PANDUAN PRAKTIKUM

BASIS DATA DASAR

Course Code / Course Credit Unit (CCU) **:** SI0044 / (3 / 1)
Disusun oleh : Mira Ziveria, S.Si, MT

KALBIS INSTITUTE
Kurikulum 2017

KATA PENGANTAR

Segala puji bagi Allah SWT karena dengan karunia-Nya Modul Praktikum SI0044 Basis Data Dasar ini dapat diselesaikan.

Modul ini disusun berdasarkan Garis-garis Besar Proses Pembelajaran (GBPP) mata kuliah SI0044 Basis Data Dasar, khususnya bagian Rencana Pembelajaran Praktikum. Modul ini bertujuan untuk memberikan panduan bagi mahasiswa Kalbis Institute yang mengambil mata kuliah SI0044 Basis Data Dasar khususnya kelas praktikum yaitu mahasiswa prodi Sistem Informasi semester 1.

Kami sangat mengharapkan masukan yang positif dari para pengajar maupun mahasiswa untuk kesempurnaan modul ini, yang dapat menjadi bahan pertimbangan sebagai bahan perbaikan di edisi berikutnya. Semoga modul praktikum ini dapat memberikan manfaat dalam memahami materi perkuliahan praktikum Basis Data Dasar di Kalbis Institute.

Jakarta, Juli 2017

Mira Ziveria

DAFTAR ISI

Pertemuan ke-1

MySQL dan SQL

1. TUJUAN

Mahasiswa mampu melakukan instalasi MySQL, memahami fungsi pada MySQL seperti mengaktifkan direktori MySQL Server, masuk dan keluar dari Server MySQL, dan bantuan pada MySQL

2. TEORI SINGKAT

MySQL

MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya sangat cepat, multi user serta menggunakan perintah dasar SQL (Structured Query Language). MySQL merupakan dua bentuk lisensi, yaitu FreeSoftware dan Shareware. MySQL yang biasa kita gunakan adalah MySQL FreeSoftware yang berada dibawah lisensi GNU/GPL (General Public License).

MySQL merupakan sebuah database server yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. MySQL pertama kali dirintis oleh seorang programmer database bernama Michael Widenius. Selain database server, MySQL juga merupakan program yang dapat mengakses suatu database MySQL yang berposisi sebagai server, yang berarti program kita berposisi sebagai Client. Jadi MySQL adalah sebuah database yang dapat digunakan sebagai Client maupun server. Database MySQL merupakan suatu perangkat lunak database yang berbentuk database relasional atau disebut Relational Database Management System (RDBMS) yang menggunakan suatu bahasa permintaan yang bernama SQL (Structured Query Language).

Kelebihan MySQL

Database MySQL memiliki beberapa kelebihan dibanding database lain, diantaranya:

- MySQL merupakan Database Management System (DBMS)
- MySQL sebagai Relation Database Management System (RDBMS) atau disebut dengan database relational
- MySQL merupakan sebuah database server yang free, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya
- MySQL merupakan sebuah database client
- MySQL mampu menerima query yang bertupuk dalam satu permintaan atau Multi-Threading
- MySQL merupakan database yang mampu menyimpan data berkapasitas sangat besar hingga berukuran GigaByte sekalipun
- MySQL diidukung oleh driver ODBC, artinya database MySQL dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti Visual Basic dan Delphi.
- MySQL adalah database menggunakan enkripsi password, jadi database ini cukup aman karena memiliki password untuk mengaksesnya
- MySQL merupakan Database Server yang multi user, artinya database ini tidak hanya digunakan oleh satu pihak orang akan tetapi dapat digunakan oleh banyak pengguna
- MySQL mendukung field yang dijadikan sebagai kunci primer dan kunci unik (Unique)
- MySQL memiliki kecepatan dalam pembuatan table maupun peng-update an table

SQL

SQL (Structured Query Language) adalah sebuah bahasa permintaan database yang terstruktur. Bahasa SQL ini dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam database maupun merelasikan antar database.
SQL dibagi menjadi tiga bentuk Query, yaitu :

DDL (Data Definition Language)

DDL adalah sebuah metode Query SQL yang berguna untuk mendefinisikan data pada sebuah database, Query yang dimiliki DDL adalah :

- CREATE : Digunakan untuk membuat Database dan Tabel
- Drop : Digunakan untuk menghapus tabel dan database
- Alter : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, baik menambah Field (Add), mengganti nama Field (Change) ataupun menamakannya kembali (Rename) dan menghapus Field (Drop).

DML (Data Manipulation Language)

DML adalah sebuah metode Query yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari Query DML ini untuk melakukan pemanipulasian database yang telah dibuat. Query yang dimiliki DML adalah:

- INSERT : Digunakan untuk memasukkan data pada Tabel Database
- UPDATE : Digunakan untuk pengubahan terhadap data yang ada pada Tabel Database
- DELETE : Digunakan untuk Penhapusan data pada Tabel Database

DCL (Data Control Language)

DCL adalah sebuah metode Query SQL yang digunakan untuk memberikan hak otorisasi mengakses database, mengalokasikan space, pendefinisian space, dan pengauditan penggunaan database. Query yang dimiliki DCL adalah :

- GRANT : Untuk mengizinkan uer mengaksest tabel dalam database.
- REVOKE : Untuk membatalkan izin hak user, yang ditetapkan oleh perintah GRANT
- COMMIT : Mentapkan penyimpanan database
- ROLLBACK : Membatalkan penyimpanan database

3. PELAKSANAAN PRAKTIKUM

Instalasi MySQL

Pada praktikum ini, kita akan melakukan proses instalasi program MySQL versi **5.5.32** dan **versi 5.6.12** basis Windows yang dirilis sekitar April 2013. Disarankan untuk mengunduh (***download***) atau menggunakan MySQL rilis terbaru yang telah dikeluarkan dan dinyatakan oleh MySQL sebagai versi yang stabil (atau ***recommended***).

Catatan:

MySQL terbaru per April 2013 adalah **MySQL 5.5.32** dan **MySQL 5.6.12**. Mengunduh (***download***) file rilis terbaru dapat dilakukan melalui laman resmi MySQL dengan alamat **http://dev.mysql.com**. Silakan Anda kunjungi situs tersebut untuk memeriksanya.

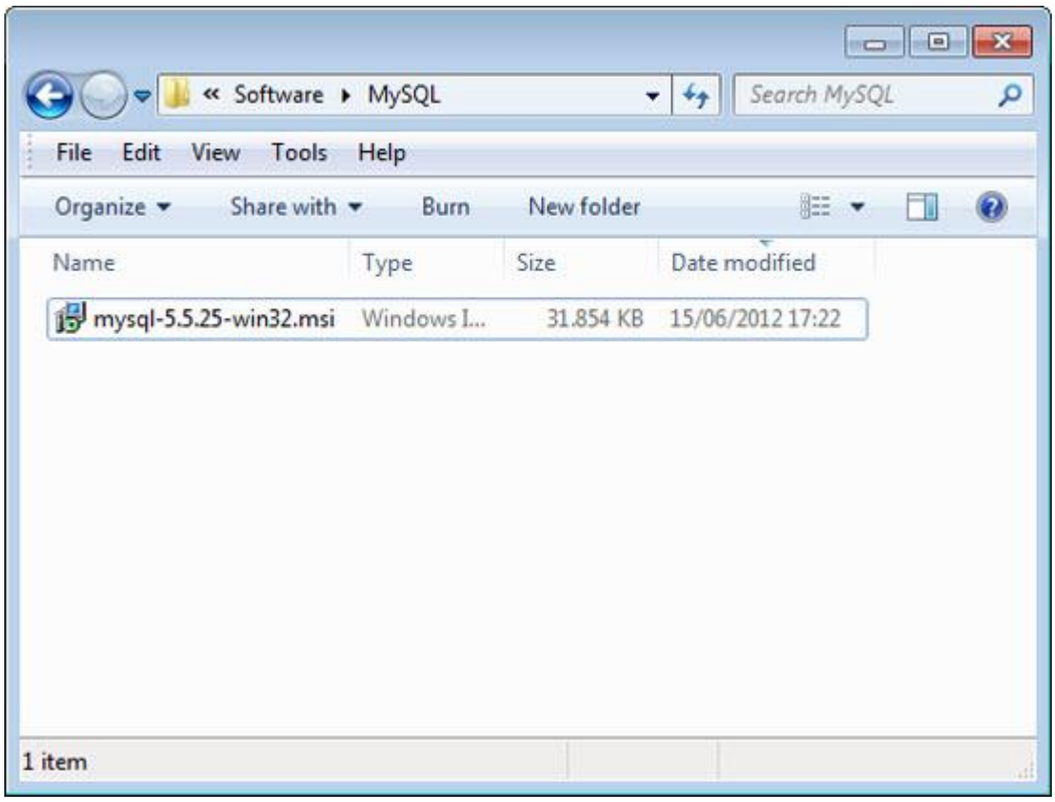
Bila Anda sudah masuk ke dalam laman MySQL, pilihlah platform sistem operasi yang Anda gunakan dengan mengaktifkan pilihan "**Select platform**" yang ada. Sistem operasi yang dapat digunakan antara lain: **Microsoft Windows, Debian Linux, Suse Linux, Oracle & Red Hat Linux, Linux - generik, Sun Solaris, Max OS X**, dan **FreeBSD**.

Pada platform MS Windows, tersedia paket instalasi MSI Installer dan ZIP Archive.

- **Windows MSI Installer (x86), 32-bit atau 64-bit**
  Merupakan paket instalasi langkah demi langkah untuk Windows yang sangat mudah untuk dilakukan. Direkomendasikan untuk menggunakan saja paket instalasi ini.
- **Windows ZIP Archive (x86), 32-bit atau 64-bit**
  Merupakan paket instalasi yang dikompresi dalam format ZIP dan harus dilakukan secara manual. Bagi mereka yang telah mahir dan memahami proses instalasinya, silakan gunakan paket instalasi ini.

Proses instalasi MySQL basis Windows cukup mudah dilakukan. Anda hanya perlu untuk mengikuti langkah-langkah di bawah ini dan kemudian dapat segera menjalankan MySQL. Mari kita mulai:

- Cari dan pilihlah file **mysql-5.5.32-win32.msi** dari folder dimana Anda menyimpan hasil unduh tersebut.



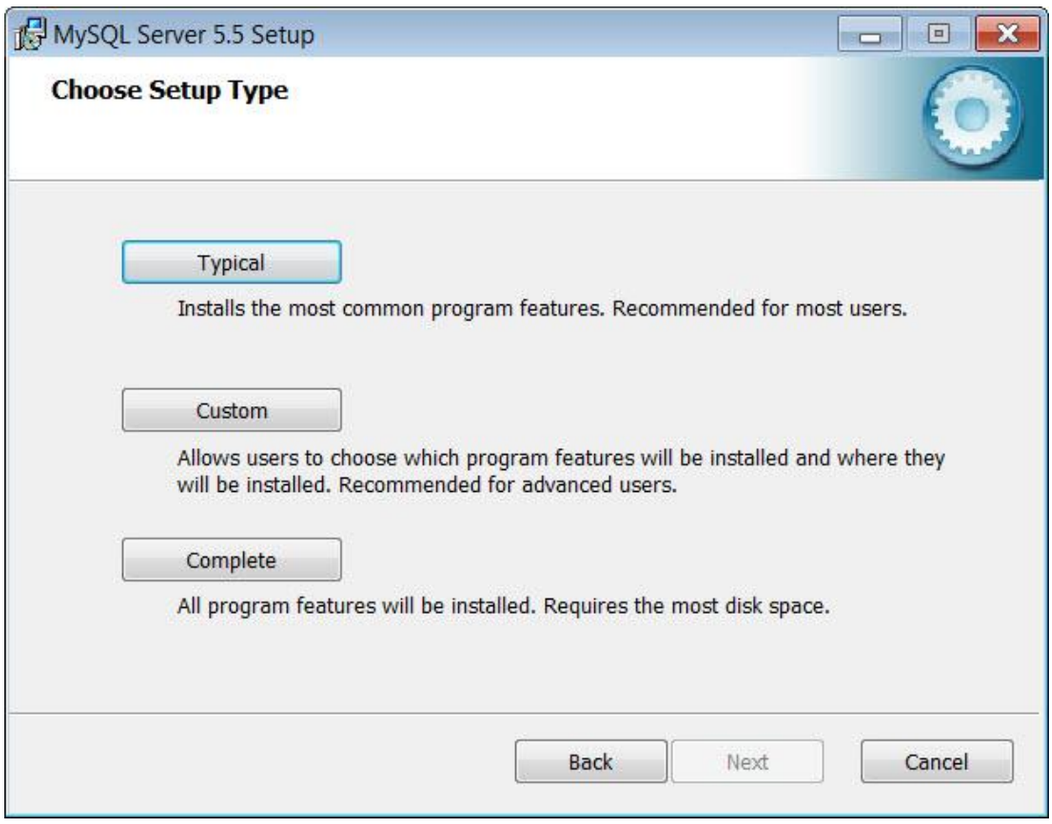
- Jalankan file tersebut dengan mengklik ganda atau menekan tombol **ENTER**. Tunggu sebentar hingga tampil layar selamat datang pada proses instalasi, dan kemudian dapat dilanjutkan dengan menekan tombol **Next**.

- Pada tampilan **End-User License Agreement**, centang pilihan ***I accept the terms in the License Agreement*** yang menyatakan persetujuan mengenai isi perjanjian lisensi aplikasi MySQL, dan lanjutkan dengan mengklik tombol **Next**.

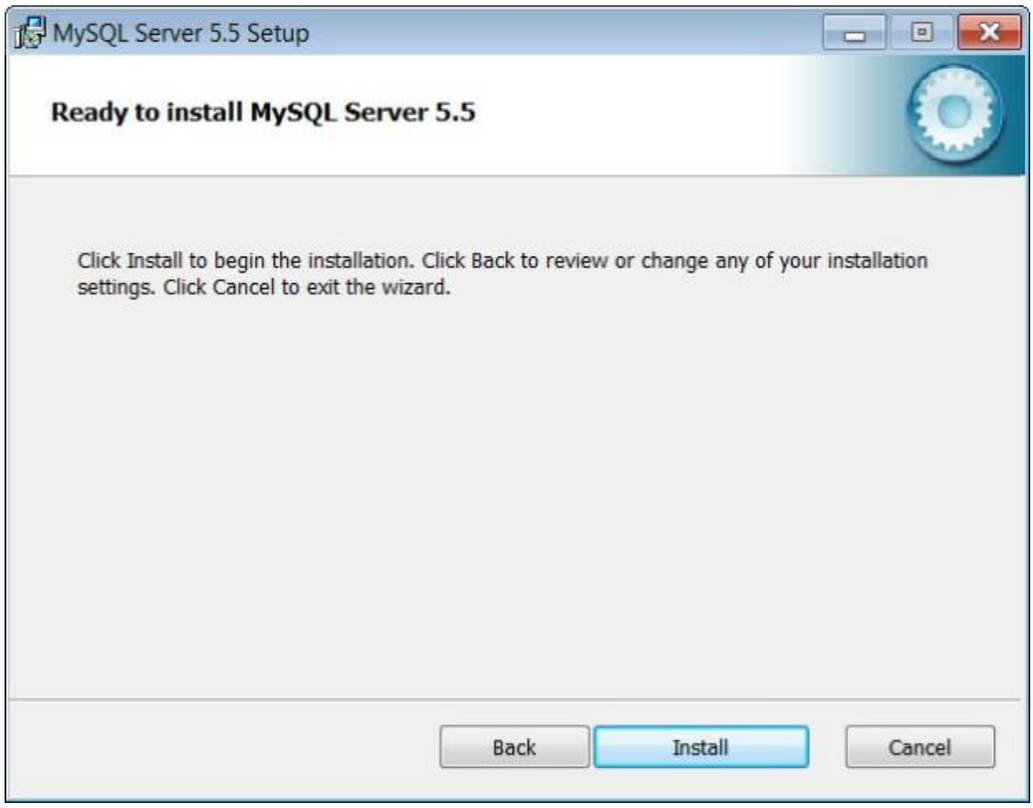
- Berikutnya berupa pilihan jenis instalasi apa yang akan digunakan. Ada 3 buah pilihan yang dapat Anda pilih yaitu, **Typical, Complete** dan **Custom**.

- Pilihan **Typical** merupakan pilihan yang paling mudah dan disarankan bagi sebagian besar pemakai.
- Pilihan **Complete** akan menginstal semua fitur yang tersedia.
- Sedangkan pilihan **Custom** hanya disarankan bagi mereka yang telah sangat memahami sistem MySQL.

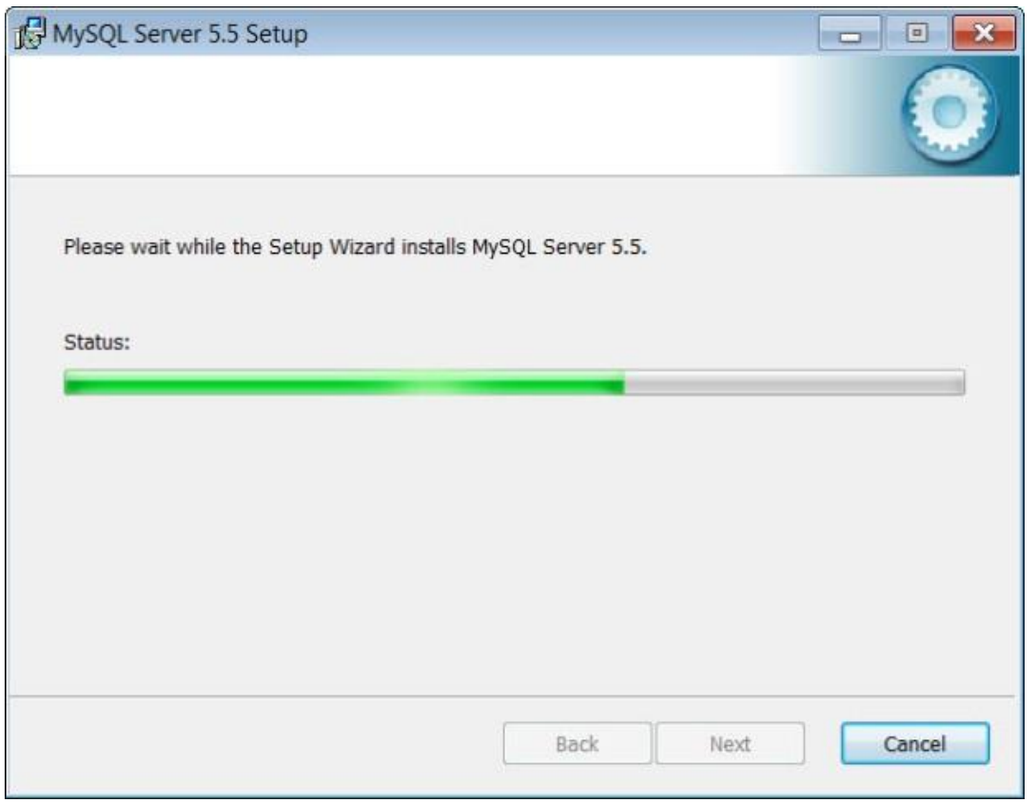
Pada contoh ini kita akan menggunakan instalasi jenis **Typical**, dan lanjutkan dengan meng-klik tombol **Typical**.



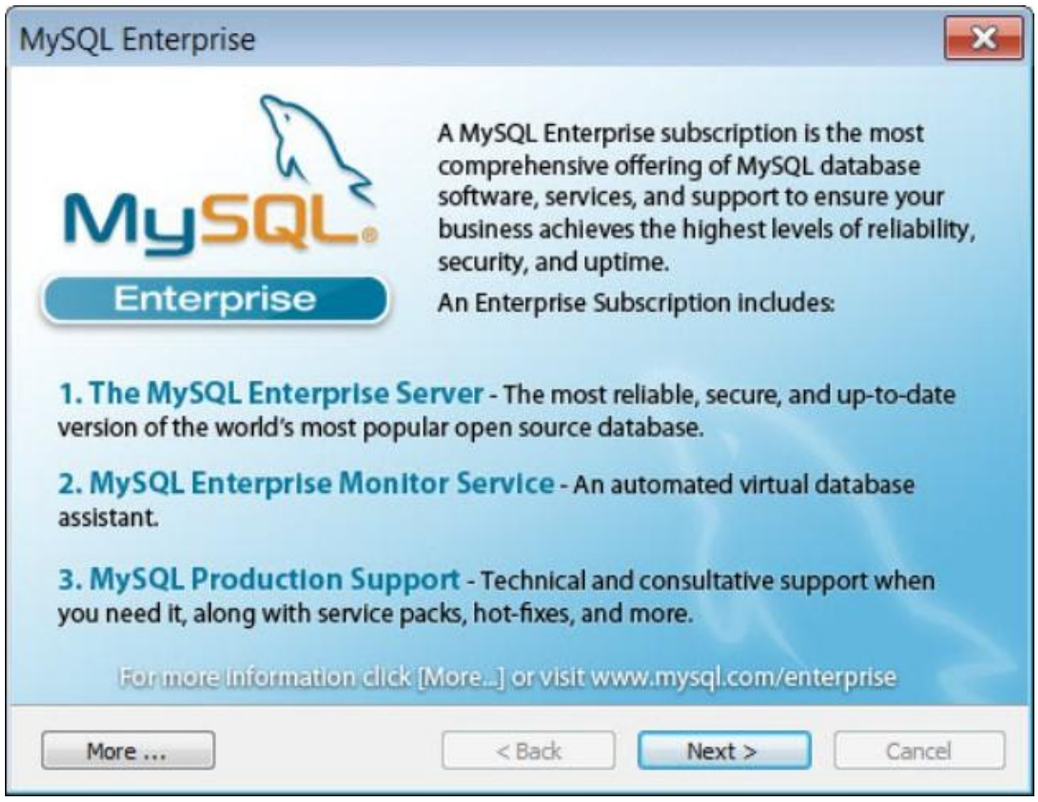
- Sekarang kondisi telah siap untuk menjalankan proses instalasi MySQL. Untuk memulai proses instalasi, klik pada tombol **Install**.



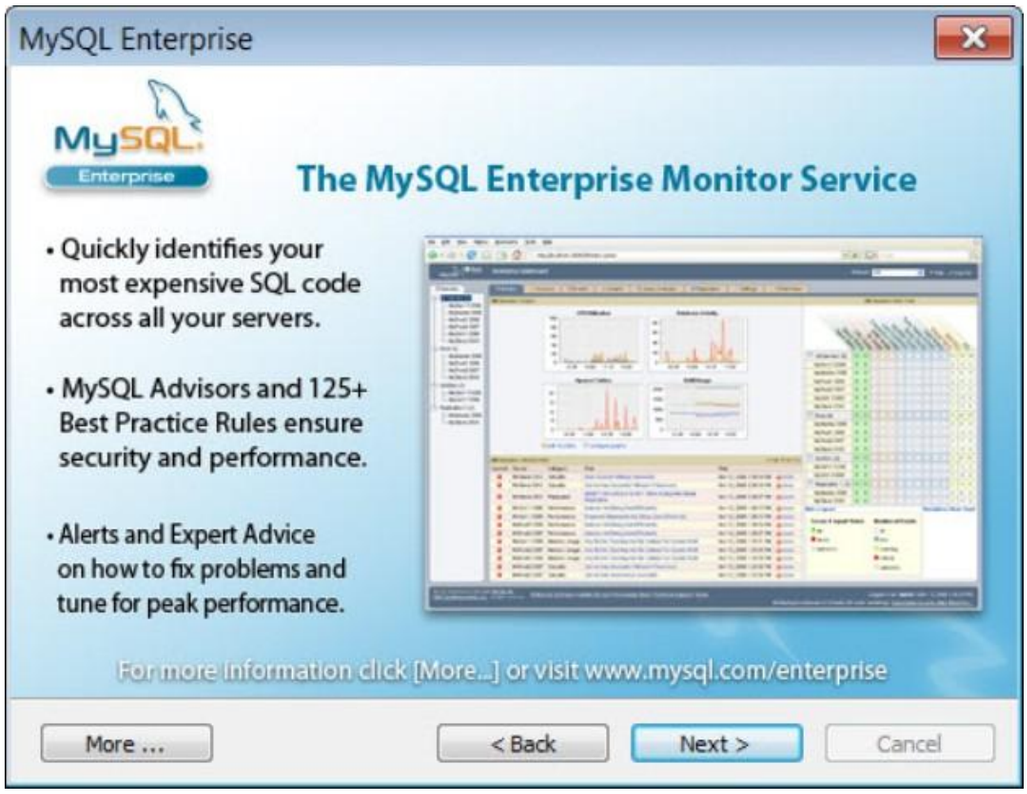
- Tunggulah sebentar hingga proses persiapan instalasi selesai.



- Berikutnya MySQL akan menampilkan informasi produk tentang MySQL Enterprise. Silakan dilanjutkan dengan menekan tombol **Next**.

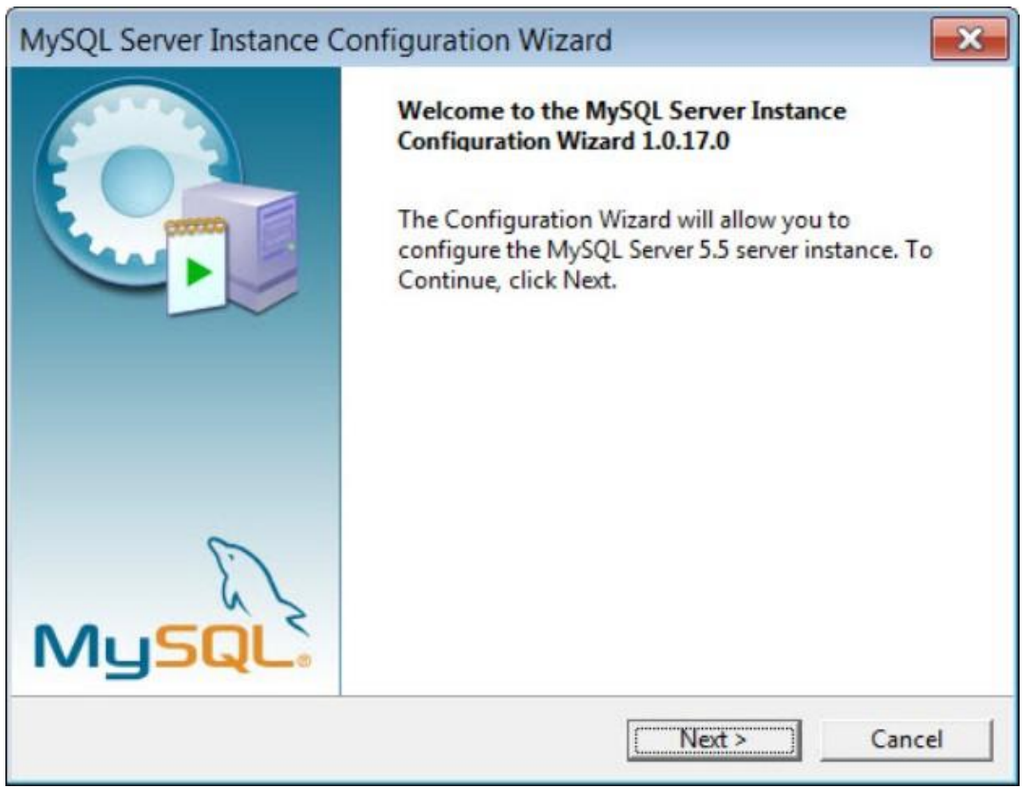


- Masih dengan tampilan informasi produk tentang MySQL Enterprise. Silakan dilanjutkan dengan menekan tombol **Next**.

