TIPE DATA DAN DDL

OBJEKTIF :

1. Mahasiswa mampu menggunakan tipe data yang sesuai dengan kebutuhan kolom.
2. Mahasiswa mampu membuat *database*, menggunakan *database*, membuat tabel, memodifikasi tabel, menghapus tabel, dan menghapus *database*.

3.1 TIPE DATA PADA MYSQL

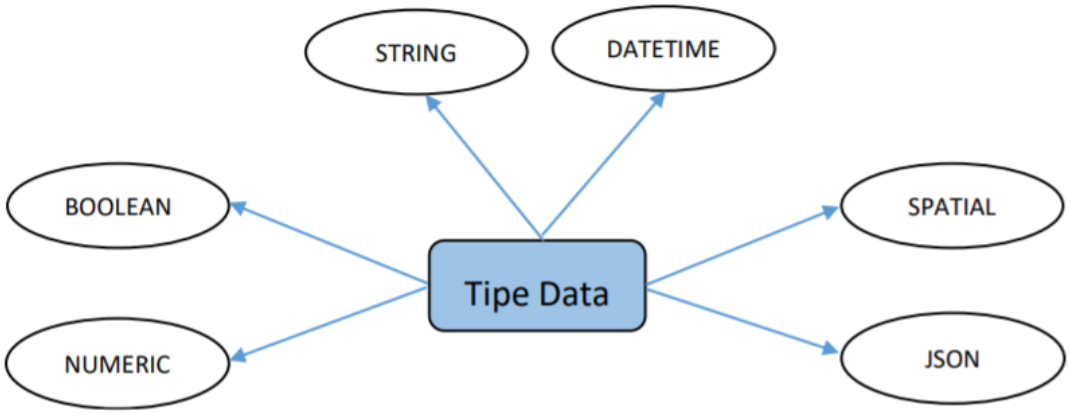
Tipe data digunakan mendefinisikan kolom saat membuat tabel. Maksud dari mendefinisikan kolom adalah penetapan tipe data ini akan berpengaruh pada batas-batas nilai yang disimpan ke setiap kolom. Sebagai contoh, jika kita menggunakan tipe data `integer`, maka kolom tersebut hanya dapat menyimpan angka yang bulat antara -32.786 hingga 32,767. Sehingga tidak mungkin untuk memasukkan data diluar batas nilai tersebut, apalagi data yang berupa `string` atau kumpulan huruf.

`Tabel Mahasiswa`

`NPM CHAR(8)`	`Nama VARCHAR(20)`
14117654	Rizky Aditya
15117529	Syifa Ravina

Contohnya pada Tabel Mahasiswa. Pada kolom NPM dan Nama menggunakan tipe data yang berbeda. Untuk kolom NPM menggunakan tipe data `char` dengan panjang karakter maksimal yang dapat diinput sebanyak 8 karakter. Untuk kolom Nama menggunakan tipe data `varchar` dengan panjang karakter maksimal yang dapat diinput sebanyak 20 karakter. Untuk menggunakan sebuah tipe data disesuaikan pada kebutuhan kolom tersebut. Misalnya, jika kita ingin menginput sebuah bilangan kita bisa menggunakan tipe data `integer`.

Tipe data dibagi menjadi 5, seperti pada gambar berikut :



3.1.1 TIPE DATA NUMERIC

Tipe data `numeric` digunakan untuk data yang berisi angka.

Tabel 3.1 Tipe Data `Numeric`

Tipe	Ukuran	*Range*
`Tinyint`	1 bit	-128 s/d 127
`Smallint`	2 bit	-32768 s/d 32767
`Mediumint`	3 bit	-8388608 s/d 8388607
`Int`	4 bit	-2147483648 s/d 2147483647
`Bigint`	8 bit	-9223372036854775808 s/d 9223372036854775807
`Decimal`	*Optional*	(M,D) tergantung angka yang diinput
`Float`	4 bit	0 s/d 23
`Double`	8 bit	24 s/d 53
`Bit`	Maksimal 64 bit	1 s/d 64

Dalam tipe data `numeric` dibagi menjadi 9 tipe data, yaitu :

3.1.1.1 Tinyint

`Tinyint` adalah bilangan bulat yang berukuran sangat kecil. `Tinyint` digunakan untuk menyimpan bilangan bulat positif dan juga negatif yang dapat dihitung.

