Bab III
Database

3.1 DBMS

Untuk keperluan penyimpanan data dibuatlah database. DBMS (Database Management System) adalah perangkat lunak yang digunakan untuk mengelola kumpulan atau koleksi data, di mana data tersebut diorganisasikan atau disusun ke dalam suatu model data. Adapun fungsi DBMS adalah untuk penyimpanan, pengambilan, dan perubahan data.

Data adalah komponen yang diolah dan disimpan sehingga menghasilkan informasi dan dapat dijadikan sebagai bahan pengambilan keputusan.

Struktur data adalah suatu koleksi atau kelompok data yang dapat dikarakteristikkan oleh organisasi serta operasi yang didefenisikan terhadapnya. Struktur data terdiri dari data sederhana dan majemuk.

Contoh struktur data sederhana adalah array dan record. Adapun struktur data majemuk terdiri dari linear dan non linear. Contoh linear adalah stack (tumpukan) dan queue (antrian). Adapun contoh non linear adalah graph, pohon (tree), pohon biner (binary tree), pohon cari biner (binary search tree).

Sebuah DBMS seringkali menyembunyikan detail tentang bagaimana data disimpan dan dikelola dalam sebuah database, dengan tujuan agar memudahkan pengguna dalam menggunakan DBMS tersebut. Tingkatan itu terdiri dari:

a. Level fisik (physical level)

   Merupakan level abstraksi data paling rendah, yang menggambarkan bagaimana data disimpan dalam kondisi sebenarnya. Level ini sangat kompleks karena struktur data dijelaskan secara rinci.

b. Level konseptual (conceptual level)

   Level ini menggambarkan data apa yang disimpan dalam database dan menjelaskan bagaimana hubungan antar data secara keseluruhan. Seorang pengguna dalam level ini dapat mengetahui bahwa data mahasiswa disimpan

pada tabel mahasiswa, tabel KRS, tabel transkrip dan lain sebagainya. Level ini biasa dipakai oleh seorang Database Administrator (DBA).

c. Level pandangan (view level)

   Merupakan level abstraksi data paling tinggi, yang menggambarkan atau menampilkan sebagian dari keseluruhan database, disesuaikan dengan kebutuhan pengguna.

