati	1982-06-09	2007-08-30	25
Mawar	1985-07-07	2007-08-30	22
Yenny	1985-06-09	2007-08-30	22
Rossa	1987-07-07	2007-08-30	20
Marpaung	1988-08-10	2007-08-30	19
Yono	1989-06-09	2007-08-30	18
Bambang	1982-03-21	2007-08-30	25
Yosy	1982-06-09	2007-08-30	25
Siti	1988-11-12	2007-08-30	18

9 rows in set (0.00 sec)

Cukup panjang perintahnya ya. Disini kita menggunakan fungsi **CURRENT_DATE** yang mengambil nilai dari tanggal saat ini pada sistem komputer Anda. **YEAR** adalah fungsi yang mengambil nilai tahun. Kemudian **AS** adalah singkatan dari **Alias**, yang seolah-olah memberikan nama lain (*alias name*) pada kolom atau pada hasil suatu proses. Sedangkan **RIGHT** adalah fungsi yang mengambil nilai dari sekian karakter dari sisi kanan sebuah target. Misal **RIGHT("APRI", 3)** maka akan menghasilkan karakter "PRI".

Keterangan lengkap dari fungsi-fungsi MySQL ini dapat Anda baca pada <http://dwiaprisetyorini.blogspot.com/2010/01/fungsi-string-di-mysql.html>

Baik kita lanjutkan tutorial ini dengan perintah-perintah lainnya. Mari...

Kita akan menampilkan karyawan yang kota kelahirannya di "Bandung":

```
mysql> select * from karyawan
      -> where kota="Bandung" ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
1	Ahmad Zobari	L	Bandung	41011	1977-10-02	1000000
2	Sundariwati	P	Bandung	40123	1978-11-12	1250000
8	Melia	P	Bandung	40123	1979-11-12	1200000
13	Dadan	L	Bandung	41011	1975-10-02	1450000
19	Donno	L	Bandung	41011	1971-10-02	1850000
25	Subur	L	Bandung	41011	1977-10-02	2150000
32	Susilowati	P	Bandung	40123	1973-11-12	1125000
37	Andika	L	Bandung	41011	1978-10-02	1725000

8 rows in set (0.03 sec)

Kita tampilkan karyawan yang kota kelahirannya **bukan** di Bandung:

```
mysql> select * from karyawan
      -> where kota != "bandung" ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
3	Ryan Cakep	L	Jakarta	12111	1981-03-21	1500000
4	Zukarman	L	Bekasi	17211	1978-08-10	1750000
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
6	Mawar	P	Bogor	12345	1985-07-07	2250000
7	Zobari	L	Jakarta	41011	1976-10-02	1100000
9	Zanda Cute	L	Jakarta	12111	1980-03-21	1300000
10	Maman	L	Bekasi	17211	1977-08-10	1400000
11	Yenny	P	Bogor	00000	1985-06-09	1150000
12	Rossa	P	Jakarta	12345	1987-07-07	1350000
14	Wawan	P	Semarang	40123	1971-11-12	1600000
15	The Cute	L	Jakarta	12111	1977-03-21	1700000
16	Marpaung	L	Surabaya	17211	1988-08-10	1800000
17	Yono	P	Bogor	00000	1989-06-09	1900000
18	Dian	P	Jakarta	12345	1980-07-07	1650000
20	Ratu	P	Yogyakarta	40123	1972-11-12	1950000
21	Bambang	L	Jakarta	12111	1982-03-21	2100000
22	Dadang	L	Surabaya	17211	1977-08-10	2200000
23	Yuliawati	P	Bogor	00000	1974-06-09	2300000
24	Miranda	P	Bogor	12345	1980-07-07	2400000
26	Banowati	P	Malang	40123	1978-11-12	2350000
27	Gungun	L	Jakarta	12111	1981-03-21	2450000
28	Gunadi	L	Bekasi	17211	1978-08-10	2125000
29	Yossy	P	Bogor	00000	1982-06-09	2225000
30	Melia	P	Malang	12345	1981-07-07	2325000
31	Anwar	L	Purwakarta	41011	1972-10-02	2425000
33	Rahmat	L	Jakarta	12111	1977-03-21	1225000

```

| 34| Zamzam      |L      |Bekasi      |17211      | 1974-08-10|1325000|
| 35| Nenny        |P      |Medan       |00000      | 1972-06-09|1425000|
| 36| Mardiatun    |P      |Bogor       |12345      | 1975-07-07|1625000|
| 38| Siti         |P      |Medan       |40123      | 1988-11-12|1825000|
| 39| Rohimat      |L      |Jakarta     |12111      | 1980-03-21|1925000|
| 40| Beno         |L      |Bekasi      |17211      | 1978-08-10|1175000|
| 41| Yanti        |P      |Jakarta     |00000      | 1981-06-09|1275000|
| 42| Miranti      |P      |Medan       |12345      | 1975-07-07|1375000|
+-----+-----+-----+-----+-----+-----+
34 rows in set (0.00 sec)