Bentuk umum :

```
(column_name TINYINT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `tinyint` yaitu untuk mendeklarasikan usia seseorang.

```
(UsiaSeseorang TINYINT);
```

3.1.1.2 Smallint

`Smallint` adalah bilangan bulat yang berukuran kecil dan lebih besar dari `tinyint`. `Smallint` digunakan untuk menyimpan bilangan bulat positif dan juga negatif yang dapat dihitung.

Bentuk umum :

```
(column_name SMALLINT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `smallint` yaitu untuk mendeklarasikan jumlah halaman buku.

```
(JumlahHalaman SMALLINT);
```

3.1.1.3 Mediumint

`Mediumint` adalah bilangan bulat yang berukuran sedang. `Mediumint` digunakan untuk menyimpan bilangan bulat positif dan juga negatif yang dapat dihitung.

Bentuk umum :

```
(column_name MEDIUMINT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `mediumint` yaitu untuk mendeklarasikan jumlah mahasiswa.

```
(JumlahMahasiswa MEDIUMINT);
```

3.1.1.4 Int

`Int` adalah bilangan bulat yang berukuran standar. `Int` digunakan untuk menyimpan bilangan bulat positif dan juga negatif yang dapat dihitung.

Bentuk umum :

```
(column_name INT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `int` yaitu untuk mendeklarasikan jumlah kota di dunia.

```
(JumlahKotaDunia INT);
```

3.1.1.5 Bigint

`Bigint` adalah bilangan bulat yang berukuran besar. `Bigint` digunakan untuk menyimpan bilangan bulat positif dan juga negatif yang dapat dihitung.

Bentuk umum :

```
(column_name BIGINT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `bigint` yaitu untuk mendeklarasikan jumlah penduduk di dunia.

```
(JumlahPendudukDunia BIGINT);
```

3.1.1.6 Decimal

`Decimal` merupakan tipe data *fixed point* dengan nilai *input* tetap. `Decimal` digunakan untuk angka yang mengandung desimal.

Bentuk umum :

```
(column_name DECIMAL(M,D));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `M` : Merupakan total digit keseluruhan. Kisaran M adalah 1 hingga 65.
- `D` : Merupakan jumlah digit di belakang koma. Kisaran D adalah 0 dan 30. D (<=) M.

Contoh penggunan dari `decimal` untuk mendeklarasikan nilai phi dengan nama kolom Phi.

```
CREATE TABLE TipeDataDecimal (Phi DECIMAL(3,2));
INSERT INTO TipeDataDecimal VALUES (3.14);
SELECT*FROM TipeDataDecimal;
```

Menghasilkan *output* :

	Phi
▶	3.14

`DECIMAL(3,2)` artinya 3 angka keseluruhan dan 2 angka di belakang koma. Jika tidak menginputkan nilainya, maka nilai *default* adalah `DECIMAL(10,0)`.

3.1.1.7 Float

`Float` adalah tipe data *floating point* dengan nilai *input* tidak tetap. `Float` menyimpan secara akurat sebanyak 7 angka, dimanapun letak komanya.

Bentuk umum :

```
(column_name FLOAT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunan dari `float` untuk mendeklarasikan nilai phi dengan nama kolom Phi.

