BUKU KOMPUTER GRATIS DARI ACHMATIM.NET

# MySQL 5

## Dari Pemula Hingga Mahir

Achmad Solichin


ACHMAD SOLICHIN, HTTP://ACHMATIM.NET
UNIVERSITAS BUDI LUHUR, JAKARTA

***…masih ku persembahkan untuk Indonesia tanah air tercinta, dan untuk Chotimatul Musyarofah istri tersayang serta Muhammad Lintang putra tercinta kami…***

# Lisensi Dokumen

# Kata Pengantar

**MySQL** merupakan *software database open source* yang paling populer di dunia, dimana saat ini digunakan lebih dari 100 juta pengguna di seluruh dunia. Dengan kehandalan, kecepatan dan kemudahan penggunaannya, MySQL menjadi pilihan utama bagi banyak pengembang *software* dan aplikasi baik di *platform* web maupun *desktop*. Pengguna MySQL tidak hanya sebatas pengguna perseorangan maupun perusahaan kecil, namun perusahaan seperti Yahoo!, Alcatel-Lucent, Google, Nokia, Youtube, Wordpress dan Facebook juga merupakan pengguna MySQL.

MySQL pertama kali dibuat dan dikembangkan di Swedia, yaitu oleh David Axmark, Allan Larsson dan Michael "Monty" Widenius. Mereka mengembangkan MySQL sejak tahun 1980-an. Saat ini versi MySQL yang sudah stabil mencapai versi 5x, dan sedang dikembangkan versi 6x. Untuk lebih lengkapnya dapat dilihat di situs resmi MySQL[1].

Buku berjudul "**MySQL 5: Dari Pemula Hingga Mahir**" ini mencoba membahas MySQL secara praktis, disajikan secara terstruktur dan disertai contoh-contoh dan latihan untuk membantu pemahaman. Buku ini diharapkan dapat membantu Anda menguasai MySQL hingga mahir. Buku ini sangat cocok bagi Anda yang baru mempelajari MySQL maupun bagi Anda yang ingin lebih memperdalam MySQL sebagai salah satu *software* database terkemuka saat ini.

Buku ini terbagi menjadi 4 (empat) bagian. Bagian pertama merupakan bagian pendahuluan yang membahas mengenai penjelasan singkat MySQL dan juga langkah instalasi MySQL serta *software* pendukung lainnya. Bagian kedua adalah Dasar-dasar MySQL yang menjelaskan mengenai perintah-perintah dasar dari MySQL termasuk fungsi-fungsi di dalam MySQL. Pada bagian ketiga dipaparkan mengenai perintah-perintah MySQL yang lebih kompleks seperti penggabungan antar tabel, *trigger, views* dan *stored procedure*. Selanjutnya pada bagian yang terakhir akan dijelaskan mengenai penyajian laporan dan proses *backup, restore* database MySQL.

[1] http://www.mysql.com

Akhirnya penulis berharap agar buku ini bermanfaat bagi perkembangan ilmu dan pengetahuan di Indonesia, khususnya dalam hal pengetahuan database MySQL. Saran dan kritik untuk perbaikan buku ini sangat penulis harapkan. Saran, kritik dan masukan mengenai buku ini dapat disampaikan melalui email ke penulis di achmatim@gmail.com atau melalui situs penulis di http://achmatim.net.

Penulis

Achmad Solichin

# DAFTAR ISI

# Bagian 1
# Pendahuluan

# Bab 1
# Sekilas Tentang MySQL

- Pengenalan Database, DBMS, dan RDBMS
- Beberapa Istilah Database
- Hierarki Database
- Pengenalan Database MySQL

## Pengenalan Database, DBMS dan RDBMS

**Basis data** (atau ***database***) adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut (http://id.wikipedia.org/wiki/Database). Database digunakan untuk menyimpan informasi atau data yang terintegrasi dengan baik di dalam komputer.

Untuk mengelola *database* diperlukan suatu perangkat lunak yang disebut **DBMS** (*Database Management System*). DBMS merupakan suatu sistem perangkat lunak yang memungkinkan user (pengguna) untuk membuat, memelihara, mengontrol, dan mengakses *database* secara praktis dan efisien. Dengan DBMS, user akan lebih mudah mengontrol dan memanipulasi data yang ada.

Sedangkan **RDBMS** atau *Relationship Database Management System* merupakan salah satu jenis DBMS yang mendukung adanya *relationship* atau hubungan antar tabel. Di samping RDBMS, terdapat jenis DBMS lain, misalnya *Hierarchy* DBMS, *Object Oriented* DBMS, dsb.

Beberapa software atau perangkat lunak DBMS yang sering digunakan dalam aplikasi program antara lain :

- DB2 - http://www-306.ibm.com/software/data/db2/
- Microsoft SQL Server - http://www.microsoft.com/sql/
- Oracle - http://www.oracle.com
- Sybase - http://www.sybase.com/
- Interbase - http://www.borland.com/interbase
- Teradata - http://www.teradata.com/
- Firebird - http://www.firebirdsql.org/
- MySQL – http://www.mysql.com
- PostgreSQL - http://www.postgresql.org/

# Beberapa Istilah Database

**Table**

Sebuah tabel merupakan kumpulan data (nilai) yang diorganisasikan ke dalam baris (record) dan kolom (field). Masing-masing kolom memiliki nama yang spesifik dan unik.

**Field**

*Field* merupakan kolom dari sebuah table. *Field* memiliki ukuran *type* data tertentu yang menentukan bagaimana data nantinya tersimpan.

**Record**

Field merupakan sebuah kumpulan nilai yang saling terkait.

**Key**

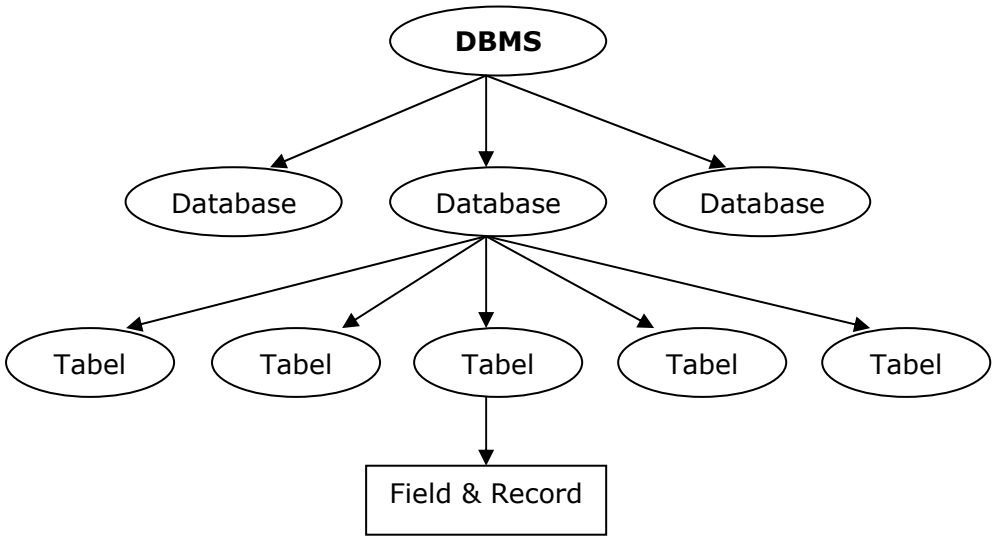
Key merupakan suatu field yang dapat dijadikan kunci dalam operasi tabel. Dalam konsep database, key memiliki banyak jenis diantaranya Primary Key, Foreign Key, Composite Key, dll.

**SQL**

SQL atau Structured Query Language merupakan suatu bahasa (*language*) yang digunakan untuk mengakses database. SQL sering disebut juga sebagai query.

# Hierarki Database

Dalam konsep database, urutan atau hierarki database sangatlah penting. Urutan atau hierarki database digambarkan dalam gambar sbb :



# MySQL

**MySQL** adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak seperti PHP atau Apache yang merupakan software yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia yaitu **MySQL AB**. MySQL AB memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "**Monty**" Widenius.

MySQL dapat didownload di situs resminya, http://www.mysql.com.

Fitur-fitur MySQL antara lain :

- **Relational Database System**. Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.
- **Arsitektur Client-Server**. MySQL memiliki arsitektur client-server dimana server database MySQL terinstal di server. Client MySQL dapat berada di komputer yang sama dengan server, dan dapat juga di komputer lain yang berkomunikasi dengan server melalui jaringan bahkan internet.
- **Mengenal perintah SQL standar**. SQL (Structured Query Language) merupakan suatu bahasa standar yang berlaku di hampir semua software database. MySQL mendukung SQL versi SQL:2003.
- Mendukung **Sub Select**. Mulai versi 4.1 MySQL telah mendukung select dalam select (sub select).
- Mendukung **Views**. MySQL mendukung views sejak versi 5.0
- Mendukung **Stored Prosedured (SP)**. MySQL mendukung SP sejak versi 5.0
- Mendukung **Triggers**. MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.
- Mendukung **replication**.
- Mendukung transaksi.
- Mendukung **foreign key**.

- Tersedia fungsi GIS.
- Free (bebas didownload)
- Stabil dan tangguh
- Fleksibel dengan berbagai pemrograman
- Security yang baik
- Dukungan dari banyak komunitas
- Perkembangan software yang cukup cepat.

# Bab 2
# Instalasi MySQL dan Software Pendukung

- Instalasi MySQL di Windows
- Instalasi Software Pendukung MySQL

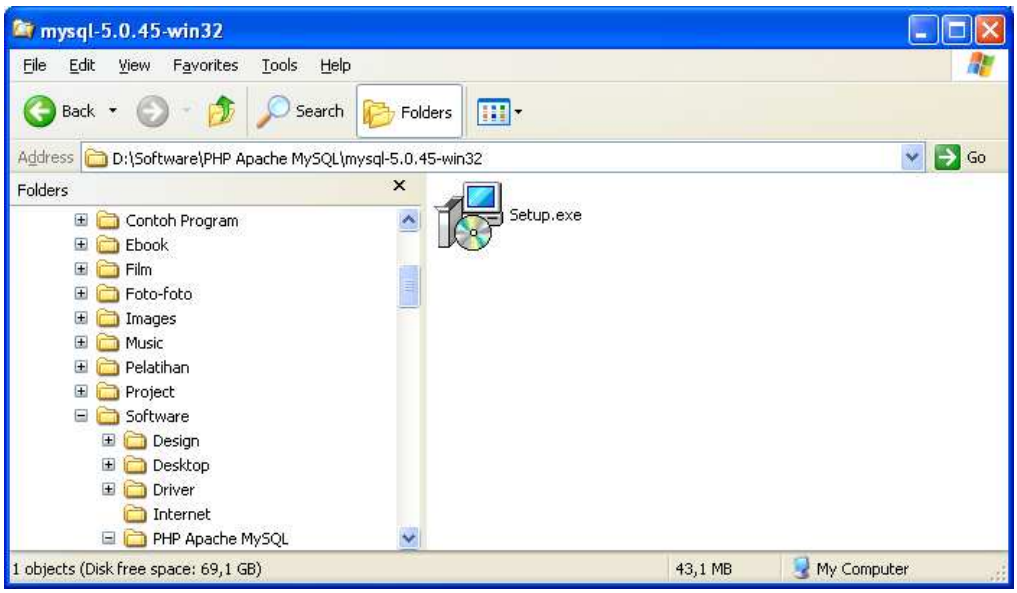
## Instalasi MySQL di Windows

### Persiapan

1. Download Source MySQL di http://www.mysql.com/downloads/
   MySQL versi terakhir saat materi ini dibuat adalah MySQL 5.0.45. Silahkan Anda download versi terakhir tersebut dan simpan di komputer Anda. Pada dasarnya, instalasi untuk setiap versi MySQL tidak jauh berbeda.

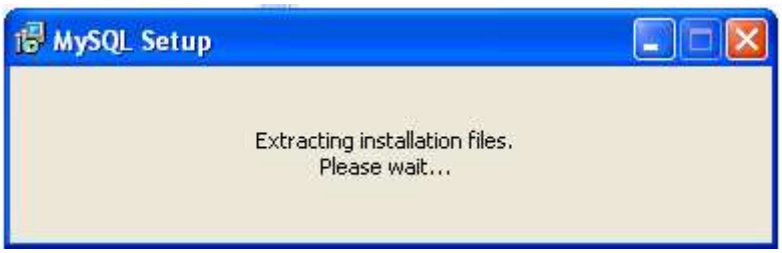
#### Proses Instalasi MySQL

1. Setelah Anda mendapatkan source MySQL, selanjutnya Anda perlu meng-ekstrak file tersebut ke komputer Anda.
2. Jalankan file **Setup.exe** yang ada di dalam folder source MySQL. Lihat gambar berikut ini !



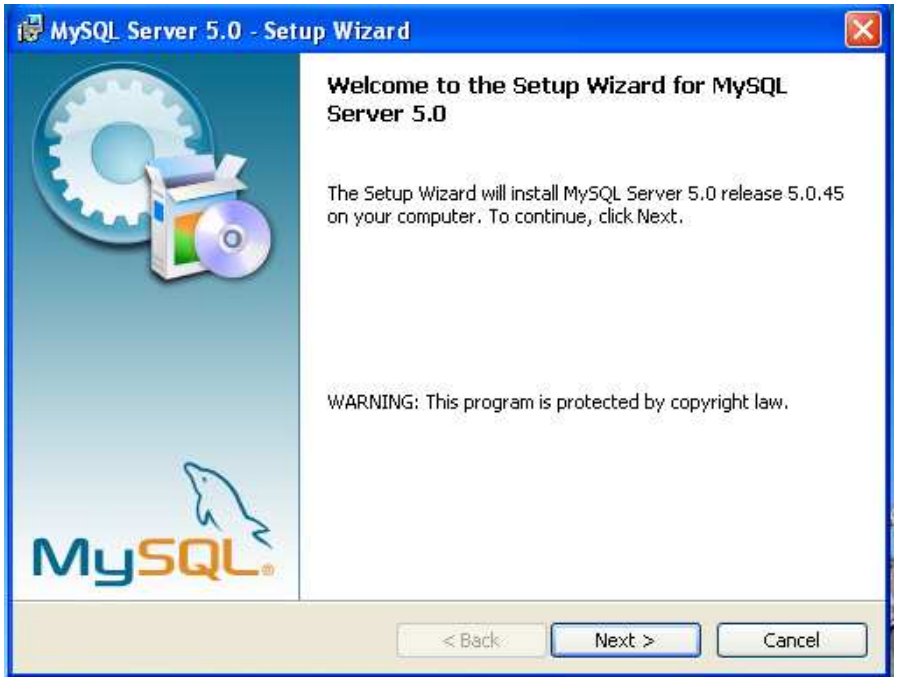

Gambar 2.1. File Setup.exe

3. MySQL Setup akan mengekstrak file instalasi MySQL seperti pada gambar berikut ini.




Gambar 2.2. Proses Instalasi Dimulai

4. Selanjutnya akan ditampilkan window **MySQL Server 5.0 Setup Wizard for MySQL**. Klik tombol **Next** untuk memulai proses instalasi.

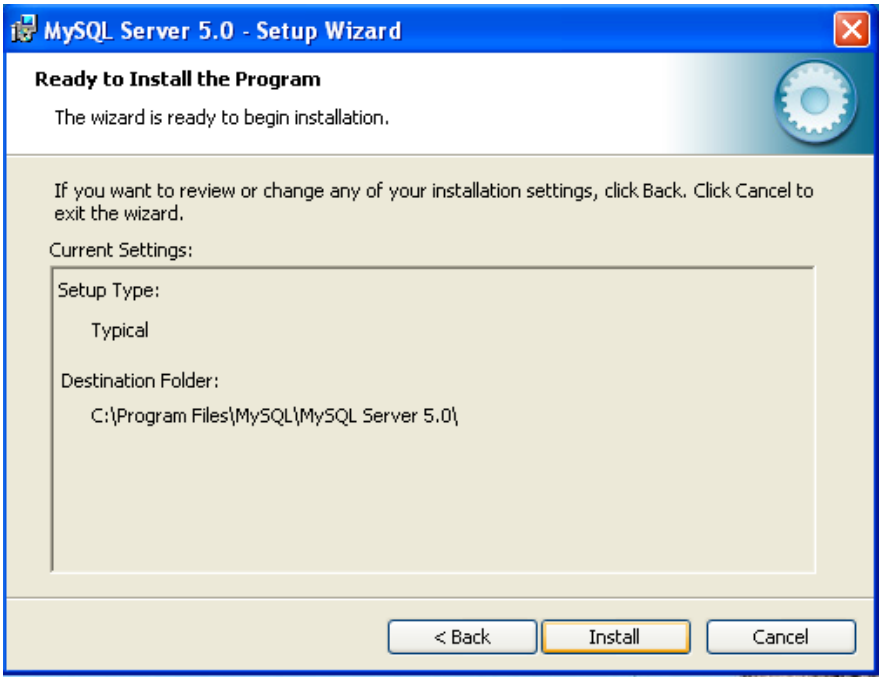



Gambar 2.3. Memulai Proses Instalasi

5. Selanjutnya akan ditampilkan pilihan untuk memilih cara instalasi. Pilih **Typical** jika kita ingin menginstall MySQL yang umumnya digunakan.


Gambar 2.4. Pilih tipe instalasi

6. Selanjutnya akan ditampilkan window informasi konfigurasi MySQL, yaitu tipe instalasi dan folder tujuan instalasi. Klik **Install** untuk memulai proses instalasi.

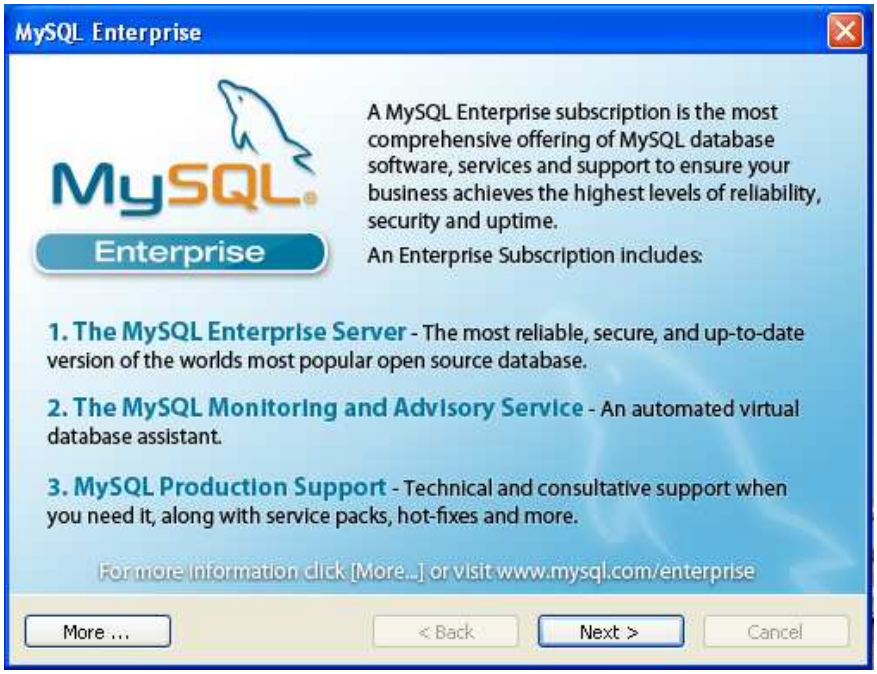



Gambar 2.5. Window Informasi Konfigurasi Instalasi

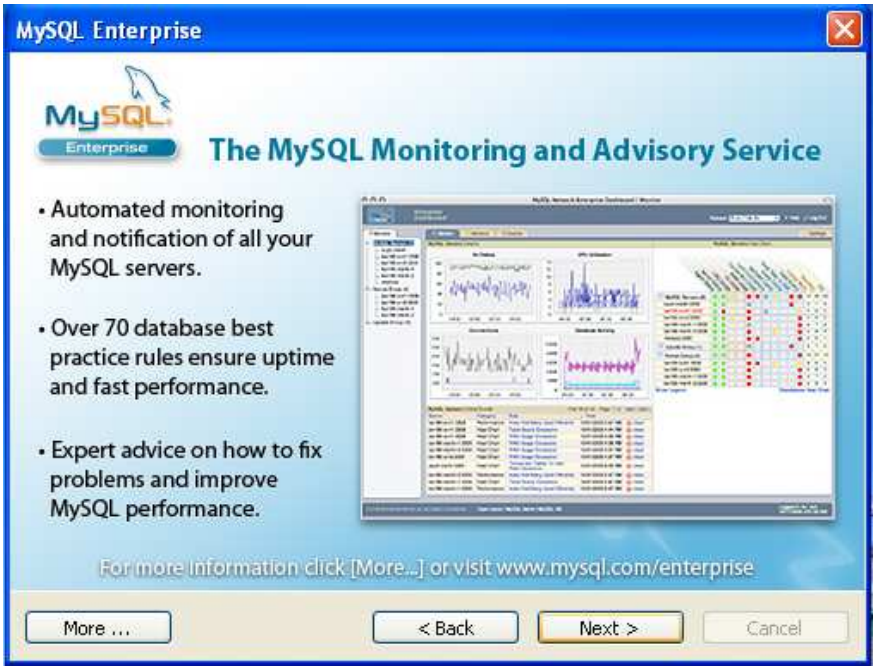
7. Proses instalasi dimulai.


Gambar 2.6. Window Setup Type

8. Selanjutnya ditampilkan window informasi mengenai **MySQL Enterprise**. Klik **Next** untuk melanjutkan.




Gambar 2.7. Window MySQL Enterprise




Gambar 2.8. Window informasi MySQL Monitoring

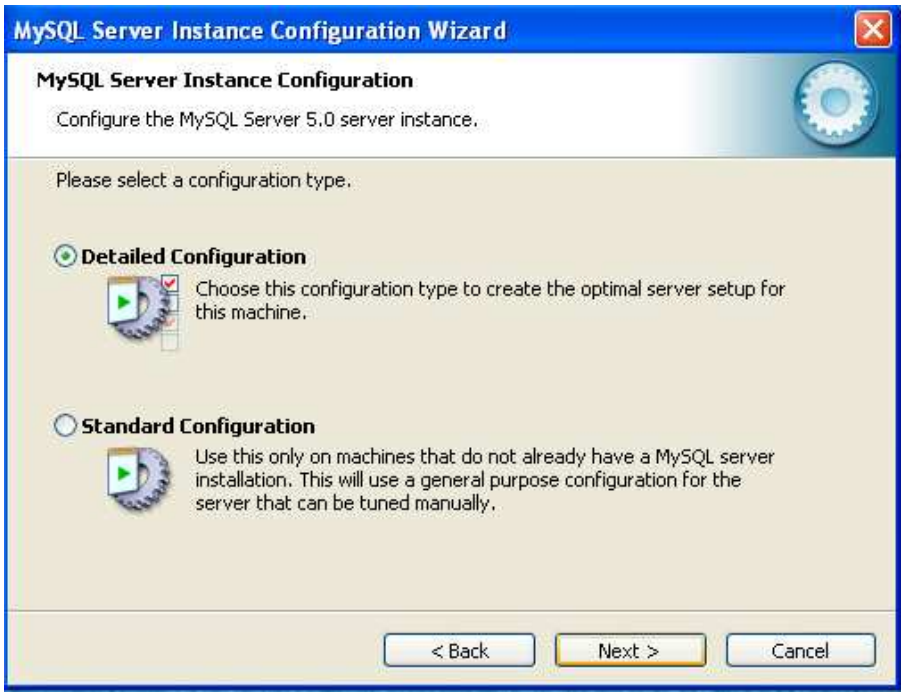
9. Proses instalasi selesai dan akan ditampilkan seperti pada gambar di bawah ini. Jika kita ingin langsung mengkonfigurasikan server MySQL (password, service dll) maka pilihkan checkbox **Configure the MySQL Server now** dan tekan tombol **Finish.**


Gambar 2.9. Proses instalasi Selesai

10. Selanjutnya ditampilkan window **MySQL Server Instance Configuration Wizard**. Klik **Next** untuk melanjutkan.


Gambar 2.10. Window MySQL Server Instance Configuration Wizard

11. Selanjutnya terdapat pilihan tipe konfigurasi yang diinginkan, **Detailed Configuration** atau **Standard Configuration**. Pilih dan klik **Next** untuk melanjutkan.

