DATABASE?

Database adalah kumpulan data yang umumnya menggambarkan aktivitas dari satu atau lebih organisasi yang saling berhubungan.

Ketika kita menganalisis kebutuhan informasi dari sebuah organisasi, kita berusaha untuk mengidentifikasi:

- Entitas (Entity): objek yang berbeda atau khusus dalam organisasi yang akan direpresentasikan di dalam database.
- Atribut (Attribute): sifat atau karakteristik yang menggambarkan suatu aspek dari objek yang ingin kita catat.
- Relasi (Relationship): hubungan atau keterkaitan antara entitas-entitas tersebut.

Ada DBMS dan RDBMS, apa itu?

- DBMS (Database Management System) adalah perangkat lunak yang digunakan untuk membuat, mengelola, dan mengakses database. Contohnya seperti Microsoft Access dan SQLite.
- RDBMS (Relational Database Management System) adalah jenis DBMS yang menyimpan data dalam bentuk tabel-tabel yang saling berhubungan (relasional). Contohnya seperti MySQL, PostgreSQL, dan Oracle.

Jadi, RDBMS adalah DBMS yang lebih canggih, karena mendukung hubungan antar tabel menggunakan kunci (keys) seperti *primary key* dan *foreign key*.

Fokus kita kedepannya adalah menggunakan DBMS

DBMS (Database Management System) adalah sistem perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, memelihara, dan mengontrol akses ke database.

DBMS menyediakan fasilitas berikut:

- Data Definition Language (DDL): bahasa yang memungkinkan pengguna untuk menentukan tipe data, struktur data, serta batasan (constraint) pada data yang akan disimpan di dalam database.
- Data Manipulation Language (DML): digunakan untuk menyisipkan (insert), memperbarui (update), menghapus (delete), dan mengambil (retrieve) data dari database.
- Kontrol akses: DBMS juga menyediakan akses yang terkontrol agar hanya pengguna yang berwenang yang dapat mengakses atau memodifikasi data.

- Pro & Kontra DBMS
    - Pro
        - Mengontrol redundansi data: DBMS membantu mengurangi duplikasi data yang tidak perlu.
        - Konsistensi data: Data tetap seragam dan akurat di seluruh sistem.
        - Berbagi data: Beberapa pengguna dapat mengakses dan menggunakan data secara bersamaan.
        - Meningkatkan integritas data: DBMS menjaga keakuratan dan keandalan data.
        - Keamanan yang lebih baik: DBMS menyediakan mekanisme untuk mengatur siapa saja yang dapat melihat atau mengubah data.
    - Kontra
        - Kompleksitas: DBMS cukup rumit untuk dipelajari dan dikelola.
        - Ukuran: Membutuhkan ruang penyimpanan yang besar.
        - Biaya perangkat keras tambahan: Kadang memerlukan perangkat keras yang lebih kuat untuk menjalankannya.
        - Dampak besar jika terjadi kegagalan: Jika sistem DBMS gagal, banyak aplikasi yang bergantung padanya juga akan terpengaruh.

Program aplikasi database:

- Program aplikasi: program komputer yang berinteraksi dengan database dengan cara mengirimkan permintaan (request) yang sesuai ke DBMS.
- Bahasa query (query language): bahasa pemrograman khusus yang digunakan untuk mencari dan mengubah isi database**.**
- Structured Query Language (SQL): merupakan standar umum (de facto standard) yang digunakan sebagai bahasa query pada sistem RDBMS (Relational Database Management System)**.**

Kenapa kita memakai MySQL?

MySQL adalah database relasional open source yang paling banyak digunakan di dunia. MySQL berfungsi sebagai penyimpanan data utama (primary relational data store) untuk banyak website, aplikasi, dan produk komersial populer.

- Netflix,
- NASA,
- Amazon,
- Twitter,
- Spotify,
- YouTube,
- Etc (https://www.mysql.com/customers/industry/)

Entity-Relationship Diagram (ER Diagram)

Konsep Dasar :

Diagram ER merupakan model konseptual untuk menggambarkan struktur logis dari basis data berbasis grafis. Terdapat Entity, Relationship dan Atribut.

Macam macam atribut :

- Atribut Key
- Atribut Simple
- Atribut Komposit
- Atribut Single valued
- Atribut Multi valued
- Atribut Turunan (Derived)

Key

- Penggunaan key merupakan cara untuk membedakan suatu entitas didalam himpunan entitas dengan entitas lain
- Secara konsep, masing-masing entitas (nilainya) berbeda, perbedaannya terlihat pada isi dari masing-masing atributnya.
- Oleh karena itu, dibutuhkan suatu atribut yang memiliki nilai yang menjadi pembeda dengan entitas lain
- Key satu atau gabungan dari beberapa atribut yang dapat membedakan semua record dalam relasi secara unik

Primary key dan Foreign Key

- Primary Key:
    - Merupakan salah satu dari candidate key yang terpilih
    - Pemilihan primary key dari sejumlah candidate key umumnya didasari oleh:
        - Key tersebut lebih sering untuk dijadikan sebagai acuan
        - Key tersebut lebih ringkas
        - Jaminan keunikan key tersebut lebih baik
    - Contoh:
        - Mahasiswa = (NIM, NAMA, ALAMAT, LAHIR)
        - Primary key:
            - (NIM)

- Foreign key:
    - merupakan atribut dalam set entity yang merujuk ke primary key dari set entity yang lain.
    - Berfungsi untuk menghubungkan antara kedua entity set.