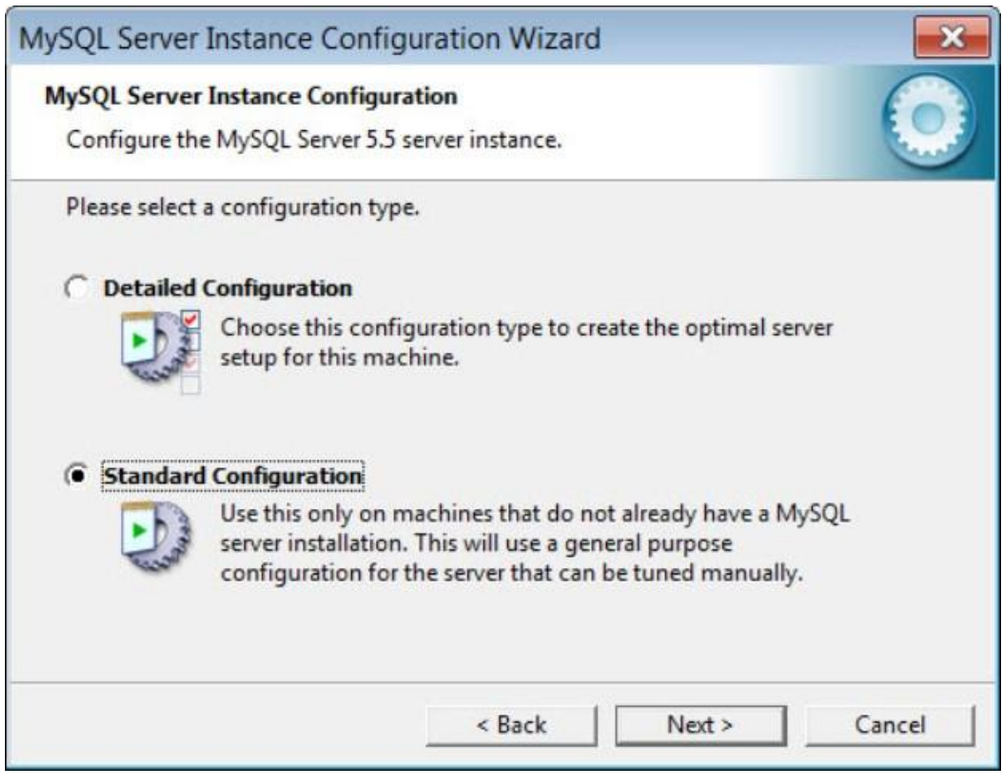


- Tunggu beberapa saat hingga proses instalasi selesai. Setelah itu dilanjutkan dengan proses konfigurasi MySQL (**Configure the MySQL Server now**) dengan menekan tombol **Finish**.

- Tampilan berikutnya selamat datang pada proses konfigurasi MySQL. Lanjutkan dengan menekan tombol **Next**.

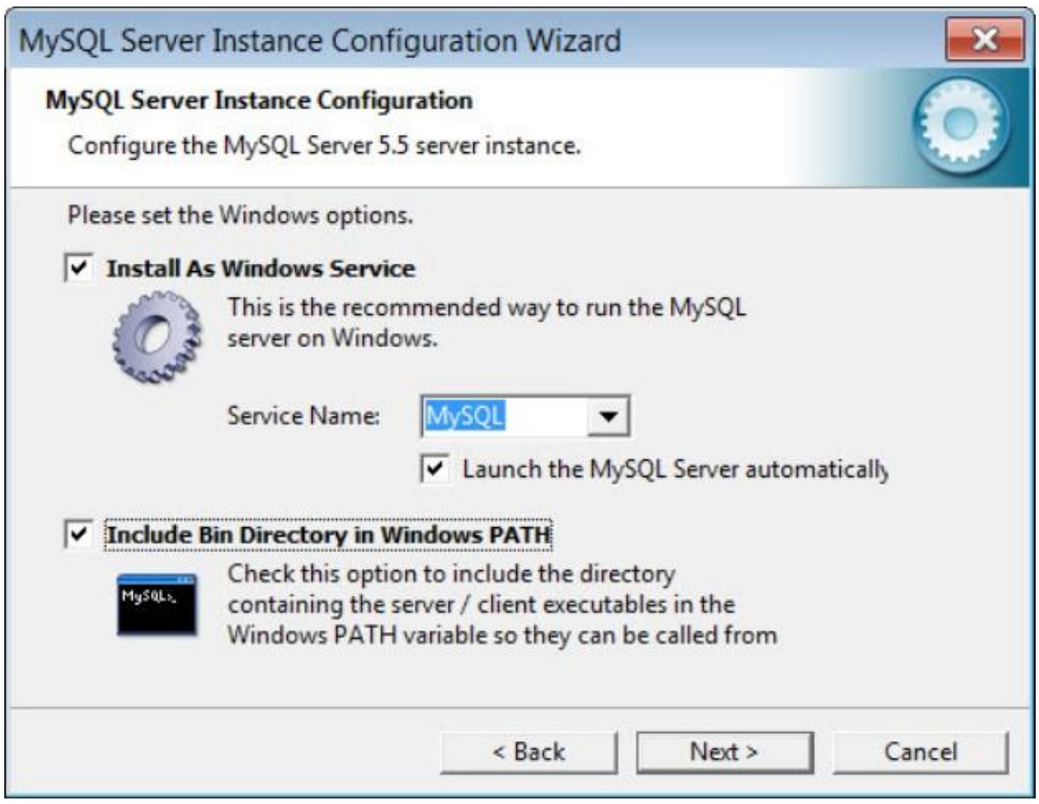


- Ada dua pilihan jenis konfigurasi, **Detailed Configuration** dan **Standard Configuration**. Pada contoh ini kita memilih ***Standard Configuration***. Klik tombol **Next** untuk melanjutkan.



- Tampilan berikutnya, disarankan untuk mengaktifkan pilihan **Install as Windows Service** dan juga **Launch the MySQL Server automatically**. Dengan pilihan ini maka setiap komputer Anda dinyalakan, secara otomatis program MySQL server akan dijalankan. Begitupun sebaiknya aktifkan pilihan "**Include Bin Directory in Windows Path"**.

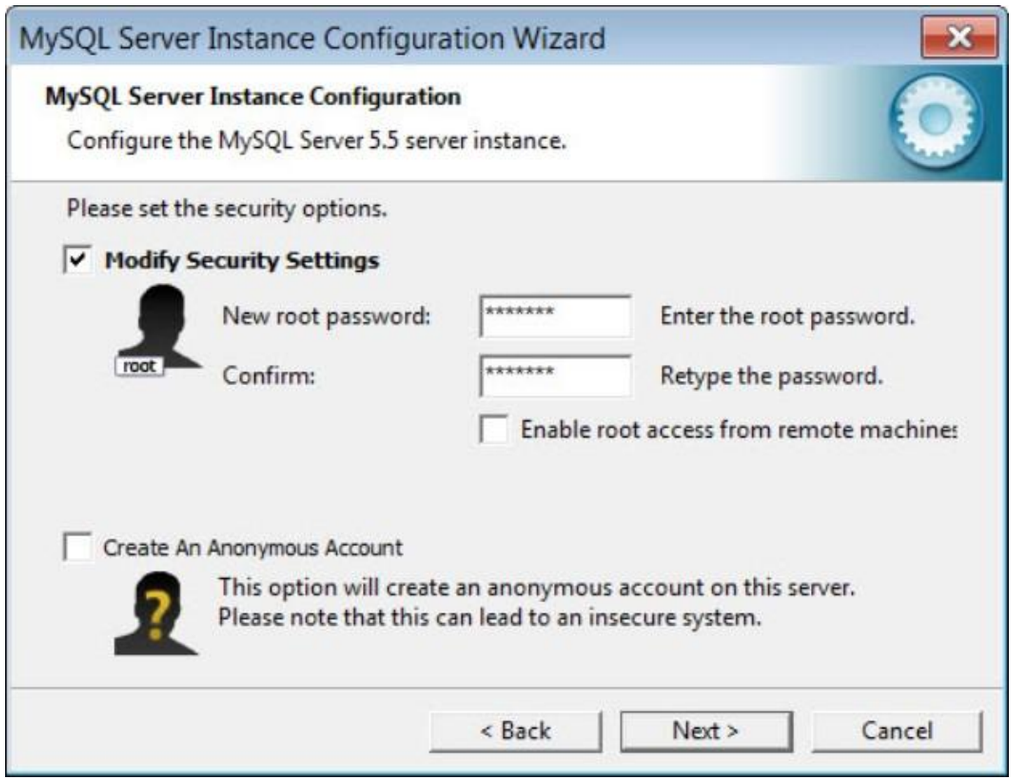
  Program-program MySQL biasanya disimpan di dalam direktori **C:Program Files/MySQL/MySQL Server 5.5/Bin**. Dengan mengaktifkan pilihan ini, maka Anda dapat menjalankan atau memanggil program MySQL langsung dari **DOS Prompt** atau **Command Prompt.**



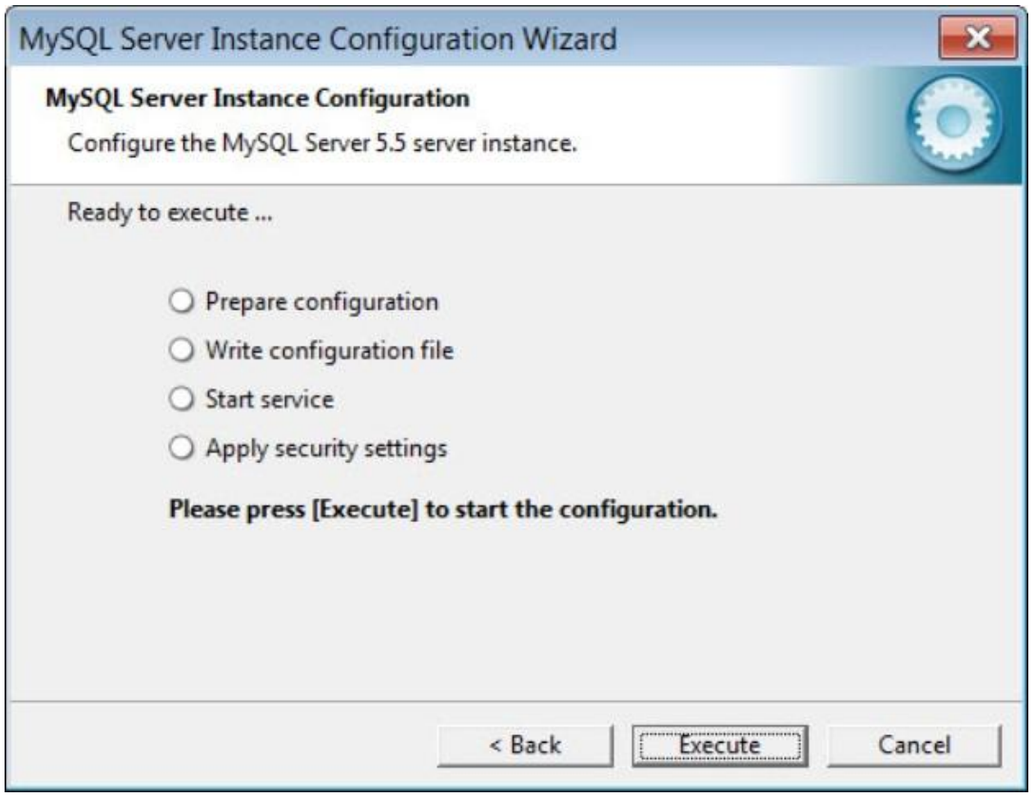
- Tampilan berikutnya, mengenai sistem keamanan server MySQL. Sebaiknya Anda memberikan password khusus sebagai **Root**, dan tidak memberikan peluang kepada orang lain untuk memasuki sistem Anda tanpa password. Maka aktifkan pilihan **Modify Security Setting** dan masukkan password **Root** Anda dengan seksama.

  Tetapi, **matikan** pilihan **Create An Anonymous Account**. Dengan demikian tidak sembarangan orang dapat masuk menggunakan MySQL server Anda.

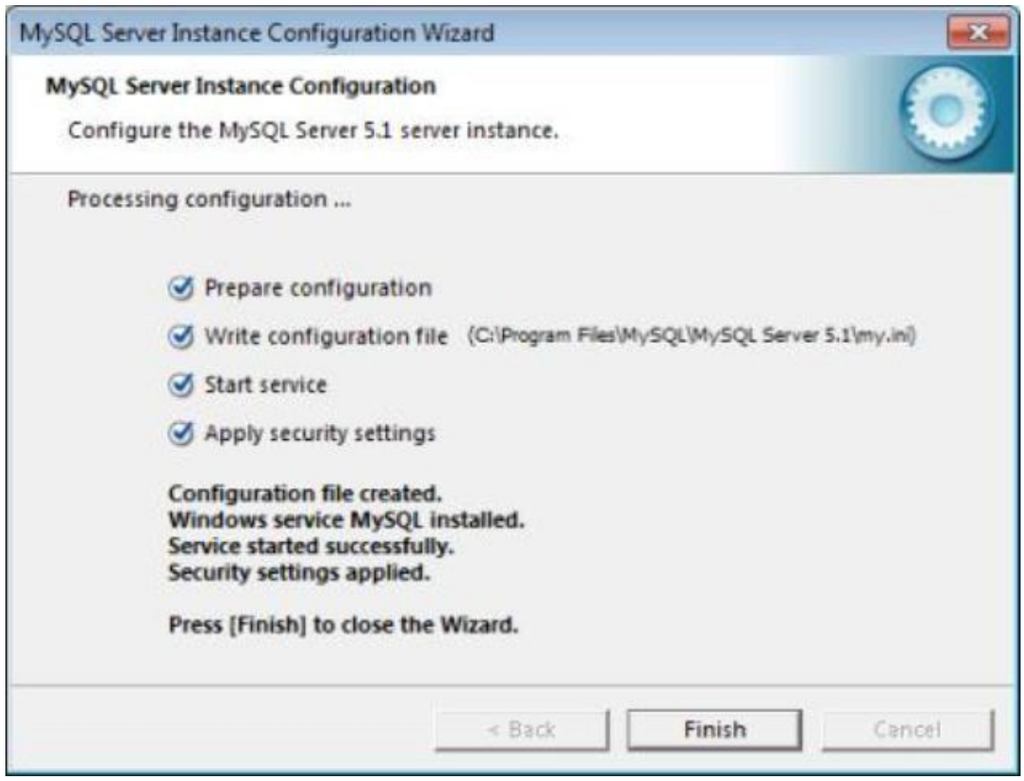
  Satu hal lagi, disarankan **mematikan** pilihan **Enable Root access from remote machines**. Hal ini untuk mencegah celah-celah yang bisa dimasuki oleh orang-orang yang tidak bertanggungjawab menyelinap ke dalam sistem kita. Lanjutkan dengan menekan tombol **Next**.



- Bila Anda telah yakin untuk melanjutkan, klik tombol **Execute**.

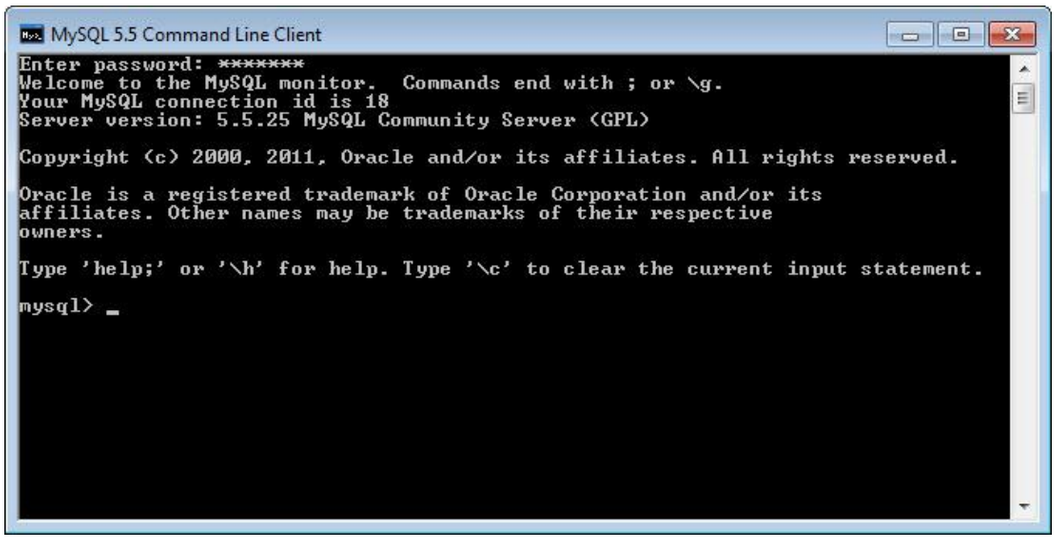


- Tunggulah sebentar hingga proses konfigurasi selesai. Kemudian klik tombol **Finish** untuk keluar dari proses konfigurasi sistem MySQL.

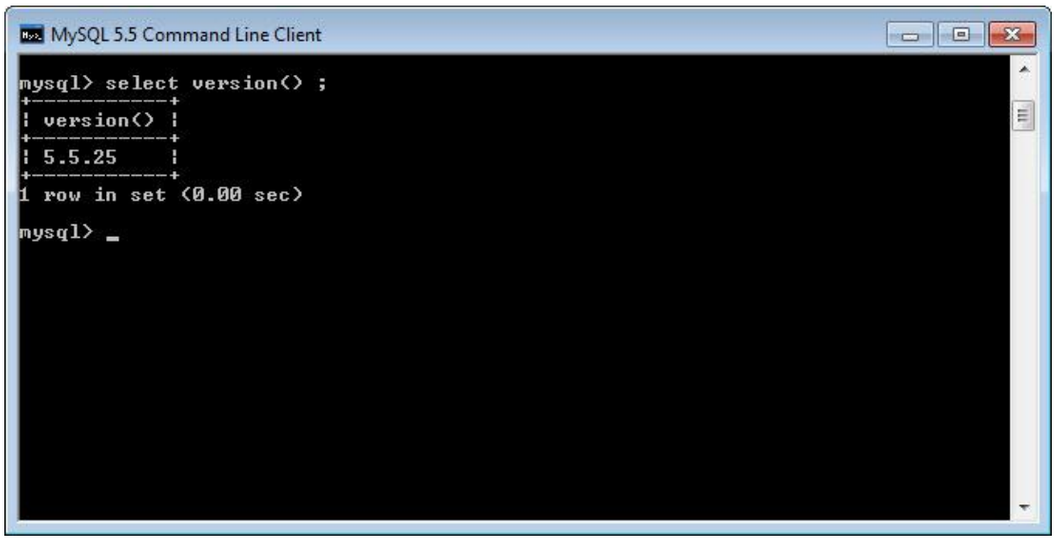


- Untuk menguji proses konfigurasi, jalankan program MySQL Server melalui menu **Start > Programs > MySQL > MySQL Server 5.1 > MySQL Command Line Client**.

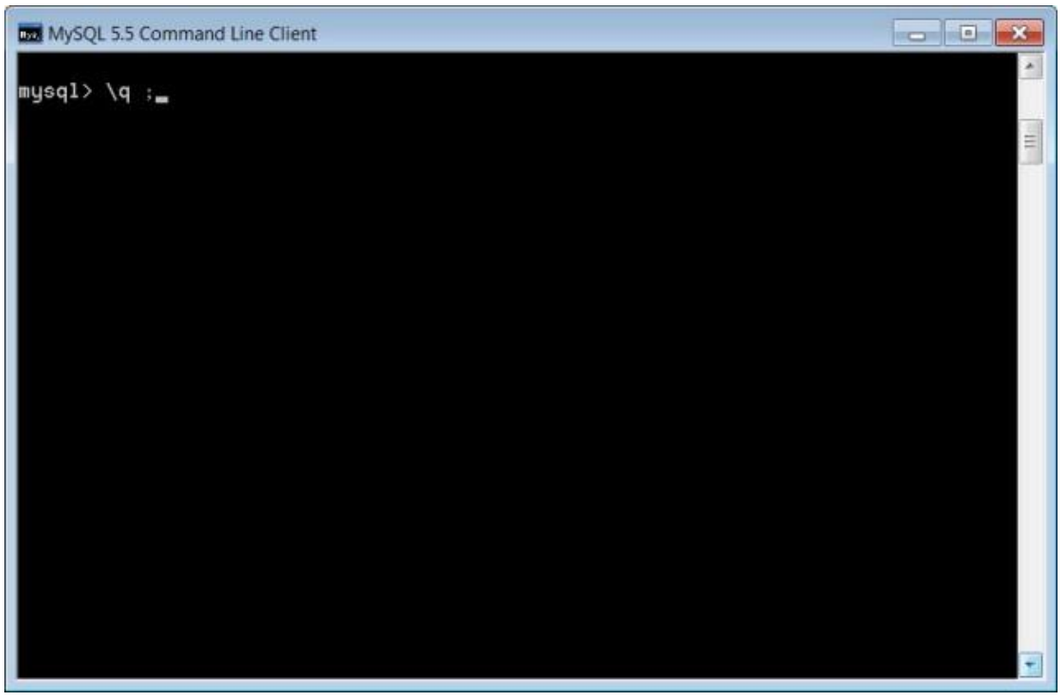
- Bila berhasil, sebuah jendela ***DOS/Command Prompt*** akan tampil. Pada baris enter ***password***: ketikkan password yang telah Anda buat pada langkah (12) di atas.



- Kita coba untuk menjalan sebuah perintah MySQL, misal **select version()** untuk melihat versi MySQL yang baru saja kita install:



- Untuk keluar dari sistem MySQL, ketikkan perintah **"q ;"** atau klik pada tombol **"X"**

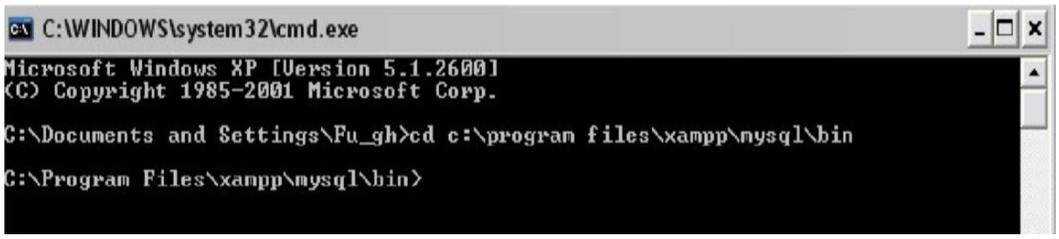


- Hingga tahapan ini proses instalasi dan konfigurasi MySQL 5.5.x telah berhasil Anda lakukan.

Mengaktifkan direktori MySQL Server

Untuk dapat menggunakan MySQL terlebih dahulu aktifkan Server MySQL dengan menghidupkan daemond MySQL. Program MySQL yang digunakan pada modul ini adalah XAMPP 1.7, maka untuk menjalankan daemond MySQL terdapat pada direktori yaitu C:\Program Files\Xampp\Mysql\Bin

Untuk masuk kedalam Server MySQL, bukalah MS-DOS Prompt Anda melalui Run kemudian ketik Command atau cmd. Maka Anda dapat masuk ke dalam direktori MySQL melalui MS-DOS Promtp seperti dibawah ini:



Masuk dan Keluar dari Server MySQL

MySQL adalah sebuah database server yang sangat aman. MySQL memiliki kemampuan memanajemen user dalam mengakses. Jadi, tidak sembarang user dapat mengakses sebuah database yang diciptakan MySQL. Maka sebelum Anda memiliki User untuk mengakses MySQL Anda juga dapat mengakses database MySQL menggunakan User Root.

Berikut adalah perintah yang digunakan untuk mengkoneksikan kedalam Server MySQL:

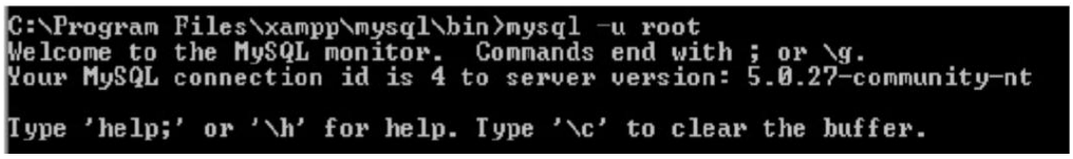
Shell > MySQL –u Root –p

Enter Password: *********

Keterangan : Tanda –u menerangkan bahwa kita akan masuk menggunakan User Name bernama Root.

Tanda –p menyatakan kita akan masuk menggunakan Password.

Berikut adalah perintah yang digunakan untuk mengkoneksikan kedalam Server MySQL melalui Root :
Shell> Mysql –u root
Untuk dapat keluar dari Server MySQL kita dapat mengetikkan:



Intruksi quit ata \q :
Mysql> quit
Bye
Mysql> \q
Bye

Bantuan pada MySQL

Database MySQL menyediakan beberapa fasilitas bantuan yang berguna untuk mendokumentasikan atau memanipulasikan server yaitu dengan cara mengetikan intruksi \h atau \?.

Mysql> \?

Semua Query harus diakhiri dengan tanda titik koma (;). Tanda ini menunjukkan bahwa query telah berakhir dan siap dieksekusi.

Help (\h) : Digunakan untuk menampilkan file bantuan pada MySQL
? (\?) : Perintah ini sama dengan perintah Help
Clear (\c) : Berguna untuk membersihkan atau menggagalkan semua perintah yang telah berjalan dalam suatu prompt
Connect (\r) : Untuk melakukan penyegaran koneksi ke dalam database yang ada pada Server Host
Ego (\G) : Berguna untuk menampilkan data secara horizontal.
Go (\g) : Memberi perintah server untuk mengeksekusi
tee (\T) : Mengatur tempat file yang akan didokumentasikan.

Contoh :
mysql> \T d : \belajar mysql.doc
Logging to file ‘d : \data.doc;’
Note (\t) : Akhir dari (\T) yang berguna untuk mendokumentasikan semua query.
Print (\p) : Mencetak semua query yang telah kita perintahkan ke layar.
Prompt (\R) : Mengubah prompt standar sesuai keinginan.
Source (\.) : Berguna untuk mengeksekusi query dari luar yang berbentuk SQL
Use (\u) : Berguna untuk memasuki database yang akan digunakan maupun mengganti database yang akan di gunakan.

4. LATIHAN

- Cobalah lakukan instalasi MySQL secara berkelompok pada laptop atau notebook masing-masing kelompok.
- Cobalah untuk mempraktekkan fungsi-fungsi yang ada pada MySQL seperti mengaktifkan direktori MySQL Server, masuk dan keluar dari server MySQL, dan bantuan pada MySQL.

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 1

Pertemuan ke-2

Administrasi MySQL

1. TUJUAN

Mahasiswa mampu melakukan administrasi MySQL, membuat user baru, memberi wewenang user, tipe data MySQL, skema basis data, pembuatan basis data dan menggunakan basis data.

2. TEORI SINGKAT

Administrasi MySQL

MySQL selaku database server yang mampu berjalan pada jaringan, tentu saja MySQL harus memiliki kemampuan khusus yang berguna untuk melakukan manajemen user atau mendukung system database yang bersifat client/server.

Membuat User baru

Untuk dapat menciptkan user baru pada database MySQL yang terdapat pada tabel user. Dapat dilakukan dengan menggunakan pernyataan SQL bernama INSERT. Sintax seperti berikut :

```
INSERT INTO user(host,user,password) VALUES('%','nama_user','password');
```

Memberikan Wewenang Untuk User

Apabila user telah dibuat terlebih dahulu dan lupa untuk memberikan hak wewenang untuk user. Kita dapat memberikan hak wewenang dengan menggunakan perintah Query UPDATE. Sintax yang digunakan seperti berikut :

```
UPDATE user
SET select_priv     ='y',
Insert_priv         ='y',
Update_priv         ='y',
Delete_priv         ='y',
Create_priv         ='y',
Drop_priv           ='y',
Alter_priv          ='y'
WHERE user          ='nama_user';
```

Tipe Data pada MySQL

Tipe data adalah suatu bentuk pemodelan data yang dideklarasikan pada saat melakukan pembuatan tabel. Tipe data ini akan mempengaruhi setiap data yang akan dimasukkan ke dalam sebuah tabel. Data yang akan dimasukkan harus sesuai dengan tipe data yang dideklarasikan.