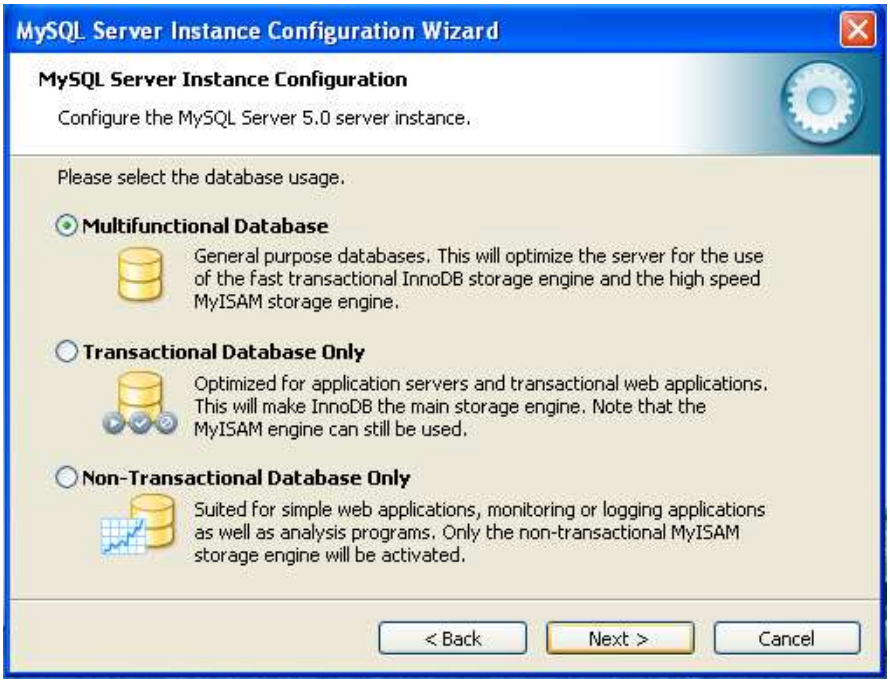



Gambar 2.11. Window Pilihan tipe konfigurasi

12. Selanjutnya terdapat pilihan tipe server yang diinginkan, **Developer, Server,** atau **Dedicated MySQL Server.** Pilih salah satu dan klik **Next** untuk melanjutkan.

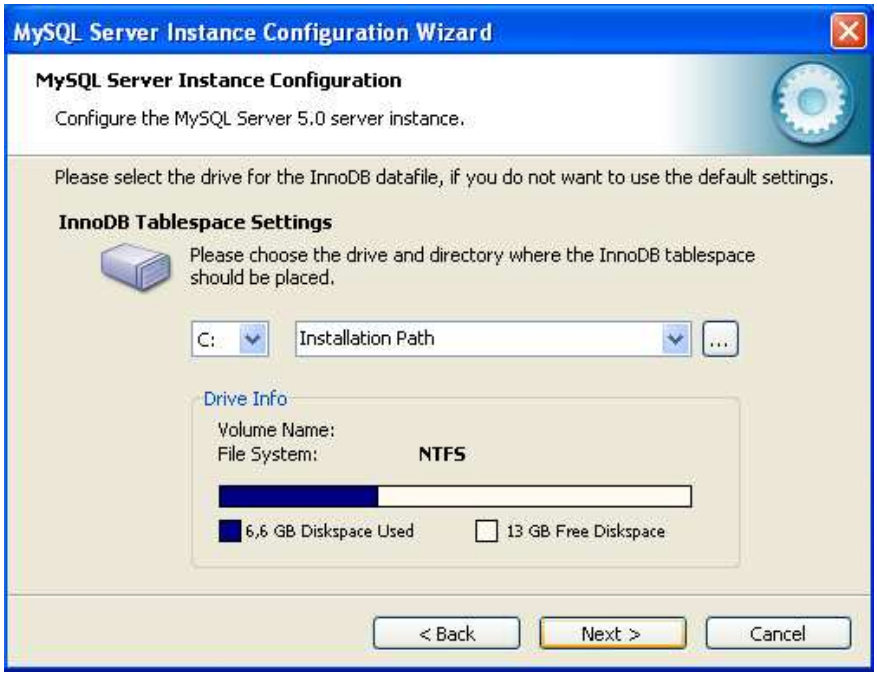

Gambar 2.12. Pilihan tipe server MySQL

13. Selanjutnya terdapat pilihan penggunaan database MySQL, untuk **Multifunctional, Transactional Only** atau **Non-Transactional Only.** Pilih salah satu dan klik **Next** untuk melanjutkan.



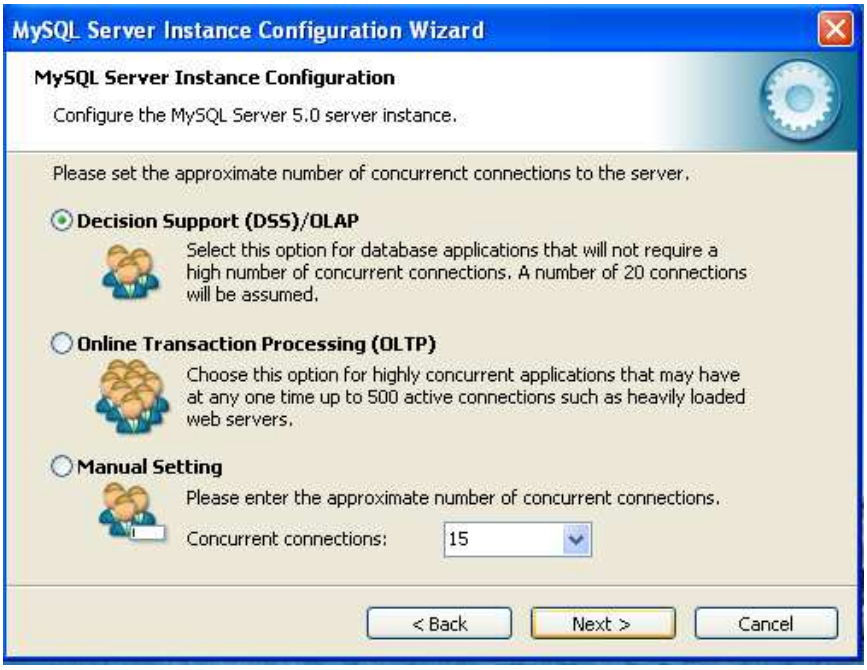

Gambar 2.13. Window Pilihan penggunaan Database.

14. Selanjutnya terdapat *setting*-an **InnoDB Tablespace Settings** dimana diminta memilih tempat untuk *tablespace* InnoDB**.** Klik **Next** untuk melanjutkan.



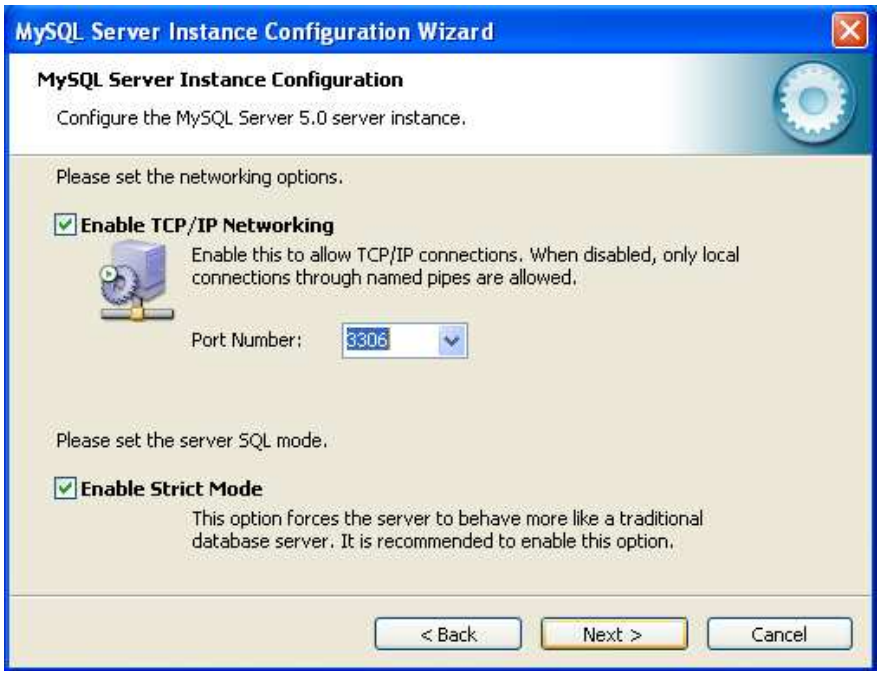

Gambar 2.14. Window InnoDB Tablespace Settings.

15. Selanjutnya terdapat pilihan perkiraan seberapa besar koneksi user ke server**.** Pilih salah satu dan klik **Next** untuk melanjutkan.



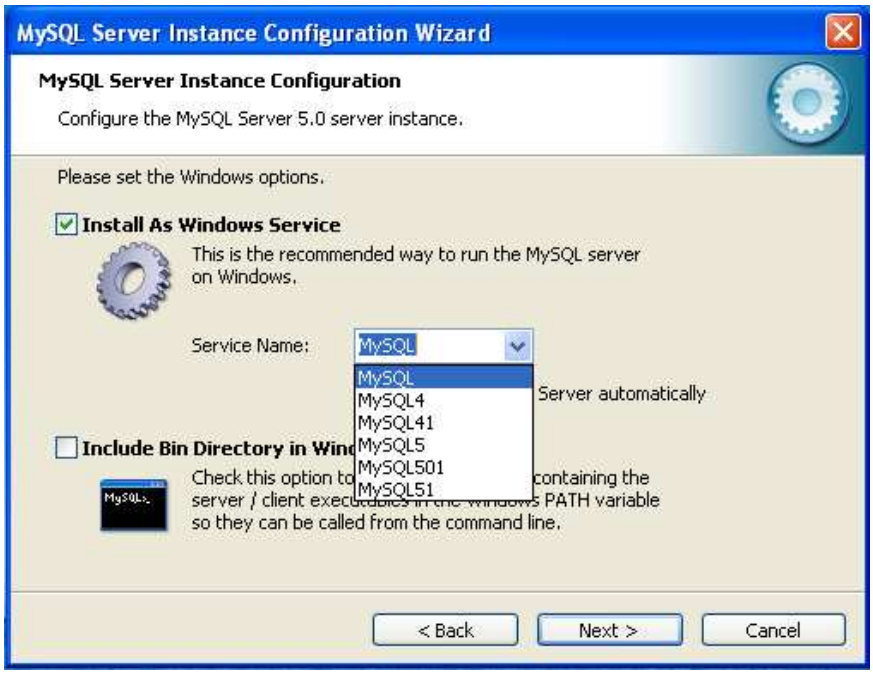

Gambar 2.15. Pilihan Perkiraan Seberapa Besar Koneksi User ke Server

16. Selanjutnya terdapat window untuk memilih nomor PORT yang digunakan untuk MySQL. **Next** untuk melanjutkan.




Gambar 2.16. Window pilihan port MySQL.

17. Selanjutnya terdapat pilihan nama service MySQL yang akan digunakan oleh Windows. Pilih salah satu dan klik **Next** untuk melanjutkan.

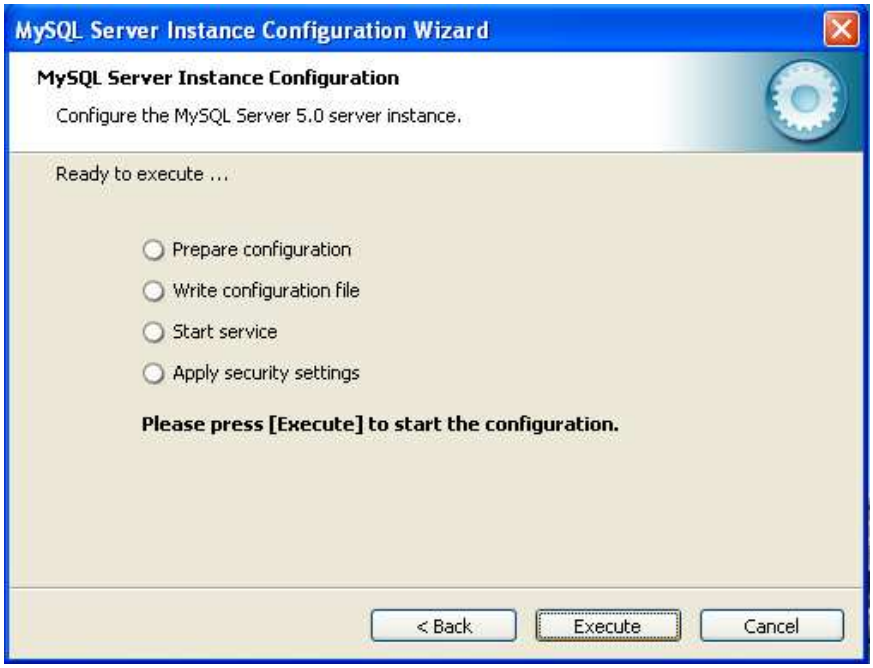



Gambar 2.17. Window pilihan Nama Service MySQL.

18. Selanjutnya diminta memodifikasi *security***.** Pilih password untuk root (user tertinggi di MySQL) dan klik **Next** untuk melanjutkan.


Gambar 2.18. Window Security Setting.

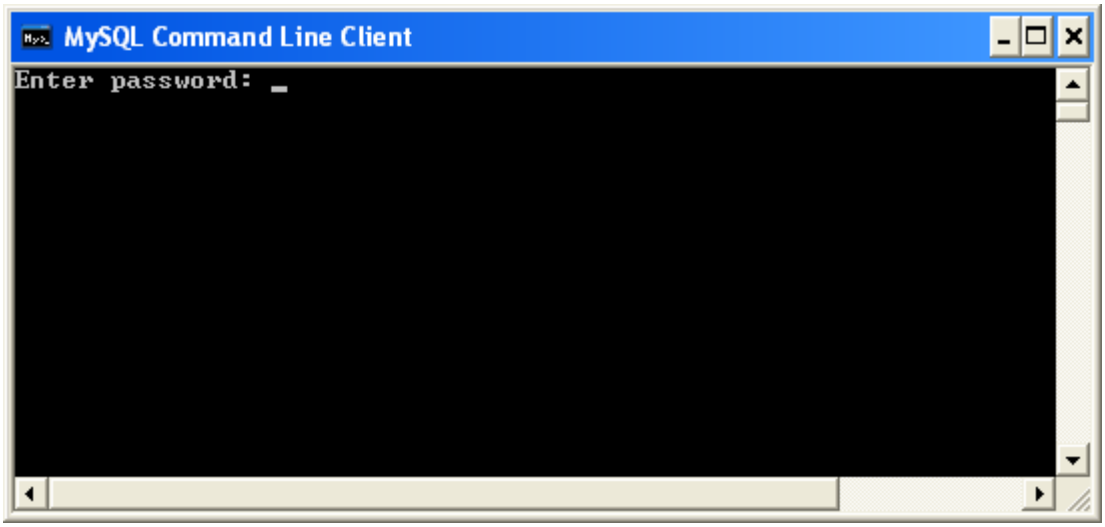
19. Proses konfigurasi selesai dan klik **Execute** untuk menyimpan konfigurasi dan menjalankan servis MySQL.




Gambar 2.19. Proses konfigurasi server MySQL selesai.

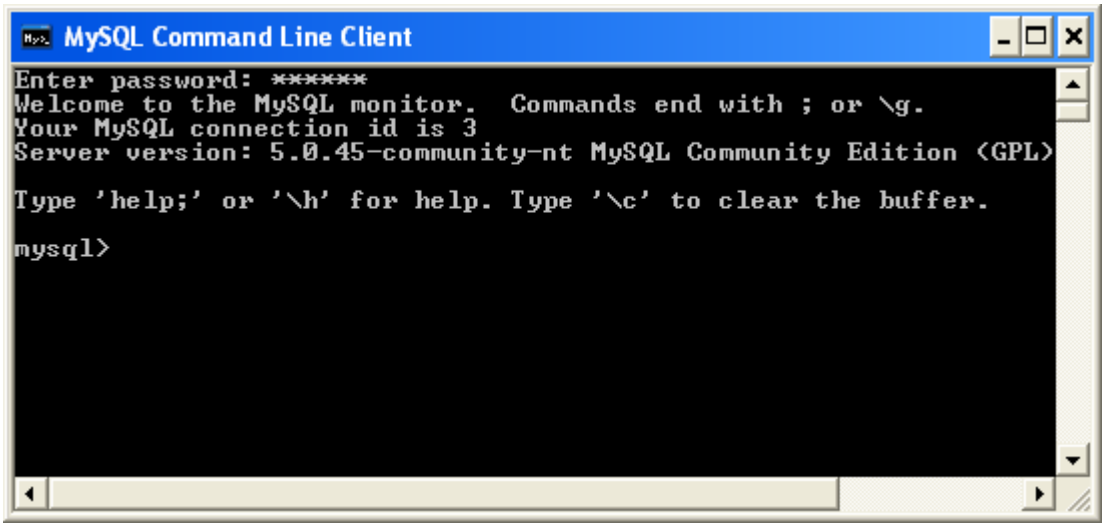
## Koneksi ke Server MySQL dengan MySQL Client

MySQL menyediakan tools untuk melakukan koneksi ke server MySQL, yaitu MySQL Command-Line Client. Tools tersebut dapat diakses dari menu **Start > All Programs > MySQL > MySQL Server 5> MySQL Command Line Client**. Tampilannya kurang lebih tampak pada gambar berikut ini:




Gambar 12.13. MySQL *Command Line Client*

Untuk melakukan koneksi ke server MySQL, Anda cukup mengetikkan password koneksi MySQL. Password ini didefinisikan pada saat proses instalasi. Jika passwordnya benar, maka akan ditampilkan window sbb :




Gambar 2.20. Koneksi ke Server MySQL dengan User root

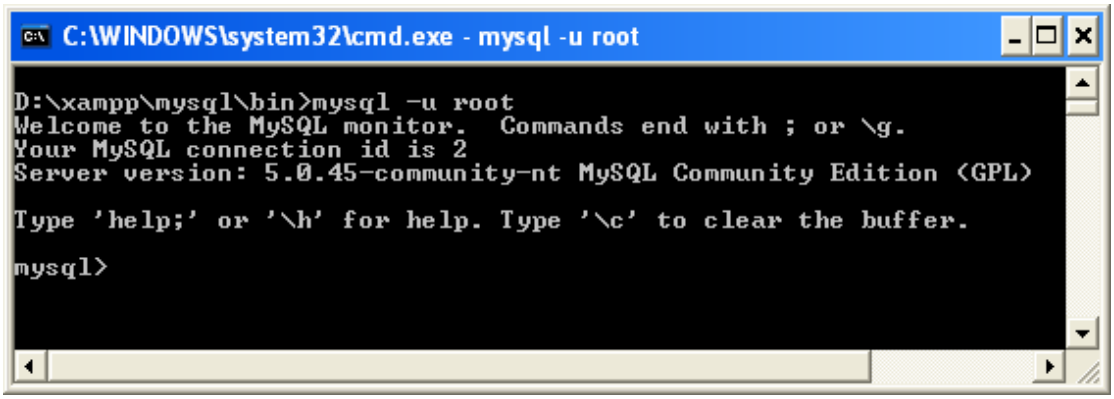
Setelah koneksi ke server MySQL berhasil dilakukan, maka akan ditampilkan prompt **mysql**> seperti pada gambar 12.14. Query atau perintah-perintah MySQL dapat dituliskan pada prompt MySQL ini. Akhiri setiap query dengan titik-koma (;). Selanjutnya untuk keluar dari server MySQL dapat dilakukan dengan mengetikkan perintah **quit** atau **\q** pada *prompt* mysql>.

## Berbagai MySQL Client untuk Administrasi Server MySQL

Berikut ini beberapa *tools* yang biasa digunakan dalam mempermudah administrasi server MySQL. *Tools* berikut ini hanya digunakan untuk mempermudah administrasi MySQL, jadi tidak harus digunakan.

### 1. MySQL Command Line Client

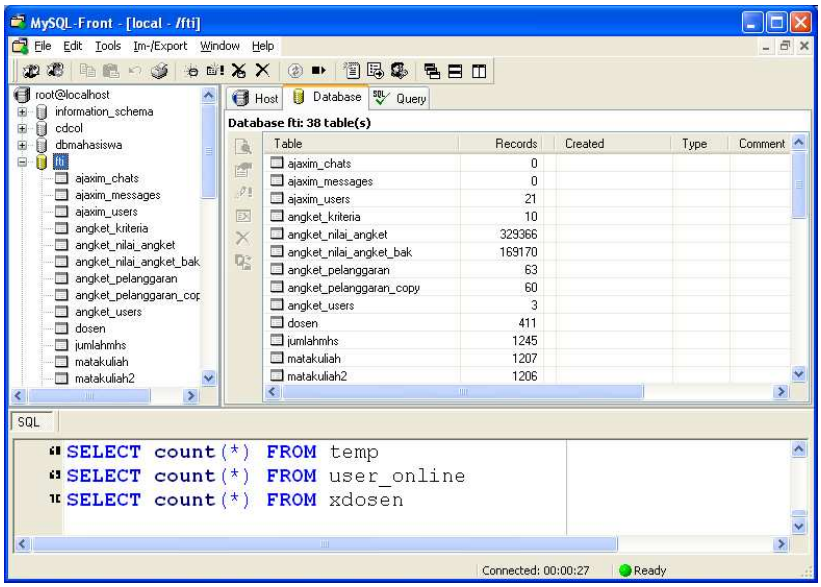
MySQL Command Line Client merupakan tools *default* MySQL yang sudah disertakan dalam file instalasi MySQL. Aplikasi ini dapat digunakan untuk melakukan koneksi ke MySQL melalui *text-based mode*.




Gambar 2.21. Tampilan MySQL command line client

## 2. MySQL-Front

MySQL-Front merupakan front-end MySQL berbasis Windows yang cukup banyak digunakan. MySQL-Front memiliki user interface yang cukup mudah digunakan, bahkan oleh user pemula. Pada awalnya MySQL-Front merupakan software yang free, namun mulai versi 3.0 ke atas, software ini menjadi software yang bersifat shareware dengan masa percobaan selama 30 hari. Jika Anda ingin mencoba software ini, cobalah MySQL-Front versi 2.5 karena selain masih bebas untuk didownload, versi 2.5 cukup stabil dan sudah teruji. Situs resmi MySQL-Front beralamat di http://www.mysqlfront.de

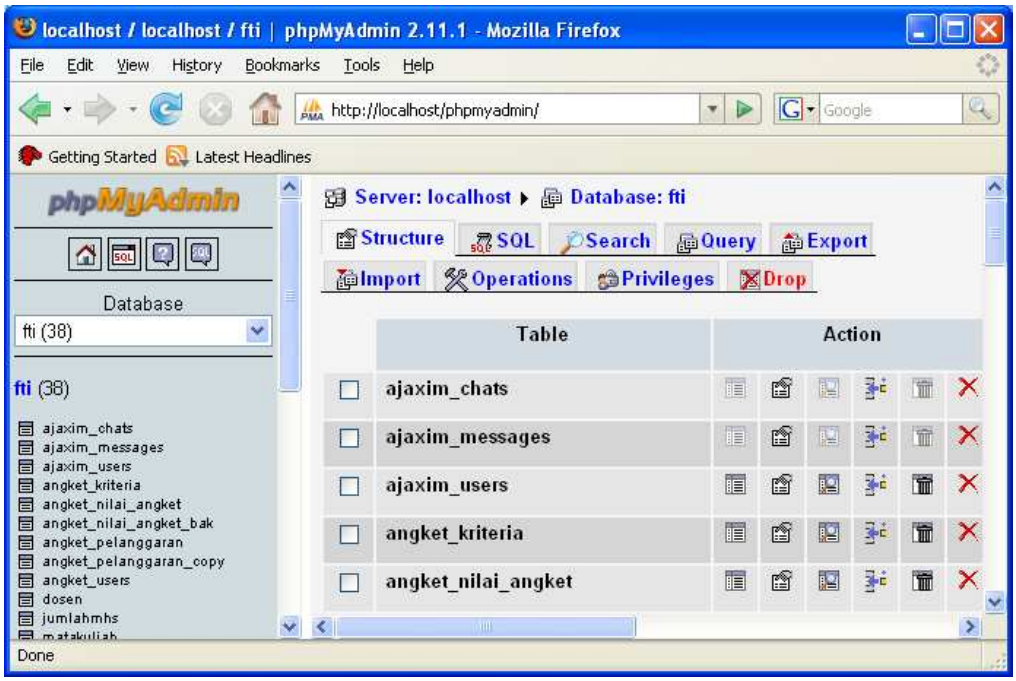



Gambar 2.22. Tampilan MySQL Front

## 3. PHPMyAdmin

PHPMyAdmin merupakan front-end MySQL berbasis web. PHPMyAdmin dibuat dengan menggunakan PHP. Saat ini, PHPMyAdmin banyak digunakan dalam

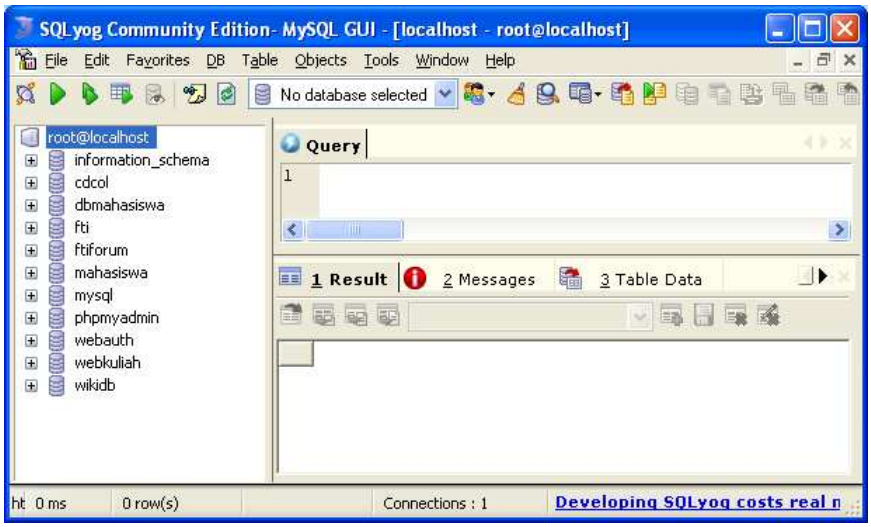
hampir semua penyedia hosting yang ada di internet. PHPMyAdmin mendukung berbagai fitur administrasi MySQL termasuk manipulasi database, tabel, index dan juga dapat mengeksport data ke dalam berbagai format data. PHPMyAdmin juga tersedia dalam 50 bahasa lebih, termasuk bahasa Indonesia. PHPMyAdmin dapat didownload secara gratis di http://www.phpmyadmin.net




Gambar 2.23. Tampilan halaman PHPMyAdmin

**4. SQLYog**

SQLYog merupakan salah satu front-end MySQL yang cukup populer saat ini. Dengan dukungan fitur yang cukup banyak dan lengkap, SQL Yog tersedia versi commercial dan community (free). SQLYog dapat didownload di situsnya http://www.webyog.com




Gambar 2.24. Tampilan layar SQLYog

**5. MySQL Administrator** dan **MySQL Query Browser**

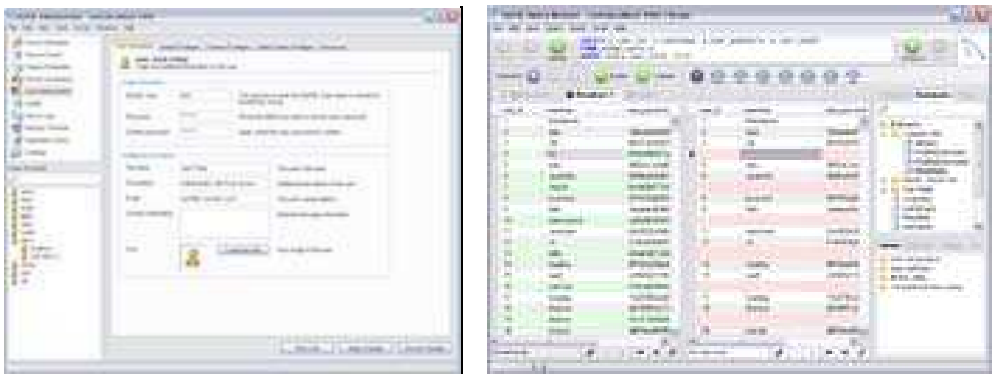
MySQL Administrator dan MySQL Query Browser merupakan tools administrasi database MySQL yang tersedia di situs resmi MySQL (http://www.mysql.com). Keduanya dapat didownload di alamat http://www.mysql.com/products/tools/.

