PRAKTIKUM SISTEM BASIS DATA

Modul 2: SQL Data Definition Language (DDL)

Deskripsi :

Modul ini berisi materi mengenai penulisan perintah Structured Query Language (SQL) untuk Data Definition Language (DDL). Terdapat beberapa perintah SQL DDL yang digunakan untuk membuat, mengubah, dan menghapus baik basis data maupun tabel. Selain itu pada modul ini juga berisi materi untuk menambah, mengubah, dan menghapus record pada basis data.

Tujuan :

Tujuan pembelajaran yang akan dicapai pada modul ini adalah :

(1) Mahasiswa mengetahui cara membuat, mengubah, dan menghapus database dan tabel pada Sistem Basis Data.
(2) Mahasiswa mengetahui cara membuat, mengubah, dan menghapus record pada sebuah tabel basis data.

Materi dan Pembagian waktu :

Materi dan pembagian waktu pelaksanaan perkuliahan meliputi :

No.	Materi	Waktu
1	Membuat, mengubah, dan menghapus database dan tabel	60 menit
2	Latihan perintah SQL DDL	60 menit
3	Evaluasi/Post Test	30 menit
	Total Waktu	150 menit

Penilaian :

Pada modul 2 dilakukan proses penilaian terhadap kemampuan mahasiswa berupa pre-test dengan nilai 1.5 poin serta post-test sebesar 4.5 poin.

Capaian Kompetensi Mahasiswa :

Kompetensi yang akan dicapai oleh mahasiswa pada modul ini meliputi :

1. Mahasiswa dapat membuat, mengubah, dan menghapus database dan tabel menggunakan perintah SQL DDL
2. Mahasiswa dapat membuat, mengubah, dan menghapus record pada tabel basis data menggunakan perintah SQL DDL.

Alat dan Bahan :

1. Modul Praktikum
2. Komputer
3. Viewer LCD

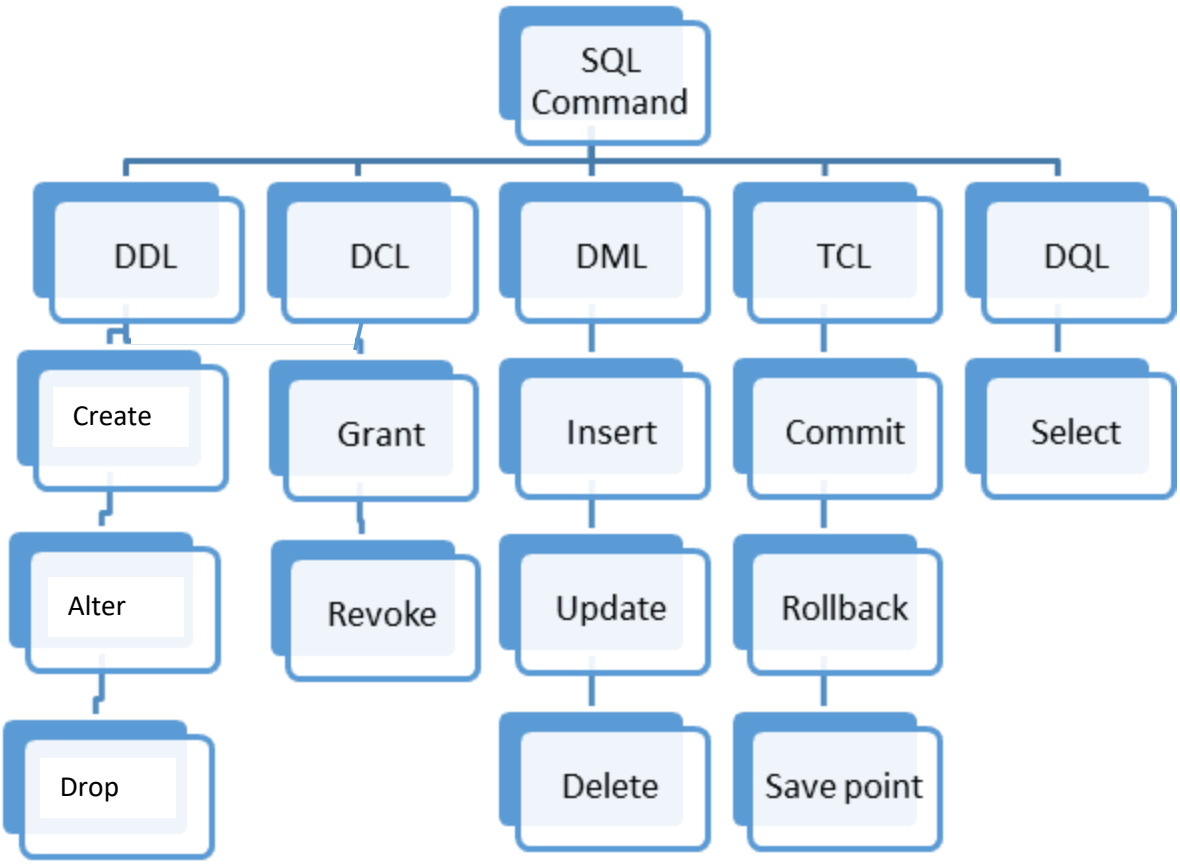
Materi 1 : Membuat, Mengubah, dan Menghapus Database dan Tabel dengan perintah SQL DDL

