**MySQL**

**Pengertian MySQL**

MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya dengan cepat, serta merupakan multiuser dengan menggunakan dasar SQL (*structured query language*). MySQL merupakan dua bentuk lisensi, yaitu FreeSoftware dan Shareware. MySQL yang biasa digunakan adalah MySQL FreeSoftware yang berada di bawah lisensi GNU / GPL (*general public license*). MySQL merupakan sebuah database server yang free, yang berarti bebas digunakan secara pribadi maupun usaha tanpa harus membeli atau membayar lisensinya.

MySQL pertama kali dirilis oleh seorang programmer database yang bernama Michael Widenius. Selain database server, MySQL juga merupakan program yang dapat mengakses suatu database MySQL sebagai server, sehingga program yang dibuat sebagai client. Jadi MySQL adalah sebuah database yang dapat digunakan sebagai cliend maupun server. Database MySQL merupakan sebuah perangkat lunak database yang berbentuk database relasional atau disebut *Relational Database Management System* (RDBMS) yang menggunakan suatu bahasa permintaan SQL (*structured query language*).

**Sejarah MySQL**

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

Pada tahun 1995 David Axmark dari Detro HB berusaha menekan TcX untuk merelease MySQL di Internet. Ia juga membuat dokumen MySQL yang di build untuk GNU configure utility untuk linux dan solaris. Sekarang ini MySQL bekerja untuk banyak platform serta tersedia source codenya.

**Kelebihan MySQL**

Terdapat beberapa kelebihan database MySQL yang dimiliki, diantaranya adalah :

1. MySQL merupakan database management system (DBMS)
2. MySQL sebagai RDBMS (*relationam database management system*)
3. MySQL merupakans sebua database server yang free, artinya bebas menggunakan database untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi
4. MySQL merupakan sebuah database client
5. MySQL mampu menerima Query yang bertumpuk dalam satu permintaan atau multithreading
6. MySQL merupakan database yang mampu menyimpan data berkapasistas sangat besar hingga ukuran GigaByte
7. MySQL didukung oleh driver ODBC, dengan arti database MySQL dapat diakses menggunakan aplikasi apa saja
8. MySQL adalah database yang menggunakan enkripsi password
9. MySQL adalah database yang multiuser
10. MySQL mendukung field yang dijadikan sebagai kunci primer dan kunci unique
11. MySQL memiliki kecepatan dalam pembuatan tabel dan mengupdate table

**Pengertian SQL (*structured query language*)**

SQL (*structured query language*) adalah sebuah bahasa pemrograman database yang terstruktur, bahasa SQL ini dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam database maupun merelasikan antar database.

**Tipe – 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. Berikut berbagai tipe data pada MySQL dapat dilihat pada tabel berikut :

|  |  |
| --- | --- |
| **Tipe Data** | **Keterangan** |
| TINYINT | Ukuran 1 byte. bilangan bulat kecil dengan jangkuan untuk bilangan yang beranda : -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 jangkuan untuk bilangan yang beranda : -32768 s/d 32767 dan untuk yang tidak bertanda : 0 s/d 65535 |
| MEDIUMINT | Ukuran 3 byte. bilangan bulat dengan jangkuan untuk bilangan yang beranda : -8388608 s/d 8388607 dan untuk yang tidak bertanda : 0 s/d 16777215 |
| INT | Ukuran 4 byte. bilangan bulat kecil dengan jangkuan untuk bilangan yang beranda : -2147483648 s/d 2147483647 dan untuk yang tidak bertanda : 0 / 4294967295 |
| INTEGER | Ukuran 4 byte. Sinonim dari int |
| BIGINT | Ukuran 8 byte. bilangan bulat terbesar dengan jangkuan untuk bilangan yang beranda : -922337203685477808, s/d 9223372036854775806 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 | Ukuran M byte. bilangan pecahan, misalnya DECIMAL (5,2) dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99 |
| NUMERIC | Ukuran M byte. bilangan pecahan, misalnya DECIMAL (5,2 )dapat digunakan untuk menyimpan bilangan -99,99 s/d 99,99 |
| 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 dan jangkauan dari ‘1000-01-01’ s/d ‘9999-12-31’. |
| TIMESTAMP | Ukuran 4 Byte. Tanggal dan 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 |
| 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 CHAR 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. Tipe 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 GIGABYTE. 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) |

Terdapat beberapa perintah SQL yang digunakan, diantaranya adalah :

1. **DDL (*Data Definition Language*)**

DDL merupakana kelompok perintah yang berfungsi untuk mendefinisikan atribut – atribut basis data, tabel, atribut (kolom), Batasan – Batasan terhadap suatu atribut, serta hubungan antar tabel.

