

RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

Diera Globalisasi Teknologi Informasi sekarang ini peran Data base Menejemen sistem (DBMS) sangatlah penting. Pemrosesan DBMS menjadi perangkat andalan yg kehadirannya sangat diperlukan oleh berbagai institusi dan perusahaan, untuk mengolah data menjadi terpusat sehingga memudahkan informasi. Oleh karena itu diperlukan teknik komputasi handal yang bisa memberikan informasi secara cepat dan akurat. Buku ini memberikan gambaran secara jelas dan mudah bagaimana memahami masalah terkait dengan DBMS. Selain mempercepat pemerolehan informasi juga dapat meningkatkan pelayanan. MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multithread, multi-user.

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.

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.

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

Mendukung transaksi antar tabel, Mendukung row-level-locking, Mendukung Foreign-Key Constraints, Crash recovery.

Tipe Tabel 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.



RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

Endang Setyawati, M.Kom, dkk

Endang Setyawati, M.Kom
Dr. Ir. H. Sarwani, M.T., M.M.
Nadion Wijoyo, S.E., S.H., S.Sos., S.Pd., M.H., M.M., Ak., CA., QWP®, CPHCM®, C.PS®
Nyoto Soehamoko, S.Kom.

RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)



RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

Endang Setyawati, M.Kom.

Dr. Ir. H. Sarwani, M.T., M.M.

**Hadion Wijoyo, S.E., S.H., S.Sos., S.Pd., M.H., M.M., Ak.,
CA., QWP®, CPHCM®, C.PS®**

Nyoto Soeharmoko, S.Kom.



pena persada

PENERBIT CV. PENA PERSADA

**RELATIONAL DATABASE
MANAGEMENT SYSTEM (RDBMS)**

Penulis :

Endang Setyawati, M.Kom.
Dr. Ir. H. Sarwani, M.T., M.M.
Hadion Wijoyo, S.E., S.H., S.Sos., S.Pd., M.H., M.M., Ak.,
CA., QWP®, CPHCM®, C.PS®
Nyoto Soeharmoko, S.Kom.

ISBN : 978-623-6837-14-6

Cover Design:

Retnani Nur Brilliant

Layout :

Nisa Falahia

Penerbit CV. Pena Persada

Redaksi :

Jl. Gerilya No. 292 Purwokerto Selatan, Kab. Banyumas
Jawa Tengah

Email : penerbit.penapersada@gmail.com

Website : penapersada.com

Phone : (0281) 7771388

Anggota IKAPI

All right reserved

Cetakan pertama : 2020

Hak Cipta dilindungi oleh undang-undang. Dilarang
memperbanyak karya tulis ini dalam bentuk apapun tanpa izin
penerbit.

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 MySQL¹.

Buku berjudul "**Relational Database Management System**" 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

¹ <http://www.mysql.com>

dijelaskan mengenai penyajian laporan dan proses *backup, restore* database MySQL.

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.

Penulis,

Endang Setyawati, M.Kom.

DAFTAR ISI

KATA PENGANTAR	iii
DAFTAR ISI	v
BAGIAN I PENDAHULUAN	vi
Bab 1. Sekilas Tentang MySQL	1
Bab 2. Instalasi MySQL dan Software Pendukung	68
BAGIAN II DASAR-DASAR MySQL	84
Bab 3. Merancang Database	85
Bab 4. Dasar-dasar SQL	94
Bab 5. Fungsi-fungsi MySQL	113
BAGIAN III PERINTAH MySQL LANJUTAN	129
Bab 6. Perintah MySQL Lanjutan	130
Bab 7. Administrasi dan Keamanan di MySQL	149
Bab 8. Trigger dan Views	156
Bab 9. Function dan Stored Procedure	162
BAGIAN IV LAPORAN DI MySQL	172
Bab 10. Laporan di MySQL.....	173
Bab 11. Backup, Restore dan Import di MySQL	181
DAFTAR PUSTAKA	187

BAGIAN I

PENDAHULUAN

BAB 1.

PENGENALAN BASIS DATA

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

A. Pengenalan Database, DBMS dan RDBMS

Sekarang ini peran basis data sangatlah penting. Pemrosesan basis data menjadi perangkat andalan yg kehadirannya sangat diperlukan oleh berbagai institusi dan perusahaan. Basis data tidak hanya mempercepat pemerolehan informasi tetapi juga meningkatkan pelayanan .

Basis data dapat diartikan sebagai kumpulan data tentang suatu benda / kejadian yg saling berhubungan satu sama lain. Sedangkan data merupakan fakta yg mewakili suatu obyekseperti manusia hewan yg dapat dicatat dan mempunyai arti yg implisit . Data dicatat/rekam dalam bentuk angka huruf simbol gambar bunyi/kombinasinya.

Basis data merupakan penyajian suatu aspek dari dunia nyata. Basis data merupakan kumpulan data dari berbagai sumber yg secara logika mempunyai arti implisit. Basis data perlu dirancang dibangun dan data dikumpulkan untuk suatu tujuan.

Kelebihan dan kelemahan penggunaan pendekatan basis data antara lain :

1. Keuntungan pendekatan basis data
 - a. Pemusatan kontrol data
 - b. Pemakaian data bersama
 - c. Data yg bebas
 - d. Kemudahan dalam pembuatan program aplikasi baru
 - e. Pemakaian secara langsung
 - f. Data yg berlebihan dapat dikontrol

2. Kelemahan pendekatan basis data

Kelemahan yg ada pada pendekatan basis data

- a. Biaya Mahal
- b. Sangat komplek
- c. Resiko Data yang tepusat

Gambaran Basis data dan Penerapanya

- a. Dalam dunia pendidikan
 - 1) Berapakah jml mahasiswa yg mengikuti kuliah basis data.
 - 2) Siapakah yang akan lulus periode september bulan ini
 - 3) Berapa presentase mhs yg tidak melakukan registrasi pada semester ini
 - 4) Berapakah jumlah SKS yg diperoleh oleh mhs yang bernama Tuti.
- b. Dalam Perusahaan kridit Mobil
 - 1) Berapakah Pelanggan harus melunasi kridit mobil setelah menunggak dua bulan
 - 2) Berapa persen dia harus membayar dendanya
 - 3) Dan Sampai kapan dia harus lunas/ selesai.

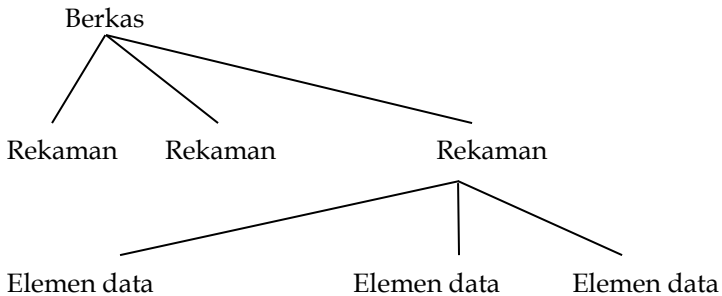
B. Data dan Informasi

Sebelum membahas istilah basis data terlebih dahulu kita mengetahui istilah data dan informasi. Data adalah fakta mengenai obyek, data dinyatakan dengan nilai (angka, deretan karakter atau simbol), sejumlah penulis menggunakan data untuk menyatakan nilai yg secara aktual terkandung dalam basis data sedangkan informasi digunakan untuk menyatakan makna nilai ketika dipahami oleh pengguna.

Informasi adalah hasil analisis dan sintesis terhadap data dengan kata lain informasi dapat dikatakan sebagai data yang telah diorganisasikan kedalam bentuk yang sesuai dengan kebutuhan seseorang .

Dalam sitem basis data relasional, berkas mewakili komponen yg disebut tabel atau relasi

Dalam sistem basis data relasional berkas mewakili komponen yang disebut tabel/relasi gambar hirarki data



1. Apakah Basis data itu sebenarnya?
 - a. Istilah Basis data menurut Chou[10] adalah kumpulan informasi bermanfaat yang diorganisasikan kedalam tatacara yg khusus.
 - b. Menurut Febbri dan Schwab [1] basis data adalah sistem berkas terpadu yang dirancang terutama untuk meminimalkan pengulangan data.
 - c. Menurut Date[3] basis data dianggap sebagai tempat untuk sekumpulan berkas data terkomputerisasi. Yg pada dasarnya adalah sistem komputerisasi yg tujuan utamanya adlah memelihara informasi dan membuat informasi tsb tersedia saat dibutuhkan.

2. Pengguna Sistem Basis Data bisa melakukan berbagai operasi al:
 - a. Menambah file baru kesistem basis data
 - b. Mengosongkan berkas
 - c. Menyisipkan data ke sutu berkas
 - d. Mengambil data yg ada pada suatu berkas
 - e. Mengubah data pada suatu berkas
 - f. Menghapus data pada suatu berkas
 - g. Menyajikan suatu informasi yg diambil dari sejumlah berkas

3. Sejarah Kemunculan Basis Data

- a. Menurut sejarah sistem pemrosesan basis data terbentuk setelah masa sistem pemrosesan manual dan sistem pemrosesan berkas.
- b. Sistem pemrosesan basis data dimaksudkan untuk mengatasi kelemahan-kelemahan yang ada pada sistem pemrosesan berkas . Sistem ini dikenal dengan sebutan DBMS (database management system) .

4. Evolusi teknologi Basis data

Perkembangan teknologi basis data tidak lepas dari perkembangan perangkat keras dan perangkat lunak. Perkembangan teknologi jaringan komputer dan komunikasi data merupakan salah satu penyumbang kemajuan penerapan basis data, yg kmd melahirkan sistem basis data terdistribusi. ATM / anjungan tunai mandiri yg banyak diterapkan pada perbankan di Indosesia.

- a. Kecepatan : Mesin dapat mengambil atau mengubah data jauh lebih cepat dari pada manusia
- b. Mengurangi kejemuian : orang cenderung menjadi bosan kalau melakukan hal yang sama berulang-ulang.
- c. Kekinian : informasi yang tersedia pada DBMS akan bersifat muntahir dan akurat setiap saat.

1. Komponen Utama DBMS dibagi Menjadi 4 Macam :

- a. Perangkat keras
- b. Data
- c. Perangkat lunak
- d. Pengguna

Perangkat keras: berupa komputer dan bagian-bagian didalamnya spt prosesor memori dan hardisk hal ini digunakan untuk melakukan pemprosesan dan juga untuk menyimpan basis data.

Data: didalam basis data mempunyai sifat terpadu (berarti bahwa berkas-berkas data yg ada pada basis data saling terkait dan berbagi) data dapat dipakai oleh sejumlah pengguna dalam waktu bersamaan. Sifat ini biasa terdapat pada sistem multi user kebalikan dari single user. Hanya memungkinkan satu orang yg bisa mengakses suatu data pada suatu waktu.

Perangkat lunak: berkedudukan antara basis data / dt yg disimpan dalam hardisk dan pengguna PL inilah yg berperan melayani permintaan-permintaan pengguna.

Pengguna dibagi menjadi 3 kategori :pengguna akhir , pemrogram aplikasi, administrator basis data pengguna akhir dibagi menjadi 2 yaitu pengguna aplikasi (orang mengoperasikan program aplikasi yg dibuat oleh pemrogram) dan pengguna interaktif adl orang yg dpt memberikan perintah beraras tinggi pd antar muka basis data yg tersedia.

DBA/Data base administrator (Administrator Basis Data adl orang yang bertanggung jawab terhadap pengelolaan basis data).

2. Modelbasis Data

Model basis data menyatakan hubungan antar rekaman yang tersimpan dalam basis data . Beberapa literatur menggunakan istilah struktur data logis untuk menyatakan keadaan ini.

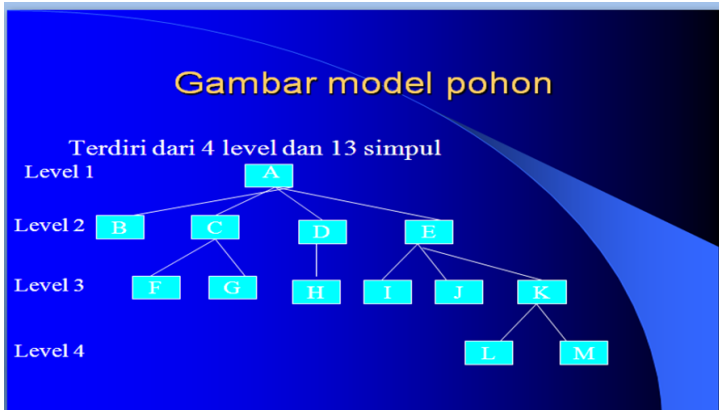
- a. Hirarkis
- b. Jaringan
- c. Relasional

C. Model Hirarkis

1. Model Pohon

Menggunakan pola hubungan orang tua anak. Setiap simpul (biasa dinyatakan dengan lingkaran atau kotak) menyatakan sekumpulan medan. Simpul yang terhubung ke simpul pada level dibawahnya disebut orang tua. Setiap orang tua bisa memiliki satu hubungan 1:1 atau beberapa

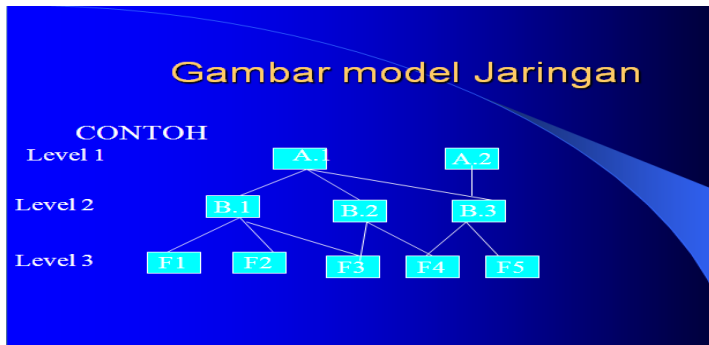
anak hubungan 1: M tetapi setiap anak hanya memiliki satu orang tua . Simpul-simpul yg dibawahhi oleh simpul orang tua disebut anak. Simpul orang tua yg tidak memiliki disebut akar . Simpul yg tak memiliki anak disebut daun. Adapun hubungan antara anak dan orang tua disebut cabang . Contoh gambar dibawah ini terdiri atas 4 level dan 13 simpul.



-A berkedudukan sebagai akar dan berkedudukan sebagai orang tua dari simpul BCDE, keempat simpul ini disebut sebagai anak simpul A. Dan C bisa berkedudukan sebagai orang tua yaitu orang tua dari FG. Adapun simpul FGHIJLM disebut sebagai daun.

2. Model Jaringan

Disentrararisasikan pada tahun 1971 oleh Data Base Task Group (DBTG). Disebut juga dg model CODASYL (Conference on Data Systems Languages). Karena DBTG mrp bagian dari CODASYL. Model jaringan hampir sama dg model Hirarkis dengan perbedaan suatu simpul anak bisa memiliki lebih dari satu orang tua.



Contoh produk DBMS yg menggunakan model jaringan adalah CAIDMS/DB dari Computer Associates International Inc sebelumnya dikenal sebagai IDMS (Integrated Database Management System yg dikembangkan oleh Cullinet Software Inc).

3. Model Relasional

Model relasi ini mirip model yang paling sederhana sehingga mudah digunakan dan dipahami oleh pengguna serta merupakan yg paling populer saat ini. Yaitu menggunakan sekumpulan tabel berdimensi dua (yang disebut relasi atau tabel) dengan masing-masing relasi tersusun atas tupel/baris dan atribut. Relasi ini dirancang utk menghilangkan kemubaziran data dan menggunakan kunci tamu untuk berhubungan dg relasi lain.

Contoh Model Relasional

NAMA DOSEN	KELAS	MAHASISWA
Axl Adilla	Basis Data	Weli
Axl Adilla	Basis Data	Reni
Axl Adilla	Basis Data	Abdul
Axl Adilla	Desain grafis	Weli
Axl Adilla	Desain grafis	Abdul

4. Contoh Database Relational

NIM	NAMA_MHS	KODEMK	NAMA_MK
001	Weli	KDB01	DATABASE
002	Reni	KMT02	MATEMATIKA
003	Abdul	KDG03	DESAIN GRAFIS

NIM_MHS	KODE_MK	NILAI
001	KDB01	A
001	KMT02	A
001	KDG03	A
002	KDB01	B
003	KDB01	A
003	KDG03	A

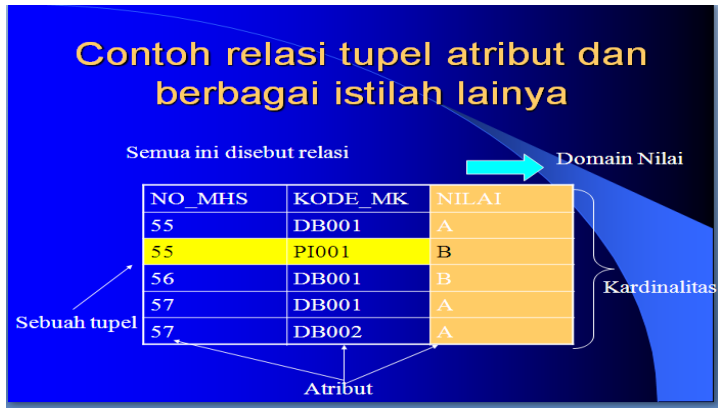
Ada beberapa sifat yg melekat pd suatu relasi

- Tak ada tupel /baris yang sama
- Urutan tupel tidaklah penting
- Setiap atribut memiliki nama yg unik
- Letak atribut bebas
- Setiap atribut memiliki nilai tunggal dan jenisnya sama untuk semua tupel

Pada model relasional jml tupel suatu relasi disebut kardinalitas dan jml atribut suatu relasi disebut derajat/degree atau arity. Relasi yg berderajat satu hanya memiliki satu atribut disebut unary, yg berderajat dua disebut binary yg berderajat tiga disebut ternary.

Istilah lain yg terdapat pada model relasional adalah domain. Domain adalah himpunan nilai yg berlaku bagi suatu atribut.

Bagian yg menyebabkan tidak adanya tupel yg kembar adl yg disebut kunci primer.



5. Macam Macam Perintah DBMS

Semua DBMS paling tidak mempunyai dua macam perintah yg digunakan utk mengelola dt mengorganisasikan data yaitu:

a. Perintah DDL

- 1) Bahasa Definisi Data (DDL/Data Definition Language)
- 2) Bahasa Manipulasi Data (DML/Data Manipulation Language)
- 3) DCL (Data Control Language) yg berkaitan dg pengaturan sekuritas terhadap basis data
- 4) DDL adalah perintah yg biasa digunakan oleh administrator basis data (DBA) utk mendefinikan skema ke DBMS.
- 5) Skema adalah diskripsilengkap tentang struktur medan rekaman dan hubungan data pada basis data. DDL juga digunakan utk menciptakan , mengubah, menghapus, basis data.

b. Perintah DML

DML adalah perintah yg biasa digunakan untuk mengubah memanipulasi dan mengambil data pada basis data. DML dibagi menjadi dua :

- 1) Prosedural yg menuntut pengguna menentukan data apa saja yg diperlukan dan bagaimana cara mendapatkannya.
- 2) Nonprosedural yg menuntut pengguna menentukan data apa saja yg diperlukan tetapi tidak perlu menyebutkan cara mendapatkannya.

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

- 1) DB2 - <http://www-306.ibm.com/software/data/db2>
- 2) Microsoft SQL Server - <http://www.microsoft.com/sql/>
- 3) Oracle - <http://www.oracle.com>
- 4) Sybase - <http://www.sybase.com/>
- 5) Interbase - <http://www.borland.com/interbase>
- 6) Teradata - <http://www.teradata.com/>
- 7) Firebird - <http://www.firebirdsql.org/>
- 8) MySQL - <http://www.mysql.com>
- 9) PostgreSQL - <http://www.postgresql.org/>

6. Bekerja Dengan Microsoft Access

Microsoft Access merupakan salah satu program aplikasi database. Satu Database Management System (DBMS) berisi satu koleksi data yang saling berelasi dan satu set program untuk mengakses data tersebut. Data base adalah kumpulan dari file-file yang saling berelasi, relasi tersebut biasanya ditunjukkan dengan pemakaian kunci dari tiap file yang ada, atau kumpulan file yang mempunyai kaitan antara satu file dengan file yang lain sehingga membentuk satu bangunan data untuk memberikan informasi tertentu. Basis data juga bisa diartikan sebagai kumpulan data tentang suatu benda atau kejadian yang

saling berhubungan satu sama lain. Database relasi biasanya disusun dalam bentuk kolom dan baris. Kolom disebut juga sebagai field yang biasa dipakai untuk pengelompokan jenis data, misalnya nama, alamat, kota. Baris disebut sebagai record biasanya dipakai untuk membedakan satu set data dengan data lain, atau dengan kata lain sebagai berikut :

Database

Database adalah sekelompok tabel data berisi informasi-informasi yang saling berhubungan. Suatu database dapat terdiri dari satu atau lebih tabel.

Tabel

Tabel adalah tempat untuk menampung data atau sekelompok record data, masing - masing berisi informasi yang sejenis.

Record

Record adalah sebuah entri tunggal dalam tabel, entri tersebut terdiri dari sejumlah field data.

Field

Field adalah item tertentu dari data dalam record. Setiap satu informasi diletakkan pada sebuah field. Ketika pembuatan tabel, harus ditetapkan tipe, panjang maksimum, dan atribut lainnya untuk sebuah field.

Indeks

Indeks adalah tipe tabel tertentu yang berisi nilai-nilai field tertentu (ditetapkan oleh pemakai) dan disimpan dalam urutan tertentu (juga ditetapkan oleh pemakai)

Query

Query adalah perintah SQL yang dirancang untuk memanggil kelompok record tertentu dari satu tabel atau lebih untuk melakukan operasi pada tabel.