```
CREATE TABLE TipeDataFloat (Phi FLOAT);
INSERT INTO TipeDataFloat VALUES (3.14678954389);
SELECT*FROM TipeDataFloat;
```

Menghasilkan *output* :

	Phi
▶	3.14679

3.1.1.8 Double

`Double` adalah tipe data *floating point* dengan nilai input tidak tetap. `Double` menyimpan secara akurat sebanyak 15 angka.

Bentuk umum :

```
(column_name DOUBLE);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunan dari `double` untuk mendeklarasikan nilai phi dengan nama kolom phi.

```
CREATE TABLE TipeDataDouble (Phi DOUBLE);
INSERT INTO TipeDataDouble VALUES (3146789543878965436543329);
SELECT*FROM TipeDataDouble;
```

Menghasilkan *output* :

	Phi
▶	3.146789543878966e23

3.1.1.9 Bit

`Bit` adalah satuan bit terendah yang mempunyai nilai biner 1 dan 0.

Bentuk umum :

```
(column_name BIT(N));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `N` : Merupakan panjang karakter yang diinput .

Sebagai contoh, disini akan terbentuk tabel jadwal kuliah dengan kolom hari menggunakan tipe data `bit`. Sehingga dimasukkan nilai untuk kolom hari yaitu B'1111000', seperti berikut :

```
CREATE TABLE JadwalKuliah (Tahun INT, MingguKe INT, Hari BIT(7));
INSERT INTO JadwalKuliah VALUES (2019,1,B'1111000');
SELECT*FROM JadwalKuliah;
```

Pada *output* akan menghasilkan 120 di kolom hari karena nilai 1111000 adalah 120 :

	Tahun	MingguKe	Hari
▶	2019	1	120

3.1.2 TIPE DATA BOOLEAN

Tipe data `boolean` adalah tipe data yang menyimpan dua nilai, yaitu nilai *True* dan nilai *False*. 1 mewakili nilai *True*, 0 mewakili nilai *False*.

Bentuk umum :

```
(column_name BOOLEAN);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Sebagai contoh, boolean untuk mendeklarasikan status mahasiswa yang lulus atau tidak lulus. Lulus *(True*=1), tidak lulus (*False*=0).

Contoh penggunan dari `decimal` untuk mendeklarasikan nilai phi dengan nama kolom Phi.

```
CREATE TABLE Jadwal (IDMahasiswa CHAR(8) PRIMARY KEY, MataKuliah VARCHAR(20),
StatusHasilUjian BOOLEAN);
```

Dan pada *output*, 1 mewakili nilai *True* dan 0 mewakili nilai *False.*

	IDMahasiswa	MataKuliah	StatusHasilUjian
	14117621	Sistem Basis Data	0
▶	14117898	Statistika	1

3.1.3 TIPE DATA STRING

Tipe data `string` merupakan gabungan dari huruf dan juga angka.

Tabel 3.2 Tipe Data `String`

Tipe	**Ukuran**	***Range***
`Char`	M x W	0 s/d 255
`Varchar`	(Panjang +1) bit	0 s/d 255
`Binary`	M bit	0 <= M <= 255
`Varbinary`	(Panjang +1) bit	0 s/d 255
`Tinyblob`	(Panjang + 1) bit	0 s/d 25
`Blob`	(Panjang + 2) bit	0 s/d 65535
`Mediumblob`	(Panjang + 3) bit	0 s/d 16777215
`Longblob`	(Panjang + 4) bit	0 s/d 4294967295
`Blob`	(Panjang + 2) bit	0 s/d 65535
`Mediumblob`	(Panjang + 3) bit	0 s/d 16777215
`Tinytext`	(Panjang + 1) bit	0 s/d 4294967295
`Text`	(Panjang + 2) bit	0 s/d 65535
`Mediumtext`	(Panjang + 3) bit	0 s/d 16777215
`Longtext`	(Panjang + 4) bit	0 s/d 4294967295
`Enum`	1 atau 2 bit tergantung *range*	0 s/d 65535
`Set`	1, 2, 3, 4, or 8 bit tergantung *range*	Sampai dengan 64

Dalam `string` dibagi menjadi 14 tipe data, yaitu :

3.1.3.1 Char

`Char` digunakan untuk menyimpan karakter dengan panjang tetap. `Char` menghitung *white space* atau spasi kosong.