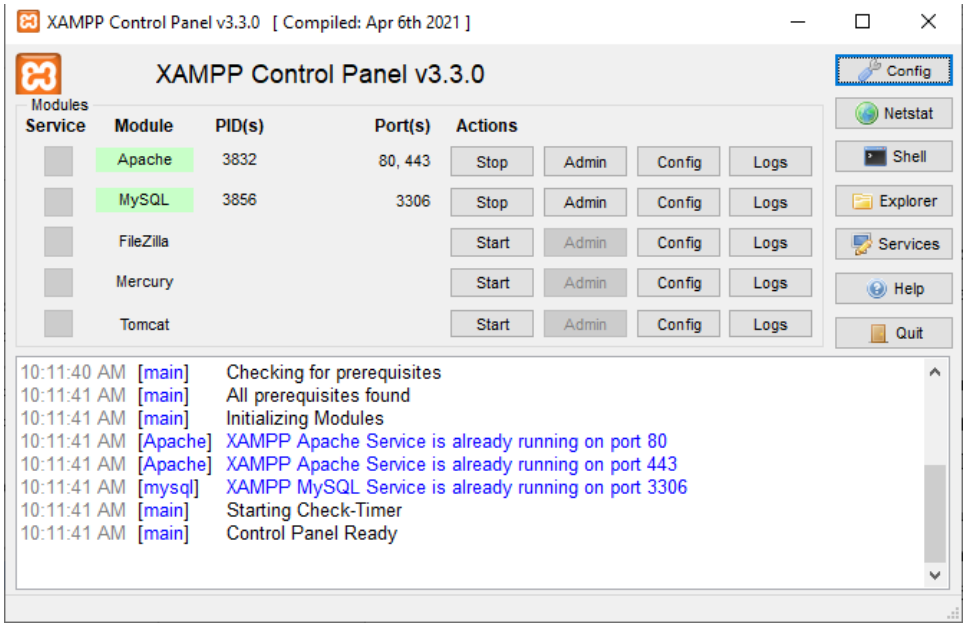
3.2 Software Database

Database server berfungsi untuk menyimpan data, melayani permintaan data, control recovery dan lainnya. Terdapat banyak database server, misalnya MySQL, MariaDB, Oracle DB Server, IBM DB2, MS SQL Server, Firebird, dan PostgreSQL. Ada yang hanya berjalan di atas OS tertentu, seperti MS SQL Server di atas MS Windows, tetapi ada pula yang multi platform, yaitu MySQL, Oracle, Firebird, dan PostgreSQL.

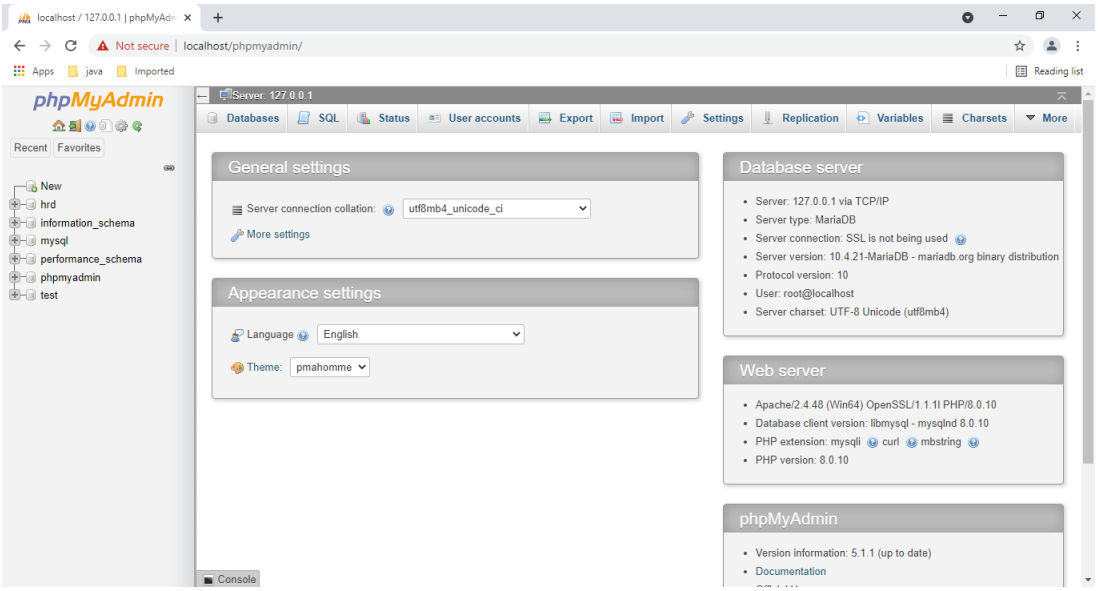
MySQL adalah RDBMS (Relational Database Management System), multi-threaded dan open source yang diciptakan pertama kali oleh Michael “Monty” Widenius.

Setelah diakusisi oleh SUN Micro, dan akhirnya jatuh ke Oracle, akhirnya Michael Monty keluar dari perusahaan tersebut dan mengembangkan lanjutan dari MySQL yang disebut dengan nama baru MariaDB. Meskipun demikian, semua yang ada di MySQL adalah 100% kompatibel dengan MariaDB, sejauh ini hanya nama saja yang berbeda. Pada materi ini, bila disebut MariaDB maka yang dimaksud adalah keduanya, MariaDB dan MySQL.

Untuk software instalasi MySQL di lab ini digunakan XAMPP (cross-platform, Apache, MySQL, PHP and Perl). Adapun services yang dihidupkan hanya Apache dan MySQL.