```

Perintah di atas dapat juga menggunakan tanda "<>", dan hasilnya tetap sama dengan di atas:

```

mysql> select * from karyawan
-> where kota <> "bandung" ;
+-----+-----+-----+-----+-----+-----+-----+
| noid | nama      | jenkel | kota      | kodepos | tgllahir | gaji   |
+-----+-----+-----+-----+-----+-----+-----+
| 3    | Ryan Cakep | L      | Jakarta   | 12111   | 1981-03-21 | 1500000 |
| 4    | Zukarman   | L      | Bekasi    | 17211   | 1978-08-10 | 1750000 |
| 5    | Yuliatwati | P      | Bogor     | 00000   | 1982-06-09 | 2000000 |
| 6    | Mawar      | P      | Bogor     | 12345   | 1985-07-07 | 2250000 |
| 7    | Zobari     | L      | Jakarta   | 41011   | 1976-10-02 | 1100000 |
| 9    | Zanda Cute | L      | Jakarta   | 12111   | 1980-03-21 | 1300000 |
| 10   | Maman      | L      | Bekasi    | 17211   | 1977-08-10 | 1400000 |
| 11   | Yenny      | P      | Bogor     | 00000   | 1985-06-09 | 1150000 |
| 12   | Rossa      | P      | Jakarta   | 12345   | 1987-07-07 | 1350000 |
| 14   | Wawan      | P      | Semarang  | 40123   | 1971-11-12 | 1600000 |
| 15   | The Cute   | L      | Jakarta   | 12111   | 1977-03-21 | 1700000 |
| 16   | Marpaung   | L      | Surabaya  | 17211   | 1988-08-10 | 1800000 |
| 17   | Yono       | P      | Bogor     | 00000   | 1989-06-09 | 1900000 |
| 18   | Dian       | P      | Jakarta   | 12345   | 1980-07-07 | 1650000 |
| 20   | Ratu       | P      | Yogyakarta | 40123   | 1972-11-12 | 1950000 |
| 21   | Bambang    | L      | Jakarta   | 12111   | 1982-03-21 | 2100000 |
| 22   | Dadang     | L      | Surabaya  | 17211   | 1977-08-10 | 2200000 |
| 23   | Yuliatwati | P      | Bogor     | 00000   | 1974-06-09 | 2300000 |
| 24   | Miranda    | P      | Bogor     | 12345   | 1980-07-07 | 2400000 |
| 26   | Banowati   | P      | Malang    | 40123   | 1978-11-12 | 2350000 |
| 27   | Gunung     | L      | Jakarta   | 12111   | 1981-03-21 | 2450000 |
| 28   | Gunadi     | L      | Bekasi    | 17211   | 1978-08-10 | 2125000 |
| 29   | Yossy      | P      | Bogor     | 00000   | 1982-06-09 | 2225000 |
| 30   | Melia      | P      | Malang    | 12345   | 1981-07-07 | 2325000 |
| 31   | Anwar      | L      | Purwakarta | 41011   | 1972-10-02 | 2425000 |
| 33   | Rahmat     | L      | Jakarta   | 12111   | 1977-03-21 | 1225000 |
| 34   | Zamzam     | L      | Bekasi    | 17211   | 1974-08-10 | 1325000 |
| 35   | Nenny      | P      | Medan     | 00000   | 1972-06-09 | 1425000 |
| 36   | Mardiatun  | P      | Bogor     | 12345   | 1975-07-07 | 1625000 |
| 38   | Siti       | P      | Medan     | 40123   | 1988-11-12 | 1825000 |
| 39   | Rohimat    | L      | Jakarta   | 12111   | 1980-03-21 | 1925000 |
| 40   | Beno       | L      | Bekasi    | 17211   | 1978-08-10 | 1175000 |
| 41   | Yanti      | P      | Jakarta   | 00000   | 1981-06-09 | 1275000 |
| 42   | Miranti    | P      | Medan     | 12345   | 1975-07-07 | 1375000 |
+-----+-----+-----+-----+-----+-----+-----+
34 rows in set (0.00 sec)

```

Sekarang kita tampilkan karyawan dengan kota kelahiran bukan di Bandung, Jakarta dan Bekasi. Tampilan data diurut berdasarkan nama kota. Bagaimana bentuk perintahnya?

```
mysql> select * from karyawan
-> where kota <> "bandung"
-> and kota <> "Jakarta"
-> and kota <> "Bekasi"
-> order by kota ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
36	Mardiatun	P	Bogor	12345	1975-07-07	1625000
29	Yossy	P	Bogor	00000	1982-06-09	2225000
24	Miranda	P	Bogor	12345	1980-07-07	2400000
23	Yuliawati	P	Bogor	00000	1974-06-09	2300000
17	Yono	P	Bogor	0000	1989-06-09	1900000
6	Mawar	P	Bogor	12345	1985-07-07	2250000
11	Yenny	P	Bogor	00000	1985-06-09	1150000
30	Melia	P	Malang	12345	1981-07-07	2325000
26	Banowati	P	Malang	40123	1978-11-12	2350000
38	Siti	P	Medan	40123	1988-11-12	1825000
35	Nenny	P	Medan	00000	1972-06-09	1425000
42	Miranti	P	Medan	12345	1975-07-07	1375000
31	Anwar	L	Purwakarta	41011	1972-10-02	2425000
14	Wawan	P	Semarang	40123	1971-11-12	1600000
16	Marpaung	L	Surabaya	17211	1988-08-10	1800000
22	Dadang	L	Surabaya	17211	1977-08-10	2200000
20	Ratu	P	Yogyakarta	40123	1972-11-12	1950000

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

Hampir mirip dengan perintah di atas, tetapi selain diurut berdasarkan kota, nama karyawan pun ikut diurut. Kita coba dengan perintah dibawah:

```
mysql> select * from karyawan
-> where kota <> "bandung"
-> and kota <> "Jakarta"
-> and kota <> "Bekasi"
-> order by kota and nama ;
```


noid	nama	jenkel	kota	kodepos	tgllahir	gaji
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
38	Siti	P	Medan	40123	1988-11-12	1825000
36	Mardiatun	P	Bogor	12345	1975-07-07	1625000
35	Nenny	P	Medan	00000	1972-06-09	1425000
31	Anwar	L	Purwakarta	41011	1972-10-02	2425000
30	Melia	P	Malang	12345	1981-07-07	2325000
29	Yosy	P	Bogor	00000	1982-06-09	2225000
26	Banowati	P	Malang	40123	1978-11-12	2350000
24	Miranda	P	Bogor	12345	1980-07-07	2400000
23	Yuliawati	P	Bogor	00000	1974-06-09	2300000
22	Dadang	L	Surabaya	17211	1977-08-10	2200000
20	Ratu	P	Yogyakarta	40123	1972-11-12	1950000
17	Yono	P	Bogor	00000	1989-06-09	1900000
16	Marpaung	L	Surabaya	17211	1988-08-10	1800000
14	Wawan	P	Semarang	40123	1971-11-12	1600000
11	Yenny	P	Bogor	00000	1985-06-09	1150000
6	Mawar	P	Bogor	12345	1985-07-07	2250000
42	Miranti	P	Medan	12345	1975-07-07	1375000

18 rows in set (0.00 sec)

Coba perhatikan hasilnya. Apakah ini hasil yang kita inginkan? Keliatannya ada yang tidak beres... Kita coba lagi dengan menambah tanda kurung (dan) pada bagian perintah

"ORDER BY", siapa tahu berhasil:

```
mysql> select * from karyawan
-> where kota <> "bandung"
-> and kota <> "Jakarta"
-> and kota <> "Bekasi"
-> order by (kota and nama) ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
38	Siti	P	Medan	40123	1988-11-12	1825000
36	Mardiatun	P	Bogor	12345	1975-07-07	1625000
35	Nenny	P	Medan	00000	1972-06-09	1425000
31	Anwar	L	Purwakarta	41011	1972-10-02	2425000
30	Melia	P	Malang	12345	1981-07-07	2325000
29	Yosy	P	Bogor	00000	1982-06-09	2225000
26	Banowati	P	Malang	40123	1978-11-12	2350000
24	Miranda	P	Bogor	12345	1980-07-07	2400000
23	Yuliawati	P	Bogor	00000	1974-06-09	2300000
22	Dadang	L	Surabaya	17211	1977-08-10	2200000
20	Ratu	P	Yogyakarta	40123	1972-11-12	1950000
17	Yono	P	Bogor	00000	1989-06-09	1900000
16	Marpaung	L	Surabaya	17211	1988-08-10	1800000
14	Wawan	P	Semarang	40123	1971-11-12	1600000
11	Yenny	P	Bogor	00000	1985-06-09	1150000
6	Mawar	P	Bogor	12345	1985-07-07	2250000
42	Miranti	P	Medan	12345	1975-07-07	1375000

18 rows in set (0.00 sec)

Hm, masih belum tepat juga. Kita coba lagi:

```
mysql> select * from karyawan
-> where kota <> "bandung"
-> and kota <> "Jakarta"
-> and kota <> "Bekasi"
-> order by kota, nama ;
```

noid	nama	jenkel	kota	kodepos	tgllahir	gaji
36	Mardiatun	P	Bogor	12345	1975-07-07	1625000
6	Mawar	P	Bogor	12345	1985-07-07	2250000
24	Miranda	P	Bogor	12345	1980-07-07	2400000
11	Yenny	P	Bogor	00000	1985-06-09	1150000
17	Yono	P	Bogor	00000	1989-06-09	1900000
29	Yossy	P	Bogor	00000	1982-06-09	2225000
5	Yuliawati	P	Bogor	00000	1982-06-09	2000000
23	Yuliawati	P	Bogor	00000	1974-06-09	2300000
26	Banowati	P	Malang	40123	1978-11-12	2350000
30	Melia	P	Malang	12345	1981-07-07	2325000
42	Miranti	P	Medan	12345	1975-07-07	1375000
35	Nenny	P	Medan	00000	1972-06-09	1425000
38	Siti	P	Medan	40123	1988-11-12	1825000
31	Anwar	L	Purwakarta	41011	1972-10-02	2425000
14	Wawan	P	Semarang	40123	1971-11-12	1600000
22	Dadang	L	Surabaya	17211	1977-08-10	2200000
16	Marpaung	L	Surabaya	17211	1988-08-10	1800000
20	Ratu	P	Yogyakarta	40123	1972-11-12	1950000

18 rows in set (0.00 sec)

Nah, ternyata sekarang baru berhasil. Coba sekali lagi perhatikan permintaannya: *"tampilkan karyawan dengan kota kelahiran **bukan** di **Bandung, Jakarta dan Bekasi**. Tampilan data diurut berdasarkan nama kota **dan** juga nama karyawan."* Walaupun ada kata "**dan**" di sini, tetapi tidak semata-mata kita bisa menggunakan operator logika **AND**. Memang diperlukan kejelian dan coba-coba dalam permainan logika ini.

4.2 Fungsi Statistik Dasar

Sekarang siapa saja yang gajinya diantara Rp 1.500.000 dan Rp 2.500.000? Tampilan data diurut berdasarkan kolom gaji dan nama karyawan

```
mysql> select * from karyawan
-> where gaji >= 1500000
-> and gaji <= 2500000
-> order by gaji, nama ;
```

```
+-----+-----+-----+-----+-----+-----+-----+
|noid|nama      |jenkel|kota      |kodepos|tgllahir |gaji  |
+-----+-----+-----+-----+-----+-----+-----+
| 3|Ryan Cakep|L      |Jakarta   |12111  |1981-03-21|1500000|
| 14|Wawan     |P      |Semarang  |40123  |1971-11-12|1600000|
| 36|Mardiatun |P      |Bogor     |12345  |1975-07-07|1625000|
| 18|Dian      |P      |Jakarta   |12345  |1980-07-07|1650000|
| 15|The Cute  |L      |Jakarta   |12111  |1977-03-21|1700000|
| 37|Andika    |L      |Bandung   |41011  |1978-10-02|1725000|
| 4|Zukarman  |L      |Bekasi    |17211  |1978-08-10|1750000|
| 16|Marpaung  |L      |Surabaya  |17211  |1988-08-10|1800000|
| 38|Siti      |P      |Medan     |40123  |1988-11-12|1825000|
| 19|Donno     |L      |Bandung   |41011  |1971-10-02|1850000|
| 17|Yono      |P      |Bogor     |00000  |1989-06-09|1900000|
| 39|Rohimat   |L      |Jakarta   |12111  |1980-03-21|1925000|
| 20|Ratu       |P      |Yogyakarta|40123  |1972-11-12|1950000|
| 5|Yuliawati |P      |Bogor     |00000  |1982-06-09|2000000|
| 21|Bambang   |L      |Jakarta   |12111  |1982-03-21|2100000|
| 28|Gunadi    |L      |Bekasi    |17211  |1978-08-10|2125000|
| 25|Subur     |L      |Bandung   |41011  |1977-10-02|2150000|
| 22|Dadang    |L      |Surabaya  |17211  |1977-08-10|2200000|
| 29|Yossy     |P      |Bogor     |00000  |1982-06-09|2225000|
| 6|Mawar     |P      |Bogor     |12345  |1985-07-07|2250000|
| 23|Yuliawati |P      |Bogor     |00000  |1974-06-09|2300000|
| 30|Melia     |P      |Malang    |12345  |1981-07-07|2325000|
| 26|Banowati  |P      |Malang    |40123  |1978-11-12|2350000|
| 24|Miranda   |P      |Bogor     |12345  |1980-07-07|2400000|
| 31|Anwar     |L      |Purwakarta|41011  |1972-10-02|2425000|
| 27|Gungun    |L      |Jakarta   |12111  |1981-03-21|2450000|
+-----+-----+-----+-----+-----+-----+-----+
26 rows in set (0.00 sec)
```