KARDINALITAS

Pemetaan Kardinalitas Relasi

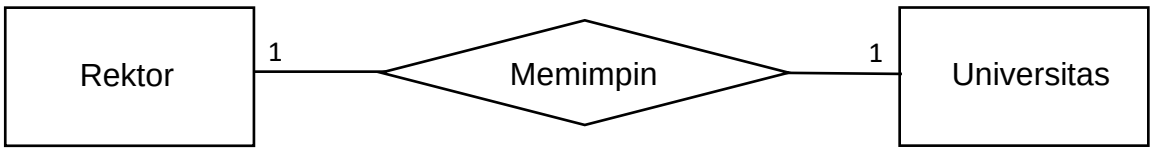
Menggambarkan banyaknya jumlah maksimum entitas dapat berelasi dengan entitas pada himpunan entitas yang lain

Untuk Himpunan relasi biner pemetaan kardinalitasnya dapat merupakan salah satu dari tipe-tipe berikut :

- Satu ke Satu (*One to one*)
- Satu ke Banyak (*One to many*)
- Banyak ke Satu (*Many to one*)
- Banyak ke Banyak (*Many to many*)

One to One

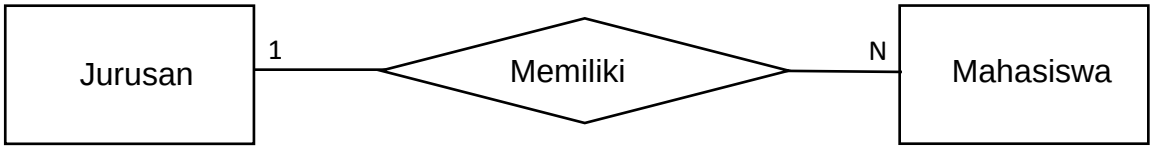
- Setiap record dari tabel E1 berhubungan dengan paling banyak satu record dari tabel E2, dan begitu juga sebaliknya setiap record dari tabel E1 berhubungan dengan paling banyak satu record dari tabel E2
- Relasi ini tidak umum karena data dalam bentuk relasi seperti ini dapat digabung dalam sebuah tabel
- Relasi One to One dapat digunakan jika ingin membagi sebuah tabel yang memiliki banyak field
- Contoh: 1 Rektor memimpin 1 Universitas, tidak pernah mungkin 1 rektor memimpin banyak universitas



One To Many

Setiap *record* dari tabel E1 dapat berhubungan dengan banyak *record* pada tabel E2, tetapi tidak sebaliknya, dimana setiap *record* dari tabel E2 berhubungan dengan paling banyak satu *record* pada tabel E1

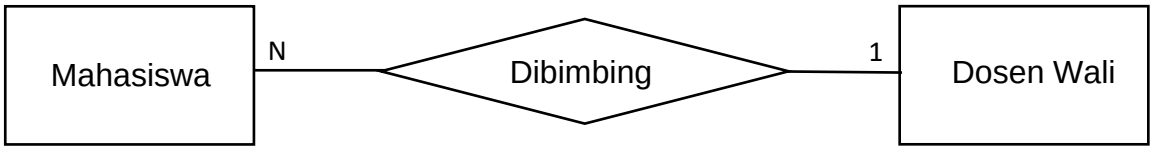
Contoh : 1 jurusan memiliki banyak mahasiswa**,** tetapi setiap mahasiswa hanya terdaftar pada 1 jurusan.



Many To One

Setiap *record* dari tabel E1 dapat berhubungan dengan paling banyak satu *record* pada tabel E2, tetapi tidak sebaliknya, dimana setiap *record* dari tabel E2 berhubungan dengan banyak *record* pada tabel E1

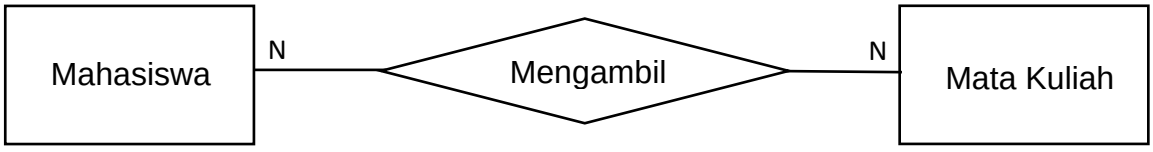
Contoh : Banyak mahasiswa dibimbing oleh 1 dosen wali, tetapi setiap dosen wali membimbing banyak mahasiswa.