Berbagai type data pada MySQL dapat dilihat pada tabel berikut :

Type Data	Keterangan
TINYINT	Ukuran 1 byte. Bilangan bulat terkecil, dengan jangkauan untuk bilangan bertanda: -128 sampai dengan 127 dan untuk yang tidak bertanda : 0 s/d 255. Bilangan tak bertandai dengan kata UNSIGNED
SMALLINT	Ukuran 2 Byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -32768 s/d 32767 dan untuk yang tidak bertanda : 0 s/d 65535
MEDIUMINT	Ukuran 3 byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -8388608 s/d 8388607 dan untuk yang tidak bertanda : 0 s/d 16777215
INT	Ukuran 4 byte. Bilangan bulat dengan jangkauan untuk bilangan bertanda : -2147483648 s/d 2147483647 dan untuk yang tidak bertanda : 0 s/d 4294967295
INTEGER	Ukuran 4 byte. Sinonim dari int
BIGINT	Ukuran 8 byte. Bilangan bulat terbesar dengan jangkauan untuk bilangan bertanda : -9223372036854775808 s/d 9223372036854775807 dan untuk yang tidak bertanda : 0 s/d 1844674473709551615
FLOAT	Ukuran 4 byte. Bilangan pecahan
DOUBLE	Ukuran 8 byte. Bilangan pecahan
DOUBLEPRECISION	Ukuran 8 byte. Bilangan pecahan
REAL	Ukuran 8 byte. Sinonim dari DOUBLE
DECIMAL (M,D)	Ukuran M byte. Bilangan pecahan, misalnya DECIMAL(5,2 dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99
NUMERIC (M,D)	Ukuran M byte. Sinonim dari DECIMAL, misalnya NUMERIC(5,2) dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99

Type Data untuk Bilangan (Number)

Type Data	Keterangan
DATETIME	Ukuran 8 byte. Kombinasi tanggal dan jam, dengan jangkauan dari '1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'
DATE	Ukuran 3 Byte. Tanggal dengan jangkauan dari '1000-01-01' s/d '9999-12-31'
TIMESTAMP	Ukuran 4 byte. Kombinasi tanggal dan jam, dengan jangkauan dari '1970-01-01 00:00:00' s/d '2037'
TIME	Ukuran 3 byte. Waktu dengan jangkauan dari '839:59:59' s/d '838:59:59'
YEAR	Ukuran 1 byte. Data tahun antara 1901 s/d 2155

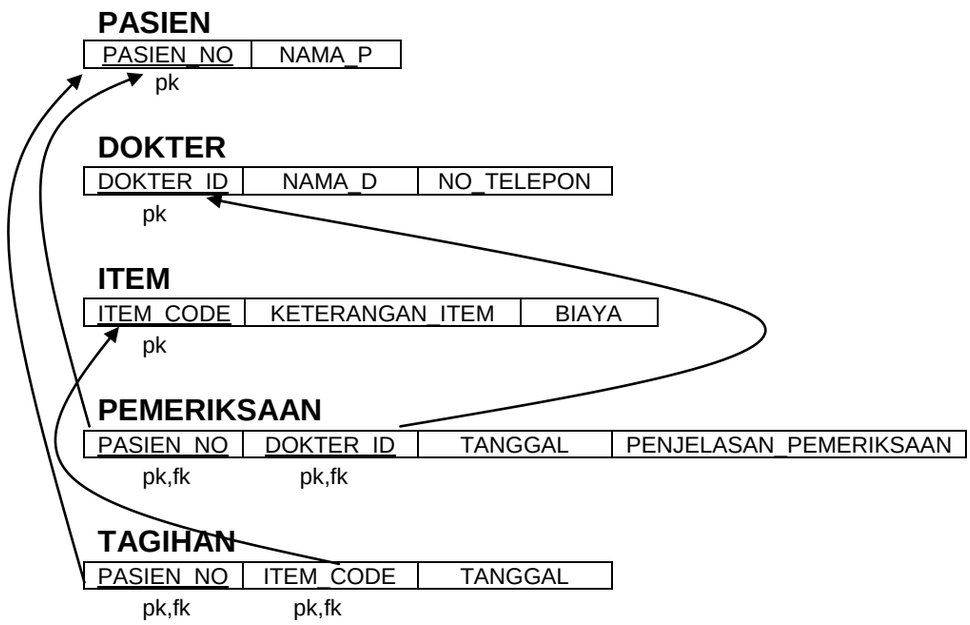
Type Data untuk Tanggal dan Jam

Type Data	Keterangan
CHAR	Mampu menangani data hingga 255 karakter. Tipe data CHAR mengharuskan untuk memasukkan data yang telah ditentukan oleh kita.
VARCHAR	Mampu menangani data hingga 255 karakter. Tipe data VARCHAR tidak mengharuskan untuk memasukkan data yang telah ditentukan oleh kita.
TINYBLOB, TINYTEXT	Ukuran 255 byte. Mampu menangani data sampai 2^8-1 data.
BLOB, TEXT	Ukuran 65535 byte. Type string yang mampu menangani data hingga 2^16-1 (16M-1) data.
MEDIUMBLOB, MEDIUMTEXT	Ukuran 16777215 byte. Mampu menyimpan data hingga 2^24-1 (16M-1) data.
LONGBLOB, LONGTEXT	Ukuran 4294967295 byte. Mampu menyimpan data hingga berukuran GIGA BYTE. Tipe data ini memiliki batas penyimpanan hingga 2^32-1 (4G-1) data.
ENUM('nilai1','nilai2',...,'nilaiN')	Ukuran 1 atau 2 byte. Tergantung jumlah nilai enumerasinya (maksimum 65535 nilai)
SET('nilai1','nilai2',...,'nilaiN')	1,2,3,4 atau 8 byte, tergantung jumlah anggota himpunan (maksimum 64 anggota)

Type Data untuk Karakter dan Lain-lain

Skema Basis data

Basis Data (database) adalah sebuah media utama yang harus dibuat dalam membangun sebuah basis data agar nantinya dapat kita letakkan beberapa tabel dengan field-fieldnya. Berikut adalah contoh Skema Basis Data Rumah Sakit:



Pembuatan Basis data

Perintah yang digunakan untuk menciptakan database pada MySQL dengan Syntax berikut:

CREATE DATABASE nama_database;

Pada contoh diatas, query OK menyatakan bahwa pembuatan database dengan nama pendaftaran berhasil dibuat, untuk melihat database yang ada pada MySQL dapat menggunakan Sintax berikut;
SHOW DATABASES;

Untuk menghapus Database yang telah dibuat dapat menggunakan query SQL berikut :

DROP DATABASE nama_database;
Drop berarti menghapus. Query SQL ini berfungsi untuk menghapus sebuah database, seperti contoh berikut :

mysql> drop database pendaftaran;
Query OK, 0 rows affected (0.02 sec)

Menggunakan Basis data
Untuk menggunakan database yang telah dibuat dapat menggunakan query SQL berikut:

USE DATABASE nama_database;
Use berarti menggunakan database yang telah dibuat, seperti contoh berikut :

3. PELAKSANAAN PRAKTIKUM

Membuat User Baru
Membuat user baru dengan host = localhost, nama user = haris dan password 'SI060017'

mysql> INSERT INTO user(host, user, password)
VALUES('localhost','haris',MD5('SI060017'));
Query OK, 1 row affected, 4 warnings (0.00 sec)

Setelah Anda memberikan perintah diatas, berikan perintah :
FLUSH PRIVILEGES;

Contoh :
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

Memberikan Wewenang Untuk User
Memberikan hak akses user bernama haris dengan privilages insert, update, delete, create, drop, dan alter dengan sintaks:

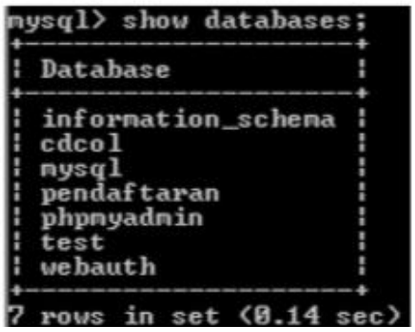
```
UPDATE user
SET select_priv     =’y’,
Insert_priv         =’y’,
Update_priv         =’y’,
Delete_priv         =’y’,
Create_priv         =’y’,
Drop_priv           =’y’,
Alter_priv          =’y’
WHERE user          =’haris’;
```

Pembuatan Basis data
Membuat database pendaftaran dengan Syntax berikut:

mysql> create database pendaftaran;
Query OK, 1 row affected (0.11 sec)

Pada contoh diatas, query OK menyatakan bahwa pembuatan database dengan nama pendaftaran berhasil dibuat, untuk melihat database yang ada pada MySQL dapat menggunakan Sintax berikut:

SHOW DATABASES;
Contoh :



Untuk menghapus database pendaftaran yang telah dibuat dapat menggunakan query SQL berikut :

mysql> drop database pendaftaran;
Query OK, 0 rows affected (0.02 sec)

Menggunakan Basis data
Untuk menggunakan Database pendaftaran yang telah dibuat dapat menggunakan query SQL berikut :

mysql> use database pendaftaran;
Database changed

4. LATIHAN

- Dengan perintah INSERT buatlah user baru dengan host = localhost, nama user = Nama_Anda dan password 'Password_Anda'
- Berikan perintah FLUSH PRIVILAGES untuk user yang telah dibuat
- Berikan hak akses user bernama Nama_Anda dengan privilages insert, update, delete, create, drop, dan alter
- Buatlah basis data Rumah Sakit dengan perintah CREATE DATABASE
- Lihatlah database yang telah dibuat dengan perintah SHOW DATABASES
- Aktifkan basis data Rumah Sakit dengan perintah USE
- Hapus database Rumah Sakit

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 2

Pertemuan ke-3

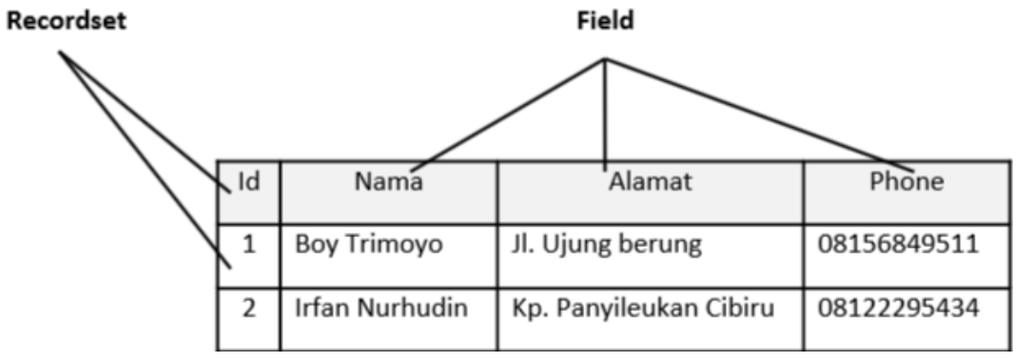
Skema MySQL

1. TUJUAN

Mahasiswa mampu memahami skema dalam SQL dan mampu melakukan pembutan database dan tabel sederhana, melihat struktur table, menghapus tabel dan database, membuat indeks, mendefinisikan nilai null/not null, membuat kolom unik, dan membuat nilai default.

2. TEORI SINGKAT

Tabel adalah obyek utama yang harus ada pada sebuah basis data karena di dalamnya semua data akan disimpan. Tabel terletak pada sebuah database, sehingga pembuatan tabel dilakukan setelah sebuah database telah dibuat. Dalam tabel terdapat baris dan kolom. Baris diistilahkan dengan recordset dan kolom dengan field.



Id	Nama	Alamat	Phone
1	Boy Trimoyo	Jl. Ujung berung	08156849511
2	Irfan Nurhudin	Kp. Panyileukan Cibiru	08122295434

Untuk membuat sebuah tabel atau lebih, database harus diaktifkan dulu karena tabel akan dimasukkan ke dalam database yang akan diaktifkan.

Sintax untuk mengaktifkan database adalah :
USE nama_database;

Setelah masuk ke dalam database Anda dapat membuat sebuah tabel atau lebih. Untuk membuat tabel dapat menggunakan sintax dibawah ini :
CREATE TABLE nama_tabel (field-1 type(length), field-2 type(length), field-3 type(length), …… ….(….));

Untuk melihat tabel yang ada pada database dapat menggunakan SIntax berikut ;
SHOW TABLES;

Melihat Struktur Tabel
Setelah tabel dibuat, Anda dapat melihat tipe data dan panjang recordset dengan cara menampilkan struktur tabel.

Perintah yang digunakan untuk menampilkan struktur tabel adalah :
DESC nama_tabel;
Atau
DESCRIBE nama_tabel;

Menghapus Tabel
Untuk menghapus tabel yang telah dibuat dapat menggunakan query SQL berikut :
DROP TABLE nama_tabel;

Membuat Indeks
Untuk membuat Indeks menggunakan query SQL berikut :
CREATE INDEX nama_idx1,...,nama_idxn on nama_tabel (kolom_indeks1,…,kolom_idxn);

Mendefinisikan Nilai Null/Not Null
Nilai null/not null dapat didefinisikan pada saat pembuatan database, yaitu dengan menambahkan NULL atau NOT NULL dibelakang nama kolom.

Membuat Kolom Unique
Kolom Unique adalah sebuah bentuk kolom yang tidak mengizinkan adanya data kembar. Apabila pada proses input terdapat data kembar maka proses tersebut akan digagalkan atau ditolak oleh database.

Syntax untuk menciptakan Kolom unik (Unique) adalah :
CREATE TABLE nama_tabel (field-1 type(length), field-2 type(length), …… ….(….),UNIQUE (field-1,field- 2));

Membuat Nilai Default
Nilai default dapat didefinisikan pada saat pembuatan database, yaitu dengan menambahkan DEFAULT dibelakang nama kolom.

3. PELAKSANAAN PRAKTIKUM

Membuat database dan tabel sederhana

- Buatlah database PENDAFTARAN dengan perintah sbb:
  mysql> create dababase pendaftaran;
  Database created

- Aktifkan database PENDAFTARAN dengan perintah sbb:
  mysql> use pendaftaran;
  Database changed

- Buatlah tabel data_diri dengan struktur sebagai berikut:

FIELD	TIPE DATA
No	int(3)
Nama	varchar(35)
Alamat	varchar(60)
Email	varchar(40)
no_telepon	varchar(15)
Sex	Char(1)

Perintah:
```
Mysql-> create table data_diri (
    -> no int(3),
    -> nama varchar(35),
    -> alamat varchar(60),
```

```
    -> email varchar(40),
    -> no_telepon varchar(15),
    -> sex char(1));
Query OK, 0 rows affected (0.08 sec)
```

Buatlah tabel data_diri dengan struktur sebagai berikut:

- Melihat tabel yang ada pada database dapat menggunakan SIntax berikut ;

SHOW TABLES;

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

Melihat Struktur Tabel

Melihat atau menampilkan struktur tabel data_diri adalah :
DESC data_dir;
Atau
DESCRIBE data_diri;

```
mysql> desc data_diri;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| no         | int(3)      | YES  |     | NULL    |       |
| nama       | varchar(35) | YES  |     | NULL    |       |
| alamat     | varchar(60) | YES  |     | NULL    |       |
| email      | varchar(40) | YES  |     | NULL    |       |
| no_telepon | varchar(15) | YES  |     | NULL    |       |
| sex        | char(1)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.11 sec)
```

Menghapus Tabel

Menghapus tabel data_diri dengan perintah:
mysql> drop table data_diri;
Query OK, 0 rows affected (0.03 sec)

Membuat Indeks

Membuat indeks untuk kolom nana pada tabel data_diri:
mysql> create index idx_nama on data_diri (nama);

Mendefinisikan Nilai Null/Not Null

Mendefinisikan nilai not null untuk kolom no dan nama pada tabel data_diri:

```
mysql> create table data_diri (
    -> no int(3)                NOT NULL,
    -> nama varchar(35)         NOT NULL,
    -> alamat varchar(60),
    -> email varchar(40),
    -> no_telepon varchar(15),
    -> sex char(1));
```

Membuat Kolom Unique

Membuat kolom unik untuk kd_panggilan dan panggilan pada tabel probadi:

```
mysql> Create table pribadi (
    -> kd_pribadi CHAR(3),
    -> panggilan char(4),
    -> nama varchar(35),
    -> email varchar(50),
    -> sex char(1),
    -> UNIQUE (kd_pribadi,panggilan));
Query OK, 0 rows affected (0.08 sec)
```

Membuat Nilai Default

Membuat nilai default “00000” untuk no_telepon pada tabel data_diri:

```
mysql> create table data_diri (
        -> no int(3),
        -> nama varchar(35),
        -> alamat varchar(60),
        -> email varchar(40),
        -> no_telepon varchar(15)       DEFAULT ‘00000’,
        -> sex char(1));
```

4. LATIHAN

1. Buatlah basis data RumahSakit
2. Buatlah tabel pasien, dokter dan pemeriksaan dengan skema sbb:

PASIEN

FIELD	TIPE DATA
PASIEN_NO	Char(4)
NAMA_P	Varchar(20)

DOKTER

FIELD	TIPE DATA
DOKTER_ID	Char(4)
NAMA_D	Varchar(20)
NO_TELEPON	Varchar(12)

PEMERIKSAAN

FIELD	TIPE DATA
PASIEN_NO	Char(4)
DOKTER_ID	Char(4)
TANGGAL	Date
NO_TELEPON	Varchar(12)
PENJELASAN_PEMERIKSAAN	Varchar(30)

3. Buatlah indeks untuk kolom nama pasien dan nama dokter
4. Buatlah batasan NULL untuk kolom pasien_no dan nama_p pada tabel PASIEN
5. Buatlah primary key untuk tabel pasien, dokter, dan pemeriksaan
6. Buatlah nilai default “Kontrol Rutin” untuk PENJELASAN_PEMERIKSAAN yang ada pada tabel PEMERIKSAAN

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 3

Pertemuan ke-4

Modifikasi Skema SQL

1. TUJUAN

Mahasiswa mampu melakukan modifikasi tabel, merubah ukuran/tipe field, menambah, mengubah dan menghapus kolom pada table, mengganti nama table, membuat primary dan foreign key, dan klausa CHECK

2. TEORI

Modifikasi Tabel
Perubahan tabel yang telah dibuat akan selalu dilakukan mengingat perkembangan database, termasuk diantaranya menambahkan beberapa field pada tabel, mengganti nama field maupun tabel.

Mengganti nama tabel
Query SQL untuk merubah nama tabel dengan menggunakan RENAME.
Sintax seperti berikut :
RENAME TABLE tabel_lama TO tabel_baru;

Mengganti nama kolom
Query SQL untuk mengganti nama kolom dengan menggunakan CHANGE.
Sintax seperti berikut:
ALTER table nama_tabel change nama_field_lama nama_field_baru tipe_data;

Merubah ukuran/tipe field
Query SQL untuk untuk merubah ukuran/tipe data menggunakan perintah MODIFY. Perubahan yang terjadi hanya pada tipe data yang digunakan oleh field/kolomtertentu. Tipe data baru langsung disebutkan dibelakang nama field/kolom, tanpa harus menyebutkan tipe data lama.

Sintax seperti berikut:
ALTER table nama_tabel modify nama_field tipe_data_baru;

Menambah Field pada Tabel
Menambah kolom dapat diartikan sebagai langkah untuk menyisipkan field baru pada sebuah tabel. Untuk melakukan penambahan Field maka ALTER spesifikasi yang digunakan adalah ADD.

Sintax yang digunakan adalah :
ALTER TABLE nama_tabel ADD nama_field Type_data(length);

Menghapus Field pada Tabel
Pada pembuatan database pasti terdapat kesalahan seperti pada field tabel yang berlebihan dan lain-lain. Untuk melakukan Penghapusan Field maka ALTER spesifikasi yang digunakan adalah DROP.

Sintax yang digunakan adalah :
ALTER TABLE nama_tabel DROP nama_field;

Membuat Primary Key

Dalam membuat sebuah database, kita akan menemukan sebuah record yang data nya tidak boleh sama dengan record yang lain. Agar data tidak kembar maka harus membuat sebuah kolom yang di deklarasikan sebagai kunci primer (primary key), Primary key hanya diperboleh kan dibuat satu kunci.

Syntax untuk menciptakan kunci primer (primary key) adalah :

```
CREATE TABLE nama_tabel (
                field-1 type(length) PRIMARY KEY,
                field-2 type(length), …… ….(….));
```

ATAU

```
CREATE TABLE nama_tabel (
                field-1 type(length),
                field-2 type(length),
                ……
                ……
                PRIMARY KEY (….));
```

Membuat Foreign Key

Syntax untuk menciptakan foreign key adalah:

```
CREATE TABLE nama_tabel (
        field-1 type(length),
        field-2 type(length),
        ……,
        ……,
        PRIMARY KEY (kolom-PK),
        FOREIGN KEY (kolom FK) REFERENCES Nama_Tabel (kolom_rujukan));
```

Kalusa CHECK

Constaint CHECK digunakan untuk membatasi rentang nilai yang dapat ditempatkan dalam suatu kolom. Jika mendefinisikan suatu constraint CHECK pada suatu tabel, maka dapat membatasi nilai dalam kolom tertentu berdasarkan nilai-nilai dalam kolom lain pada baris.

Sintax:

```
Create tabel nama_tabel
        (nama_kolom1 tipe_data1 (ukuran) CHECK(syarat nama_kolom1),
        nama_kolom2 tipe_data2 (ukuran),
        .....,
        .....);
```

3. PELAKSANAAN PRAKTIKUM

Lakukanlah praktikum dengan menggunakan tabel data_diri berikut:

FIELD	TIPE DATA
No	int(3)
Nama	varchar(35)
Alamat	varchar(60)
Email	varchar(40)
no_telepon	varchar(15)
Sex	Char(1)

Mengganti nama tabel
Mengganti nama tabel data_diri menjadi data_pribadi dengan sintaks:

```
mysql> rename table data_diri to data_pribadi;
Query OK, 0 rows affected (0.02 sec)
```

Mengganti nama kolom
Mengganti nama kolom no menjadi nomor pada tabel data_diri dengan sintaks:
```
mysql>alter table data_diri change no nomor char(10);
```

Merubah ukuran/tipe field
Merubah ukuran/tipe data untuk kolom nama menjadi varchar(50) pada tabel data_diri dengan sintaks:
```
mysql>alter table data_diri modify nama varchar(50);
```

Menambah Field pada Tabel
Menambah field/kolom gol_darah pada tabel data_diri dengan sintaks:
```
mysql> alter table data_diri add gol_darah char(1);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Menghapus Field pada Tabel
Menghapus kolom gol_darah pada tabel data_diri dengan sintaks:
```
mysql> alter table data_diri drop gol_darah;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Membuat Primary Key
Membuat kolom primary key pada tabel data_diri dengan sintaks:

```
mysql> create table data_diri (
    -> no int(3) PRIMARY KEY,
    -> nama varchar(35),
    -> alamat varchar(60),
    -> email varchar(40),
    -> no_telepon varchar(15),
    -> sex char(1));
```

ATAU
```
mysql> create table data_diri (
    -> no int(3),
    -> nama varchar(35),
    -> alamat varchar(60),
    -> email varchar(40),
    -> no_telepon varchar(15),
    -> sex char(1)
   -> PRIMARY KEY (no));
```

Membuat Foreign Key
Membuat kolom no_divisi sebagai kolom foreign key pada tabel data_diri dengan merujuk kepada tabel divisi dengan menggunakan sintaks:

```
mysql> create table data_diri (
    -> no int(3),
    -> nama varchar(35),
    -> alamat varchar(60),
    -> email varchar(40),
    -> no_telepon varchar(15),
    -> sex char(1),
    -> no_divisi int(4),
   -> PRIMARY KEY (no),
    -> FOREIGN KEY (no_divisi) REFERENCES divisi (no_divisi));
```

Kalusa CHECK

Membuat constaint CHECK pada tabel data_diri untuk kolom no nilainya lebih besar dari nol dengan sintaks:

```
mysql> create table data_diri (
    -> no int(3) PRIMARY KEY CHECK (no>0),
    -> nama varchar(35),
    -> alamat varchar(60),
    -> email varchar(40),
    -> no_telepon varchar(15),
    -> sex char(1));
```

4. LATIHAN

Perhatikan Skema Basis Data Rumah Sakit berikut:

PASIEN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY
NAMA_P	Varchar(20)	

DOKTER

FIELD	TIPE DATA	CONSTRAINT
DOKTER_ID	Char(4)	PRIMARY KEY
NAMA_D	Varchar(20)	
NO_TELEPON	Varchar(12)	

PEMERIKSAAN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY dan FOREIGN KEY
DOKTER_ID	Char(4)	PRIMARY KEY dan FOREIGN KEY
TANGGAL	Date	
NO_TELEPON	Varchar(12)	
PENJELASAN_PEMERIKSAAN	Varchar(30)	

1. Buatlah basis data RumahSakit
2. Buatlah tabel PASIEN, DOKTER dan PEMERIKSAAN sesuai dengan skema diatas, lengkapi dengan PRIMARY dan FOREIGN KEY
3. Gantilah nama tabel PEMERIKSAAN menjadi PERAWATAN
4. Ubahlah ukuran/tipe data nama_d pada nama dokter menjadi varchar (30)
5. Tambahkan kolom umur pada tabel pasien
6. Hapuslah kolom umur pada tabel pasien

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 4

Pertemuan ke-5

Constraint Key

1. TUJUAN

Mahasiswa mampu melakukan menambah constraint primary key, menambah constraint foreign key, menambah nilai default, manghapus constraint primary dan foreign key

2. TEORI SINGKAT

Primary key adalah satu kolom atau gabungan beberapa kolom pada table MySQL yang memiliki nilai unik. Berdasarkan banyaknya anggota komunitas Belajar SQL kami di Facebook yang menanyakan bagaimana menambahkan primary key pada table yang sudah ada dengan menggunakan bahasa SQL, maka artikel berikut coba memberikan contoh untuk melakukan hal tersebut.

Syntax Dasar

```
ALTER TABLE NAMA_TABLE ADD PRIMARY KEY(KOLOM1, KOLOM2, ...);
```

Contoh Primary Key dengan 1 Kolom

Untuk contoh pembuatan primary key dengan 1 kolom, berikut adalah gambar struktur table yang akan digunakan.

Field	Type	Comment
CUSTOMERNUMBER	int(11) NULL	
CUSTOMERNAME	varchar(50) NULL	
CONTACTLASTNAME	varchar(50) NULL	
CONTACTFIRSTNAME	varchar(50) NULL	
PHONE	varchar(50) NULL	
ADDRESSLINE1	varchar(50) NULL	
ADDRESSLINE2	varchar(50) NULL	
CITY	varchar(50) NULL	
STATE	varchar(50) NULL	
POSTALCODE	varchar(15) NULL	
COUNTRY	varchar(50) NULL	
SALESREPEMPLOYEENUMBER	int(11) NULL	
CREDITLIMIT	bigint(20) NULL	

Nama table tersebut adalah **CUSTOMERS** dan satu kolom, yaitu kolom **CUSTOMERNAME** akan digunakan sebagai primary key.

Berikut adalah perintah SQL untuk membuat primary key tersebut.

```
ALTER TABLE CUSTOMERS ADD PRIMARY KEY(CUSTOMERNUMBER);
```

Perhatikan bahwa kolom CUSTOMERNUMBER telah diberi tanda simbol kunci dan dari sebelumnya bisa mengandung nilai NULL menjadi NOT NULL.