Bentuk umum :

```
(column_name CHAR(M));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `M` : Merupakan panjang karakter yang diinput. Nilai M dari 0 hingga 255 karakter .

Contoh penggunan dari `char` untuk mendeklarasikan NPM.

```
(NPM CHAR(8));
```

3.1.3.2 Varchar

`Varchar` digunakan untuk menyimpan karakter dengan panjang yang tidak tetap. `Varchar` tidak menghitung *white space* atau spasi kosong.

Bentuk umum :

```
(column_name VARCHAR(M));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `M` : Merupakan panjang karakter yang diinput. Nilai M dari 0 hingga 65535 karakter .

Contoh penggunan dari `varchar` untuk mendeklarasikan nama.

```
(Nama VARCHAR(20));
```

3.1.3.3 Binary

`Binary` digunakan untuk menyimpan nilai biner dengan panjang tetap. `Binary` bersifat *case sensitif. Case sensitif* adalah perbedaan penggunaan dalam penggunaan huruf besar dan kecil. Untuk tipe data binary sama seperti `char` dalam hal penyimpanan. `Binary` ini menyimpan karakter sesuai dengan parameter yang diinput. Nilai yang dihasilkan dalam tipe data ini adalah 0 dan 1.

Bentuk umum :

```
(column_name BINARY(M));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `M` : Merupakan panjang karakter yang diinput.

Sebagai contoh, disini terdapat *output* dari `binary` yang membuktikan bahwa `binary` menghasilkan *output* 1 dan 0 :

```
CREATE TABLE Nilai (NilaiMatematika BINARY(10));
INSERT INTO Nilai SET NilaiMatematika = 'HasilUjian';
SELECT HEX(NilaiMatematika), NilaiMatematika = 'HasilUjian', NilaiMatematika =
'HasilUjian\0\0' FROM Nilai;
```

Menghasilkan *output* :

	HEX(NilaiMatematika)	NilaiMatematika = 'HasilUjian'	NilaiMatematika = 'HasilUjian\0\0'
▶	486173696C556A69616E	1	0

3.1.3.4 Varbinary

`Varbinary` digunakan untuk menyimpan nilai biner dengan panjang tidak tetap. `varbinary` juga bersifat *case sensitif.*

Bentuk umum :

```
(column_name VARBINARY(M));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `M` : Merupakan panjang karakter yang diinput.

Untuk contohnya sama seperti `binary`, yang berbeda hanya pada hal penyimpanannya.

3.1.3.5 Tinyblob

`Tinyblob` digunakan untuk menyimpan data biner dengan ukuran sangat kecil, dimana karakter akan disimpan dalam bentuk `bit` .

3.1.3.6 Blob

`Blob` digunakan untuk menyimpan nilai biner dengan ukuran kecil.

3.1.3.7 Mediumblob

`Mediumblob` digunakan untuk menyimpan nilai biner dengan ukuran sedang.

3.1.3.8 Longblob

`Longblob` digunakan untuk menyimpan nilai biner dengan ukuran besar.

3.1.3.9 Tinytext

`Tinytext` digunakan untuk menyimpan kolom teks dengan ukuran sangat kecil.

Bentuk umum :

```
(column_name TINYTEXT);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `tinytext` yaitu untuk mendeklarasikan sebuah URL.