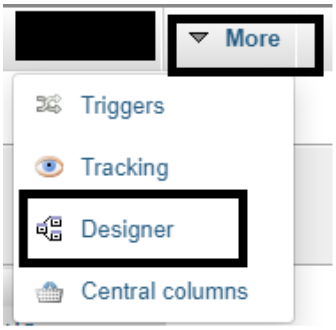


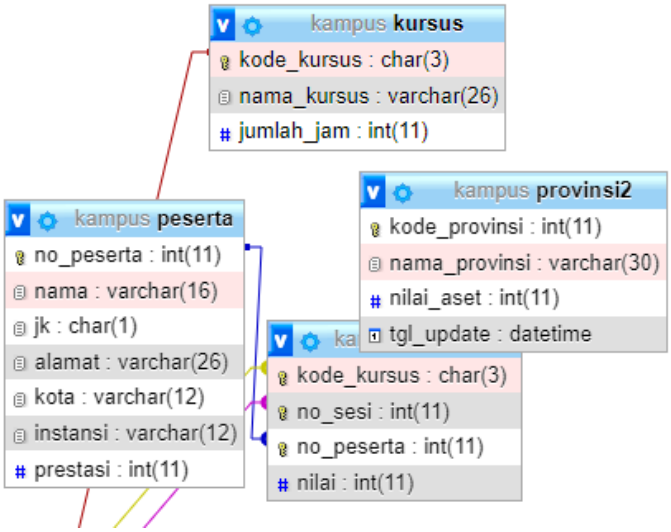
Software database client bawaan dari XAMPP adalah phpMyAdmin. Cara mengaksesnya dengan https://localhost/phymyadmin.



Untuk melihat desain database pada DB Client phpMyAdmin, caranya:

a. Pilih database, misal kampus
b. Pilih menu lainnya (more) → Designer





Untuk stop dan start service MySQL, selain lewat XAMPP bisa juga lewat command prompt:

```
C:\> net stop mysql
```

Atau di linux:

```
# service mysql restart
```

Ada juga software database client lain, misalnya Navicat, Toad, dan MySQL Workbench.

Mesin penyimpanan (storage engine) adalah sistem manajemen modul perangkat lunak yang digunakan untuk membuat, membaca, dan meng-update data dari database, alias tempat penyimpanan data. Ada dua tipe storage engine di MySQL, yaitu transactional dan non-transactional.

Dua storage engine yang sangat populer saat ini di MySQL adalah InnoDB dan MyISAM. Jika ada proses transaksi, pilih InnoDB. InnoDB adalah storage engine standar untuk MySQL versi 5.5 ke atas

Sebelum membuat object database--table, view, function, dan procedure--perlu disiapkan tempat penampungan dulu. Di sistem operasi, tempat ini disebut folder/direktori. Di MySQL, tempat ini disebut database. Database secara umum dibagi dua, yaitu:

- Milik sistem, contohnya information_schema, performance_schema, phpmyadmin, dan mysql
- Milik user, contohnya hrd, marketing, dan produksi

MySQL mempunyai system catalog atau data dictionary yang terletak di INFORMATION_SCHEMA. Sistem ini dapat digambarkan sebagai kamus data sistem terperinci yang menjelaskan semua objek di dalam database, termasuk data tentang nama tabel, pencipta tabel dan tanggal pembuatan, jumlah kolom di setiap tabel, tipe data yang sesuai dengan setiap kolom, nama file indeks, pembuat indeks, pengguna resmi, dan hak akses.

```
MariaDB [kampus]> use information_schema;

show tables;

SELECT CONCAT(table_schema,'.',table_name) as
table_name, table_rows
FROM information_schema.tables
WHERE table_rows > 5
AND table_schema not
in('information_schema','mysql','performance_schema');
ORDER BY table_rows desc;
```