Berikut adalah perintah – perintah di dalam DDL, yaitu :

1. **CREATE**

Perintah create digunakan untuk membuat sebuah database dan tabel. Berikut sintaks yang digunakan, diantaranya :

* + Perintah Create untuk membuat database

**CREATE DATABASE** nama\_database;

Keterangan :

Nama\_database : nama dari database yang akan dibuat

Contoh :

Membuat database penggajian karayawan

**CREATE DATABASE** penggajian\_karyawan**;**

* + Perintah Create untuk membuat tabel

Berikut bentuk sintaks :

**CREATE TABLE nama\_table (field1 type(length), field2 type(length), field3 type(length),………., fieldn type(length));**

Contoh :

Membuat table karyawan pada database penggajian karyawan

**CREATE TABLE** karyawan **(**No\_Peg varchar(10), Nama varchar(20), Tgl\_Lahir Date, Alamat Varchar(30), No\_telepon Int(20));

1. **DROP**

Perintah drop adalah perintah untuk menghapus tabel yang telah dibuat. Berikut sintaks dari perintah drop :

**DROP TABLE** nama\_table**;**

Contoh :

Perintah untuk menghapus tabel karyawan

**CREATE DATABASE** karyawan**;**

1. **ALTER**

Alter merupakan perintah DDL yang ketiga yang berfungsi untuk memodifikasi tabel. Terdapat beberapa perintah yang dapat digunakan dengan perintah alter, diantaranya :

Contoh tabel :

|  |  |  |
| --- | --- | --- |
| **Field** | **Tipe\_data** | **Size** |
| No\_Peg | Varchar | 10 |
| Nama | Varchar | 20 |
| Tgl\_Lahir | Date |  |
| Alamat | Varchar | 30 |
| No\_telepon | Int | 20 |

* **Menambah kolom / field**

Berikut sintaks yang digunakan untuk menambahkan kolom / field pada sebuah tabel.

**ALTER** **TABLE** nama\_tabel **ADD** nama\_kolom typedata(size);

Pada perintah menambahkan kolom / field terdapat beberapa penerapan yang dapat dilakukan, diantaranya :

1. Bentuk tanpa menentukan posisi.

Contoh :

Untuk menambahkan kolom “pengalaman” ke tabel karyawan, query yang akan dibentuk seperti :

**ALTER** **TABLE** Karyawan **ADD** Pengalaman char(30);

1. Bentuk dengan meletakan field / kolom diawal, tambahkan sintaks first :

**ALTER** **TABLE** Karyawan **ADD** Pengalaman char(30) **FIRST**;

1. Bentuk untuk menyisipkan field setelah field tertentu, tambahkan sintaks after:

**ALTER** **TABLE** Karyawan **ADD** Pengalaman char(30) AFTER ALAMAT;

* **Menghapus kolom / field**

Berikut bentuk sintaks alter untuk menghapus kolom / field pada sebuah tabel:

**ALTER** **TABLE** nama\_tabel **DROP** nama\_kolom;

Contoh :

Untuk mendrop / hapus kolom “pengalaman” dari tabel karyawan, query yang akan digunakan adalah :

**ALTER** **TABLE** Karyawan **DROP** Pengalaman;

* **Membuat suatu *constraint* (*primary key dan foreign key*)**

Constraint adalah mekanisme untuk membatasi domain (nilai – nilai yang diperbolehkan) untuk sebuah kolom dalam tabel. Dengan memberikan constraint pada sebuah kolom, server dapat menjamin kondisi – kondisi, sebagai berikut :

* Keunikan data
* Memberikan nilai default
* Mengharuskan nilai baru mengikuti pola tertentu
* Mengharuskan nilai baru adalah anggota dari sebuah daftar yang telah ditentukan

Sintaks membuat constraint dengan alter :

**ALTER** **TABLE** nama\_table

**ADD CONSTRAINT** nama\_constraint **PRIMARY KEY** nama\_kolom;

Contoh :

Menambahkan primary key pada kolom / field no\_peg di tabel karyawan

**ALTER** **TABLE** Karyawan

**ADD CONSTRAINT** pk\_NOPEG **PRIMARY KEY** No\_Peg;

* **Mengubah tipe data atau lebar kolom pada tabel.**

Berikut bentuk sintaks alter :

**ALTER** **TABLE** nama\_table **MODIFY** nama\_kolom tipedata(size);

Keterangan :

* Nama\_table : nama tabel yang akan diubah tipe data atau lebar kolomnya.
* Field : kolom yang akan diubah tipe data atau lebarnya
* Tipedata(size) : tipe data baru atau tipe data laman dengan lebarnya

Contoh :

Untuk mengubah tipe data pada kolom / field nama di tabel karyawan dengan char(20).