Filter

Filter digunakan bersama urutan indeks dan sort untuk menentukan data mana yang diproses atau ditampilkan.

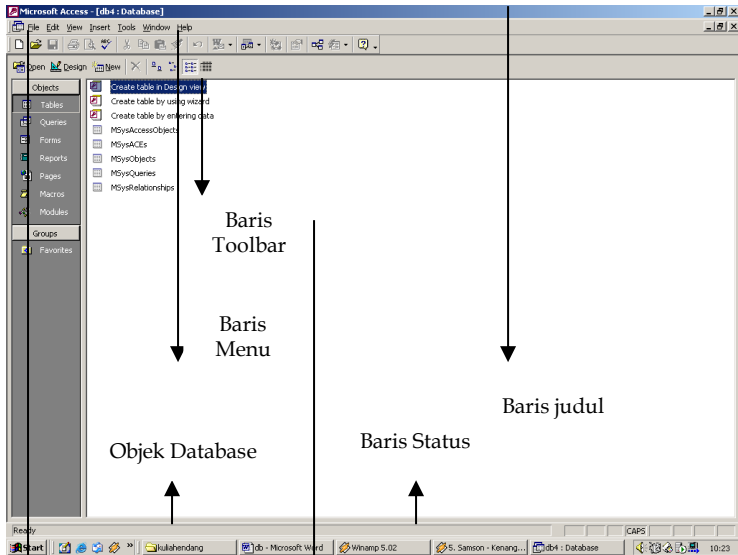
View

View data terdiri atas jumlah record yang tampak (diproses) dan urutan penampilannya. View khususnya dikendalikan oleh filter dan indeks.

Tabel Mahasiswa

NIM	NAMA	ALAMAT
001	Axl Adilla	Purwokerto
002	Dina Astri Megawati	Semarang
003	Satrio	Tegal
004	Ikfina Eka Putri Agustin	Banjarnegara
005	Devia Ramayani	Surabaya

Field



Icon Kontrol Menu
Jendela Database

7. Bekerja dengan Tabel

Tabel adalah tempat untuk menampung data. Sebuah tabel memiliki sejumlah field yang berisi informasi yang spesifik. Misalnya tabel mahasiswa memiliki field NPM, Nama Mahasiswa, Telp, dan sebagainya. Sebelum bekerja dengan tabel, Sebuah tabel harus dirancang terlebih dahulu, dengan menentukan jumlah field, nama tiap field dan jenis data yang akan dimasukkan pada masing - masing field. Tiap field hanya bisa menerima satu jenis data saja. Ada sembilan jenis data untuk field, yaitu :

Text

Jenis data text digunakan untuk menyimpan data yang tidak membutuhkan perhitungan. Field text dapat menampung sampai dengan 255 karakter, dan ukuran defaultnya adalah 50 karakter.

Memo

Jenis data memo digunakan untuk menyimpan data teks yang panjangnya lebih dari 255 karakter. Memo dapat menampung sampai dengan 64.000 karakter. Memo tidak dapat diindeks atau disortir.

Number

Jenis data number digunakan untuk menyimpan nilai numeris untuk perhitungan matematis.

Date/Time

Digunakan bagi field yang menampung data tanggal atau jam.

Currency

Digunakan untuk menampung bilangan. Bilangan yang disimpan dalam field jenis currency tidak akan terkena pembulatan pada saat perhitungan.

AutoNumber

Digunakan untuk membuat field yang secara otomatis akan memasukkan bilangan pada saat kita menambahkan sebuah record.

Yes/No

Membuat field untuk menampung dua jenis keadaan, ya atau tidak.

OLE Object

Membuat field untuk menyimpan objek OLE, file dokumen, gambar, dan file lain yang dibuat dalam program lain.

Lookup Wizard

Dengan menggunakan Lookup Wizard, kita dapat membuat field yang menampilkan salah satu dari dua macam daftar yang memudahkan pemasukan data.

- a. Daftar Lookup, menampilkan nilai-nilai yang dirujuk dari sebuah tabel atau query yang telah ada.
- b. Daftar nilai, menampilkan kumpulan nilai yang tidak bisa diubah yang dimasukkan pada saat membuat field ini.

a. Membuat tabel baru

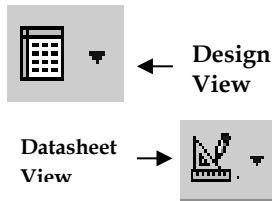
Untuk membuat tabel baru :

1. Dari jendela **Tables**, klik **New** atau klik tombol shortcut **New Object**, kemudian pilih **New Table**.
2. Pada kotak dialog **New Table**, sorot **Design View**, lalu klik **OK**.



3. Pada kotak dialog yang muncul, ada tiga kolom yang dapat diisi, yaitu **Field Name** untuk nama field, **Data Type** untuk jenis data dari field dan **Description** untuk mengisikan keterangan yang dapat membantu menjelaskan fungsi dari field yang bersangkutan (optional)
4. Isi semua data yang diperlukan, kemudian simpan dengan memilih **Save** dari menu **File**. Kotak dialog **Save As** akan muncul, ketikkan nama tabel.
5. Klik **OK**.

b. Menambahkan Field



Untuk menambahkan field pada tabel yang telah dibuat :

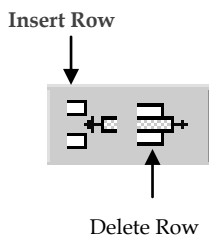
1. Dari jendela **Tables**, sorot tabel yang akan mendapat tambahan field.
2. Klik **Design**.
3. Bawa kursor ke tempat yang masih kosong untuk menambahkan field.
4. Tekan **Ctrl+S**.

Untuk mengisi tabel, dari tampilan **design**, dapat langsung berpindah ke tampilan **datasheet** dengan mengklik tombol shortcut **datasheet View**, dan dari tampilan **datasheet** dapat kembali ke tampilan **design** dengan mengklik tombol shortcut **design view**.

c. Menghapus Field

Untuk menghapus field, tabel dalam tampilan design terlebih dahulu harus dibuka, kemudian sorot field yang ingin dihapus pada kolom field name. Dari menu Edit, klik Delete Row atau klik tombol shortcut Delete Row pada toolbar.

d. Menyisipkan Field



Untuk menyisipkan field :

1. Buka tabel pada tampilan **design**, lalu bawa kursor ke tempat dimana field baru akan disisipkan.
2. Pada menu **Insert Field** atau klik tombol **Insert Row** pada toolbar.
3. Isikan nama dan jenis data field yang baru ke baris kosong yang muncul.
4. Simpan.

e. Mengganti Nama Tabel

Untuk mengganti nama tabel :

1. Sorot tabel dengan mengklikkan pointer mouse
2. Pilih Rename dari menu Edit.
3. Di kotak dialog yang muncul, ketikkan nama tabel yang baru.
4. Setelah selesai, tekan enter atau klik di mana saja di luar kotak dialog.

f. Menghapus Tabel

Untuk menghapus sebuah tabel, sorot nama tabel yang akan dihapus pada jendela **Tables** lalu tekan tombol **Del**. Penghapusan juga dapat dilakukan dengan memilih **Delete** dari menu **edit**.

g. Menghubungkan Tabel

Tabel-tabel dalam database MS-Access dapat saling memiliki hubungan (*relationship*). Adanya kemungkinan untuk membuat hubungan inilah yang menjadikan MS-Access lebih terasa gunanya.

h. Primary Key

Kekuatan sistem database relasional terletak pada kemampuannya untuk secara cepat menemukan dan menampilkan informasi yang disimpan dalam table-table terpisah dengan menggunakan query, form dan report. Untuk dapat melakukannya dengan cepat, tiap tabel seharusnya menyertakan sebuah atau beberapa field yang secara unik menandai tiap record yang disimpan dalam tabel. Field unik ini disebut *primary key* dari tabel itu. Jadi *primary key* adalah :

Suatu field atau kombinasi field yang secara unik mengidentifikasi sebuah record sekaligus membedakannya dengan record yang lain. Biasanya yang

sering digunakan adalah field berisi nomor identitas yang tidak akan pernah sama untuk tiap orang.

Sekali primary key sebuah tabel ditentukan untuk menjamin keunikan, MS-Access tidak akan mengizinkan untuk menyimpan nilai yang sama atau nilai kosong (*null*) di dalam field *primary key*.

Ada tiga jenis primary key yang dapat didefinisikan dalam MS-Access, yaitu primary key AutoNumber, satu field dan banyak field.

1) Primary Key AutoNumber

Field dari jenis AutoNumber dapat diatur agar secara otomatis memasukkan bilangan berurutan pada saat tiap record ditambahkan pada tabel itu.

2) Primary Key satu Field

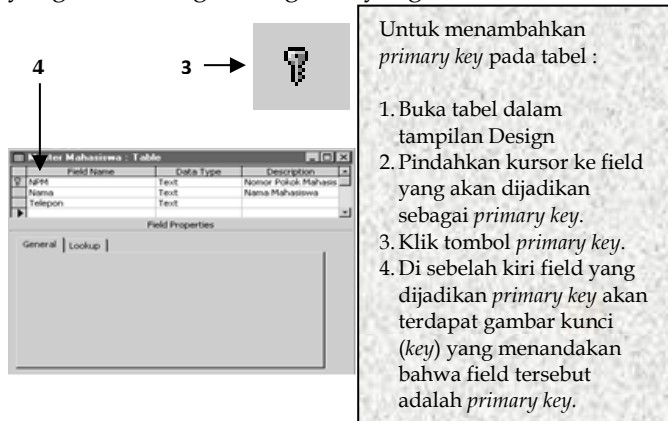
Jika terdapat sebuah field yang berisi nilai - nilai unik seperti NPM, maka field tersebut dapat ditetapkan sebagai *primary key*.

3) Primary Key banyak Field

Bila terdapat suatu ketidakpastian terhadap keunikan suatu field, dapat ditetapkan dua field atau lebih sebagai *primary key*.

4) Menambahkan Primary Key

Sebelum menambahkan *primary key* ke sebuah tabel, harus ditentukan terlebih dahulu field unik yang tidak mengandung data yang sama.



4

3 →

Field Name	Data Type	Description
NPM	Text	Nomor Pokok Mahasiswa
Nama	Text	Nama Mahasiswa
Telepon	Text	

Field Properties

General | Lookup

Untuk menambahkan *primary key* pada tabel :

1. Buka tabel dalam tampilan Design
2. Pindahkan kursor ke field yang akan dijadikan sebagai *primary key*.
3. Klik tombol *primary key*.
4. Di sebelah kiri field yang dijadikan *primary key* akan terdapat gambar kunci (*key*) yang menandakan bahwa field tersebut adalah *primary key*.

Setelah membuat *primary key*, harus ada jaminan bahwa nilai yang dimasukkan ke field tersebut unik, tidak boleh ada data yang persis sama dan bukan nilai kosong (*null*).

5) Menghilangkan Primary Key

Untuk menghilangkan *primary key* :

1. Buka tabel dalam tampilan design, lalu bawa kursor ke field *primary key* yang akan dihapus.
2. Klik tombol *primary key*.

6) Hubungan di antara dua tabel

Hubungan di antara dua tabel bekerja dengan mencocokkan data dalam field - field kunci, biasanya sebuah field dengan nama yang sama di kedua tabel. Umumnya

digunakan *primary key* salah satu tabel sebagai field kunci.

Tabel tempat field *primary key* ini disebut tabel primer. *Primary key* yang menyediakan pengidentifikasian unik, dicocokkan dengan entri yang sesuai dalam tabel lain. Entri yang sesuai ini disebut *Foreign key*.

Antara dua buah tabel dapat terjadi hubungan satu ke satu, satu ke banyak atau banyak ke banyak.

7) Hubungan satu ke Satu (One to One)

Dalam hubungan ini, tiap record dalam tabel A memiliki satu record yang cocok dalam tabel B dan tiap record dalam tabel B memiliki satu record yang cocok dalam tabel A.

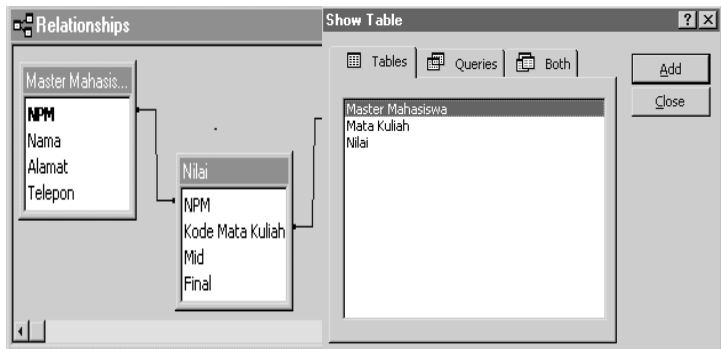
8) Membuat hubungan satu ke satu

Sebelum membuat hubungan satu ke satu, pastikan tabel - tabel yang akan dihubungkan telah mempunyai *primary key*.

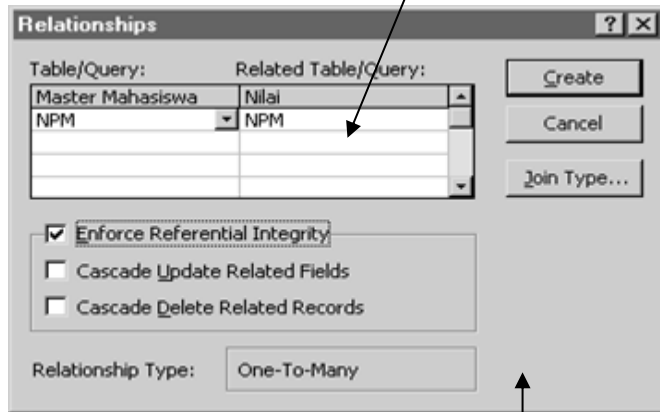


Untuk membuat hubungan satu ke satu :

1. Dari menu **Tools**, klik **Relationship**, atau klik tombol **Relationships**.
2. Jendela **Relationships** akan tampil. Klik tombol **Show Table** pada toolbar untuk menampilkan jendela **show table**.
3. Pada jendela **show table**, sorot tabel - tabel yang akan dihubungkan, lalu klik **Add**.
4. Tutup kotak dialog **Show Table** dengan mengklik **Close**.
5. Buatlah hubungan dengan menyeret field unik tabel pertama ke field yang sama jenisnya pada tabel kedua. Akan muncul kotak dialog **Referential Integrity** yang meminta konfirmasi.



Ada dua buah kolom pada kotak dialog Relationships. Kolom pertama memuat nama field dari tabel utama, kolom kedua memuat nama field yang terhubung ke tabel utama.



Jenis hubungan yang terjadi ditunjukkan dalam kotak Relationship Type.

Pada kotak dialog Relationship juga terdapat kotak periksa Enforce Referential Integrity. Kotak periksa ini harus diberi tanda bila ingin menerapkan *referential integrity*.

9) Referential Integrity

Referential Integrity adalah suatu sistem peraturan untuk menjamin *validitas* hubungan antara sejumlah record dalam table-table yang terkait.

Referential Integrity hanya bisa diterapkan bila beberapa syarat berikut terpenuhi :

- a) Field yang cocok dari tabel primer adalah primary key atau memiliki satu indeks yang unik.
- b) Field - field yang terkait memiliki jenis data yang sama.
- c) Kedua tabel yang terkait berada dalam database yang sama.

Peraturan yang harus dipenuhi bila kita menerapkan referential Integrity :

- a) Tidak boleh memasukkan nilai dalam field foreign key dari tabel yang terhubung yang tidak ada dalam primary key dari tabel primer.
- b) Tidak boleh menghapus record dari tabel primer, jika ada record yang cocok dalam tabel yang terhubung.
- c) Tidak boleh mengubah nilai primary key dalam tabel primer, jika field itu memiliki record - record yang terhubung.

10) Menghapus hubungan

Untuk menghapus hubungan :

1. Pada jendela **Relationships**, klik garis yang ingin dihapus sehingga garis itu menjadi lebih tebal.
2. Tekan **Del**.

11) Hubungan satu ke banyak (One to Many)

Dalam hubungan satu ke banyak, sebuah record dalam tabel A dapat memiliki lebih dari satu record yang cocok dalam tabel B, namun sebuah record dalam tabel B hanya punya satu record yang cocok dalam tabel A.

12) Membuat hubungan satu ke banyak

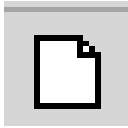
Membuat hubungan satu ke banyak prosedurnya sama dengan membuat hubungan satu ke satu. Hanya saja, primary key dari tabel pertama dihubungkan ke sebuah field yang tidak unik pada tabel kedua.

13) Hubungan banyak ke banyak

Dalam hubungan banyak ke banyak, sebuah record dalam tabel A dapat memiliki lebih dari satu record yang cocok dalam tabel B. Demikian pula, sebuah record dalam tabel B dapat memiliki lebih dari satu record yang cocok dalam tabel A.

Kedua tabel tidak dapat langsung dihubungkan. Untuk menghubungkan kedua tabel ini, harus dibuat sebuah tabel baru yang akan menjembatani keduanya. Tabel ketiga ini biasa disebut *junction table*. Primary key dari tabel ketiga ini minimal memiliki dua field, yaitu *foreign key* dari tabel A maupun tabel B.

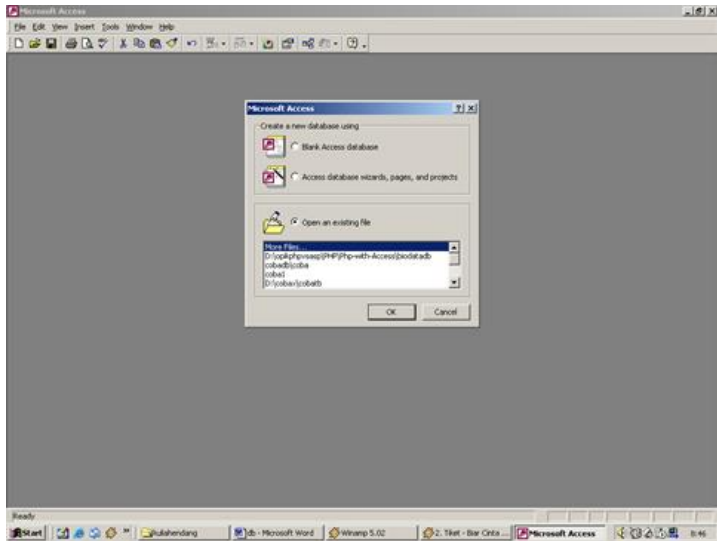
8. Membuat Database Microsoft Access



Untuk membuat database baru, lakukan langkah - langkah berikut ini :

1. Dari menu **File**, pilih **New Database (Ctrl+N)** atau mengklik tombol new database pada **Toolbar**.
2. Sorot icon **Blank Database**, kemudian tekan **enter** atau klik **OK**.
3. Pada kotak dialog **File New Database**, pada kotak **Save in** pilih folder (direktori) tempat menyimpan file database. Pada kotak **File name** ketikkan nama file database.
4. Klik **Create**. Akan muncul tampilan database yang masih kosong.

Contoh pembuatan Database seperti dibawah ini



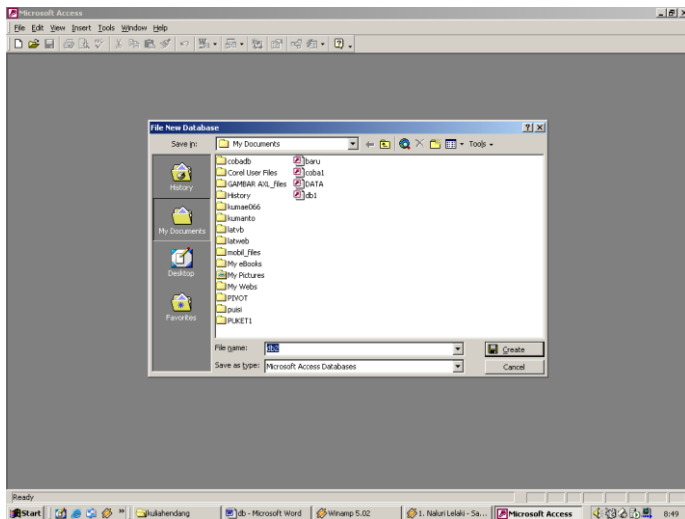
Perintah tersebut diatas dengan maksud :

- a. Apabila perintah Blank Access database untuk membuat sebuah database baru yang masih kosong.
- b. Access data wizards, pages and projects, untuk membuat sebuah database baru dengan menggunakan panduan access
- c. Open an exiting file, untuk membuka file database yang ada

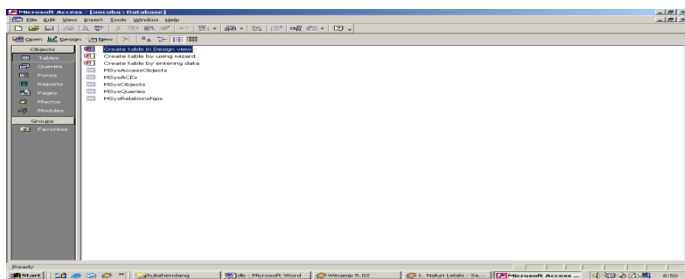
Contoh Pembuatan database dengan acces secara sederhana, Dengan menggunakan 3 macam table yaitu :

- a. Table Master file Biodata mahasiswa
- b. Table Master File Matakuliah
- c. Table Transaksi, untuk mendapatkan nilai

- a. Pilih perintah Blank Access database untuk memulainya
Hasilnya terlihat pada gambar dibawah ini.