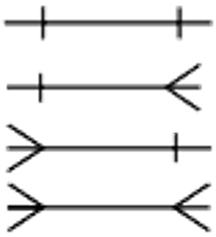
Many To Many

Setiap *record* dari tabel A dapat berhubungan dengan banyak *record* pada tabel B, dan demikian juga sebaliknya, dimana setiap *record* dari tabel B dapat berhubungan dengan banyak *record* pada tabel A

Contoh: Banyak mahasiswa dapat mengambil banyak mata kuliah, dan setiap mata kuliah dapat diambil oleh banyak mahasiswa.



Alternatif Simbol Kardinalitas



One To One

One To Many

Many To One

Many To Many

NORMALISASI

Kelebihan & Kekurangan Normalisasi :

Kelebihan:

- Mengeliminasi modification anomalies
- Mengurangi duplikasi data sehingga space penyimpanan data lebih hemat

Kekurangan:

- SQL query akan lebih rumit, terutama untuk mengakses data dari banyak tabel.
- DBMS akan berjalan lebih lambat karena perlu kerja yang ekstra.

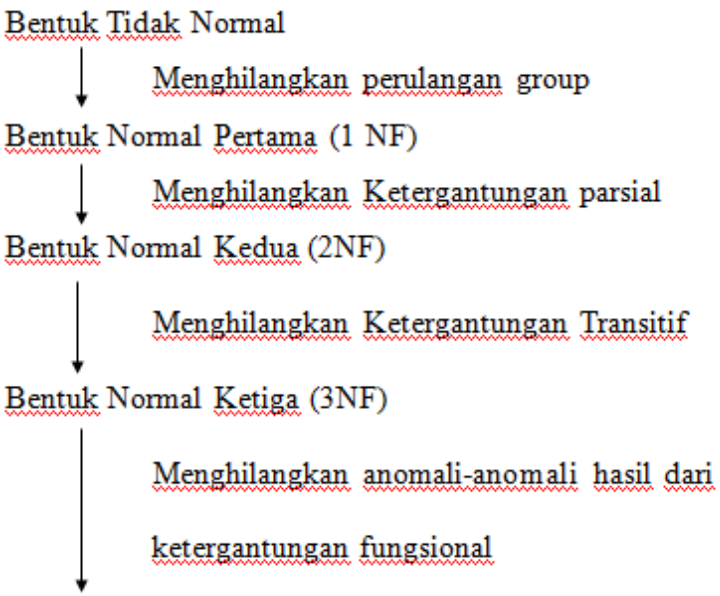
Functional Dependency (FD) :

- Untuk melakukan normalisasi, harus menentukan Functional Dependency (FD) atau ketergantungan fungsional terlebih dahulu.
- FD Disimbolkan dengan A -> B yang artinya B memiliki ketergantungan dengan A
- A secara fungsional menentukan B atau B secara fungsional tergantung pada A
- Jika setiap baris pada suatu tabel terdapat nilai A yang sama, maka nilai B juga pasti sama

Ada 3 Functional Dependency :

- Functional Dependency Penuh
- Functional Dependency Parsial
- Functional Dependency Transitif

Tahapan Normalisasi :



SQL DATA DEFINITION LANGUAGE (DDL)

Yang akan dipelajari :

- CREATE - digunakan untuk membuat database serta objek-objek di dalamnya**,** seperti table, index, view, stored procedure, function, dan trigger.
- ALTER – digunakan untuk mengubah struktur dari database atau tabel yang sudah ada.
- DROP – digunakan untuk menghapus objek dari database, seperti tabel, index atau view.
- RENAME – digunakan untuk mengganti nama object dalam database
- UPDATE - catatan: *perintah UPDATE bukan bagian dari DDL**,** tetapi termasuk dalam DML (Data Manipulation Language)**,** karena digunakan untuk memperbarui data di dalam tabel**,** bukan struktur databasenya.*

Database :

- Membuat database

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

- Select database

```
USE nama_database;
```

- Melihat database

```
SHOW DATABASES;
```

- Menghapus database

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

Tipe Data Numerik :

Data Type Syntax	Description
TINYINT	It is a very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. We can specify a width of up to 4 digits. It takes 1 byte for storage.
SMALLINT	It is a small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. We can specify a width of up to 5 digits. It requires 2 bytes for storage.
MEDIUMINT	It is a medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. We can specify a width of up to 9 digits. It requires 3 bytes for storage.
INT	It is a normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. We can specify a width of up to 11 digits. It requires 4 bytes for storage.
BIGINT	It is a large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. We can specify a width of up to 20 digits. It requires 8 bytes for storage.
FLOAT(m,d)	It is a floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals, and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float type. It requires 2 bytes for storage.
DOUBLE(m,d)	It is a double-precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double. It requires 8 bytes for storage.
DECIMAL(m,d)	An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.
BIT(m)	It is used for storing bit values into the table column. Here, M determines the number of bit per value that has a range of 1 to 64.
BOOL	It is used only for the true and false condition. It considered numeric value 1 as true and 0 as false.
BOOLEAN	It is Similar to the BOOL.