Contoh Primary Key dengan 2 Kolom

Untuk pembuatan primary key dengan beberapa bkolom, berikut adalah contoh struktur table yang akan digunakan.

Field	Type	Comment
ORDERNUMBER	int(11) NULL	
PRODUCTCODE	varchar(50) NULL	
QUANTITYORDERED	int(11) NULL	
PRICEEACH	bigint(20) NULL	
ORDERLINENUMBER	int(11) NULL	

Nama table tersebut adalah **ORDERDETAILS** dan dua kolom, yaitu kolom **ORDERNUMBER** dan **PRODUCTCODE** akan digunakan sebagai primary key.

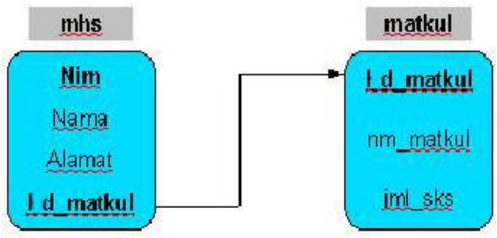
Berikut adalah perintah SQL untuk membuat primary key tersebut.

```
ALTER TABLE ORDERDETAILS ADD PRIMARY KEY (ORDERNUMBER, PRODUCTCODE);
```

Terlihat kedua kolom table, yaitu ORDERNUMBER dan PRODUCTCODE telah diberi simbol kunci dan keduanya tidak boleh mengandung nilai NULL.

Foreign Key Di MySQL

Dalam relational basis data, terdapat istilah kunci utama (primary key) dan kunci tamu (foreign key). Tujuan utama dari adanya kedua istilah tersebut adalah pengidentifikasian tiap table dan kejelasan hubungan antara 2 atau lebih table. Primary key adalah satu atau beberapa kolom pada table yang mengidentifikasikan tiap kolom dan baris pada table tersebut. Sedangkan Foreign Key adalah satu atau beberapa kolom pada table yang merupakan primary key pada table satu namun diletakan pada table dimana tablenya berelasi dengan table dirinya.



Pada gambar diatas, nim pada table mhs adalah primary key. Sedangkan primary key pada table matkul adalah id_matkul. Lalu id_matkul menempel pada table mhs yang kita sebut sebagai foreign key.

Bisa juga disebut table yang berisi foreign key sebagai table anak karena table tersebut mengait pada table lain. Sedangkan table yang terkait saya sebut sebagai

table induk. Pada contoh table di atas, mhs sebagai table anak sedangkan matkul sebagai table induk.

Pada MYSQL, kita harus menambahkan perintah ON DELETE [opsi] dan ON UPDATE [opsi]pada table yang mereferensikan foreign key. Opsi pada perintah tersebut jelasnya dibawah ini.

1. RESTRICT, Jika tabel anak berisi nilai dalam kolom yang mengkait yang nilainya sama dengan di kolom terkait pada tabel induk, baris dalam tabel induk tidak bisa dihapus, dan nilai di kolom terkait tidak dapat diupdate. Ini adalah opsi default jika klausa ON DELETE atau ON UPDATE tidak dispesifikasikan.

2. CASCADE, Baris-baris dalam tabel anak yang berisi nilai-nilai yang juga terdapat dalam kolom terkait dari tabel induk dihapus ketika barisbaris yang berkaitan dihapus dari tabel induk. Baris-baris dalam tabel anak yang berisi nilai-nilai yang juga terdapat dalam kolom terkait dari tabel induk diupdate ketika nilai-nilai yang berkaitan diupdate dalam tabel induk.

3. SET NULL, Nilai-nilai dalam kolom yang mengkait dari tabel anak diset ke NULL saat baris-baris dengan data terkait dalam tabel induk dihapus dari tabel induk atau ketika data terkait dalam tabel induk diupdate. Untuk menggunakan opsi ini, semua kolom-kolom yang mengkait dalam tabel anak harus mengijinkan nilai NULL.

4. NO ACTION Tidak ada aksi yang diambil dalam tabel anak ketika baris-baris dihapus dari tabel induk atau nilai-nilai dalam kolom terkait dalam tabel induk diupdate.

5. SET DEFAULT Nilai-nilai dalam kolom-kolom yang mengkait dari tabel anak diset ke nilai default mereka ketika baris-baris dihapus dari tabel induk atau kolom terkait dari tabel induk diupdate.

Ketika kita mendefinisikan foreign key dengan FOREIGN KEY(id_matkul) REFERENCES matkul(id_matkul).

Maka perintah ON DELETE dan ON UPDATE nya ber-Opsi RESTRICT karena defaultnya dari references foreign key, yang artinya tidak boleh dihapus atau diupdate. Itulah yang menyebabkan error bila kita mendelete atau mengapdate table induk.

3. PELAKSANAAN PRAKTIKUM

Buatlah tabel mhs dan matkul dengan struktur sebagai berikut:

Table mhs

Nim	Nama	Alamat	Id_matkul
10107644	Adiputra Artupida	Bekasi	IF123
10107634	Hermansyah	Bandung	IF222
10107635	Budianto Nugroho	Semarang	IF111
10107636	Silan	Bekasi	IF333

Table Matkul

Id_matkul	Nm_matkul	Jml_sks
IF123	Pemrograman Web	3
IF111	Pemrograman C	3
IF222	Pemrograman Java	3
IF333	Pemrograman OOP	3

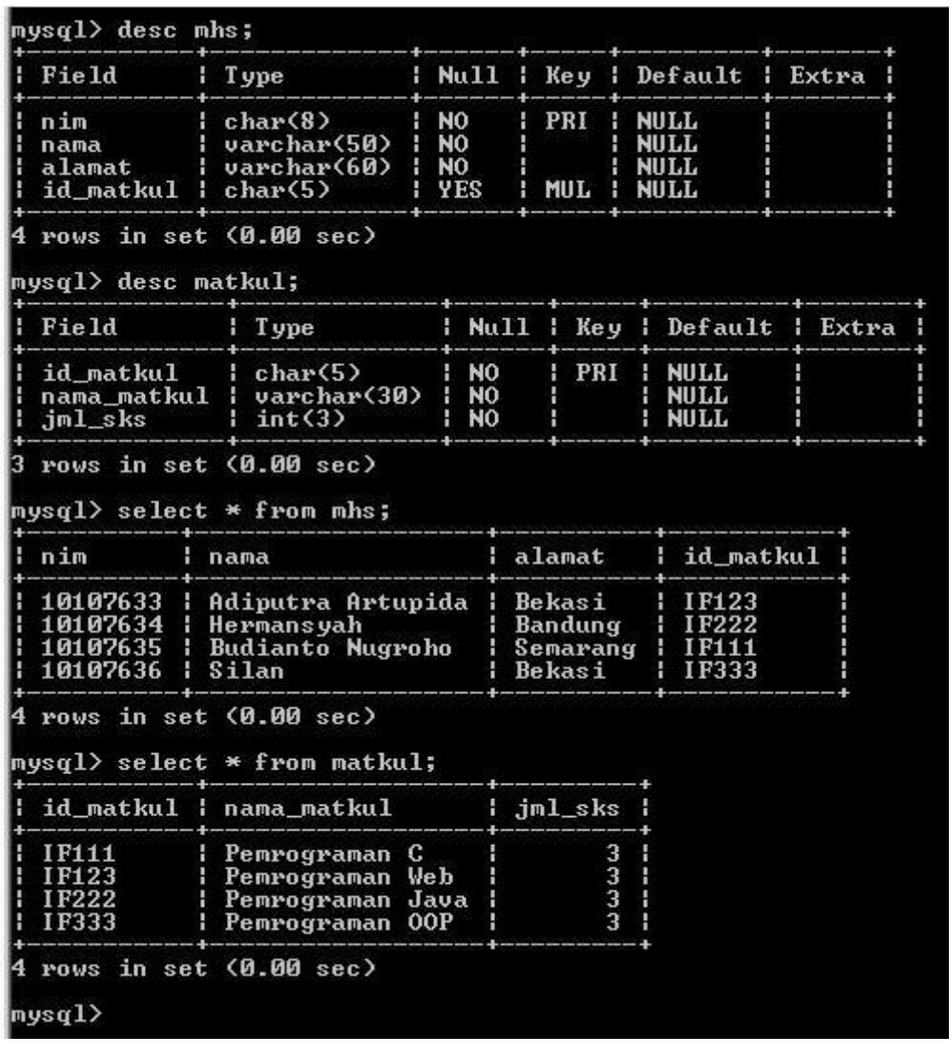
Berikut sintaksnya :

```
CREATE TABLE IF NOT EXISTS matkul(
id_matkul CHAR(5) NOT NULL PRIMARY KEY,
nm_matkul VARCHAR(30) NOT NULL,
jml_sks INT(3) NOT NULL
);

CREATE TABLE IF NOT EXISTS mhs(
nim CHAR(8) NOT NULL PRIMARY KEY,
nama VARCHAR(50) NOT NULL,
alamat VARCHAR(60) NOT NULL,
id_matkul CHAR(5),
FOREIGN KEY(id_matkul) REFERENCES matkul(id_matkul)
);

INSERT INTO mhs VALUES
('10107633','Adiputra Artupida','Bekasi','IF123'),
('10107634','Hermansyah','Bandung','IF222'),
('10107635','Budianto Nugroho','Semarang','IF111'),
('10107636','Silan','Bekasi','IF333');

INSERT INTO matkul VALUES
('IF123','Pemrograman Web',3),
('IF111','Pemrograman C',3),
('IF222','Pemrograman Java',3),
('IF333','Pemrograman OOP',3);
```



```
mysql> desc mhs;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nim       | char(8)     | NO   | PRI | NULL    |       |
| nama      | varchar(50) | NO   |     | NULL    |       |
| alamat    | varchar(60) | NO   |     | NULL    |       |
| id_matkul | char(5)     | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc matkul;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| id_matkul   | char(5)     | NO   | PRI | NULL    |       |
| nama_matkul | varchar(30) | NO   |     | NULL    |       |
| jml_sks     | int(3)      | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> select * from mhs;
+----------+------------------+----------+------------+
| nim      | nama             | alamat   | id_matkul  |
+----------+------------------+----------+------------+
| 10107633 | Adiputra Artupida | Bekasi   | IF123      |
| 10107634 | Hermansyah       | Bandung  | IF222      |
| 10107635 | Budianto Nugroho | Semarang | IF111      |
| 10107636 | Silan            | Bekasi   | IF333      |
+----------+------------------+----------+------------+
4 rows in set (0.00 sec)

mysql> select * from matkul;
+------------+-----------------+---------+
| id_matkul  | nama_matkul     | jml_sks |
+------------+-----------------+---------+
| IF111      | Pemrograman C   |       3 |
| IF123      | Pemrograman Web |       3 |
| IF222      | Pemrograman Java |       3 |
| IF333      | Pemrograman OOP |       3 |
+------------+-----------------+---------+
4 rows in set (0.00 sec)

mysql>
```

Perhatikan baris kode pada table mhs,

FOREIGN KEY(id_matkul) REFERENCES matkul(id_matkul).
Itulah cara mendefinikan foreign key pada mysql. Id_matkul didefinisikan sebagai foreign key yang mereferensikan(mengacu) pada table matkul pada kolom id_matkul.

Bagaimana bila data matkul misalnya data dengan id_matkul='IF123' dihapus? Apakah data mahasiswa yang mengambil id_matkul akan dihapus juga atau akan dibiarkan apa adanya?

DELETE FROM matkul WHERE id_matkul='IF123'
Maka pasti terjadi error

"Cannot delete or update a parent row: a foreign key constaint fails ('test'.'mhs',CONSTRAINT 'mhs_ibfk_3' FOREIGN KEY ('id_matkul') REFERENCES 'matkul' ('id_matkul))

Maksudnya adalah tidak dapat menghapus atau mengupdate kolom pada table induk karena bereferensi pada table mhs. Lalu bagaimana solusinya?

Ketika kita mendefinisikan foreign key dengan FOREIGN KEY(id_matkul) REFERENCES matkul(id_matkul).

Maka perintah ON DELETE dan ON UPDATE nya ber-Opsi RESTRICT karena defaultnya dari references foreign key, yang artinya tidak boleh dihapus atau diupdate. Itulah yang menyebabkan error bila kita mendelete atau mengapdate table induk.

Coba sekarang kita ubah struktur table dari mhs dengan merubah references foreign key dengan opsi ON DELETE dan ON UPDATE CASCADE.

```
mysql> ALTER TABLE mhs
    -> ADD FOREIGN KEY (id_matkul) REFERENCES matkul(id_matkul)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE;
Query OK, 4 rows affected (0.38 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> DELETE FROM matkul WHERE id_matkul='IF123';
Query OK, 1 row affected (0.14 sec)

mysql> SELECT * FROM matkul;
+-----------+------------------+----------+
| id_matkul | nama_matkul      | jml_sks  |
+-----------+------------------+----------+
| IF111     | Pemrograman C    |        3 |
| IF222     | Pemrograman Java |        3 |
| IF333     | Pemrograman OOP  |        3 |
+-----------+------------------+----------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM mhs;
+----------+------------------+----------+------------+
| nim      | nama             | alamat   | id_matkul  |
+----------+------------------+----------+------------+
| 10107634 | Hermansyah       | Bandung  | IF222      |
| 10107635 | Budianto Nugroho | Semarang | IF111      |
| 10107636 | Silan            | Bekasi   | IF333      |
+----------+------------------+----------+------------+
3 rows in set (0.00 sec)
```

Karena kita menggunakan OPSI CASCADE pada ON DELETE dan ON UPDATE nya maka bila kita menghapus salah satu kolom pada table induk maka table anak pun akan terhapus juga. Terlihat bahwa data mahasiswa yang bernama Adiputra Artupida dihapus karena Adiputra Artupida mengambil matkul dengan id ‘IF123’ dimana id tersebut telah dihapus pada table matkul.

Yang lainnya silakan improve sendiri. Saran kalau kita ingin merelasikan foreign key biasakan menggunakan perintah ON UPDATE dan ON DELETE dengan OPSI yang disesuaikan dengan kebutuhan.

4. LATIHAN

Perhatikan Skema Basis Data Rumah Sakit berikut:

PASIEN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY
NAMA_P	Varchar(20)	

DOKTER

FIELD	TIPE DATA	CONSTRAINT
DOKTER_ID	Char(4)	PRIMARY KEY
NAMA_D	Varchar(20)	
NO_TELEPON	Varchar(12)	

PEMERIKSAAN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY dan FOREIGN KEY
DOKTER_ID	Char(4)	PRIMARY KEY dan FOREIGN KEY
TANGGAL	Date	
NO_TELEPON	Varchar(12)	
PENJELASAN_PEMERIKSAAN	Varchar(30)	

7. Buatlah basis data RumahSakit tanpa primary dan foreign key
8. Tambahkan constraint primary key pada tabel pasien
9. Tambahkan constraint primary key pada tabel dokter
10. Tambahkan constraint primary key pada tabel pemeriksaan
11. Tambahkan constraint foreign key pada tabel pemeriksaan
12. Tambahkan nilai default pada tabel pemeriksaan

5. TUGAS

Implementasikan Basis Data Rumah Sakit berikut:

PASIEN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY
NAMA_P	Varchar(20)	

DOKTER

FIELD	TIPE DATA	CONSTRAINT
DOKTER_ID	Char(4)	PRIMARY KEY
NAMA_D	Varchar(20)	
NO_TELEPON	Varchar(12)	

ITEM

FIELD	TIPE DATA	CONSTRAINT
ITEM_CODE	Char(3)	PRIMARY KEY
KETERANGAN_ITEM	Varchar(30)	
BIAYA	Numeric(10,2)	

PEMERIKSAAN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY dan FOREIGN KEY
DOKTER_ID	Char(4)	PRIMARY KEY dan FOREIGN KEY
TANGGAL	Date	
NO_TELEPON	Varchar(12)	
PENJELASAN_PEMERIKSAAN	Varchar(30)	

TAGIHAN

FIELD	TIPE DATA	CONSTRAINT
PASIEN_NO	Char(4)	PRIMARY KEY dan FOREIGN KEY
ITEM_CODE	Char(3)	PRIMARY KEY dan FOREIGN KEY
TANGGAL	Date	

PASIEN

PASIEN_NO	NAMA_P
3249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Pertemuan ke-6

Data Definition Language

1. TUJUAN

Mahasiswa mampu menggunakan perintah create, alter, drop untuk database yang kompleks dengan jumlah tabel lebih dari tiga.

2. TEORI SINGKAT

DDL atau Data Definition Language adalah bagian dari sql yang digunakan untuk mendefinisikan data dan objek database. Apabila perintah ini digunakan, entri akan dibuat ke dalam kamus data dari SQL. Perintah DDL sebagai berikut :

Perintah	Keterangan
Create Database	Membuat database
Create Table	Membuat tabel
Create Index	Membuat index
Create View	Membuat View
Alter Table	Mengubah atau menyisipkan kolom ke dalam tabel
Drop Database	Menghapus database
Drop Table	Menghapus tabel dari database
Drop Index	Menghapus index
Drop View	Menghapus view
Grand	Memberikan ijin akses kepada user

Indeks dalam database dapat diumpamakan seperti indeks dalam sebuah buku yang tebal, sehingga item tertentu dapat ditemukan dengan cepat. Sebuah indeks dalam basis data berfungsi untuk mempercepat pencarian data berdasarkan kolom tertentu. Perintah untuk membuat indeks sebagai berikut :

```
Create (unique) Index namajndex on namajabel (namajcolom);
```

Keterangan :

1. ***unique*** **->** pilihan perincian yang dapat digunakan untuk menguatkan nilai data di dalam kolom nama index menjadi unik.
2. ***namajndex*** **->** nama index yang baru.
3. ***nama_tabel*** **->** nama tabel yang berisi kolom index akan dibuat.
4. ***nama_kolom*** -> nama dari kolom tempat index akan dibuat. Yang terdiri dari

Data Definition Language (DDL) digunakan untuk membuat dan menghancurkan database dan objek database. Perintah-perintah ini terutama akan digunakan oleh database administrator selama fase setup dan penghapusan proyek database.
Struktur dan penggunaan perintah DDL empat dasar:

a. **CREATE**
   Instalasi sistem manajemen database (DBMS) pada komputer memungkinkan Anda untuk membuat dan mengelola banyak database independen
b. **USE**
   Perintah USE memungkinkan Anda untuk menentukan database yang ingin bekerja dengan Anda dalam DBMS.
c. **ALTER**
   Setelah Anda telah membuat tabel dalam database, Anda mungkin ingin memodifikasi definisi itu.Perintah ALTER yang memungkinkan Anda untuk

membuat perubahan pada struktur tabel tanpa menghapus dan menciptakan tabel baru dengan nama yang berbeda.

d. **DROP**

Perintah terakhir dari Data Definition Language, DROP yang memungkinkan kita untuk menghapus seluruh objek database dari DBMS. Gunakan perintah ini dengan hati-hati! Ingat bahwa perintah DROP menghapus data keseluruhan struktur dari database Anda.

Constraint

Constraint adalah batasan atau aturan yang ada pada table. MySQL menyediakan beberapa tipe constraint berikut :

- ***NOT NULL***
  Suatu kolom yang didefinisikan dengan constraint NOT NULL tidak boleh berisi nilai NULL. Kolom yang befungsi sebagai kunci primer (primary key) otomatis tidak boleh NULL.
- ***UNIQUE***
  Mendefinisikan suatu kolom menjadi bersifat unik, artinya antara satu data dengan data lainnya namanya tidak boleh sama, misal alamat email.
- ***PRIMARY KEY***
  Constraint PRIMARY KEY membentuk key yang unik untuk suatu table.
- ***FOREIGN KEY***

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

3. PELAKSANAAN PRAKTIKUM

Praktikum 1 : Membuat Tabel

Perintah yang digunakan untuk membuat tabel menggunakan perintah berikut :

```
CREATE TABLE namatabel
(
Field1 TipeData1 NOT NULL PRIMARY KEY,
Field2 TipeData2
);
```

Keterangan :

- ***nama_tabel* ->** nama yang diberikan di tabel baru. Nama tabel maksimal terdiri dari 8 karakter. Tidak boleh memakai spasi, terdiri dari huruf.
- ***Field* ->** nama yang diberikan untuk kolom baru, maksimal terdiri dari 10 karakter. Tidak boleh memakai spasi, terdiri dari huruf, angka dan lain-lain.
- ***type_data* ->** jenis data yang nilainya dimasukkan dalam kolom yang telah ditentukan.
- lebar_data^ nomor spasi karakter untuk mengikuti data yang dimasukkan dalam kolom yang telah ditentukan.
- ***Constraint* ->** batasan yg digunakan utk field seperti NOT Null, Primary Key.

Contoh, buat tabel Departemen, Untuk membuat tabel dalam database kepegawaian terlebih dahulu gunakan perintah ***USE namadatabase*** untuk memilih database yg akan digunakan.

```
mysql> use kepegawaian;
Database changed
```

Kemudian gunakan perintah untuk membuat tabel seperti berikut :

```
mysql> Create Table Departemen
    -> (Nomor int Not Null Primary Key,
    -> Nama varchar(15),
    -> JmlPegawai int,
    -> NoKTP int);
Query OK, 0 rows affected (0.12 sec)
```

Praktikum 2 : Mendefinisikan Nilai Default

Nilai default adalah nilai yang otomatis diberikan oleh sistem untuk suatu kolom ketika ada penambahan baris baru, sementara nilai pada kolom tersebut tidak diisi oleh pengguna, perintah sebagai berikut :

```
mysql> Create table Pegawai
    -> ( NoKTP int Not Null,
    -> NmDepan varchar(25) ,
    -> NmBlk varchar(25),
    -> JnsKel varchar(10),
    -> Alamat varchar(25) Default Null,
    -> Gaji int,
    -> NoKTPpimpinan int,
    -> Nomor int);
Query OK, 0 rows affected (0.08 sec)
```

Praktikum 3 : Membuat dan Menghapus Indeks

Buat index data Departemen berdasarkan Nomor dengan nama indeks dept, maka perintah yang digunakan sebagai berikut :

```
mysql> Create index dept ON Departemen(Nomor);
Query OK, 1 row affected (0.05 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

Untuk menghapus indeks yang telah dibuat gunakan perintah berikut :

```
Drop Index nama_index on nama_tabel;
```

Contoh:

```
mysql> DROP index dept on Departemen;
Query OK, 1 row affected (0.09 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

Praktikum 4 : Mendefinisikan Primary Key Pada Tabel

Terdapat tiga cara untuk mendefinisikan primary key. Berikut ini adalah perintah mendefinisikan primary key untuk Field1

```
CREATE TABLE namatabel
(
      Field1 TipeData1 NOT NULL PRIMARY KEY,
      Field2 TipeData2
);
```

Atau

```
CREATE TABLE namatabel
```

```
(
        Field1 TipeData1,
        Field2 TipeData2,
        PRIMARY KEY(Field1)
);
```

Atau

```
ALTER TABLE namatabel ADD CONSTRAINT PRIMARY KEY
(namakolom);
```

Tabel pegawai diatas belum mempunyai primary key, maka utk menambahkan primary key gunakan perintah berikut :

```
mysql> Alter table pegawai ADD Constraint Primary Key(NoKTP);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> _
```

Praktikum 5 : Menghapus Primary Key Pada Tabel

Perintah untuk menghapus primary key pada tabel sebagai berikut :

Cara 1 : Jika primary key dibuat dengan menggunakan alter table

```
ALTER TABLE namatabel DROP CONSTRAINT namaconstraint;
```

Cara 2 : Jika primary key dibuat melalui create table :

```
ALTER TABLE namatabel DROP PRIMARY KEY;
```

```
mysql> Alter table pegawai Drop Primary Key;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Praktikum 6 : Menambah Kolom Baru Pada Tabel

Perintah untuk menambah kolom baru pada tabel seperti berikut :

```
ALTER TABLE namatabel ADD fieldbaru tipe;
```

Lakukan perintah berikut untuk menambahkan field/atribut NoTelp pada tabel pegawai :

```
mysql> Alter table pegawai ADD NoTelp varchar(15);
Query OK, 0 rows affected (0.41 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Praktikum 7 : Mengubah Tipe Data atau Lebar Kolom Pada Tabel

Perintah yang digunakan :

```
ALTER TABLE namatabel MODIFY COLUMN field tipe
```

Lakukan perintah berikut untuk mengubah tipe data dan lebar kolom NoTelp pada tabel pegawai dari varchar(15) menjadi char(12)

```
mysql> alter table pegawai MODIFY Column NoTelp char(12);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Praktikum 8 : Mengubah Nama Kolom

Perintah yang digunakan :

```
ALTER TABLE namatabel CHANGE COLUMN NamaKolomLama
NamaKolomBaru tipedata;
```

Berikut ini perintah untuk mengubah nama kolom NoTelp menjadi Telp :

```
mysql> Alter table pegawai Change Column NoTelp Telp varchar(12);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Praktikum 9 : Menghapus Kolom Pada Tabel

Perintah untuk menghapus kolom pada tabel:

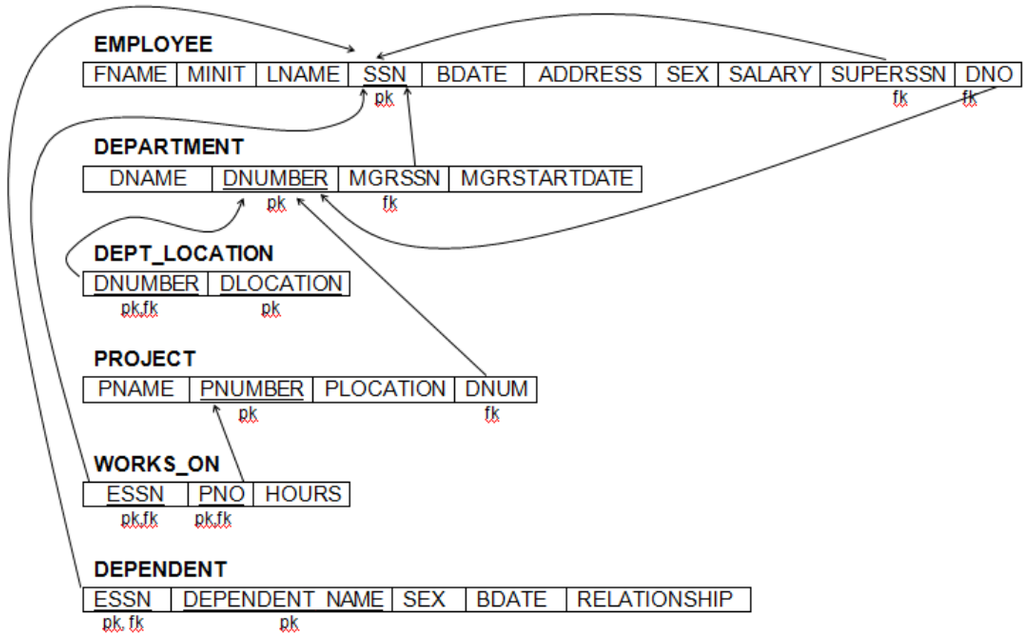
```
ALTER TABLE namatabel DROP COLUMN namakolom;
```