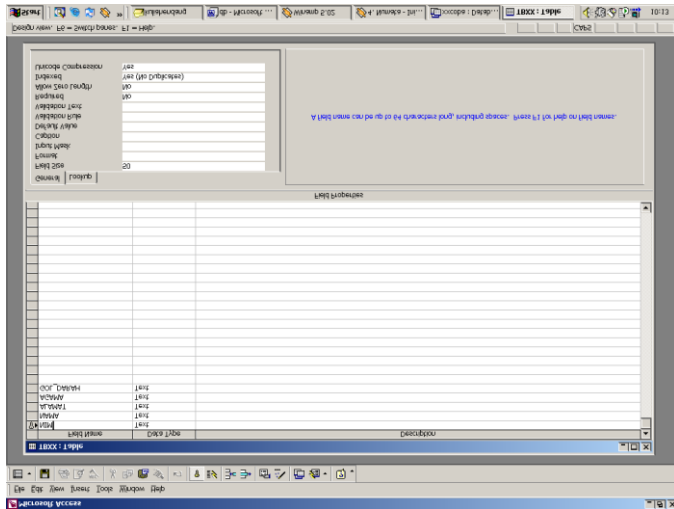


- b. File New Database pilih create ini untuk nama tablenya
c. Create table pilih design View seperti gambar dibawah



- d. Isi untuk field name dan data typenya
Untuk Field table biodata mahasiswa pada gambar dibawah, dengan catatan bahwa untuk table master Field NIM sebagai kunci Utama (Primery key) dihubungkan ke table transaksi Field NIM sebagai kunci tamu (Foreign key).

FIELD	TYPE
NIM	TEXT
NAMA	TEXT
ALAMAT	TEXT
AGAMA	TEXT
GOLONGAN_DARA	TEXT

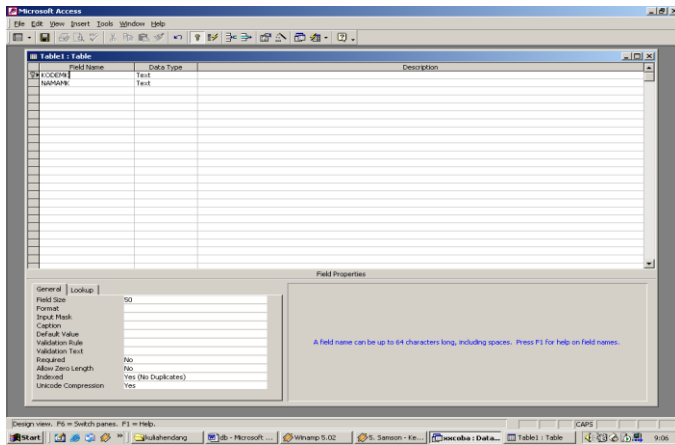


e. Isilah dari table master untuk biodata mahasiswa

NIM	NAMA	ALAMAT	AGAMA	GOL_DARAH
001	Axl Adilla	Purwokerto	Islam	A
002	Dina Astri Megawati	Semarang	Islam	A
003	Satrio	Tegal	Islam	B
004	Ikfina Eka Putri Agustin	Banjarnegara	Islam	O
005	Devia Ramayani	Surabaya	Kristen	B

f. Untuk Field table Master Matakuliah pada gambar dibawah, dengan catatan bahwa untuk table master Field KODEMK sebagai kunci Utama (Primery key) dihubungkan ke table transaksi Field KODEMK sebagai kunci tamu (Foreign key).

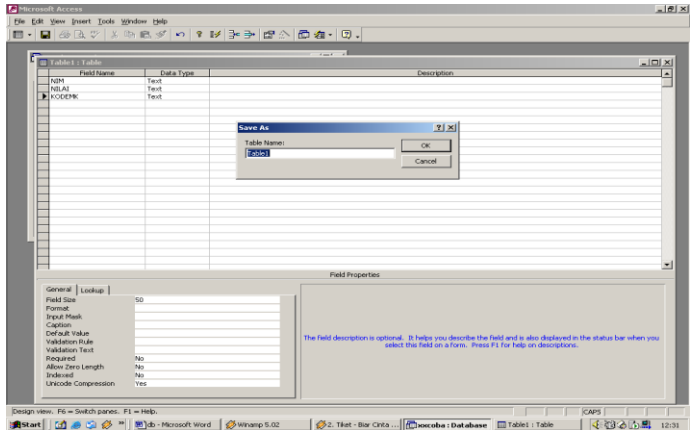
FIELD	TYPE
KODEMK	TEXT
NAMAMK	TEXT



g. Isilah dari table master untuk Matakuliah

KODEMK	NAMA_MATAKULIAH
MK01	DATABASE
MK02	SISTEM INFORMASI MANAJEMEN
MK03	ARTIFISIAL INTELGENSI
MK04	PAKET PROGRAM NIAGA
MK05	ITERAKSI MANUSIA DAN KOMPUTER

h. Untuk Field table Transaksi Nilai Matakuliah pada gambar dibawah, dengan catatan bahwa untuk table Transaksi Field yang digunakan adalah NIM, NILAI, KODEMK, MATAKULIAH sebagai kunci Utama (Primery key) NIM, KODEMK dihubungkan ke table transaksi Field NIM, KODEMK sebagai kunci tamu (Foreign key).

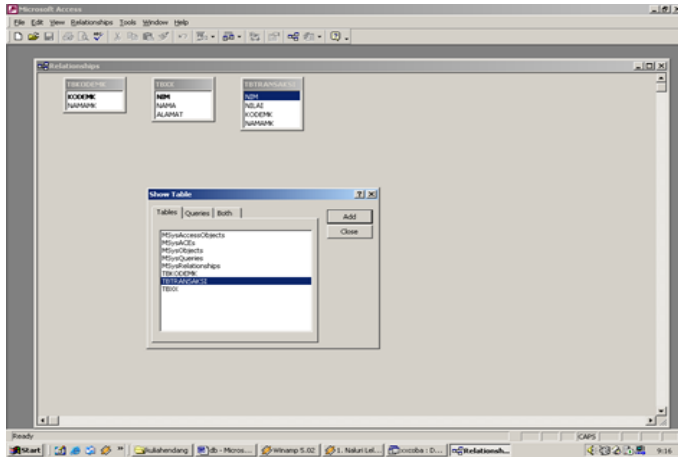


i. Isilah dari table Transaksi untuk Matakuliah

NIM	NILAI	KODEMK
001	A	MK01
001	A	MK02
002	B	MK02
002	C	MK03
003	A	MK03
003	C	MK04
004	B	MK04
004	D	MK05
005	A	MK05
005	D	MK05

Selanjutnya membuat Tool relational database untuk membuat dan mengatur hubungan antar table. Langkah pertama dalam proses ini adalah mendefinisikan relasi antar table. Pada umumnya untuk kunci utama /primary key pada table master akan memberikan nama

yang unik sebagai identitas yang dimiliki dengan table transaksi sebagai kunci tamu/foreign key.



Ada tiga pilihan untuk jenis relasi antar table:

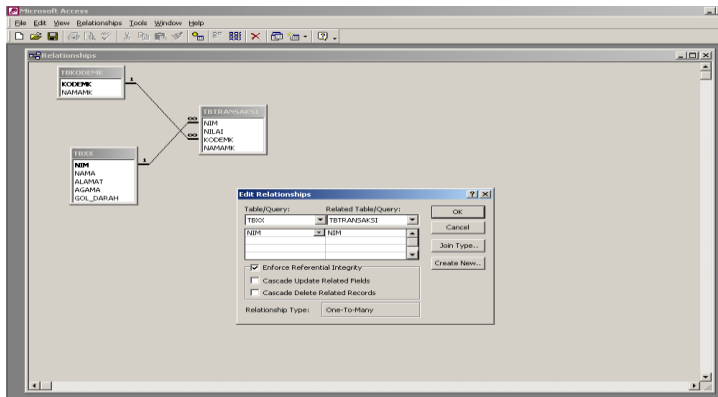
- One to Many relationship (satu ke Banyak) Jenis Relasi yang Paling Umum, sebuah record dalam table KDMatakuliah bisa mempunyai banyak record yang bersesuaian dengan table transaksi tetapi sebuah record dalam table transaksi hanya memiliki sebuah record yang bersesuaian dalam table Kdmatakuliah.
- Many to many relationship (Banyak ke Banyak) sebuah record dalam table KDMatakuliah bisa mempunyai banyak record yang bersesuaian dalam table transaksi dan sebuah record dalam table transaksi dapat memiliki banyak record yang bersesuaian dalam table Kdmatakuliah.

Catatan Kalau ingin menghubungkan banyak ke banyak diperlukan table penghubung sebagai pembantu dengan model didekomposisikan lagi sehingga membentuk table one to many

- One to one relationship (satu ke satu) setiap record dalam table KDMatakuliah hanya dapat mempunyai satu record yang bersesuaian dengan table transaksi dan sebaliknya juga demikian.

Catatan hubungkan one to one tidak umum karena table dengan field yang sama bisa digabungkan melalui queri dan interjoin akan dibahas di bawah lebih lanjut, terutama untuk keamanan data

Pada jendela kerja relationships yg sedang ditampilkan pilih dan klik 2X garis antar table muncul



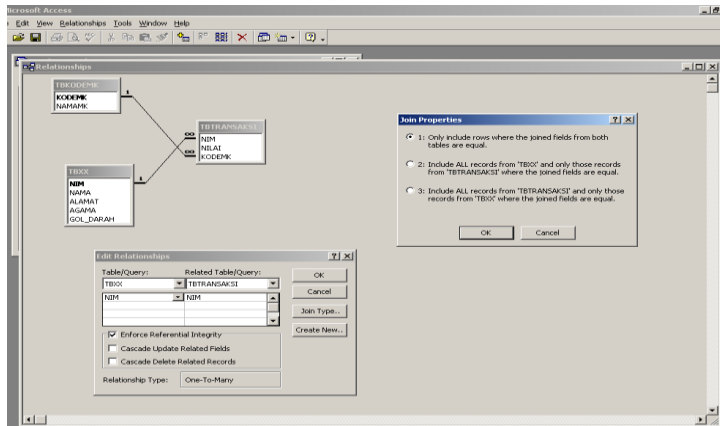
Keterangan apabila kita ingin menggunakan aturan dalam relational:

- ENFORCE REFERENTIAL INTEGRITY (jika anda menghendaki supaya anda memperlakukan aturan RI terhadap relasi yg anda modifikasi)
- CASCADE UPDATE RELATED FIELDS (perubahan pada primary key pd table primer secara otomatis mengubah nilai pada record-record yang bersesuaian dalam table yang memiliki relasi dengan table primer)
- CASCADE DELETE RELATED RECORDS (penghapusan record pada table primary mengakibatkan penghapusan record-record yang bersesuaian pada table yang direlasikan dengan table primer).

Join Type Ada 3 Macam :

- a. ONLY INCLUDE ROWS WHERE THE JOINED FIELDS FROMS BOTH TABLE ARE EQUAL (pilihan default artinya hanya menampilkan record-record yg bersesuaian dari kedua table yg memiliki relasi)
- b. INCLUDE ALL RECORD FROM AA AND ONLY THOSE RECORDS FROM BB WHERE THE JOINED FIELDS ARE EQUAL(menampilkan semua record dalam table yg direlasikan dan hanya record-record yg bersesuaian pada table primer)
- c. INCLUDE ALL RECORD FROM BB AND ONLY THOSE RECORDS FROM AA WHERE THE JOINED FIELDS ARE EQUAL (menampilkan semua record dalam table primer dan hanya record-record yang bersesuaian pada table yang direlasikan).

Keterangan Lebih lanjut dalam pembahasan query



D. Bekerja dengan Query

1. Perintah Query

- a. SELECT QUERY (umum mengambil dari satu table/ lebih dengan kriteria tertentu bias digunakan utk mengelompokan jumlah data/rata-rata/total)

- b. **PARAMETER QUERY** (digunakan sbg kreteria utk mengambil dt suatu nil yg ingin anda sisipkan ke dlm field)
- c. **CROSTAB QUERY**(menampilkan nilai-nil yg telah diolah (jml total, jml nil/rata-ratadari suatu field spt pivottabel wizard).
- d. **ACTION QUERY**(query yg membuat perubahan thd satu atau beberapa record sekaligus)
- e. **SQL QUERY**(query yg dibuat menggunakan pernyataan SQL.

2. Select Query

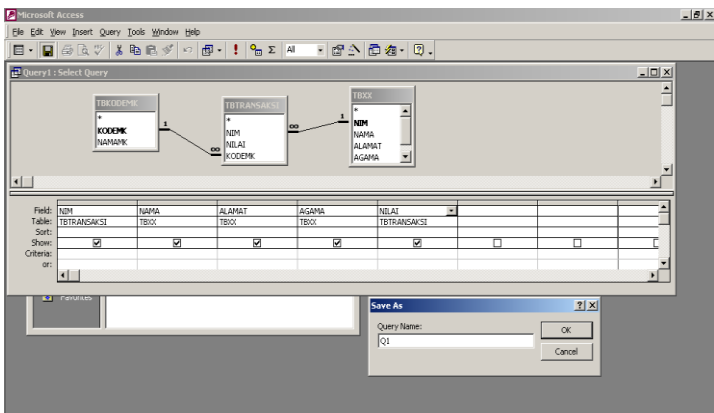
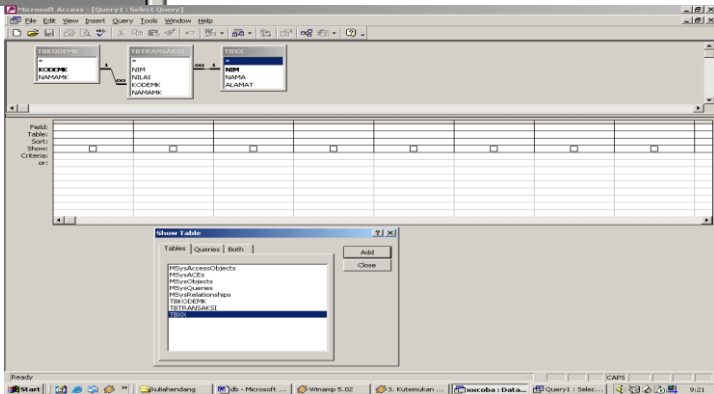
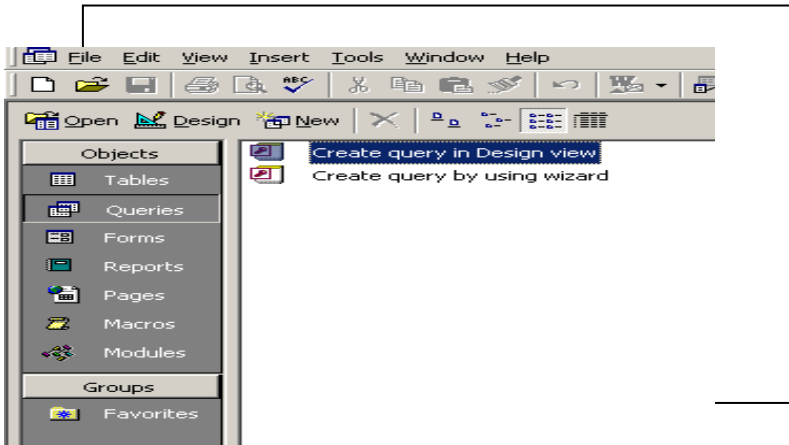
Select query adalah jenis query yang paling umum. Dengan select query bisa diperoleh data dari satu atau beberapa tabel dan menampilkan hasilnya dalam sebuah datasheet dimana record-record dapat diperbaharui dengan beberapa pembatasan. Select query dapat juga digunakan untuk mengelompokkan record-record, menghitung jumlah, rata-rata dan sebagainya.

3. Membuat Select Query

Untuk membuat select query :

1. Dari jendela database, klik tab **Queries**, lalu klik **New**.
2. Pilih **Design View**, lalu klik **OK**.
3. Pada kotak dialog **Show Table**, pilih satu atau beberapa tabel yang akan dibuat query. Pilih tabel dengan menyorot nama tabel, lalu klik **Add**. Lakukan untuk semua tabel yang dipilih.
4. Klik **Close** untuk menutup kotak dialog.
5. Di kotak daftar field, klik ganda nama field yang ingin disertakan ke kisi - kis **desain query**. Pastikan ada tanda centang di kotak periksa di sel **show** pada kolom field yang akan ditampilkan.
6. Pada sel **sort** isikan **Ascending** untuk mengurutkan record secara menaik atau **Descending** bila secara menurun.
7. Tutup jendela **desain query**.
8. Klik **OK**.

Untuk melihat tampilan Query, dari jendela **Queries** sorot nama query yang ingin dilihat lalu klik **Open**.



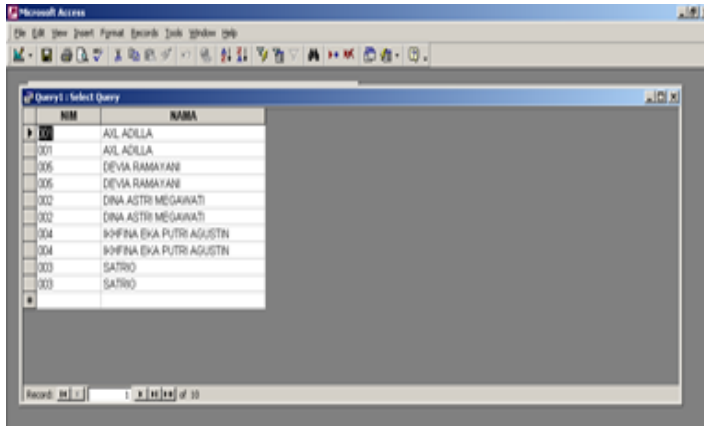
NIM	NAMA	ALAMAT	AGAMA	NILAI
001	AXL ADILLA	PURWOKERTO	ISLAM	A
002	DINA ASTRID MEGAWATI	SEMARANG	ISLAM	B
003	SATRIO	TEGAL	ISLAM	A
004	IKHFINA EKA PUTRI AGUSTIN	BANJARNEGARA	ISLAM	B
005	DEVIA RAMAYANI	SURABAYA	KRISTEN	A
001	AXL ADILLA	PURWOKERTO	ISLAM	A
002	DINA ASTRID MEGAWATI	SEMARANG	ISLAM	C
003	SATRIO	TEGAL	ISLAM	C
004	IKHFINA EKA PUTRI AGUSTIN	BANJARNEGARA	ISLAM	D
005	DEVIA RAMAYANI	SURABAYA	KRISTEN	D

4. Memfilter dengan Query

Filter adalah suatu fasilitas MS-Access yang mirip dengan query, dan dapat dimanfaatkan untuk menyaring data pada tabel. Filter tidak dapat diberi nama, sehingga sebuah tabel hanya dapat dihubungkan dengan satu filter saja. Jika dibuat sebuah filter baru, filter lama otomatis akan terhapus. Agar filter lama tidak hilang, filter tersebut dapat disimpan dalam suatu query.

Field:	NIM	NAMA						
Table:	TRANSAKSI	BIODATA						
Sort:	Ascending							
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteria:								
or:								

Contoh Penggunaan sortir dengan untuk field NIM pada table Transaksi dan field NAMA table master Biodata mahasiswa diurutkan berdasarkan field NAMA, hasilnya seperti pada table dibawah.



Untuk membuat Filter :

1. Buka query dalam tampilan **design**.
2. Pada sel **Criteria** isikan nilai atau ekspresi field yang ingin dijadikan dasar pemfilteran.
3. Pindahlah ke tampilan **Query View** untuk melihat query hasil dengan mengklik tombol **Query View** pada toolbar.
4. Simpan.

5. Menghapus Field dari Query

Untuk menghapus field dari query :

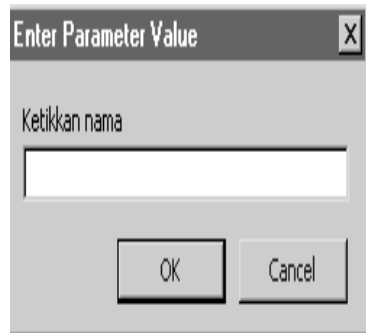
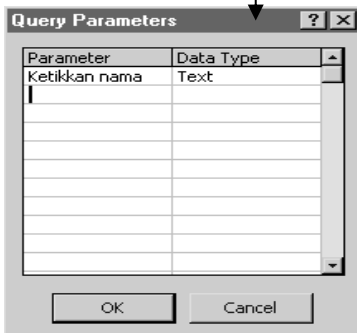
1. Klik field **selector** dari kolom field yang ingin dihapus lalu tekan tombol **Del**.
2. Simpan perubahan pada **query**.
3. Klik tombol **Datasheet** pada toolbar untuk melihat hasilnya.

6. Parameter Query

Parameter query adalah query yang jika dijalankan akan menampilkan sebuah kotak dialog yang meminta pemakai memasukkan suatu informasi, seperti kriteria untuk memperoleh sejumlah record atau suatu nilai yang ingin disimpan dalam sebuah field.

Untuk membuat Parameter Query :

1. Pada tampilan **design query**, isi kisi - kisi query, pada sel **Criteria**, masukkan string dalam kurung siku, misalnya [Ketikkan Nama]. Kurung siku yang mengapit string menandakan bahwa bila seseorang membuka query ini nanti, akan ditampilkan sebuah kotak dialog yang akan meminta masukkan.
2. Tentukan **type data** dari **parameter** dengan memilih parameter dari menu **query**.
3. Pada sel pertama dari kolom **parameter** ketikkan kata yang sama dengan yang telah diketikkan pada sel **Criteria**, namun tanpa tanda kurung.
4. Tentukan jenis data **parameter**. Tekan TAB untuk pergi ke sel **Data Type** lalu pilih tipe data.
5. Tutup tampilan design.