Tipe Data Waktu dan Tanggal :

Data Type Syntax	Maximum Size	Explanation
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	The default is 4 digits. It takes 1 byte for storage.
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'yyyy-mm-dd'. It takes 3 bytes for storage.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'. It takes 3 bytes plus fractional seconds for storage.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'yyyy-mm-dd hh:mm:ss'. It takes 5 bytes plus fractional seconds for storage.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC.	Displayed as 'YYYY-MM-DD HH:MM:SS'. It takes 4 bytes plus fractional seconds for storage.

Tipe Data String :

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Fixed-length strings. Space padded on the right to equal size characters.
VARCHAR(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Variable-length string.
TINYTEXT(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Here size is the number of characters to store.
MEDIUMTEXT(size)	It can have a maximum size of 16,777,215 characters.	Here size is the number of characters to store.
LONGTEXT(size)	It can have a maximum size of 4GB or 4,294,967,295 characters.	Here size is the number of characters to store.
BINARY(size)	It can have a maximum size of 255 characters.	Here size is the number of binary characters to store. Fixed-length strings. Space padded on the right to equal size characters. (introduced in MySQL 4.1.2)
VARBINARY(size)	It can have a maximum size of 255 characters.	Here size is the number of characters to store. Variable-length string. (introduced in MySQL 4.1.2)
ENUM	It takes 1 or 2 bytes that depend on the number of enumeration values. An ENUM can have a maximum of 65,535 values.	It is short for enumeration, which means that each column may have one of the specified possible values. It uses numeric indexes (1, 2, 3...) to represent string values.
SET	It takes 1, 2, 3, 4, or 8 bytes that depends on the number of set members. It can store a maximum of 64 members.	It can hold zero or more, or any number of string values. They must be chosen from a predefined list of values specified during table creation.

Membuat Sebuah Tabel :

```
CREATE TABLE [IF NOT EXISTS] nama_tabel(
     nama_kolom1 tipe_data(ukuran) [NULL|NOT NULL],
     nama_kolom2 tipe_data(ukuran) [NULL|NOT NULL],
     ...,
     table_constraints
);
```

Keterangan :

IF NOT EXISTS → digunakan agar tidak muncul error jika tabel dengan nama yang sama sudah ada.

table_constraints → digunakan untuk menentukan **kendala (constraints)** pada tabel, seperti:

- **PRIMARY KEY** : menentukan kolom kunci utama.
- **FOREIGN KEY** : menentukan kolom kunci tamu (relasi antar tabel).
- **UNIQUE** : memastikan nilai kolom tidak boleh duplikat.

Contoh Membuat Tabel :

```
CREATE TABLE IF NOT EXISTS mahasiswa (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nim CHAR(10) UNIQUE NOT NULL,
    nama VARCHAR(50) NOT NULL,
    jurusan VARCHAR(30),
    alamat TEXT,
);
```

Merubah Kolom dalam Tabel :

- Menambahkan 1 kolom di dalam tabel

```
ALTER TABLE nama_tabel
ADD nama_kolom_baru tipe_data(ukuran) [NULL|NOT NULL]
[FIRST | AFTER nama_kolom]
```

- Menambahkan lebih dari 1 kolom di dalam tabel

```
ALTER TABLE nama_tabel
ADD nama_kolom_baru tipe_data(ukuran) [NULL|NOT NULL]
[FIRST | AFTER nama_kolom],
ADD nama_kolom_baru tipe_data(ukuran) [NULL|NOT NULL]
[FIRST | AFTER nama_kolom],
…
;
```

- Memodifikasi kolom yang sudah ada di dalam tabel

```
ALTER TABLE nama_tabel
MODIFY nama_kolom tipe_data(ukuran) [NULL|NOT NULL]
[FIRST | AFTER column_name];
```

Menghapus Tabel dan Kolom :

- Menghapus 1 tabel

```
DROP TABLE nama_tabel;
```

- Menghapus lebih dari 1 tabel

```
DROP TABLE nama_tabel1, nama_tabel2, …;
```

- Menghapus Kolom

```
ALTER TABLE nama_tabel
DROP COLUMN nama_kolom;
```

Mengganti Nama Kolom :

- Mengganti nama tabel

```
ALTER TABLE nama_tabel_lama
RENAME TO nama_tabel_baru
```

- Mengganti nama kolom

```
ALTER TABLE nama_tabel
CHANGE COLUMN nama_kolom_lama nama_kolom_baru
tipe_data(ukuran) [NULL|NOT NULL]
[FIRST | AFTER nama_kolom]
```