Waktu : 90 menit

Metode Pembelajaran : TC dan Praktikum.

1.1. Pendahuluan

Structured Query Language (SQL) merupakan perintah yang dapat digunakan untuk membuat database beserta struktur dari tabel pada sebuah basis data. SQL juga berisi sejumlah perintah yang dapat digunakan melakukan manipulasi data, administrasi data, dan sejumlah query untuk mengekstrak informasi yang terdapat pada sebuah database (Coronel, Morris, & Rob, 2013). Perintah SQL dikelompokkan menjadi 4 kelompok utama yaitu Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), dan Transaction Control Language (TCL). DDL merupakan perintah SQL yang bertujuan untuk membuat, mengubah, dan menghapus database dan objek lain pada sebuah sistem basis data. Sementara itu DML merupakan perintah SQL untuk melakukan manipulasi data pada tabel basis data. DCL dan TCL merupakan perintah query yang digunakan untuk melakukan kontrol terhadap data dan transaksi pada sistem basis data. Skema pembagian perintah SQL dapat dilihat pada Gambar 1 dan fungsi dari setiap perintah SQL dapat dilihat pada tabel 1.



Gambar 1. Klasifikasi Perintah SQL (JigsawAcademy, 2021)

Tabel 1. Deskripsi Perintah SQL

Perintah	Deskripsi
DDL	
CREATE	Untuk membuat basis data dan objek pada basis data seperti tabel, index, views, store procedure, function, dan triggers)
ALTER	Untuk mengubah struktur dari database
DROP	Untuk menghapus objek pada basis data
TRUNCATE	Untuk menghapus semua record pada sebuah tabel basis data
COMMENT	Untuk menambahkan komentar pada kamus data
RENAME	Untuk mengubah nama objek
DML	
INSERT	Untuk menambah record pada tabel
UPDATE	Untuk mengubah data pada tabel
DELETE	Untuk menghapus record pada tabel
MERGE	Untuk menggabungkan perintah insert dan update pada tabel
CALL	Untuk memanggil sebuah perintah PL/SQL atau subprogram dari Java
EXPLAIN PLAN	Untuk melihat jalur pengaksesan data
LOCK TABLE	Untuk mengatur kontrol konkurensi pada tabel basis data
DCL	
GRANT	Untuk memberikan hak akses terhadap pengguna basis data
REVOKE	Untuk menghapus hal akses dari pengguna basis data
TCL	
COMMIT	Untuk menyetujui semua transaksi
ROLLBACK	Untuk membatalkan sebuah transaksi
SAVEPOINT	Sama seperti ROLLBACK dan menambahkan catatan pada setiap kelompok
SET TRANSACTION	Untuk menentukan spesifikasi transaksi.
DQL	
SELECT	Untuk melihat/mengambil/mengkueri data pada basis data