Untuk melihat hasilnya, Open Query atau klik tombol shortcut Query View. Akan ditampilkan kotak dialog yang meminta masukkan dari pemakai.

Untuk menghapus parameter, dari tampilan **design query**, pilih **parameter** dari menu **query**. Sorot isinya lalu tekan Del.

Field:	npm
Table:	mhs
Sort:	Ascending
Show:	<input checked="" type="checkbox"/>
Criteria:	
or:	

- Buatlah **sort** secara **Ascending** untuk npm pada tabel *Mhs* dengan mengklik tombol **design** pada tab Query.
- Pada sel **Field** pilih npm
- Pada sel **Sort** pilih *Ascending*.
- Simpan Query dengan nama *QMhs*.
- Untuk melihat hasilnya klik tombol **Open** dari tab **Query**.

- Buatlah **filter** untuk menampilkan semua mahasiswa yang berdomisili di kota "Jakarta" dengan mengisi kriteria pada sel **Criteria** dengan *Like "Jakarta"*.
- Sebelumnya pilih "*Kota*" pada sel **Field**.
- Simpan query dengan nama *Q2Mhs*.
- Untuk melihat hasilnya klik tombol **Open** dari tab **Query**.

Field:	kota
Table:	mhs
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	Like "Jakarta"
or:	

7. Membuat Filter

NIM	NILAI	KODEMK
001	80	MK01
001	80	MK02
002	70	MK02
002	60	MK03
003	80	MK03
003	60	MK04
004	70	MK04
004	50	MK05
005	80	MK05
005	50	MK05


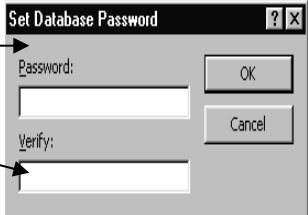

Membuat query dengan update yaitu merubah nilai dalam table transaksi melalui query secara otomatis pada table transaksi menambah 10 % dari field nilai yang ada.

8. Membuat Password

Salah satu bentuk perlindungan yang disediakan MS-Access adalah dengan menerapkan password. Bila password diterapkan pada sebuah database, semua pemakai harus mengetikkan password yang benar sebelum dapat membuka database tersebut.

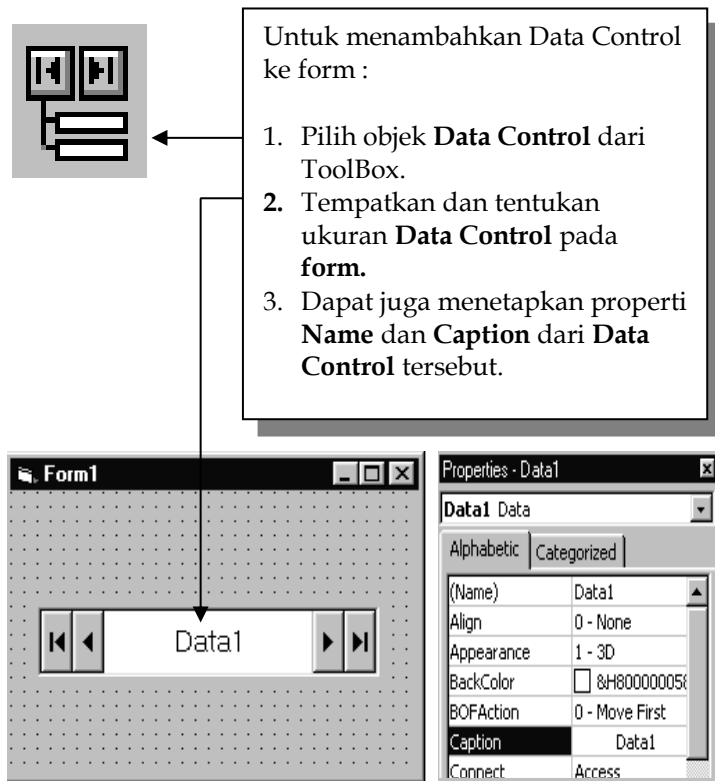
Untuk membuat password :

1. Buka database dengan mengklik **Open** dari menu **File**.
2. Pada kotak dialog **Open**, sorot icon database yang akan dilindungi dengan password.
3. Klik kotak periksa **Exclusive** dalam kotak dialog tersebut.
4. Klik **Open**.
5. Setelah database terbuka, klik **Security** dari menu **Tools**, lalu pilih **Set Database Password**.
6. Pada kotak dialog **Set Database password**, ketikkan password di kotak **password**.
7. Klik pointer mouse di kotak **Verify**, lalu ketikkan password sekali lagi, untuk memastikan tidak terjadi salah ketik.
8. Klik **OK**.
9. Selanjutnya bila database akan dibuka akan muncul kotak dialod **Password**. Ketikkan password kemudian klik **OK**.



9. Mengakses database menggunakan Data Control

Data Control dirancang untuk menyediakan cara yang mudah untuk mengakses database, diantaranya Data Control menyediakan fungsi-fungsi pemindahan record untuk aplikasi, dengan adanya tombol-tombol yang ekuivalen dengan metode MoveFirst, MovePrevious, MoveNext dan MoveLast.



Untuk menambahkan Data Control ke form :

1. Pilih objek **Data Control** dari ToolBox.
2. Tempatkan dan tentukan ukuran **Data Control** pada **form**.
3. Dapat juga menetapkan properti **Name** dan **Caption** dari **Data Control** tersebut.

The diagram illustrates the steps for adding a Data Control to a form. It shows a toolbox icon with a Data Control symbol, a list of instructions, a form with a Data1 control, and a Properties window for the Data1 control.

Properties - Data1	
Data1 Data	
Alphabetic Categorized	
(Name)	Data1
Align	0 - None
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H80000005
BOFAction	0 - Move First
Caption	Data1
Connect	Access

Keuntungan menggunakan Data Control adalah diperlukan lebih sedikit pemrograman untuk mengembangkan aplikasi akses data. Namun pada Data Control tidak terdapat fungsi penghapusan record.

10. Properti Data Control

Beberapa properti Data control yang digunakan untuk mengakses database yaitu :

a. Properti DatabaseName

Properti DatabaseName menentukan nama file database yang akan diakses termasuk nama lengkap path-nya.



b. Properti RecordSource

Properti RecordSource menentukan sumber recordset, dalam hal ini adalah nama tabel yang akan digunakan.



c. Properti Connect

Properti ini hanya diperlukan bila digunakan database selain Access, misalnya FoxPro, Dbase III, dan lain - lain, karena defaultnya adalah Access.



d. Properti Tindakan BOF dan EOF

Properti tindakan memberitahu Data Control apa yang harus dilakukan bila mencapai awal file (BOF) atau akhir file (EOF). Untuk properti BOF, terdapat dua pilihan yaitu :

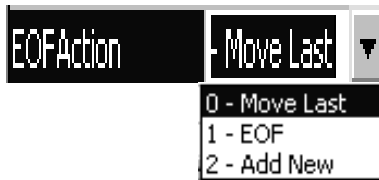
- Mengeksekusi metode MoveFirst untuk menetapkan penunjuk record pada record pertama dan flag BOF pada false.

- Menetapkan flag BOF ke True.














Sedangkan untuk properti EOF terdapat tiga pilihan yaitu :

- Mengeksekusi metode MoveLast untuk menetapkan penunjuk record pada record terakhir dan flag EOF pada false.
- Menetapkan flag EOF ke True.
- Mengeksekusi metode AddNew untuk menetapkan penambahan record baru.



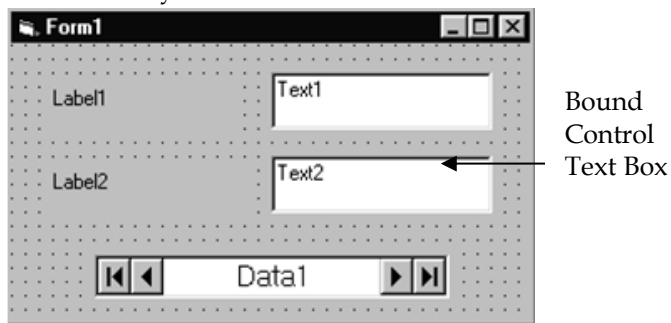
e. Menggunakan Bound Control

Bound Control adalah objek kontrol Visual Basic yang dapat di-link ke field – field pada satu atau dua Data Control. Visual Basic menyediakan beberapa Bound Control yaitu :

Picture box		Combo Box	
Label		DB-List Box	
Text Box		DB-Combo Box	
Check Box		DB-Grid	
Image Box		Mask Edit	
List Box			

Bound Control menampilkan data dari field dan sumber data yang ditentukan pada properti - properti kontrol. Sumber data untuk suatu Bound Control selalu suatu Data Control. Ketika penunjuk record berpindah dari satu record ke record lain dengan menggunakan Data Control, Bound Control diperbarui sehingga menampilkan record yang sedang ditunjuk oleh penunjuk record.

Untuk menempatkan salah satu Bound Control pada form, sama caranya dengan menempatkan control-control lainnya.



f. Properti Bound Control

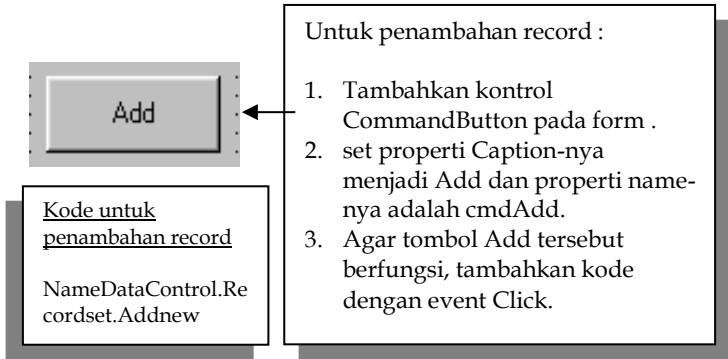
- Agar Bound Control dapat bekerja, harus dikaitkan dengan Data Control yang menyajikan recordset (record - record yang terdapat Terdapat properti DataField, untuk menentukan field yang akan di-link.pada tabel yang memiliki link dengan Data Control), melalui properti DataSource.
- Selain properti DataSource, terdapat properti DataField untuk menentukan field yang akan di-link.

g. Menggunakan Tombol Perintah untuk melengkapi Data Control

Seperti telah dijelaskan sebelumnya, walaupun Data Control fleksibel, namun Data Control tidak mempunyai beberapa fungsi, seperti fungsi penghapusan record. Untuk menutupi kekurangan ini,

dapat ditambahkan fungsi - fungsi pada layar entri data dengan menggunakan perintah - perintah program yang dinyatakan pada suatu tombol perintah.

h. Penambahan record



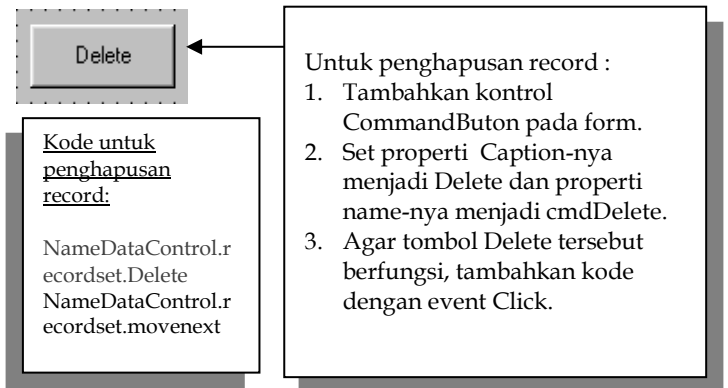
Untuk penambahan record :

1. Tambahkan kontrol CommandButton pada form .
2. set properti Caption-nya menjadi Add dan properti name-nya adalah cmdAdd.
3. Agar tombol Add tersebut berfungsi, tambahkan kode dengan event Click.

Kode untuk penambahan record

```
NameDataControl.Recordset.Addnew
```

i. Penghapusan record



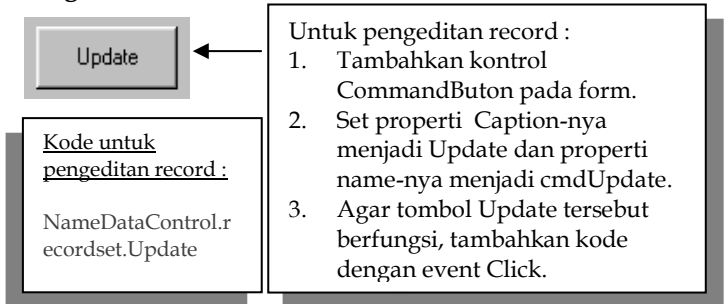
Untuk penghapusan record :

1. Tambahkan kontrol CommandButon pada form.
2. Set properti Caption-nya menjadi Delete dan properti name-nya menjadi cmdDelete.
3. Agar tombol Delete tersebut berfungsi, tambahkan kode dengan event Click.

Kode untuk penghapusan record:

```
NameDataControl.recordset.Delete  
NameDataControl.recordset.movenext
```

j. Pengeditan record











Untuk pengeditan record :







1. Tambahkan kontrol CommandButon pada form.
2. Set properti Caption-nya menjadi Update dan properti name-nya menjadi cmdUpdate.
3. Agar tombol Update tersebut berfungsi, tambahkan kode dengan event Click.






Kode untuk pengeditan record :

```
NameDataControl.recordset.Update
```

<p>Pointer</p> 	<p>Pointer bukan unit kontrol, hanya berfungsi untuk memindahkan atau mengubah ukuran objek kontrol dan form.</p>
<p>PictureBox</p> 	<p>Awalan : Pic Merupakan kontrol untuk menampilkan file bitmaps, metafiles, icon, GIF dan JPEG.</p>
<p>Label</p> 	<p>Awalan : Lbl Kontrol yang menampilkan text namun pemakai tidak dapat berinteraksi dengannya ataupun mengubahnya.</p>
<p>TextBox</p> 	<p>Awalan : Txt Kontrol yang menampilkan text dimana pemakai dapat mengisinya ataupun melihat text yang ditampilkan sebagai</p>

	output suatu proses.
<p>Frame</p> 	<p>Awalan : Fra</p> <p>Merupakan kontrol untuk mengidentifikasi sebuah grup pengontrolan.</p>
<p>CommandButton</p> 	<p>Awalan : Cmd</p> <p>Kontrol ini akan membuat sebuah tombol yang biasanya digunakan untuk mengeksekusi sebuah tindakan</p>
<p>CheckBox</p> 	<p>Awalan : Chk</p> <p>Kontrol ini akan memberikan perintah pilihan Benar/Salah atau Ya/Tidak. Memungkinkan untuk memilih beberapa pilihan sekaligus dalam suatu kelompok pada saat bersamaan.</p>
<p>OptionButton</p> 	<p>Awalan : Opt</p> <p>Kontrol ini merupakan bagian dari option button lain yang akan membentuk grup pilihan. Hanya memungkinkan pemakai untuk memilih salah satu dari beberapa pilihan yang ada pada satu grup.</p>

<p>ComboBox</p> 	<p>Awalan : Cbo</p> <p>Merupakan kontrol dengan kombinasi antara TextBox dan ListBox. Dengan kontrol ini pemakai dapat mengetikkan pilihan atau memilih item lewat <i>drop-down list</i>.</p>
<p>ListBox</p> 	<p>Awalan : Lst</p> <p>Kontrol ini akan menampilkan daftar item dimana pemakai dapat memilih salah satu dari item - item yang ditampilkan.</p>
<p>Horizontal ScrollBar</p> 	<p>Awalan : Hsb</p> <p>Kontrol ini memungkinkan pemakai untuk memilih suatu objek selama masih dalam jangkauan horizontal objek. Selain itu kontrol juga bisa digunakan sebagai input untuk memasukkan nilai suatu data.</p>
<p>Vertikal ScrollBar</p> 	<p>Awalan : Vsb</p> <p>Sama dengan Horizontal ScrollBar hanya saja untuk jangkauan vertikal.</p>
<p>Timer</p> 	<p>Awalan : Tmr</p> <p>Merupakan kontrol untuk mengeksekusi waktu kejadian pada rutin program termasuk di dalamnya adalah selang (interval) waktu.</p>
<p>DriveListBox</p> 	<p>Awalan : Drv</p> <p>Kontrol ini akan menampilkan daftar drive pada komputer pemakai dan memungkinkan pemakai untuk memilih sebuah drive.</p>

<p>DirListBox</p> 	<p>Awalan : Dir</p> <p>Kontrol ini akan menampilkan daftar directori pada drive terpilih. Kontrol ini memungkinkan pemakai untuk memilih sebuah directori dan path.</p>
<p>FileListBox</p> 	<p>Awalan : Fil</p> <p>Kontrol ini akan menampilkan daftar file pada direktori terpilih dan memungkinkan pemakai untuk memilih sebuah file.</p>
<p>Shape</p> 	<p>Awalan : Shp</p> <p>Kontrol ini memungkinkan pemrogram untuk menambahkan lingkaran, elips dan persegi empat pada form.</p>
<p>Line</p> 	<p>Awalan : Lin</p> <p>Kontrol ini memungkinkan pemrogram untuk membuat garis lurus pada form.</p>
<p>Image</p> 	<p>Awalan : Img</p> <p>Kontrol ini akan menampilkan gambar bitmap, metafile, icon, GIF, JPEG. Perbedaannya dengan PictureBox adalah kontrol ini memiliki kecepatan akses lebih cepat tetapi sedikit properti dan eventnya.</p>

E. Menggunakan Variabel, Tipe Data dan Array

1. Variabel

Variabel merupakan tempat untuk menyimpan nilai sementara dari suatu perhitungan. Untuk mendeklarasikan sebuah variabel digunakan pernyataan :

Dim Variabel [As Type]

Melalui **As** dapat mendefinisikan sendiri tipe data atau tipe objek dari variabel yang dideklarasikan. Tipe data ini misalnya integer, string atau variant.

Variabel untuk sebuah prosedur hanya boleh digunakan pada prosedur di tempat mereka dideklarasikan, dan biasanya digunakan dalam perhitungan yang menggunakan temporary.

Dim Contoh1 As Integer
Dim Contoh2 As String

2. Konstanta

Konstanta merupakan variabel yang nilai di dalamnya selalu tetap. Konstanta diperlukan jika dibutuhkan sebuah nilai tetap yang harus muncul di banyak bagian dari rutin. Kemungkinan lain penggunaan konstanta adalah untuk mengingat angka yang sulit.

Const Phi = 3.14159265358979
Const nama = "Gatotkaca"

3. Tipe Data

Tipe Data	Penyimpanan	Jangkauan
Integer	2 byte	-32.768 sampai 32.767
Long	4 byte	-2.147.483.648 sampai 2.147.483.647
Single	4 byte	-3,403823 E38 sampai -1,401298 E-45 untuk nilai negatif. 1,401298 E-45 sampai 3,402823 E38 untuk nilai positif

Double	8 byte	-1,79769313486232 E308 sampai -4,94065645841247 E-324 untuk nilai negatif 4,94065645841247 E-324 sampai 1,79769313486232 E308 untuk nilai positif
Decimal	8 byte	+/- 79.228.162.514.264.337.593.54 3.950.335 untuk bilangan tanpa angka desimal. +/- 7, 92281625142643375935439503 35 untuk bilangan dengan desimal. 0,000000000000000000000000 0001 bilangan terkecil yang mungkin
Currency	8 byte	-922.337.203.685.477,5808 sampai 922.337.203.685.477, 5807
String	1 byte/Char	0 sampai 2 E32 karakter
Byte	1 byte	0 - 255
Boolean	2 byte	True atau False
Date	8 byte	1 Januari 100 sampai 31 December 9999
Object	4 byte	Referensi Objek
Variant	16 byte + 1 byte untuk tiap karakter	Null, Error Nilai numeric sampai jangkauan tipe data Double, karakter teks, objek atau array.

Tipe data **Variant** yang akan menyajikan semua jenis data yang didefinisikan pada visual basic. Jika sebuah variabel dideklarasikan tanpa perintah **As**, maka standar dari tipe data yang digunakan adalah variant.

Dim Jumlah, Contoh1, Contoh2

Tipe data Numeric menyimpan data berupa angka, tipe data string menyimpan rangkaian karakter, tipe data boolean menyimpan data berupa benar/salah, tipe data Date menyimpan data berupa tanggal dan waktu, dan tipe data objek menyimpan data berupa objek. Pendeklarsiannya harus menggunakan perintah **Private**, **Public**, **Dim** atau **Static**.

Private contoh1 **As Long**
Public contoh2 **As String**
Dim contoh3 **As Currency**, contoh5 **As Integer**
Static contoh6 **As Double**, contoh7 **As Single**, Contoh8 **As Boolean**

4. Array

Array merupakan salah satu fasilitas agar dapat menyimpan data secara berrurutan dalam sebuah nama variabel. Dalam array data tersimpan dengan menggunakan indeks untuk memudahkan pencarian kembali data tersebut.

Array mempunyai batas atas dan batas bawah, dimana data akan tersimpan di antara kedua batas tersebut. Semua elemen data yang tersimpan dalam sebuah variabel array mempunyai tipe data yang sama. Semua tipe data dapat dideklarasikan bagi variabel array.

Dim Contoh1 (19) **As Integer**
Public Contoh2 (80) **As Long**

5. Struktur Kontrol

Struktur kontrol merupakan pengatur aliran program, berbentuk rangkaian perintah yang harus ditulis untuk memenuhi beberapa keadaan, yaitu :

- a. Mengulang sebagian rutin karena tidak terpenuhinya suatu kondisi.
- b. Melanjutkan sebuah pernyataan bila kondisi terpenuhi.
- c. Memilih sebuah pilihan dari beberapa alternatif bila sebuah kondisi terpenuhi.