```
(URLWebsite TINYTEXT);
```

3.1.3.10 Mediumtext

`Mediumtext` digunakan untuk menyimpan kolom teks dengan ukuran sedang.

Bentuk umum :

```
(column_name MEDIUMTEXT);
```

Penjelasan :

- `column_name`: Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `mediumtext` untuk mendeklarasikan artikel berita dalam sebuah *website.*

```
(ArtikelBerita MEDIUMTEXT);
```

3.1.3.11 Longtext

`Longtext` digunakan untuk menyimpan kolom teks dengan ukuran besar.

Bentuk umum :

```
(column_name LONGTEXT);
```

Penjelasan :

- `column_name`: Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `longtext` untuk mendeklarasikan deskripsi sejarah di dunia.

```
(DeskripsiSejarah LONGTEXT);
```

3.1.3.12 Enum

`Enum` digunakan untuk memilih data sudah ditentukan sebelumnya. Satu pilihan *value* atau nilai hanya untuk satu data.

Bentuk umum :

```
(column_name ENUM ('value_1','value_2','value_3',…));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `value_1, value_2, …` : Merupakan isi atau data yang diinput ke dalam tabel.

Sebagai contoh, dalam tabel olahraga terdapat kolom nama olahraga yang dideklarasikan diawal voli pantai, sepak bola, dan tenis meja. Kemudian, diisi lagi oleh data voli pantai, tenis meja, dan futsal.

```
CREATE TABLE Olahraga (NamaOlahraga ENUM('Voli Pantai','Sepak Bola','Tenis Meja'));
INSERT INTO Olahraga VALUES ('Voli Pantai');
INSERT INTO Olahraga VALUES ('Tenis Meja');
INSERT INTO Olahraga VALUES ('Futsal');
SELECT*FROM Olahraga;
```

Menghasilkan *output* :

	NamaOlahraga
▶	Voli Pantai
	Tenis Meja

Pada *output*, kolom futsal tidak ada karena kolom futsal tidak dideklarasikan di awal.

3.1.3.13 Set

`Set` digunakan untuk memilih data sudah ditentukan sebelumnya. Satu pilihan *value* atau nilai bisa lebih dari satu data.

Bentuk umum :

```
(column_name ENUM ('value_1','value_2','value_3',…));
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `value_1, value_2, …` : Merupakan isi atau data yang diinput ke dalam tabel.

Sebagai contoh, dalam tabel olahraga terdapat kolom nama olahraga yang dideklarasikan diawal voli pantai, sepak bola, dan tenis meja. Kemudian, diisi lagi oleh data voli pantai, tenis meja, dan futsal.

```
CREATE TABLE Olahraga (NamaOlahraga SET('Voli Pantai','Sepak Bola','Tenis Meja'));
INSERT INTO Olahraga VALUES ('Voli Pantai');
INSERT INTO Olahraga VALUES ('Tenis Meja,Sepak Bola');
INSERT INTO Olahraga VALUES ('Futsal,Tenis Meja');
SELECT*FROM Olahraga;
```

Menghasilkan *output* :

	NamaOlahraga
▶	Voli Pantai
	Sepak Bola,Tenis Meja

Pada *output*, kolom futsal dan tenis meja tidak ada karena kolom futsal tidak dideklarasikan di awal.

3.1.4 TIPE DATA DATETIME

Tipe data `datetime` adalah tipe data yang digunakan untuk membuat format waktu dan tanggal.

Tabel 3.3 Tipe Data `Datetime`

Tipe	Ukuran	*Range*
`Year`	1 bit	00-01-01 00:00:00 s/d 9999-12-31 23:59:5
`Date`	3 bit	1000-01-01 s/d 9999-12-31
`Time`	3 bit	1901 s/d 2155, dan 0000
`Datetime`	8 bit	1000-01-01 00:00:00 s/d 9999-12-31 23:59:59
`Timestamp`	4 bit	1970-01-01 00:00:01 UTC s/d 2038-01-19 03:14:07 UTC

Dalam `datetime` dibagi menjadi 5, yaitu :

3.1.4.1 Year

`Year` digunakan untuk menyimpan nilai tahun. Nilai tahun 70-99 akan dikonversi menjadi 1970 – 1999. Nilai tahun 00-69 akan dikonversi menjadi 2000-2069.

Bentuk umum :

```
(column_name YEAR);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan dari `year` yaitu untuk mendeklarasikan tahun bayaran pada universitas.

```
(TahunBayaran YEAR);
```

3.1.4.2 Date

`Date` digunakan untuk menyimpan nilai tanggal. Dideklarasikan dengan format tahun-bulan-tanggal. Format tanggal bisa menggunakan slash (/) dan format jam bisa menggunakan titik (.).