Beberapa fitur **MySQL Administrator**, antara lain:

- Administrasi user.
- Halaman monitoring server.
- Optimatisasi MySQL
- Informasi umum keadaan server
- Status *replication*.
- *Cross-platform*.

Beberapa fitur **MySQL Query Browser**, antara lain:

- Tampilan dan menu yang mudah (*user-friendly*).
- Mendukung beberapa window hasil (*result preview*) sekaligus.
- Kemudahan dalam menulis query dengan *visual tools*.
- Manipulasi database.
- Membuat dan manipulasi tabel.
- SQL *statements debugging*.



Gambar 2.25. Tampilan layar MySQL Administrator dan MySQL Control Center

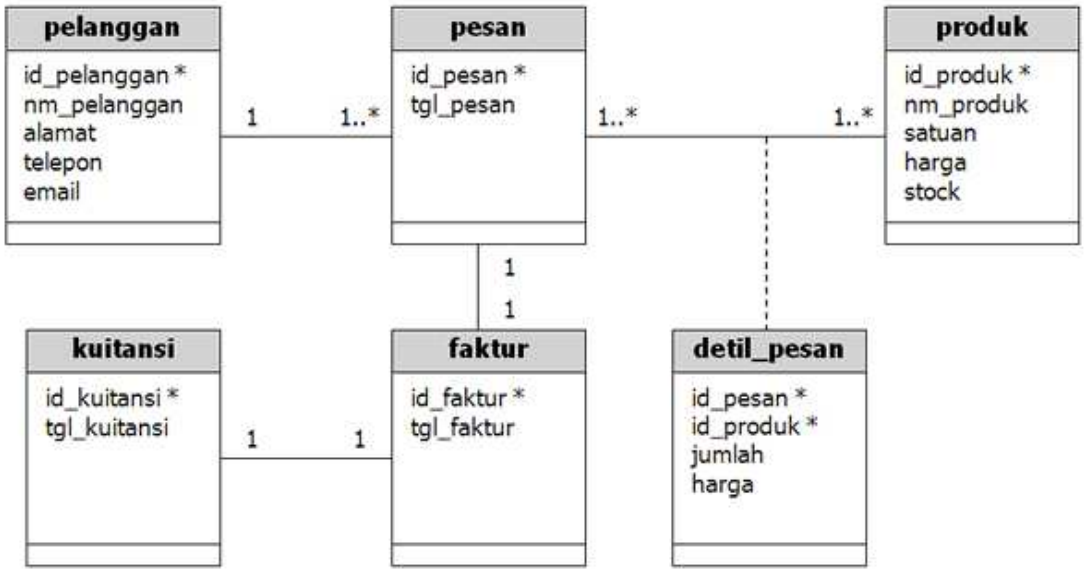
# Bagian 2
# Dasar-dasar MySQL

# Bab 3
# Merancang Database

- Tipe Table MySQL
- Tipe Field MySQL
- Merancang suatu database yang baik.

Merancang database merupakan hal yang pertama kali harus dilakukan sebelum membuat suatu aplikasi berbasis database. Rancangan database yang baik akan menentukan seberapa baik sebuah aplikasi dibangun. Orang yang bertanggung jawab dalam merancang database biasanya disebut sebagai seorang sistem analis.

Berikut ini contoh sederhana sebuah rancangan database dalam pada **Sistem Pemesanan Barang (ordering system)**. Rancangan database disajikan dalam bentuk class diagram.




Gambar 3.1. Contoh Class Diagram Sistem Pemesanan Barang

## Tipe-tipe Tabel MySQL

**Salah** satu kelebihan dari MySQL adalah Anda dapat mendefinisikan tipe untuk tiap tabel. MySQL mendukung beberapa tipe tabel, tergantung konfigurasi saat proses instalasi MySQL. MySQL memiliki 3 (tiga) tipe data utama, yaitu MyISAM, InnoDB dan HEAP.

Jika kita tidak menyebutkan tipe tabel saat membuat tabel, maka tipe tabel otomatis akan dibuat sesuai konfigurasi default server MySQL. Hal ini ditentukan oleh variabel `default-table-type` di file konfigurasi MySQL.

## MyISAM

Tipe tabel MyISAM merupakan tipe tabel yang sederhana, stabil dan mudah digunakan. Jika kita akan menyimpan data sederhana yang tidak terlalu rumit, maka gunakanlah tipe tabel ini. Kelebihan utama MyISAM adalah kecepatan dan kestabilannya. Jika kita memilih tipe tabel MyISAM, maka MySQL secara otomatis akan menentukan salah satu dari tiga jenis tabel MyISAM, yaitu :

a. **MyISAM static**. Jenis ini digunakan ketika semua kolom dalam tabel didefinisikan dengan ukuran yang pasti (fixed). Dengan kata lain, tidak ada kolom yang memiliki tipe seperti VARCHAR, TEXT dan BLOB. Karena sifatnya yang fixed, maka jenis ini akan lebih cepat, aman dan stabil.
b. **MyISAM dymanic**. Jenis ini digunakan ketika terdapat kolom dengan tipe yang dinamis, seperti tipe kolom VARCHAR. Keuntungan utama dari jenis ini adalah ukuran yang dinamis. Jadi sifatnya lebih efektif karena ukuran data (file) menyesuaikan isi dari masing-masing kolom (field).
c. **MyISAM Compressed**. Kedua jenis MyISAM, static dan dynamic dapat dikompresi menjadi satu jenis yaitu MyISAM Compressed dengan perintah `myisamchk`. Tentunya hasilnya lebih kecil dari segi ukuran. Tabel yang terkompresi tidak dapat dikenakan operasi seperti INSERT, UPDATE dan DELETE.

## InnoDB

Tipe tabel InnoDB merupakan tipe tabel MySQL yang mendukung proses transaksi. Tipe ini memiliki beberapa keunggulan, antara lain:

a. Mendukung transaksi antar tabel.
b. Mendukung row-level-locking.
c. Mendukung Foreign-Key Constraints.
d. Crash recovery.

## HEAP

Tabel dengan tipe HEAP tidak menyimpan datanya di hardisk, tetapi menyimpan di RAM (memori). Tipe tabel ini biasanya digunakan sebagai tabel sementara (temporary). Tabel secara otomatis akan dihapus (hilang) dari MySQL saat koneksi ke server diputus atau server MySQL dimatikan.

### Tipe Tabel yang Lain

Selain 3 (tiga) tipe tabel diatas, yaitu MyISAM, InnoDB dan HEAP, MySQL juga mendukung tipe tabel yang lain, yaitu:

a. **BDB**. Tipe tabel ini mirip tipe tabel InnoDB, namun penggunaannya belum maksimal.
b. **Archieve**. Tipe ini tersedia sejak MySQL versi 4.1. Tipe ini digunakan untuk menyimpan tabel yang terkompresi, dimana biasanya digunakan dalam proses backup.
c. **CSV**. Tipe ini digunakan untuk menyimpan data dalam bentuk file text yang dibatasi dengan koma (delimiter). Tipe ini tersedia sejak MySQL versi 4.1.
d. **NDB Table** (MySQL Cluster). Tersedia sejak MySQL versi 4.1.
e. **Federated** (External Tables). Tipe ini tersedia sejak MySQL versi 5.0.

## Tipe-tipe Field (Kolom) MySQL

MySQL memiliki cukup banyak tipe data untuk field (kolom) tabel. Tipe field (kolom) ini menentukan besar kecilnya ukuran suatu tabel. Tipe field di MySQL setidaknya terbagi menjadi beberapa kelompok, yaitu numerik, string, date-and-time, dan kelompok himpunan (set dan enum). Masing-masing tipe field memiliki batasan lebar dan ukurannya.

### Tipe Numeric

Tipe data numerik digunakan untuk menyimpan data numeric (angka). Ciri utama data numeric adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe numerik:

1. TINYINT
   **Penggunaan** : digunakan untuk menyimpan data bilangan bulat positif dan negatif.
   **Jangkauan** : -128 s/d 127
   **Ukuran** : 1 byte (8 bit).

2. SMALLINT

   **Penggunaan** : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

   **Jangkauan** : -32.768 s/d 32.767

   **Ukuran** : 2 byte (16 bit).

3. MEDIUMINT

   **Penggunaan** : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

   **Jangkauan** : -8.388.608 s/d 8.388.607

   **Ukuran** : 3 byte (24 bit).

4. INT

   **Penggunaan** : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

   **Jangkauan** : -2.147.483.648 s/d 2.147.483.647

   **Ukuran** : 4 byte (32 bit).

5. BIGINT

   **Penggunaan** : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

   **Jangkauan** : ± 9,22 x $10^{18}$

   **Ukuran** : 8 byte (64 bit).

6. FLOAT

   **Penggunaan** : digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi tunggal.

   **Jangkauan** : -3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38.

   **Ukuran** : 4 byte (32 bit).

7. DOUBLE

   **Penggunaan** : digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi ganda.

   **Jangkauan** : -1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.

   **Ukuran** : 8 byte (64 bit).

8. REAL
   Merupakan sinonim dari DOUBLE.

9. DECIMAL
   **Penggunaan** : digunakan untuk menyimpan data bilangan pecahan positif dan negatif.
   **Jangkauan** : -1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.
   **Ukuran** : 8 byte (64 bit).

10. NUMERIC
    Merupakan sinonim dari DECIMAL.

## Tipe Date dan Time

Tipe data *date* dan *time* digunakan untuk menyimpan data tanggal dan waktu. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe *date* dan *time*:

11. DATE
    **Penggunaan** : digunakan untuk menyimpan data tanggal.
    **Jangkauan** : 1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)
    **Ukuran** : 3 byte.

12. TIME
    **Penggunaan** : digunakan untuk menyimpan data waktu.
    **Jangkauan** : -838:59:59 s/d +838:59:59 (HH:MM:SS)
    **Ukuran** : 3 byte.

13. DATETIME
    **Penggunaan** : digunakan untuk menyimpan data tanggal dan waktu.
    **Jangkauan** : '1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'
    **Ukuran** : 8 byte.

14. YEAR
    **Penggunaan** : digunakan untuk menyimpan data tahun dari tanggal.
    **Jangkauan** : 1900 s/d 2155
    **Ukuran** : 1 byte.

## Tipe String (Text)

Tipe data string digunakan untuk menyimpan data string (text). Ciri utama data string adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe string:

15. CHAR
    **Penggunaan** : digunakan untuk menyimpan data string ukuran tetap.
    **Jangkauan** : 0 s/d 255 karakter

16. VARCHAR
    **Penggunaan** : digunakan untuk menyimpan data string ukuran dinamis.
    **Jangkauan** : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

17. TINYTEXT
    **Penggunaan** : digunakan untuk menyimpan data text.
    **Jangkauan** : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

18. TEXT
    **Penggunaan** : digunakan untuk menyimpan data text.
    **Jangkauan** : 0 s/d 65.535 ($2^{16}$ - 1) karakter

19. MEDIUMTEXT
    **Penggunaan** : digunakan untuk menyimpan data text.
    **Jangkauan** : 0 s/d $2^{24}$ - 1 karakter

20. LONGTEXT
    **Penggunaan** : digunakan untuk menyimpan data text.
    **Jangkauan** : 0 s/d $2^{32}$ - 1 karakter

## Tipe BLOB (Biner)

Tipe data blob digunakan untuk menyimpan data biner. Tipe ini biasanya digunakan untuk menyimpan kode-kode biner dari suatu file atau object. BLOB

merupakan singkatan dari Binary Large Object. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe blob:

21. BIT (sejak versi 5.0.3)
    **Penggunaan** : digunakan untuk menyimpan data biner.
    **Jangkauan** : 64 digit biner

22. TINYBLOB
    **Penggunaan** : digunakan untuk menyimpan data biner.
    **Jangkauan** : 255 byte

23. BLOB
    **Penggunaan** : digunakan untuk menyimpan data biner.
    **Jangkauan** : $2^{16}$ - 1 byte

24. MEDIUMBLOB
    **Penggunaan** : digunakan untuk menyimpan data biner.
    **Jangkauan** : $2^{24}$ - 1 byte

25. LONGBLOB
    **Penggunaan** : digunakan untuk menyimpan data biner.
    **Jangkauan** : $2^{32}$ - 1 byte

## Tipe Data yang Lain

Selain tipe data di atas, MySQL juga menyediakan tipe data yang lain. Tipe data di MySQL mungkin akan terus bertambah seiring dengan perkembangan versi MySQL. Berikut ini beberapa tipe data tambahan MySQL:

26. ENUM
    **Penggunaan** : enumerasi (kumpulan data).
    **Jangkauan** : sampai dengan 65535 string.

27. SET
    **Penggunaan** : combination (himpunan data).
    **Jangkauan** : sampai dengan 255 string anggotas.

# Merancang Database yang Baik

Seperti telah disebutkan sebelumnya, bahwa rancangan database menentukan suatu aplikasi efektif atau tidak, efisien atau tidak, baik atau tidak. Pembahasan mengenai bagaimana merancang database yang baik tentunya sangat panjang. Kita dapat mencari referensi terkait dengan perancangan database.

## Beberapa Aturan Merancang Database yang Baik.

a. Tabel dalam database tidak boleh mengandung record (data) ganda, atau dengan kata lain tidak boleh ada redudancy data. Jika terdapat data yang sama, maka perlu dilihat kembali rancangan tabelnya.
b. Setiap tabel dalam database, harus memiliki field (kolom) yang unik. Field ini disebut sebagai Primary Key.
c. Tabel harus sudah normal.
d. Besar atau ukuran database hendaknya dibuat seminimal mungkin. Hal ini ditentukan oleh pemilihan tipe data yang tepat.
e. Merancang database hendaknya memperhatikan apakah rancangan dapat menampung data (record) sesuai yang dibutuhkan oleh aplikasi.

## Tips Penamaan Identifier.

a. Penamaan identifier (database, tabel, kolom) di MySQL bersifat case-sensitive. Penamaan identifier hendaknya konsisten untuk semua tabel dalam suatu database. Kita dapat menggunakan model lower-case, UPPER-CASE, camelCase dll.
b. Nama database, tabel dan kolom maksimal 64 karakter.
c. Hindari penggunaan karakter khusus, seperti üàû, karena bisa bermasalah dalam sistem operasi yang lain.
d. Pilih nama untuk field (kolom) yang mencerminkan isi dari data yang disimpan.

# Bab 4
# Dasar-dasar SQL

- Pendahuluan
- Perintah DDL
- Perintah DML

## Pendahuluan

**SQL** merupakan singkatan dari *Structured Query Language*. SQL atau juga sering disebut sebagai query merupakan suatu bahasa (*language*) yang digunakan untuk mengakses database. SQL dikenalkan pertama kali dalam IBM pada tahun 1970 dan sebuah standar ISO dan ANSII ditetapkan untuk SQL. Standar ini tidak tergantung pada mesin yang digunakan (IBM, Microsoft atau Oracle). Hampir semua software database mengenal atau mengerti SQL. Jadi, perintah SQL pada semua software database hampir sama.

Terdapat 3 (tiga) jenis perintah SQL, yaitu :

1. DDL atau *Data Definition Language*

   DDL merupakan perintah SQL yang berhubungan dengan pendefinisian suatu struktur database, dalam hal ini *database* dan *table*. Beberapa perintah dasar yang termasuk DDL ini antara lain :
   - CREATE
   - ALTER
   - RENAME
   - DROP

2. DML atau *Data Manipulation Language*

   DML merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data atau *record* dalam table. Perintah SQL yang termasuk dalam DML antara lain :
   - SELECT
   - INSERT
   - UPDATE
   - DELETE

3. DCL atau *Data Control Language*

   DCL merupakan perintah SQL yang berhubungan dengan manipulasi user dan hak akses (priviledges). Perintah SQL yang termasuk dalam DCL antara lain :

   - GRANT
   - REVOKE

## Membuat, Menampilkan, Membuka dan Menghapus Database

### Membuat Database

Sintaks umum SQL untuk membuat suatu database adalah sebagai berikut :

```
CREATE DATABASE [IF NOT EXISTS] nama_database;
```

Bentuk perintah di atas akan membuat sebuah database baru dengan nama `nama_database`. Aturan penamaan sebuah database sama seperti aturan penamaan sebuah variabel, dimana secara umum nama database boleh terdiri dari huruf, angka dan *under-score* (_). Jika database yang akan dibuat sudah ada, maka akan muncul pesan error. Namun jika ingin otomatis menghapus database yang lama jika sudah ada, aktifkan option `IF NOT EXISTS`.

Berikut ini contoh perintah untuk membuat database baru dengan nama "**penjualan**" :

```
CREATE DATABASE penjualan;
```

Jika query di atas berhasil dieksekusi dan database berhasil dibuat, maka akan ditampilkan pesan kurang lebih sebagai berikut :

```
Query OK, 1 row affected (0.02 sec)
```

### Menampilkan Database

Untuk melihat database yang baru saja dibuat atau yang sudah ada, dapat menggunakan perintah sebagai berikut :

```
SHOW DATABASES;
```

Hasil dari perintah di atas akan menampilkan semua database yang sudah ada di MySQL. Berikut ini contoh hasil dari query di atas :

```
+--------------+
| Database     |
+--------------+
| penjualan    |
| mysql        |
| test         |
+--------------+
3 rows in set (0.02 sec)
```

## Membuka Database

Sebelum melakukan manipulasi tabel dan record yang berada di dalamnya, kita harus membuka atau mengaktifkan databasenya terlebih dahulu. Untuk membuka database "**penjualan**", berikut ini querynya :

```
USE penjualan;
```

Jika perintah atau query di atas berhasil, maka akan ditampilkan pesan sebagai berikut :

```
Database changed
```

## Menghapus Database

Untuk menghapus suatu database, sintaks umumnya adalah sbb :

```
DROP DATABASE [IF EXISTS] nama_database;
```

Bentuk perintah di atas akan menghapus database dengan nama `nama_database`. Jika databasenya ada maka database dan juga seluruh tabel di dalamnya akan dihapus. Jadi berhati-hatilah dengan perintah ini! Jika nama database yang akan dihapus tidak ditemukan, maka akan ditampilkan pesan error. Aktifkan option `IF EXISTS` untuk memastikan bahwa suatu database benar-benar ada.

Berikut ini contoh perintah untuk menghapus database dengan nama "**penjualan**" :

```
DROP DATABASE penjualan;
```

# Membuat, Mengubah dan Menghapus *Table*

## Membuat Tabel Baru

Bentuk umum SQL untuk membuat suatu *table* secara sederhana sebagai berikut :

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

Bentuk umum di atas merupakan bentuk umum pembuatan tabel yang sudah disederhanakan. Penamaan tabel dan field memiliki aturan yang sama dengan penamaan database.

Sebagai contoh, kita akan membuat tabel baru dengan struktur sebagai berikut :

Nama tabel : **pelanggan**

| No | Nama Field | Tipe | Panjang |
|---|---|---|---|
| 1 | **id_pelanggan *** | Varchar | 5 |
| 2 | nm_pelanggan | Varchar | 30 |
| 3 | alamat | Text | - |
| 4 | telepon | Varchar | 20 |
| 5 | email | Varchar | 50 |

Untuk membuat tabel tersebut di atas, query atau perintah SQL-nya adalah sebagai berikut :

```
CREATE TABLE pelanggan (
 id_pelanggan varchar(5) NOT NULL,
 nm_pelanggan varchar(30) NOT NULL,
 alamat text,
 telepon varchar (20),
 email varchar (50),
 PRIMARY KEY(id_pelanggan)
);
```

Jika query untuk membuat tabel di atas berhasil dijalankan, maka akan ditampilkan pesan sebagai berikut :

```
Query OK, 0 rows affected (0.16 sec)
```

Pada perintah di atas, beberapa hal yang perlu diperhatikan :

- `CREATE TABLE` merupakan perintah dasar dari pembuatan table.
- `pelanggan` merupakan nama tabel yang akan dibuat.
- `id_pelanggan`, `nm_pelanggan`, `alamat`, `telepon` dan `email` merupakan nama *field*.
- `varchar` dan `text` merupakan tipe data dari field
- `NOT NULL` merupakan option untuk menyatakan bahwa suatu field tidak boleh kosong.
- `PRIMARY KEY` merupakan perintah untuk menentukan field mana yang akan dijadikan primary key pada tabel.
- `5`, `10`, `30` dan `50` di belakang tipe data merupakan panjang maksimal dari suatu *field*.
- Untuk tipe data date dan text (juga date dan blob) panjang karakter maksimalnya tidak perlu ditentukan.
- Jangan lupa akhiri perintah dengan titik-koma (;)

Selanjutnya untuk melihat tabel mhs sudah benar-benar sudah ada atau belum, ketikkan perintah berikut ini :

```
SHOW TABLES;
```

Perintah di atas akan menampilkan seluruh tabel yang sudah ada dalam suatu database. Contoh hasil dari perintah di atas adalah sebagai berikut :

```
+---------------------+
| Tables_in_penjualan |
+---------------------+
| pelanggan           |
+---------------------+
1 rows in set (0.01 sec)
```

Untuk melihat struktur tabel "**mhs**" secara lebih detail, cobalah perintah atau query sebagai berikut :

```
DESC pelanggan;
```

**DESC** merupakan singkatan dari **DESCRIBE** (dalam query bisa ditulis lengkap atau hanya 4 karakter pertama) dan **pelanggan** adalah nama tabel yang akan dilihat strukturnya. Dari perintah di atas, akan ditampilkan struktur tabel **pelanggan** sebagai berikut :

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id_pelanggan | varchar(5)  | NO   | PRI |         |       |
| nm_pelanggan | varchar(30) | NO   |     |         |       |
| alamat       | text        | YES  |     | NULL    |       |
| telepon      | varchar(20) | YES  |     | NULL    |       |
| email        | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Dari struktur tabel mhs yang ditampilkan di atas, dapat diketahui bahwa :

- Terdapat 5 (lima) field dengan tipe masing-masing.
- Primary Key dari tabel **pelanggan** adalah **id_pelanggan**. Lihat kolom **Key** pada field **id_pelanggan**.
- Untuk field **id_pelanggan** dan **nm_pelanggan** *defaultnya* tidak boleh kosong. Lihatlah kolom **Null** dan **Default** pada field id_pelanggan dan nm_pelanggan.
- Untuk field **alamat**, **telepon** dan **email** *default*-nya boleh kosong. Lihatlah kolom **Null** dan **Default** pada field alamat dan telepon.

## Mengubah Struktur Table dengan ALTER

Untuk mengubah struktur suatu tabel, bentuk umum perintah SQL-nya sebagai berikut :

```
ALTER TABLE nama_tabel alter_options;
```

dimana :

- `ALTER TABLE` merupakan perintah dasar untuk mengubah tabel.
- `nama_tabel` merupakan nama tabel yang akan diubah strukturnya.
- `alter_options` merupakan pilihan perubahan tabel. Option yang bisa digunakan, beberapa di antaranya sebagai berikut :
  - » **ADD** `definisi_field_baru`

    Option ini digunakan untuk menambahkan field baru dengan "**definisi_field_baru**" (nama field, tipe dan option lain).
  - » **ADD INDEX** `nama_index`

Option ini digunakan untuk menambahkan index dengan nama "**nama_index"** pada tabel.