6. Struktur Pengambilan Keputusan

If ...Then

Struktur ini digunakan untuk mengeksekusi satu atau lebih perintah yang menyatakan keadaan.

```
If kondisi Then perintah  
(untuk perintah satu baris)  
  
    If kondisi Then  
        Perintah 1  
        Perintah 2  
        ...  
    End If  
(untuk perintah dengan banyak baris)
```

If ... Then ... Else

Struktur ini mirip dengan struktur **If ... Then**, hanya saja digunakan untuk banyak blok perintah.

```
If kondisi 1 Then  
[perintah blok 1]  
ElseIf kondisi 2 Then  
[perintah blok 2]  
Else  
[perintah blok -n]  
End if
```

Select Case

Struktur ini dapat digunakan sebagai alternatif pengganti dari struktur kontrol **If ... Then ... Else**. **Select Case** mempunyai penulisan yang lebih mudah sehingga penulisan rutin dapat menjadi lebih efisien. Struktur ini secara selektif akan langsung mengeksekusi sebuah blok perintah dari beberapa pilihan yang diberikan.

```
Select Case kondisi
  Case ekspresi 1
    [perintah blok 1]
  Case ekspresi 2
    [perintah blok 2]
  Case else
    [perintah blok -n]
End Select
```

7. Struktur Pengulangan

Struktur pengulangan digunakan untuk mengulang sebagian dari rutin, sehingga tidak perlu lagi menulis ulang rutin sebanyak pengulangan yang diinginkan.

Do ... Loop

Struktur ini digunakan untuk mengulang sebuah blok perintah sampai jumlah tertentu.

<pre>Do While kondisi Perintah Loop</pre>	<pre>Do Until kondisi Perintah Loop</pre>
<pre>Do Perintah Loop While kondisi</pre>	<pre>Do Perintah Loop Until kondisi</pre>

For ... Next

Struktur kontrol **Do ... Loop** paling baik digunakan jika tidak diketahui dengan pasti berapa kali akan diadakan pengulangan blok

perintah. Jika telah diketahui secara pasti berapa kali akan diadakan pengulangan, agar penulisan rutin lebih efisien dapat digunakan struktur kontrol **For ... Next**.

```
For variabel = awal To akhir [Step pertambahan]
    Perintah
Next [variabel]
```

For Each ... Next

Struktur kontrol **For Each ... Next** digunakan untuk mengulang sekelompok perintah bagi tiap elemen pada suatu koleksi objek, misalnya kumpulan kontrol pada sebuah form.

Penggunaan struktur kontrol **For Each ... Next** sangat berguna jika tidak diketahui secara pasti jumlah elemen pada koleksi objek

```
For Each elemen In grup
    Perintah
Next elemen
```

8. Kotak Pesan, Input dan Dialog

a. Kotak Pesan

Sebuah kotak pesan dapat digunakan sebagai input atau pun output dari aplikasi Visual Basic. Meskipun demikian, kotak pesan lebih banyak digunakan sebagai output unuk menampilkan pesan keadaan yang sedang terjadi pada aplikasi.

Untuk menciptakan sebuah kotak pesan, Visual Basic telaha menyediakan prosedur internal, yaitu **MsgBox**.

```
MsgBox (Prompt [, parameter] [,judul] [, filehelp, konteks])
```

b. Prompt

Teks yang dituliskan disini akan ditampilkan pada kotak pesan. Panjang pesan maksimum yang dapat ditampilkan adalah 1024 karakter.

c. Parameter

Parameter menentukan jenis tombol yang akan ditampilkan oleh kotak pesan.

Jenis parameter yang dapat digunakan yaitu :

Parameter	Nilai	Keterangan
VbOKOnly	0	Tombol OK saja
VbOKCancel	1	Tombol OK dan Cancel
VbAbortRetry Ignore	2	Tombol Abort, Retry dan Ignore
VbYesNoCancel	3	Tombol Yes, No dan Cancel
VbYesNo	4	Tombol Yes dan No
VbRetryCancel	5	Tombol Retry dan Cancel
VbDefaultButton1	0	Tombol pertama sebagai default
VbDefaultButton2	256	Tombol kedua sebagai default
VbDefaultButton3	512	Tombol ketiga sebagai default

Selain tombol, pada bagian parameter dapat pula ditentukan jenis icon yang ingin ditampilkan pada kotak pesan. Fungsi icon ini untuk membedakan jenis kotak pesan yang ditampilkan.

Parameter	Nilai	Keterangan
VbCritical	16	Menampilkan icon pesan kritis
VbQuestion	32	Menampilkan icon pesan pertanyaan
VbExclamation	48	Menampilkan icon pesan peringatan

VbInformation	64	Menampilkan icon pesan informasi
---------------	----	----------------------------------

9. Pengembalian Nilai

Sebuah kotak pesan yang direspon akan mengembalikan sebuah nilai yang berasosiasi dengan tombol yang ditekan. Dengan memahai nilai yang dikembalikan sebuah kotak pesan, dapat diatur kejadian yang akan dieksekusi selanjutnya saat sebuah tombol ditekan.

Konstanta	Nilai	Tombol yang dipilih
VbOK	1	OK
VbCancel	2	Cancel
VbAbort	3	Abort
VbRetry	4	Retry
VbIgnore	5	Ignore
VbYes	6	Yes
VbNo	7	No

a. Kotak Input

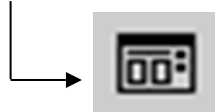
Visual Basic memberikan metode pemasukkan input menggunakan **Input Box** yang akan menampilkan sebuah kotak input.

InputBox (prompt[,judul][, default][, xpos][,ypos][,fileHelp, konteks])

Prompt merupakan teks yang akan ditulis untuk menerangkan data yang akan dimasukkan pada kotak input. Judul merupakan teks yang akan ditulis sebagai judul kotak input pada title bar. Default adalah nilai standar yang akan selalu ditulis pada kotak pengisian.

b. Kotak Dialog

Kotak dialog biasanya ditampilkan untuk peristiwa membuka file, menyimpan file, mengubah setting huruf, mengatur warna dan saat mencetak file ke printer. Visual Basic menyediakan kontrol **Common Dialog** untuk membuat kotak dialog.



```
Nama_kontrol_CommonDialog.metode  
ComDi1.ShowOpen  
  
Atau  
  
Nama_kontrol_CommonDialog.action = nomor  
ComDi1.action = 1
```

Adapun kotak dialog yang dapat dibuat adalah :

Jenis Kotak Dialog	Metode	Nomor
Kotak dialog Open	ShowOpen	1
Kotak dialog Save As	ShowSave	2
Kotak dialog Color	ShowColor	3
Kotak dialog Font	ShowFont	4
Kotak dialog Print	ShowPrinter	5
Memanggil Window Help	ShowHelp	6

10. Menu

Menu merupakan salah satu fasilitas menarik yang dapat ditambahkan pada program aplikasi. Sebuah menu dapat berisi item menu, judul submenu, item submenu dan garis pemisah. Setiap isi menu tersebut berasosiasi dengan kontrol menu yang dibuat melalui menu editor.

a. Menu Editor

Dengan menu editor, dapat ditambahkan isi sebuah menu yang sudah ada, membuat menu baru, atau juga menghapus sebagian isi menu yang sudah jadi.

Untuk menampilkan menu editor, pilih menu Tools → Menu Editor, atau klik icon Menu Editor



Pada Toolbar.

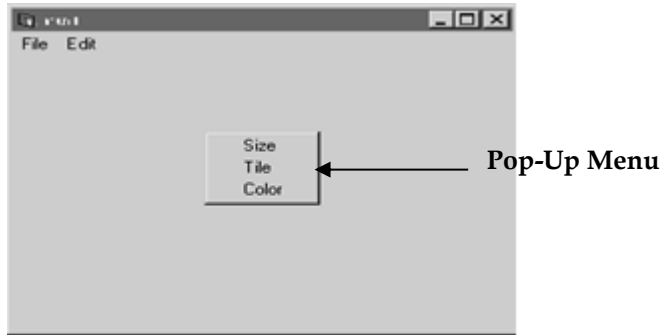


Membuat Kontrol Menu

Cara membuat kontrol menu dengan menu editor :

- 1) Pilih Form yang akan diberi menu, lalu tampilkan menu editor.
- 2) Pada kotak Caption ketikkan teks yang ingin ditampilkan pada menu.
- 3) Pada kotak Name, ketikkan nama yang akan digunakan untuk pemanggilan kontrol di jendela kode.
- 4) Tombol panah kiri atau panah kanan dapat digunakan untuk mengubah posisi kontrol menu apakah sebagai judul menu, item menu, judul submenu, item submenu dan sebagainya.
- 5) Mengatur properti untuk kontrol menu.
- 6) Klik Next untuk membuat menu kontrol lainnya. Klik Insert untuk menyisipkan kontrol menu yang baru di antara kontrol menu yang sudah ada. Tombol panah

atas atau panah bawah dapat digunakan untuk berpindah di antara control-control menu yang terdapat pada kotak daftar.



- 7) Klik OK jika telah selesai bekerja dengan Menu Editor.

b. Properti pada Menu Editor

Beberapa Properti pada Menu Editor yaitu :

- 1) **Checked** : Bila properti ini diberi tanda check maka pada menu yang telah dibuat setelah dijalankan akan terlihat tanda check yang artinya menu tersebut sedang aktif.
- 2) **Enabled** : Bila properti ini di nonaktifkan maka menu yang dibuat setelah dijalankan tidak dapat diklik atau nonaktif.
- 3) **Visible** : Bila properti ini dinonaktifkan maka menu yang dibuat setelah dijalankan tidak akan tampak pada layout.
- 4) **Windowlist** : Properti ini digunakan pada aplikasi yang menggunakan MDI form. Kegunaan windowlist ini adalah untuk menampilkan childform - childform yang telah dibuka.

c. Membuat Pop-Up Menu

Pop-Up menu adalah menu yang ditampilkan di atas form dan terpisah dari menu bar. Lokasi pemunculan menu bergantung dari tempat menekan tombol mouse.

Untuk membuat Pop-Up menu, Visual Basic telah menyediakan metode yaitu

PopupMenu.

```
[Objek.]PopupMenu namamenu [, flags [,x[,y[, bold]]]]
```

F. Multiple Document Interface (MDI)

MDI adalah salah satu fasilitas Visual Basic, yang memungkinkan pemrogram untuk membuat sejumlah form di dalam sebuah form induk. Program yang dibuat dengan Visual Basic hanya dapat memakai satu buah form MDI. Form MDI tersebut dapat menampung form-form lain yang tidak terbatas jumlahnya. Form yang ada di dalam sebuah form MDI biasa disebut dengan *childform*.

Menggunakan Form MDI

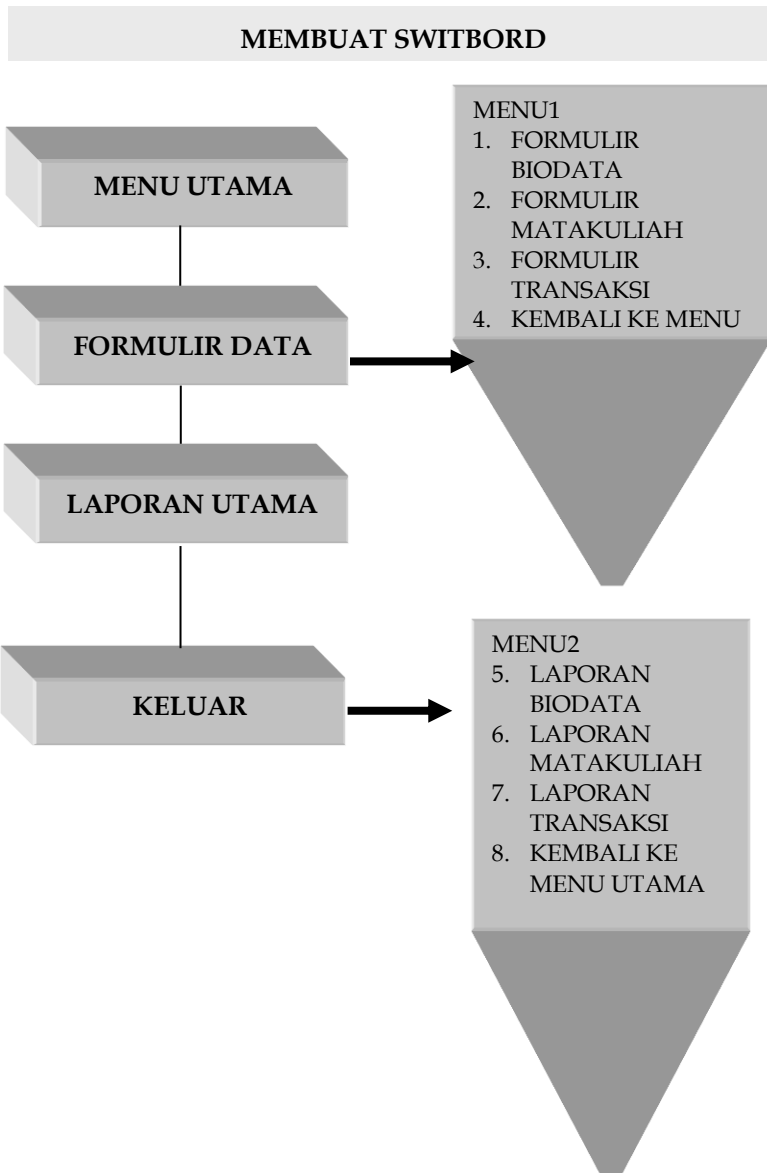
1. Untuk membuat Form MDI pilih menu Project, kemudian pilih Add MDI Form.
2. Untuk membuat *child form*, pilih menu Project, kemudian pilih Add Form. Atau membuka form yang sudah ada. Lalu atur properti MDIChild dari form tersebut bernilai **True**.

G. Database Microsoft Access

Visual Basic telah menyediakan suatu fasilitas untuk pengelolaan database. Program yang dibuat menggunakan Visual Basic dapat membaca format file database yang umum digunakan saat ini. Format tersebut adalah :

1. Dbase III dan Dbase IV
2. Paradox
3. Foxpro
4. Btrieve
5. Microsoft Access, merupakan format standar yang digunakan Visual Basic.

H. SWITBORD



I. MENGGUNAKAN MENU NAVIGASI BERUPA SWITH-BORD DAN FORM

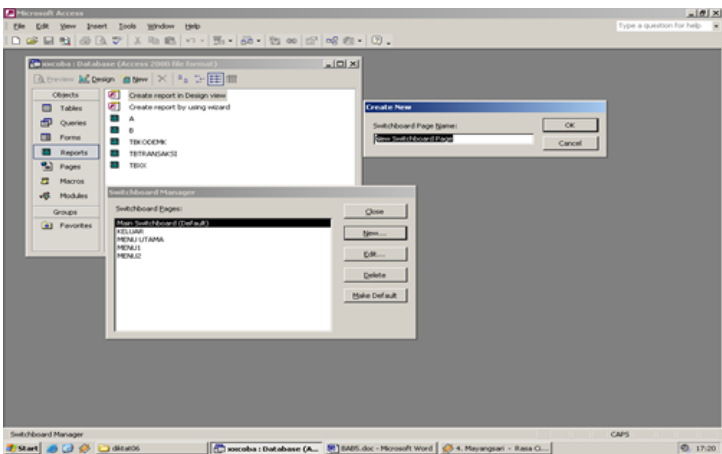
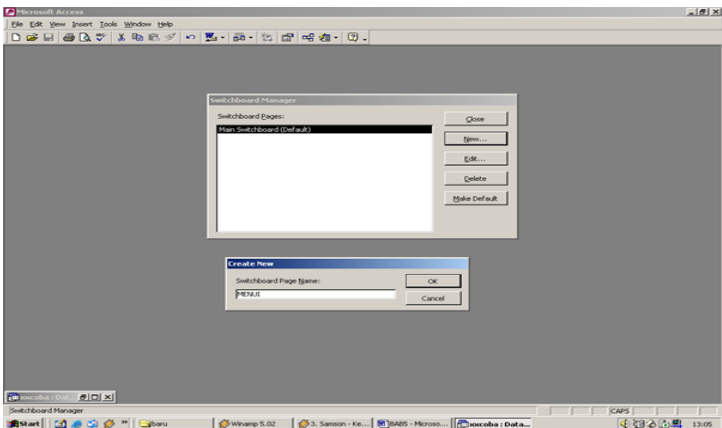
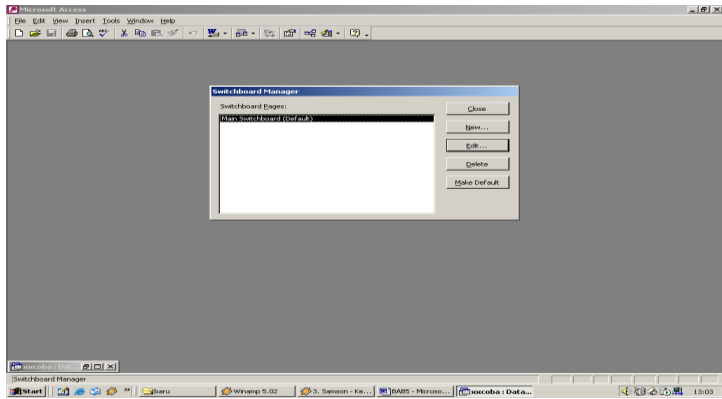
Switchboard digunakan untuk memudahkan suatu obyek pada file database yang telah dirancang seperti pada kasus database menggunakan Knoda. Switchboard relatif mudah dipahami dan sangat sederhana pembuatanya, hanya menggunakan tiga tabel, untuk menyelesaikan pada kasus database manajemen sistem Bapendik. Hubungan antara tabel transaksi dengan tabel master file akan terbentuk relationships sebagai penghubung, untuk mendapatkan informasi dan pendukung yang dibantu dengan query. Selanjutnya didalam pembuatan formulir juga menggunakan ketiga tabel tersebut, yaitu tabel biodata mahasiswa sebagai master dan tabel matakuliah sebagai master, serta tabel transaksi dan untuk Report / laporan menggunakan ketiga tabel biodata mahasiswa sebagai master dan tabel matakuliah sebagai master, serta tabel transaksi.

Langkah-langkah membuat menu navigasi berupa switbord sebagai berikut

Membuat menu navigasi berupa swictboard

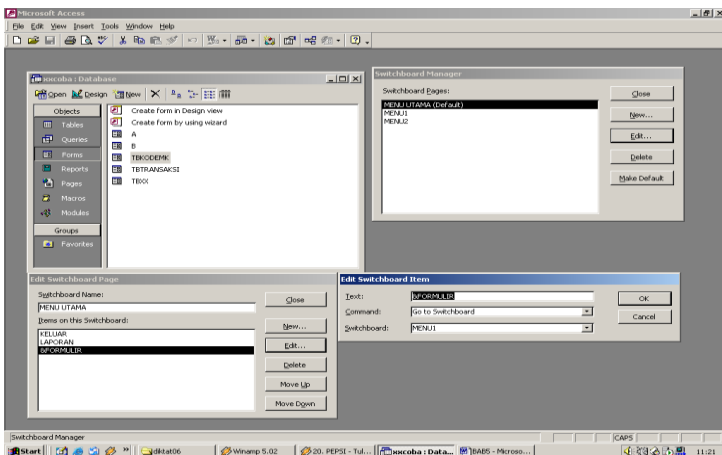
1. Aktifkan file database yang ingin dilengkapi menu navigasi
2. Pilih Menu tool, database utilities, switchbord Manager

Contoh kotak dialog switchbord



1. Tulislah rancanganya seperti diatas, untuk nama halaman “ MAIN SWICTBOARD DEFAULT “ ganti dengan MENU UTAMA klik perintah EDIT kemudian CLOSE

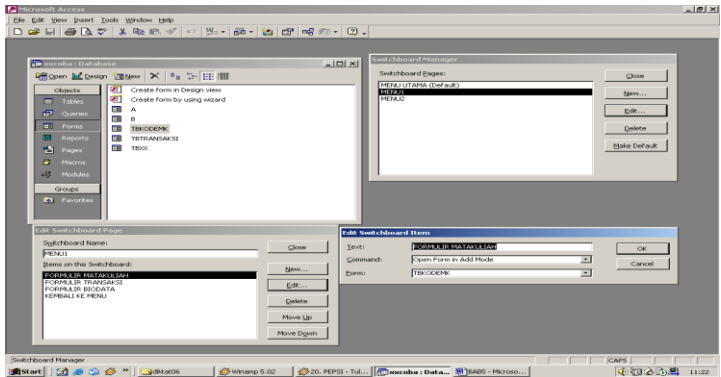
TEXT	COMMAND	SWITHBOARD
&FORMULIR DATA	GOTO SWICTBOARD	MENU 1
LAPORAN UTAMA	GOTO SWICTBOARD	MENU 2
KELUAR	EXIT APPLICATION	



2. Buat halaman baru switchboard dengan perintah NEW seperti pada gambar dibawah ini
3. Membuat form baru switboard

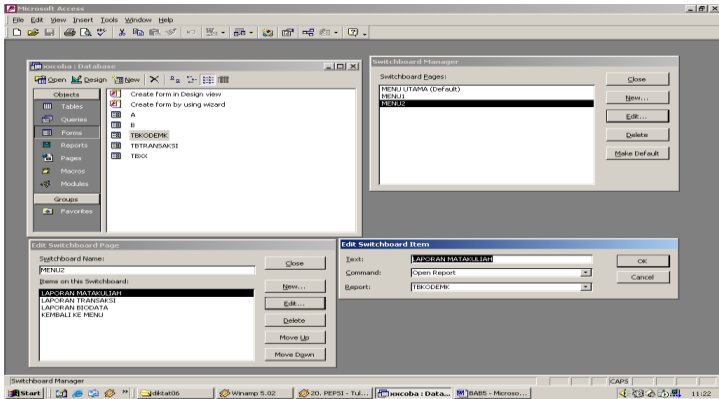
TEXT	COMMAND	SWITHBOARD
&FORMULIR BIODATA MAHASISWA	OPEN FORM IN EDIT MODE	SESUAI DENGAN NAMA FORMULIR TABEL FORM (FORTBBIO)
FORMULIR MATAKULIAH	OPEN FORM IN EDIT MODE	SESUAI DENGAN NAMA FORMULIR TABEL FORM (FORTBMK)