Lakukan perintah berikut untuk menghapus field/kolom Telp pada tabel pegawai :

```
mysql> alter table pegawai Drop Column Telp;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4. LATIHAN

Implementasikanlah skema basis data COMPANY sesuai dengan skema Gambar 1!



Gambar 1. **Skema Basis Data COMPANY**

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	09-jan-55	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	08-dec-45	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-jul-58	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-jun-31	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-sep-52	975 Are Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-jul-62	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-mar-59	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	10-nov-27	450 Stone, Houston, TX	M	55000	null	1

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-may-78
Administration	4	987654321	01-jan-85
Headquarters	1	888665555	19-jan-71

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
Produce	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	05-apr-76	Daughter
333445555	Theodore	M	25-oct-73	Son
333445555	Joy	F	03-may-46	Spouse
987654321	Abner	M	29-feb-32	Spouse
123456789	Michael	M	01-jan-78	Son
123456789	Alice	F	31-dec-78	Daughter
123456789	Elizabeth	F	05-may-57	Spouse

DEPT_LOCATION

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Gambar 2. **Basis Data COMPANY**

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 7

Pertemuan ke-8

Data Manipulation Language

1. TUJUAN

Mahasiswa mampu memahami manipulasi data menggunakan perintah SQL-DML menggunakan perintah INSERT, Perintah UPDATE, dan Perintah DELETE.

2. TEORI SINGKAT

DML adalah fungsi untuk melakukan manipulasi database yang telah dibuat. Perintah DML adalah:

INSERT	:	Digunakan untuk memasukkan data pada Tabel Database
UPDATE	:	Digunakan untuk pengubahan terhadap data yang ada pada Tabel Database
DELETE	:	Digunakan untuk Penhapusan data pada tabel Database

INSERT
Memasukkan data atau entry data, dalam semua program yang menggunakan query SQL sebagai standar permintaannya, digunakan perintah INSERT. Syarat untuk memasukkan data adalah telah terciptanya tabel pada sebuah database. Sintax yang digunakan adalah :
INSERT INTO nama_tabel VALUES (‘isi_field1’, ‘isi_field2’, ‘isi_field3’,….., ‘isi_fieldN’);

Contoh :

```
mysql> insert into tb_tamu values('1','Boi trimoyo','ujung
berung','bo_i77@yahoo.com','085613548789');
```

Query OK, 1 row affected (0.05 sec)
Maka data telah masuk ke dalam tabel seperti berikut :

```
mysql> select * from tb_tamu;
+----+-------------+--------------+-------------------+--------------+
| no | nama        | alamat       | email             | no_telp      |
+----+-------------+--------------+-------------------+--------------+
|  1 | Boi trimoyo | ujung berung | bo_i77@yahoo.com | 085613548789 |
+----+-------------+--------------+-------------------+--------------+
1 row in set (0.01 sec)
```

UPDATE
Memperbarui isi data atau update data adalah sebuah proses meremajakan data lama menjadi data yang lebih baru. Namun tidak semua data dalam database yang perlu diremajakan, melainkan sebagian data yang dianggap perlu untuk diremajakan. Query SQL yang digunakan adalah UPDATE yang di ketikkan seperti berikut :
UPDATE nama_tabel SET
field_1 = ‘data_baru’,
field_2 =’data_baru’,…… ,Field_N =’data_baru’;

Contoh :

```
mysql> update tb_tamu set
    -> nama='irfan nurhudin' where nama="Boi trimoyo";
```

Query OK, 1 row affected (0.08 sec)
Rows matched: 1 Changed: 1 Warnings: 0
Maka hasilnya akan berubah seperti berikut :

```
mysql> select * from tb_tamu;
+----+----------------+--------------+-------------------+--------------+
| no | nama           | alamat       | email             | no_telp      |
+----+----------------+--------------+-------------------+--------------+
|  1 | irfan nurhudin | ujung berung | bo_i77@yahoo.com  | 085613548789 |
+----+----------------+--------------+-------------------+--------------+
1 row in set (0.00 sec)
```

Data yang asalnya bernama Boi trimoyo berubah menjadi irfan nurhudin yang dihasilkan dari query SQL UPDATE.

DELETE

Unutk menghapus data, MySQL memiliki query bernama DELETE. Penggunaannya diikuti dengan nama data yang akan dihapus. Berikut Sintax untuk menghapus semua data yang terdapat pada tabel :

DELETE FROM nama_tabel;

Sedangkan berikut sintax untuk menghapus data yang diinginkan dari sebuah tabel :

DELETE FROM nama_tabel WHERE kondisi;

Contoh :

Isikan data pada tabel tb_tamu seperti dibawah ini :

```
+----+-----------------+--------------+---------------------+--------------+
| no | nama            | alamat       | email               | no_telp      |
+----+-----------------+--------------+---------------------+--------------+
|  1 | irfan nurhudin  | ujung berung | bo_i77@yahoo.com    | 085613548789 |
|  2 | Boi trimoyo     | cibiru       | bo_i77@yahoo.com    | 082246864846 |
|  3 | Muswanto        | kopo         | muswanto@yahoo.com  | 0229166478   |
|  4 | mochammad ridwan | bale endah  | mory_89@yahoo.com   | 02270598723  |
+----+-----------------+--------------+---------------------+--------------+
4 rows in set (0.00 sec)
```

Untuk menghapus data nomor 4 yang terdapat pada tabel tb_tamu maka gunakan query sepertin dibawah ini :

```
mysql> delete from tb_tamu where no='4';
Query OK, 1 row affected (0.03 sec)
```

Maka hasilnya akan seperti dibawah ini bahwa data nomor 4 yang bernama ridwan telah dihapus menggunakan query DELETE :

```
mysql> select * from tb_tamu;
+----+----------------+--------------+---------------------+--------------+
| no | nama           | alamat       | email               | no_telp      |
+----+----------------+--------------+---------------------+--------------+
|  1 | irfan nurhudin | ujung berung | bo_i77@yahoo.com    | 085613548789 |
|  2 | Boi trimoyo    | cibiru       | bo_i77@yahoo.com    | 082246864846 |
|  3 | Muswanto       | kopo         | muswanto@yahoo.com  | 0229166478   |
+----+----------------+--------------+---------------------+--------------+
3 rows in set (0.00 sec)
```

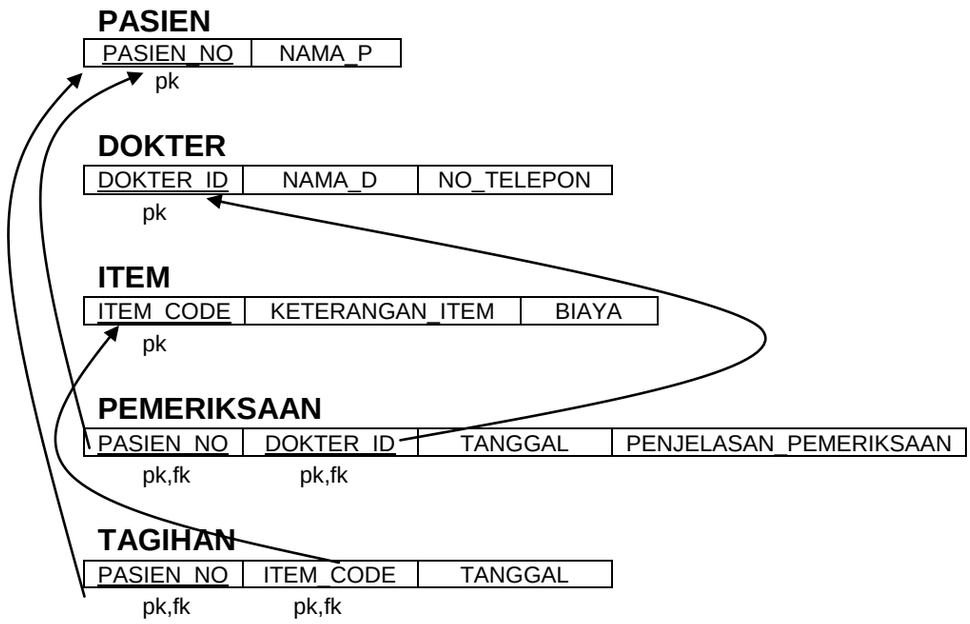
3. PELAKSANAAN PRAKTIKUM

Untuk menggunakan perintah DDL, dapat dilakukan sebagai berikut:

13. Mengaktifkan Direktori MySQL

    Untuk dapat menggunakan MySQL terlebih dahulu aktifkan Server MySQL dengan menghidupkan daemond MySQL. Program MySQL yang digunakan pada modul ini adalah XAMPP 1.7, maka untuk menjalankan daemond MySQL terdapat pada direktori yaitu C:\Program Files\Xampp\Mysql\Bin
14. Buat skema basis data dengan perintah DDL
15. Buat perintah DML

4. LATIHAN



Gambar A. **Skema Basis Data RUMAH SAKIT**

PASIEN

PASIEN_NO	NAMA_P
3249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Gambar B. **Basis Data RUMAH SAKIT**

Berdasarkan skema Basis Data RUMAH SAKIT yang ada pada Gambar A, buatlah perintah DML untuk kasus berikut:

1. Menambahkan data baru ke table pasien dengan nomor pasien = 5631 dan namanya adalah Williams, dengan asumsi sebelumnya data tsb belum ada.
2. Menambahkan data baru ke table pemeriksaan dengan nomor pasien = 6213, id dokternya = A233, tanggal pemeriksaannya 22 November 2006, dengan asumsi sebelumnya data tsb belum ada dan table pasien dan dokternya sudah seperti yang ada di Basis Data Rumah Sakit.
3. Menambahkan penjelasan pemeriksaan untuk nomor pasien = 6213, id dokternya = A233 yaitu Kontrol Rutin, dengan asumsi sebelumnya nilai kolom penjelasan pemeriksaan masih kosong, dan nilai kolom yang lainnya sudah terisi.
4. Merubah biaya pemakaian Television menjadi Rp 50.000, dengan asumsi sebelumnya biayanya = Rp 75.000,-
5. Menghapus data dokter Arnold, dengan asumsi data tersebut tidak pernah digunakan pada table pemeriksaan.

6. TUGAS

Sesuai GBPP **ada** tugas pada pertemuan 8

Pertemuan ke-9

Data Control Language

1. TUJUAN

Mahasiswa mampu mengontrol akses user pada basis data, menggunakan perintah GRANT dan perintah REVOKE

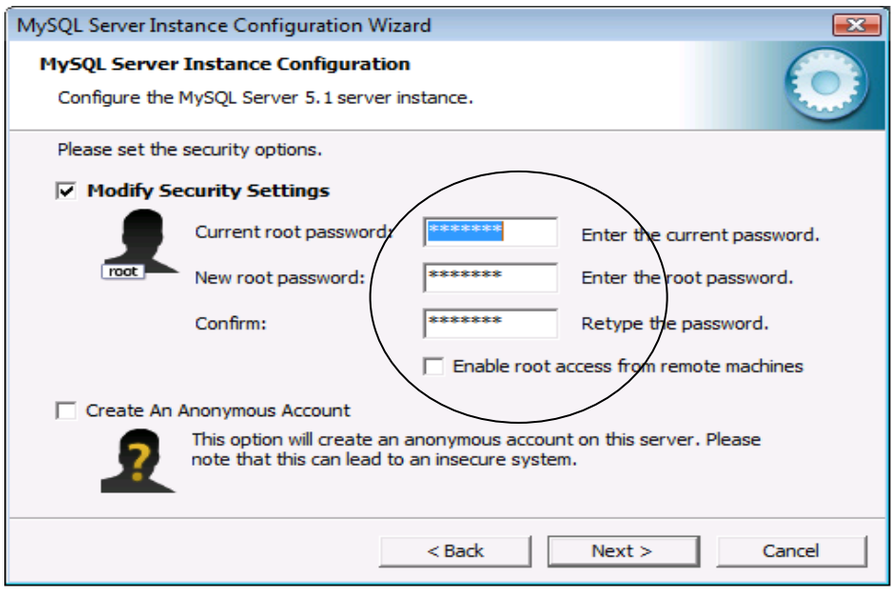
2. TEORI SINGKAT

Pendahuluan

Pada saat instalasi MySQL terdiri dari dua cara :

a. Menggunakan paket aplikasi MySQL (window atau Linux)
b. Menggunakan aplikasi gabungan yaitu xampp

Jika menggunakan paket aplikasi MySQL pada saat instalasi dapat mengaktifkan security setting dengan mengisi root password



Menggunakan aplikasi gabungan tidak ada fasilitas security atau password

Sangat berbahaya karena siapapun dapat mengakses dengan mudah terhadap sistem database. Dalam database mysql terdapat lima buah tabel yang dapat digunakan untuk mengatur user dan izin akses masing-masing user – user privileges. Yaitu : user, db, host, tables_priv dan columns_priv. Kelima tabel ini disebut *grant tables*

Fungsi dari kelima tabel tersebut :

a. User
   Berisi data user yang mendapatkan izin akses MySQL, asal koneksi dan izin akses kepada user
   Tingkatan akses : Global
b. Db
   Mengatur database apa saja yang dapat diakses oleh seorang user dan jenis izin aksesnya

Tingkatan akses : Database

c. Host
   Mengatus asl host yang diperkenankan bagi user untuk mengakses MySQL, jika lebih dari satu host
   Tingkatan akses : Database
d. tables_priv
   Mengatur tabel apa saja yang dapat diakses oleh seorang user dan jenis izin aksesnya
   Tingkatan akses : Tabel
e. columns_priv
   Mengatur kolom (field) apa saja yang dapat diakses oleh seorang user dan jenis izin aksesnya
   Tingkatan akses : Kolom – field

Jenis Izin Akses User – User Privileges

Izin akses bagi user terdirli dari tiga bagian, yaitu :

1. Tingkatan akses uset biasa
   Mencakup izin akses kedalam database atau kolom, yaitu :
   a. ALTER
   b. CRETATE
   c. DELETE
   d. DROP
   e. INDEX
   f. INSERT
   g. SELECT
   h. UPDATE
   i. REFERENCES
2. Tingkatan akses administrator –Global administrative
   Hanya digunakan oleh user setingkat root atau administrator dan tidak diberikan kepada user biasa, yaitu :
   a. FILE
   b. PROCESS
   c. RELOAD
   d. SHUTDOWN
   e. CREATE TEMPORARY TABLE
   f. EXCUTE
   g. LOCK TABLES
   h. REPLICATION CLIENT
   i. REPLICATION SLAVE
   j. SHOW DATABASES
   k. SUPER
3. Tingkatan Akses khusus – Special privileges
   Dapat diterapkan pada setiap user dengan izin akses sebagai berikut :
   a. ALL
   b. USAGE

Menghapus Anonym User

Dengan tabel user, kita dapat mengetahui bahwa setiap kolom – field mewakili masing- masing 1 jenis izin akses user. Jika terdapat terdapat user yang kosong pada kolom user (tanpa nama user), dengan user dan password yang kosong, maka siapapun dapat masuk ke dalam database server mysql.

Dan jika dalam kolom host terdapat ”%”, berarti user yang bersangkutan dapat mengakses mysql dari komputer mana saja.

Untuk langkah pengamanan awal dapat lakukan perintah

delete from user **where** user=’’ ;

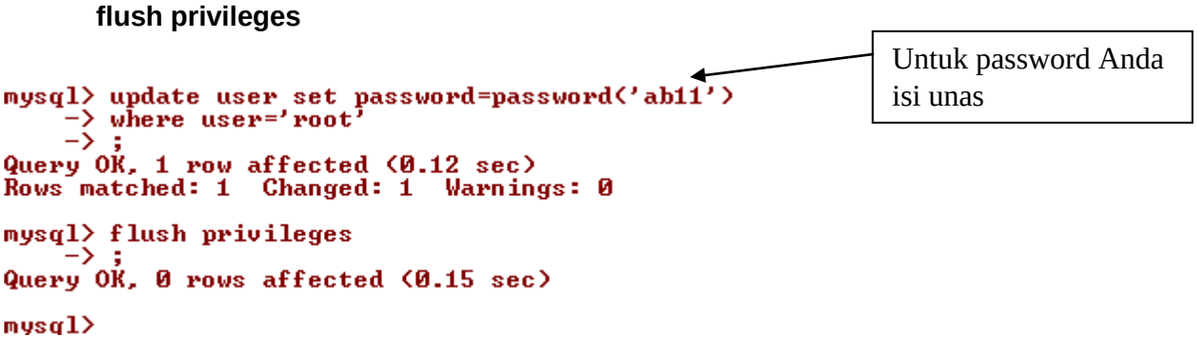
Memberikan Password Untuk Root

Dapat dilakukan dengan perintah Update

update user **set** password=**password(’xxxxxxxxxx’)**
where user=’root’ ;

Lanjutkan dengan perintah FLUSH

flush privileges



Fungsi flush

Mysql membaca grant tables hanya satu kali pada saat server pertama kali jalankan, perintah flush akan memerintahkan kepada sistem untuk membaca ulang kelima grant tables tanpa harus me=restart server mysql.

Manajemen User

Untul MySQL versi 3.22. keatas dalam manajemen user dapat menggunakan perintah GRANT dan REVOKE

Perintah GRANT

Dipergunakan untuk membuat user baru dengan izin aksesnya

Bentuk umum :

GRANT jenis_akses (nama_kolom) **ON** nama_database
TO nama_user **IDENTIFIED BY** ”nama_password”
[**WITH GRANT** pilihan_akses]

Perintah REVOKE

Untuk menghapus izin akses user

Bentuk umum :

REVOKE jenis_akses **ON** nama_database
FROM nama_user

Perintah DELETE

Untuk menghapus user secara permanen

Memberikan Izin Akses tertentu
Jika akan memberikan izin akses SELECT, INSERT, UPDATE dan DELETE kepada user maka hanya boleh dilakukan oleh user dalam akses root atau user yang diberikan zin akses setingkat administrator.

Memberikan izin akses per tabel dan per kolom
Dengan perintah grant dapat digunakan untuk memberikan izin akses per tabel dan per kolom tabel.

Memberikan izin akses berdasarkan lokasi pengakses
Admin atau root atau memberikan izin akses berdasarkan lokasi atau membatasi komputer mana saja yang dapat mengakses MySQL server.

Menghapus izin akses
Menggunakan perintah REVOKE, penggunaan perintah revoke ini hanya menghapus izin akses untuk user tertentu, bukan penghapus user. User yang bersangkutan tetap dapat login ke MySQL.

Bentuk umum :

REVOKE jenis_akses **ON** nama_database
FROM nama_user

Konsep memberikan izin akses user – privileges user merupakan hal yang sangat penting dalam menyangkut masalah keamanan pada MySQL.

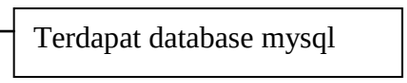
3. PELAKSANAAN PRAKTIKUM

Mengaktifkan Keamanan Standar
Jika menggunakan aplikasi gabungan xampp, setelah selesai instalasi, maka pertama kali harus dilakukan adalah mengaktifkan kata sandi untuk *root* dan menghapus *anonym user*

Anonym user adalah user tanpa identitas dan password
Aktifkan dan masuk ke dalam sistem sebagi root dan jalankan MySQL
Lakukan perintah untuk melihat database

```
mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cdcol              |
| latihan1           |
| latihvien1db       |
| mysql              |
| phpmyadmin         |
| test               |
| webauth            |
+--------------------+
8 rows in set (0.04 sec)

mysql>
```



Aktifkan database mysql dan lihat tabel

```
mysql> use mysql ;
Database changed
mysql> show tables
    -> ;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| proc                      |
| procs_priv                |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
17 rows in set (0.01 sec)

mysql>
```

Table user, data akses user

Untuk berlatih, lihat dulu struktur tabel user
Lakukan perintah describe user

```
mysql> describe user
    -> ;
+------------------------+-----------------------------------+------+-----+---------+-------+
| Field                  | Type                              | Null | Key | Default | Extra |
+------------------------+-----------------------------------+------+-----+---------+-------+
| Host                   | char(60)                          | NO   | PRI |         |       |
| User                   | char(16)                          | NO   | PRI |         |       |
| Password               | char(16)                          | NO   |     |         |       |
| Select_priv            | enum('N','Y')                     | NO   |     | N       |       |
| Insert_priv            | enum('N','Y')                     | NO   |     | N       |       |
| Update_priv            | enum('N','Y')                     | NO   |     | N       |       |
| Delete_priv            | enum('N','Y')                     | NO   |     | N       |       |
| Create_priv            | enum('N','Y')                     | NO   |     | N       |       |
| Drop_priv              | enum('N','Y')                     | NO   |     | N       |       |
| Reload_priv            | enum('N','Y')                     | NO   |     | N       |       |
| Shutdown_priv          | enum('N','Y')                     | NO   |     | N       |       |
| Process_priv           | enum('N','Y')                     | NO   |     | N       |       |
| File_priv              | enum('N','Y')                     | NO   |     | N       |       |
| Grant_priv             | enum('N','Y')                     | NO   |     | N       |       |
| References_priv        | enum('N','Y')                     | NO   |     | N       |       |
| Index_priv             | enum('N','Y')                     | NO   |     | N       |       |
| Alter_priv             | enum('N','Y')                     | NO   |     | N       |       |
| Show_db_priv           | enum('N','Y')                     | NO   |     | N       |       |
| Super_priv             | enum('N','Y')                     | NO   |     | N       |       |
| Create_tmp_table_priv  | enum('N','Y')                     | NO   |     | N       |       |
| Lock_tables_priv       | enum('N','Y')                     | NO   |     | N       |       |
| Execute_priv           | enum('N','Y')                     | NO   |     | N       |       |
| Repl_slave_priv        | enum('N','Y')                     | NO   |     | N       |       |
| Repl_client_priv       | enum('N','Y')                     | NO   |     | N       |       |
| Create_view_priv       | enum('N','Y')                     | NO   |     | N       |       |
| Show_view_priv         | enum('N','Y')                     | NO   |     | N       |       |
| Create_routine_priv    | enum('N','Y')                     | NO   |     | N       |       |
| Alter_routine_priv     | enum('N','Y')                     | NO   |     | N       |       |
| Create_user_priv       | enum('N','Y')                     | NO   |     | N       |       |
| ssl_type               | enum('','ANY','X509','SPECIFIED') | NO   |     |         |       |
| ssl_cipher             | blob                              | NO   |     |         |       |
| x509_issuer            | blob                              | NO   |     |         |       |
| x509_subject           | blob                              | NO   |     |         |       |
| max_questions          | int(11) unsigned                  | NO   |     | 0       |       |
| max_updates            | int(11) unsigned                  | NO   |     | 0       |       |
| max_connections        | int(11) unsigned                  | NO   |     | 0       |       |
| max_user_connections   | int(11) unsigned                  | NO   |     | 0       |       |
+------------------------+-----------------------------------+------+-----+---------+-------+
37 rows in set (0.05 sec)

mysql>
```

Menghapus Anonym User

Dengan tabel user, kita dapat mengetahui bahwa setiap kolom – field mewakili masing- masing 1 jenis izin akses user.

Kita tampilkan dulu data pada kolom, user, host dan password
Perintah :

Select user, host, password **from** user ;

```
mysql> select user, host, password from user
    -> ;
+------+-----------+----------+
| user | host      | password |
+------+-----------+----------+
| root | localhost |          |
| pma  | localhost |          |
+------+-----------+----------+
2 rows in set (0.04 sec)

mysql> _
```

Jika terdapat terdapat user yang kosong pada kolom user (tanpa nama user), dengan user dan password yang kosong, maka siapapun dapat masuk ke dalam database server mysql.
Dan jika dalam kolom host terdapat ”%”, berarti user yang bersangkutan dapat mengakses mysql dari komputer mana saja.

Untuk langkah pengamanan awal dapat lakukan perintah

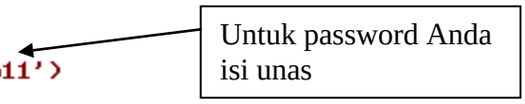
delete from user **where** user=’’ ;

Memberikan Password Untuk Root
Dapat dilakukan dengan perintah Update

update user **set** password=**password(’xxxxxxxxxx’)**
where user=’root’ ;

Lanjutkan dengan perintah FLUSH

flush privileges




```
mysql> update user set password=password('ab11')
    -> where user='root'
    -> ;
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> flush privileges
    -> ;
Query OK, 0 rows affected (0.15 sec)

mysql>
```

Fungsi flush :
Mysql membaca grant tables hanya satu kali pada saat server pertama kali jalankan, perintah flush akan memerintahkan kepada sistem untuk membaca ulang kelima grant tables tanpa harus me=restart server mysql.

Coba Anda periksa dengan perintah :

Select user, host, password **from** user ;

Hasil di kolom password berisi kode acak

```
mysql> select user, host, password from user
    -> ;
+------+-----------+-------------------+
| user | host      | password          |
+------+-----------+-------------------+
| root | localhost | 2f881bb778207d8a |
| pma  | localhost |                   |
+------+-----------+-------------------+
2 rows in set (0.00 sec)

mysql>
```

Untuk memcoba password, keluar dari aplikasi mysql dengan \q
Kemudian coba untuk mengakses kembali tanpa password dan dengan password

```
C:\Program Files\xampp\mysql\bin>mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)

C:\Program Files\xampp\mysql\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 5.0.24a-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Membuat User Baru
Dengan perintah GRANT
Contoh membuat user vivien

```
mysql> grant all privileges on *.* to vivien
    -> identified by 'nova'
    -> with grant option
    -> ;
Query OK, 0 rows affected (0.60 sec)

mysql>
```

Tingkatan akses adalah ALL , user vivien sebagai administrator
ON *.* = dapat meng-akses semua database
TO vivien dapat ditulis TO vivien@% atau TO vivien@localhost

Buat user baru dengan nama “ayyi”
dengan perintah

```
mysql> grant usage on *.* to ayyi
    -> identified by 'fathin'
    -> ;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

User ayyi dapat login ke MySQL dapat mengakses semua database tetapi tidak memiliki izin akses, jadi sebagai dummy user atau blank user

Lihat tabel user

```
mysql> use mysql
Database changed
mysql> select user, host, password from user
    -> ;
+--------+-----------+-------------------+
| user   | host      | password          |
+--------+-----------+-------------------+
| root   | localhost | 2f881bb778207d8a  |
| pma    | localhost |                   |
| vivien | %         | 533e988a566a25d4  |
| ayyi   | %         | 2711be8879736eb1  |
+--------+-----------+-------------------+
4 rows in set (0.05 sec)

mysql> _
```

Bagaimana dengan izin akses ? caranya dengan memeriksa tabel user, yaitu kolom privileges, dengan perintah SELECT * FROM USER

```
mysql> select user, select_priv, insert_priv, update_priv
    -> delete_priv, create_priv, drop_priv
    -> from user
    -> ;
+--------+-------------+-------------+-------------+-------------+-----------+
| user   | select_priv | insert_priv | delete_priv | create_priv | drop_priv |
+--------+-------------+-------------+-------------+-------------+-----------+
| root   | Y           | Y           | Y           | Y           | Y         |
| pma    | N           | N           | N           | N           | N         |
| vivien | Y           | Y           | Y           | Y           | Y         |
| ayyi   | N           | N           | N           | N           | N         |
+--------+-------------+-------------+-------------+-------------+-----------+
4 rows in set (0.01 sec)

mysql> select user, reload_priv, shutdown_priv, process_priv
    -> file_priv, grant_priv
    -> from user
    -> ;
+--------+-------------+---------------+-----------+------------+
| user   | reload_priv | shutdown_priv | file_priv | grant_priv |
+--------+-------------+---------------+-----------+------------+
| root   | Y           | Y             | Y         | Y          |
| pma    | N           | Y             | N         | N          |
| vivien | Y           | Y             | Y         | Y          |
| ayyi   | N           | N             | N         | N          |
+--------+-------------+---------------+-----------+------------+
4 rows in set (0.00 sec)


mysql> select user, references_priv, index_priv, alter_priv
    -> from user
    -> ;
+--------+-----------------+------------+------------+
| user   | references_priv | index_priv | alter_priv |
+--------+-----------------+------------+------------+
| root   | Y               | Y          | Y          |
| pma    | N               | N          | N          |
| vivien | Y               | Y          | Y          |
| ayyi   | N               | N          | N          |
+--------+-----------------+------------+------------+
4 rows in set (0.00 sec)

mysql> _
```

User vivien bertanda ” Y ” dapat mengakses semua
User ayyi bertanda ” N ” tidak dapat mengakses

Coba gunakan user ” ayyi ”
Keluar dari MySQL dan login kembali dengan user ” ayyi ” password ” fathin ”
Tampilan :

```
C:\Program Files\xampp\mysql\bin>mysql -u ayyi -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 5.0.24a-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Coba lihat database yang dapat diakses
Tampilan :

```
mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)
```

User ayyi tidak dapat mengakses database yang sudah terbentuk di mysql hanya database information_schema

Coba membuat database

```
mysql> create database latih1 ;
ERROR 1044 (42000): Access denied for user 'ayyi'@'%' to database 'latih1'
mysql>
```

Access denied, tidak diberikan izin untuk create

Memberikan Izin Akses tertentu

Jika akan memberikan izin akses SELECT, INSERT, UPDATE dan DELETE kepada user ayyi yang hanya dapat digunakan pada database latihdb1

Pemberian izin akses hanya boleh dilakukan oleh user dalam akses root atau user yang diberikan zin akses setingkat administrator.