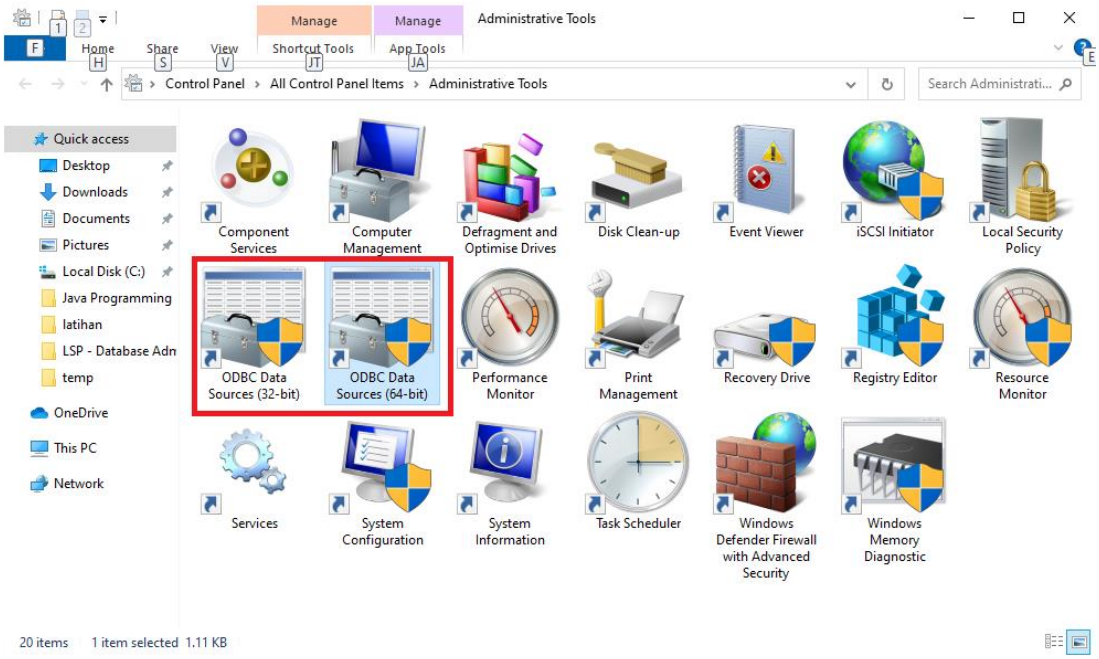
```
MariaDB [information_schema]> SELECT CONCAT(table_schema,'.',table_name) as table_name, table_rows
    -> FROM information_schema.tables
    -> WHERE table_rows > 5
    -> AND table_schema not in('information_schema','mysql','performance_schema');
+-------------------+------------+
| table_name        | table_rows |
+-------------------+------------+
| hrd.pendaftar     |         21 |
| kampus.instruktur |          7 |
| kampus.kabupaten  |          8 |
| kampus.kursus     |         19 |
| kampus.pendaftar  |         21 |
| kampus.pengajar   |         10 |
| kampus.peserta    |          7 |
| kampus.provinsi   |          6 |
| kampus.sesi       |         11 |
+-------------------+------------+
9 rows in set (0.005 sec)
```

3.3 Program Akses Database

Database server tidak secara langsung berinteraksi dengan user. Oleh karena itu dikenalkan konsep Client Server, yang mempunyai karakteristik di antaranya:

- Layanan (service). server bertindak sebagai service provider.
- Berbagi sumber daya (sharing resource)
- Pesan berbasis komunikasi
- Mix-and-Match: perbedaan platform server dan client.

Untuk mengakses database dari pemrograman ada beberapa cara sesuai bahasa pemrograman yang digunakan, antara lain menggunakan MySQLi (MySQL Improved), PDO (PHP Data Object), OLEDB (Object Linking and Embedding, Database), dan ODBC (Open Database Connectivity) di Windows.



Adapun komponen utama ODBC adalah ODBC API, ODBC Driver Manager, ODBC Database Drive, dan Data Source.

Apabila koneksi database dilakukan via program web seperti PHP, maka dibutuhkan web server seperti Apache. Untuk administrasi beberapa web di dalam sebuah server, bisa dilakukan pemisahan dengan konsep virtual host (satu server terdiri dari beberapa web server dengan konsep pemisahan direktori). Sebagai contoh web www.abc.com, www.def.com, dan www.ghi.com terletak dalam satu mesin,

hanya berbeda direktori. Virtual host dilakukan dengan mengubah file /etc/hosts dan web server configuration.