Beberapa hal yang harus diperhatikan dalam penulisan perintah SQL meliputi :

- Perintah SQL tidak Case Sensitive
- Perintah SQL dapat ditulis lebih dari 1 baris
- Setiap clauses biasanya diletakkan pada baris yang berbeda
- Setiap Keyword dapat dibuat pada baris yang terpisah
- Spasi dapat diberikan untuk mempermudah pembacaan
- Setiap perintah SQL diakhiri dengan tanda semicolon(;)

1.2. Perintah CREATE

1.2.1. Membentuk Database

Database atau basis data merupakan objek pertama yang harus dibuat pada sebuah sistem basis data.

Perintah SQL untuk membentuk database adalah :

```
CREATE DATABASE database_name;
```

Contoh :

- `CREATE DATABASE basisdata1;`
- `CREATE DATABASE registrasi;`

Setelah dibentuk, sebelum kita bisa mengisi komponen tabel dan sebagainya, kita perlu menyorot/mengaktifkan database tersebut dengan perintah **`USE`** *`database_name;`*

1.2.2. Membentuk Tabel

Setelah basis data terbentuk maka selanjutnya pengembang dapat membentuk objek tabel pada basis data. Pembentukan tabel didasarkan pada rancangan model basis data yang telah dibangun dan kamus data yang telah didefinisikan. Perintah SQL untuk membentuk tabel adalah sebagai berikut :

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Datatype atau tipe data yang digunakan di MySQL/MariaDB dapat dilihat kembali pada modul pertama. Atau dapat juga dibaca di W3Schools (W3Schools, 2021).

Constraint pada sebuah tabel basis data dapat beragam, beberapa diantaranya yang sering digunakan adalah :

- NOT NULL : tidak boleh kosong
- PRIMARY KEY : kunci yang membedakan record satu dengan yang lain
- FOREIGN KEY : kunci tamu yang mereferensikan ke tabel induk
- UNIQUE : tidak boleh kembar
- DEFAULT : nilai default
- CHECK : membatasi nilai yang boleh dimasukkan ke field
- AUTO_INCREMENT : sebagai bantuan untuk menciptakan sekuens angka

Contoh pembentukan tabel basis data dengan berbagai konstrain (W3Schools, 2021) :

```
CREATE TABLE movies (
    movieID char(5) NOT NULL PRIMARY KEY,
    movieName text NOT NULL,
    duration int NOT NULL,
    parentalRating varchar(15),
    releaseDate date NOT NULL
);


CREATE TABLE spareparts (
    partID int PRIMARY KEY AUTO_INCREMENT,
    partName varchar(255) NOT NULL,
    price int NOT NULL,
    stock int DEFAULT 1,
    rackLocation varchar(20) NOT NULL
);
CREATE TABLE employees (
    empID int AUTO_INCREMENT,
    nama varchar(255) NOT NULL,
    tgllahir date NOT NULL,
    gender enum(`M`,`F`),
    email varchar(100),
    CONSTRAINT employeesPK PRIMARY KEY (empID),
    CONSTRAINT employeesUQ UNIQUE (email)
);


CREATE TABLE stores (
    storeID int PRIMARY KEY AUTO_INCREMENT,
    storeName varchar(255) NOT NULL,
    address varchar(255)
    empCount int,
    phone varchar(20),
    city varchar(50),
    CONSTRAINT storesCK CHECK (empCount>10)
);
```

Sebuah tabel basis data baru juga dapat dibuat berdasarkan tabel yang sudah ada sebelumnya. Berikut perintah SQL untuk membentuk tabel berdasarkan tabel yang telah ada sebelumnya :

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Contoh:

```
CREATE TABLE storesDIY AS
    SELECT *
    FROM stores;
```