Sekarang berapa orang karyawan yang gajinya di bawah Rp 2.000.000?

```
mysql> select count(*) from karyawan
-> where gaji < 2000000 ;
+-----+
| count(*) |
+-----+
|      29 |
+-----+
1 row in set (0.01 sec)
```

Berapakah gaji rata-rata karyawan?

```
mysql> select avg(gaji) from karyawan ;
+-----+
| avg(gaji) |
+-----+
| 1719642.8571 |
+-----+
1 row in set (0.41 sec)
```

Berapakah nilai gaji yang terbesar?

```
mysql> select max(gaji) from karyawan ;
+-----+
| max(gaji) |
+-----+
| 2450000 |
+-----+
1 row in set (0.00 sec)
```

Dan berapakah jumlah gaji seluruh karyawan?

```
mysql> select sum(gaji) from karyawan ;
+-----+
| sum(gaji) |
+-----+
| 72225000 |
+-----+
1 row in set (0.00 sec)
```

Soal Latihan

1. Buka tabel **peserta**.
 - Tambahkan field jnsKursus varchar (30) Not Null dan field Biaya INT(12) NOT NULL default 0
 - Isikan jnsKursus dan Biaya pada masing-masing record
2. Tampilkan seluruh data peserta dimana tanggal lahirnya sebelum 01 Januari 1985urut berdasarkan nama
3. Tampilkan seluruh data peserta yang berasal dari kota Solo dan jenis kursus yang diambil adalah Aplikasi Perkantoran.
4. Tampilkan data peserta yang mengambil jenis kursus Aplikasi Perkantoran atau Multimedia dan berasal dari kota Solo
5. Tampilkan berapa jumlah peserta yang mengambil kursus aplikasi perkantoran
6. Tampilkan total pendapatan yang diterima dari biaya kursus

Daftar Pustaka

<http://dev.mysql.com>

[http://www.rohmat-mimi.com/download/MODUL PRAKTIKUM MY SQL-BASIS DATA](http://www.rohmat-mimi.com/download/MODUL_PRAKTIKUM_MY_SQL-BASIS_DATA)

<http://www.mysql.com>

<http://www.arbiedesign.com/index.php>

Tim Training SMK-TI.Modul MySQL

BAB V

Operator Precedence, LIKE, NOT LIKE, REGEXP

Standar Kompetensi :

1. Mahasiswa dapat mengetahui, memahami, menguasai dan mampu mengimplementasi teori, konsep dan prinsip pemrograman database MySQL dengan logika pemrograman yang benar, ringkas, dan tepat dalam penerapannya di bidang teknologi informasi

Kompetensi Dasar	Indikator
1.12 Mengenal Operator Precedence	<ul style="list-style-type: none"> Menggunakan operator precedence sebagai tingkatan hirarki dalam memproses serangkaian operator
1.13 Mengenal Operator LIKE, NOT LIKE, dan REGEXP	<ul style="list-style-type: none"> Menerapkan operator LIKE, NOT LIKE dan REGEXP ke dalam bahasa SQL

Materi

5.1 Operator Precedence

Operator precedence adalah tingkatan hirarki dalam memproses serangkaian operator.

Tabel 4.3 Operator Precedence

Tingkatan hirarki	Jenis Operator
Paling Tinggi	BINARY
	NOT !
	- (unary minus)
	* / %
	+ -
	<< >>
	&
	< <= > >= != <> IN IS LIKE REGEXP RLIKE
	BETWEEN
	AND &&
Paling Rendah	OR

Semakin keatas posisi operator, maka semakin tinggi tingkat hirarki operator tersebut. Begitu pula sebaliknya, semakin rendah posisinya maka akan semakin lemah hirarkinya. Untuk operator yang sama kuat, misal + dan - digabung dengan operator * / %, maka akan ditentukan hirarkinya tergantung dari posisi mana yang paling kiri/paling awal ditemukan. Dan untungnya posisi hirarki ini dapat diubah dengan bantuan tanda kurung "(" dan ")". Sekarang kita lihat penerapannya.

```
mysql> select 10+15-11*2, (10+15-11)*2,
             -> 2*6-5, 2*(6-5) ;
+-----+-----+-----+-----+
| 10+15-11*2 | (10+15-11)*2 | 2*6-5 | 2*(6-5) |
+-----+-----+-----+-----+
|          3 |             28 |       7 |        2 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Anda bisa perhatikan bahwa walaupun angka dan operatornya sama, tapi hasilnya bisa berbeda. Dan itu karena adanya peranan dari tanda kurung "(" dan ")" yang akan mengubah peta posisi hirarki operator.

5.2 Operator LIKE, NOT LIKE, REGEXP

Operator **LIKE**, **NOT LIKE**, **REGEXP** akan banyak kita gunakan terutama dalam operasi karakter.

A. Operator LIKE

Sekarang kita akan coba menggunakan operator **LIKE**. Operator **LIKE** ini digunakan untuk mencari data yang "**menyerupai**" atau "**hampir sama**" dengan kriteria tertentu. Biasanya untuk mencari data string/teks. Simbol "%" digunakan untuk membantu pelaksanaan operator **LIKE**. Posisi "%" sangat berpengaruh dalam menentukan kriteria. Kita langsung dengan contoh-contohnya saja ya biar lebih jelas penggunaannya. Tampilkan data karyawan yang namanya **berawalan** huruf "**a**": (perhatikan posisi simbol persennya "%")

```
mysql> select noid, nama
-> from karyawan
-> where nama LIKE "a%" ;
```

noid	nama
1	Ahmad Zobari
31	Anwar
37	Andika

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

Tampilkan data karyawan yang namanya **berawalan** huruf "d":

```
mysql> select noid, nama
-> from karyawan
-> where nama LIKE "d%" ;
```

noid	nama
13	Dadan
18	Dian
19	Donno
22	Dadang

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

Tampilkan data karyawan yang namanya **berakhiran** huruf "i". Perhatikan posisi penulisan tanda "%":

```
mysql> select noid, nama
-> from karyawan
-> where nama LIKE "%i" ;
```

noid	nama
1	Ahmad Zobari
2	Sundariwati
5	Yuliawati
7	Zobari
23	Yuliawati
26	Banowati
28	Gunadi
32	Susilowati
38	Siti
41	Yanti
42	Miranti

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


Tampilkan data karyawan yang namanya berakhiran "**wati**":

```
mysql> select noid, nama
      -> from karyawan
      -> where nama LIKE "%wati" ;
```

noid	nama
2	Sundariwati
5	Yuliawati
23	Yuliawati
26	Banowati
32	Susilowati

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

Bagaimana caranya agar operator LIKE dapat membedakan huruf besar dan kecil...
Sederhana saja, cukup dengan menambahkan kata **BINARY** saja setelah perintah LIKE
(sehingga perintahnya menjadi **LIKE BINARY**). Kita lihat contohnya...