Berikut ini beberapa hal singkat tentang pemrograman:

- Variabel, konstanta, loop
- If-Else yang mempunyai operator relasi: ==, <=, >=, !=
- Array, yang mempunyai karakteristik:
  - ✓ Berupa kumpulan nilai data
  - ✓ Bertipe data sama
  - ✓ Dapat diakses secara random

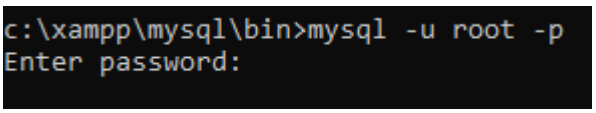
3.4 Create Database

Koneksi ke MySQL via command prompt sebagai berikut:

```
C:\xampp\mysql\bin>mysql -u root
```

Apabila diinginkan ada prompt password, tambahkan –p:

```
C:\xampp\mysql\bin>mysql -u root -p
```



Jika ternyata root tidak punya password, di prompt “Enter password:” cukup tekan <Enter>.

Cara membuat database dengan perintah:

```
MariaDB [(none)]> create database hrd;
```

Untuk menghapus database yang sudah pernah dibuat, dapat dilakukan dengan instruksi DROP.

```
MariaDB [(none)]> drop database hrd;
```

Untuk mengetahui ukuran database:

```
SELECT table_schema “Nama Database”,
SUM( data_length + index_length) / 1024 / 1024 “Ukuran
Database (MB)”
FROM information_schema.TABLES
GROUP BY table_schema;
```

```
MariaDB [marketing]> SELECT table_schema "Nama Database",
    -> SUM( data_length + index_length) / 1024 / 1024 "Ukuran Database (MB)"
    -> FROM information_schema.TABLES
    -> GROUP BY table_schema;
+--------------------+----------------------+
| Nama Database      | Ukuran Database (MB) |
+--------------------+----------------------+
| hrd                |           0.03125000 |
| information_schema |           0.20312500 |
| kampus             |           0.17187500 |
| marketing          |           0.03125000 |
| mysql              |           2.25000000 |
| performance_schema |           0.00000000 |
| phpmyadmin         |           0.39062500 |
+--------------------+----------------------+
7 rows in set (0.102 sec)
```

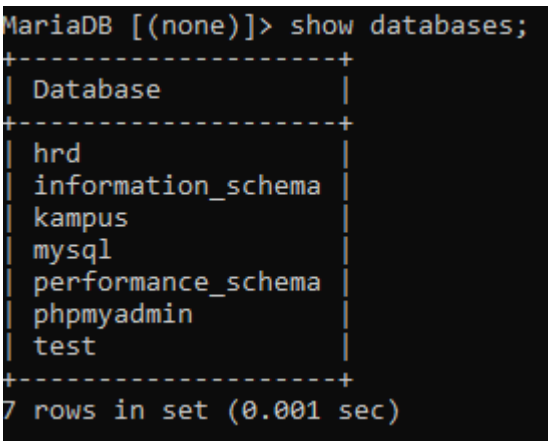
Bab IV
Table

Data disimpan di sebuah table. Adapun table berada di dalam database. Antar table bisa mengandung relasi untuk memastikan referential integrity (Foreign Key).

Sebagai key di sebuah table adalah Primary Key. Untuk mengurutkan data pada sebuah table digunakan clustered index. Primary Key termasuk clustered index.