BASIC SQL QUERY

Contoh Database :

```
MariaDB [kuliah]> desc stock_buku;
+--------------+---------+------+-----+---------+----------------+
| Field        | Type    | Null | Key | Default | Extra          |
+--------------+---------+------+-----+---------+----------------+
| id_buku      | int(11) | NO   | PRI | NULL    | auto_increment |
| judul_buku   | text    | YES  |     | NULL    |                |
| jumlah_stock | int(11) | YES  |     | NULL    |                |
+--------------+---------+------+-----+---------+----------------+
3 rows in set (0.003 sec)
```

Kemudian isi dengan ini :

```
MariaDB [kuliah]> insert into stock_buku (judul_buku, jumlah_stock) values
    -> ("Principles and Practices of Interconnection Networks",10),
    -> ("Building DMZs for Enterprise Networks",20),
    -> ("Virtual Private Networks", 11),
    -> ("Computer Networking: a Top Down Approach",4),
    -> ("Computer Networking Illuminated", 7);
Query OK, 5 rows affected (0.071 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Menampilkan Seluruh Data Pada 1 Tabel :

Syntax: SELECT * FROM nama_tabel;

Contoh: SELECT * FROM stock_buku;

```
MariaDB [kuliah]> select * from stock_buku;
+---------+-----------------------------------------------------+--------------+
| id_buku | judul_buku                                          | jumlah_stock |
+---------+-----------------------------------------------------+--------------+
|       1 | Principles and Practices of Interconnection Networks |           10 |
|       2 | Building DMZs for Enterprise Networks               |           20 |
|       3 | Virtual Private Networks                            |           11 |
|       4 | Computer Networking: a Top Down Approach            |            4 |
|       5 | Computer Networking Illuminated                     |            7 |
+---------+-----------------------------------------------------+--------------+
5 rows in set (0.000 sec)
```

Menampilkan Seluruh Data Beberapa Kolom :

Syntax: SELECT nama_kolom_1,…, nama_kolom_n FROM nama_tabel;

Contoh: SELECT judul_buku FROM stock_buku;

```
MariaDB [kuliah]> select judul_buku from stock_buku;
+-----------------------------------------------------+
| judul_buku                                          |
+-----------------------------------------------------+
| Principles and Practices of Interconnection Networks |
| Building DMZs for Enterprise Networks               |
| Virtual Private Networks                            |
| Computer Networking: a Top Down Approach            |
| Computer Networking Illuminated                     |
+-----------------------------------------------------+
5 rows in set (0.000 sec)
```

Membatasi Jumlah Baris Data yang Ditampilkan (LIMIT)

Syntax: SELECT * FROM nama_tabel LIMIT baris_awal,jumlah_baris;

Contoh:

- SELECT id_buku, judul_buku FROM stock_buku LIMIT 2;
- SELECT id_buku, judul_buku FROM stock_buku LIMIT 2, 2;

```
MariaDB [kuliah]> select id_buku, judul_buku from stock_buku limit 2;
+---------+-----------------------------------------------------+
| id_buku | judul_buku                                          |
+---------+-----------------------------------------------------+
|       1 | Principles and Practices of Interconnection Networks |
|       2 | Building DMZs for Enterprise Networks               |
+---------+-----------------------------------------------------+
2 rows in set (0.000 sec)
```

```
MariaDB [kuliah]> select id_buku, judul_buku from stock_buku limit 2,2;
+---------+-----------------------------------------+
| id_buku | judul_buku                              |
+---------+-----------------------------------------+
|       3 | Virtual Private Networks                |
|       4 | Computer Networking: a Top Down Approach |
+---------+-----------------------------------------+
2 rows in set (0.000 sec)
```

Menampilkan Sebagian Data Berdasarkan Kondisi Tertentu (WHERE)

Syntax: SELECT * FROM nama_tabel WHERE kondisi;

Contoh: SELECT * FROM stock_buku WHERE jumlah_stock > 10;

```
MariaDB [kuliah]> select * from stock_buku where jumlah_stock > 10;
+---------+-----------------------------------------+--------------+
| id_buku | judul_buku                              | jumlah_stock |
+---------+-----------------------------------------+--------------+
|       2 | Building DMZs for Enterprise Networks   |           20 |
|       3 | Virtual Private Networks                |           11 |
+---------+-----------------------------------------+--------------+
2 rows in set (0.000 sec)
```

Menampilkan Sebagian Data Berdasarkan Kondisi Tidak Sama Dengan (<>)

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom <> kondisi;

Contoh: SELECT * FROM stock_buku WHERE id_buku <> 3;