```
mysql> select noid, nama
      -> from karyawan
      -> where nama LIKE BINARY "a%" ;
```

Empty set (0.34 sec)

Kenapa hasilnya menjadi "**Empty set**"? Kita coba dengan mengubah perintah tadi menjadi:

```
mysql> select noid, nama
      -> from karyawan
      -> where nama LIKE BINARY "A%" ;
```

noid	nama
1	Ahmad Zobari
31	Anwar
37	Andika

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

Ya dengan menggunakan LIKE BINARY, penulisan huruf "a" akan dibedakan artinya dengan "A". Jelas kan?

Sekarang bagaimana kalau kita ingin menampilkan data, dengan kriteria bukan diawal atau diakhir kalimat, tapi berada **diantara** sebuah kata/kalimat? Misal ada berapa nama karyawan yang memiliki kata "**lia**"? . Perhatikan posisi penulisan tanda "%":.

```
mysql> select noid, nama
      -> from karyawan
      -> where nama LIKE BINARY "%lia%" ;
+-----+-----+
| noid | nama      |
+-----+-----+
| 5    | Yuliawati |
| 8    | Melia     |
| 23   | Yuliawati |
| 30   | Melia     |
+-----+-----+
4 rows in set (0.00 sec)
```

Atau memiliki kata "**Di**" pada namanya?

```
mysql> select noid, nama
      -> from karyawan
      -> where nama LIKE BINARY "%Di%" ;
+-----+-----+
| noid | nama |
+-----+-----+
| 18   | Dian |
+-----+-----+
1 row in set (0.00 sec)
```

B. Operator REGEXP

Operator **REGEXP** (singkatan dari **REG**ular **EXP**ressions) merupakan bentuk lain dari operator LIKE, dengan fungsi yang lebih disempurnakan. Operator REGEXP biasanya ditemani juga dengan simbol-simbol tertentu dalam melaksanakan tugasnya, seperti:

Tabel 5.1 Simbol Operator REGEXP

Simbol	Keterangan
.	Satu tanda titik (.) untuk mewakili satu karakter
[?]	Untuk mewakili beberapa karakter atau range yang ditentukan.
^	Untuk posisi awal dari sebuah kriteria yang ditentukan
\$	Untuk posisi akhir dari sebuah kriteria yang ditentukan

Kita langsung saja pada contohnya. Tampilkan nama karyawan yang **berawalan** huruf 'a':

```
mysql> select noid, nama
      -> from karyawan
      -> where nama REGEXP "^a" ;
```

noid	nama
1	Ahmad Zobari
31	Anwar
37	Andika

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

Tampilkan data karyawan yang namanya **berawalan** huruf "d".

```
mysql> select noid, nama
      -> from karyawan
      -> where nama REGEXP "^d" ;
```

noid	nama
13	Dadan
18	Dian
19	Donno
22	Dadang

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

Tampilkan nama karyawan yang **berawalan** huruf 'a' sampai dengan huruf 'd':

```
mysql> select noid, nama
      -> from karyawan
      -> where nama REGEXP "[a-d]"
      -> order by nama ;
```

noid	nama
1	Ahmad Zobari
37	Andika
31	Anwar
21	Bambang
26	Banowati
40	Beno
13	Dadan
22	Dadang
18	Dian
19	Donno

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

Tampilkan data karyawan yang namanya **berakhiran huruf "i"**:

```
mysql> select noid, nama
-> from karyawan
-> where nama REGEXP "i$"
-> order by nama ;
```

```
+-----+-----+
| noid | nama          |
+-----+-----+
| 1    | Ahmad Zobari  |
| 26   | Banowati      |
| 28   | Gunadi        |
| 42   | Miranti       |
| 38   | Siti          |
| 7    | Zobari        |
| 2    | Sundariwati   |
| 32   | Susilowati    |
| 41   | Yanti         |
| 5    | Yuliawati     |
| 23   | Yuliawati     |
+-----+-----+
11 rows in set (0.00 sec)
```

Tampilkan data karyawan yang namanya **berakhiran "wati"**:

```
mysql> select noid, nama
-> from karyawan
-> where nama REGEXP "wati$"
-> order by nama ;
```

```
+-----+-----+
| noid | nama          |
+-----+-----+
| 26   | Banowati      |
| 2    | Sundariwati   |
| 32   | Susilowati    |
| 5    | Yuliawati     |
| 23   | Yuliawati     |
+-----+-----+
5 rows in set (0.00 sec)
```

Tampilkan nama karyawan yang **panjangnya 10 karakter**:

```
mysql> select noid, nama
-> from karyawan
-> where nama REGEXP "^.....$" ;
```

```
+-----+-----+
| noid | nama          |
+-----+-----+
| 3    | Ryan Cakep    |
| 9    | Zanda Cute    |
| 32   | Susilowati    |
+-----+-----+
2 rows in set (0.00 sec)
```

Atau perintah di atas bisa juga ditulis dengan:

```
mysql> select noid, nama
      -> from karyawan
      -> where nama REGEXP "^.{10}$" ;
```

noid	nama
3	Ryan Cakep
9	Zanda Cute
32	Susilowati

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

Nah, kurang lebih itulah dasar-dasar MySQL. Ini baru tutorial pengenalan. Kita akan bertemu lagi dengan tutorial berikutnya, **dasar-dasar database relasi**. Kemudian dilanjutkan dengan tutorial penerapan sederhana "**database relasi dengan menggunakan 2 buah tabel**".

Soal Latihan

1. Buka tabel **peserta**.
2. Tampilkan nama peserta yang diawali dengan huruf A
3. Tampilkan nama peserta yang diawali dengan huruf A dan diakhiri dengan huruf i
4. Tampilkan nama peserta yang berawalan d – g dan panjang karakternya 10

Daftar Pustaka

<http://dev.mysql.com>

[http://www.rohmat-mimi.com/download/MODUL PRAKTIKUM MY SQL-BASIS DATA](http://www.rohmat-mimi.com/download/MODUL_PRAKTIKUM_MY_SQL-BASIS_DATA)

<http://www.mysql.com>

<http://www.arbiedesign.com/index.php>

Tim Training SMK-TI.Modul MySQL

BAB VI

Database Relasi

Standar Kompetensi :

1. Mahasiswa dapat mengetahui, memahami, menguasai dan mampu mengimplementasi teori, konsep dan prinsip pemrograman database MySQL dengan logika pemrograman yang benar, ringkas, dan tepat dalam penerapannya di bidang teknologi informasi

Kompetensi Dasar			Indikator
2.1	Mengenal	Model Database	▪ Menyebutkan dan menjelaskan macam-macam model database
2.2	Mengenal	Database Relasi	▪ Menjelaskan definisi database relasi ▪

Materi

6.1 Model Database

Model database adalah suatu konsep yang terintegrasi dalam menggambarkan hubungan (relationships) antar data dan batasan-batasan (constraint) data dalam suatu sistem database. Model data yang paling umum, berdasarkan pada bagaimana hubungan antar record dalam database (Record Based Data Models), terdapat tiga jenis, yaitu :

- a. Model Database Hirarki (Hierarchical Database Model)
- b. Model Database Jaringan (Network Database Model)
- c. Model Database Relasi (Relational Database Model)

Model database hirarki dan jaringan merupakan model database yang tidak banyak lagi dipakai saat ini, karena adanya berbagai kelemahan dan hanya cocok untuk struktur hirarki dan jaringan saja. Artinya tidak mengakomodir untuk berbagai macam jenis persoalan dalam suatu sistem database. Yang paling banyak dipakai saat ini

adalah model database relasi, karena mampu mengakomodir berbagai permasalahan dalam sistem database. Berikut keterangan tentang model database ini.

6.2 Model Database Relasi

Model database relasi merupakan model database yang paling banyak digunakan saat ini, karena paling sederhana dan mudah digunakan serta yang paling penting adalah kemampuannya dalam mengakomodasi berbagai kebutuhan pengelolaan database. Sebuah database dalam model ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (record) dan kolom (field), pertemuan antara baris dengan kolom disebut item data (data value), tabel-tabel yang ada di hubungkan (relationship) sedemikian rupa menggunakan field-field kunci (Key field) sehingga dapat meminimalkan duplikasi data.

Model database relasi ini dikemukakan pertama kali oleh E.F. Codd, salah seorang pakar dalam bidang database. Sering juga model ini disebut Database relasi.

6.3 Tingkatan Data Dalam Database Relasi

Dalam suatu sistem database relasi, data yang tersimpan dalam DBMS mempunyai tingkatan-tingkatan, sebagai berikut :

A. Karakter (Characters)

Merupakan bagian terkecil dalam database, dapat berupa karakter numerik (angka 0 s.d 9), huruf (A - Z, a - z) ataupun karakter-karakter khusus, seperti *, &. %, # dan lain-lain.

B. Field atau Attribute

Merupakan bagian dari record yang menunjukkan suatu item data yang sejenis, Misalnya : field nama, file NIM dan lain sebagainya. Setiap field harus mempunyai nama dan tipe data tertentu. Isi dari field di sebut Data Value. Dalam tabel database, field ini disebut juga kolom.

C. Record atau Tupple

Tuple/Record adalah kumpulan data value dari attribute yang berkaitan sehingga dapat menjelaskan sebuah entity secara lengkap. Misal : Record entity mahasiswa adalah kumpulan data value dari field nobp, nama, jurusan dan alamat per-barisnya. Dalam tabel database, Record disebut juga baris.

D. Table/Entity

Entity merupakan sesuatu yang dapat diidentifikasi dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian atau konsep yang informasinya akan disimpan di database. Misal pada sistem database akademik, yang menjadi entity adalah, mahasiswa, dosen, matakuliah dan lain-lain. Dalam aplikasi nantinya, penggunaan istilah Entity sering di samakan dengan istilah Tabel. (Entity = table). Disebut tabel, karena dalam merepresentasikan datanya di atur dalam bentuk baris dan kolom. Baris mewakili 1 record dan kolom mewakili 1 field. Dalam sistem database tradisional, entity/table ini disebut juga dengan file.

E. Database

Kumpulan dari tabel-tabel yang saling berelasi, disusun secara logis, sehingga menghasilkan informasi yang bernilai guna dalam proses pengambilan keputusan

6.4 Sifat Yang Melekat Pada Suatu Tabel

- ✚ Tidak boleh ada record yang sama (kembar)
- ✚ Urutan record tidak terlalu penting, karena data dalam record dapat diurut sesuai dengan kebutuhan.
- ✚ Setiap field harus mempunyai nama yang unik (tidak boleh ada yang sama).
- ✚ Setiap field mesti mempunyai tipe data dan karakteristik tertentu

6.5 Jenis Hubungan Antar Tabel

Jenis hubungan antar tabel dalam model database relasi, juga didefinisikan dengan:

- ✚ Satu ke satu (One to One)

- ✚ Satu ke Banyak (One to Many)
- ✚ Banyak ke satu (Many to One)
- ✚ Banyak ke Banyak (Many to Many)

A. Satu Ke Satu (One to One)

Relasi tabel one-to-one (satu-satu) adalah relasi 2 tabel dengan primary key (PK) dan foreign key (FK). Ini dilakukan dengan meletakkan kolom one-to-one ke tabel baru. Sebetulnya relasi ini jarang digunakan. Namun ada beberapa alasan relasi ini digunakan:

1. Memindahkan data ke tabel lain memungkinkan untuk membuat query yang lebih cepat.
2. Mengisolasi dan menghindarkan nilai NULL pada tabel utama.
3. Membuat sebagian data susah diakses

Contoh tabel:

Tbl_Pegawai

- id_pegawai (PK)
- nama_depan
- nama_belakang

Tbl_Gaji

- kode_asuransi (PK)
- level_gaji
- id_pegawai (FK)

Tabel pegawai dan gaji didesain one-to-one untuk mendapatkan keuntungan:

1. Query untuk mendapatkan kode_asuransi lebih cepat
2. Data gaji lebih aman

B. Satu Ke Banyak (One to Many)

Artinya satu record pada entity A ber-relasi dengan beberapa record pada entity B, tapi tidak sebaliknya, setiap record pada entity B ber-relasi paling banyak satu record dengan entity A. Dalam diagram E-R, relasi ini disimbolkan dengan angka 1 untuk menyatakan satu dan huruf M atau N untuk menyatakan banyak.

Contoh : Dalam proses belajar mengajar di sekolah dasar misalnya, satu orang guru mengajar beberapa (banyak) murid, tetapi satu kelas (beberapa murid) hanya diajar oleh satu guru.

C. Banyak Ke Banyak (Many to Many)

Artinya beberapa record pada entity A dapat ber-relasi dengan beberapa record juga pada entity B, begitu juga sebaliknya, beberapa record pada entity B dapat ber-relasi dengan beberapa record juga pada entity A. Dalam diagram E-R, relasi ini disimbolkan dengan huruf M atau N untuk menyatakan banyak.

Contoh : Dalam hubungan antara mahasiswa dengan dosen pada perguruan tinggi, yaitu seorang dosen mengajar banyak mahasiswa, sebaliknya seorang mahasiswa dapat diajar oleh beberapa dosen, sehingga terjadi hubungan banyak ke banyak.

6.6 Relasi Database dengan MySQL

MySQL mempunyai kemampuan untuk menggabungkan dua tabel atau lebih guna mendapatkan informasi yang diinginkan. Proses yang dilakukan dengan nama JOIN. Dalam penggabungan dari beberapa tabel (join) ada beberapa hal yang perlu diperhatikan, antara lain :

1. Setiap kolom disebutkan dengan bentuk ,
2. Tabel-tabel yang dilibatkan dalam query perlu disebutkan dalam Klausa FORM dengan antar tabel dipisah oleh koma.
3. Kondisi dalam WHERE menentukan macam join yang terbentuk

Sebelum kita lanjut membahas tentang relasi database di MySQL, kita akan membuat database baru ...(supaya Anda tidak bosan dengan database latihan1 pada bab sebelumnya ☺). Okey...mari kita buat database baru .

Misalnya : Anda diminta untuk membuat sistem penjualan barang oleh *marketing freelance* di perusahaan Oryn Textile. Barang berupa Kaos Kaki @ Rp 3500

1. Kita buat database, dbOryn (masih ingat kan gimana buatnya???)
2. Kita buat dua tabel sbb :

a. Tabel tbMarketing (untuk mencatat data marketing)

b. Tabel tbJual (untuk mencatat penjualan barang)

3. Berikut struktur tabelnya :

Tabel tbMarketing

Field	Type	Null	Key	Default	Extra
NoID	char(4)	NO	PRI	NULL	
Nama	varchar(20)	NO		NULL	
NoHP	varchar(12)	NO		NULL	
Alamat	varchar(20)	NO		NULL	

Tabel tbJual

Field	Type	Null	Key	Default	Extra
NoJual	char(4)	NO	PRI	NULL	
NoID	char(4)	NO		NULL	
TglJual	date	NO		NULL	
Quantity	smallint(3)	NO		NULL	

4. Isikan data pada tabel seperti berikut :

Tabel tbMarketing

NoID	Nama	NoHP	Alamat
M001	Dwi Apri Setyorini	081888777666	Gumpang Kartasura
M002	Pipin Widyaningsih	081555666777	Gebyok Kartasura
M003	Haryanto	081222444666	Gedongan Solo
M004	Wijiyanto	081393837360	Tipes Solo
M005	Sri Sumarlinda	081999555222	Colomadu Solo

Tabel tbJual

NoJual	NoID	TglJual	Quantity
J001	M001	2010-01-13	25
J002	M002	2010-01-13	20
J003	M004	2010-01-13	30
J004	M006	2010-01-13	15

Jika kita perhatikan di tabel tbJual, tidak menginformasikan nama Marketing yang melakukan penjualan. Mari kita coba menuliskan perintah SQL untuk menampilkan NoJual, NoID, Nama, TglJual dan Quantity, tentunya dengan menggunakan relasi. Berikut perintah SQL yang digunakan ;