Sebagai contoh user ”root” atau user ”iyus” (nama Anda sendiri)

Jika menggunakan root

Perintah :

```
C:\Program Files\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.24a-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Jika menggunakan user ”iyus” (nama Anda sendiri)

Perintah :

```
C:\Program Files\xampp\mysql\bin>mysql -u iyus -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 5.0.24a-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> _
```

Gunakan salah satu dan Buat database baru dengan nama ’latihdb1’

```
mysql> create database latihdb1 ;
Query OK, 1 row affected (0.00 sec)

mysql> _
```

Berikan izin akses ke database latihdb1

Aktifkan database mysql dan lihat tabel –

```
mysql> use mysql ;
Database changed
mysql> show tables ;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| proc                      |
| procs_priv                |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
17 rows in set (0.00 sec)
```

Berikan izin untuk insert, update, delete, create pada user ayyi

```
mysql> grant select, insert, update, delete, create
    -> on latihdb1.*
    -> to ayyi ;
Query OK, 0 rows affected (0.03 sec)
```

Lihat perubahan izin akses

```
mysql> select user, select_priv, update_priv, insert_priv,
    -> delete_priv, create_priv
    -> from db where user='ayyi'
    -> ;
+------+-------------+-------------+-------------+-------------+-------------+
| user | select_priv | update_priv | insert_priv | delete_priv | create_priv |
+------+-------------+-------------+-------------+-------------+-------------+
| ayyi | Y           | Y           | Y           | Y           | Y           |
+------+-------------+-------------+-------------+-------------+-------------+
1 row in set (0.00 sec)

mysql> _
```

Coba mengakses dengan user ayyi
Dan lihat database yang dapat diakses

```
C:\Program Files\xampp\mysql\bin>mysql -u ayyi -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 5.0.24a-com

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| latihdb1           |
+--------------------+
2 rows in set (0.00 sec)

mysql>
```

Database latihdb1 dapat diakses dan jika terdapat akses yang tidak diizinkan maka lakukan peritah GRANT untuk izin akses pada user ayyi, dengan ketetapan user admin sebagai root.

Contoh :

```
mysql> grant select, create, insert
    -> on latihdb1.*
    -> to ayyi ;
Query OK, 0 rows affected (0.04 sec)

mysql> _
```

Contoh user ayyi membuat table pegawai dalam database latihdb1

```
mysql> use latihdb1
Database changed
mysql> show tables ;
Empty set (0.00 sec)

mysql> create table pagawai
    -> (nip int unsigned auto_increment primary key,
    -> nama varchar(35) not null,
    -> gender varchar(2),
    -> alamat varchar(30),
    -> tgllahir date null default '0000-00-00')
    -> ;
Query OK, 0 rows affected (0.06 sec)

mysql> show tables ;
+--------------------+
| Tables_in_latihdb1 |
+--------------------+
| pagawai            |
+--------------------+
1 row in set (0.00 sec)

mysql> _
```

Buatlah database akademik dengan tabel mahasiswa, matakuliah dan kelas.
Struktur tabel mahasiswa :

```
mysql> create table mahasiswa
    -> (NIM varchar(9) not null primary key, Nama varchar(25),
    -> Tempatlahir varchar(15),
    -> Tgllahir date null default '0000-00-00',
    -> Gender varchar(1), Alamat varchar(30),
    -> Kota varchar(15),
    -> KdPos varchar(5)) ;
Query OK, 0 rows affected (0.29 sec)

mysql>
```

Struktur tabel Matakuliah :

```
mysql> create table Matakuliah
    -> (KodeMK varchar(7),
    -> NamaMK varchar(15),
    -> SKS numeric(1));
Query OK, 0 rows affected (0.09 sec)

mysql> _
```

Struktur tabel kelas :

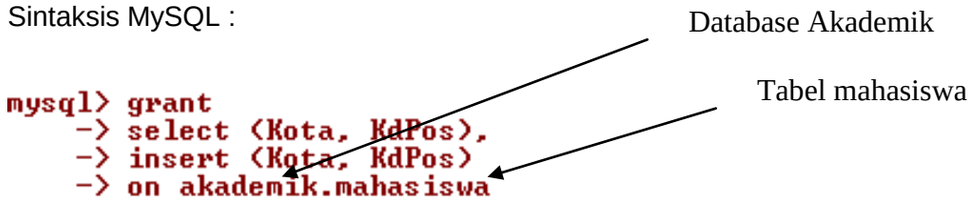
```
mysql> create table Kelas
    -> (KodeKelas varchar(5),
    -> Jurusan varchar(10),
    -> Fakultas varchar(10)) ;
Query OK, 0 rows affected (0.05 sec)
```

Memberikan izin akses per tabel dan per kolom

Dengan perintah grant dapat digunakan untuk memberikan izin akses per tabel dan per kolom tabel.

Contoh user ayyi diberikan izin akses SELECT dan INSERT untuk kolom kota dan kdpos pada tabel mahasiwa.

Sintaksis MySQL :




```
mysql> grant
    -> select (Kota, KdPos),
    -> insert (Kota, KdPos)
    -> on akademik.mahasiswa
    -> to ayyi
    -> identified by 'ai' ;
Query OK, 0 rows affected (0.08 sec)

mysql>
```

Pengaruh dari perintah grant tables, sebelumnya kita coba menampilkan tabel table_priv.
Sintaksis pertama :

```
mysql> select host, db, user, table_name, grantor, timestamp
    -> from tables_priv where user='ayyi' ;
+------+----------+------+------------+----------------+---------------------+
| host | db       | user | table_name | grantor        | timestamp           |
+------+----------+------+------------+----------------+---------------------+
| %    | akademik | ayyi | mahasiswa  | root@localhost | 2009-11-18 10:55:52 |
+------+----------+------+------------+----------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

Keterangan :

Host	: **%** , Dapat diakses di semua komputer terkoneksi jaringan
Db	: Database **Akademik** yang dapat diakses
User	: User **Ayyi**
Table_name	: **Mahasiswa** yang dapat diakses
Grantor	: **root@localhost**, yang memberikan izin akses
Timestamp	: Tanggal pemberian izin **18 November 2009 jam 10.55.52**

Sintaksis kedua :

```
mysql> select host, db, user, table_priv, column_priv
    -> from tables_priv where user='ayyi' ;
+------+----------+------+------------+---------------+
| host | db       | user | table_priv | column_priv   |
+------+----------+------+------------+---------------+
| %    | akademik | ayyi |            | Select,Insert |
+------+----------+------+------------+---------------+
1 row in set (0.00 sec)

mysql> _
```

Keterangan :

Column_priv	: Select dan Insert , izin akses kolom yang diberikan

Untuk melihat kolom mana saja yang diberikan izin akses select dan insert, dengan perintah :
Sintaksis MySQL

```
mysql> select * from columns_priv where user='ayyi' ;
+------+----------+------+------------+-------------+---------------------+--------------+
| Host | Db       | User | Table_name | Column_name | Timestamp           | Column_priv  |
+------+----------+------+------------+-------------+---------------------+--------------+
| %    | akademik | ayyi | mahasiswa  | Kota        | 2009-11-18 10:55:52 | Select,Insert |
| %    | akademik | ayyi | mahasiswa  | KdPos       | 2009-11-18 10:55:52 | Select,Insert |
+------+----------+------+------------+-------------+---------------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

Keterangan :
Column_name: Kota dan KdPos, kolom yang diberikan izin akses
Column_priv : Izin akses kedua kolom tersebut adalah select dan insert

Anda coba kewenangan apa saja yang dapat dilakukan oleh user ayyi.
Keluar dari mysql
Dan login kembali dengan user ayyi :

```
C:\Program Files\xampp\mysql\bin>mysql -u ayyi -h localhost -p
Enter password: **
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18 to server version: 5.0.24a-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Lakukan perintah :

```
mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| akademik           |
| latihdb1           |
+--------------------+
3 rows in set (0.00 sec)

mysql> use akademik
Database changed
mysql> show tables ;
+--------------------+
| Tables_in_akademik |
+--------------------+
| mahasiswa          |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

Lihat database aktif

Akatifkan database akademik

Lihat tabel aktif

Lakukan perintah select record yang ada pada tabel mahasiswa

```
mysql> select * from mahasiswa ;
ERROR 1143 (42000): SELECT command denied to user 'ayyi'@'localhost' for column 'NIM' in table 'maha
siswa'
```

User ayyi tidak diizikan untuk menampilkan data pada tabel mahasiswa
Izin akses nya adalah select untuk kolom kota dan kode pos
Lakukan perintah select kota dan kodepos

```
mysql> select kota, kdpos from mahasiswa ;
+---------+-------+
| kota    | kdpos |
+---------+-------+
| Jaktim  | 21485 |
| Bogor   | 48751 |
| Bandung | 45654 |
| Depok   | 16784 |
| Depok   | 16417 |
+---------+-------+
5 rows in set (0.00 sec)

mysql>
```

List data kota dan kode pos

Bagaimana dengan perintah delete tabel

```
mysql> delete from mahasiswa ;
ERROR 1142 (42000): DELETE command denied to user 'ayyi'@'localhost' for table 'mahasiswa'
mysql>
```

Perintah delete ditolak
Bagaimana dengan perintah UPDATE kota

```
mysql> update mahasiswa set kota='Bandung'
    -> where kota='Bogor'
    -> ;
ERROR 1142 (42000): UPDATE command denied to user 'ayyi'@'localhost' for table 'mahasiswa'
```

Bagimana dengan perintah insert

```
mysql> insert into mahasiswa
    -> (NIM, Nama, tempatlahir, Tgllahir, Gender, Alamat, Kota, Kdpos)
    -> values
    -> ('10207011','Budi Kusuma','Jakarta','1987-12-25','P','JL. Kecapi 83','Jakpus','11485')
    -> ;
ERROR 1143 (42000): INSERT command denied to user 'ayyi'@'localhost' for column 'NIM' in table 'maha
siswa'
mysql>
```

Untuk perintah update dan insert data tidak dapat dilakukan oleh user ayyi

Bagimana user ayyi dapat mengakses tabel mahasiswa
Keluar dari MySQL dan login kembali menggunakan user root
Dan berikan izin akses untuk tabel mahasiswa kepada user ayyi

```
mysql> grant all privileges
    -> on akademik.mahasiswa
    -> to ayyi ;
Query OK, 0 rows affected (0.14 sec)

mysql>
```

Perintah ini memberikan status root kepada user ayyi, hanya izin akses lengkap di database akademik tabel mahasiswa (on akademik.mahasiswa)

Coba keluar dan login kembali dengan user ayyi
Lakukan perintah update dan select

```
mysql> use akademik ;
Database changed
mysql> update mahasiswa set kota='Bandung'
    -> where kota='Bogor'
    -> ;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Perintah update dapat dilakukan

Lakukan perintah select data keseluruhan

```
mysql> select * from mahasiswa ;
+----------+-----------------+------------+------------+--------+-------------------+---------+-------+
| NIM      | Nama            | Tempatlahir | Tgllahir   | Gender | Alamat            | Kota    | KdPos |
+----------+-----------------+------------+------------+--------+-------------------+---------+-------+
| 10207010 | Tantri Kumala   | Jakarta    | 1985-12-21 | P      | Raden Saleh 83    | Jaktim  | 21485 |
| 20207002 | Iwan Eka Setio  | Bogor      | 1986-04-15 | L      | Kenanga no 101    | Bandung | 48751 |
| 30207003 | Puji Lestari    | Bandung    | 1986-07-10 | P      | Jl.Mawar no 31    | Bandung | 45654 |
| 40207004 | Fairuz Salsabil | Depok      | 1985-11-17 | L      | Jl, Kerinci No 11 | Depok   | 16784 |
| 50207005 | Fathin Qushayyi | Depok      | 1984-10-25 | L      | Jl.Kamboja No 111 | Depok   | 16417 |
+----------+-----------------+------------+------------+--------+-------------------+---------+-------+
5 rows in set (0.00 sec)
```

Bagaimana dengan menambah data gunakan perintah insert ?

```
mysql> insert into mahasiswa
    -> (NIM, Nama, tempatlahir, Tgllahir, Gender, Alamat, Kota, Kdpos)
    -> values
    -> ('10207011','Budi Kusuma','Jakarta','1987-12-25','P','JL. Kecapi 83','Jakpus','11485')
    -> ;
Query OK, 1 row affected (0.00 sec)

mysql> select * from mahasiswa ;
+----------+-----------------+------------+------------+--------+-------------------+---------+-------+
| NIM      | Nama            | Tempatlahir | Tgllahir   | Gender | Alamat            | Kota    | KdPos |
+----------+-----------------+------------+------------+--------+-------------------+---------+-------+
| 10207010 | Tantri Kumala   | Jakarta    | 1985-12-21 | P      | Raden Saleh 83    | Jaktim  | 21485 |
| 20207002 | Iwan Eka Setio  | Bogor      | 1986-04-15 | L      | Kenanga no 101    | Bandung | 48751 |
| 30207003 | Puji Lestari    | Bandung    | 1986-07-10 | P      | Jl.Mawar no 31    | Bandung | 45654 |
| 40207004 | Fairuz Salsabil | Depok      | 1985-11-17 | L      | Jl, Kerinci No 11 | Depok   | 16784 |
| 50207005 | Fathin Qushayyi | Depok      | 1984-10-25 | L      | Jl.Kamboja No 111 | Depok   | 16417 |
| 10207011 | Budi Kusuma     | Jakarta    | 1987-12-25 | P      | JL. Kecapi 83     | Jakpus  | 11485 |
+----------+-----------------+------------+------------+--------+-------------------+---------+-------+
6 rows in set (0.00 sec)
```

Perintan insert dapat dilakukan dan jumlah data terdiri 6 record

Bagaimana dengan select untuk tabel mata kuliah dan kelas ?

```
mysql> select * from matakuliah ;
ERROR 1142 (42000): SELECT command denied to user 'ayyi'@'localhost' for table 'matakuliah'
mysql> select * from kelas ;
ERROR 1142 (42000): SELECT command denied to user 'ayyi'@'localhost' for table 'kelas'
mysql>
```

Untuk mengakses tabel matakuliah dan kelas tidak di izinkan

Memberikan izin akses berdasarkan lokasi pengakses

Admin atau root atau memberikan izin akses berdasarkan lokasi atau membatasi komputer mana saja yang dapat mengakses MySQL server.
Contoh pemberian izin akses :

```
mysql> grant all privileges
    -> on akademik.mahasiswa
    -> to ayyi@localhost ;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> grant all privileges
    -> on akademik.mahasiswa
    -> to ayyi@127.0.0.1 ;
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges
    -> on akademik.mahasiswa
    -> to ayyi@'%' ;
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges
    -> on akademik.mahasiswa
    -> to ayyi@'www.dimanasaja.com' ;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Perlu diperhatikan format pemberian izin, berhubungan dengan cara login ke MySQL, contoh : Jika pemberian izin akse dengan no IP komputer maka login juga harus mencantumkan no IP

Yang lebih fleksibel menggunakan tanda ”%” dapat diakses dimana saja ke server MySQL.

Menghapus izin akses

User ayyi dapat mengakses tabel yang ada di database akademik

```
mysql> use akademik ;
Database changed
mysql> show tables ;
+--------------------+
| Tables_in_akademik |
+--------------------+
| kelas              |
| mahasiswa          |
| matakuliah         |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

Sebagai contoh : user ayyi dihapus izin akses untuk database akademik.

Sintaksis :

```
mysql> revoke all privileges
    -> on akademik.*
    -> from ayyi ;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Jangan lupa diakhir dengan perintah flush

```
mysql> flush privileges
    -> ;
Query OK, 0 rows affected (0.20 sec)

mysql>
```

Kasus jika suatu root telah memberikan izin akses hanya untuk satu tabel, seperti user ayyi telah dahulu diberikan all privileges kepada tabel mahasiswa, maka untuk menghapusnya disesuaikan dengan perintah grant.

```
mysql> use akademik ;
Database changed
mysql> show tables ;
+------------------+
| Tables_in_akademik |
+------------------+
| mahasiswa          |
+--------------------+
1 row in set (0.00 sec)
```

Untuk menghapus izin akses di tabel mahasiswa, lakukan perintah (izin aksesnya sama dengan perintah grant)

```
mysql> revoke all privileges
    -> on akademik.mahasiswa
    -> from ayyi ;
Query OK, 0 rows affected (0.01 sec)

mysql> flush privileges ;
Query OK, 0 rows affected (0.01 sec)

mysql> _
```

4. LATIHAN

Membuat User Baru

- Buatlah dua User Baru dengan menggunakan perintah GRANT dengan ketentuan sebagai berikut:

  User pertama adalah Nama_Anda dimana yaitu user tersebut sebagai administrator ON *.* = dapat meng-akses semua database TO Nama_Anda dapat ditulis TO Nama_Anda@% atau TO Nama_Anda@localhost

  User kedua adalah Nama_Teman_Anda yang dapat login ke MySQL dan dapat mengakses semua database tetapi tidak memiliki izin akses, jadi sebagai dummy user atau blank user

- Periksalah tabel user yaitu kolom privileges dengan perintah SELECT * FROM USER. User Nama_Anda bertanda ” Y ” dapat mengakses semua dan User Nama_Teman_Anda bertanda ” N ” tidak dapat mengakses
- Keluar dari MySQL dan login kembali dengan user Nama_Teman_Anda. Coba perhatikan database yang dapat diakses. User tersebut tidak dapat mengakses database yang sudah terbentuk di mysql, hanya database information_schema
- Coba membuat database menggunakan user Nama_Teman_anda, maka akan muncul pesan Access denied, tidak diberikan izin untuk create.

Memberikan Izin Akses tertentu

- Dengan perintah GRANT berikan izin untuk insert, update, delete, create pada user Nama_Teman_Anda
- Lihat perubahan izin akses
- Coba mengakses dengan user Nama_Teman_Anda dan lihat database yang dapat diakses. Database RumahSakit dapat diakses dan jika terdapat akses yang tidak diizinkan maka lakukan peritah GRANT untuk izin akses pada user Nama_Teman_Anda, dengan ketetapan user admin sebagai root.
- Dengan menggunakan user Nama_Teman_Anda, buatlah tabel dalam database RumahSakit.

Memberikan izin akses per tabel dan per kolom

- Dengan perintah GRANT, berikanlah izin akses SELECT dan INSERT untuk kolom Nama dan NOTelepon pada tabel DOKTER.
- Tampilkan tabel table_priv

Menghapus izin akses

- Dengan perintah REVOKE, hapuslah izin akses database RumahSakit untuk User Nama_Teman_Anda
- Akhiri dengan perintah FLUSH

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 9

Pertemuan ke-10

Simple Query

1. TUJUAN

Mahasiswa mampu menggunakan klausa SELECT, klausa WHERE, klausa FROM, operasi penamaan ulang (AS), operasi alias, operator relasional, operator AND, OR dan NOT

2. TEORI SINGKAT

SELECT merupakan instruksi yang popular digunakan dalam SQL. Instruksi ini berfungsi untuk memilih spesifik kolom dari satu atau beberapa tabel.

Secara umum instruksi SELECT adalah sebagai berikut :
SELECT kolom1, kolom2, kolom-n,…
FROM nama_tabel
WHERE predikat/kondisi

Jika klausa WHERE tidak digunakan atau diberikan, maka record atau data yang diseleksi adalah seluruh Data dalam tabel. Predikat atau kondisi yang diberikan setelah klausa WHERE menyatakan kualifikasi dari record yang harus ditemukan, apabila memenuhi syarat, maka record tersebut akan dipilih dan ditampilkan. Artinya, bahwa dengan menggunakan klausa WHERE, maka seleksi yang dilakukan bukan pada seluruh record, melainkan hanya pada record yang memenuhi syarat.

Bentuk umum klausa WHERE adalah sebagai berikut :
WHERE kolom <operator><nilai>

Beberapa operator yang berlaku meliputi :

No	Operator	Arti
1	=	Sama dengan
2	<>	Tidak sama, atau dapat juga !=
3	<	Kurang dari
4	<=	Kurang dari sama dengan
5	>	Lebih dari
6	>=	Lebih dari sama dengan

Logika AND, OR dan NOT
Selain menggunakan operator di atas, klausa WHERE juga dapat digabungkan dengan menggunakan logika AND, OR, dan NOT. Hal ini bertujuan untuk menggabungkan lebih dari satu kondisi maupun negasi.

SELECT * merupakan karakter khusus yang menyatakan bahwa kolom yang akan dipilih adalah seluruh kolom yang terdapat pada tabel tersebut.

a. Memberi nama lain pada kolom
   SELECT nama_kolom_lama AS nama_kolom_baru FROM namatabel
   Contoh:
   SELECT salary AS Gaji from dosen;

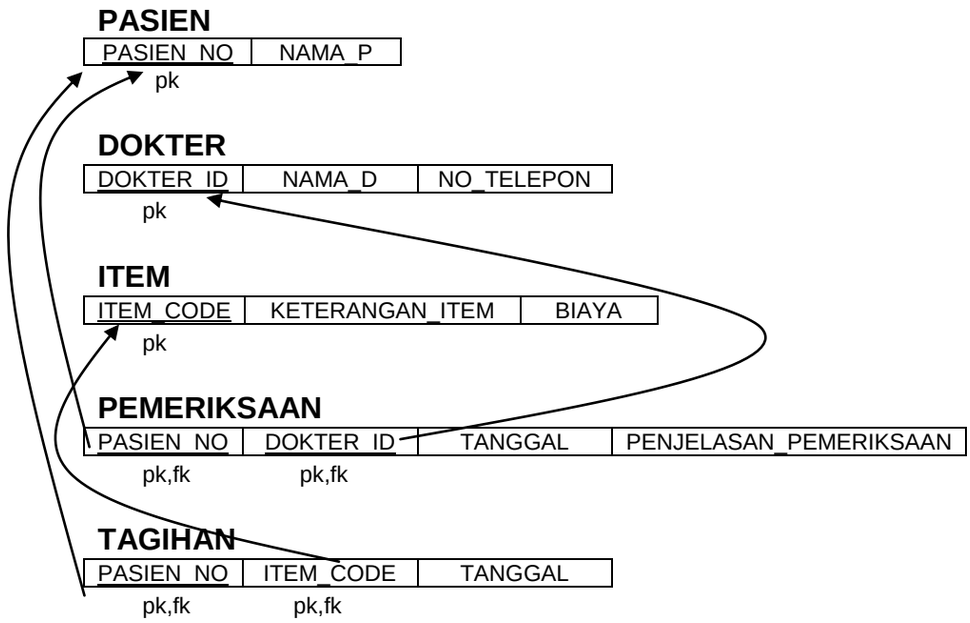
b. Menggunakan alias untuk nama tabel
SELECT nama_alias .namakolom1, nama_alias .namakolom2 FROM namatabel nama_alias;

Contoh:
SELECT m.nama_mhs, m.alamat FROM mahasiswa m;

c. Menampilkan data lebih dari dua tabel
SELECT * FROM nama_tabel1, nama_tabel2, nama_tabel-n;
Contoh : SELECT * FROM ruang, prodi;

3. PELAKSANAAN PRAKTIKUM

a. Implementasikanlah basis data Rumah Sakit berikut:



Gambar A. **Skema Basis Data RUMAH SAKIT**

PASIEN

PASIEN_NO	NAMA_P
3249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Gambar B. **Basis Data RUMAH SAKIT**

b. Tampilkan data pasien yang nomornya 3249

```
select *
from   pasien
Where pasien_no = '3249';
```

c. Tampilkan nomor dan nama pasien yg pernah melakukan kontrol rutin!

```
Select  Pasien.pasien_no, Pasien.nama_p
From    Pasien, Pemeriksaan
Where Pasien.pasien_no = Pemeriksaan.pasien_no and
        Pemeriksaan.penjelasan_pemeriksaan = 'Kontrol Rutin';
```

ATAU

```
Select  P.pasien_no, P.nama_p
From    Pasien P, Pemeriksaan R
Where P.pasien_no = R.pasien_no and
        R.penjelasan_pemeriksaan = 'Kontrol Rutin';
```

d. Tampilkan nama pasien yg pernah diperiksa oleh dokter Wolcox!