Berikut ini contoh pembuatan duah buah table beserta relasinya, yang akan digunakan di lab selanjutnya.

```
C:\>cd c:\xampp\mysql\bin
c:\xampp\mysql\bin>mysql -u root
MariaDB [(none)]> create database kampus;
MariaDB [(none)]> show databases;
                -- menampilkan daftar database yang ada
```



```
use kampus;

CREATE TABLE provinsi (
kode_provinsi int Primary Key,
nama_provinsi varchar(30),
nilai_aset int);

CREATE TABLE kabupaten (
kode_kabupaten int Primary Key,
nama_kabupaten varchar(30),
jumlah_penduduk int,
kode_provinsi int,
CONSTRAINT fk_kabupaten_provinsi FOREIGN KEY
(kode_provinsi) references provinsi(kode_provinsi)
);

desc provinsi;
-- menampilkan informasi table provinsi

select database();
-- menampilkan database yang sedang diakses (kampus)

show tables;
-- menampilkan daftar table di database current (kampus)
```

```
MariaDB [kampus]> show tables;
+-----------------+
| Tables_in_kampus |
+-----------------+
| instruktur      |
| kabupaten       |
| kursus          |
| pendaftar       |
| peserta         |
| provinsi        |
| provinsi2       |
| sesi            |
| vprovinsi1      |
| vprovinsi2      |
+-----------------+
10 rows in set (0.001 sec)
```

```
check table provinsi;
-- menampilkan status sebuah table
```

```
MariaDB [kampus]> check table provinsi;
+-----------------+-------+----------+----------+
| Table           | Op    | Msg_type | Msg_text |
+-----------------+-------+----------+----------+
| kampus.provinsi | check | status   | OK       |
+-----------------+-------+----------+----------+
```

```
show open tables;
-- menampilkan status table locking
```

```
MariaDB [kampus]> show open tables;
+------------+---------------------------+--------+-------------+
| Database   | Table                     | In_use | Name_locked |
+------------+---------------------------+--------+-------------+
| kampus     | provinsi2                 |      0 |           0 |
| phpmyadmin | pma__table_uiprefs        |      0 |           0 |
| mysql      | servers                   |      0 |           0 |
| phpmyadmin | pma__users                |      0 |           0 |
| mysql      | procs_priv                |      0 |           0 |
| mysql      | table_stats               |      0 |           0 |
| mysql      | index_stats               |      0 |           0 |
| mysql      | plugin                    |      0 |           0 |
| phpmyadmin | pma__column_info          |      0 |           0 |
| kampus     | pendaftar                 |      0 |           0 |
| mysql      | time_zone_transition      |      0 |           0 |
| mysql      | time_zone_leap_second     |      0 |           0 |
| kampus     | kabupaten                 |      0 |           0 |
| mysql      | time_zone                 |      0 |           0 |
| kampus     | sesi                      |      0 |           0 |
| kampus     | kursus                    |      0 |           0 |
```

Untuk menghapus table digunakan perintah DROP <namatable>. Adapun untuk mengubah struktur table bisa digunakan perintah alter, contoh:

ALTER TABLE provinsi ADD tgl_update datetime;

Adapun contoh tipe data kolom sebagai berikut:

a. Integer

   Menampung bilangan bulat dan tidak mengandung pecahan.

b. Varchar

   Variable character, penyimpanan sesuai ukuran isi. Misal varchar(50), tetapi hanya diisi data “halo”, maka yang disimpan di hard disk hanya 4 bytes.

c. Char

Jumlah karakter yang disimpan sesuai deklarasi. Misal char(50), tetapi hanya diisi data “halo”, maka yang disimpan tetap 50 bytes dengan cara ditambah spasi.

Transaksi di sebuah table dikenal dengan istilah DML (Data Manipulation Language), yang terdiri atas:

- Insert: menambah baris (record) baru. Record adalah kumpulan data yang terdiri dari berbagai macam value.
- Update: mengubah data
- Delete: menghapus record

Setelah melakukan banyak proses DML, sebaiknya dilakukan optimalisasi tabel dengan menggunakan perintah:

optimize table <nama_tabel>;