» **ADD PRIMARY KEY** (field_kunci)

Option untuk menambahkan primary key pada tabel

» **CHANGE** field_yang_diubah definisi_field_baru

Option untuk mengubah field_yang_diubah menjadi definisi_field_baru

» **MODIFY** definisi_field

Option untuk mengubah suatu field menjadi definisi_field

» **DROP** nama_field

Option untuk menghapus field nama_field

» **RENAME TO** nama_tabel_baru

Option untuk mengganti nama tabel

Beberapa contoh variasi perintah ALTER untuk mengubah struktur suatu tabel antara lain :

1. Menambahkan field "**tgllahir**" ke tabel **pelanggan**

```
ALTER TABLE pelanggan ADD tgllahir date NOT NULL;
```

2. Menambahkan primary key pada suatu tabel

```
ALTER TABLE pelanggan ADD PRIMARY KEY(id_pelanggan);
```

3. Mengubah **tipe field tgllahir** menjadi varchar dalam tabel **pelanggan**

```
ALTER TABLE pelanggan MODIFY tgllahir varchar(8) NOT
NULL;
```

4. Menghapus field **tgllahir** dari tabel **pelanggan**

```
ALTER TABLE pelanggan DROP tgllahir;
```

## Mengubah Nama Tabel

Untuk mengubah nama suatu tabel, dapat menggunakan perintah SQL sbb :

```
RENAME TABLE pelanggan TO plg;
ALTER TABLE plg RENAME TO pelanggan;
```

Perintah di atas akan mengubah tabel **pelanggan** menjadi **plg** dan sebaliknya.

### Menghapus Tabel

Untuk menghapus sebuah tabel, bentuk umum dari perintah SQL adalah sebagai berikut :

```
DROP TABLE nama_tabel;
```

Contohnya kita akan menghapus tabel dengan nama "**pelanggan**" maka perintah SQL-nya adalah :

```
DROP TABLE pelanggan;
```

## Menambah Record dengan INSERT

Bentuk umum perintah SQL untuk menambahkan *record* atau data ke dalam suatu tabel adalah sebagai berikut :

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

atau dapat dengan bentuk sebagai berikut :

```
INSERT INTO nama_tabel(field1,field2,...)
VALUES ('nilai1','nilai2',...);
```

atau dapat juga dengan bentuk sebagai berikut :

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

Sebagai contoh, kita akan menambahkan sebuah record ke dalam tabel **pelanggan** yang telah kita buat sebelumnya. Berikut ini perintah SQL untuk menambahkan sebuah record ke dalam tabel **pelanggan** :

```
INSERT INTO pelanggan VALUES ('P0001', 'Achmad
Solichin','Jakarta Selatan', '0217327762',
'achmatim@gmail.com');
```

Jika perintah SQL di atas berhasil dieksekusi maka akan ditampilkan pesan sebagai berikut :

```
Query OK, 1 row affected (0.00 sec)
```

Setelah perintah SQL di atas berhasil dieksekusi, maka record atau data dalam tabel pelanggan akan bertambah. Jalankan perintah berikut ini untuk melihat isi tabel pelanggan !

```
SELECT * FROM pelanggan;
```

Dan berikut ini hasil dari perintah SQL di atas :

```
+--------------+-----------------+-----------------+------------+-----
---------------+
| id_pelanggan | nm_pelanggan    | alamat          | telepon    |
email     |
+--------------+-----------------+-----------------+------------+-----
---------------+
| P0001        | Achmad Solichin | Jakarta Selatan | 0217327762 |
achmatim@gmail.com |
+--------------+-----------------+-----------------+------------+-----
---------------+
1 row in set (0.00 sec)
```

## Mengedit Record dengan UPDATE

Proses update bisa sewaktu-waktu dilakukan jika terdapat data atau record dalam suatu tabel yang perlu diperbaiki. Proses update ini tidak menambahkan data (record) baru, tetapi memperbaiki data yang lama. Perubahan yang terjadi dalam proses update bersifat permanen, artinya setelah perintah dijalankan tidak dapat di-*cancel* (*undo*).

Bentuk umum perintah SQL untuk mengedit suatu *record* atau data dari suatu tabel adalah sebagai berikut :

```
UPDATE nama_tabel SET field1='nilaibaru'
[WHERE kondisi];
```

Pada perintah untuk update di atas :

- `UPDATE` merupakan perintah dasar untuk mengubah *record* tabel.
- `nama_tabel` merupakan nama tabel yang akan diubah *recordnya*.
- Perintah `SET` diikuti dengan *field-field* yang akan diubah yang mana diikuti juga dengan perubahan isi dari masing-masing *field*. Untuk mengubah nilai

dari beberapa *field* sekaligus, gunakan koma (,) untuk memisahkan masing-masing *field*.

- Perintah `WHERE` diikuti oleh kondisi tertentu yang menentukan record mana yang akan diedit (diubah). Perintah `WHERE` ini boleh ada boleh juga tidak. Jika `WHERE` tidak ditambahkan pada perintah update maka semua *record* dalam tabel bersangkutan akan berubah.

Perhatikan beberapa contoh perintah UPDATE tabel **pelanggan** berikut ini !

1. Mengubah alamat menjadi "Tangerang" untuk pelanggan yang mempunyai id 'P0001'

```
UPDATE pelanggan SET alamat='Tangerang' WHERE
id_pelanggan='P0001';
```

Dan jika query di atas berhasil dieksekusi maka akan ditampilkan hasil sebagai berikut :

```
Query OK, 1 row affected (0.27 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

2. Mengubah email menjadi "budi@luhur.com" dan alamat menjadi "Bandung" untuk pelanggan yang mempunyai id_pelanggan 'P0002'

```
UPDATE pelanggan SET email='budi@luhur.com',
alamat='Bandung' WHERE id_pelanggan='P0002';
```

## Menghapus Record dengan DELETE

Proses delete dilakukan jika terdapat data atau record dalam suatu tabel yang perlu dihapus atau dihilangkan. Perubahan yang terjadi dalam proses *delete* bersifat permanen, artinya setelah perintah dijalankan tidak dapat di-*cancel* (*undo*). Jadi berhati-hatilah dengan perintah *delete* !

Bentuk umum perintah SQL untuk menghapus suatu *record* atau data dari tabel adalah sebagai berikut :

```
DELETE FROM nama_tabel [WHERE kondisi];
```

Pada perintah untuk *delete* di atas :

- `DELETE FROM` merupakan perintah dasar untuk menghapus suatu *record* dari tabel.
- `nama_tabel` merupakan nama tabel yang akan dihapus *recordnya*.
- Perintah `WHERE` diikuti oleh kondisi tertentu yang menentukan record mana yang akan dihapus (didelete). Perintah `WHERE` ini boleh ada boleh juga tidak. Namun demikian, jika `WHERE` tidak ditambahkan pada perintah delete maka semua *record* dalam tabel bersangkutan akan **terhapus**. Jadi jangan lupa menambahkan WHERE jika kita tidak bermaksud mengosongkan tabel

Perhatikan beberapa contoh perintah DELETE dari tabel **pelanggan** berikut ini !

1. Menghapus data pelanggan yang mempunyai id_pelanggan P0005

```
DELETE FROM pelanggan WHERE id_pelanggan='P0005';
```

Dan jika query di atas berhasil dieksekusi dan record yang akan dihapus ada, maka akan ditampilkan hasil sebagai berikut :

```
Query OK, 1 row affected (0.11 sec)
```

2. Menghapus semua pelanggan yang beralamat di "Bandung"

```
DELETE FROM pelanggan WHERE alamat='Bandung';
```

## Menampilkan Record dengan SELECT

Perintah SELECT digunakan untuk menampilkan sesuatu. Sesuatu di sini bisa berupa sejumlah data dari tabel dan bisa juga berupa suatu ekspresi. Dengan SELECT kita bisa mengatur tampilan atau keluaran sesuai tampilan yang diinginkan.

Bentuk dasar perintah SELECT data dari tabel adalah sebagai berikut :

```
SELECT [field | *] FROM nama_tabel [WHERE kondisi];
```

Perhatikan beberapa contoh perintah SELECT dari tabel **pelanggan** berikut ini !

1. Menampilkan seluruh data atau record (*) dari tabel **pelanggan**

```
SELECT * FROM pelanggan;
```

Dan jika query di atas berhasil dieksekusi maka akan ditampilkan hasil sebagai berikut :

```
+--------------+-----------------+-----------------------------------+--
----------+---------------------+
| id_pelanggan | nm_pelanggan    | alamat                            |
telepon | email               |
+--------------+-----------------+-----------------------------------+--
----------+---------------------+
| P0001        | Achmad Solichin | Jakarta Selatan                   |
0217327762 | achmatim@gmail.com  |
| P0002        | Agus Rahman     | Jl H Said, Tangerang              |
0217323234 | agus20@yahoo.com    |
| P0003        | Doni Damara     | Jl. Raya Cimone, Jakarta Selatan |
0214394379 | damara@yahoo.com    |
| P0004        | Reni Arianti    | Jl. Raya Dago No 90               |
0313493583 | renren@yahoo.co.id  |
| P0005        | Dewi Aminah     | Jl Arjuna No 40                   |
0314584883 | aminahoke@plasa.com |
| P0006        | Chotimatul M    | RT 04 RW 02 Kel Pinang sari       |
0219249349 | fixiz@yahoo.co.id   |
+--------------+-----------------+-----------------------------------+--
----------+---------------------+
6 rows in set (0.00 sec)
```

2. Menampilkan *field* **id_pelanggan** dan **nm_pelanggan** dari seluruh pelanggan dalam tabel **pelanggan**

```
SELECT id_pelanggan, nm_pelanggan FROM pelanggan;
```

Jika query di atas berhasil dieksekusi maka akan ditampilkan hasil sebagai berikut :

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0001        | Achmad Solichin |
| P0002        | Agus Rahman     |
| P0003        | Doni Damara     |
| P0004        | Reni Arianti    |
| P0005        | Dewi Aminah     |
| P0006        | Chotimatul M    |
+--------------+-----------------+
6 rows in set (0.00 sec)
```

3. Menampilkan id, nama dan alamat dari data pelanggan yang mempunyai id **P0006**

```
SELECT id_pelanggan, nm_pelanggan, alamat
FROM pelanggan WHERE id_pelanggan = 'P0006';
```

Hasil query di atas adalah sbb :

```
+--------------+--------------+----------------------------+
| id_pelanggan | nm_pelanggan | alamat                     |
+--------------+--------------+----------------------------+
| P0006        | Chotimatul M | RT 04 RW 02 Kel Pinang sari |
+--------------+--------------+----------------------------+
1 row in set (0.00 sec)
```

4. Menampilkan id, nama dan email data semua pelanggan yang mempunyai email di **yahoo**

```
SELECT id_pelanggan, nm_pelanggan, email
FROM pelanggan WHERE email LIKE '%yahoo%';
```

Hasil query di atas adalah sbb :

```
+--------------+--------------+--------------------+
| id_pelanggan | nm_pelanggan | email              |
+--------------+--------------+--------------------+
| P0002        | Agus Rahman  | agus20@yahoo.com   |
| P0003        | Doni Damara  | damara@yahoo.com   |
| P0004        | Reni Arianti | renren@yahoo.co.id |
| P0006        | Chotimatul M | fixiz@yahoo.co.id  |
+--------------+--------------+--------------------+
4 rows in set (0.00 sec)
```

Berikut ini operator **perbandingan** yang dapat digunakan untuk membandingkan dua buah nilai dalam MySQL :

- **Operator =**, akan bernilai TRUE jika nilai yang dibandingkan sama.
- **Operator !=** atau **<>**, akan bernilai TRUE jika nilai yang dibandingkan TIDAK SAMA (berbeda).
- **Operator >**, akan bernilai TRUE jika nilai yang pertama lebih besar dari nilai kedua.
- **Operator >=**, akan bernilai TRUE jika nilai yang pertama lebih besar atau sama dengan nilai kedua.
- **Operator <**, akan bernilai TRUE jika nilai yang pertama lebih kecil dari nilai kedua.

- **Operator <=**, akan bernilai TRUE jika nilai yang pertama lebih kecil atau sama dengan nilai kedua.

5. Menampilkan data semua pelanggan yang beralamat di **Jakarta Selatan** dan mempunyai email di **gmail**.

```
SELECT id_pelanggan, nm_pelanggan, alamat, email
FROM pelanggan WHERE alamat = 'Jakarta Selatan' &&
email LIKE '%gmail.com';
```

Hasil query di atas adalah sbb :

```
+--------------+-----------------+-----------------+--------------------+
| id_pelanggan | nm_pelanggan    | alamat          | email              |
+--------------+-----------------+-----------------+--------------------+
| P0001        | Achmad Solichin | Jakarta Selatan | achmatim@gmail.com |
+--------------+-----------------+-----------------+--------------------+
1 row in set (0.00 sec)
```

Berikut ini operator **penghubung** yang dapat digunakan untuk menghubungkan antara dua kondisi dalam MySQL :

- **Operator &&** atau **AND**, akan menghubungkan dua kondisi dimana akan bernilai TRUE jika kedua kondisi bernilai TRUE.
- **Operator ||** atau **OR**, akan menghubungkan dua kondisi dimana akan bernilai TRUE jika salah satu atau kedua kondisi bernilai TRUE.
- **Operator !**, akan me-*reverse* nilai suatu kondisi logika.

6. Menampilkan semua data pelanggan secara urut berdasarkan **nama pelanggan** dengan perintah **ORDER BY**

```
SELECT id_pelanggan, nm_pelanggan
FROM pelanggan ORDER BY nm_pelanggan;
```

Hasil query di atas adalah sbb :

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0001        | Achmad Solichin |
| P0002        | Agus Rahman     |
| P0006        | Chotimatul M    |
| P0005        | Dewi Aminah     |
| P0003        | Doni Damara     |
| P0004        | Reni Arianti    |
+--------------+-----------------+
6 rows in set (0.00 sec)
```

7. Menampilkan semua data pelanggan secara urut berdasarkan **nama pelanggan** secara **DESCENDING**

```
SELECT id_pelanggan, nm_pelanggan
FROM pelanggan ORDER BY nm_pelanggan DESC;
```

Hasil query di atas adalah sbb :

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0004        | Reni Arianti    |
| P0003        | Doni Damara     |
| P0005        | Dewi Aminah     |
| P0006        | Chotimatul M    |
| P0002        | Agus Rahman     |
| P0001        | Achmad Solichin |
+--------------+-----------------+
6 rows in set (0.00 sec)
```

8. Menampilkan 3 record (data) pertama dari tabel **pelanggan** secara urut berdasarkan **nama pelanggan** dengan **LIMIT**

```
SELECT id_pelanggan, nm_pelanggan
FROM pelanggan ORDER BY nm_pelanggan LIMIT 0,3;
```

Hasil query di atas adalah sbb :

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0001        | Achmad Solichin |
| P0002        | Agus Rahman     |
| P0006        | Chotimatul M    |
+--------------+-----------------+
3 rows in set (0.00 sec)
```

**Keterangan**

Pada query di atas bentuk **LIMIT** digunakan untuk membatasi hasil tampilan. LIMIT banyak digunakan untuk menampilkan data yang relatif banyak. Format fungsi LIMIT adalah sebagai berikut :

```
LIMIT awal, jumlah_record
```

9. Menampilkan jumlah record yang ada di tabel **pelanggan**.

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

Hasil query di atas adalah sbb :

```
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

# Bab 5
# Fungsi-fungsi MySQL

- Fungsi String
- Fungsi Tanggal dan Waktu
- Fungsi Numerik
- Fungsi Lainnya

## Fungsi String

MySQL memiliki banyak sekali fungsi yang berhubungan dengan operasi String. Berikut ini beberapa fungsi string yang disediakan MySQL.

### CONCAT (str1, str2, ...)

Fungsi ini digunakan untuk menggabungkan dua atau lebih string (kolom). Sebagai contoh, misalnya akan menggabungkan kolom **alamat** dan **telepon** pada tabel **pelanggan** menjadi sebuah kolom:

```
SELECT nm_pelanggan, CONCAT(alamat,' ',telepon)
FROM pelanggan;
```

Hasil keluarannya:

```
+-----------------+-----------------------------------------+
| nm_pelanggan    | concat(alamat,' ',telepon)              |
+-----------------+-----------------------------------------+
| Achmad Solichin | Jakarta Selatan 0217327762              |
| Agus Rahman     | Jl H Said, Tangerang 0217323234         |
| Doni Damara     | Jl. Raya Cimone, Jakarta Selatan 0214394379 |
| Reni Arianti    | Jl. Raya Dago No 90 0313493583          |
| Dewi Aminah     | Jl Arjuna No 40 0314584883              |
| Chotimatul M    | RT 04 RW 02 Kel Pinang sari 0219249349  |
+-----------------+-----------------------------------------+
```

### CONCAT_WS (separator, str1, str2, ...)

Fungsi ini digunakan untuk menggabungkan dua atau lebih string (kolom) dengan separator diantara masing-masing string. Contoh:

```
SELECT CONCAT_WS (',','Adi','Ida','Edi');
```

Hasil keluarannya:

```
Adi, Ida, Edi
```

**SUBSTR (string, awal, panjang)**

**SUBSTRING (string, awal, panjang)**

**SUBSTRING (string FROM awal FOR panjang)**

**MID (string, awal, panjang)**

Fungsi ini digunakan untuk mengambil atau memotong string dimulai dari karakter **awal** sebanyak **panjang** karakter. Sebagai catatan bahwa di MySQL, index string dimulai dengan 1, bukan 0. Contoh:

```
SELECT SUBSTRING ('Budi Luhur',1,4);
```

Hasil keluarannya:

```
Budi
```

**LENGTH (string)**
**OCTET_LENGTH (string)**
**CHAR_LENGTH (string)**
**CHARACTER_LENGTH (string)**

Fungsi ini digunakan untuk menghitung panjang suatu string. Contoh:

```
SELECT LENGTH ('Budi Luhur');
```

Hasil keluarannya:

```
5
```

**LEFT (string, panjang)**

Fungsi ini digunakan untuk memotong string dari sebelah kiri sebanyak **panjang** karakter. Contoh:

```
SELECT LEFT ('Budi Luhur', 4);
```

Hasil keluarannya:

```
Budi
```

**RIGHT (string, panjang)**

Fungsi ini digunakan untuk memotong string dari sebelah kanan sebanyak **panjang** karakter. Contoh:

```
SELECT RIGHT ('Budi Luhur', 4);
```

Hasil keluarannya:

```
Uhur
```

### LTRIM (string)

Fungsi ini digunakan untuk menghilangkan spasi di awal string (kiri). Contoh:

```
SELECT LTRIM (' Budi Luhur');
```

Hasil keluarannya:

```
Budi Luhur
```

### RTRIM (string)

Fungsi ini digunakan untuk menghilangkan spasi di akhir string (kanan). Contoh:

```
SELECT RTRIM ('Budi Luhur ');
```

Hasil keluarannya:

```
Budi Luhur
```

### TRIM (string)

Fungsi ini digunakan untuk menghilangkan spasi di awal dan akhir string (kiri dan kanan). Contoh:

```
SELECT TRIM (' Budi Luhur ');
```

Hasil keluarannya:

```
Budi Luhur
```

### REPLACE (string, from_str, to_str)

Fungsi ini digunakan untuk mengganti suatu string dengan string yang lain. Contoh:

```
SELECT REPLACE ('www.mysql.com', 'w', 'x' );
```

Hasil keluarannya:

```
xxx.mysql.com
```

**REPEAT (string, jumlah)**

Fungsi ini digunakan untuk menduplikasi suatu string sebanyak **jumlah**. Contoh:

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

Hasil keluarannya:

```
MontMontMont
```

**REVERSE (string)**

Fungsi ini digunakan untuk membalik string. Contoh:

```
SELECT REVERSE ('mysql.com');
```

Hasil keluarannya:

```
moc.lqsym
```

**LCASE (string)**

**LOWER (string)**

Fungsi ini digunakan untuk mengubah string menjadi huruf kecil (lower-case). Contoh:

```
SELECT LOWER ('MySQL');
```

Hasil keluarannya:

```
Mysql
```

**UCASE (string)**

### UPPER (string)

Fungsi ini digunakan untuk mengubah string menjadi huruf kapital (upper-case). Contoh:

```
SELECT UPPER ('mysql');
```

Hasil keluarannya:

```
MYSQL
```

## Fungsi Tanggal dan Waktu

Selain fungsi string, MySQL juga memiliki fungsi-fungsi yang berhubungan dengan operasi tanggal dan waktu (date and time). Berikut ini beberapa fungsi tanggal dan waktu yang disediakan MySQL.

### NOW ()
### SYSDATE()

Fungsi ini digunakan untuk mendapatkan tanggal dan waktu sistem sekarang Contoh:

```
SELECT NOW();
```

Hasil keluarannya:

```
2008-02-19 20:00:31
```

### MONTH (tanggal)

Fungsi ini digunakan untuk mendapatkan urutan bulan (integer) dari suatu tanggal yang diberikan dalam setahun, dimana 1=Januari, 2=Februari, dst. Contoh:

```
SELECT MONTH ('1982-06-05');
```

Hasil keluarannya:

```
6
```

### WEEK (tanggal)

Fungsi ini digunakan untuk mendapatkan urutan minggu (integer) dari suatu tanggal yang diberikan dalam setahun. Contoh:

```
SELECT WEEK ('1982-06-05');
```

Hasil keluarannya:

```
22
```

### YEAR (tanggal)

Fungsi ini digunakan untuk mendapatkan bilangan tahun dari suatu tanggal yang diberikan. Contoh:

```
SELECT YEAR (now());
```

Hasil keluarannya:

```
2008
```

### HOUR (waktu)

Fungsi ini digunakan untuk mendapatkan bilangan jam dari suatu parameter waktu yang diberikan. Contoh:

```
SELECT HOUR (now());
```

Hasil keluarannya:

```
20
```

### MINUTE (waktu)

Fungsi ini digunakan untuk mendapatkan bilangan menit dari suatu parameter waktu yang diberikan. Contoh:

```
SELECT MINUTE (now());
```

Hasil keluarannya:

```
8
```

### SECOND (waktu)

Fungsi ini digunakan untuk mendapatkan bilangan detik dari suatu waktu yang diberikan. Contoh:

```
SELECT SECOND (now());
```

Hasil keluarannya:

```
53
```

**DATE_ADD(date,INTERVAL expr type)**
**DATE_SUB(date,INTERVAL expr type)**
**ADDDATE(date,INTERVAL expr type)**
**SUBDATE(date,INTERVAL expr type)**

Fungsi-fungsi diatas digunakan untuk menambah suatu tanggal. Contoh:

```
SELECT DATE_ADD(now(), INTERVAL 1 DAY);
```

Hasil keluarannya:

```
2008-02-20 20:12:17
```

**DATE_FORMAT(date, format)**

Fungsi ini digunakan untuk mem-format tampilan tanggal.

**TIME_FORMAT(time, format)**

Fungsi ini digunakan untuk mem-format tampilan waktu.