FORMULIR TRANSAKSI	OPEN FORM IN EDIT MODE	SESUAI DENGAN NAMA FORMULIR TABEL FORM (FORTRANS)
KEMBALI KE MENU	GOTO SWICTBOARD	MENU UTAMA

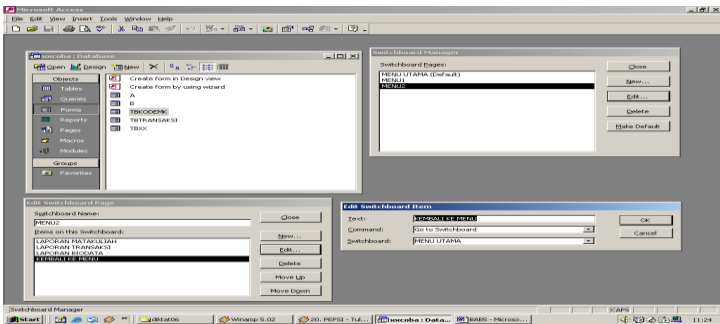


4. Membuat Report / laporan pada switboard

TEXT	COMMAND	SWITHBOARD
LAPORAN BIODATA MAHASISWA	OPEN REPORT	SESUAI DENGAN NAMA FORMULIR TABEL FORM (REPTBBIO)
LAPORAN MATAKULIAH	OPEN REPORT	SESUAI DENGAN NAMA FORMULIR TABEL FORM (REPTBMK)
LAPORAN TRANSAKSI	OPEN REPORT	SESUAI DENGAN NAMA FORMULIR TABEL FORM (REPTRANS)
KEMBALI KE MENU	GOTO SWICTBOARD	MENU UTAMA

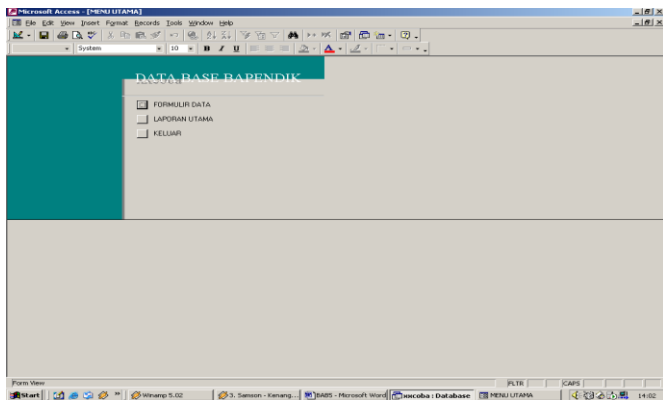


CLOSE



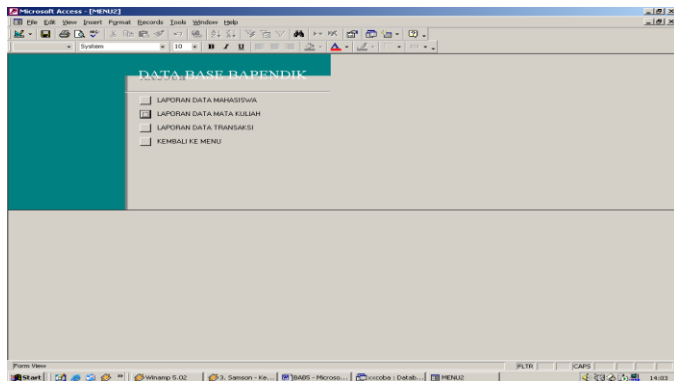
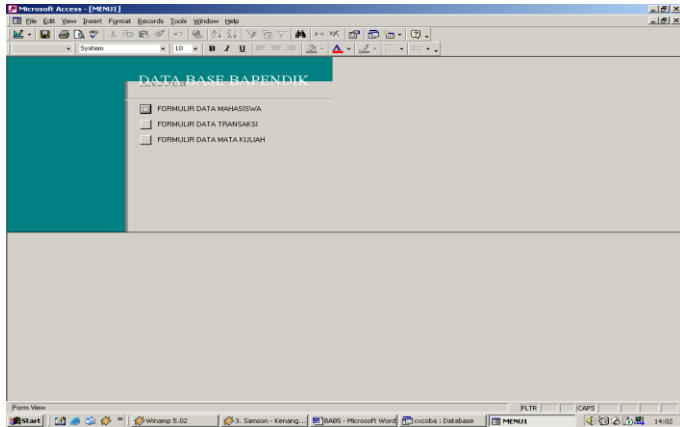
1. HASIL DARI SWITCHBORD MENU UTAMA

- a. Membuka informasi data formulir
- b. Membuka informasi data laporan
- c. Keluar dari database



2. HASIL DARI SWITCBORD MENU 1

- a. Formulir biodata mahasiswa
- b. Formulir daftar matakuliah
- c. Formulir transaksi



3. HASIL DARI SWITCBORD MENU 2

- a. Laporan biodata mahasiswa
- b. Laporan daftar matakuliah
- c. Laporan transaksi
- d. Kembali ke menu

Menjalankan Hasil Menu Navigasi Berupa Switbord & Design

Menjalankan menu navigasi berupa switboard

1. Buka file database kemudian jendela kerja database klik forms yang ada dibawah object atau pilih menu view database objects forms
2. Pilih switbord kemudian klik tombol toolbar open atau klik 2 X pada switboard

Apabila hasil dari menu kurang memuaskan bisa di modifikasi sesuai dengan selera seni masing-masing yaitu dengan cara :

Buka file database yang anda inginkan kemudian pada jendela kerja database, klik form yang ada dibawah object atau menu view database object form, klik switbord kemudian tombol toolbar design / form design.

Beberapa Istilah Database

1. 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.

2. Field

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

3. Record

Field merupakan sebuah kumpulan nilai yang saling terkait.

4. 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.

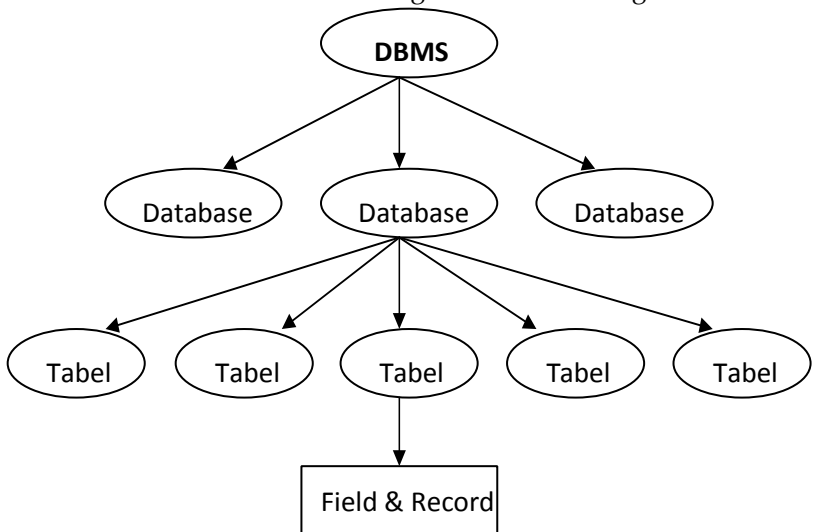
5. SQL

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

6. Hierarki Database

Dalam konsep database, urutan atau hierarki database sangatlah penting.

Urutan atau hierarki database digambarkan dalam gambar sbb:



BAB 2

INSTALASI MYSQL DAN SOFTWARE PENDUKUNG

- ❖ Instalasi MySQL di Windows
- ❖ Instalasi Software Pendukung MySQL

A. 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 :**

1. **Relational Database System.** Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.
2. **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.

3. **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.
4. Mendukung **Sub Select.** Mulai versi 4.1 MySQL telah mendukung select dalam select (sub select).
5. Mendukung **Views.** MySQL mendukung views sejak versi 5.0
6. Mendukung **Stored Prosedured (SP).** MySQL mendukung SP sejak versi 5.0
7. Mendukung **Triggers.** MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL telah meningkatkan kemampuan trigger pada versi 5.1.
8. Mendukung **replication.**
9. Mendukung transaksi.
10. Mendukung **foreign key.**
11. Tersedia fungsi GIS.
12. Free (bebas didownload)
13. Stabil dan tangguh
14. Fleksibel dengan berbagai pemrograman
15. Security yang baik
16. Dukungan dari banyak komunitas
17. Perkembangan software yang cukup cepat.

B. Instalasi MySQL di Windows

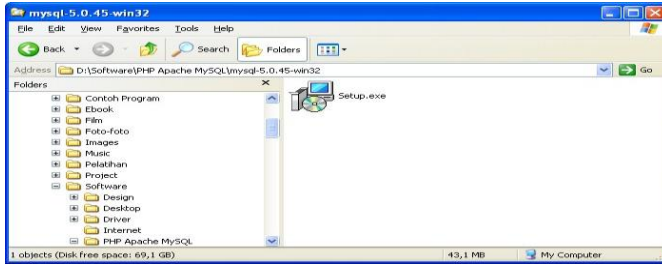
1. Persiapan

- a. *Download Source MySQL di <http://www.mysql.com/downloads/>*

Versi MySQL yang digunakan dalam penulisan adalah MySQL 5 Silahkan Anda download versi terakhir tersebut dan simpan di komputer Anda. Pada dasarnya, instalasi untuk setiap versi MySQL tidak jauh berbeda.

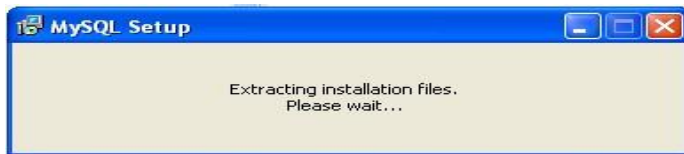
2. Proses Instalasi MySQL

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



Gambar 2.1. File Setup.exe

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



Gambar 2.2. Proses Instalasi Dimulai

- d. 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

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



Gambar 2.4. Pilih tipe instalasi

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



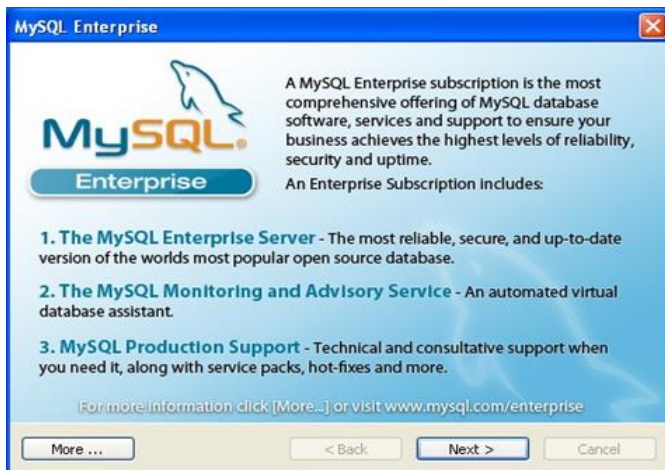
Gambar 2.5. Window Informasi Konfigurasi Instalasi

g. Proses instalasi dimulai.

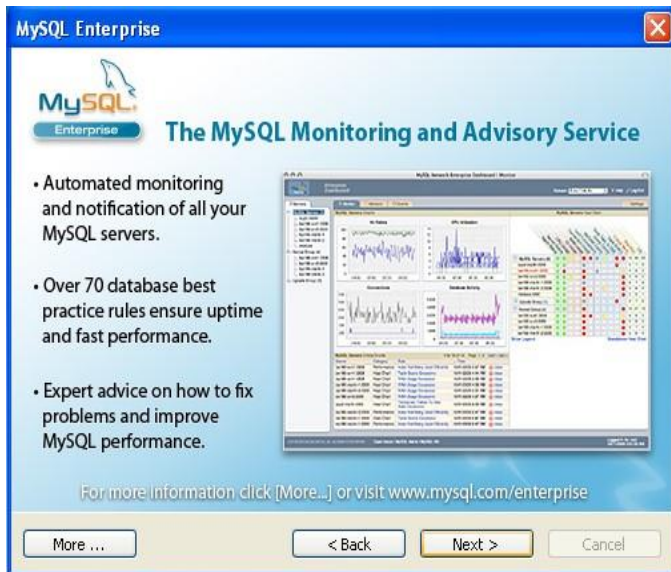


Gambar 2.6. Window Setup Type

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



Gambar 2.7. Window MySQL Enterprise



Gambar 2.8. Window informasi MySQL Monitoring

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



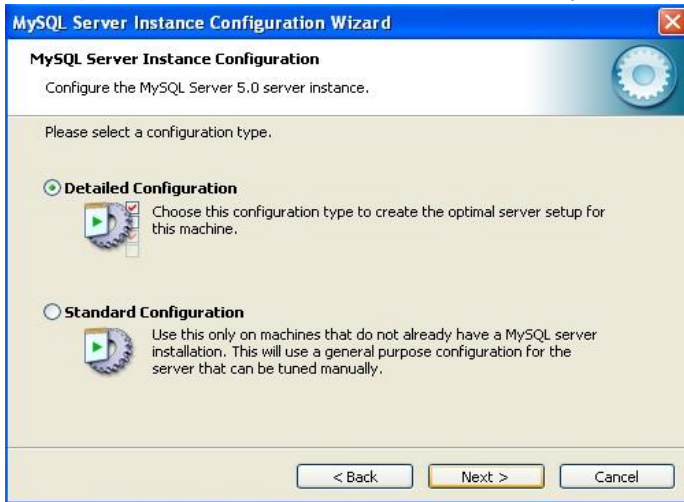
Gambar 2.9. Proses instalasi Selesai

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



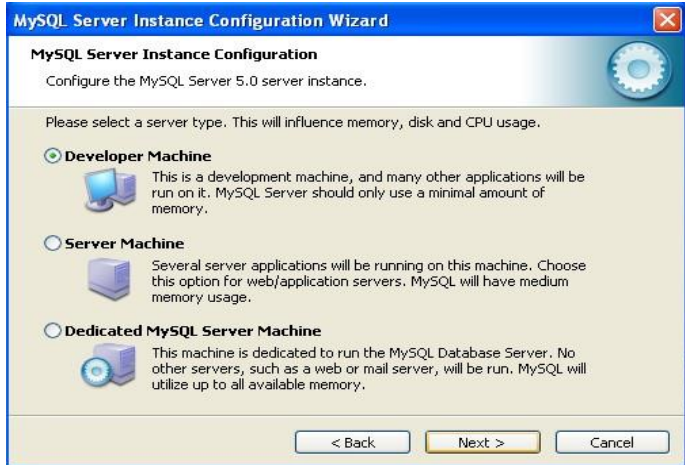
Gambar 2.10. Window MySQL Server Instance Configuration Wizard

- k. 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

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



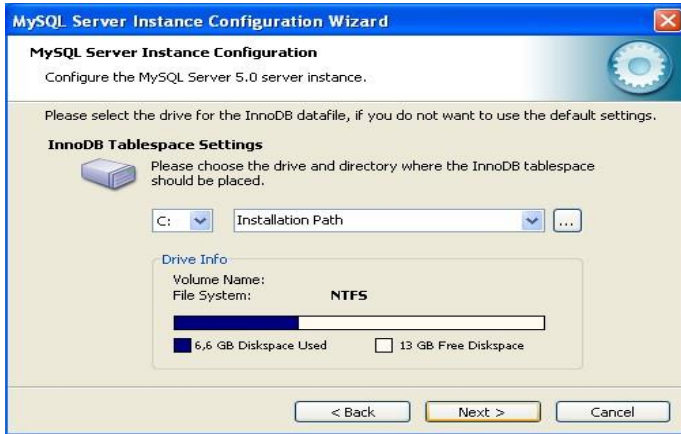
Gambar 2.12. Pilihan tipe server MySQL

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



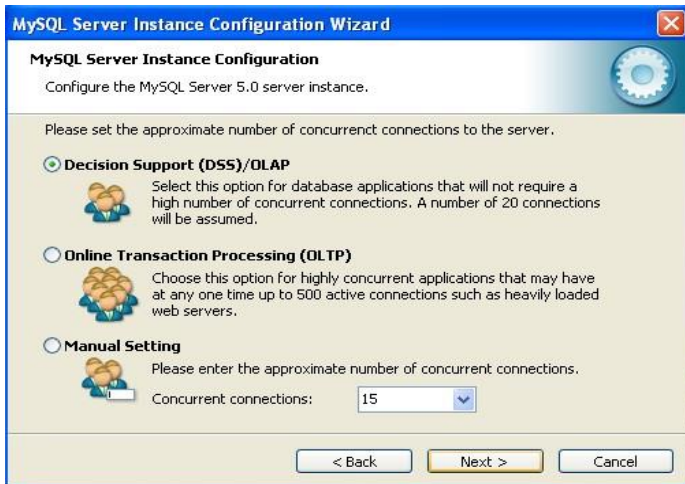
Gambar 2.13. Window Pilihan penggunaan Database.

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



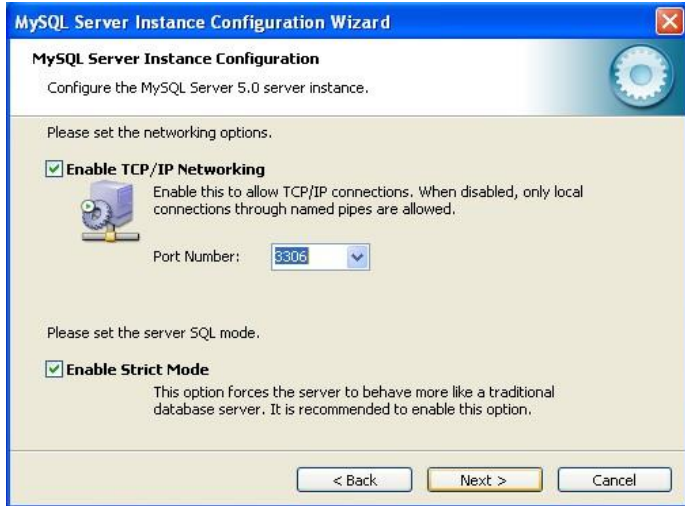
Gambar 2.14. Window InnoDB Tablespace Settings.

- o. 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

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



Gambar 2.16. Window pilihan port MySQL.

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



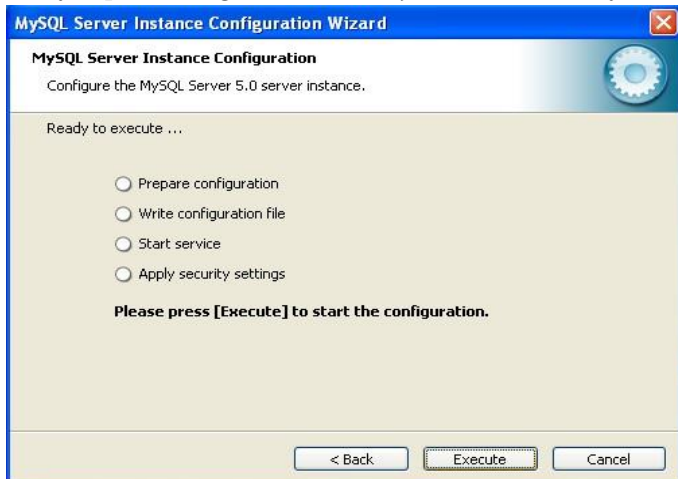
Gambar 2.17. Window pilihan Nama Service MySQL.

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



Gambar 2.18. Window Security Setting.

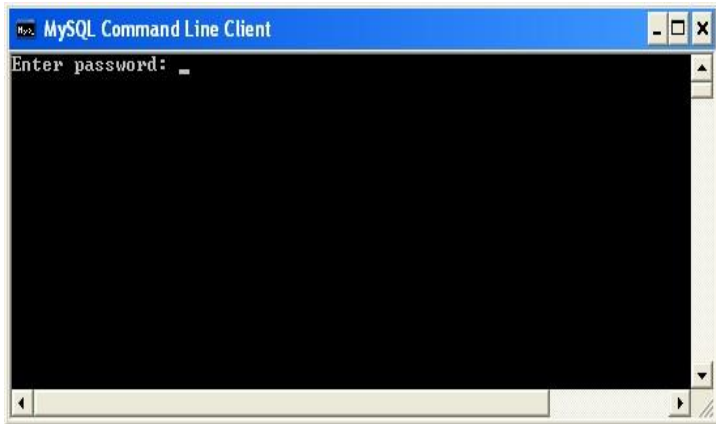
- s. Proses konfigurasi selesai dan klik **Execute** untuk menyimpan konfigurasi dan menjalankan servis MySQL.



Gambar 2.19. Proses konfigurasi server MySQL selesai.