```
MariaDB [kuliah]> select * from stock_buku where id_buku <>  3;
+---------+-----------------------------------------------------+--------------+
| id_buku | judul_buku                                          | jumlah_stock |
+---------+-----------------------------------------------------+--------------+
|       1 | Principles and Practices of Interconnection Networks |           10 |
|       2 | Building DMZs for Enterprise Networks               |           20 |
|       4 | Computer Networking: a Top Down Approach            |            4 |
|       5 | Computer Networking Illuminated                     |            7 |
+---------+-----------------------------------------------------+--------------+
4 rows in set (0.000 sec)
```

Menampilkan Sebagian Data Berdasarkan 2 Atau Lebih Kondisi (AND)

Syntax: SELECT * FROM nama_tabel WHERE kondisi_1 AND kondisi_2, … kondisi_n;

Contoh: SELECT * FROM stock_buku WHERE id_buku >=2 AND jumlah_stock > 4;

```
MariaDB [kuliah]> select * from stock_buku where id_buku >= 2 AND jumlah_stock > 4;
+---------+-----------------------------------------+--------------+
| id_buku | judul_buku                              | jumlah_stock |
+---------+-----------------------------------------+--------------+
|       2 | Building DMZs for Enterprise Networks   |           20 |
|       3 | Virtual Private Networks                |           11 |
|       5 | Computer Networking Illuminated         |            7 |
+---------+-----------------------------------------+--------------+
3 rows in set (0.001 sec)
```

Menampilkan Sebagian Data Berdasarkan 2 Atau Lebih Kondisi (OR)

Syntax: SELECT * FROM nama_tabel WHERE kondisi_1 OR kondisi_2, … kondisi_n;

Contoh: SELECT * FROM stock_buku WHERE id_buku >=2 OR jumlah_stock > 50;

```
MariaDB [kuliah]> select * from stock_buku where id_buku >= 2 OR jumlah_stock > 50;
+---------+-----------------------------------------+--------------+
| id_buku | judul_buku                              | jumlah_stock |
+---------+-----------------------------------------+--------------+
|       2 | Building DMZs for Enterprise Networks   |           20 |
|       3 | Virtual Private Networks                |           11 |
|       4 | Computer Networking: a Top Down Approach |            4 |
|       5 | Computer Networking Illuminated         |            7 |
+---------+-----------------------------------------+--------------+
4 rows in set (0.001 sec)
```

Menampilkan Sebagian Data Berdasarkan 2 Atau Lebih Kondisi (AND OR)

Syntax: SELECT * FROM nama_tabel WHERE kondisi_1 AND kondisi_2, … OR kondisi_n;

Contoh: SELECT * FROM stock_buku WHERE (id_buku = 2 AND judul_buku = ‘Building DMZs for Enterprise Networks’) OR (jumlah_stock < 10);

```
MariaDB [kuliah]> select * from stock_buku where (id_buku = 2 and judul_buku = 'Building DMZs for Enterprise Networks') or (jumlah_stoc
k < 10);
+---------+-----------------------------------------+--------------+
| id_buku | judul_buku                              | jumlah_stock |
+---------+-----------------------------------------+--------------+
|       2 | Building DMZs for Enterprise Networks   |           20 |
|       4 | Computer Networking: a Top Down Approach |            4 |
|       5 | Computer Networking Illuminated         |            7 |
+---------+-----------------------------------------+--------------+
3 rows in set (0.000 sec)
```

Menampilkan Sebagian Data Berdasarkan Beberapa Nilai Tertentu (IN)

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom IN (nilai_1, nilai_2, … , nilai_n);

Contoh: SELECT * FROM stock_buku WHERE id_buku IN (1, 3, 5);

```
MariaDB [kuliah]> select * from stock_buku where id_buku IN (1, 3, 5);
+---------+-----------------------------------------------------------+--------------+
| id_buku | judul_buku                                                | jumlah_stock |
+---------+-----------------------------------------------------------+--------------+
|       1 | Principles and Practices of Interconnection Networks      |           10 |
|       3 | Virtual Private Networks                                  |           11 |
|       5 | Computer Networking Illuminated                           |            7 |
+---------+-----------------------------------------------------------+--------------+
3 rows in set (0.000 sec)
```

Menampilkan Sebagian Data Berdasarkan Rentang Nilai Tertentu (BETWEEN)

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom BETWEEN nilai_1 AND nilai_2;

Contoh: SELECT id_buku, judul_buku FROM stock_buku WHERE id_buku BETWEEN 2 AND 4;

```
MariaDB [kuliah]> select id_buku, judul_buku from stock_buku where id_buku between 2 and 4;
+---------+----------------------------------------+
| id_buku | judul_buku                             |
+---------+----------------------------------------+
|       2 | Building DMZs for Enterprise Networks  |
|       3 | Virtual Private Networks               |
|       4 | Computer Networking: a Top Down Approach |
+---------+----------------------------------------+
3 rows in set (0.001 sec)
```

Menampilkan Sebagian Data Berdasarkan Kondisi Tertentu (NOT)

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom NOT kondisi;

Contoh: SELECT id_buku, judul_buku FROM stock_buku WHERE id_buku NOT BETWEEN 2 AND 4;

