PRAKTIKUM BASIS DATA 2
PERTEMUAN 1

TRI WAHYU QUR'ANA, S.Kom, M.Kom

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ISLAM KALIMANTAN
MUHAMMAD ARSYAD AL BANJARI
BANJARMASIN
2021

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

Terdapat 3 (tiga) jenis perintah SQL, yaitu :

1. **DDL** atau Data Definition Language

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

   - **CREATE**
   - **ALTER**
   - **RENAME**
   - **DROP**

2. **DML** atau Data Manipulation Language

   DML merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data atau record dalam table. Perintah SQL yang termasuk dalam DML antara lain :

   - **SELECT**
   - **INSERT**
   - **UPDATE**
   - **DELETE**

3. **DCL** atau Data Control Language

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

   - **GRANT**
   - **REVOKE**

Membuat, Menampilkan, Membuka dan Menghapus Database

Membuat Database

Sintaks umum SQL untuk membuat suatu database adalah sebagai berikut :

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

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

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

```
CREATE DATABASE penjualan;
```

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

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

Menampilkan Database

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

```
SHOW DATABASES;
```

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

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

Membuka Database

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

```
USE penjualan;
```

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

```
Database changed
```

Menghapus Database

Untuk menghapus suatu database, sintaks umumnya adalah sbb :

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

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

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

```
DROP DATABASE penjualan;
```

Membuat Tabel Baru

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

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

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

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

Nama tabel **: pelanggan**

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

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

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

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

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

Pada perintah di atas, beberapa hal yang perlu diperhatikan :

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

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

```
SHOW TABLES;
```

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

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

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

```
DESC pelanggan;
```

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

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

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

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

Mengubah Struktur Table dengan ALTER

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

```
ALTER TABLE nama_tabel alter_options;
```

dimana :

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

  - **ADD** definisi_field_baru

    Option ini digunakan untuk menambahkan field baru dengan "**definisi_field_baru"** (nama field, tipe dan option lain).

  - **ADD INDEX** nama_index

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

  - **ADD PRIMARY KEY** (field_kunci)

    Option untuk menambahkan primary key pada tabel

  - **CHANGE** field_yang_diubah definisi_field_baru

    Option untuk mengubah field_yang_diubah menjadi definisi_field_baru

  - **MODIFY** definisi_field

    Option untuk mengubah suatu field menjadi definisi_field

  - **DROP** nama_field

    Option untuk menghapus field nama_field

  - **RENAME TO** nama_tabel_baru

    Option untuk mengganti nama tabel

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

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

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

2. Menambahkan primary key pada suatu tabel

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

3. Mengubah tipe field tgllahir menjadi varchar dalam tabel pelanggan

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

4. Menghapus field tgllahir dari tabel pelanggan

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

Mengubah Nama Tabel

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

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

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

Menghapus Tabel

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

```
DROP TABLE nama_tabel;
```

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

```
DROP TABLE pelanggan;
```

Menambah Record dengan INSERT

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

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

atau dapat dengan bentuk sebagai berikut :

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

atau dapat juga dengan bentuk sebagai berikut :

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

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

```
INSERT INTO pelanggan VALUES ('P0001', 'Yusuf Hadiwinata','JakartaSelatan', '085692019009',
'yusuf.hadiwinata@gmail.com')
```

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

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

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

```
SELECT * FROM pelanggan;
```

Dan berikut ini hasil dari perintah SQL di atas :

```
+--------------+-----------------+----------------+--------------+--------------------------+
| id_pelanggan | nm_pelanggan    | alamat         | telepon      | email                    |
+--------------+-----------------+----------------+--------------+--------------------------+
| P0001        | Yusuf Hadiwinata | Jakarta Selatan |085692019009 | yusuf.hadiwinata@gmail.com |
+--------------+-----------------+----------------+--------------+--------------------------+
1 row in set (0.08 sec)
```

Merubah Record dengan INSERT

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

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

```
UPDATE nama_tabel SET field1='nilaibaru'

[WHERE kondisi];
```

Pada perintah untuk update di atas :

- UPDATE merupakan perintah dasar untuk mengubah record tabel.
- nama_tabel merupakan nama tabel yang akan diubah recordnya.
- Perintah SET diikuti dengan *field-field* yang akan diubah yang mana diikuti juga dengan perubahan isi dari masing-masing field. Untuk mengubah nilai dari beberapa field sekaligus, gunakan koma (,) untuk memisahkan masing- masing field.
- Perintah WHERE diikuti oleh kondisi tertentu yang menentukan record mana yang akan diedit (diubah). Perintah WHERE ini boleh ada boleh juga tidak. Jika WHERE tidak ditambahkan pada perintah update maka semua record dalam tabel bersangkutan akan berubah.