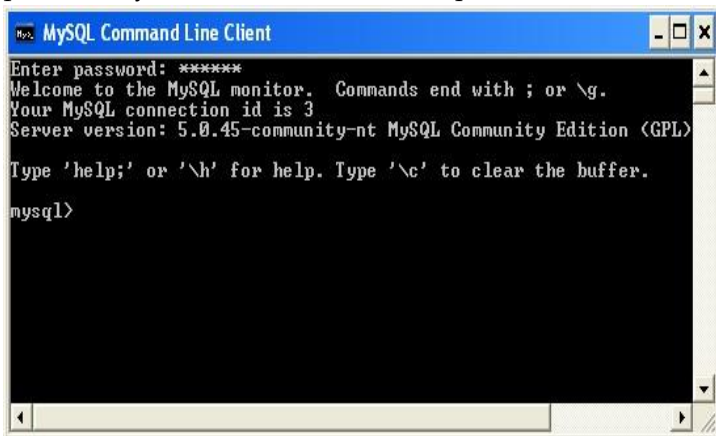
3. Koneksi ke Server MySQL dengan MySQL Client

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

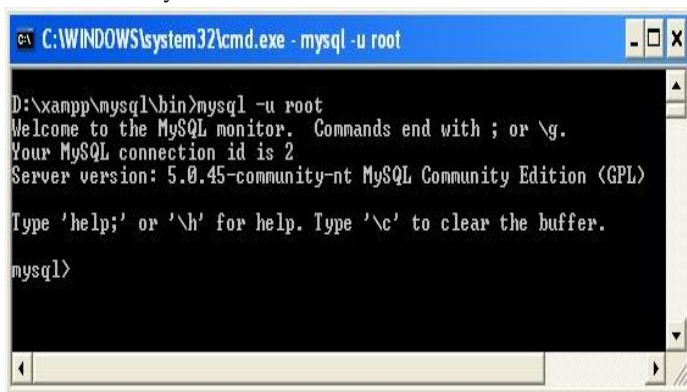
Setelah koneksi ke server MySQL berhasil dilakukan, maka akan ditampilkan prompt **mysql>** seperti pada gambar 12.14. Query atau perintah-perintah MySQL dapat dituliskan pada prompt MySQL ini. Akhiri setiap query dengan titik-koma (;). Selanjutnya untuk keluar dari server MySQL dapat dilakukan dengan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

4. 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.

a. 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*.

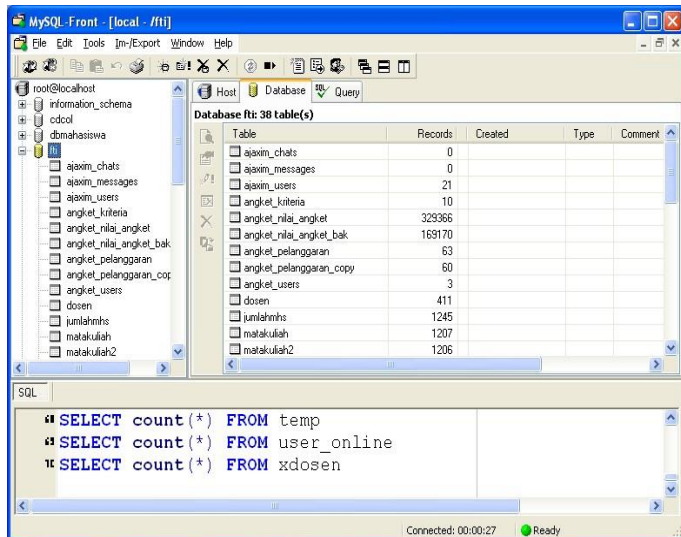


Gambar 2.21. Tampilan MySQL command line client

b. 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

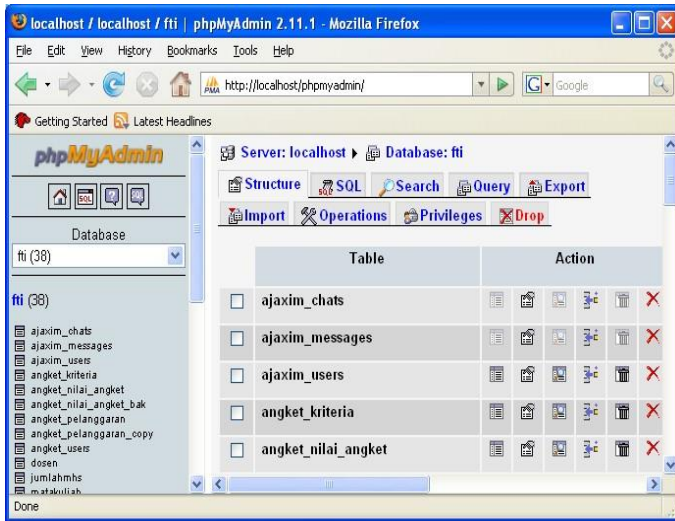
awalnya MySQL-Front merupakan software yang free, namun mulai versi 3.0 ke atas, software ini menjadi software yang bersifat shareware dengan masa percobaan selama 30 hari. Jika Anda ingin mencoba software ini, cobalah MySQL-Front versi 2.5 karena selain masih bebas untuk didownload, versi 2.5 cukup stabil dan sudah teruji. Situs resmi MySQL-Front beralamat di <http://www.mysqlfront.de>



Gambar 2.22. Tampilan MySQL Front

c. PHPMyAdmin

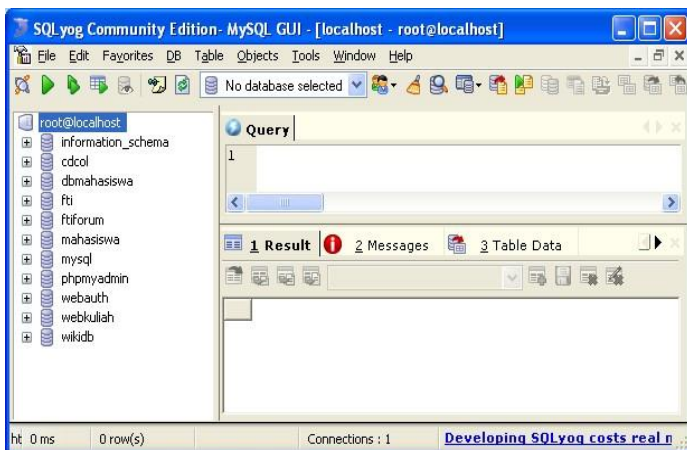
PHPMyAdmin merupakan front-end MySQL berbasis web. PHPMyAdmin dibuat dengan menggunakan PHP. Saat ini, PHPMyAdmin banyak digunakan dalam hampir semua penyedia hosting yang ada di internet. PHPMyAdmin mendukung berbagai fitur administrasi MySQL termasuk manipulasi database, tabel, index dan juga dapat mengekspor 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

d. 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

e. **MySQL Administrator dan MySQL Query Browser**

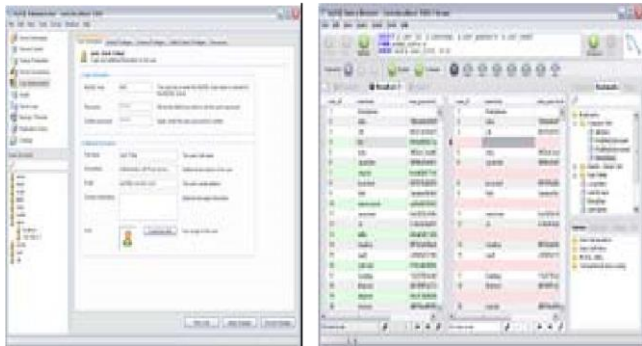
MySQL Administrator dan MySQL Query Browser merupakan tools administrasi database MySQL yang tersedia di situs resmi MySQL (<http://www.mysql.com>). Keduanya dapat didownload di alamat <http://www.mysql.com/products/tools/>.

Beberapa fitur MySQL Administrator, antara lain:

- 1) Administrasi user.
- 2) Halaman monitoring server.
- 3) Optimisasi MySQL
- 4) Informasi umum keadaan server
- 5) Status *replication*.
- 6) *Cross-platform*.

Beberapa fitur MySQL Query Browser, antara lain:

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



Gambar 2.25. Tampilan layar MySQL Administrator dan MySQL Control Center

BAGIAN 2

DASAR-DASAR MYSQL

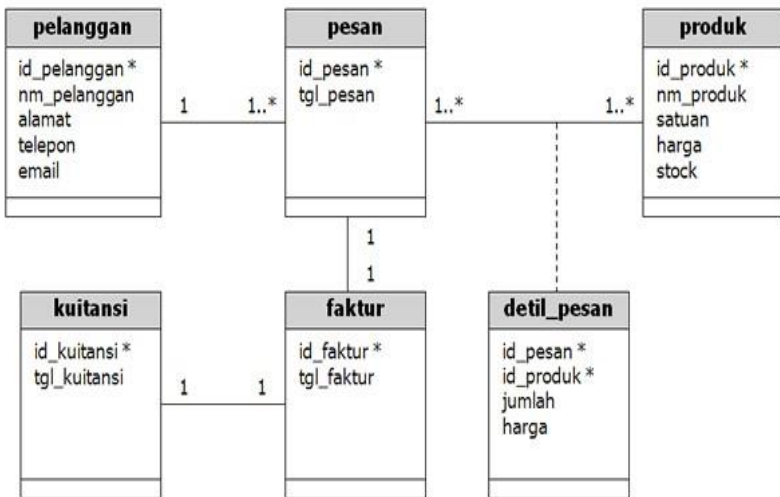
BAB 3.

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

A. 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.

1. 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 dynamic.** 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 `mysamchk`. Tentunya hasilnya lebih kecil dari segi ukuran. Tabel yang terkompresi tidak dapat dikenakan operasi seperti INSERT, UPDATE dan DELETE.

2. 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.

3. 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:

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

B. 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-andtime, dan kelompok himpunan (set dan enum). Masing-masing tipe field memiliki batasan lebar dan ukurannya.

1. 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 penambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe numerik:

a. TINYINT

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

Jangkauan : -128 s/d 127

Ukuran : 1 byte (8 bit).

b. SMALLINT

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

Jangkauan : -32.768 s/d 32.767

Ukuran : 2 byte (16 bit).

c. 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).

d. 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).

e. BIGINT

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

Jangkauan : $\pm 9,22 \times 10^{18}$

Ukuran : 8 byte (64 bit).

f. FLOAT

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif & 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).

g. DOUBLE

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

Jangkauan : -1.79...E+308 s/d -2.22...E-308, 0, dan 2.2...E-308 s/d 1.79...E+308.

Ukuran : 8 byte (64 bit).

h. REAL, Merupakan sinonim dari DOUBLE.

i. DECIMAL

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

Jangkauan : -1.79...E+308 s/d -2.22...E-308, 0, dan 2.2...E-308 s/d 1.79...E+308.

Ukuran : 8 byte (64 bit).

j. NUMERIC, merupakan sinonim dari DECIMAL.

2. 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*:

a. DATE

Penggunaan : digunakan untuk menyimpan data tanggal.

Jangkauan : 1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)

Ukuran : 3 byte.

b. TIME

Penggunaan : digunakan untuk menyimpan data waktu.

Jangkauan : -838:59:59 s/d +838:59:59 (HH:MM:SS)

Ukuran : 3 byte.

c. 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.

d. YEAR

Penggunaan : digunakan untuk menyimpan data tahun dari tanggal.

Jangkauan : 1900 s/d 2155

Ukuran : 1 byte.

3. 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:

a. CHAR

Penggunaan : digunakan untuk menyimpan data string ukuran tetap.

Jangkauan : 0 s/d 255 karakter

b. 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)

c. 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)

d. TEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d 65.535 ($2^{16} - 1$) karakter

e. MEDIUMTEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d $2^{24} - 1$ karakter

f. LONGTEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d $2^{32} - 1$ karakter

4. 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:

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

- b. TINYBLOB
Penggunaan : digunakan untuk menyimpan data biner.
Jangkauan : 255 byte

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

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

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

5. 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:

- a. ENUM
Penggunaan : enumerasi (kumpulan data).
Jangkauan : sampai dengan 65535 string.

- b. SET
Penggunaan : combination (himpunan data).
Jangkauan : sampai dengan 255 string anggotas.

C. 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.

1. Beberapa Aturan Merancang Database yang Baik.

- a. Tabel dalam database tidak boleh mengandung record (data) ganda, atau dengan kata lain tidak boleh ada redundancy 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.

2. Tips Penamaan Identifier.

- a. Penamaan identifier (database, tabel, kolom) di MySQL bersifat casesensitive. 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 üàü, 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

A. 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 ANSI 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 :

- a. CREATE
- b. ALTER
- c. RENAME
- d. 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 :

- a. SELECT
- b. INSERT

- c. UPDATE
- d. 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 :

- a. GRANT
- b. REVOKE

B. Membuat, Menampilkan, Membuka dan Menghapus Database

1. 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)
```

2. 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)
```

3. 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
```

4. 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;
```

C. Membuat, Mengubah dan Menghapus *Table*

1. 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 :

- a. CREATE TABLE merupakan perintah dasar dari pembuatan table.
- b. pelanggan merupakan nama tabel yang akan dibuat.

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

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

a. 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 :

b. 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 mahasiswa 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**.

2. 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 :

a. Menambahkan field “**tgllahir**” ke tabel **pelanggan**

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

b. Menambahkan primary key pada suatu tabel

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

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

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

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

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

e. 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.

3. 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;
```

a. 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)
```

b. 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 :

- 1) UPDATE merupakan perintah dasar untuk mengubah *record* tabel.
- 2) nama_tabel merupakan nama tabel yang akan diubah *recordnya*.
- 3) 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 masingmasing *field*.
- 4) 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';
```

c. 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 :

- 1) DELETE FROM merupakan perintah dasar untuk menghapus suatu *record* dari tabel.
- 2) nama_tabel merupakan nama tabel yang akan dihapus *recordnya*.
- 3) 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';
```

D. 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 :

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

- a. **Operator &&** atau **AND**, akan menghubungkan dua kondisi dimana akan bernilai TRUE jika kedua kondisi bernilai TRUE.
 - b. **Operator ||** atau **OR**, akan menghubungkan dua kondisi dimana akan bernilai TRUE jika salah satu atau kedua kondisi bernilai TRUE.
 - c. **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

A. Fungsi String

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

1. 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 |
+-----+-----+
```


2. **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
```

3. **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
```

4. **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
```

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
```

6. *RIGHT (string, panjang)*

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

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

Hasil keluarannya:

```
Uhur
```

7. *LTRIM (string)*

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

Contoh:

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

Hasil keluarannya:

```
Budi Luhur
```

8. *RTRIM (string)*

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

Contoh:

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

Hasil keluarannya:

```
|Budi Luhur
```

9. *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
```

10. *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
```

11. *REPEAT (string, jumlah)*

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

Contoh:

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

Hasil keluarannya:

```
|MontMontMont
```

12. REVERSE (string)

Fungsi ini digunakan untuk membalik string. Contoh:

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

Hasil keluarannya:

```
moc.lqsym
```

13. LCASE (string) LOWER (string)

Fungsi ini digunakan untuk mengubah string menjadi huruf kecil (lowercase). Contoh:

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

Hasil keluarannya:

```
Mysql
```

14. UCASE (string)

15. UPPER (string)

Fungsi ini digunakan untuk mengubah string menjadi huruf kapital (uppercase). Contoh:

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

Hasil keluarannya:

```
MYSQL
```

B. 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.

1. *NOW () SYSDATE()*

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

```
SELECT NOW();
```

Hasil keluarannya:

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

2. *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
```

3. *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
```

4. *YEAR (tanggal)*

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

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

Hasil keluarannya:

```
|2008
```

5. **HOUR** (*waktu*)

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

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

Hasil keluarannya:

```
|20
```

6. **MINUTE** (*waktu*)

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

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

Hasil keluarannya:

```
|8
```

7. **SECOND** (*waktu*)

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

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

Hasil keluarannya:

```
|53
```

8. *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
```

9. *DATE_FORMAT(date, format)*

Fungsi ini digunakan untuk mem-format tampilan tanggal.

10. *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
```

C. Fungsi Numerik

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

1. OPERASI ARITMATIKA

Operasi aritmatika dalam MySQL terdiri dari:

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

Contoh penggunaan:

```
SELECT 10+20;
```

Hasil keluarannya:

```
30
```

```
SELECT 10/3;
```

Hasil keluarannya:

```
3.3333
```

2. ABS(x)

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

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

Hasil keluarannya:

```
20
```


3. MOD(m, n)

Fungsi digunakan untuk mengoperasikan m modulus n. Contoh:

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

Hasil keluarannya:

```
1
```

4. FLOOR(x)

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

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

Hasil keluarannya:

```
10
```

5. CEILING(x)

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

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

Hasil keluarannya:

```
11
```

6. 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
```

7. POW(x) POWER(x, n)

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

Contoh:

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

Hasil keluarannya:

```
1024
```

8. RAND() RAND(x)

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

```
SELECT RAND();
```

Hasil keluarannya:

```
0.96589817662341
```

9. TRUNCATE(x, d)

Fungsi digunakan untuk memotong bilangan x sepanjang d tempat desimal.

Contoh:

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

Hasil keluarannya:

```
10.2
```

D. Fungsi Lainnya

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

1. **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
```

2. **COUNT(range)**

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

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

Hasil keluarannya:

```
5
```

3. **MAX(range)**

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

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

Hasil keluarannya:

```
93
```

4. **MIN(range)**

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

Contoh:

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

Hasil keluarannya:

```
|40
```

5. SUM(range)

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

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

Hasil keluarannya:

```
|450
```

6. AVG(range)

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

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

Hasil keluarannya:

```
|78
```

7. 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
```

8. DATABASE()

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

```
SELECT DATABASE();
```

Hasil keluarannya:

```
Penjualan
```

9. USER() SYSTEM_USER() SESSION_USER()

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

Contoh:

```
SELECT USER();
```

Hasil keluarannya:

```
root@localhost
```

10. 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-decrypt (decode). Biasanya fungsi ini digunakan untuk menyimpan password login. Contoh:

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

Hasil keluarannya:

```
*AA1420F182E88B9E5F874F6FBE7459291E8F4601
```

11. ENCODE(str, pass)

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

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

Hasil keluarannya:

```
câT♠e |
```

12. DECODE(encrypted_str, pass)

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

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

Hasil keluarannya:

```
Qwerty
```

Contoh dengan password salah:

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

Hasil keluarannya:

```
WkΦPH:
```

13. 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
```

14. 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
```

15. VERSION()

Fungsi digunakan untuk mengambil versi MySQL yang digunakan. Contoh:

```
|SELECT VERSION();
```

Hasil keluarannya:

```
|5.0.45-community-nt
```

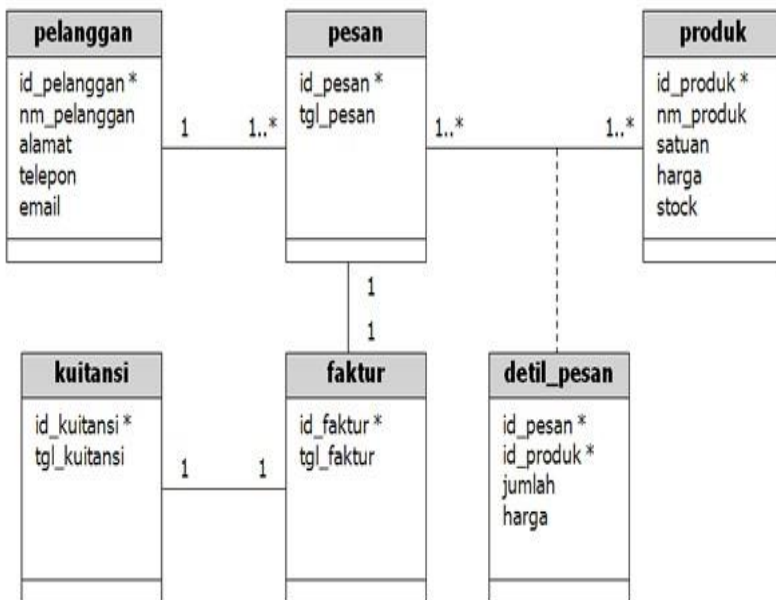
BAGIAN 3
PERINTAH MYSQL
LANJUTAN

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-rollback). 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 detail_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_pesanan */
DROP TABLE IF EXISTS detil_pesanan; CREATE TABLE detil_pesanan (
id_pesanan 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_pesanan,id_produk),
KEY FK_detil_pesanan (id_produk),

KEY id_pesanan (id_pesanan),
CONSTRAINT FK_detil_pesanan FOREIGN KEY (id_produk)
REFERENCES produk (id_produk),
CONSTRAINT FK_detil_pesanan2 FOREIGN KEY (id_pesanan)
REFERENCES pesanan (id_pesanan)
) 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_pesanan int(5) NOT
NULL, tgl_faktur date NOT NULL, PRIMARY KEY (id_faktur),
KEY id_pesanan (id_pesanan),
CONSTRAINT faktur_ibfk_1 FOREIGN KEY (id_pesanan)
REFERENCES pesanan (id_pesanan)
) 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.

A. 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 antartabel 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.