```
Select  Pasien.nama_p
From    Pasien, Pemeriksaan, Dokter
Where Pasien.pasien_no = Pemeriksaan.pasien_no and
        Dokter.dokter_id = Pemeriksaan.dokter_id and
        Dokter.nama_d = 'Wilcox';
```

ATAU

```
Select  P.nama_p
From    Pasien P, Pemeriksaan R, Dokter D
Where P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        D.nama_d = 'Wilcox';
```

e. Tampilkan nama dokter (kolom nama_d ditulis NAMA_DOKTER) yg no teleponnya 329-1848

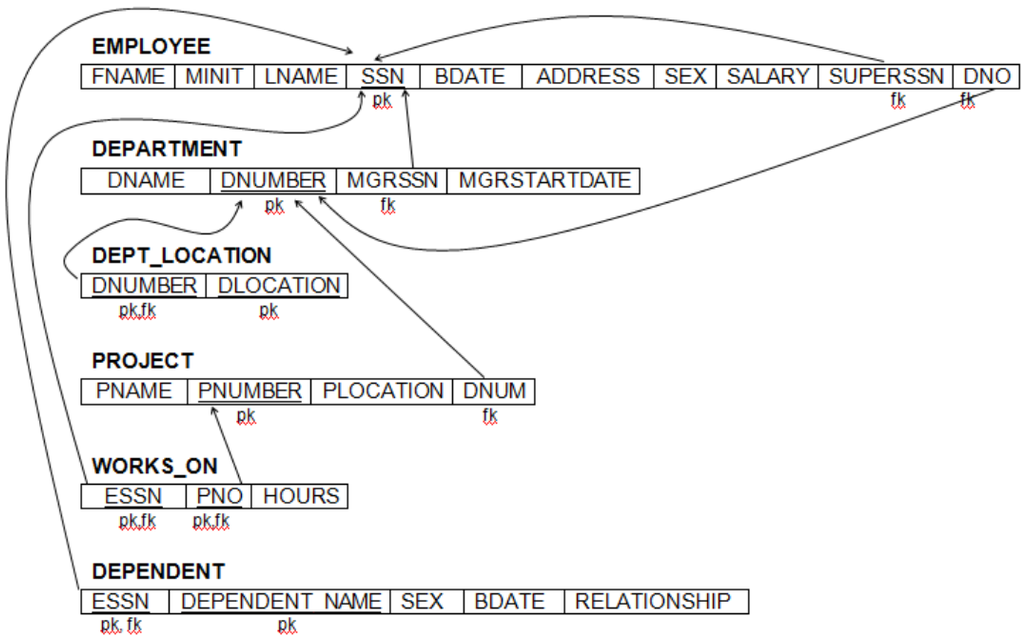
```
Select nama_d as NAMA_DOKTER
From dokter
Where no_telp = '329-1848';
```

4. LATIHAN

1. Tampilkan nama pasien untuk pasien_no = 2298 atau 4139
2. Tampilkan nama dokter yang pernah melakukan observasi pra bedah pada pasiennya
3. Tampilkan nama pasien yang pernah di Obsevasi pasca bedah oleh dokter Margaret atau dokter Nusca pada tanggal 15/11/06
4. Tampilkan penjelasan pemeriksaan apa saja yang pernah dijalani oleh pasien secara unik, tampilkan nama kolom penjelasan_pemeriksaan = KETERANGAN_PEMERIKSAAN dan urutkan berdasarkan abjad
5. Berikan uraian berupa nama pasien, tanggal pemeriksaan, dan penjelasan pemeriksaan untuk pasien yang diperiksa tanggal 22/11/06 untuk Kontrol Rutin
6. Berikan nama-nama dokter yang pernah memeriksa pasien bernama Steve

5. TUGAS

Implementasikan basis data COMPANY dan buatlah query dengan beberapa kriteria.



Gambar 1. **Skema Basis Data COMPANY**

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	09-jan-55	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	08-dec-45	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-jul-58	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-jun-31	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-sep-52	975 Are Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-jul-62	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabber	987987987	29-mar-59	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	10-nov-27	450 Stone, Houston, TX	M	55000	null	1

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-may-78
Administration	4	987654321	01-jan-85
Headquarters	1	888665555	19-jan-71

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
Produce	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	05-apr-76	Daughter
333445555	Theodore	M	25-oct-73	Son
333445555	Joy	F	03-may-46	Spouse
987654321	Abner	M	29-feb-32	Spouse
123456789	Michael	M	01-jan-78	Son
123456789	Alice	F	31-dec-78	Daughter
123456789	Elizabeth	F	05-may-57	Spouse

DEPT_LOCATION

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Gambar 2. **Basis Data COMPANY**

Pertemuan ke-11

Operator MySQL

1. TUJUAN

Mahasiswa mampu menggunakan operator relasional, operator like, distinct, between, dan operasi pengurutan (order by)

2. TEORI SINGKAT

OPERATOR RELASIONAL
Operator yang digunakan yntuk perbandingan antara dua buah nilai.
Jenis dari operator ini adalah = , >, <, >=, <=, <>
Contoh:
select * from mhs where alamat=’sleman’;select * from mhs where jml_saudara > 5;

Operator LIKE
Operator LIKE atau NOT LIKE sangat bermanfaat dalam mencari suatu data.Operasi ini digunakan dengan menyebutkan tanda wildcard berupa garis bawah (_) atau (%). Tanda garis bawah (_) berarti sebuah karakter apa saja.

Contoh a_ucocok dengan anu, aku, alu, abu dan tidak cocok untuk andu, ambu ataupun allu. Tanda % berarti cocok dengan kata apa saja dan berapapun panjangnya

Contoh:
select nim,nama,alamat from mhs where nama like 'a%';

Operator DISTINCT
Operator DISTINCT digunakan untuk menghilangkan duplikasi.

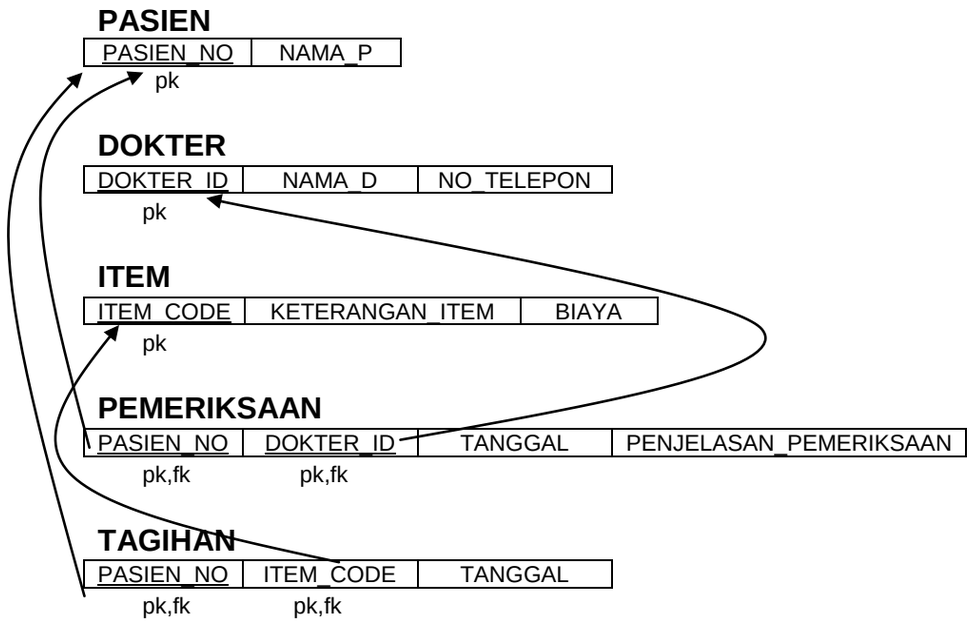
Contoh:
select distinct(nama) from mhs;

Operator BETWEEN dan NOT BETWEEN
Operator between ini untuk menangani operasi “jangkauan”
Contoh:
select * from mhs where tgl_lhr between '1980-01-01' and '1982-12-29';

Operator ORDER BY
Operator ORDER BY digunakan untuk mengurutkan table hasil query berdasarkan satu atau beberapa atribut

Contoh:
select * from mhs oeder by nama

3. PELAKSANAAN PRAKTIKUM



Gambar A. **Skema Basis Data RUMAH SAKIT**

PASIEN

PASIEN_NO	NAMA_P
3249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Gambar B. **Basis Data RUMAH SAKIT**

1. Implementasikanlah basis data Rumah Sakit diatas.
2. Tampilkanlah nama-nama item yang biayanya lebih besar dari Rp 150.000

```
Select  keterangan_item
From    item
Where biaya > 150000;
```

3. Tampilkan nama-nama item yang yg biayanya antara Rp100.000 sampai dengan Rp200.000!

```
Select  keterangan_item
From    item
Where biaya between 100000 and 200000;
```

4. Tampilkan nama dokter secara unik, urutkan berdasarkan nama dokter!

```
Select      distinct (nama_d)
From        dokter
Order By    nama_d;
```

5. Tampilkan semua nama pasien yang berawalan huruf M!

```
Select      *
From        pasien
Order By    nama_p like ‘M%’;
```

4. LATIHAN

Berdasarkan skema Basis Data RUMAH SAKIT yang ada pada Gambar A, buatlah perintah dengan menggunakan operator MySQL untuk kasus berikut:

1. Tampilkan tabel dokter yang diurutkan berdasarkan nama dokter
2. Tampilkan item yang biayanya tidak berada diantara Rp50000 dan Rp100000
3. Tampilkan nama dan nomor telepon dokter yang namanya diawali huruf M dan nomor teleponnya ada angka 1, urutkan berdasarkan namanya
4. Berikan nama-nama dokter yang pernah memeriksa pasien bernama David
5. Tampilkan penjelasan pemeriksaan secara unik
6. Tampilkan penjelasan pemeriksaan apa saja yang pernah dijalani oleh pasien secara unik dan urutkan berdasarkan abjad
7. Tampilkan nama – nama item dan biayanya, hanya untuk item yang digunakan oleh pasien bernama Mary yang biayanya antara Rp 20.000 sampai dengan Rp 250.000, urutkan dari biaya yang tertinggi.
8. Carilah nama dokter yang huruf ke-2 namanya adalah huruf a dan nomor teleponnya ada angka 1 atau angka 8

5. TUGAS

Sesuai GBPP **ada** tugas pada pertemuan 11

Pertemuan ke-12

Join Query

1. TUJUAN

Mahasiswa mampu memahami query multi table dengan berbagai kriteria menggunakan perintah VIEW, subquery, join query, inner join, dan outer join.

2. TEORI SINGKAT

JOIN QUERY

Untuk menggabungkan tabel pada MySQL, kita gunakan klausa JOIN. Pada MySQL terdapat dua macam bentuk join, yaitu INNER JOIN, LEFT OUTER JOIN, dan RIGHT OUTER JOIN. Format penulisannya adalah sebagai berikut:

```
SELECT nama_kolom
FROM tabel
INNER JOIN | LEFT OUTER JOIN | RIGHT OUTER JOIN tabel ON kondisi
```

Selain menggunakan klausa ON untuk mendefinisikan kondisi, kita dapat menggunakan klausa USING, format penulisannya adalah:

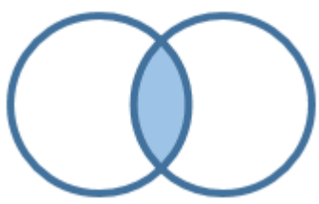
```
SELECT nama_kolom
FROM tabel
INNER JOIN | LEFT OUTER JOIN | RIGHT OUTER JOIN tabel USING(nama_kolom)
```

Klausa USING ini akan menggunakan nama kolom yang ada di dalam tanda kurung untuk menghubungkan kedua tabel, kolom ini harus ada pada tabel yang ingin dihubungkan dan harus memiliki nama yang sama.

Biasanya kolom yang berhubungan didefinisikan sebagai Primary Keydan Foreign Key, namun tidak masalah jika keduanya bukan primary key maupun foreign key.

INNER atau CROSS JOIN

Cara pertama untuk menggabungkan tabel adalah menggunakan inner join. Dengan inner join, tabel akan digabungkan berdasarkan data yang sama, yang ada pada kedua tabel, jika di gambarkan dalam bentuk diagram venn, bentuk inner join seperti tampak pada gambar berikut:



Pada MySQL, penulisan INNER JOIN dapat dilakukan dengan dua cara yaitu (1) menggunakan klausa INNER JOIN (2) menggunakan klausa CROSS JOIN (3)

cukup menggunakan klausa JOIN saja. Kita bebas menggunakan keduanya asal konsisten, supaya lebih simpel, gunakan JOIN saja.

OUTER JOIN

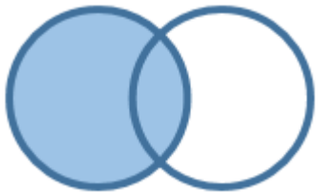
Cara kedua untuk menggabungkan tabel pada MySQL adalah menggunakan outer join. Pada outer join, data pada salah satu tabel akan ditampilkan semua, sedangkan data pada tabel yang lain hanya akan ditampilkan jika data tersebut ada pada tabel pertama.

Pada MySQL, OUTER JOIN dibagi menjadi dua, yaitu LEFT OUTER JOIN dan RIGHT OUTER JOIN.

LEFT OUTER JOIN

Pada LEFT OUTER JOIN, semua data pada tabel sebelah kiri akan ditampilkan, sedangkan data pada tabel disebelah kanan hanya akan ditampilkan jika data terkait pada tabel tersebut muncul di tabel sebelah kiri.

Jika di gambarkan dalam bentuk diagram venn, bentuk LEFT OUTER JOIN tampak pada gambar berikut:

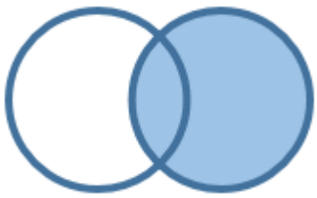


LEFT OUTER JOIN dapat ditulis menggunakan dua cara, yaitu (1) dengan klausa LEFT OUTER JOIN, (2) cukup dengan klausa LEFT JOIN saja, kita bebas memilih salah satu yang penting konsisten, saya sendiri prefer menggunakan bentuk kedua karena lebih simpel.

RIGHT OUTER JOIN

Kebalikan dari LEFT OUTER JOIN, pada RIGHT OUTER JOIN, data pada tabel sebelah kanan akan ditampilkan semua, sedangkan data pada sebelah kiri hanya ditampilkan jika data terkait pada tabel tersebut muncul pada tabel sebelah kanan.

Jika digambarkan dalam bentuk diagram venn, maka, bentuk right outer join akan tampak seperti gambar berikut:



Sama seperti LEFT OUTER JOIN, RIGHT OUTER JOIN juga dapat ditulis menggunakan dua cara, yaitu ditulis secara utuh atau cukup RIGHT JOINsaja. Saya sendiri lebih prefer menggunakan RIGHT JOIN saja.

Pada contoh diatas, terlihat bahwa semua data pada tabel disebelah kanan, yaitu tabel penjualan akan ditampilkan semua, sedangkan pada tabel sebelah kiri hanya ditampilkan yang data id_pelanggan nya muncul pada tabel penjualan.

Perlukah RIGHT JOIN?

Jika kita teliti lebih lanjut, sebenarnya right join hanya memindah posisi tabel, dari kiri ke kanan, contoh query pada right join dapat kita ubah dengan menjadi LEFT JOIN dengan mengubah posisi tabel

IMPLISIT JOIN

Sejauh ini, kita menampilkan data dari beberapa tabel MySQL dengan menggunakan klausa JOIN.

Selain menggunakan klausa JOIN, terdapat satu cara lagi untuk menggabungkan tabel MySQL, yaitu menggunakan implisit join, disebut implisit join karena kita tidak menggunakan klausa JOIN, pada implisit join, kriteria hubungan antar tabel di definisikan pada klausa WHERE.

Implisit join mensyaratkan kedua tabel memiliki data yang sama (WHERE pl.id_pelanggan = pn.id_pelanggan), sehingga implisit join ini hanya berlaku pada INNER JOIN, dan tidak bisa digunakan untuk OUTER JOIN.

Implisit JOIN ini merupakan cara lama ketika pertama kali standar SQL dibuat, setelah muncul standar yang lebih baru (SQL2) maka mulai digunakanlah klausa JOIN. Saya sendiri prefer menggunakan klausa JOIN karena lebih mudah dibaca dan dipahami, terutama hubungan antara tabel yang digabungkan.

Pada bentuk klausa JOIN, hubungan antar tabel dinyatakan pada klausa ON atau USING, sedangkan filter datanya dilakukan pada klausa WHERE

VIEW

View adalah perintah query yang disimpan pada database dengan suatu nama tertentu, sehingga bisa digunakan setiap saat untuk melihat data tanpa menuliskan ulang query tersebut.

Syntax dasar perintah untuk membuat view adalah sebagai berikut :

```
CREATE
    [OR REPLACE]
    VIEW view_name [(column_list)]
    AS select_statement
```

Kita menggunakan opsi OR REPLACE jika kita ingin mengganti view dengan nama yang sama dengan perintah tersebut. Jika tidak maka perintah CREATE VIEW akan menghasilkan error jika nama view yang ingin dibuat sudah ada sebelumnya.

Sintax dasar perintah untuk menampilkan daftar view adalah sebagai berikut :

```
SHOW FULL TABLES IN hr WHERE TABLE_TYPE LIKE 'VIEW';
```

Sintax dasar perintah untuk mengupdate view adalah sebagai berikut :

```
CREATE OR REPLACE VIEW nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_tabel
WHERE kondisi;
```

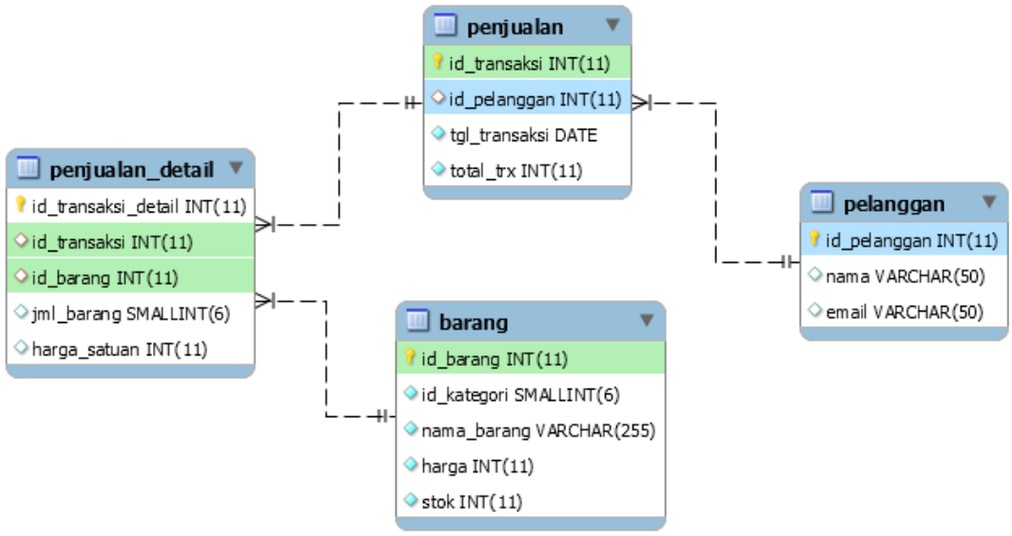
Sintax dasar perintah untuk menghapus view adalah sebagai berikut :

```
DROP VIEW nama_view;
```

3. PELAKSANAAN PRAKTIKUM

JOIN QUERY

Untuk mempraktekkan join query kita akan gunakan empat tabel, yaitu: tabel pelanggan, produk, transaksi, dan transaksi_detail. Buatlah keempat tabel tersebut dan lengkapi tabel dengan isinya. Struktur dan hubungan keempat tabel tersebut tampak seperti pada gambar berikut:



Dari keempat tabel tersebut, kita hanya menggunakan beberapa diantaranya. Adapun datanya adalah sebagai berikut:

Tabel barang:

```
+-----------+-------------+-------------+---------+------+
| id_barang | id_kategori | nama_barang | harga   | stok |
+-----------+-------------+-------------+---------+------+
|         1 |           1 | RAM         |  230000 |    4 |
|         2 |           1 | Mainboard   | 1250000 |    7 |
|         3 |           1 | Mouse       |   80000 |    6 |
|         4 |           3 | Mousepad    |   35000 |    3 |
|         5 |           3 | Keyboard    |   80000 |    5 |
+-----------+-------------+-------------+---------+------+
```

Tabel pelanggan:

```
+--------------+---------+-------------------+
| id_pelanggan | nama    | email             |
+--------------+---------+-------------------+
|            1 | Alfa    | alfa@yahoo.com    |
|            2 | Beta    | beta@yahoo.com    |
|            3 | Charlie | charlie@gmail.com |
|            4 | Delta   | delta@gmail.com   |
+--------------+---------+-------------------+
```

Tabel penjualan:

```
+--------------+--------------+---------------+-----------------+
| id_transaksi | id_pelanggan | tgl_transaksi | total_transaksi |
+--------------+--------------+---------------+-----------------+
|            1 |            1 | 2017-02-22    |          230000 |
|            2 |            3 | 2017-02-22    |          195000 |
|            3 |            2 | 2017-01-01    |         1710000 |
|            4 |            1 | 2017-02-04    |          310000 |
|            5 |         NULL | 2017-02-10    |           80000 |
+--------------+--------------+---------------+-----------------+
```

INNER atau CROSS JOIN

Untuk menampilkan data pelanggan yang melakukan pesanan, jalankan query sebagai berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
JOIN penjualan pn ON pl.id_pelanggan = pn.id_pelanggan
```

Jika menggunakan klausa USING, maka query akan berbentuk seperti berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
JOIN penjualan pn USING(id_pelanggan)
```

Hasil:

```
+--------------+---------+---------------+-----------------+
| id_pelanggan | nama    | tgl_transaksi | total_transaksi |
+--------------+---------+---------------+-----------------+
|            1 | Alfa    | 2017-02-22    |          230000 |
|            3 | Charlie | 2017-02-22    |          195000 |
|            2 | Beta    | 2017-01-01    |         1710000 |
|            1 | Alfa    | 2017-02-04    |          310000 |
+--------------+---------+---------------+-----------------+
```

Pada contoh diatas, terlihat bahwa

- Pelanggan dengan nama Delta tidak muncul, hal ini disebabkan pelanggan tersebut tidak pernah melakukan transaksi.
- Transaksi dengan id_trx 5 tidak muncul, karena transaksi tersebut memiliki nilai id_transaksi NULL, sehingga tidak terhubung ke pelanggan manapun.

LEFT OUTER JOIN

Untuk menampilkan semua data pelanggan beserta data transaksinya, jalankan query berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
LEFT JOIN penjualan USING(id_pelanggan)
```

Hasil:

```
+--------------+---------+---------------+-----------------+
| id_pelanggan | nama    | tgl_transaksi | total_transaksi |
+--------------+---------+---------------+-----------------+
|            1 | Alfa    | 2017-02-22    |          230000 |
|            3 | Charlie | 2017-02-22    |          195000 |
|            2 | Beta    | 2017-01-01    |         1710000 |
|            1 | Alfa    | 2017-02-04    |          310000 |
|            4 | Delta   | NULL          |            NULL |
+--------------+---------+---------------+-----------------+
```

Pada contoh diatas, terlihat bahwa dengan LEFT JOIN, data pada tabel sebelah kiri, yaitu tabel pelanggan akan ditampilkan semua, sedangkan data pada tabel sebelah kanan hanya akan ditampilkan jika nilai kolom id_pelanggan nya muncul pada tabel pertama, yaitu id_pelanggan 1, 2, dan 3

RIGHT OUTER JOIN

Untuk menampilkan semua data transaksi beserta data pelanggannya, jalankan query berikut:

```
SELECT pl.id_pelanggan, nama, tgl_transaksi, total_transaksi
FROM pelanggan pl
RIGHT JOIN penjualan USING(id_pelanggan)
```

Hasil:

```
+--------------+---------+--------------+---------------+-----------------+
| id_pelanggan | nama    | id_transaksi | tgl_transaksi | total_transaksi |
+--------------+---------+--------------+---------------+-----------------+
|            1 | Alfa    |            1 | 2017-02-22    |          230000 |
|            1 | Alfa    |            4 | 2017-02-04    |          310000 |
|            2 | Beta    |            3 | 2017-01-01    |         1710000 |
|            3 | Charlie |            2 | 2017-02-22    |          195000 |
|         NULL | NULL    |            5 | 2017-02-10    |           80000 |
+--------------+---------+--------------+---------------+-----------------+
```

Pada contoh diatas, terlihat bahwa semua data pada tabel disebelah kanan, yaitu tabel penjualan akan ditampilkan semua, sedangkan pada tabel sebelah kiri hanya ditampilkan yang data id_pelanggan nya muncul pada tabel penjualan.

Query pada right join dapat kita ubah dengan menjadi LEFT JOIN dengan mengubah posisi tabel, praktekkanlah contoh berikut:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi, total_transaksi
FROM penjualan pl
LEFT JOIN pelanggan USING(id_pelanggan)
```

Perhatikan pada contoh diatas, kita balik posisi tabel penjualan dan tabel pelanggan. Jika query terebut dijalankan, hasil yang kita peroleh adalah:

```
+--------------+---------+--------------+---------------+-----------------+
| id_pelanggan | nama    | id_transaksi | tgl_transaksi | total_transaksi |
+--------------+---------+--------------+---------------+-----------------+
|            1 | Alfa    |            1 | 2017-02-22    |          230000 |
|            1 | Alfa    |            4 | 2017-02-04    |          310000 |
|            2 | Beta    |            3 | 2017-01-01    |         1710000 |
|            3 | Charlie |            2 | 2017-02-22    |          195000 |
|         NULL | NULL    |            5 | 2017-02-10    |           80000 |
+--------------+---------+--------------+---------------+-----------------+
```

Perhatikan bahwa hasil diatas sama persis dengan hasil pada contoh right join, jadi kesimpulannya, agar memudahkan, cukup gunakan salah satu bentuk outer join saja, LEFT JOIN atau RIGHT JOIN, saya sendiri prefer menggunakan LEFT JOIN.

IMPLISIT JOIN

Gabungan tabel pelanggan dan penjualan, jalankan query berikut:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi, total_transaksi
FROM pelanggan pl, penjualan pn
WHERE pl.id_pelanggan = pn.id_pelanggan
```

Hasil yang diperoleh:

```
+--------------+---------+--------------+---------------+-----------------+
| id_pelanggan | nama    | id_transaksi | tgl_transaksi | total_transaksi |
+--------------+---------+--------------+---------------+-----------------+
|            1 | Alfa    |            1 | 2017-02-22    |          230000 |
|            3 | Charlie |            2 | 2017-02-22    |          195000 |
|            2 | Beta    |            3 | 2017-01-01    |         1710000 |
|            1 | Alfa    |            4 | 2017-02-04    |          310000 |
+--------------+---------+--------------+---------------+-----------------+
```

Perhatikan bahwa hasil tersebut sama persis dengan hasil pada contoh INNER JOIN, sehingga dapat disimpulkan bahwa implisit join = inner join.

Implisit join mensyaratkan kedua tabel memiliki data yang sama (WHERE pl.id_pelanggan = pn.id_pelanggan), sehingga implisit join ini hanya berlaku pada INNER JOIN, dan tidak bisa digunakan untuk OUTER JOIN.