Bentuk umum :

```
(column_name DATE);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contohnya penggunaan `date` yaitu untuk mendeklarasikan tanggal pada suatu kuitansi.

```
(TanggalKuitansi DATE);
```

3.1.4.3 Time

`Time` digunakan untuk menyimpan nilai waktu dalam format 24 jam.

Bentuk umum :

```
(column_name TIME);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan `time` yaitu untuk mendeklarasikan waktu mulai dalam acara lomba lari.

```
(WaktuMulai TIME);
```

3.1.4.4 Datetime

`Datetime` digunakan untuk menyimpan nilai hari tanggal dan jam secara bersamaan. Format tanggal bisa menggunakan slash (/) dan format jam bisa menggunakan titik (.).

Bentuk umum :

```
(column_name DATETIME);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Contoh penggunaan `datetime` yaitu untuk mendeklarasikan hari dan waktu pemenang lari dalam suatu perlombaan.

```
(PemenangLari DATETIME);
```

3.1.4.5 Timestamp

`Timestamp` digunakan untuk menyimpan nilai tanggal, waktu, dan zona waktu. Untuk mengubah zona waktu menggunakan `SET time_zone`.

Bentuk umum :

```
(column_name TIMESTAMP);
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.

Sebagai contoh, disini kita membuat tabel bernama waktu dengan menggunakan zona waktu ke '+00:00' UTC dengan menggunakan pernyataan `SET time_zone`.

```
CREATE TABLE Waktu (DetailWaktu TIMESTAMP);
SET time_zone='+00:00';
INSERT INTO Waktu VALUES ('2020-09-12 00:00:01');
```

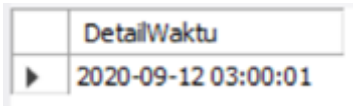
Menghasilkan *output* :

	DetailWaktu
▶	2020-09-12 00:00:01

Ubah zona waktu ke '+03:00' UTC dengan menggunakan pernyataan `SET time_zone`.

```
SET time_zone='+03:00';
```

Menghasilkan *output* :



	DetailWaktu
▶	2020-09-12 03:00:01

3.1.5 TIPE DATA SPATIAL

Tipe data `spatial` adalah tipe data yang berisi berbagai macam nilai geometris dan geografis. Dalam `spatial` dibagi menjadi 8, yaitu :

3.1.5.1 Geometry

`Geometry` digunakan untuk menyimpan semua nilai dari spatial.

3.1.5.2 Point

`Point` digunakan untuk menyimpan koordinat x dan koordinat y.

3.1.5.3 Linestring

`Linestring` adalah garis yang terdiri dari tepat dua titik.

3.1.5.4 Polygon

`Polygon` adalah sebuah nilai poligon.

3.1.5.5 Multipolygon

`Multipolygon` digunakan untuk menyimpan banyak nilai dari nilai poligon.

3.1.3.6 Multipoint

`Multipoint` digunakan untuk menyimpan banyak nilai dari `point` (koordinat x dan koordinat y).

3.1.5.7 Multilinestring

`Multilinestring` digunakan untuk banyak garis yang terdiri lebih dari dua titik.

3.1.5.8 Geometrycollection

`Geometrycollection` digunakan untuk menampilkan banyak nilai dari `spatial`.