Berikut ini format tampilan tanggal dan waktu, dan penggunaannya:

%M : Nama bulan (January ... December)
%W : Nama hari dalam seminggu (Sunday...Saturday)
%D : Urutan hari dalam sebulan
%Y : Tahun, 4 digit
%y : Tahun, 2 digit
%a : Nama hari dalam seminggu (Sun...Saturday)
%H : Jam, dalam format 24.
%i : Menit, 00-59
%s : Detik, 00-59

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

Hasil keluarannya:

```
20-02-2008 20:12:17
```

# Fungsi Numerik

MySQL memiliki fungsi-fungsi yang berhubungan dengan operasi numerik, berikut ini contohnya:

### OPERASI ARITMATIKA

Operasi aritmatika dalam MySQL terdiri dari:

+ : Pertambahan
- : Pengurangan
* : Perkalian
/ : Pembagian
% : Sisa hasil bagi, modulus

Contoh penggunaan:

```
SELECT 10+20;
```

Hasil keluarannya:

```
30
```

```
SELECT 10/3;
```

Hasil keluarannya:

```
3.3333
```

### ABS(x)

Fungsi digunakan untuk mengambil nilai absolut dari bilangan x. Contoh:

```
SELECT ABS(-20);
```

Hasil keluarannya:

```
20
```

### MOD(m, n)

Fungsi digunakan untuk mengoperasikan m modulus n. Contoh:

```
SELECT MOD(10,3);
```

Hasil keluarannya:

```
1
```

### FLOOR(x)

Fungsi digunakan untuk mengambil nilai integer terbesar yang tidak lebih besar dari x. Contoh:

```
SELECT FLOOR(10.3576);
```

Hasil keluarannya:

```
10
```

### CEILING(x)

Fungsi digunakan untuk mengambil nilai integer terkecil yang tidak lebih kecil dari x. Contoh:

```
SELECT CEILING(10.3576);
```

Hasil keluarannya:

```
11
```

### ROUND(x)
### ROUND(x, d)

Fungsi digunakan untuk melakukan pembulatan bilangan x sebanyak d tempat presisi. Contoh:

```
SELECT ROUND(10.3576, 2);
```

Hasil keluarannya:

```
10.36
```

**POW(x)**
**POWER(x, n)**

Fungsi digunakan untuk melakukan mengambil hasil pemangkatan dari $x^n$. Contoh:

```
SELECT POW(2, 10);
```

Hasil keluarannya:

```
1024
```

**RAND()**
**RAND(x)**

Fungsi digunakan untuk mengambil nilai random diantara 0 s/d 1.0. Contoh:

```
SELECT RAND();
```

Hasil keluarannya:

```
0.96589817662341
```

**TRUNCATE(x, d)**

Fungsi digunakan untuk memotong bilangan x sepanjang d tempat desimal. Contoh:

```
SELECT TRUNCATE(10.28372, 1);
```

Hasil keluarannya:

```
10.2
```

# Fungsi Lainnya

Selain fungsi yang berhubungan dengan string, date-and-time, dan numerik MySQL juga memiliki fungsi-fungsi khusus, diantaranya :

### GREATEST(nil1, nil2, ...)

Fungsi digunakan untuk mengambil nilai terbesar dari suatu kumpulan nilai. Contoh:

```
SELECT GREATEST(2,5,2,6,3,7,4,2,5,1);
```

Hasil keluarannya:

```
7
```

### COUNT(range)

Fungsi digunakan untuk mengambil jumlah baris dari suatu query. Contoh:

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

Hasil keluarannya:

```
5
```

### MAX(range)

Fungsi digunakan untuk mengambil nilai terbesar dari suatu ekspresi (query). Contoh:

```
SELECT MAX(nilai) FROM nilai_ujian;
```

Hasil keluarannya:

```
93
```

### MIN(range)

Fungsi digunakan untuk mengambil nilai terkecil dari suatu ekspresi (query). Contoh:

```
SELECT MIN(nilai) FROM nilai_ujian;
```

Hasil keluarannya:

```
40
```

### SUM(range)

Fungsi digunakan untuk menjumlahkan total nilai dari suatu ekspresi (query). Contoh:

```
SELECT SUM(nilai) FROM nilai_ujian;
```

Hasil keluarannya:

```
450
```

### AVG(range)

Fungsi digunakan untuk menghitung rata-rata nilai dari suatu ekspresi (query). Contoh:

```
SELECT AVG(nilai) FROM nilai_ujian;
```

Hasil keluarannya:

```
78
```

### OPERASI BITWISE

Operasi bitwise dalam MySQL terdiri dari:

| | |
|---|---|
| \| | : Bitwise OR |
| & | : Bitwise AND |
| << | : Shift Kiri |
| >> | : Shift Kanan |
| ~ | : Invert, negasi |

Contoh penggunaan:

```
SELECT 4 | 2;
```

Hasil keluarannya:

```
6
```

### DATABASE()

Fungsi digunakan untuk mengambil nama database yang sedang aktif (terbuka). Contoh:

```
SELECT DATABASE();
```

Hasil keluarannya:

```
Penjualan
```

### USER()
### SYSTEM_USER()
### SESSION_USER()

Fungsi digunakan untuk mengambil user yang sedang digunakan (aktif). Contoh:

```
SELECT USER();
```

Hasil keluarannya:

```
root@localhost
```

### PASSWORD(str)

Fungsi digunakan untuk melakukan enkripsi suatu string. Sifat utama dari fungsi password() ini adalah hasilnya selalu sama untuk setiap string yang sama. String hasil dari fungsi password() tidak dapat di-decript (decode). Biasanya fungsi ini digunakan untuk menyimpan password login. Contoh:

```
SELECT PASSWORD('qwerty');
```

Hasil keluarannya:

```
*AA1420F182E88B9E5F874F6FBE7459291E8F4601
```

### ENCODE(str, pass)

Fungsi digunakan untuk melakukan enkripsi suatu string **str** menggunakan password atau key **pass**. Contoh:

```
SELECT ENCODE('qwerty', 'password');
```

Hasil keluarannya:

```
câ┬♠e|
```

### DECODE(encripted_str, pass)

Fungsi digunakan untuk melakukan dekripsi suatu string **encripted_str** menggunakan password atau key **pass**. Jika passwordnya benar, maka string aslinya akan benar. Contoh:

```
SELECT DECODE('câ┬♠e|', 'password');
```

Hasil keluarannya:

```
Qwerty
```

Contoh dengan password salah:

```
SELECT DECODE('câ┬♠e|', 'ngasal');
```

Hasil keluarannya:

```
WkΦPH:
```

### MD5(str)

Fungsi digunakan untuk melakukan enkripsi suatu string **str** menggunakan metode md5. Fungsi ini juga tidak dapat didekripsi. Contoh:

```
SELECT MD5('qwerty');
```

Hasil keluarannya:

```
d8578edf8458ce06fbc5bb76a58c5ca4
```

### LAST_INSERT_ID()

Fungsi digunakan untuk mengambil id terakhir dalam proses insert dimana tabelnya mengandung field yang bersifat AUTO INCREMENT. Contoh:

```
SELECT LAST_INSERT_ID();
```

Hasil keluarannya:

```
231
```

### VERSION()

Fungsi digunakan untuk mengambil versi MySQL yang digunakan. Contoh:

```
SELECT VERSION();
```

Hasil keluarannya:

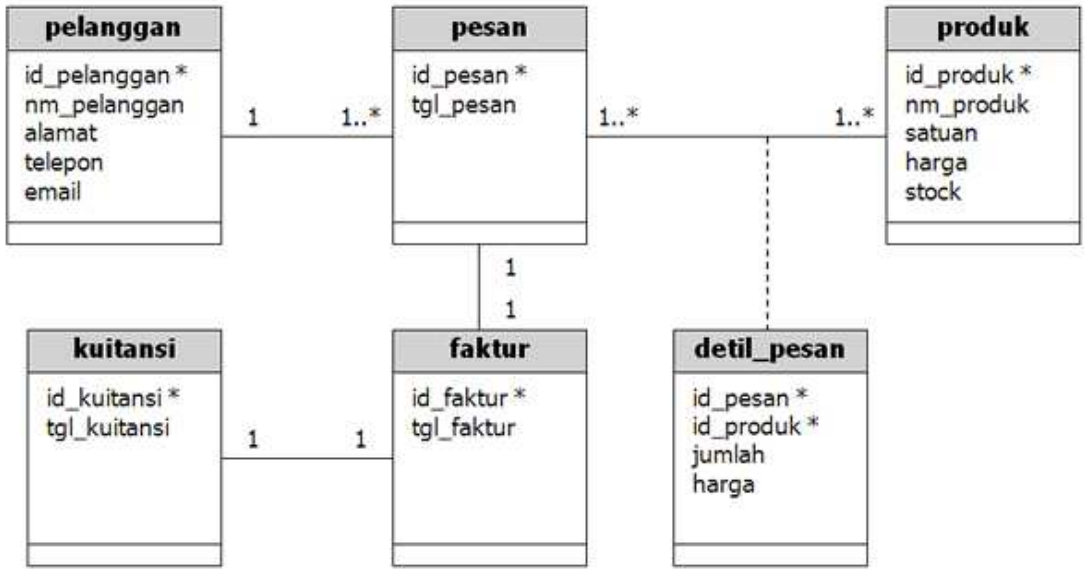
```
5.0.45-community-nt
```

# Bagian 3
# Perintah MySQL Lanjutan

# Bab 6
# Perintah MySQL Lanjutan

- Perintah SELECT dari Banyak Tabel
- Pengelompokkan Hasil Query dengan GROUP BY
- HAVING
- SubSELECT
- Menampilkan Record secara Random
- Transaksi

Pada bab ini akan dijelaskan beberapa perintah SQL lanjutan yang lebih kompleks seperti join antar tabel, grouping, sub select, random search dan transaksi (commit-and-roolback). Untuk mempermudah penjelasan, maka semua contoh yang disajikan di bab ini mengacu pada **pemodelan data konseptual** Sistem Pemesanan (Penjualan) Barang sbb:




Gambar 6.1. Pemodelan Data Konseptual

Untuk membuat tabel-tabel dari rancangan di atas, kita akan menggunakan tipe tabel **InnoDB** karena nantinya kita akan menggunakan transaksi di sistem tersebut. Dan berikut ini **spesifikasi basis data** dari pemodelan data konseptual di atas:

```
Tabel pelanggan
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| id_pelanggan | varchar(5)  | NO   | PRI |         |       |
```

```
| nm_pelanggan | varchar(40) | NO   |     |         |       |
| alamat       | text        | NO   |     |         |       |
| telepon      | varchar(20) | NO   |     |         |       |
| email        | varchar(50) | NO   |     |         |       |
+--------------+-------------+------+-----+---------+-------+

Tabel pesan
+--------------+------------+------+-----+---------+----------------+
| Field        | Type       | Null | Key | Default | Extra          |
+--------------+------------+------+-----+---------+----------------+
| id_pesan     | int(5)     | NO   | PRI | NULL    | auto_increment |
| id_pelanggan | varchar(5) | NO   | MUL |         |                |
| tgl_pesan    | date       | NO   |     |         |                |
+--------------+------------+------+-----+---------+----------------+

Tabel produk
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| id_produk | varchar(5)    | NO   | PRI |         |       |
| nm_produk | varchar(30)   | NO   |     |         |       |
| satuan    | varchar(10)   | NO   |     |         |       |
| harga     | decimal(10,0) | NO   |     | 0       |       |
| stock     | int(3)        | NO   |     | 0       |       |
+-----------+---------------+------+-----+---------+-------+

Tabel detil_pesan
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| id_pesan  | int(5)        | NO   | PRI |         |       |
| id_produk | varchar(5)    | NO   | PRI |         |       |
| jumlah    | int(5)        | NO   |     | 0       |       |
| harga     | decimal(10,0) | NO   |     | 0       |       |
+-----------+---------------+------+-----+---------+-------+

Tabel faktur
+------------+--------+------+-----+---------+----------------+
| Field      | Type   | Null | Key | Default | Extra          |
+------------+--------+------+-----+---------+----------------+
| id_faktur  | int(5) | NO   | PRI | NULL    | auto_increment |
| id_pesan   | int(5) | NO   |     |         |                |
| tgl_faktur | date   | NO   |     |         |                |
+------------+--------+------+-----+---------+----------------+

Tabel kuitansi
+--------------+--------+------+-----+---------+----------------+
| Field        | Type   | Null | Key | Default | Extra          |
+--------------+--------+------+-----+---------+----------------+
| id_kuitansi  | int(5) | NO   | PRI | NULL    | auto_increment |
| id_faktur    | int(5) | NO   |     |         |                |
| tgl_kuitansi | date   | NO   |     |         |                |
+--------------+--------+------+-----+---------+----------------+
```

Berikut ini disajikan perintah SQL untuk membuat tabel-tabel di atas:

```
/*Table structure for table detil_pesan */
DROP TABLE IF EXISTS detil_pesan;
CREATE TABLE detil_pesan (
  id_pesan int(5) NOT NULL,
  id_produk varchar(5) NOT NULL,
  jumlah int(5) NOT NULL default '0',
  harga decimal(10,0) NOT NULL default '0',
  PRIMARY KEY  (id_pesan,id_produk),
  KEY FK_detil_pesan (id_produk),
```

```
  KEY id_pesan (id_pesan),
  CONSTRAINT FK_detil_pesan FOREIGN KEY (id_produk)
  REFERENCES produk (id_produk),
  CONSTRAINT FK_detil_pesan2 FOREIGN KEY (id_pesan)
  REFERENCES pesan (id_pesan)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*Table structure for table faktur */
DROP TABLE IF EXISTS faktur;
CREATE TABLE faktur (
  id_faktur int(5) NOT NULL auto_increment,
  id_pesan int(5) NOT NULL,
  tgl_faktur date NOT NULL,
  PRIMARY KEY  (id_faktur),
  KEY id_pesan (id_pesan),
  CONSTRAINT faktur_ibfk_1 FOREIGN KEY (id_pesan)
  REFERENCES pesan (id_pesan)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Table structure for table kuitansi */
DROP TABLE IF EXISTS kuitansi;
CREATE TABLE kuitansi (
  id_kuitansi int(5) NOT NULL auto_increment,
  id_faktur int(5) NOT NULL,
  tgl_kuitansi date NOT NULL,
  PRIMARY KEY  (id_kuitansi),
  KEY FK_kuitansi (id_faktur),
  CONSTRAINT FK_kuitansi FOREIGN KEY (id_faktur)
  REFERENCES faktur (id_faktur)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Table structure for table pelanggan */
DROP TABLE IF EXISTS pelanggan;
CREATE TABLE pelanggan (
  id_pelanggan varchar(5) NOT NULL,
  nm_pelanggan varchar(40) NOT NULL,
  alamat text NOT NULL,
  telepon varchar(20) NOT NULL,
  email varchar(50) NOT NULL,
  PRIMARY KEY  (id_pelanggan)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 CHECKSUM=1
DELAY_KEY_WRITE=1 ROW_FORMAT=DYNAMIC;

/*Table structure for table pesan */
DROP TABLE IF EXISTS pesan;
CREATE TABLE pesan (
  id_pesan int(5) NOT NULL auto_increment,
  id_pelanggan varchar(5) NOT NULL,
  tgl_pesan date NOT NULL,
  PRIMARY KEY  (id_pesan),
  KEY id_pelanggan (id_pelanggan),
  CONSTRAINT pesan_ibfk_1 FOREIGN KEY (id_pelanggan)
  REFERENCES pelanggan (id_pelanggan)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

/*Table structure for table produk */
DROP TABLE IF EXISTS produk;
CREATE TABLE produk (
  id_produk varchar(5) NOT NULL,
  nm_produk varchar(30) NOT NULL,
  satuan varchar(10) NOT NULL,
```

```
  harga decimal(10,0) NOT NULL default '0',
  stock int(3) NOT NULL default '0',
  PRIMARY KEY  (id_produk)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Selanjutnya, untuk memudahkan dalam pemberian contoh, isilah tabel-tabel diatas dengan record (data) secukupnya.

# Perintah SELECT dari Banyak Tabel dengan JOIN

Di dalam suatu RDBMS tentunya sudah menjadi suatu kewajaran jika dalam satu database dapat terdiri dari beberapa tabel. Masing-masing tabel tersebut berhubungan satu sama lain atau dengan kata lain memiliki relasi. Relasi antar-tabel dapat berupa relasi 1-1, 1-M, atau M-N. Sebagai contoh terlihat pada gambar pemodelan data konseptual (class diagram) di atas. Tabel **pelanggan** berhubungan dengan **pesan**, **pesan** dengan **barang**, dsb.

Pada praktisnya, terkadang kita juga memerlukan tampilan data yang tidak hanya berasal dari 1 (satu) tabel, namun bisa dari beberapa tabel sekaligus. Contohnya, dari class diagram diatas, kita ingin menampilkan nama pelanggan berikut transaksi yang pernah dilakukannya. Dari contoh tersebut, kita harus bisa menggabungkan minimal dua tabel, yaitu **pelanggan** dan **pesan**.

Untuk menggabungkan 2 (dua) atau lebih tabel, kita dapat menggunakan bentuk perintah JOIN.

## Inner Join

Dengan inner join, tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi. Sebagai contoh, kita akan menggabungkan tabel **pelanggan** dan **pesan** dimana kita akan menampilkan daftar pelanggan yang pernah melakukan pemesanan (transaksi). Isi tabel pelanggan dan pesan adalah sebagai berikut :

Tabel **pelanggan** (hanya ditampilkan id, nama dan email).

```
+--------------+----------------+--------------------+
| id_pelanggan | nm_pelanggan   | email              |
+--------------+----------------+--------------------+
| P0001        | Achmad Solichin | achmatim@gmail.com |
| P0002        | Budianto       | budi@luhur.com     |
| P0003        | Hasan          | hasan02@yahoo.com  |
```

```
| P0004        | Amin Riyadi     | aminudin@plasa.com |
+--------------+-----------------+--------------------+
```

Tabel **pesan** :

```
+----------+--------------+------------+
| id_pesan | id_pelanggan | tgl_pesan  |
+----------+--------------+------------+
|        1 | P0001        | 2008-02-02 |
|        2 | P0002        | 2008-02-05 |
|        3 | P0002        | 2008-02-10 |
|        4 | P0004        | 2008-01-20 |
|        5 | P0001        | 2007-12-14 |
+----------+--------------+------------+
```

### *Cara 1 : Penggabungan dengan WHERE*

Bentuk umum

```
SELECT tabel1.*, tabel2.* FROM tabel1, tabel2
WHERE tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan, pesan
WHERE pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
+--------------+-----------------+----------+------------+
```

Pada hasil perintah query di atas terlihat bahwa terdapat 5 (lima) transaksi yang dilakukan oleh 3 (tiga) orang pelanggan. Jika kita lihat kembali isi tabel pelanggan di atas, maka terdapat satu pelanggan yang tidak ditampilkan yaitu yang memiliki id pelanggan P0003. Pelanggan tersebut tidak ditampilkan karena belum pernah melakukan transaksi.

### *Cara 2 : Penggabungan dengan INNER JOIN*

Bentuk umum

```
SELECT tabel1.*, tabel2.*
FROM tabel1 INNER JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan INNER JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
+--------------+-----------------+----------+------------+
```

## Outer Join

Dengan outer join, tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang NULL (kosong) di satu sisi. Sebagai contoh, kita akan menggabungkan tabel **pelanggan** dan **pesan** dimana kita akan menampilkan daftar pelanggan yang pernah melakukan pemesanan (transaksi).

Outer Join terbagi menjadi 2 (dua) yaitu LEFT JOIN dan RIGHT. Berikut ini bentuk umum dan contohnya:

### *LEFT JOIN*

Bentuk umum

```
SELECT tabel1.*, tabel2.*
FROM tabel1 LEFT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan LEFT JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0003        | Hasan           |     NULL | NULL       |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
+--------------+-----------------+----------+------------+
```

Berbeda dengan hasil sebelumnya (inner join), penggunaan left join akan menampilkan juga data pelanggan dengan id P0003, walaupun pelanggan tersebut belum pernah bertransaksi. Dan pada kolom id_pesan dan tgl_pesan untuk pelanggan P0003 isinya NULL, artinya di tabel kanan (pesan) pelanggan tersebut tidak ada.

### *RIGHT JOIN*

Bentuk umum

```
SELECT tabel1.*, tabel2.*
FROM tabel1 RIGHT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan RIGHT JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+--------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+--------------+-----------------+----------+------------+
| P0001        | Achmad Solichin |        1 | 2008-02-02 |
| P0002        | Budianto        |        2 | 2008-02-05 |
| P0002        | Budianto        |        3 | 2008-02-10 |
| P0004        | Amin Riyadi     |        4 | 2008-01-20 |
| P0001        | Achmad Solichin |        5 | 2007-12-14 |
```

```
+--------------+----------------+----------+------------+
```

Dengan right join, tabel yang menjadi acuan adalah tabel sebelah kanan (tabel pesan), jadi semua isi tabel pesan akan ditampilkan. Jika data pelanggan tidak ada di tabel pelanggan, maka isi tabel pesan tetap ditampilkan.

### Menggabungkan Tiga Tabel

Untuk menggabungkan tiga tabel atau lebih, pada dasarnya sama dengan penggabungan 2 (dua) tabel. Sebagai contoh misalnya kita akan menampilkan barang-barang yang dipesan beserta nama barang dan harganya untuk pemesanan dengan nomor **1**. Berikut ini perintah SQL-nya:

```
SELECT pesan.id_pesan, produk.id_produk, produk.nm_produk,
detil_pesan.harga, detil_pesan.jumlah
FROM pesan, detil_pesan, produk
WHERE pesan.id_pesan=detil_pesan.id_pesan AND
detil_pesan.id_produk=produk.id_produk
AND pesan.id_pesan='1'
```

Hasilnya :

```
+----------+-----------+------------+-------+--------+
| id_pesan | id_produk | nm_produk  | harga | jumlah |
+----------+-----------+------------+-------+--------+
|        1 | B0001     | Buku Tulis |  2700 |      2 |
|        1 | B0003     | Penggaris  |  3000 |      3 |
|        1 | B0004     | Pensil     |  2000 |      1 |
+----------+-----------+------------+-------+--------+
```

## Pengelompokkan Hasil Query dengan GROUP BY

Hasil query terkadang perlu dikelompokkan berdasarkan kriteria atau kondisi tertentu. Misalnya kita akan menampilkan jumlah barang yang dibeli untuk masing-masing transaksi (pemesanan).

Perhatikan perintah query berikut ini dan lihat hasilnya:

```
SELECT pesan.id_pesan, pesan.tgl_pesan,
detil_pesan.jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan;
```

Hasilnya :

```
+----------+------------+--------+
| id_pesan | tgl_pesan  | jumlah |
+----------+------------+--------+
|        1 | 2008-02-02 |      2 |
|        1 | 2008-02-02 |      3 |
|        1 | 2008-02-02 |      1 |
|        2 | 2008-02-05 |      1 |
|        2 | 2008-02-05 |      5 |
|        2 | 2008-02-05 |      1 |
|        3 | 2008-02-10 |      5 |
|        4 | 2008-01-20 |     10 |
+----------+------------+--------+
```

Jika kita perhatikan hasil perintah query di atas, kita akan mendapatkan jumlah barang yang terjadi untuk setiap transaksi, namun hasil tampilannya masih per-barang. Artinya jumlah yang ditampilkan masih berupa jumlah barang untuk masing-masing barang.

Agar jumlah barang ditampilkan per-transaksi (pemesanan), maka kita dapat menggunakan fungsi GROUP BY dan juga SUM untuk menjumlahkan jumlah barang. Berikut ini perintah query dengan group by dan count.

```
SELECT pesan.id_pesan, pesan.tgl_pesan,
SUM(detil_pesan.jumlah) as jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY id_pesan;
```

Hasilnya :

```
+----------+------------+--------+
| id_pesan | tgl_pesan  | jumlah |
+----------+------------+--------+
|        1 | 2008-02-02 |      6 |
|        2 | 2008-02-05 |      7 |
|        3 | 2008-02-10 |      5 |
|        4 | 2008-01-20 |     10 |
+----------+------------+--------+
```

Selain hasil di atas, kita juga dapat menggunakan tambahan WITH ROLLUP di belakang group by untuk menampilkan jumlah total seluruh barang. Berikut ini perintah query dan hasilnya:

```
SELECT pesan.id_pesan, pesan.tgl_pesan,
SUM(detil_pesan.jumlah) as jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY id_pesan WITH ROLLUP;
```

Hasilnya :

```
+----------+------------+--------+
| id_pesan | tgl_pesan  | jumlah |
+----------+------------+--------+
|        1 | 2008-02-02 |      6 |
|        2 | 2008-02-05 |      7 |
|        3 | 2008-02-10 |      5 |
|        4 | 2008-01-20 |     10 |
|     NULL | 2008-01-20 |     28 |
+----------+------------+--------+
```

# HAVING

Perintah query berikut ini akan menampilkan jumlah item (jenis) barang untuk tiap transaksi.

```
SELECT pesan.id_pesan, COUNT(detil_pesan.id_produk) as
jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY pesan.id_pesan
```

Hasilnya :

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        1 |      3 |
|        2 |      3 |
|        3 |      1 |
|        4 |      1 |
+----------+--------+
```

Dari hasil query di atas tampak bahwa ditampilkan jumlah item barang untuk semua transaksi. Selanjutnya bagaimana jika kita ingin hanya menampilkan data yang jumlah item barangnya lebih dari 2 (dua)? Mungkin kita langsung berfikir untuk menggunakan WHERE seperti perintah query sebagai berikut:

```
SELECT pesan.id_pesan, COUNT(detil_pesan.id_produk) as
jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
AND jumlah > 2
GROUP BY pesan.id_pesan
```

Hasilnya ternyata tidak sesuai yang diinginkan. Lihat hasilnya sebagai berikut:

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        1 |      1 |
|        2 |      1 |
|        3 |      1 |
|        4 |      1 |
+----------+--------+
```

Hal tersebut terjadi karena kondisi dalam WHERE tidak dapat diterapkan pada fungsi agregrasi seperti COUNT, SUM, AVG dll.

Untuk menyeleksi suatu fungsi agregasi, kita tidak dapat menggunakan WHERE, namun kita dapat menggunakan HAVING. Berikut ini perintah query yang menggunakan HAVING:

```
SELECT pesan.id_pesan, COUNT(detil_pesan.id_produk) as
jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY pesan.id_pesan
HAVING jumlah > 2
```

Lihat hasilnya sebagai berikut:

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        1 |      3 |
|        2 |      3 |
+----------+--------+
```

## SubSELECT

Mulai versi 4.1, MySQL mendukung perintah query SubSELECT dimana memungkinkan untuk melakukan query di dalam query. Misalnya kita akan menampilkan data yang kondisinya merupakan hasil dari query lain.

Perintah SubSELECT memiliki banyak variasi. Berikut ini beberapa variasi bentuk perintah SubSELECT.

```
SELECT ... WHERE col=[ANY|ALL] (SELECT ...);

SELECT ... WHERE col [NOT] IN (SELECT ...);

SELECT ROW(val1,val2,..) =[ANY] (SELECT col1,col2,..);

SELECT ... WHERE col = [NOT] EXISTS (SELECT ...);
```

```
SELECT ... FROM (SELECT ...) AS name WHERE ...;
```

Dan berikut ini beberapa contoh perintah query yang menggunakan SubSELECT.

- Menampilkan daftar pelanggan yang pernah melakukan transaksi (pemesanan).

```
SELECT id_pelanggan, nm_pelanggan FROM pelanggan
WHERE id_pelanggan IN (SELECT id_pelanggan FROM
pesan);
```

Hasilnya sebagai berikut:

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0001        | Achmad Solichin |
| P0002        | Budianto        |
| P0004        | Amin Riyadi     |
+--------------+-----------------+
```

- Menampilkan data pemesanan dengan jumlah barang terbanyak.