Untuk melihat hubungan antar tabel dinyatakan pada klausa ON atau USING, sedangkan filter datanya dilakukan pada klausa WHERE, yaitu:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi, total_transaksi
FROM pelanggan pl
LEFT JOIN penjualan pn USING (id_pelanggan)
WHERE pl.id_pelanggan = 2 OR pl.id_pelanggan = 1
```

sedangkan pada implisit JOIN, hubungan antar tabel dan filter datanya, semua didefinisikan pada klausa WHERE, misal:

```
SELECT pl.id_pelanggan, nama, id_transaksi, tgl_transaksi, total_transaksi
FROM pelanggan pl, penjualan pn
WHERE pl.id_pelanggan = pn.id_pelanggan
        AND (pl.id_pelanggan = 2 OR pl.id_pelanggan = 1)
```

VIEW

View adalah perintah query yang disimpan pada database dengan suatu nama tertentu, sehingga bisa digunakan setiap saat untuk melihat data tanpa menuliskan ulang query tersebut.

Syntax dasar perintah untuk membuat view adalah sebagai berikut :

```
CREATE
    [OR REPLACE]
    VIEW view_name [(column_list)]
    AS select_statement
```

Kita menggunakan opsi OR REPLACE jika kita ingin mengganti view dengan nama yang sama dengan perintah tersebut. Jika tidak maka perintah CREATE VIEW akan menghasilkan error jika nama view yang ingin dibuat sudah ada sebelumnya.

Sintax dasar perintah untuk menampilkan daftar view adalah sebagai berikut :

```
SHOW FULL TABLES IN hr WHERE TABLE_TYPE LIKE 'VIEW';
```

Sintax dasar perintah untuk mengupdate view adalah sebagai berikut :

```
CREATE OR REPLACE VIEW nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_tabel
WHERE kondisi;
```

```
+-----------+-------------+-------------+---------+------+
| id_barang | id_kategori | nama_barang | harga   | stok |
+-----------+-------------+-------------+---------+------+
|         1 |           1 | RAM         |  230000 |    4 |
|         2 |           1 | Mainboard   | 1250000 |    7 |
|         3 |           1 | Mouse       |   80000 |    6 |
|         4 |           3 | Mousepad    |   35000 |    3 |
|         5 |           3 | Keyboard    |   80000 |    5 |
+-----------+-------------+-------------+---------+------+
```

Berdasarkan tabel BARANG diatas, akan dibuat VIEW dengan nama view_barang sebagai berikut:

```
CREATE VIEW view_barang_1 AS
SELECT id_barang, nama_barang, harga, stok
FROM barang
WHERE id_kategori = 1;
```

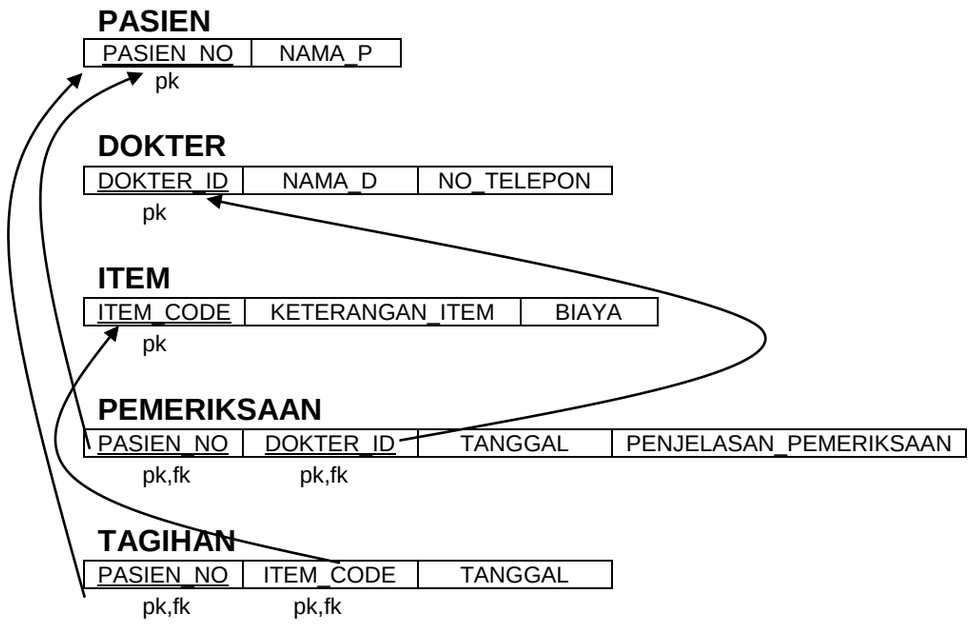
Mengupdate VIEW view_barang_1 hanya untuk yang stok lebih dari 5:

```
CREATE OR REPLACE VIEW view_barang_1 AS
SELECT id_barang, nama_barang, harga, stok
FROM barang
WHERE id_kategori = 1 AND stok>5;
```

Menghapus VIEW view_barang_1 sebagai berikut:

```
DROP VIEW view_barang_1;
```

4. LATIHAN



Gambar A. **Skema Basis Data RUMAH SAKIT**

PASIEN

PASIEN_NO	NAMA_P
3249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Gambar B. **Basis Data RUMAH SAKIT**

1. Implementasikanlah basis data Rumah Sakit diatas.
2. Dengan menggunakan INNER atau CROSS JOIN, tampilkan data dokter yang memeriksa pasiennya. Gunakan ON pada querynya.
3. Dengan menggunakan INNER atau CROSS JOIN, tampilkan data dokter yang memeriksa pasiennya. Gunakan USING pada querynya.
4. Dengan menggunakan LEFT JOIN, tampilkan semua data dokter beserta data transaksi pemeriksaannya.
5. Dengan menggunakan RIGHT JOIN, tampilkan semua data transaksi pemeriksaan beserta data dokternya.
6. Buatlah VIEW yang menyimpan pasien_no, nama_p, dokter_id, nama_d, tanggal.

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 12

Pertemuan ke-13

Operasi Himpunan

1. TUJUAN

Mahasiswa mampu menggunakan perintah union, difference (in/not in), intersection, any dan all, exist dan not exist.

2. TEORI SINGKAT

UNION

UNION merupakan operator atau perintah yang digunakan untuk menggabungkan hasil query atau isi data dari 2 (dua) tabel atau lebih denganketentuan jumlah, nama dan tipe data atau kolom dari masing– masing tabel yang akan ditampilkan datanya harus sama. Perintah ini terdiri dari dua jenis yaitu UNION dan UNION ALL. Untuk menghasilkan suatu data set harus disisipkan diantara perintah SELECT. UNION berguna untuk menampilkan hasil gabungan dari 2 tabel.

Sintaks query SQL

```
SELECT column_name (s) FROM table_name1
UNION
SELECT column_name (s) FROM table_name2;
```

INTERSECT

INTERSECT merupakan operator atau perintah yang digunakan untuk memperoleh irisan data hasil query atau isi data dari 2 (dua) tabel atau lebih dengan ketentuan jumlah, nama dan tipe data atau kolom dari masing – masing tabel yang akan ditampilkan datanya harus sama. Pada MySQL tidak terdapat operator INTERSECT akan tetapi sebagai gantinya dapat menggunakan operator IN. INTERSECT berguna untuk menampilkan irisan dari 2 tabel.

Sintaks query SQL

```
SELECT column_name (s) FROM table_name1
WHERE column_name (s) IN
SELECT column_name (s) FROM table_name2;
```

DIFFERENCE/EXCEPT (IN/NOT IN)

EXCEPT merupakan operator atau perintah yang digunakan untuk memperoleh data hasil query dari luar irisan data dari 2 (dua) tabel atau lebih dengan ketentuan jumlah, nama dan tipe data atau kolom dari masing – masing tabel yang akan ditampilkan datanya harus sama (kebalikan dari INTERSECT) Pada MySQL tidak terdapat operator EXCEPT akan tetapi sebagai gantinya dapat menggunakan operator NOT IN. DIFFERENCE/EXCEPT berguna untuk menampilkan perkecualian dari luar irisan 2 tabel.

```
SELECT column_name (s) FROM table_name1
WHERE column_name (s) NOT IN
SELECT column_name (s) FROM table_name2;
```

ANY dan ALL

Operator ANY digunakan untuk menampilkan record yang terkait dengan instruksi subquery. Operator ANY akan menghasilkan nilai TRUE (benar) jika paling tidak salah satu perbandingan dengan hasil subquery menghasilkan nilai TRUE.

Ilustrasinya:

Gaji > ANY (S)

Jika sub query S menghasilkan G1, G2,...,Gn, maka kondisi diatas identik dengan:

(Gaji > G1) OR (Gaji>G2) OR ... OR (Gaji > Gn)

Misalkan perintah untuk menampilkan semua data SKS yang jumlahnya bukan yang terkecil:

```
SELECT * FROM MatKuliah
WHERE SKS>=ANY (SELECT SKS FROM MataKuliah)
```

Operator ALL digunakan untuk melakukan perbandingan dengan subquery. Kondidi dengan ALL menghasilkan nilai TRUE (benar) jika subquery tidak menghasilkan apapun atau jika perbandingan menghasilkan TRUE untuk setiap nilai query terhadap hasil subquery.

Misalkan perintah untuk menampilkan data SKS yang jumlahnya paling tinggi:

```
SELECT * FROM MataKuliah
WHERE SKS >=ALL (SELECT SKS FROM MataKuliah)
```

EXIST dan NOT EXIST

Kata kunci EXIST dan NOT EXIST dirancang hanya untuk digunakan di subquery. Kata kunci menghasilkan nilai TRUE atau FALSE EXIST akan mengirim nilai TRUE jika dan hanya jika terdapat sedikitnya satu baris di table hasil yang dikirim oleh subquery. EXIST mengirim nilai FALSE jika subquery mengirm table kosong NOT EXIST kebalian dan EXIST. Karena EXIST dan NOT EXIST hanya memeriksa keberadaan baris-baris di table hasil subquery.

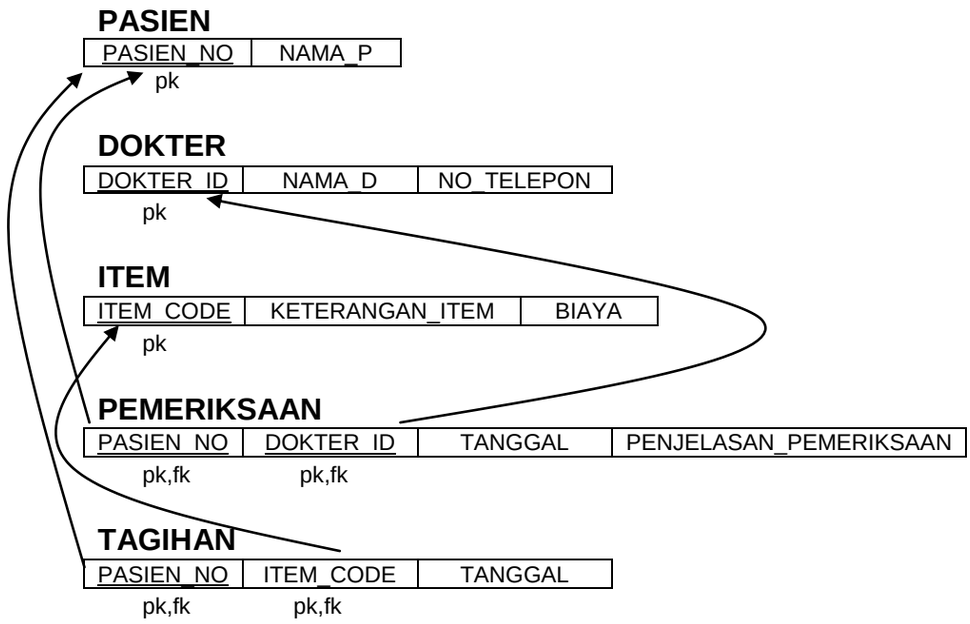
Sintaks EXIST:

```
SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

Sintaks NOT EXIST:

```
SELECT column1 FROM t1 WHERE NOT EXISTS (SELECT * FROM t2);
```

3. PELAKSANAAN PRAKTIKUM



Gambar A. **Skema Basis Data RUMAH SAKIT**

PASIEN

PASIEN_NO	NAMA_P
249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Gambar B. **Basis Data RUMAH SAKIT**

1. Implementasikanlah basis data Rumah Sakit diatas.
2. Dengan operator SET (UNION/INTERSECT/EXEPT), tampilkanlah nama-nama dokter yg pernah memeriksa pasien bernama Jhon atau pernah memeriksa pasien bernama Williams!

   Perintah SQL:

```
Select  D.nama_d
From    Pasien P, Dokter D, Pemeriksaan R
Where   P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        P.nama_p = 'Jhon'
UNION
(Select D.nama_d
From    Pasien P, Dokter D, Pemeriksaan R
Where   P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        P.nama_p = 'Williams)';
```

3. Dengan operator SET (UNION/INTERSECT/EXEPT), tampilkanlah nama-nama dokter yg pernah memeriksa pasien bernama Jhon dan pernah juga memeriksa pasien bernama Williams!

   Perintah SQL:

```
Select  D.nama_d
From    Pasien P, Dokter D, Pemeriksaan R
Where   P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        P.nama_p = 'Jhon'
IN
(Select D.nama_d
From    Pasien P, Dokter D, Pemeriksaan R
Where   P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        P.nama_p = 'Williams');
```

4. Dengan operator SET (UNION/INTERSECT/EXEPT), tampilkanlah nama-nama dokter yg pernah memeriksa pasien bernama Jhon namun tidak pernah memeriksa pasien bernama Williams!

   Perintah SQL:

```
Select  D.nama_d
From    Pasien P, Dokter D, Pemeriksaan R
Where   P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        P.nama_p = 'Jhon'
NOT IN
(Select D.nama_d
From    Pasien P, Dokter D, Pemeriksaan R
Where   P.pasien_no = R.pasien_no and
        D.dokter_id = R.dokter_id and
        P.nama_p = 'Williams');
```

5. Tampilkanlah semua data ITEM yang BIAYA nya bukan yang terkecil

Perintah SQL:

```
Select  *
From    Item
Where biaya >= ANY (Select biaya from Item);
```

6. Tampilkanlah data ITEM yang biayanya paling tinggi

Perintah SQL:

```
Select  *
From    Item
Where biaya >= ALL (Select biaya from Item);
```

7. Dengan perintah EXIST, tampilkanlah nama pasien yang pernah melakukan Kontrol Rutin

Perintah SQL:

```
Select  nama_p
From    Pasien
Where EXIST (
        Select  *
        From    Pemeriksaan
        Where Penjelasan_Pemeriksaan = 'Kontrol Rutin');
```

4. LATIHAN

Dengan menggunakan basis data Rumah Sakit kerjakanlah latihan berikut:

1. Tentukan ketergantungan fungsional pada tabel pasien, dokter, item, pemeriksaan dan tagihan!
2. Perhatikan tabel pasien, dokter, item, pemeriksaan dan tagihan! Apakah pada tabel-tabel tersebut terdapat ketergantungan fungsional transitif dan ketergantungan fungsional penuh?
3. Dengan operator SET (UNION/INTERSECT/EXEPT), tampilkanlah nama-nama pasien yg pernah diperiksa oleh dokter Wilcox atau dokter Arnold!
4. Dengan operator SET (UNION/INTERSECT/EXEPT), tampilkanlah nama-nama pasien yg pernah diperiksa oleh dokter Wilcox dan pernah juga diperiksa oleh dokter Arnold!
5. Dengan operator SET (UNION/INTERSECT/EXEPT), tampilkanlah nama-nama pasien yg pernah diperiksa oleh dokter Wilcox namun tidak pernah diperiksa oleh dokter Arnold!
6. Dengan perintah ANY, tampilkanlah semua data ITEM yang BIAYA nya bukan yang terbesar
7. Dengan perintah ALL, tampilkanlah data ITEM yang biayanya paling rendah
8. Dengan perintah EXIST, tampilkanlah nama dokter yang pernah memeriksa pasien tanggal 15/11/06

5. STRATEGI PRAKTIKUM

Untuk mengerjakan latihan pada modul 13 ini, skema dan isi basis data Rumah Sakit sudah diimplementasikan sebelumnya.

Pertemuan ke-14

FUNGSI AGREGAT

1. TUJUAN

Mahasiswa mampu menggunakan fungsi agregat, group by, dan having.

2. TEORI SINGKAT

GROUP BY

Group By adalah fungsi untuk mengelompokkan data dalam sebuah kolom yang ditunjuk. Fungsi ini akan menghasilkan kelompok data dengan menghilangkan data yang sama dalam satu tabel. Maka apabila dalam satu kolom terdapat beberapa data yang sama maka data yang akan ditampilkan hanya salah satu.

Sintax yang digunakan seperti berikut :
SELECT * FROM nama_tabel GROUP BY nama_kolom;

HAVING

HAVING berlaku untuk kelompok query group dan berfungsi seperti WHERE. Klausa HAVING digunakan untuk menentukan kondisi bagi klausa GROUP BY. Hanya kelompok atau group yang memenuhi kriteria HAVING saja yang akan diproses atau dihasilkan.

Sintax yang digunakan seperti berikut :
SELECT * FROM nama_tabel GROUP BY nama_kolom HAVING kondisi_nilai;

Penggunaan klausa Where, Group By, dan Having ada baiknya memperhatikan hal berikut :

- Where digunakan untuk menampilkan data yang difilter /baris – baris dari operasi yang dinyatakan oleh perintah From
- Group By digunakan untuk mengelompokkan hasil dari klausa Where
- Having digunakan untuk memfilter baris – baris dari hasil pengelompokkan.

FUNGSI AGREGAT

Fungsi Agregat merupakan fungsi yang berhubungan dengan sekumpulan data pada database sehingga sering disebut pula sebagai fungsi grup atau ringkasan. Fungsi ini menerima sekumpulan data dan mengembalikan nilai tunggal sebagai hasilnya. Ada beberapa fungsi agregat yang dikenal di dalam MySql, namun ada 5 fungsi utama yang sering digunakan yaitu:

SUM	digunakan untuk menghitung total nilai dari kolom tertentu
COUNT	digunakan untuk menghitung jumlah record.
AVG	digunakan untuk menampikan nilai rata-rata dari suatu kolom
MAX	digunakan untuk menampikan nilai tertinggi dari suatu kolom
MIN	digunakan untuk menampikan nilai terendah dari suatu kolom

Sintaks Fungsi Agregat

```
SELECT [SUM|COUNT|AVG|MAX|MIN] (nama_kolom)
FROM nama_tabel
WHERE kondisi
[GROUP BY nama_grup]
```

3. PELAKSANAAN PRAKTIKUM

Sebelum latihan, buatlah tabel buku dengan struktur tabel sebagai berikut.

```
mysql> CREATE TABLE buku (
    ->   kode_buku CHAR(5) PRIMARY KEY,
    ->   judul_buku VARCHAR(50),
    ->   pengarang VARCHAR(30),
    ->   penerbit VARCHAR(30),
    ->   tahun YEAR,
    ->   kategori VARCHAR(30),
    ->   harga INT(7),
    ->   tgl_inventaris DATE
    -> );
Query OK, 0 rows affected (0.15 sec)

mysql> DESC buku;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| kode_buku      | char(5)     | NO   | PRI | NULL    |       |
| judul_buku     | varchar(50) | YES  |     | NULL    |       |
| pengarang      | varchar(30) | YES  |     | NULL    |       |
| penerbit       | varchar(30) | YES  |     | NULL    |       |
| tahun          | year(4)     | YES  |     | NULL    |       |
| kategori       | varchar(30) | YES  |     | NULL    |       |
| harga          | int(7)      | YES  |     | NULL    |       |
| tgl_inventaris | date        | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

Isi Tabel Buku sebagai berikut:

kode_buku	judul_buku	pengarang	penerbit	tahun	kategori	harga	tgl_inventaris
B0001	Harry Potter	JK Rowling	British Press	2013	Fiksi	50000	2015-03-01
B0002	Sistem Basis Data	Abdul Kadir	Andi Offset	2013	Buku Teks	40000	2015-03-01
B0003	Sistem Basis Data	Fathansyah	ITB Press	2013	Buku Teks	30000	2015-03-01
B0004	Prophet Muhammad	Amir Abdullah	Madina Press	2014	Biografi	45000	2015-03-01
B0005	Ketika Cinta Bertasbih	Habiburahaman ES	Madina Press	2014	Fiksi	75000	2015-02-01
B0006	Pemrograman Basis Data	Abdul Kadir	Andi Offset	2015	Buku Teks	67000	2015-02-01

Fungsi Agregat: COUNT

- *Hitung jumlah record tabel buku*

```
mysql> SELECT COUNT(*)
    -> FROM buku;
+----------+
| COUNT(*) |
+----------+
|        6 |
```

```
+----------+
1 row in set (0.00 sec)
```

- *Hitung jumlah record tabel buku dengan nama kolom **jum_rec***

```
mysql> SELECT COUNT(*) AS jum_rec
    -> FROM buku;
+---------+
| jum_rec |
+---------+
|       6 |
+---------+
1 row in set (0.03 sec)
```

- *Hitung jumlah record untuk tahun 2013*

```
mysql> SELECT COUNT(*) AS jum_rec
    -> FROM buku
    -> WHERE tahun = 2013;
+---------+
| jum_rec |
+---------+
|       3 |
+---------+
1 row in set (0.03 sec)
```

Fungsi Agregat SUM

- *Hitung total harga*

```
mysql> SELECT SUM(harga) AS total_harga
    -> FROM buku;
+-------------+
| total_harga |
+-------------+
|      307000 |
+-------------+
1 row in set (0.03 sec)
```

- *Hitung total harga untuk tahun 2013*

```
mysql> SELECT SUM(harga) AS total_harga
    -> FROM buku
    -> WHERE tahun=2013;
+-------------+
| total_harga |
+-------------+
|      120000 |
+-------------+
1 row in set (0.00 sec)
```

Fungsi Agregat MAX

- *Tampilkan harga tertinggi*

```
mysql> SELECT MAX(harga) AS harga_tertingi
    -> FROM buku;
+---------------+
| harga_tertingi |
+---------------+
```

```
|          75000 |
+----------------+
1 row in set (0.00 sec)
```

- *Tampilkan harga tertinggi untuk tahun 2013*

```
mysql> SELECT MAX(harga) AS harga_tertinggi
    -> FROM buku
    -> WHERE tahun=2013;
+-----------------+
| harga_tertinggi |
+-----------------+
|           50000 |
+-----------------+
1 row in set (0.00 sec)
```

Fungsi Agregat: MIN

- *Tampilkan harga terendah*

```
mysql> SELECT MIN(harga) AS harga_terendah
    -> FROM buku;
+----------------+
| harga_terendah |
+----------------+
|          30000 |
+----------------+
1 row in set (0.00 sec)
```

- *Tampilkan harga terendah untuk tahun 2013*

```
mysql> SELECT MIN(harga) AS harga_terendah
    -> FROM buku
    -> WHERE tahun=2013;
+----------------+
| harga_terendah |
+----------------+
|          30000 |
+----------------+
1 row in set (0.00 sec)
```

Fungsi Agregat AVG

- *Tampilkan harga rata-rata*

```
mysql> SELECT AVG(harga) AS harga_rerata
    -> FROM buku;
+--------------+
| harga_rerata |
+--------------+
|   51166.6667 |
+--------------+
1 row in set (0.00 sec)
```

- *Tampilkan harga rata-rata untuk tahun 2013*

```
mysql> SELECT AVG(harga) AS harga_rerata
    -> FROM buku
    -> WHERE tahun=2013;
+--------------+
| harga_rerata |
```

```
+--------------+
|   40000.0000 |
+--------------+
1 row in set (0.00 sec)
```

kode_buku	judul_buku	pengarang	penerbit	tahun	kategori	harga	tgl_inventaris
B0001	Harry Potter	JK Rowling	British Press	2013	Fiksi	50000	2015-03-01
B0002	Sistem Basis Data	Abdul Kadir	Andi Offset	2013	Buku Teks	40000	2015-03-01
B0003	Sistem Basis Data	Fathansyah	ITB Press	2013	Buku Teks	30000	2015-03-01
B0004	Prophet Muhammad	Amir Abdullah	Madina Press	2014	Biografi	45000	2015-03-01
B0005	Ketika Cinta Bertasbih	Habiburahaman ES	Madina Press	2014	Fiksi	75000	2015-02-01
B0006	Pemrograman Basis Data	Abdul Kadir	Andi Offset	2015	Buku Teks	67000	2015-02-01

GROUP BY

Kelompokkan dan hitunglah buku berdasarkan tahun terbitnya!

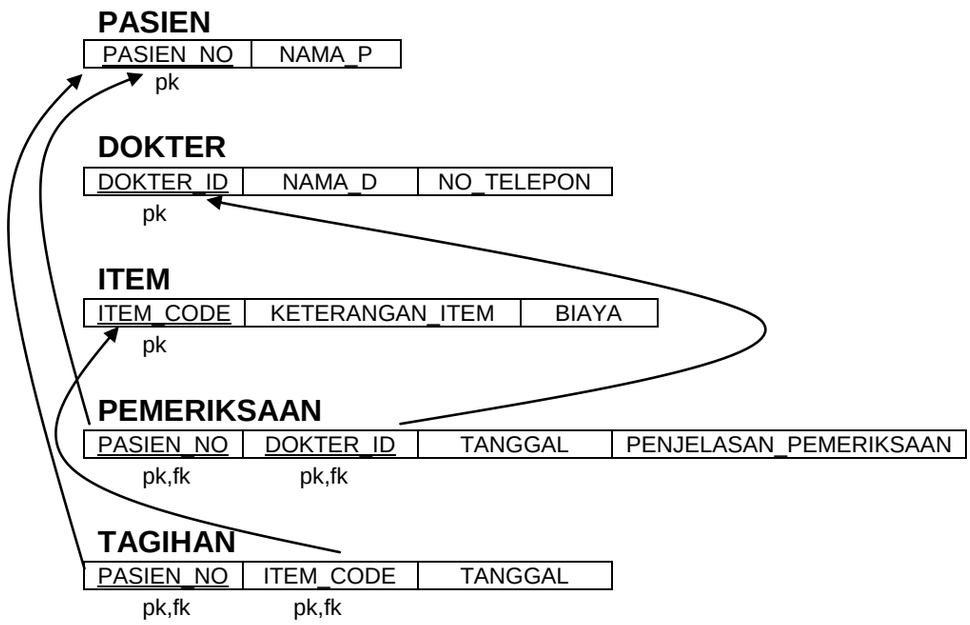
```
mysql> SELECT tahun, COUNT(kode_buku)
    -> FROM buku
    -> GROUP BY tahun;
```

HAVING

Kelompokkan dan hitunglah buku berdasarkan tahun terbitnya, hanya untuk yang jumlah bukunya lebih dari 2!

```
mysql> SELECT tahun, COUNT(kode_buku)
    -> FROM buku
    -> GROUP BY tahun
    -> HAVING COUNT(kode_buku)>2;
```

4. LATIHAN



Gambar A. **Skema Basis Data RUMAH SAKIT**

PASIEN

PASIEN_NO	NAMA_P
3249	Mary
6213	David
1379	John
5631	Williams
2298	Steve
3157	Ruth
6083	May
4139	Carl

DOKTER

DOKTER_ID	NAMA_D	NO_TELEPON
A233	Wilcox	329-1848
K324	Nusca	516-3947
K425	Thomas	473-4928
G897	Margaret	123-0195
S765	Martinez	111-4891
J173	Barbara	537-8976
B467	Terry	178-323
P376	Arnold	987-1234

ITEM

ITEM_CODE	KETERANGAN_ITEM	BIAYA
200	Room semi-pr	200.000
205	Television	50.000
307	X-ray Tipe A	100.000
413	Lab test Tipe A	150.000
562	USG-Bone Tipe A	300.000

PEMERIKSAAN

PASIEN_NO	DOKTER_ID	TANGGAL	PENJELASAN_PEMERIKSAAN
1379	A233	12/11/06	Kontrol Rutin
1379	G897	15/11/06	Observasi pasca bedah
3249	A233	10/11/06	Observasi pra bedah
3249	K425	15/11/06	Observasi pasca bedah
6213	A233	22/11/06	Kontrol Rutin
6213	P376	20/11/06	Observasi pra bedah
6213	B467	27/11/06	Observasi pasca bedah
5631	A233	01/12/06	Cek hasil lab

TAGIHAN

PASIEN_NO	ITEM_CODE	TANGGAL
1379	307	13/11/06
3249	307	11/11/06
3249	205	13/11/06
3249	200	12/11/06
6083	200	01/12/06
2298	200	05/12/06
2298	307	06/12/06
2298	413	10/12/06

Gambar B. **Basis Data RUMAH SAKIT**

1. Implementasikanlah basis data Rumah Sakit diatas.
2. Hitunglah berapa tagihan pasien bernama John
3. Hitunglah berapa jumlah item yang digunakan oleh Steve
4. Untuk setiap pasien, tampilkan pasien_no, nama pasien dan jumlah dokter yang pernah memeriksanya!
5. Untuk setiap pasien, tampilkan pasien_no, nama pasien dan hitunglah jumlah item yang digunakan, hanya untuk pasien yang menggunakan lebih dari 2 item.
6. Lakukan normalisasi sampai dengan 3 NF untuk basis data Rumah Sakit dengan skema sebagai berikut:

RUMAH SAKIT "JAKARTA"

Identitas Pasien

Nomor : 3249

Nama : Mary

PEMERIKSAAN

Dokter_Id	Nama Dokter	Nomor Telepon	Tanggal	Penjelasan Pemeriksaan
A233	Wilcox	329-1848	10/11/06	Kontrol Rutin
K425	Thomas	473-4928	15/11/06	Observasi pasca bedah

TAGIHAN

Item_Code	Keterangan Item	Biaya	Tanggal
307	X-ray Tipe A	100.000	11/11/06
205	Television	50.000	13/11/06
200	Room semi-pr	200.000	12/11/06

5. TUGAS

Sesuai GBPP tidak ada tugas pada pertemuan 13

DAFTAR PUSTAKA

1. Date, CJ. 2000. *An Introduction to Database System Seventh Edition*. New Jersey: Pearson Addison Weesley. ISBN: 979-683-185-6
2. Fatansyah. 2012. *Basis Data*. Bandung: Informatika Bandung. ISBN:978-602-8758-53-6
3. Nugroho, Adi. 2011. *Perancangan dan Implementasi Sistem Basis Data*. Andi Offset Yogyakarta. ISBN: 978-979-29-2609-5
4. Simarmata, Janner. 2007. *Perancangan Basis Data*. Andi Offset Yogyakarta. ISBN: 978-979-29-0104-7
5. Hariyanto, Bambang. 2004. *Sistem Manajemen Basis Data*. Bandung: Informatika. ISBN: 979-3338-33-4