```
MariaDB [kuliah]> select id_buku, judul_buku from stock_buku where id_buku not between 2 and 4;
+---------+-----------------------------------------------------+
| id_buku | judul_buku                                          |
+---------+-----------------------------------------------------+
|       1 | Principles and Practices of Interconnection Networks |
|       5 | Computer Networking Illuminated                     |
+---------+-----------------------------------------------------+
2 rows in set (0.000 sec)
```

Menampilkan Semua Data yang Bernilai NULL

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom IS NULL;

Contoh: SELECT * FROM stock_buku WHERE jumlah_stock IS NULL;

```
MariaDB [kuliah]> select * from stock_buku;
+---------+-----------------------------------------------------+--------------+
| id_buku | judul_buku                                          | jumlah_stock |
+---------+-----------------------------------------------------+--------------+
|       1 | Principles and Practices of Interconnection Networks |           10 |
|       2 | Building DMZs for Enterprise Networks               |           20 |
|       3 | Virtual Private Networks                            |           11 |
|       4 | Computer Networking: a Top Down Approach            |            4 |
|       5 | Computer Networking Illuminated                     |            7 |
|       6 | Network Routing Algorithms, Protocols and Architectures |         NULL |
+---------+-----------------------------------------------------+--------------+
6 rows in set (0.000 sec)

MariaDB [kuliah]> select * from stock_buku where jumlah_stock is null;
+---------+-----------------------------------------------------+--------------+
| id_buku | judul_buku                                          | jumlah_stock |
+---------+-----------------------------------------------------+--------------+
|       6 | Network Routing Algorithms, Protocols and Architectures |         NULL |
+---------+-----------------------------------------------------+--------------+
1 row in set (0.053 sec)
```

Menampilkan Semua Data yang Bernilai NOT NULL

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom IS NOT NULL;

Contoh: SELECT * FROM stock_buku WHERE jumlah_stock IS NOT NULL;

```
MariaDB [kuliah]> select * from stock_buku where jumlah_stock is not null;
+---------+-----------------------------------------------------+--------------+
| id_buku | judul_buku                                          | jumlah_stock |
+---------+-----------------------------------------------------+--------------+
|       1 | Principles and Practices of Interconnection Networks |           10 |
|       2 | Building DMZs for Enterprise Networks               |           20 |
|       3 | Virtual Private Networks                            |           11 |
|       4 | Computer Networking: a Top Down Approach            |            4 |
|       5 | Computer Networking Illuminated                     |            7 |
+---------+-----------------------------------------------------+--------------+
5 rows in set (0.000 sec)
```

Menampilkan Semua Data yang Unik (Distinct)

Syntax: SELECT DISTINCT nama_kolom_1, … nama_kolom_n FROM nama_tabel;

Contoh: SELECT DISTINCT jumlah_stock FROM stock_buku;

```
MariaDB [kuliah]> select * from stock_buku;
+---------+-----------------------------------------------------+--------------+
| id_buku | judul_buku                                          | jumlah_stock |
+---------+-----------------------------------------------------+--------------+
|       1 | Principles and Practices of Interconnection Networks |           10 |
|       2 | Building DMZs for Enterprise Networks               |           20 |
|       3 | Virtual Private Networks                            |           11 |
|       4 | Computer Networking: a Top Down Approach            |            4 |
|       5 | Computer Networking Illuminated                     |            7 |
|       6 | Network Routing Algorithms, Protocols and Architectures |           20 |
+---------+-----------------------------------------------------+--------------+
6 rows in set (0.000 sec)

MariaDB [kuliah]> select distinct jumlah_stock from stock_buku;
+--------------+
| jumlah_stock |
+--------------+
|           10 |
|           20 |
|           11 |
|            4 |
|            7 |
+--------------+
5 rows in set (0.000 sec)
```

Menampilkan Semua Data Berdasarkan Pola Tertentu (LIKE)

Syntax: SELECT * FROM nama_tabel WHERE nama_kolom LIKE pola;

Wildcard yang digunakan dalam membuat pola adalah:

- '%': pola berdasarkan 0 atau lebih karakter tertentu
- '_': pola yang nilainya terdiri dari satu karakter

WHERE judul_buku LIKE 'Com%'	Menampilkan semua nilai yang dimulai dengan "Com"
WHERE judul_buku LIKE '%s'	Menampilkan semua nilai yang diakhiri dengan "s"
WHERE judul_buku LIKE 'c%h'	Menampilkan semua nilai yang dimulai dengan "c" dan diakhiri dengan "h"
WHERE judul_buku LIKE '%ks%'	Menampilkan semua nilai "ks" di semua posisi
WHERE jumlah_stock LIKE '_'	Menampilkan semua nilai yang jumlah karakternya 1
WHERE jumlah_stock LIKE '__'	Menampilkan semua nilai yang jumlah karakternya 2
WHERE jumlah_stock LIKE '2_%'	Menampilkan semua nilai yang jumlah karakternya 2 dan dimulai dengan "2"
WHERE jumlah_stock LIKE '_0%'	Menampilkan semua nilai yang jumlah karakternya 2 dan nilai di posisi kedua adalah "0"

Menampilkan Semua Data Berdasarkan Pola Tertentu (LIKE) (Lanjutan)