```
SELECT id_pesan, jumlah FROM detil_pesan
WHERE jumlah = ( SELECT MAX(jumlah) FROM
detil_pesan);
```

Hasilnya sebagai berikut:

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        4 |     10 |
+----------+--------+
```

## Menampilkan Record secara Random

MySQL memiliki fungsi khusus yang dapat digunakan untuk menampilkan record secara acak (random). Seperti kita ketahui bahwa pada perintah SELECT record akan ditampilkan secara urut berdasarkan urutan saat penginputan (FIFO = First In First Out).

Berikut ini contoh perintah query untuk menampilkan data pelanggan secara acak (random):

```
SELECT id_pelanggan, nm_pelanggan, email
FROM pelanggan ORDER BY RAND()
```

Salah satu hasilnya sebagai berikut:

```
+--------------+----------------+--------------------+
| id_pelanggan | nm_pelanggan   | email              |
+--------------+----------------+--------------------+
| P0004        | Amin Riyadi    | aminudin@plasa.com |
| P0001        | Achmad Solichin| achmatim@gmail.com |
| P0002        | Budianto       | budi@luhur.com     |
| P0003        | Hasan          | hasan02@yahoo.com  |
+--------------+----------------+--------------------+
```

# Transaksi

MySQL merupakan software database berbasis client-server. Hal ini berarti bahwa beberapa client dapat melakukan koneksi ke server MySQL secara bersamaan. Masing-masing client dapat melakukan select, insert, update, maupun delete data di server MySQL. Hal ini tentunya dapat menjadi masalah jika terjadi bentrok antar-client.

Sebagai contoh dalam proses transaksi pemesanan barang. Jika terdapat 2 (dua) pelanggan melakukan transaksi pada waktu yang sama, misalnya melakukan pemesanan barang. Keduanya akan menambahkan data di tabel yang sama dan mungkin saja data yang dimasukkan tertukar atau tidak valid. Hal ini tidak akan terjadi jika pada saat satu pelanggan melakukan transaksi, pelanggan yang lain harus menunggu sampai proses transaksi selesai.

Untuk mengatur proses query yang terjadi dalam suatu sistem yang memiliki user banyak (multi-user-system), kita dapat memanfaatkan dua hal di MySQL. **Pertama** kita dapat **mengunci tabel** (table-locking). Cara ini dapat dilakukan jika tipe tabel yang digunakan adalah MyISAM. **Kedua**, dapat menggunakan perintah BEGIN, COMMIT dan ROLLBACK. Cara ini dapat dilakukan jika tipe tabel adalah tabel transaksi, yaitu InnoDB.

Terdapat 4 (empat) prinsip dasar transaksi yang biasa disingkat sebagai **ACID**, yaitu:

**Atomicity**. Atom merupakan komponen terkecil dari materi, atau sesuatu yang tidak dapat dibagi-bagi lagi. Prinsip ini berlaku pada proses transaksi. Semua proses (perintah) yang ada di dalam satu paket transaksi harus

selesai semua atau tidak selesai sama sekali. Dengan kata lain, dalam satu transaksi tidak boleh ada proses (perintah) yang gagal dieksekusi.

**Consistency.** Bahwa kegagalan satu proses dalam transaksi tidak akan mempengaruhi transaksi lainnya.

**Isolation**. Secara sederhana, bahwa data yang sedang digunakan dalam satu transaksi, tidak dapat digunakan oleh transaksi lainnya sebelum seluruh proses transaksi yang pertama selesai.

**Durability**. Jika sebuah transaksi selesai dieksekusi, hasilnya tetap tercatat dengan baik.

## Mengunci Tabel

Mengunci tabel (table-locking) dapat dilakukan untuk membatasi akses terhadap suatu tabel jika ada user yang sedang aktif mengakses tabel. Suatu tabel yang sedang terkunci (locked), tidak dapat diakses dengan bebas oleh user lain. Untuk mengunci tabel di MySQL dapat menggunakan kata kunci LOCK.

Bentuk umum perintah LOCK :

```
LOCK TABLE table1 locktype, table2 locktype, ...;
```

Dimana tipe lock (*locktype*) yang dapat dipilih antara lain:

**READ**. Semua user MySQL dapat membaca (mengakses) tabel, namun tabel tersebut tidak dapat diubah oleh siapapun, termasuk user yang mengeksekusi perintah *LOCK*. Dengan kata lain, tabel bersifat *read-only*.

**READ LOCAL.** Sama seperti READ, tetapi perintah INSERT dapat dilakukan selama tidak merubah record (data) yang ada.

**WRITE.** User yang aktif diperbolehkan untuk membaca dan mengubah tabel (*read-and-write*). User yang lain tidak dapat mengubah isi tabel, hanya dapat membaca tabel (*read-only*).

**LOW PRIORITY WRITE.** Seperti halnya WRITE, hanya saja user lain dapat melakukan READ LOCK.

Untuk membuka tabel yang terkunci, gunakan perintah **UNLOCK TABLES;**

Contoh *table-locking*:

```
> LOCK TABLES trans READ, customer WRITE;
> SELECT sum(value) FROM trans WHERE customer_id= some_id;
> UPDATE customer SET total_value=total
WHERE customer_id=some_id;

> UNLOCK TABLES;
```

## BEGIN, COMMIT dan ROLLBACK

BEGIN atau START TRANSACTION digunakan untuk memulai transaksi, COMMIT untuk mengakhiri transaksi dan menyimpan semua perubahan, sedangkan ROLLBACK digunakan untuk menghentikan proses transaksi dan mengabaikan semua perubahan yang telah dilakukan.

Pada tipe tabel InnoDB, tidak berlaku transaksi didalam transaksi (*nested-transaction*), artinya jika perintah BEGIN dieksekusi sebelum transaksi selesai dilakuka (perintah COMMIT), maka secara otomatis, perintah COMMIT akan dieksekusi terlebih dahulu.

Perintah transaksi diatur oleh client. Jika pada saat proses transaksi berlangsung, koneksi client dengan server terputus, maka secara otomatis, MySQL akan membatalkan semua perubahan yang sudah terjadi (seperti halnya mengeksekusi perintah ROLLBACK).

Bentuk umum perintah transaksi :

```
BEGIN;
SELECT | INSERT | UPDATE | DELETE;
COMMIT;
```

Contoh :

```
> SET AUTOCOMMIT=0;
> BEGIN;
> INSERT INTO pesan VALUES(NULL,'P0001',now());
> SET @id := LAST_INSERT_ID();
> INSERT INTO detil_pesan VALUES (@id,'B0001','2','2500');
> COMMIT;
```

# Bab 7
# Administrasi dan Keamanan di MySQL

- Keamanan di MySQL
- Memahami Hak Akses (Priviledges) di MySQL
- Grant dan Revoke di MySQL
- Menambah dan Mengatur Hak Akses User
- Menghapus Hak Akses User
- Mengganti Password User

## Keamanan di MySQL

Masalah keamanan (*security*) di MySQL merupakan hal yang tidak boleh dianggap sepele apalagi dikesampingkan. MySQL merupakan software database yang bersifat client-server, yang memungkinkan beberapa user dapat mengakses server MySQL dari mana pun. Untuk itu, server MySQL harus benar-benar aman dari akses (serangan) orang-orang yang tidak berhak.

Berikut ini beberapa hal yang harus diperhatikan dalam mengamankan server MySQL:

a. JANGAN PERNAH MEMBERI AKSES KE SEMUA USER (KECUALI USER root) untuk dapat mengakses database **mysql**. Jika seseorang dapat mengakses database ini, maka dia dapat melihat informasi user (termasuk user, password dan host) MySQL dan (mungkin) dapat menambah atau mengubah informasi tersebut.

b. Pelajari mengenai hak akses di MySQL. Perintah GRANT dan REVOKE digunakan untuk mengatur hak akses di MySQL. Sebisa mungkin jangan memberikan hak akses ke MySQL pada semua host (%). Dan cobalah untuk mengecek dengan:
   1. Cobalah login dengan perintah mysql –u root. Jika Anda berhasil login ke server, maka hal ini bisa menjadi masalah besar, karena password root masih kosong sehingga semua user dapat mengakses server MySQL.
   2. Gunakan perintah SHOW GRANTS untuk melihat semua hak akses user.

c. Jangan pernah menyimpan password dalam bentuk teks biasa di MySQL! Gunakan fungsi enkripsi searah seperti fungsi PASSWORD() dan MD5() untuk mengenkripsi isi password. Kita tidak dapat menjamin 100% bahwa server kita aman dari penyusup (*intruder*).

d. Hati-hati dalam memilih password. Pilihlah password yang mudah diingat tapi sulit ditebak oleh orang lain. Dan juga jangan gunakan kata-kata yang ada di kamus, gunakanlah kombinasi angka dan huruf.

e. Pasang *firewall* di server untuk mencegah penyusup. Hal ini dapat mencegah setidaknya 50% dari program penyusup yang ada.

f. Jangan percaya sepenuhnya terhadap data yang dimasukkan oleh user. Akan lebih baik jika kita menganggap bahwa semua user adalah '*jahat*'. Lakukan validasi data sebelum dimasukkan ke database. Hal ini biasanya dapat dilakukan di dalam bahasa pemrograman yang digunakan.

g. Hati-hati dalam mengirim atau mentransfer data lewat internet, karena mungkin ada orang lain yang dapat 'membajak' data tersebut.

Dalam hal pengamanan server MySQL, setidaknya ada beberapa faktor yang mempengaruhi. Kita belum cukup jika mengamankan satu sisi (faktor) saja, tetapi harus menyeluruh. Berikut ini beberapa faktor tersebut:

1. Server atau komputer tempat MySQL berada. Server tempat MySQL diinstall tentunya menjadi gerbang utama bagi penyusup (*intruder*). Untuk ini kita harus benar-benar memperhatikan faktor keamanan server. Kita dapat memasang firewall untuk membatasi akses penyusup ke server. Gunakan prinsip *deny-all, allow-some*, dimana kita menutup semua lubang dan hanya membuka yang diperlukan.
2. Server MySQL. Konfigurasi dan settingan dalam server MySQL juga sangat mempengaruhi keamanan data MySQL. Bagaimana jadinya jika user yang tidak berhak dapat mengakses sistem dan konfigurasi MySQL? Tentu sangat berbahaya.
3. Aplikasi (Pemrograman) yang digunakan. Aplikasi disini maksudnya adalah pemrograman yang menggunakan atau berhubungan langsung dengan MySQL. Sebagian besar penyusup akan memilih cara menyusup melalui aplikasi jika kedua hal diatas tidak dapat dilakukan. Dan banyak database yang kebobolan karena kelemahan dari sisi aplikasi. Secara sederhana kita

dapat mengakses data ke MySQL melalui konsep yang sering disebut sebagai SQLInjection.

4. User atau pengguna. User atau pengguna server MySQL juga mempengaruhi keamanan datanya. Misalnya pemilihan password yang mudah ditebak (seperti tanggal lahir), kecerobohan user yang lupa logout setelah menggunakan MySQL atau user yang menuliskan passwordnya di buku catatan.

## Memahami Hak Akses (Privileges) di MySQL

MySQL pada dasarnya merupakan sistem database yang aman. Di MySQL kita dapat mengatur hak akses tiap user terhadap data di database. MySQL memungkinkan kita mengatur hak akses user sampai pada tingkat kolom. Artinya kita dapat mengatur kolom tertentu dapat diakses oleh user siapa saja. Tentu, kita juga dapat mengatur hak akses user terhadap tabel, dan database.

Semua pengaturan hak akses (*privileges*) tersimpan di database **mysql** yang secara *default* sudah ada di sistem MySQL. Di dalam database **mysql** antara lain terdapat tabel-tabel sebagai berikut:

**user**. Tabel ini digunakan untuk menyimpan informasi user MySQL yang mencakup informasi user, password dan host user, serta informasi hak akses user.

**db**. Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap database.

**host**. Tabel ini digunakan untuk menyimpan daftar komputer (bisa berupa alamat IP, nama komputer, atau %) yang berhak mengakses suatu database.

**tables_priv**. Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap tabel. Dengan kata lain menyimpan tabel ini dapat diakses oleh siapa dengan hak akses apa saja.

**columns_priv**. Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap kolom.

**procs_priv**. Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap procedure.

**proc**. Tabel ini digunakan untuk menyimpan informasi mengenai daftar procedure dalam MySQL.

**func**. Tabel ini digunakan untuk menyimpan informasi mengenai function yang didefinisikan di MySQL.

## GRANT dan REVOKE di MySQL

Untuk mengatur hak akses di MySQL, pada dasarnya kita menggunakan bentuk perintah GRANT dan REVOKE.

Berikut ini bentuk umum perintah GRANT dan REVOKE secara sederhana :

```
GRANT priv_type
ON {tbl_name | * | *.* | db_name.*}
TO user_name [IDENTIFIED BY 'password']
[WITH GRANT OPTION]

REVOKE priv_type
ON {tbl_name | * | *.* | db_name.*}
FROM user_name
```

Berikut ini pilihan untuk priv_type dalam bentuk umum perintah GRANT dan REVOKE di atas:

```
ALL PRIVILEGES    FILE            RELOAD
ALTER             INDEX           SELECT
CREATE            INSERT          SHUTDOWN
DELETE            PROCESS         UPDATE
DROP              REFERENCES      USAGE
```

Perintah GRANT dan REVOKE dapat digunakan untuk membuat user baru maupun mengatur hak akses user yang sudah ada dengan hak akses (privileges ) tertentu. Tingkatan hak akses user dapat terbagi menjadi tingkatan global (tersimpan di tabel mysql.user), database (tersimpan di tabel mysql.host dan mysql.db), tabel (tersimpan di tabel mysql.tables_priv) dan kolom (tersimpan di tabel mysql.columns_priv).

Setiap perubahan hak akses di MySQL, termasuk menambahkan user baru, tidak akan berlaku sebelum diakhiri dengan perintah **FLUSH PRIVILEGES**.

# Menambahkan dan Mengatur Hak Akses User

Untuk menambahkan dan mengatur hak akses (*privileges*) user di MySQL, kita dapat menggunakan 2 cara. **Pertama** langsung melakukan INSERT atau UPDATE ke tabel mysql.user, dan tabel-tabel lain sesuai dengan hak aksesnya. Cara ini tidak disarankan karena mengandung resiko terjadi kesalahan.

Cara **kedua** adalah dengan perintah GRANT dan REVOKE. Perintah ini mudah dipahami dan diterapkan karena lebih sederhana. MySQL secara otomatis akan menyimpan informasi user ke tabel sesuai dengan hak aksesnya.

Berikut ini beberapa contoh menambahkan user baru di MySQL:

- Menambahkan user baru dengan nama user '**monty'** yang dapat mengakses semua database dari komputer '**localhost'** dengan password '**qwerty'.** User ini juga berhak menjalankan perintah GRANT untuk user lain.

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

- Menambahkan user baru dengan nama user '**adinda'**, tidak dapat mengakses database (*.*), hanya dapat mengakses dari komputer dengan IP '**192.168.1.5'** dan password '**qwerty'.**

```
GRANT USAGE ON *.* TO adinda@192.168.1.5
IDENTIFIED BY 'qwerty';
```

- Menambahkan user baru dengan nama user '**admin**, hanya dapat mengakses database '**mysql'**, hanya dapat mengakses dari komputer '**localhost'** dan dengan password '**qwerty'.**

```
GRANT ALL PRIVILEGES ON mysql.* TO admin@localhost
IDENTIFIED BY 'qwerty';
```

Berikut ini beberapa contoh mengatur hak akses user yang sudah ada di MySQL:

- Mengubah hak akses user '**adinda'** agar dapat mengakses database '**penjualan'**.

```
GRANT ALL PRIVILEGES ON penjualan.* TO
adinda@192.168.1.5;
```

```
FLUSH PRIVILEGES;
```

- Mengubah hak akses user '**admin'** agar dapat CREATE di database '**penjualan'**.

```
GRANT CREATE ON penjualan.* TO admin@localhost;
```

```
FLUSH PRIVILEGES;
```

## Menghapus Hak Akses User

Untuk menghapus hak akses user, dapat dilakukan dengan perintah REVOKE. Berikut ini contohnya:

- Menghapus hak akses user '**admin'** terhadap database '**penjualan'**.

```
REVOKE CREATE ON penjualan.* FROM admin@localhost;
```

```
FLUSH PRIVILEGES;
```

## Mengganti Password User

Untuk mengganti password suatu user di MySQL, kita tinggal menjalankan perintah UPDATE terhadap field Password di tabel mysql.user. Password tersebut diekripsi dengan fungsi PASSWORD().

Berikut ini perintah SQL yang dapat digunakan untuk mengganti password user:

```
UPDATE user SET Password=PASSWORD('123') WHERE
User='admin' AND Host='localhost';
```

```
SET PASSWORD FOR admin@localhost = PASSWORD ('123');
```

```
FLUSH PRIVILEGES;
```

# Bab 8
# Trigger dan Views

- Triggers
- Views

## Trigger

Trigger digunakan untuk memanggil satu atau beberapa perintah SQL secara otomatis sebelum atau sesudah terjadi proses INSERT, UPDATE atau DELETE dari suatu tabel. Sebagai contoh misalnya kita ingin menyimpan id pelanggan secara otomatis ke tabel '**log'** sebelum menghapus data di tabel **pelanggan**.

Triggers mulai dikenal di versi MySQL 5.0, dan di versi saat ini (5.0.4) fungsionalitasnya sudah bertambah. Pada versi selanjutnya (5.1) pihak pengembang MySQL berjanji akan lebih menguatkan (menambah) fitur trigger ini.

Trigger sering digunakan, antara lain untuk:

- Melakukan update data otomatis jika terjadi perubahan. Contohnya adalah dalam sistem penjualan, jika dientri barang baru maka stock akan bertambah secara otomatis.
- Trigger dapat digunakan untuk mengimplementasikan suatu sistem log. Setiap terjadi perubahan, secara otomatis akan menyimpan ke tabel log.
- Trigger dapat digunakan untuk melakukan validasi dan verifikasi data sebelum data tersebut disimpan.

### Membuat Trigger Baru

Berikut ini bentuk umum perintah untuk membuat triggers:

```
CREATE TRIGGER name
[BEFORE|AFTER] [INSERT|UPDATE|DELETE]
ON tablename
FOR EACH ROW statement
```

dimana

**BEFORE | AFTER** digunakan untuk menentukan kapan proses secara otomatis akan dieksekusi, sebelum atau sesudah proses.

**INSERT | UPDATE | DELETE** digunakan untuk menentukan event yang dijadikan trigger untuk menjalankan perintah-perintah di dalam triggers.

Statement atau perintah dalam trigger dapat berupa satu perintah saja, dan dapat juga beberapa perintah sekaligus. Jika terdapat beberapa perintah dalam trigger, maka gunakan perintah **BEGIN** dan **END** untuk mengawali dan mengakhiri perintah.

Di dalam statement trigger, kita dapat mengakses record tabel sebelum atau sesudah proses dengan menggunakan **NEW** dan **OLD. NEW** digunakan untuk mengambil record yang akan diproses (insert atau update), sedangkan **OLD** digunakan untuk mengakses record yang sudah diproses (update atau delete).

Berikut ini contoh trigger yang akan mencatat aktivitas ke tabel **log** setiap terjadi proses insert ke tabel pelanggan:

```
DELIMITER $$

CREATE TRIGGER penjualan.before_insert BEFORE INSERT ON
penjualan.pelanggan
FOR EACH ROW BEGIN
     INSERT INTO `log` (description, `datetime`, user_id)
     VALUES (CONCAT('Insert data ke tabel pelanggan id_plg
= ', NEW.id_pelanggan), now(), user());
END;
$$

DELIMITER ;
```

## Menghapus Trigger

Untuk menghapus trigger, dapat menggunakan perintah **DROP TRIGGER** dengan diikuti dengan nama tabel dan nama triggernya. Berikut ini bentuk umum dan contoh perintah untuk menghapus trigger.

```
DROP TRIGGER tablename.triggername;
```

Contoh :

```
DROP TRIGGER penjualan.before_insert;
```

# Views

Views di MySQL mulai disediakan pada versi 5.0. Views merupakan suatu tampilan tabel virtual. Views berisi perintah SELECT ke tabel dalam database. Views dapat digunakan untuk mempermudah kita dalam pembuatan laporan atau tampilan database yang diinginkan dengan cepat. Dengan kata lain, views merupakan perintah SELECT yang disimpan, sehingga setiap saat kita membutuhkannya, kita dapat langsung memanggilnya tanpa perlu mengetikkan perintah SELECT kembali.

## Membuat dan Mendefinisikan Views

View dibuat atau didefinisikan dengan menggunakan perintah CREATE VIEW. Bentuk umum perintah untuk membuat (mendefinisikan) view, sebagai berikut:

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Berikut ini contoh view untuk menampilkan data id, nama dan telepon pelanggan dari tabel pelanggan yang diurutkan berdasarkan nama pelanggan.

```
CREATE VIEW `data_plg` AS
(select id_pelanggan, nm_pelanggan, telepon
from `pelanggan` order by `nm_pelanggan`)
```

Dan untuk mengeksekusi perintah di atas, kita dapat memanggil dengan perintah SELECT seperti halnya menampilkan data dari suatu tabel. Berikut ini contoh cara pemanggilan view beserta hasil querynya.

```
SELECT * FROM data_plg;
```

```
+--------------+-----------------+--------------+
| id_pelanggan | nm_pelanggan    | telepon      |
+--------------+-----------------+--------------+
| P0001        | Achmad Solichin | 021-7437299  |
| P0004        | Amin Riyadi     | 021-3239991  |
| P0005        | Animania        | 021-93949992 |
| P0002        | Budianto        | 021-349924   |
| P0003        | Hasan           | 021-2339292  |
+--------------+-----------------+--------------+
```

Contoh lain misalnya jika kita ingin membuat view untuk menampilkan laporan jumlah barang dari setiap transaksi pemesanan yang dilakukan oleh pelanggan.

```
CREATE VIEW lap_jumlah_brg_transaksi
AS
(SELECT pesan.id_pesan, pesan.tgl_pesan,
pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
detil_pesan.jumlah
FROM pesan, detil_pesan, pelanggan
WHERE pesan.id_pesan=detil_pesan.id_pesan AND
pelanggan.id_pelanggan=pesan.id_pelanggan
GROUP BY pesan.id_pesan)
```

Dan jika dipanggil hasilnya menjadi sebagai berikut:

```
SELECT * FROM lap_jumlah_brg_transaksi;

+----------+------------+--------------+-----------------+--------+
| id_pesan | tgl_pesan  | id_pelanggan | nm_pelanggan    | jumlah |
+----------+------------+--------------+-----------------+--------+
|        1 | 2008-02-02 | P0001        | Achmad Solichin |      2 |
|        2 | 2008-02-05 | P0002        | Budianto        |      1 |
|        3 | 2008-02-10 | P0002        | Budianto        |      5 |
|        4 | 2008-01-20 | P0004        | Amin Riyadi     |     10 |
|        7 | 2008-02-24 | P0001        | Achmad Solichin |      2 |
+----------+------------+--------------+-----------------+--------+
```

## Mengubah View

View yang sudah dibuat, dapat diubah dengan perintah ALTER. Bentuk umum perintah untuk mengubah view yang sudah ada, sebagai berikut:

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
```

```
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Berikut ini contoh untuk mengubah view yang sudah ada:

```
ALTER VIEW `data_plg` AS
(select * from `pelanggan` order by `nm_pelanggan`)
```

## Menghapus View

View yang sudah dibuat, dapat dihapus dengan perintah DROP. Berikut ini bentuk umum dan contoh perintah untuk menghapus view.

```
DROP VIEW view_name;
```

Contoh :

```
DROP VIEW data_plg;
```

# Bab 9
# Function dan Stored Procedure

- Hello World!
- Membuat, Mengubah dan Menghapus SP
- Sintaks Dasar dalam SP

Function dan Stored Procedure merupakan fitur utama yang paling penting di MySQL 5. Function dan Stored Procedure merupakan suatu kumpulan perintah atau statement yang disimpan dan dieksekusi di server database MySQL. Dengan SP (Stored Procedure), kita dapat menyusun program sederhana berbasis sintaks SQL untuk menjalankan fungsi tertentu. Hal ini menjadikan aplikasi yang kita buat lebih efektif dan efisien.

Berikut ini beberapa keuntungan menggunakan Stored Procedure:

- **Lebih cepat**. Hal ini karena kumpulan perintah query dijalankan langsung di server. Berbeda dengan jika dijalankan secara sekuensial di bahasa pemrograman, akan lebih lambat karena harus *"bolak-balik"* antara client dan server.
- **Menghilangkan duplikasi proses, pemeliharaan yang mudah.** Pada dasarnya operasi yang terjadi di suatu aplikasi terhadap database adalah sama. Secara umum, di dalam aplikasi biasanya terdapat operasi untuk validasi data inputan, menambahkan record baru, mengubah record, menghapus record dan sebagainya. Dengan SP, mungkin kita dapat menghindari adanya duplikasi proses yang kurang lebih sama, sehingga pemeliharaannya juga jadi lebih mudah.
- **Meningkatkan keamanan database.** Dengan adanya SP, database akan lebih aman karena aplikasi yang memanggil SP tidak perlu mengetahui isi di dalamnya. Sebagai contoh, dalam proses menambahkan data (insert), kita membuat suatu SP khusus. Dengan demikian, saat client atau aplikasi akan menambahkan data (insert) maka tidak perlu tahu nama tabelnya, karena hanya cukup memanggil SP tersebut dengan mengirimkan parameter yang diinginkan.