```
mysql> SELECT NoJual, NoID, Nama, TglJual, Quantity
-> From tbMarketing, tbJual
-> Where tbMarketing.NoID=tbJual.NoID;
ERROR 1052 (23000): Column 'NoID' in field list is ambiguous
```

Perhatikan hasil di atas, terdapat ERROR 1052..dst. Kenapa terjadi seperti itu? Disebutkan bahwa terdapat field NoID ambiguous. Gimana nih....? Okey, kita bahas pelan-pelan...

Field NoID dimiliki oleh dua tabel yaitu tbMarketing dan tbJual, sehingga terdapat data ambigu karena kita tidak menjelaskan field NoID tersebut berasal dari tabel yang mana.

Solusinya :....

```
mysql> SELECT NoJual, tbMarketing.NoID, Nama, TglJual, Quantity
-> From tbMarketing, tbJual
-> Where tbMarketing.NoID=tbJual.NoID;
+-----+-----+-----+-----+-----+
| NoJual | NoID | Nama                | TglJual   | Quantity |
+-----+-----+-----+-----+-----+
| J001   | M001 | Dwi Apri Setyorini  | 2010-01-13 | 25       |
| J002   | M002 | Pipin Widyaningsih  | 2010-01-13 | 20       |
| J003   | M004 | Wijiyanto           | 2010-01-13 | 30       |
+-----+-----+-----+-----+-----+
3 rows in set (0.08 sec)
```

Perhatikan hasil/tabel di atas pada judul kolom Nama. Jika Anda menginginkan agar pada saat hasil ditampilkan judul kolom **Nama** diperjelas dengan diganti menjadi **Nama Marketing**, maka MySQL menyediakan perintah **AS** (kepanjangan **AliaS**) untuk mengganti judul kolom pada saat ditampilkan (catatan : perintah **AS** tidak akan merubah struktul tabel, jadi nama field pada tbMarketing tetap **Nama**)

```
mysql> SELECT NoJual, tbMarketing.NoID, Nama AS Nama_Marketing,
-> TglJual, Quantity
-> From tbMarketing, tbJual
-> Where tbMarketing.NoID=tbJual.NoID;
```

NoJual	NoID	Nama_Marketing	TglJual	Quantity
J001	M001	Dwi Apri Setyorini	2010-01-13	25
J002	M002	Pipin Widyaningsih	2010-01-13	20
J003	M004	Wijiyanto	2010-01-13	30

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

Dari data tersebut, dapat kita lihat bahwa pada tabel tbMarketing, tidak terdapat record dengan NoID M006 , dan begitu pula pada tabel tbJual tidak terdapat record dengan NoID M003 dan M005. Apabila dilakukan join seperti perintah di atas maka beberapa record tidak akan tampil seperti yang kita mau.