```
MariaDB [kuliah]> select judul_buku from stock_buku where judul_buku like 'Com%';
+----------------------------------------+
| judul_buku                             |
+----------------------------------------+
| Computer Networking: a Top Down Approach |
| Computer Networking Illuminated        |
+----------------------------------------+
2 rows in set (0.000 sec)
```

```
MariaDB [kuliah]> select judul_buku from stock_buku where judul_buku like '%s';
+-----------------------------------------------------------+
| judul_buku                                                |
+-----------------------------------------------------------+
| Principles and Practices of Interconnection Networks      |
| Building DMZs for Enterprise Networks                     |
| Virtual Private Networks                                  |
| Network Routing Algorithms, Protocols and Architectures   |
+-----------------------------------------------------------+
4 rows in set (0.000 sec)
```

Menampilkan Semua Data Berdasarkan Pola Tertentu (LIKE) (Lanjutan)

```
MariaDB [kuliah]> select judul_buku from stock_buku where judul_buku like 'c%h';
+----------------------------------------+
| judul_buku                             |
+----------------------------------------+
| Computer Networking: a Top Down Approach |
+----------------------------------------+
1 row in set (0.000 sec)
```

```
MariaDB [kuliah]> select judul_buku from stock_buku where judul_buku like '%ks%';
+-------------------------------------------------+
| judul_buku                                      |
+-------------------------------------------------+
| Principles and Practices of Interconnection Networks |
| Building DMZs for Enterprise Networks           |
| Virtual Private Networks                        |
+-------------------------------------------------+
3 rows in set (0.000 sec)
```

Menampilkan Semua Data Berdasarkan Pola Tertentu (LIKE) (Lanjutan)

```
MariaDB [kuliah]> select * from stock_buku where jumlah_stock like '_';
+---------+----------------------------------------+--------------+
| id_buku | judul_buku                             | jumlah_stock |
+---------+----------------------------------------+--------------+
|       4 | Computer Networking: a Top Down Approach |            4 |
|       5 | Computer Networking Illuminated        |            7 |
+---------+----------------------------------------+--------------+
2 rows in set (0.000 sec)
```

```
MariaDB [kuliah]> select jumlah_stock from stock_buku where jumlah_stock like '__';
+--------------+
| jumlah_stock |
+--------------+
|           10 |
|           20 |
|           11 |
|           20 |
+--------------+
4 rows in set (0.000 sec)
```

Menampilkan Semua Data Berdasarkan Pola Tertentu (LIKE) (Lanjutan)

```
MariaDB [kuliah]> select jumlah_stock from stock_buku where jumlah_stock like '2_%';
+--------------+
| jumlah_stock |
+--------------+
|           20 |
|           20 |
+--------------+
2 rows in set (0.000 sec)
```

```
MariaDB [kuliah]> select jumlah_stock from stock_buku where jumlah_stock like '_0%';
+--------------+
| jumlah_stock |
+--------------+
|           10 |
|           20 |
|           20 |
+--------------+
3 rows in set (0.000 sec)
```

Menampilkan Semua Data Sesuai Urutan (ORDER BY)

Mengurutkan data dari kecil ke besar

- Syntax: SELECT * FROM nama_tabel ORDER BY nama_kolom ASC;
- Contoh: SELECT * FROM stock_buku ORDER BY jumlah_stock ASC;

```
MariaDB [kuliah]> select * from stock_buku order by jumlah_stock asc;
+---------+-------------------------------------------------------+--------------+
| id_buku | judul_buku                                            | jumlah_stock |
+---------+-------------------------------------------------------+--------------+
|       4 | Computer Networking: a Top Down Approach              |            4 |
|       5 | Computer Networking Illuminated                       |            7 |
|       1 | Principles and Practices of Interconnection Networks  |           10 |
|       3 | Virtual Private Networks                              |           11 |
|       2 | Building DMZs for Enterprise Networks                 |           20 |
|       6 | Network Routing Algorithms, Protocols and Architectures |           20 |
+---------+-------------------------------------------------------+--------------+
```

Mengurutkan data dari besar ke kecil

- Syntax: SELECT * FROM nama_tabel ORDER BY nama_kolom DESC;
- Contoh: SELECT id_buku, jumlah_stock FROM stock_buku WHERE id_buku > 2 ORDER BY jumlah_stock DESC;

```
MariaDB [kuliah]> select id_buku, jumlah_stock from stock_buku where id_buku > 2 order by jumlah_stock desc;
+---------+--------------+
| id_buku | jumlah_stock |
+---------+--------------+
|       6 |           20 |
|       3 |           11 |
|       5 |            7 |
|       4 |            4 |
+---------+--------------+
4 rows in set (0.000 sec)
```