Selanjutnya, Stored Procedure dari segi bentuk dan sifatnya terbagi menjadi 2 (dua), yaitu FUNCTION dan PROCEDURE. Perbedaan utama antara function dan procedure adalah terletak pada nilai yang dikembalikannya (di-return). Function memiliki suatu nilai yang dikembalikan (di-return), sedangkan procedure tidak.

Umumnya suatu procedure hanya berisi suatu kumpulan proses yang tidak menghasilnya value, biasanya hanya menampilkan saja.

*Sebagai catatan bahwa dalam buku ini jika terdapat istilah SP (Stored Procedure) maka yang dimaksud adalah Function dan Procedure.*

## Hello World!

Sebagai contoh sederhana, kita akan membuat suatu SP yang akan menampilkan string "Hello World!" di layar hasil. Berikut ini perintah query untuk membuat SP tersebut:

```
DELIMITER $$
CREATE PROCEDURE hello()
    BEGIN
         SELECT "Hello World!";
    END$$
DELIMITER ;
```

Untuk memanggil procedure tersebut, gunakanlah CALL. Berikut ini contoh pemanggilan procedure dan hasil tampilannya:

```
CALL hello();
```

Hasilnya sebagai berikut:

```
+--------------+
| Hello World! |
+--------------+
| Hello World! |
+--------------+
```

## Membuat, Mengubah dan Menghapus SP

### Membuat SP

Untuk membuat SP baru, berikut ini bentuk umumnya:

```
CREATE
```

```
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body
```

**Contoh 1**. Procedure untuk menghitung jumlah pelanggan

```
DELIMITER $$
CREATE PROCEDURE jumlahPelanggan()
    BEGIN
         SELECT COUNT(*) FROM pelanggan;
    END$$
DELIMITER ;
```

Cara pemanggilan dari procedure diatas adalah dengan menggunakan **CALL jumlahPelanggan()**. Hasilnya akan ditampilkan jumlah record dari tabel pelanggan.

Berikut ini bentuk lain dari contoh diatas:

```
DELIMITER $$
CREATE PROCEDURE jumlahPelanggan2(OUT hasil AS INT)
    BEGIN
         SELECT COUNT(*) INTO hasil FROM pelanggan;
    END$$
DELIMITER ;
```

Pada bentuk procedure yang kedua di atas (**jumlahPelanggan2**), kita menyimpan hasil dari procedure ke dalam satu variabel bernama **hasil** yang bertipe **INT.** Perbedaan dari kedua bentuk di atas adalah, pada bentuk kedua, kita dapat memanggil procedure dengan SELECT, sedangkan pada yang pertama tidak bisa. Berikut ini contoh pemanggilan untuk procedure yang kedua:

```
mysql> CALL jumlahPelanggan2(@jumlah);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @jumlah AS `Jumlah Pelanggan`;
+------------------+
| Jumlah Pelanggan |
+------------------+
| 5                |
```

```
+------------------+
1 row in set (0.02 sec)
```

**Contoh 2**. Procedure untuk menghitung jumlah item barang yang pernah dibeli oleh satu pelanggan.

```
DELIMITER $$
CREATE PROCEDURE
    jumlahItemBarang (pelanggan VARCHAR(5))

    BEGIN
           SELECT SUM(detil_pesan.jumlah)
           FROM pesan, detil_pesan
           WHERE pesan.id_pesan=detil_pesan.id_pesan
           AND pesan.id_pelanggan=pelanggan;
    END$$
DELIMITER ;
```

**Contoh 3**. Function untuk menghitung jumlah produk yang tersedia (stock) untuk satu produk tertentu.

```
DELIMITER $$
CREATE FUNCTION jumlahStockBarang(produk VARCHAR(5))
    RETURNS INT

    BEGIN
        DECLARE jumlah INT;
        SELECT COUNT(*) INTO jumlah FROM produk
         WHERE id_produk=produk;
        RETURN jumlah;
    END$$
DELIMITER ;
```

Untuk memanggil suatu function, kita tidak menggunakan CALL, tetapi langsung dapat memanggil dengan SELECT. Berikut ini contoh pemanggilan untuk fungsi di atas.

```
SELECT jumlahStockBarang('B0001');
```

Dan berikut ini hasilnya:

```
+----------------------------+
| jumlahStockBarang('B0001') |
+----------------------------+
|                          1 |
+----------------------------+
```

### Mengubah SP

Untuk mengubah SP yang sudah ada, berikut ini bentuk umumnya:

```
ALTER {PROCEDURE | FUNCTION} sp_name
    [characteristic ...]
```

### Menghapus SP

Untuk menghapus SP yang sudah ada, berikut ini bentuk umumnya:

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

## Sintaks Dasar dalam SP

SP dapat dikatakan sebagai bahasa pemrograman yang berada di dalam database. Oleh karena itu, tentunya terdapat sintaks-sintaks tertentu berhubungan dengan SP tersebut, misalnya bagaimana untuk mendeklarasikan variabel, penyeleksian kondisi, perulangan dsb. Pada bagian ini akan diuraikan beberapa sintaks dasar SP yang didukung oleh MySQL.

### Variabel

Variabel digunakan untuk menyimpan suatu nilai secara temporer (sementara) di memory. Variabel akan hilang saat sudah tidak digunakan lagi.

Variabel dalam MySQL sebelum dapat digunakan, pertama kali harus dideklarasikan terlebih dahulu. Berikut ini bentuk umum pendeklarasian suatu variabel di MySQL:

```
DECLARE variable_name DATATYPE [DEFAULT value];
```

Contohnya:

```
DECLARE jumlah INT;
DECLARE kode VARCHAR(5);
DECLARE tgl_lahir DATE DEFAULT '1982-10-20';
```

Setelah dideklarasikan, suatu variabel dapat diisi dengan suatu nilai sesuai dengan tipe data yang didefinisikan saat pendeklarasian. Untuk mengisikan nilai

ke dalam suatu variabel, digunakan perintah SET. Format umumnya sebagai berikut:

```
SET variable_name = expression|value;
```

Contohnya:

```
SET jumlah = 10;
SET kode = (SELECT id_pelanggan FROM pelanggan LIMIT 1);
SET tgl_lahir = now();
```

Berikut ini contoh function **hitungUmur()** untuk menghitung umur seseorang saat ini berdasarkan tahun kelahiran yang diberikan.

```
DELIMITER $$
CREATE FUNCTION hitungUmur (lahir DATE)
   RETURNS INT
    BEGIN
      DECLARE thn_sekarang, thn_lahir INT;
      SET thn_sekarang = YEAR(now());
      SET thn_lahir = YEAR (lahir);
      RETURN thn_sekarang - thn_lahir;
    END$$
DELIMITER ;
```

## Penyeleksian Kondisi

Dengan adanya fasilitas penyeleksian kondisi, kita dapat mengatur alur proses yang terjadi dalam database kita. Di MySQL, penyeleksian kondisi terdiri dari IF, IF...ELSE dan CASE. Berikut ini bentuk umum ketiga perintah tersebut:

```
IF kondisi THEN
    perintah-jika-benar;
END IF;

IF kondisi THEN
    perintah-jika-benar;
ELSE
    perintah-jika-salah;
END IF;

CASE expression
    WHEN value THEN
        statements
    [WHEN value THEN
        statements ...]
```

```
    [ELSE
        statements]
    END CASE;
```

Berikut ini contoh penggunaan perintah IF dalam fungsi **cekPelanggan**() dimana fungsi ini memeriksa apakah pelanggan sudah pernah melakukan transaksi pemesanan barang. Jika sudah pernah, tampilkan pesan berapa kali melakukan pemesanan, jika belum tampilkan pesan belum pernah memesan.

```
DELIMITER $$
CREATE FUNCTION cekPelanggan (pelanggan varchar(5))
    RETURNS VARCHAR (100)
    BEGIN
       DECLARE jumlah INT;
       SELECT COUNT(id_pesan) INTO jumlah FROM pesan
           WHERE id_pelanggan=pelanggan;
       IF (jumlah > 0) THEN
          RETURN CONCAT("Anda sudah bertransaksi sebanyak ",
          jumlah, " kali");
       ELSE
          RETURN "Anda belum pernah melakukan transaksi";
       END IF;
    END$$
DELIMITER ;
```

Dan berikut ini contoh penggunaan perintah CASE dalam fungsi **getDiskon**() dimana fungsi ini menentukan diskon berdasarkan jumlah pesanan yang dilakukan.

```
DELIMITER $$
CREATE FUNCTION getDiskon(jumlah INT) RETURNS int(11)
    BEGIN
       DECLARE diskon INT;
       CASE
          WHEN (jumlah >= 100) THEN
             SET diskon = 10;
          WHEN (jumlah >= 50 AND jumlah < 100) THEN
             SET diskon = 5;
          WHEN (jumlah >= 20 AND jumlah < 50) THEN
             SET diskon = 3;
          ELSE SET diskon = 0;
       END CASE;
       RETURN diskon;
    END$$

DELIMITER ;
```

## Perulangan

Selain penyeleksian kondisi, MySQL juga mendukung adanya perulangan dalam querynya. Perulangan biasanya digunakan untuk mengulang proses atau perintah yang sama. Dengan perulangan, perintah akan lebih efisien dan singkat.

Berikut ini bentuk-bentuk perintah perulangan:

```
[label:] LOOP
    statements
END LOOP [label];

[label:] REPEAT
    statements
UNTIL expression
END REPEAT [label]

[label:] WHILE expression DO
    statements
END WHILE [label]
```

Contoh perulangan dengan LOOP

```
SET i=1;
ulang: WHILE i<=10 DO
    IF MOD(i,2)<>0 THEN
        SELECT CONCAT(i," adalah bilangan ganjil");
    END IF;
    SET i=i+1;
END WHILE ulang;
```

# Bagian 4
# Laporan dan Backup di MySQL

# Bab 10
# Laporan di MySQL

- Laporan dari Tabel dan Query
- Format Laporan

Laporan biasanya disajikan dalam berbagai bentuk (format file). Umumnya bentuk laporan dapat berupa file Excel, PDF, HTML, XML, dsb. Pembuatan laporan di MySQL akan lebih mudah jika kita menggunakan software bantuan. Di sini akan disampaikan bagaimana membuat laporan dengan menggunakan software **SQLYog Community Edition**.

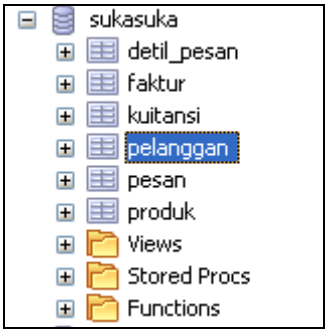
Laporan dari database umumnya didasarkan pada hasil (results) suatu perintah query SELECT. Dengan SELECT kita dapat menyeleksi tampilan atau hasil yang diinginkan dari tabel-tabel yang ada di database. SELECT dapat berupa seleksi ke satu tabel atau beberapa tabel sekaligus (join antar tabel).

## Laporan dari Tabel dan Query

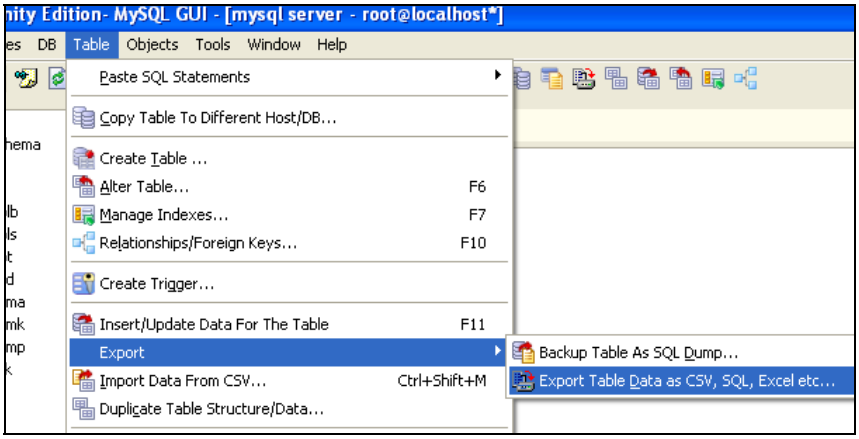
Dengan menggunakan SQLYog, kita dapat membuat laporan dari tabel (*table*-data) dan juga dari hasil query (*results*). Laporan dari tabel merupakan laporan yang menampilkan seluruh data dari suatu tabel, sedangkan laporan dari query merupakan laporan yang menampilkan seluruh data dari hasil suatu perintah query.

### Laporan dari Tabel

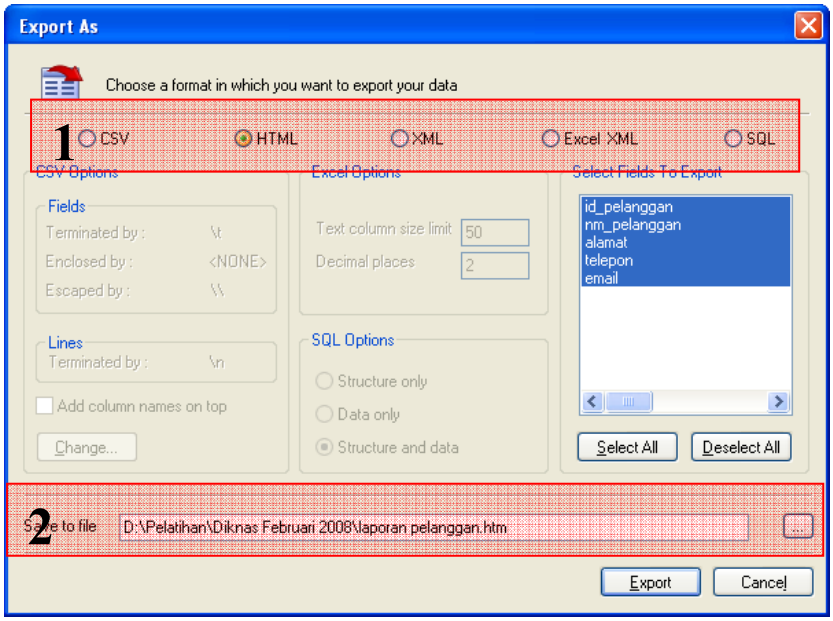
1. Pilih tabel yang akan dibuat laporan.



2. Pilih menu Table > Export > Export Table Data As CSV, SQL, Excel, etc.




3. Pilih format laporan dan tempat menyimpan hasil laporan.




4. Klik tombol Export.
5. Buka File hasil laporan.


| id_pelanggan | nm_pelanggan | alamat | telepon | email |
|---|---|---|---|---|
| P0001 | Joni | Jl Raya Sudirman<br>akarta Selatan | 085688888 | achmatim@gmail.com |
| P0002 | Adinda | Jl Raya Timur<br>akarta Selatan | 089023800 | budi@luhur.com |
| P0003 | Hasan | Jl Raya Ciledug, Jakarta Selatan | 021-2339292 | hasan02@yahoo.com |
| P0004 | Amin Riyadi | RT 04/01 Kelurahan Karang Tengah, Tangerang | 021-3239991 | aminudin@plasa.com |
| P0005 | Asih Muladi | Jl Asiman | 021-3192919 | adiana@plasa.com |

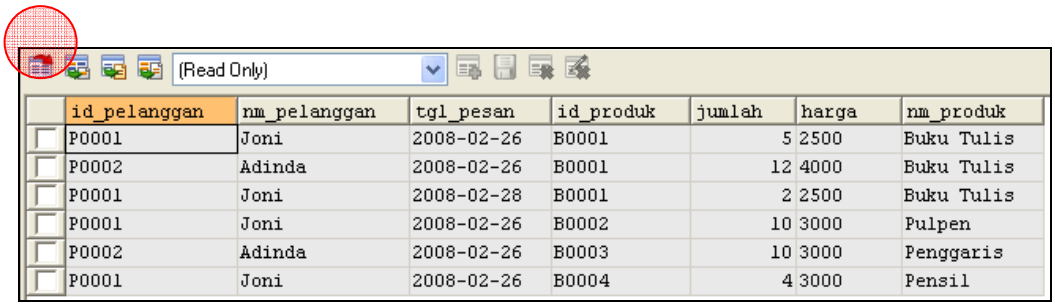
## Laporan dari Hasil Query

1. Ketikkan perintah query di jendela query.

transaksi.sql*

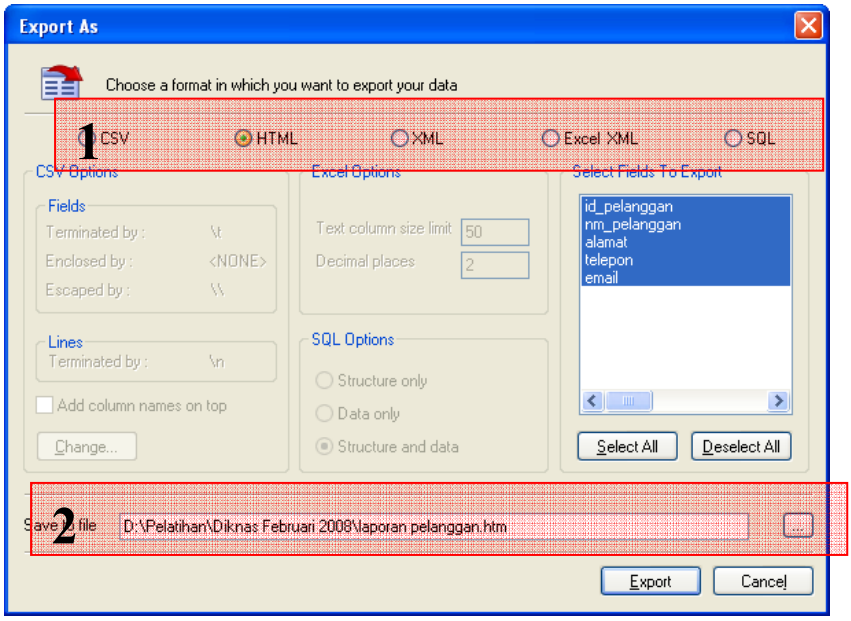
```
18  SELECT
19   pelanggan.id_pelanggan,
20   pelanggan.nm_pelanggan,
21   pesan.tgl_pesan,
22   detil_pesan.id_produk,detil_pesan.jumlah,
23   detil_pesan.harga,
24   produk.nm_produk
25  FROM
26   pelanggan, pesan, detil_pesan, produk
27  WHERE
28   pelanggan.id_pelanggan=pesan.id_pelanggan
29   AND pesan.id_pesan=detil_pesan.id_pesan
30   AND detil_pesan.id_produk=produk.id_produk;
```

2. Pilih tombol **Export As...** yang ada diatas hasil query.

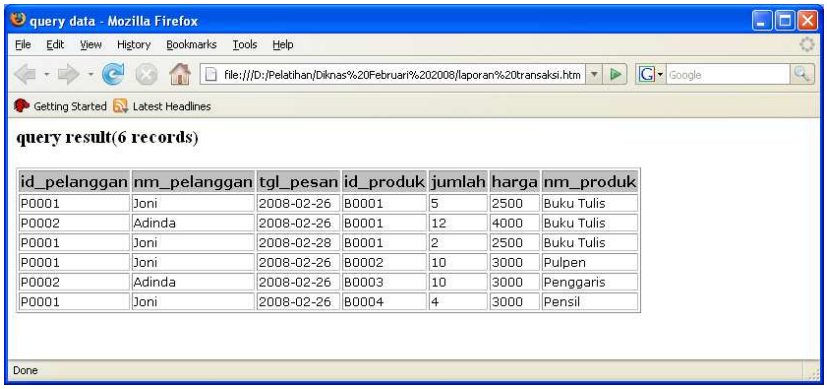



| id_pelanggan | nm_pelanggan | tgl_pesan | id_produk | jumlah | harga | nm_produk |
|---|---|---|---|---|---|---|
| P0001 | Joni | 2008-02-26 | B0001 | 5 | 2500 | Buku Tulis |
| P0002 | Adinda | 2008-02-26 | B0001 | 12 | 4000 | Buku Tulis |
| P0001 | Joni | 2008-02-28 | B0001 | 2 | 2500 | Buku Tulis |
| P0001 | Joni | 2008-02-26 | B0002 | 10 | 3000 | Pulpen |
| P0002 | Adinda | 2008-02-26 | B0003 | 10 | 3000 | Penggaris |
| P0001 | Joni | 2008-02-26 | B0004 | 4 | 3000 | Pensil |

3. Pilih format laporan dan tempat menyimpan hasil laporan.




4. Klik tombol Export.
5. Buka File hasil laporan.




| id_pelanggan | nm_pelanggan | tgl_pesan | id_produk | jumlah | harga | nm_produk |
|---|---|---|---|---|---|---|
| P0001 | Joni | 2008-02-26 | B0001 | 5 | 2500 | Buku Tulis |
| P0002 | Adinda | 2008-02-26 | B0001 | 12 | 4000 | Buku Tulis |
| P0001 | Joni | 2008-02-28 | B0001 | 2 | 2500 | Buku Tulis |
| P0001 | Joni | 2008-02-26 | B0002 | 10 | 3000 | Pulpen |
| P0002 | Adinda | 2008-02-26 | B0003 | 10 | 3000 | Penggaris |
| P0001 | Joni | 2008-02-26 | B0004 | 4 | 3000 | Pensil |

# Format Laporan

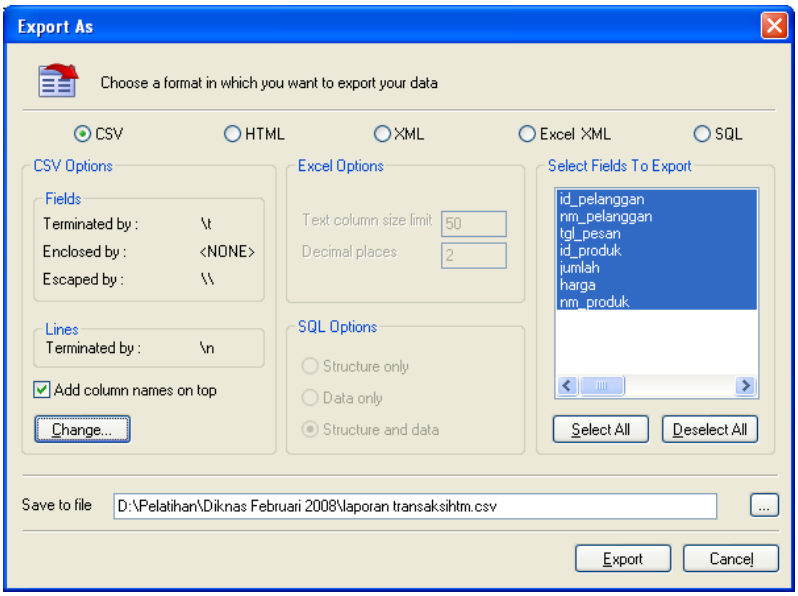
Format file laporan yang dapat di-*generate* secara langsung dengan SQLYog diantaranya file CSV, HTML, XML, Excel XML, dan SQL.

## CSV

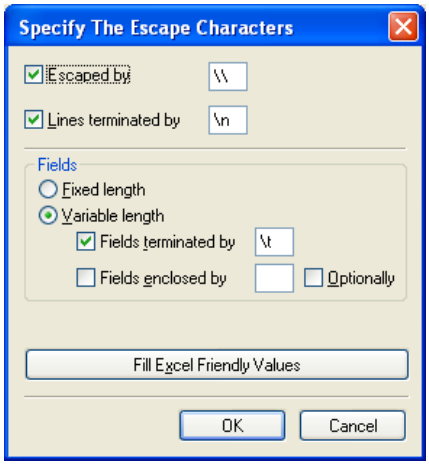
CSV merupakan singkatan dari *Comma Separated Version*. CSV merupakan suatu format data dimana setiap record dipisahkan dengan koma (,) atau titik

koma (;). Selain sederhana, format ini dapat dibuka dengan berbagai *text-editor* seperti Notepad, Word, bahkan MS Excel.

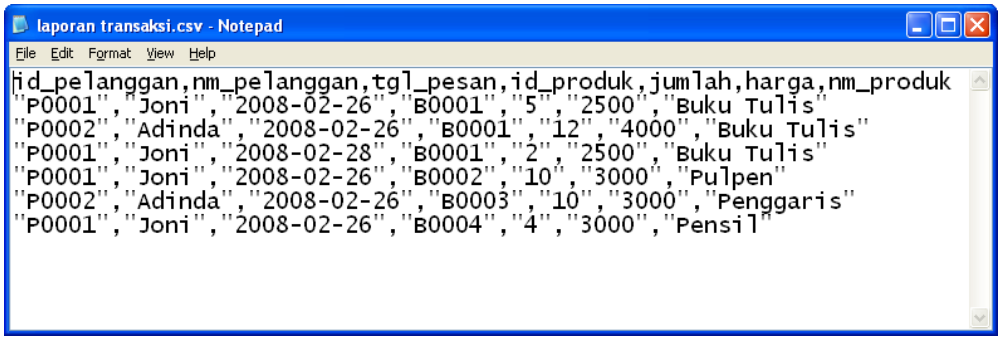
Berikut ini pilihan untuk membuat report dalam CSV.