3.1.6 TIPE DATA JSON

Tipe data `JSON` untuk menyimpan dan mengelola dokumen `JSON` dengan lebih efektif. Jenis data `JSON` asli memberikan validasi otomatis pada dokumen `JSON` dan format penyimpanan yang optimal.

3.2 PERINTAH DDL

DDL atau *Data Definition Language* berkaitan dengan perintah-perintah untuk pendefinisian objek-objek basis data. Untuk perintah DDL terbagi menjadi 5, yaitu :

3.2.1 CREATE

`CREATE` merupakan perintah yang digunakan untuk membuat. `CREATE` dibagi menjadi 2, yaitu :

3.2.1.1 CREATE DATABASE

`CREATE DATABASE` digunakan untuk membuat *database.*

Bentuk umum :

```
CREATE DATABASE database_name;
```

Penjelasan :

- `database_name` : Merupakan nama *database* yang akan dibuat.

Contoh penggunaan dari `CREATE DATABASE` yaitu untuk membuat *database* universitas.

```
CREATE DATABASE Universitas;
```

3.2.1.2 CREATE TABLE

`CREATE TABLE` digunakan untuk membuat tabel dalam *database*.

Bentuk umum :

```
CREATE TABLE table_name (column_1 data_type, column_2 data_type, column_3
data_type, ... );
```

Penjelasan :

- `column_name` : Merupakan nama kolom yang akan dibuat.
- `column_1 data_type,…` : Merupakan kolom-kolom beserta tipe datanya yang akan dibuat.

Contoh penggunaan dari `CREATE TABLE` yaitu untuk membuat tabel mahasiswa.

```
CREATE TABLE Mahasiswa (NPM CHAR(8) PRIMARY KEY, NamaMahasiswa VARCHAR(40),
Kelas VARCHAR(10), SKS VARCHAR(50));
```

3.2.2 USE

`USE` digunakan untuk memilih *database* yang akan digunakan dari *database* yang sudah ada.

Bentuk umum :

```
USE database_name;
```

Penjelasan :

- `database_name` : Merupakan nama *database* yang akan dibuat.

Contoh penggunaan dari `USE` yaitu untuk membuat *database* universitas.

```
USE Universitas;
```

3.2.3 TRUNCATE TABLE

`TRUNCATE TABLE` digunakan untuk menghapus data dalam tabel, tetapi tidak dengan tabel itu sendiri.

Bentuk umum :

```
TRUNCATE table_name;
```

Penjelasan :

- `table_name` : Merupakan nama tabel yang akan dihapus isi datanya.

Berikut data dalam tabel universitas :

	NPM	NamaMahasiswa	Kelas	SKS
	14117821	Aisil Farhana	4KA09	3
	15117098	Fathir Amri	4EA03	3
▶	17117633	Zaidan Pratama	4IA02	2

Contoh penggunaan dari `TRUNCATE TABLE` yaitu untuk menghapus isi data dalam tabel universitas.

```
TRUNCATE Mahasiswa;
```

Menghasilkan *output* :

	NPM	NamaMahasiswa	Kelas	SKS
*	NULL	NULL	NULL	NULL

Terlihat pada *output*, data dalam tabel universitas telah terhapus, tetapi tabel universitasnya tetap ada.

3.2.4 ALTER

`ALTER` merupakan perintah untuk memodifikasi tabel. `ALTER` dibagi menjadi 3, yaitu :

3.2.4.1 ALTER TABLE ADD COLUMN

`ALTER TABLE ADD COLUMN` digunakan untuk menambahkan kolom dalam tabel.

Bentuk umum :

```
ALTER TABLE table_name ADD column_name data_type;
```

Penjelasan :

- `table_name` : Merupakan nama tabel yang akan digunakan.
- `column_name data_type` : Merupakan nama kolom beserta tipe datanya yang akan ditambahkan.