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

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

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

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

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

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

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

Menghapus Record dengan DELETE

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

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

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

Pada perintah untuk delete di atas :

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

Perhatikan beberapa contoh perintah DELETE dari tabel pelanggan berikut ini !

1. Menghapus data pelanggan yang mempunyai id_pelanggan P0002

```
DELETE FROM pelanggan WHERE id_pelanggan='P0002';
```

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

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

2. Menghapus semua pelanggan yang beralamat di “Bandung”

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

Menampilkan Record dengan SELECT

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

Bentuk dasar perintah SELECTdata dari tabel adalah sebagai berikut :

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

Perhatikan beberapa contoh perintah SELECT dari tabel pelanggan berikut ini !

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

```
SELECT * FROM pelanggan;
```

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

```
+--------------+-----------------+--------------+-------------+---------------------------+
| id_pelanggan | nm_pelanggan    | alamat       | telepon     | email                     |
+--------------+-----------------+--------------+-------------+---------------------------+
| P0001        | Yusuf Hadiwinata | Bekasi       | 085692019009 | yusuf.hadiwinata@gmail.com |
| P0002        | Yoga Rimaldo    | Bandung      | 08571239501 | yoga@computradetech.com   |
| P0003        | Winny           | Jakarta Barat | 0815829221  | winny@computradetech.com  |
| P0004        | Ninda Budianto  | Jakarta Timur | 08171231234 | ninda@computradetech.com  |
+--------------+-----------------+--------------+-------------+---------------------------+
4 rows in set (0.00 sec)
```

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

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

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

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0001        | Yusuf Hadiwinata |
| P0002        | Yoga Rimaldo    |
| P0003        | Winny           |
| P0004        | Ninda Budianto  |
+--------------+-----------------+
4 rows in set (0.00 sec)
```

3. Menampilkan id, nama dan alamat dari data pelanggan yang mempunyai id **P0004**

```
SELECT id_pelanggan, nm_pelanggan, alamat
FROM pelanggan WHERE id_pelanggan = 'P0004';
```

Hasil query di atas adalah sbb :

```
+--------------+----------------+---------------+
| id_pelanggan | nm_pelanggan   | alamat        |
+--------------+----------------+---------------+
| P0004        | Ninda Budianto | Jakarta Timur |
+--------------+----------------+---------------+
1 row in set (0.00 sec)
```

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

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

Hasil query di atas adalah sbb :

```
+--------------+------------------+---------------------------+
| id_pelanggan | nm_pelanggan     | email                     |
+--------------+------------------+---------------------------+
| P0001        | Yusuf Hadiwinata | yusuf.hadiwinata@gmail.com|
+--------------+------------------+---------------------------+
1 row in set (0.02 sec)
```

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

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

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

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

Hasil query di atas adalah sbb :

```
+--------------+--------------+---------------+---------------+
| id_pelanggan | nm_pelanggan | alamat        | email         |
+--------------+--------------+---------------+---------------+
| P0005        | Rea          | Jakarta Timur | rea@gmail.com |
+--------------+--------------+---------------+---------------+
1 row in set (0.00 sec)
```

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

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

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

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0004        | Ninda Budianto  |
| P0005        | Rea             |
| P0003        | Winny           |
| P0002        | Yoga Rimaldo    |
| P0001        | Yusuf Hadiwinata |
+--------------+-----------------+
5 rows in set (0.00 sec)
```

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

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

Hasil query di atas adalah sbb :

```
+--------------+-----------------+
| id_pelanggan | nm_pelanggan    |
+--------------+-----------------+
| P0001        | Yusuf Hadiwinata |
| P0002        | Yoga Rimaldo    |
| P0003        | Winny           |
| P0005        | Rea             |
| P0004        | Ninda Budianto  |
+--------------+-----------------+
5 rows in set (0.00 sec)
```

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

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

Hasil query di atas adalah sbb :

```
+--------------+----------------+
| id_pelanggan | nm_pelanggan   |
+--------------+----------------+
| P0004        | Ninda Budianto |
| P0005        | Rea            |
| P0003        | Winny          |
+--------------+----------------+
3 rows in set (0.00 sec)
```

Keterangan

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

```
LIMIT awal, jumlah_record
```

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

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

Hasil query di atas adalah sbb :

```
+----------+
| COUNT(*) |
+----------+
| 5        |
+----------+
1 row in set (0.04 sec)
```