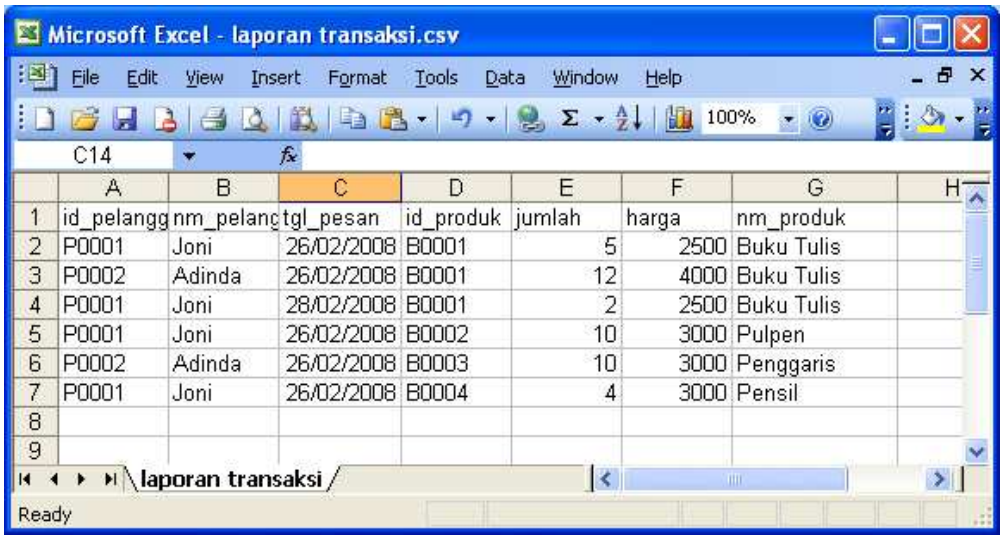



Untuk mengubah pilihan CSV seperti pemisah antar baris dsb, pilih tombol **Change**.




Contoh tampilan laporan dalam CSV jika dibuka dengan notepad dan MS Excel 2003.




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | id_pelangg | nm_pelang | tgl_pesan | id_produk | jumlah | harga | nm_produk |
| 2 | P0001 | Joni | 26/02/2008 | B0001 | 5 | 2500 | Buku Tulis |
| 3 | P0002 | Adinda | 26/02/2008 | B0001 | 12 | 4000 | Buku Tulis |
| 4 | P0001 | Joni | 28/02/2008 | B0001 | 2 | 2500 | Buku Tulis |
| 5 | P0001 | Joni | 26/02/2008 | B0002 | 10 | 3000 | Pulpen |
| 6 | P0002 | Adinda | 26/02/2008 | B0003 | 10 | 3000 | Penggaris |
| 7 | P0001 | Joni | 26/02/2008 | B0004 | 4 | 3000 | Pensil |
| 8 | | | | | | | |
| 9 | | | | | | | |

## HTML

HTML merupakan singkatan dari *HyperText Markup Language*. HTML merupakan yang dapat dibuka dengan browser (IE, Mozilla dll). Karena sifatnya yang kompatibel dengan browser maka format ini cocok dipilih jika kita menginginkan laporan dalam bentuk halaman web/internet.

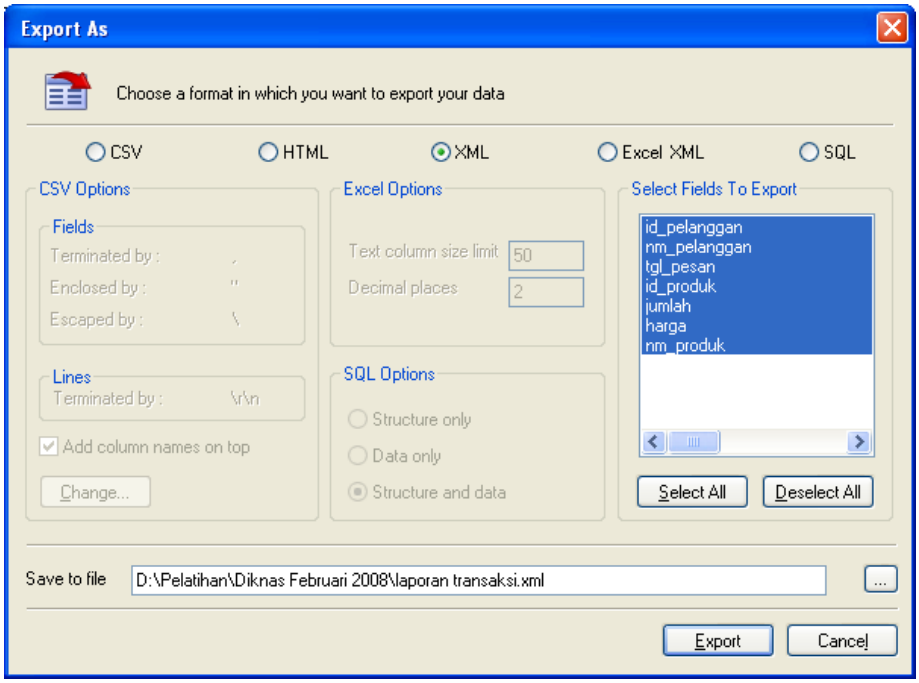
Bagaimana cara membuat laporan dalam format HTML sudah dijelaskan di bagian sebelumnya. Demikian juga tampilannya di halaman web (browser) dapat dilihat di materi sebelumnya.

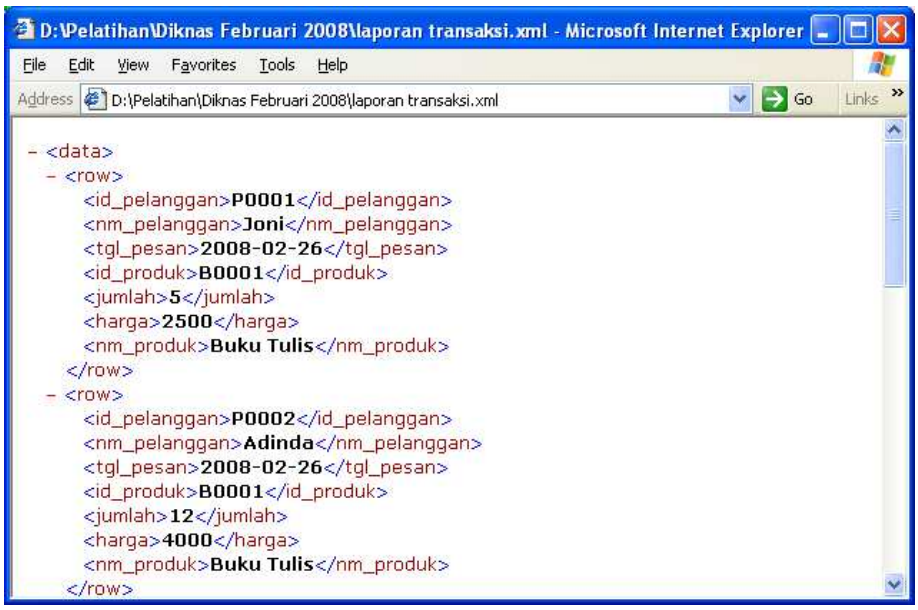
## XML

XML merupakan singkatan dari eXtensible Markup Language. Format ini sekarang sudah menjadi format baku dalam dunia komputer. Dengan XML, memungkinkan aplikasi kita dapat berkomunikasi (bertukar data) secara mudah

dengan sistem aplikasi lainnya. XML dapat dibuka dengan browser (yang mendukung XML, dan juga aplikasi lainnya seperti MS Excel 2007 dll.

Berikut ini tampilan pilihan untuk membuat report dalam XML dan juga contoh tampilan laporan dalam bentuk XML.



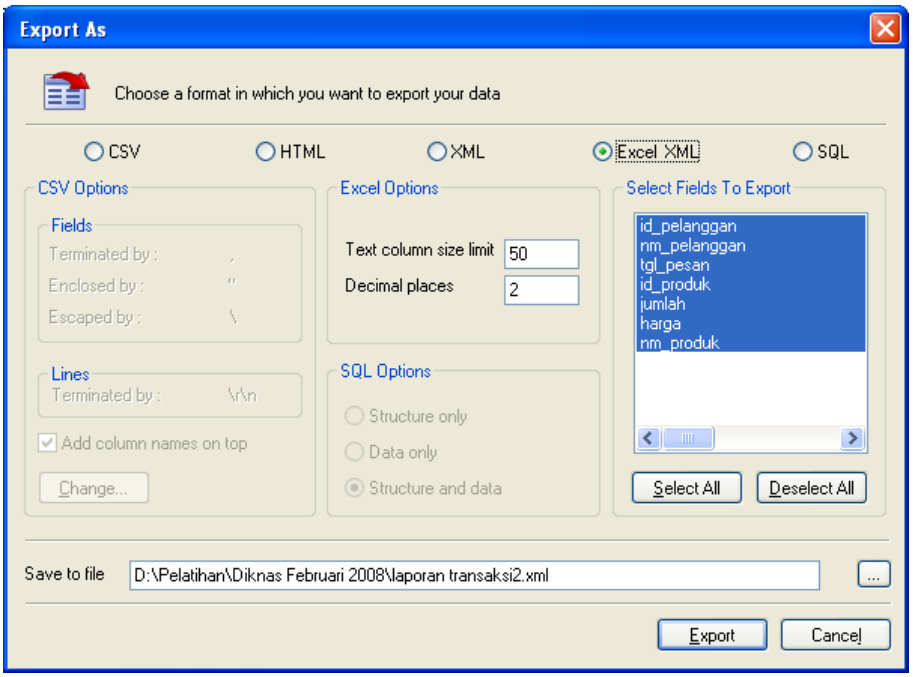




## Excel XML

Format Excel merupakan format yang paling banyak digunakan dalam pembuatan laporan. MySQLYog mendukung format ini dengan cukup baik. File

laporan yang dihasilkan merupakan file *XML-based format* yang dapat dibuka dengan MS Excel 2003 dan juga Excel 2007.

Berikut ini tampilan pilihan untuk membuat report dalam Excel XML dan juga contoh tampilan laporan dalam bentuk Excel XML (dibuka dengan MS Excel 2003).





| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | id_pelan | nm_pelan | tgl_pesan | id_produk | jumlah | harga | nm_produk | |
| 2 | P0001 | Joni | 2008-02-26 | B0001 | 5 | 2500 | Buku Tulis | |
| 3 | P0002 | Adinda | 2008-02-26 | B0001 | 12 | 4000 | Buku Tulis | |
| 4 | P0001 | Joni | 2008-02-28 | B0001 | 2 | 2500 | Buku Tulis | |
| 5 | P0001 | Joni | 2008-02-26 | B0002 | 10 | 3000 | Pulpen | |
| 6 | P0002 | Adinda | 2008-02-26 | B0003 | 10 | 3000 | Penggaris | |
| 7 | P0001 | Joni | 2008-02-26 | B0004 | 4 | 3000 | Pensil | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |

# Bab 11
# Backup, Restore dan Import di MySQL

- Backup
- Restore
- Import

## Backup

Proses backup data merupakan hal yang sangat penting dilakukan. Hal ini diperlukan untuk mengantisipasi hal-hal yang tidak diinginkan di database kita, misalnya hilangnya data, rusaknya database dsb. Sebaiknya proses backup dilakukan secara rutin dan terus-menerus.

Backup di MySQL sebenarnya ada 2 jenis, yaitu secara otomatis dan manual. Secara otomatis kita dapat menggunakan konsep *replication* , dimana server database kita secara *real-time* di-backup dengan server lain. Jika terdapat perubahan di server utama kita, maka secara otomatis perubahannya akan di-*replikasi* ke server kedua. Selain itu, kita juga dapat melakukan backup otomatis dengan bantuan software tertentu. Biasanya kita dapat membuat *schedule-backup.* Backup akan dijalankan oleh software secara otomatis setiap periode waktu tertentu.

Jenis kedua, kita dapat melakukan backup secara manual. Backup manual dilakukan oleh database administrator dengan menjalankan perintah-perintah tertentu. Backup manual ada dua bentuk, yaitu backup dalam bentuk file database dan backup dalam bentuk perintah database.

### Backup Bentuk File

Backup dalam bentuk file disini maksudnya adalah melakukan backup MySQL dengan meng-copy folder tempat database MySQL disimpan. Jika kita menggunakan sistem operasi Windows dan MySQL 5.x, secara *default* database MySQL tersimpan didalam folder **C:\Program Files\MySQL\MySQL Server 5.0\data**. Kita hanya tinggal meng-copy database yang akan kita backup dari folder tersebut ke tempat lain.

Keuntungan dari cara backup seperti ini adalah kemudahannya terutama bagi yang awam dengan database, karena tinggal *copy* dan *paste*. Namun cara

seperti ini terkadang bermasalah saat melakukan *restore* jika menggunakan sistem operasi yang berbeda.

## Backup Bentuk Perintah Database

Backup dalam bentuk perintah database sering disebut sebagai backup dalam bentuk SQL Dump. SQL Dump merupakan suatu format (file) yang berisi perintah-perintah CREATE dan INSERT yang jika dieksekusi akan menyusun struktur database dan isinya secara otomatis.

Pada dasarnya, bentuk SQL Dump dapat dilakukan dengan mengeksekusi perintah **mysqldump** atau **mysqlhotcopy** yang sudah disediakan di MySQL. Bentuk umum dari kedua perintah tersebut adalah:

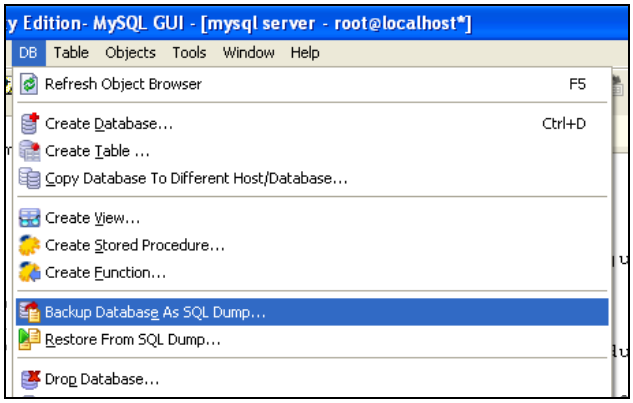
```
shell> mysqldump --tab=/path/to/some/dir --opt db_name

shell> mysqlhotcopy db_name /path/to/some/dir
```

Contoh misalnya kita akan melakukan backup semua database, sintaksnya sebagai berikut:

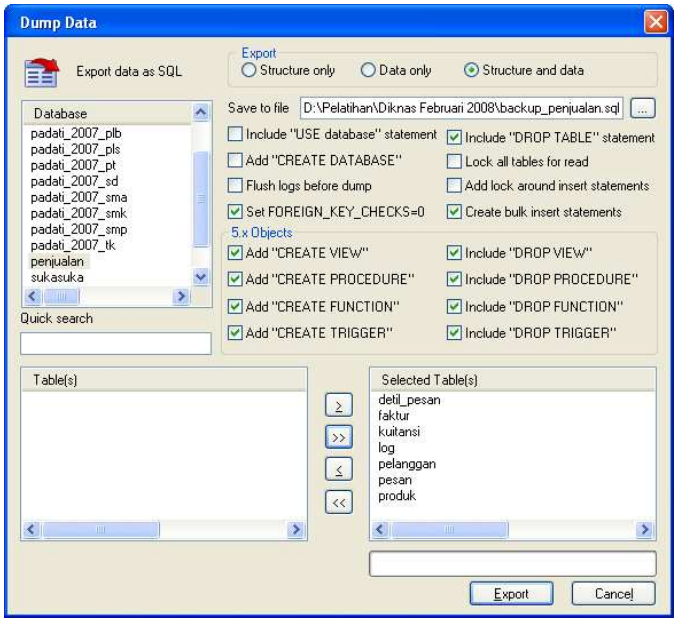
```
shell > mysqldump --all-databases > backup.sql
```

Selain dengan cara perintah manual, kita juga dapat memanfaatkan software lain seperti SQLYog. Berikut ini prosedur atau cara backup dengan memanfaatkan SQLYog.

1. Dari menu utama, pilih menu **DB > Backup Database As SQL Dump...**



2. Akan ditampilkan window "Dump Data", tentukan database yang akan dibackup, apa yang akan dibackup (data saja, strukturnya saja atau keduanya), tabel yang akan dibackup, nama file backup dan beberapa pilihan lain.




3. Tekan tombol **Export** untuk mengeksekusi backup.

Hasil backup ini berupa file dengan ekstension .SQL. File tersebut dapat di-*restore* di kemudian hari.

# Restore

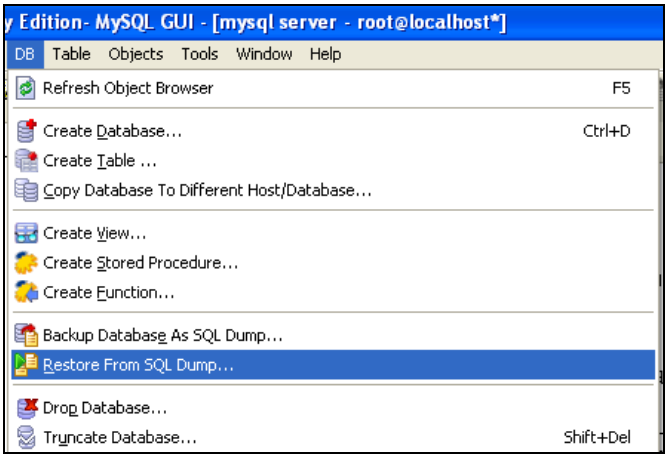
Restore merupakan prosedur yang dilaksanakan untuk mengembalikan file yang dibackup ke database MySQL. Proses restore juga dapat dilakukan melalui perintah SQL dengan memanfaatkan **mysql** dan juga bisa menggunakan software bantuan.

Jika menggunakan sintaks SQL, berikut ini contoh perintahnya:

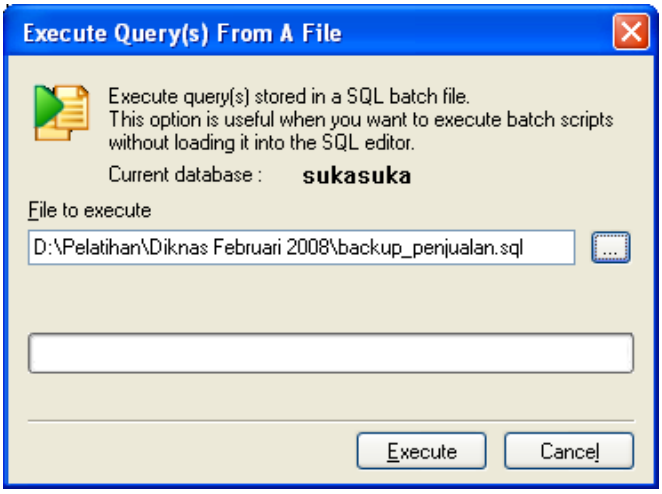
```
shell > mysql –u user –p < backup.sql
```

Sedangkan jika menggunakan SQLYog, berikut ini prosedur *restore* databasenya:

1. Dari menu utama, pilih menu **DB > Restore From SQL Dump...**




2. Akan ditampilkan window untuk memilih file backup (*.SQL).




3. Klik tombol **Execute** untuk mengeksekusi proses *restore*.

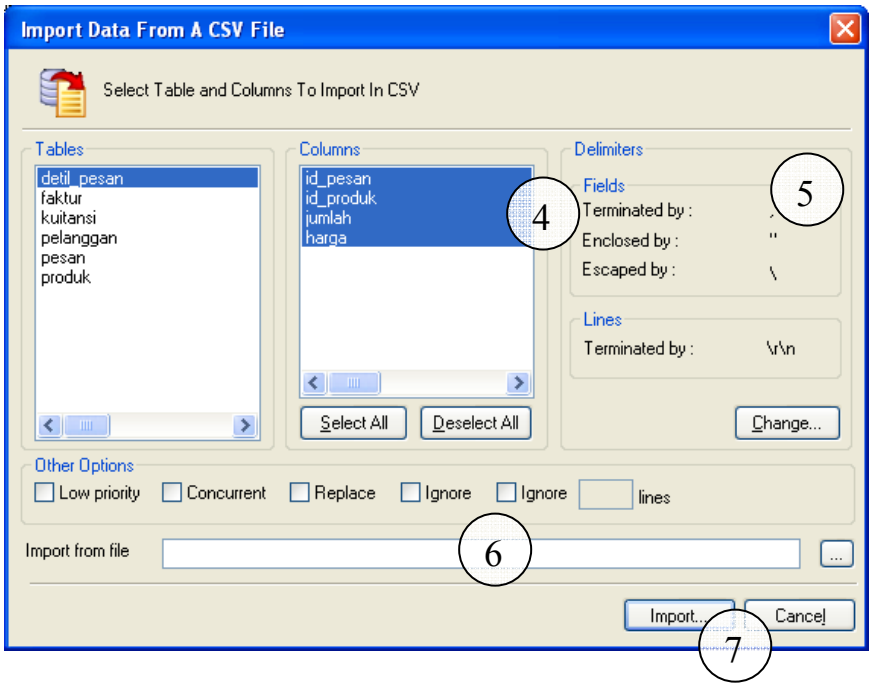
# Import

Adakalanya kita mempunyai data mentah dalam jumlah besar. Data dapat dalam bentuk Excel, CVS, SQL, atau yang lainnya. Dan jika kita ingin meng-import data tersebut, kita harus menyesuaikan format data tersebut ke format yang dikenal oleh MySQL.

Jika data dalam format .SQL maka prosedur untuk import dapat mengikuti prosedur *restore* diatas.

Berikut ini contoh prosedur import untuk format data CVS. Untuk format data yang lain, harus disesuaikan dengan format CVS.

1. Dari SQLYog, pilih (klik) tabel tujuannya.
2. Dari menu utama, pilih **Table > Import Data From CSV...**
3. Akan ditampilkan window Import Data.




4. Pilih kolom (field) tujuannya.
5. Pilih delimiter atau pemisahnya.
6. Pilih file CSV yang akan diimport
7. Klik tombol **Import...** untuk mengeksekusi perintah.

# DAFTAR PUSTAKA

Achmad Solichin. 2005. **Pemrograman Web dengan PHP dan MySQL**. Jakarta.

Allen G. Tailor. 2003. **SQL For Dummies, 5th Edition**. Wiley Publishing, Inc.

Charler A. Bell. 2007. **Expert MySQL**. Apress Publishing: New York.

Derek J. Balling, Jeremy Zawodny. 2004. **High Performance MySQL**. O'Reilly Publishing.

George Reese. 2003. **MySQL Pocket Reference.** O'Reilly Publishing.

Marc Delisle. 2009. **Mastering phpMyAdmin 3.1 for Effective MySQL Management**. Packt Publishing: Birmingham

Mark Maslakowski. 2000. **Sam's Teach Yourself MySQL in 21 Days**. Sams Publishing.

Michael Kofler. 2005. **The Definitive Guide to MySQL 5 third edition**, Apress Publishing: New York

MySQL. **Situs MySQL**. http://mysql.com. diakses 29 Januari 2010

MySQL. **MySQL Manual**. http://mysql.com.

Steven Feuerstein, Guy Harrison. 2006. **MySQL Stored Procedure Programming**. O'Reilly Publishing

Wikipedia. **MySQL on Wikipedia**. http://en.wikipedia.org/wiki/MySQL, diakses 29 Januari 2010

# TENTANG PENULIS

**Achmad Solichin.** Adalah Lulusan Teknik Informatika, Fakultas Teknologi Informasi, Universitas Budi Luhur, Jakarta (S1, 2005). Saat ini sedang menempuh pendidikan S2 di Magister Teknologi Informasi Universitas Indonesia (2008). Kegiatan sehari-hari adalah sebagai Dosen di Universitas Budi Luhur (http://www.bl.ac.id), sekaligus sebagai Kepala Laboratorium Komputer Universitas Budi Luhur (http://labkom.bl.ac.id). Kegiatan lain aktif sebagai programmer, web developer, system analyst dan memberikan pelatihan di berbagai bidang komputer serta membuat tutorial-tutorial praktis di bidang komputer. Penulis juga terlibat dalam pengembangan E-Learning di Universitas Budi Luhur. Penulis memiliki situs utama di http://achmatim.net yang berisi berbagai tutorial komputer dan buku gratis. Penulis dapat dihubungi melalui email di achmatim@gmail.com, YM achmatim, Facebook[2] dan Twitter[3].

Beberapa website yang dikembangkan oleh penulis:

- http://achmatim.net, tutorial komputer dan buku belajar gratis
- http://contohprogram.info, kumpulan contoh program dan aplikasi komputer
- http://ebook.achmatim.net, free computer ebook collections
- http://elearning.achmatim.net, situs kuliah online
- http://bestfreetutorial.com, best free tutorial for newbie
- http://hotnewsarchives.info, data mining, data warehouse, and knowledge management
- http://banksoalkomputer.info, bank soal dan latihan untuk universitas dan perguruan tinggi komputer
- http://banksoalujian.info, bank soal ujian untuk SMP dan SMA
- http://sman1sumpiuh.com, situs SMA Negeri 1 Sumpiuh

[2] http://facebook.com/achmatim
[3] http://twitter.com/achmatim