1.2.3. Membentuk Index

Index merupakan objek pada basis data yang berguna untuk mempercepat proses pengaksesan data. Objek index tidak dapat dilihat oleh pengguna. Memperbaharui sebuah tabel dengan index membutuhkan waktu lebih lama karena pembaharuan tabel akan disertai dengan pembaharuan pada file index. Jadi sebaiknya lakukan pembentukan index hanya untuk kolom/atribut penting yang sering dilakukan pencarian. Berikut perintah SQL untuk membentuk index (w3schools.com, 2017):

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);

CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Contoh :

```
CREATE INDEX idx_lastname
ON Persons (LastName);

CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

1.2.4. Membentuk View

View merupakan sebuah tabel virtual yang terbentuk berdasarkan hasil dari sebuah QUERY (SELECT). Berikut perintah untuk membentuk View pada basis data:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Contoh :

```
CREATE VIEW femaleEmp AS
SELECT *
FROM employees
WHERE gender=`F`;
```

1.3. Perintah ALTER

Perintah ALTER digunakan untuk melakukan penambahan, penghapusan, dan pengubahan atribut/kolom pada sebuah tabel.

Berikut perintah ALTER untuk **menambah kolom/atribut** baru pada basis data :

```
ALTER TABLE table_name
ADD column_name datatype;
```

Contoh :

```
ALTER TABLE Persons
ADD DateOfBirth date;
```

Perintah ALTER untuk **mengubah tipe data kolom/atribut** pada basis data :

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Contoh :

```
ALTER TABLE Persons
MODIFY COLUMN DateOfBirth year;
```

Jika ingin menambahkan tipe kolom increment otomatis, tinggal ditambahkan AUTO_INCREMENT setelah tipe data.

Perintah ALTER untuk **menghapus kolom/atribut** pada basis data :

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Contoh :

```
ALTER TABLE Persons
DROP COLUMN DateOfBirth;
```

Perintah ALTER untuk **mengubah nama kolom/atribut** pada basis data ada beberapa versi sintaks. Berikut sintaks untuk MariaDB:

```
ALTER TABLE table_name
CHANGE COLUMN original_column_name new_column_name datatype;
```

Contoh :

```
ALTER TABLE Persons
CHANGE COLUMN age usia float;
```

1.4. Perintah DROP

Perintah DROP digunakan untuk menghapus database atau tabel pada basis data. Terdapat 2 perintah DROP yaitu DROP DATABASE dan DROP TABLE. Berikut sintak perintah SQL untuk mengharus database dan tabel :

- `DROP DATABASE databasename;`
- `DROP TABLE table_name;`

1.5. Perintah TRUNCATE

Perintah TRUNCATE digunakan untuk menghapus semua record pada sebuah tabel. Berikut penulisan perintah TRUNCATE :

```
TRUNCATE table_name;
```

1.6. Perintah RENAME

Perintah RENAME digunakan untuk mengubah nama tabel. Berikut sintak untuk perintah RENAME :

```
RENAME TABLE table_name To new_table_name;
```

1.7. Perintah DESCRIBE

Perintah DESCRIBE digunakan untuk mendiskripsikan struktur tabel yang telah tercipta. Berikut sintak untuk perintah DESCRIBE :

```
DESC table_name;
```

Setelah kita membuat tabel kita bisa melanjutkan dengan pengisian tabel dan melihat hasil isiannya. Perintah pengisian dan melihat isinya akan dipelajari secara khusus pada modul-modul berikutnya. Untuk sementara cara sederhana untuk mengisi tabel adalah dengan sintaks berikut:

```
INSERT INTO table_name VALUES (attr1value, attr2value, attr3value, ...);
```

Sedangkan untuk membaca/melihat isi tabel secara keseluruhan dengan sintaks berikut:

```
SELECT * FROM table_name;
```

Materi 2 : Latihan Perintah DDL

Waktu : 60 menit

Metode Pembelajaran : Praktikum.

Berikut rancangan dan kamus data untuk sebuah Basis Data untuk Klinik Hewan.

Nama Tabel	Nama Atribut	Tipe Data	Auto Increment	Not NULL	PK or FK
hewan	idHewan	INT	Y	Y	PK
	namaHewan	VARCHAR(30)		Y	
dokter	idDokter	INT	Y	Y	PK
	namaDokter	VARCHAR(30)		Y	
obat	idObat	INT	Y	Y	PK
	namaObat	VARCHAR(30)		Y	

Nama Tabel	Nama Atribut	Tipe Data	Auto Increment	Not NULL	PK or FK	FK Reference Tabel
minuman	idMinuman	INT	Y	Y	PK	
	namaMinuman	VARCHAR(30)		Y		
karyawan	idKaryawan	INT	Y	Y	PK	
	namaKaryawan	VARCHAR(30)		Y		
pelanggan	idPelanggan	INT	Y	Y	PK	
	namaPelanggan	VARCHAR(30)		Y		

1. Buatlah database dengan nama **klinik** dan gunakan database tersebut.
2. Buatlah tabel **hewan** dengan struktur seperti pada kamus data
3. Buatlah tabel **dokter** dengan struktur seperti pada kamus data
4. Buatlah tabel **obat** dengan struktur seperti pada kamus data
5. Tambahkan atribut **harga** pada tabel **obat** dengan tipe data INT dan nilai default 50000.
6. Ubah nama tabel **hewan** menjadi **pet**.
7. Ubah nama atribut **idHewan** dan **namaHewan** menjadi **idPet** dan **namaPet**.
8. Tambahkan atribut **noHP** pada tabel **dokter** dengan tipe data VARCHAR dengan lebar yang Anda tentukan sesuai kebutuhan digit nomor HP pada umumnya. Atribut ini tidak harus diisi.

9. Tambahkan atribut **nik** pada tabel **dokter** dengan tipe data CHAR (16) dan harus diisi.

10. Tambahkan atribut **usia** pada tabel **pet** dengan tipe data VARCHAR(5) (contoh perkiraan isian “5 th”.)

11. Tambahkan atribut **gender** pada tabel **pet** dengan tipe data ENUM. Gunakan dua jenis data “M” dan “F” saja.

12. Terdapat saran bahwa sebaiknya isian dari atribut **usia** pada tabel **pet** dalam bentuk numerik (contoh: 5), ubahlah atribut ini dengan tipe data yang lebih sesuai.

13. Desainer database berpikir ulang lagi bahwa **nik** tidak dibutuhkan di tabel **dokter.** Hapus atribut tersebut.

14. Buatlah INDEX ‘idxPet’ untuk atribut **namaHewan** pada tabel **pet.**

15. Gandakan tabel **hewan** untuk keperluan backup.

16. Lakukan pengisian data PET, DOKTER, OBAT sesuai dengan tabel di bawah ini:

No.	**Nama Tabel**	**Nama Atribut**	**Record**
1.	hewan	namaPet	Vanila
		usia	5
		gender	M
2.	hewan	namaPet	Taro
		usia	1
		gender	F
3.	dokter	namaDokter	Suryani
		noHP	087777666
4.	dokter	namaDokter	Megah
		noHP	087777555
5.	dokter	namaDokter	Priya
		noHP	087777444
6.	obat	namaObat	Obat Mata Anjing
		harga	67000
7.	obat	namaObat	Obat Jantung Anjing

		harga	88000
8.	obat	namaObat	Obat Telinga Kucing
		harga	175000

17. Hapus semua data pada tabel **obat**.

References

Coronel, C., Morris, S., & Rob, P. (2013). *Database Principles Fundamentals of Design, Implementation, and Management 10th Edition.* Canada: Course Technology, Cengage Learning.

JigsawAcademy. (2021). *DML Commands in SQL: A Beginner's Guide In 6 Easy Points*. https://www.jigsawacademy.com/blogs/business-analytics/dml-commands/

W3Schools. (2021). *SQL Data Types for MySQL, SQL Server, and MS Access*. https://www.w3schools.com/sql/sql_datatypes.asp