```
mysql> SELECT NoJual, tbMarketing.NoID, Nama, TglJual, Quantity
-> From tbMarketing, tbJual
-> Where tbMarketing.NoID=tbJual.NoID;
```

Hanya akan menampilkan :

NoJual	NoID	Nama	TglJual	Quantity
J001	M001	Dwi Apri Setyorini	2010-01-13	25
J002	M002	Pipin Widyaningsih	2010-01-13	20
J003	M004	Wijiyanto	2010-01-13	30

```
3 rows in set (0.08 sec)
```

Lalu dimana yang memiliki No ID M003 dan M005 ?, untuk itulah ada beberapa join khusus. Kondisi Where menentukan macam join yang terbentukMacam-macam bentuk Penggabungan (Join)

1. CROSS JOIN, Cross Join merupakan bentuk penggabungan yang paling sederhana, tanpa ada kondisi.

Bentuk Umum : SELECT field1,field2 FROM Tabel1 CROSS JOIN tabel2;

2. INNER JOIN

Hampir sama dengan cross join tetapi diikuti dengan kondisi

Bentuk Umum : SELECT Field FROM tabel1 INNER JOIN tabel 2 ON kondisi

3. STRAIGHT JOIN

Straight Join identik dengan inner join tetapi tidak mengenal klausa where

Bentuk Umum : SELECT field FROM Tabel1 SATRIGHT JOIN tabel2

4. LEFT (OUTER) JOIN

Akan menampilkan tabel disebelah kanannya dengan NULL jika tidak terdapat hubungan antara tabel disebelah kiri.

Bentuk Umum : SELECT field FROM tabel1 LEFT JOIN tabel2 ON kondisi

5. RIGHT (OUTER) JOIN

Kebalikan dari LEFT JOIN

Bentuk Umum : SELECT field FROM tabel1 RIGHT JOIN tabel2 ON kondisi

Contoh :

Penggunaan LEFT JOIN

```
mysql> SELECT * FROM tbMarketing LEFT JOIN tbJual
-> ON tbMarketing.NoID=tbJual.NoID;
```

NoID	Nama	NoHP	Alamat	NoJual	NoID	TglJual	Quantity
M001	Dwi Apri Setyorini	081888777666	Gumpang Kartasura	J001	M001	2010-01-13	25
M002	Pipin Widyaningsih	081555666777	Gebyok Kartasura	J002	M002	2010-01-13	20
M003	Haryanto	081222444666	Gedongan Solo	NULL	NULL	NULL	NULL
M004	Wijiyanto	081393837360	Tipes Solo	J003	M004	2010-01-13	30
M005	Sri Sumarlinda	081999555222	Colomadu Solo	NULL	NULL	NULL	NULL

Penggunaan RIGHT JOIN

```
mysql> SELECT * FROM tbMarketing RIGHT JOIN tbJual
-> ON tbMarketing.NoID=tbJual.NoID;
```

NoID	Nama	NoHP	Alamat	NoJual	NoID	TglJual	Quantity
M001	Dwi Apri Setyorini	081888777666	Gumpang Kartasura	J001	M001	2010-01-13	25
M002	Pipin Widyaningsih	081555666777	Gebyok Kartasura	J002	M002	2010-01-13	20
M004	Wijiyanto	081393837360	Tipes Solo	J003	M004	2010-01-13	30
NULL	NULL	NULL	NULL	J004	M006	2010-01-13	15

Note: penggunaan SELECT * pada LEFT atau RIGHT join akan menampilkan column yang redundansi. jadi?? kita lanjut...kan

mari kita bandingkan dengan penggunaan NATURAL JOIN

```
mysql> SELECT * FROM tbMarketing NATURAL JOIN tbJual;
```

NoID	Nama	NoHP	Alamat	NoJual	TglJual	Quantity
M001	Dwi Apri Setyorini	081888777666	Gumpang Kartasura	J001	2010-01-13	25
M002	Pipin Widyaningsih	081555666777	Gebyok Kartasura	J002	2010-01-13	20
M004	Wijiyanto	081393837360	Tipes Solo	J003	2010-01-13	30

Ternyata sama dengan query yang pertama. Jadi bisa dikatakan bahwa ini penyederhanaan query pertama, namun menghapus column yang redundansi.

Soal Latihan

1. Buka database dbKursus
2. Tambahkan :
 - a. Tabel Tutor untuk menyimpan data Tutor yang mengajar
 - b. Tabel Jenis untuk menyimpan jenis kursus yang ditawarkan
 - c. Tabel Transaksi untuk menyimpan transaksi pembayaran kursus
 - d. Tabel Jadwal untuk menyimpan jadwal Kursus
3. Tampilkan Data peserta beserta Jenis kursus yang diambil
4. Tampilkan Data Tutor beserta Jenis kursus yang diampu
5. Tampilkan jenis kursus apa saja yang ditawarkan pada hari Senin
6. Tampilkan total pendapatan yang diterima untuk jenis kursus Aplikasi Perkantoran selama bulan April 2009.

Daftar Pustaka

<http://dev.mysql.com>

[http://www.rohmat-mimi.com/download/MODUL PRAKTIKUM MY SQL-BASIS DATA](http://www.rohmat-mimi.com/download/MODUL_PRAKTIKUM_MY_SQL-BASIS_DATA)

<http://www.mysql.com>

<http://www.arbiedesign.com/index.php>

Tim Training SMK-TI.Modul MySQL