**ALTER** **TABLE** Karyawan **MODIFY** nama char(20);

* **Mengubah nama kolom**

Berikut sintaks dari mengubah nama kolom pada perintah alter :

**ALTER** **TABLE** nama\_table **CHANGE** **COLUMN** nama\_kolomlama nama\_kolombaru tipedata(size);

Keterangan :

* nama\_table : nama tabel yang akan diubah nama kolomnya
* nama\_kolomlama : kolom / field yang akan diganti Namanya
* nama\_kolombaru : nama baru kolom
* tipedata : tipe data dari kolom yang diubah.

Contoh :

Untuk mengubah nama No\_telepon pada tabel karyawan

**ALTER** **TABLE** Karyawan **CHANGE COLUMN** No\_telepon no\_tlp int(20);

* **menghapus primary key pada tabel**

berikut sintaks untuk menghapus constraint primary key

**ALTER** **TABLE** nama\_table **DROP CONSTRAINT** namaconstraint;

1. mengganti nama tabel

query SQL untuk merubah nama tabel dengan menggunakan perintah **RENAME,** berikut sintaks yang digunakan adalah :

**RENAME TABLE** namatable\_lama **TO** namatable\_baru**;**

Contoh :

Mengubah nama tabel karyawan menjadi table pegawai

**RENAME TABLE** karyawan **TO** pegawai;

1. DML (*Data Manipulation Language*)

DML adalah sebuah metode query yang dapat digunakan apabila DDL telah digunakan, sehingga fungsi query DML ini untuk melakukan manipulasi terhadap database yang telah dibuat. Di dalam DML terdapat beberapa perintah yang digunakan, diantaranya adalah sebagai berikut :

1. **INSERT**

Perintah insert adalah perintah untuk memasukan data / entry data, dengan syarat untuk memasukkan data adalah telah terciptanya tabel pada sebuah database.

Terdapat dua bentuk sintaks dalam menerapkan perintah insert. Berikut sintaks yang digunakan adalah :

1. Memasukan data tanpa memanggil nama field terlebih dahulu.

**INSERT INTO nama\_table VALUES**

**(‘isi\_field1’,’isi\_field2’,’isi\_field2’,’isi\_field3’,….., ‘isi\_fieldn’);**

Contoh :

Tambahkan data karyawan di tabel karyawan

**INSERT INTO karyawan VALUES**

(‘001’,’kurniawan’,’1-01-1998’,’jakarta’,0965746);

1. Memasukkan data dengan memanggil nama field / kolomnya.

**INSERT INTO** nama\_table (nama\_field1, nama\_field2, nama\_field3, nama\_fieldn) **VALUES** (‘isi\_field1’,’isi\_field2’,’isi\_field2’,’isi\_field3’,….., ‘isi\_fieldn’);

Contoh :

Menambahkan data dengan memanggil nama field / kolom

**INSERT INTO** karyawan (No\_Peg, Nama, Tgl\_lahir, Alamat, No\_Telepon) **VALUES**

(‘001’,’kurniawan’,’1-01-1998’,’jakarta’,0965746);

1. **DELETE**

Perintah Delete merupakan perintah untuk menghapus satu atau beberapa record dalam sebuah tabel.

Berikut sintaks SQL perintah DELETE :

**DELETE FROM** nama\_table **WHERE** kondisi;

Keterangan :

* Nama\_table : berisi nama tabel dari tabel yang akan dihapus
* Kondisi : berisi baris dengan suatu kondisi

Contoh :

Menghapus baris pada tabel karyawan

**DELETE FROM** karyawan **WHERE** No\_Peg = “001”;

1. **UPDATE**

Perintah UPDATE digunakan untuk mengubah isi data pada satu atau beberapa kolom. Berikut sintaks yang digunakan untuk perintah UPDATE :

**UPDATE** nama\_tabel **SET** nama\_kolom1 = nilai1,

nama\_kolomn = nilai2 **WHERE** kondisi;

Contoh :

Mengubah isi dari kolom nama dan kolom no\_telepon pada tabel karyawan

**UPDATE** karyawan **SET** Nama = “Kurniawan sejahtera”,

No\_Telepon = 081356283 **WHERE** No\_Peg = “001”;

1. DCL (*Data Control Language)*

DCL adalah sebuah metode query SQL yang digunakan untuk memberikan hak otorisasi mengakses database, mengalokasikan space, pendefinsian space, dan pengauditan penggunaan database. Berikut adalah perintah query yang terdapat pada DCL:

1. GRANT : perintah untuk memberikan izin user mengakses tabel

dalam database.