1. 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

a. *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.

b. 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 |
+-----+-----+-----+-----+
```

2. 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:

3. 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.

4. 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.

5. 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

B. 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_pesanan, pesan.tgl_pesanan,
SUM(detil_pesanan.jumlah) as jumlah
FROM pesan, detil_pesanan
WHERE pesan.id_pesanan=detil_pesanan.id_pesanan GROUP BY
id_pesanan;
```

Hasilnya :

id_pesanan	tgl_pesanan	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 |
+-----+-----+-----+
```

C. 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 agregasi 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   |
+-----+-----+
```

D. 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.

1. 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    |
+-----+-----+
```

2. 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     |
+-----+-----+
```

E. 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 |
+-----+-----+-----+
```

F. 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:

1. **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.
2. **Consistency.** Bahwa kegagalan satu proses dalam transaksi tidak akan mempengaruhi transaksi lainnya.
3. **Isolation.** Secara sederhana, bahwa data yang sedang digunakan dalam satu transaksi, tidak dapat digunakan oleh transaksi lainnya sebelum seluruh proses transaksi yang pertama selesai.
4. **Durability.** Jika sebuah transaksi selesai dieksekusi, hasilnya tetap tercatat dengan baik.

1. 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.

a. Bentuk umum perintah LOCK :

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

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

- 1) **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*.
- 2) **READ LOCAL**. Sama seperti **READ**, tetapi perintah **INSERT** dapat dilakukan selama tidak merubah record (data) yang ada.
- 3) **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*).
- 4) **LOW PRIORITY WRITE**. Seperti halnya **WRITE**, hanya saja user lain dapat melakukan **READ LOCK**. Untuk membuka tabel yang terkunci, gunakan perintah **UNLOCK TABLES**;

c. 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;
```

2. 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 (*nestedtransaction*), 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).

a. Bentuk umum perintah transaksi :

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

b. 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

A. 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:

1. 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.
2. 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:
 - a. 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.

b. Gunakan perintah SHOW GRANTS untuk melihat semua hak akses user.

3. 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*).

4. 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.

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

6. 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.

7. 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.

B. 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:

1. **user.** Tabel ini digunakan untuk menyimpan informasi user MySQL yang mencakup informasi user, password dan host user, serta informasi hak akses user.
2. **db.** Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap database.
3. **host.** Tabel ini digunakan untuk menyimpan daftar komputer (bisa berupa alamat IP, nama komputer, atau %) yang berhak mengakses suatu database.
4. **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.
5. **columns_priv.** Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap kolom.
6. **procs_priv.** Tabel ini digunakan untuk menyimpan informasi mengenai hak akses user terhadap procedure.
7. **proc.** Tabel ini digunakan untuk menyimpan informasi mengenai daftar procedure dalam MySQL.
8. **func.** Tabel ini digunakan untuk menyimpan informasi mengenai function yang didefinisikan di MySQL.

C. 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**.

D. 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:

1. 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;
```

2. 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';
```

3. 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:

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

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

```
FLUSH PRIVILEGES;
```

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

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

```
FLUSH PRIVILEGES;
```

E. 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;
```

F. 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

A. 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.

1. 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.

2. 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 ;
```

3. Menghapus Trigger

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

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

Contoh :

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

B. 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.

1. 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

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`)
```

3. 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:

- 1. 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.
- 2. 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.
- 3. 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 menghasilkan value, biasanya hanya menampilkan saja.

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

A. 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! |  
+-----+
```

B. Membuat, Mengubah dan Menghapus SP

1. 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 |
+-----+
```

2. Mengubah SP

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

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

3. Menghapus SP

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

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

C. 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

1. 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 tg1_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 ;
```

2. 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 ;
```

3. 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**.

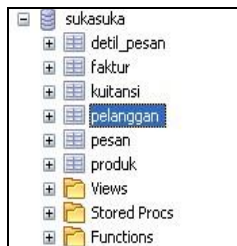
Laporan dari database umumnya didasarkan pada hasil (results) suatu perintah query SELECT. Dengan SELECT kita dapat menyeleksi tampilan atau hasil yang diinginkan dari tabel-tabel yang ada di database. SELECT dapat berupa seleksi ke satu tabel atau beberapa tabel sekaligus (join antar tabel).

A. Laporan dari Tabel dan Query

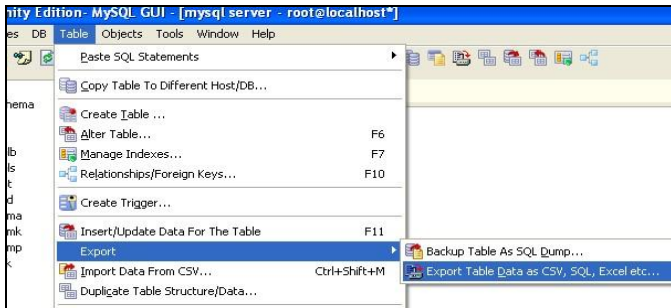
Dengan menggunakan SQLYog, kita dapat membuat laporan dari tabel (*table-data*) dan juga dari hasil query (*results*). Laporan dari tabel merupakan laporan yang menampilkan seluruh data dari suatu tabel, sedangkan laporan dari query merupakan laporan yang menampilkan seluruh data dari hasil suatu perintah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

1. Laporan dari Tabel

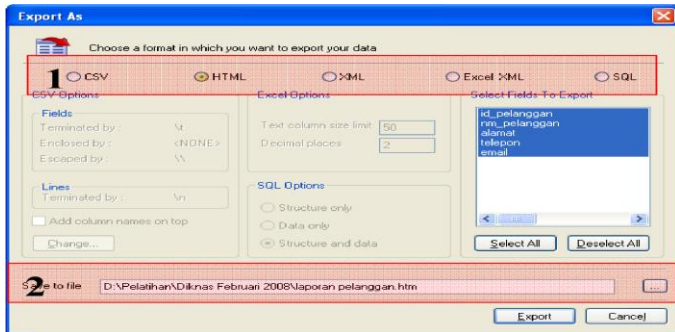
- Pilih tabel yang akan dibuat laporan.



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



- c. Pilih format laporan dan tempat menyimpan hasil laporan.



- d. Klik tombol Export.
e. Buka File hasil laporan.

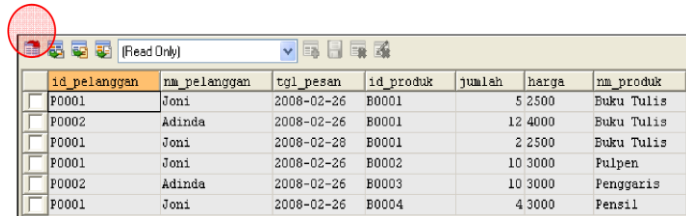


2. Laporan dari Hasil Query

- a. Ketikkan perintah query di jendela query.

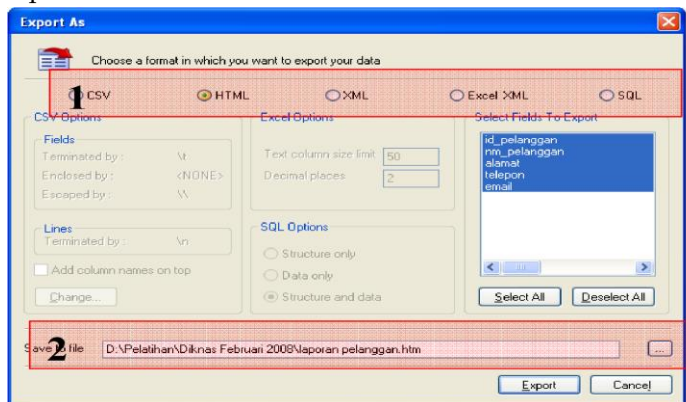
```
transaksi.sql*
18 |SELECT
19 |  pelanggan.id_pelanggan,
20 |  pelanggan.nm_pelanggan,
21 |  pesan.tgl_pesan,
22 |  detail_pesan.id_produk, detail_pesan.jumlah,
23 |  detail_pesan.harga,
24 |  produk.nm_produk
25 |FROM
26 |  pelanggan, pesan, detail_pesan, produk
27 |WHERE
28 |  pelanggan.id_pelanggan=pesan.id_pelanggan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

- b. 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

- c. Pilih format laporan dan tempat menyimpan hasil laporan.



- d. Klik tombol Export.
- e. Buka File hasil laporan.

query data - Mozilla Firefox

file:///D:/Pelatihan/Diknas%20Februari%202008/laporan%20transaksi.htm

Getting Started Latest Headlines

query result(6 records)

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

Done

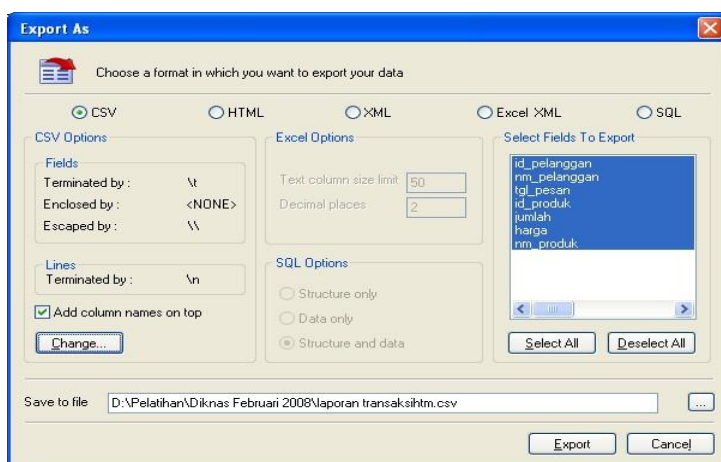
B. Format Laporan

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

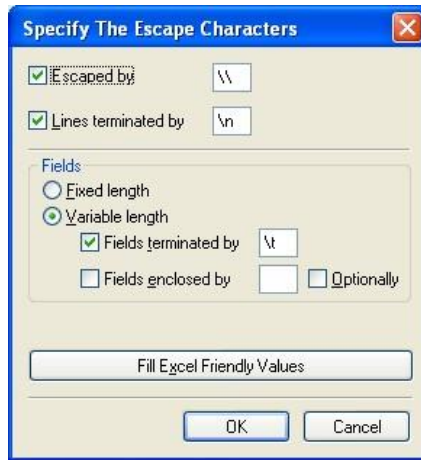
1. CSV

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.

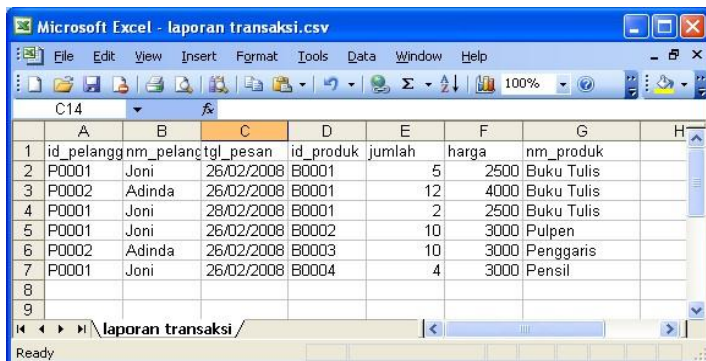
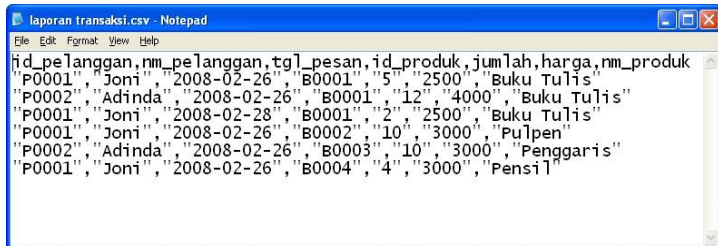
Berikut ini pilihan untuk membuat report dalam CSV.



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.



2. HTML

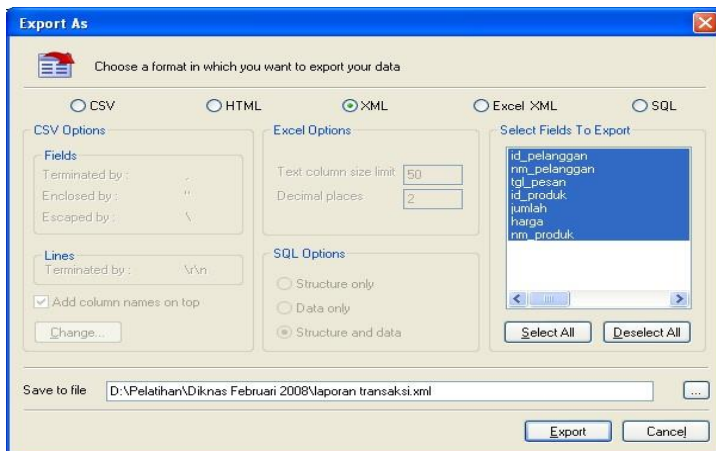
HTML merupakan singkatan dari *HyperText Markup Language*. HTML merupakan yang dapat dibuka dengan browser (IE, Mozilla dll). Karena sifatnya yang kompatibel dengan browser maka format ini cocok dipilih jika kita menginginkan laporan dalam bentuk halaman web/internet.

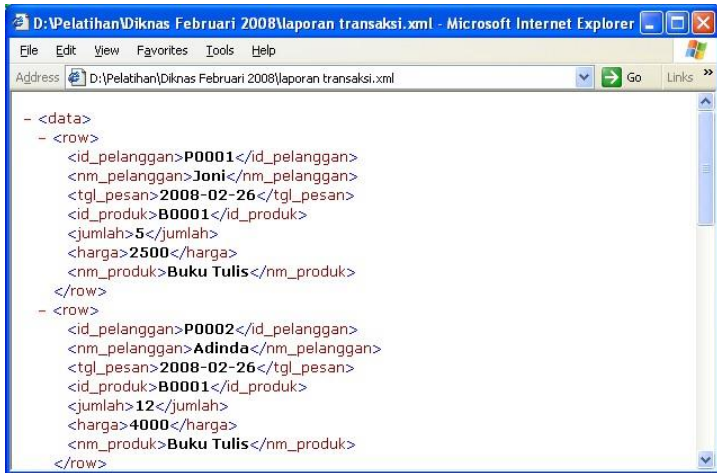
Bagaimana cara membuat laporan dalam format HTML sudah dijelaskan di bagian sebelumnya. Demikian juga tampilannya di halaman web (browser) dapat dilihat di materi sebelumnya.

3. 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.

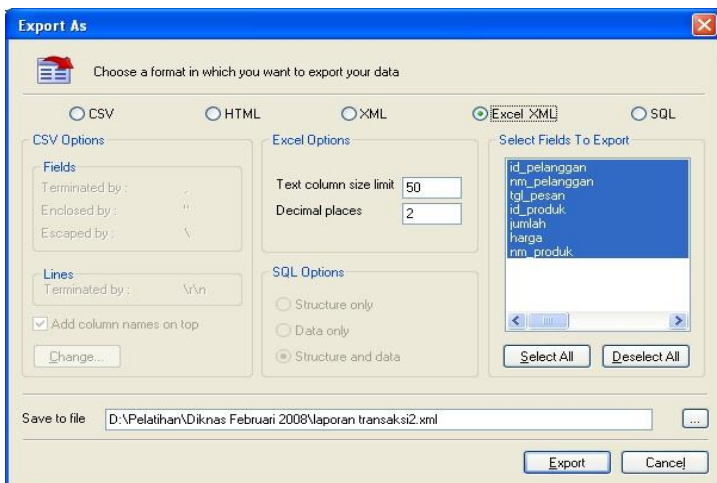




4. Excel XML

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).



Microsoft Excel - laporan transaksi2.xml

File Edit View Insert Format Tools Data Window Help

A1 id_pelanggan

	A	B	C	D	E	F	G	H
1	id_pelanggan	nm_pelanggan	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								

Sheet1

Ready

BAB 11.

BACKUP, RESTORE DAN IMPORT DI MYSQL

- ❖ Backup
- ❖ Restore
- ❖ Import

A. 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 *direplikasi* ke server kedua. Selain itu, kita juga dapat melakukan backup otomatis dengan bantuan software tertentu. Biasanya kita dapat membuat *schedulebackup*. 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.

1. 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.

2. 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:

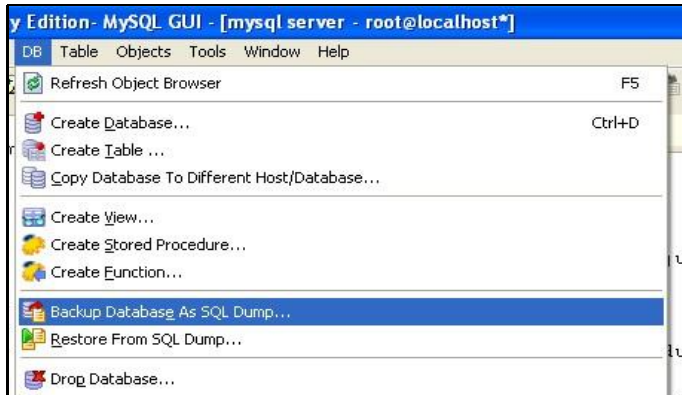
```
shell> mysqldump--tab=/path/to/some/dir--opt db_name
shell> mysqlhotcopy db_name /path/to/some/dir
```

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

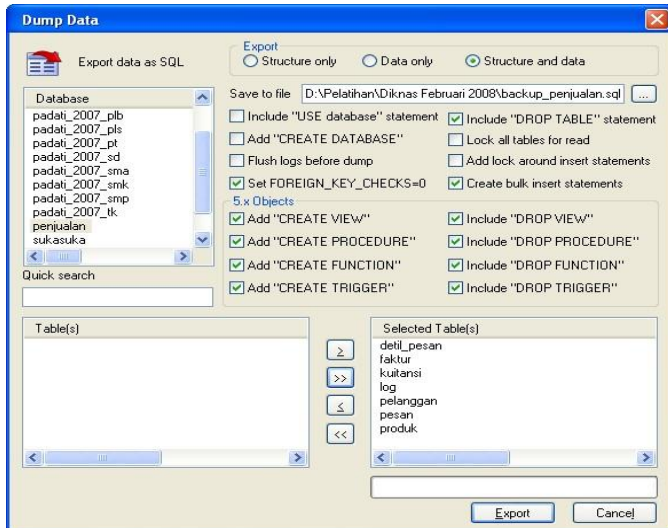
```
shell > mysqldump --all-databases > backup.sql
```

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

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



- b. 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.



- c. Tekan tombol **Export** untuk mengeksekusi backup. Hasil backup ini berupa file dengan ekstension .SQL. File tersebut dapat *direstore* di kemudian hari.

B. Restore

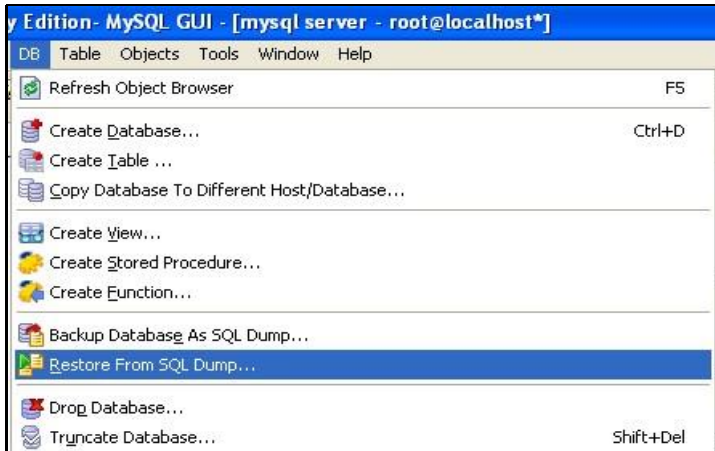
Restore merupakan prosedur yang dilaksanakan untuk mengembalikan file yang dibackup ke database MySQL. Proses restore juga dapat dilakukan melalui perintah SQL dengan memanfaatkan **mysql** dan juga bisa menggunakan software bantuan.

Jika menggunakan sintaks SQL, berikut ini contoh perintahnya:

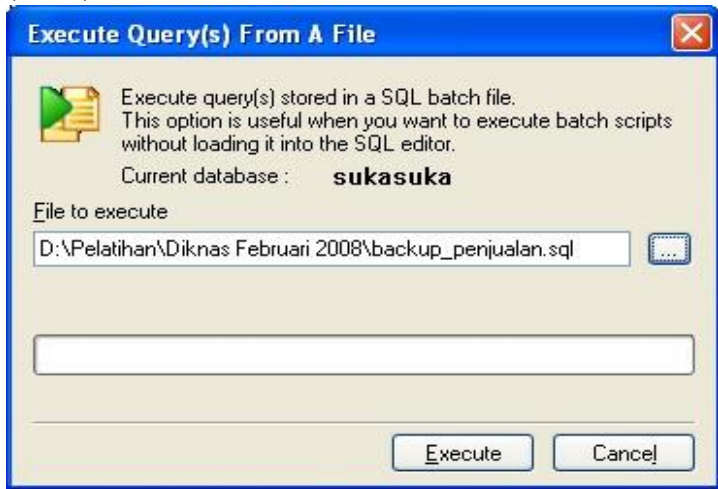
```
shell > mysql -u user -p < backup.sql
```

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*.

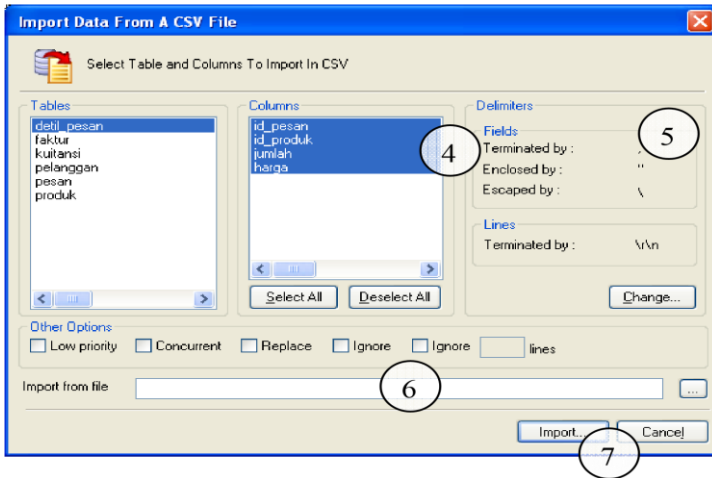
C. Import

Adakalanya kita mempunyai data mentah dalam jumlah besar. Data dapat dalam bentuk Excel, CVS, SQL, atau yang lainnya. Dan jika kita ingin mengimport 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 pemisahannya.
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. Taylor. 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 April 2017
- 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 April 2017.