Contoh penggunaan dari `ALTER TABLE ADD COLUMN` yaitu untuk menambahkan kolom alamat dan tipe data yang digunakan adalah `tinytext`.

```
ALTER TABLE Mahasiswa ADD Alamat TINYTEXT;
```

3.2.4.2 ALTER TABLE MODIFY COLUMN

`ALTER TABLE MODIFY COLUMN` digunakan untuk mengubah data dalam kolom.

Bentuk umum :

```
ALTER TABLE table_name MODIFY column_name data_type;
```

Penjelasan :

- `table_name` : Merupakan nama tabel yang akan digunakan.
- `column_name data_type` : Merupakan nama kolom beserta tipe datanya yang akan diubah.

Contoh penggunaan dari `ALTER TABLE MODIFY COLUMN` yaitu untuk mengubah kolom alamat dengan tipe data yang digunakan sebelumnya `tinytext` menjadi `varchar` dengan panjang karakter 50.

```
ALTER TABLE Mahasiswa MODIFY Alamat VARCHAR(50);
```

3.2.4.3 ALTER TABLE DROP COLUMN

`ALTER TABLE DROP COLUMN` digunakan untuk menghapus kolom dalam tabel.

Bentuk umum :

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

Penjelasan :

- `table_name` : Merupakan nama tabel yang akan digunakan.
- `column_name` : Merupakan nama kolom yang akan dihapus.

Contoh penggunaan dari `ALTER TABLE DROP COLUMN` yaitu untuk menghapus kolom SKS dalam tabel mahasiswa.

```
ALTER TABLE Mahasiswa DROP COLUMN SKS;
```

3.2.5 DROP

`DROP` merupakan perintah yang digunakan untuk menghapus. `DROP` dibagi menjadi 2, yaitu :

3.2.5.1 DROP TABLE

`DROP TABLE` digunakan untuk menghapus tabel dalam *database.*

Bentuk umum :

```
DROP TABLE table_name;
```

Penjelasan :

- `table_name` : Merupakan nama tabel yang akan dihapus.

Contoh penggunaan dari `DROP TABLE` yaitu untuk menghapus tabel mahasiswa.

```
DROP COLUMN Mahasiswa;
```

3.2.5.2 DROP DATABASE

`DROP DATABASE` digunakan untuk menghapus *database.*

Bentuk umum :

```
DROP DATABASE database_name;
```

Penjelasan :

- `database_name` : Merupakan nama *database* yang akan dihapus.

Contoh penggunaan dari `DROP DATABASE` yaitu untuk menghapus *database* universitas.

```
DROP DATABASE Universitas;
```

Referensi :

[1] Fathansyah. 2018. *Basis Data Revisi Ketiga*. Bandung: Informatika.

[2] www.mysqltutorial.org. 2020. “MySQL Data Types”, https://www.mysqltutorial.org/mysql-data-types.aspx, diakses pada 16 Juli 2020.

[3] www.mysqltutorial.org. 2020. “MySQL CREATE DATABASE”, https://www.mysqltutorial.org/mysql-create-database/, diakses pada 18 Juli 2020.

[4] www.mysqltutorial.org. 2020. “MySQL CREATE TABLE”, https://www.mysqltutorial.org/mysql-create-table/, diakses pada 18 Juli 2020.

[5] www.mysqltutorial.org. 2020. “MySQL USE DATABASE”, https://www.mysqltutorial.org/mysql-select-database/, diakses pada 18 Juli 2020.

[6] www.mysqltutorial.org. 2020. “MySQL TRUNCATE TABLE”, https://www.mysqltutorial.org/mysql-truncate-table/, diakses pada 18 Juli 2020.

[7] www.mysqltutorial.org. 2020. “MySQL DROP TABLE”, https://www.mysqltutorial.org/mysql-drop-table, diakses pada 18 Juli 2020.

[8] www.mysqltutorial.org. 2020. “MySQL DROP DATABASE”, https://www.mysqltutorial.org/mysql-drop-database/, diakses pada 18 Juli 2020.