1. REVOKE : perintah untuk membatalkan izin hak user, yang

ditetapkan oleh perintah GRANT

1. COMMIT : perintah untuk menetapkan penyimpanan database
2. ROLLBACK : perintah untuk membatalkan penyimpanan database

**Seleksi Data**

Menampilkan data adalah hal yang sangat penting karena kita harus melihat dan menyeleksi suatu data dalam tabel maupun antar tabel. Untuk melihat data atau *selection***,** queryyang digunakan adalah **SELECT** yang diikuti beberapa pernyataan khusus yang berkaitan dengan tabel yang diseleksi.

**Menampilkan data dari sebuah tabel**

Untuk menampilkan dari sebuah tabel dapat menggunakan sintaks berikut ini :

**SELECT (** field1, field2, …. , fieldN) **FROM** nama\_tabel;

Query di atas dapat diartikan bahwa data yang akan ditampilkan tabel hanya fiel – field tertentu saja dan jika ingin menampilkan seluruh field yang terdapat pada tabel dapat menggunakan sintaks sebagai berikut :

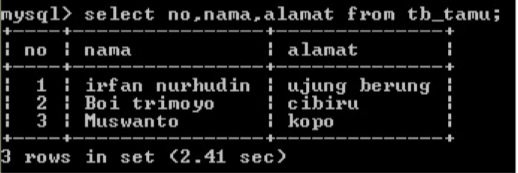
**SELECT \* FROM** nama\_tabel;

Contoh :

* Menampilkan isi / nilai keseluruhan field pada tabel dibawah ini :



* Menampilkan isi / nilai beberapa field pada tabel dibawah ini :



**Menampilkan data dengan perintah WHERE**

WHERE yang artinya dimana. Perintah ini berfungsi untuk menampilkan data dengan kondidsi tertentu. Berikut sintaks yang digunakan :

**SELECT \* FROM** nama\_tabel **WHERE** kondisi;

Contoh :

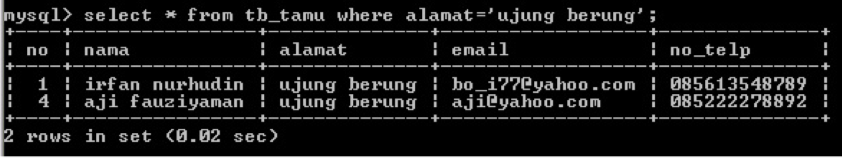
Berikut tabel yang akan di tampilkan dengan menggunakan WHERE :



Perintah untuk melakukan selection dengan where

**SELECT \* FROM** tb\_tamu **WHERE** alamat = “ujing berung”;

Hasil :



Keterangan :

Hasil diatas menampilkan semua data yang ada pada tabel tb\_tamu, dimana akan ditampilkan dari field alamat yang isi data dari field alamat hanya “ujung berung”. Sehingga yang keluar adalah data yang alamatnya di “ujung berung”.

**Menampilkan data dengan BETWEEN**

Between artinya diantara, between berfungsi untuk menampilkan data yang tertentu misalnya diantara 2000 dan 5000. Untuk menampilkan data dengan between dapat menggunakan sintaks, sebagai berikut :

**SELECT \* FROM** nama\_table **WHERE** kondisi **BETWEEN** nilai\_1 **AND** nilai\_2**;**

Contoh :

Terdapat tabel tb\_barang dengan isi tabel, sebagai berikut :

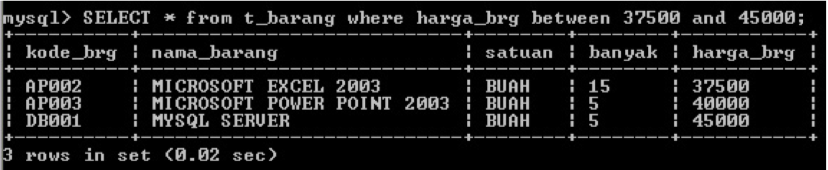


Perintah :

Tampilkan data barang pada tabel barang dengan harga barang antara 37500 dan 45000. Berikut perintah Query :

**SELECT \* FROM** t\_barang **WHERE** harga\_brg **BETWEEN** 37500 **AND** 45000**;**

Hasil :

keterangan :

Tabel diatas menunjukkan bahwa semua data ditunjukkan dari tabel t\_barang dan dimana yang ditampilkan dari field harga \_brg diantara 37500 dan 45000. Sehingga data yang ditampilkan hanya data yang bernilai 37500 dan 45000.