4.1 Insert Data

```
insert into provinsi values(1,'Jawa Timur',3000);
insert into provinsi values(2,'Banten',500);
insert into provinsi values(3,'Jawa Barat',900);
insert into provinsi values(4,'Jawa Tengah',1000);
insert into provinsi values(5,'DI Yogyakarta',500);
insert into provinsi values(6,'DKI Jakarta',4000);

insert into kabupaten values(1,'Malang',600,1);
insert into kabupaten values(2,'Probolinggi',560,1);
insert into kabupaten values(3,'Madiun',300,1);
insert into kabupaten values(4,'Indramayu',700,3);
insert into kabupaten values(5,'Cianjur',400,3);
insert into kabupaten values(6,'Boyolali',550,4);
insert into kabupaten values(7,'Blora',350,null);
insert into kabupaten values(8,'Cirebon',650,null);
```

4.2 Update Data

Update kabupaten

Set nama_kabupaten='Probolinggo'

Where kode_kabupaten=2;

4.3 Hapus Data

```
insert into kabupaten values(9,'Kabupaten9',650,null);
select * from kabupaten;
-- Ada Kabupaten9

Delete from kabupaten where kode_kabupaten=9;
select * from kabupaten;
-- Kabupaten9 tidak ada
```

4.4 Query Data

4.4.1 Tanpa fungsi

```
SELECT * FROM provinsi;
SELECT nama_provinsi, nilai_aset FROM provinsi;
SELECT * FROM provinsi ORDER BY nilai_aset DESC;
SELECT * FROM kabupaten WHERE kode_kabupaten=1;
SELECT * FROM provinsi WHERE nilai_aset > 2000;
SELECT * FROM provinsi WHERE kode_provinsi = 1 OR
kode_provinsi=2;
SELECT * FROM kabupaten WHERE jumlah_penduduk > 500 AND
kode_provinsi = 1;
SELECT * FROM provinsi WHERE nilai_aset BETWEEN 900 AND
1000;
```

Perintah tersebut sama dengan:

```
SELECT * FROM provinsi WHERE nilai_aset >= 900 AND
nilai_aset <= 1000;
```

```
SELECT * FROM kabupaten WHERE kode_provinsi IS NOT NULL;
```

Klausa like digunakan untuk mencari data berdasarkan pola kemiripan.

```
SELECT * FROM provinsi WHERE nama_provinsi
Like 'J%';

SELECT * FROM kabupaten WHERE nama_kabupaten
Like 'MA____';
-- ada 4 buah garis bawah
```

4.4.2 Dengan fungsi dan rumus

```
SELECT DISTINCT (kode_provinsi) FROM kabupaten;
-- menghilangkan record yang sama (hanya tampil sekali)

SELECT UPPER(nama_provinsi) FROM provinsi;
SELECT 0.1 * nilai_aset FROM provinsi;
```

Untuk mendapatkan tanggal berapakah 2 hari setelah tanggal sistem saat ini:

```
SELECT day(now()) + 2;
```

Untuk mendapatkan info tanggal sistem saat ini:

```
SELECT sysdate();
SELECT now();
```

Untuk mendapatkan info waktu sistem saat ini:

```
SELECT time(now());
```

4.4.3 Agregrate & Group By

Agregate adalah fungsi yang menerima koleksi nilai dan mengembalikan nilai tunggal sebagai hasilnya.

```
SELECT MAX(nilai_aset) FROM provinsi;
```

```
MariaDB [kampus]> SELECT MAX(nilai_aset) FROM provinsi;
+-----------------+
| MAX(nilai_aset) |
+-----------------+
|           10000 |
+-----------------+
1 row in set (0.000 sec)
```

```
SELECT kode_provinsi, COUNT(kode_kabupaten)
FROM kabupaten
GROUP BY kode_provinsi;
```

```
MariaDB [kampus]> SELECT kode_provinsi, COUNT(kode_kabupaten)
    -> FROM kabupaten
    -> GROUP BY kode_provinsi;
+---------------+-----------------------+
| kode_provinsi | COUNT(kode_kabupaten) |
+---------------+-----------------------+
|          NULL |                     2 |
|             1 |                     3 |
|             3 |                     2 |
|             4 |                     1 |
+---------------+-----------------------+
4 rows in set (0.000 sec)
```

```
SELECT kode_provinsi, COUNT(kode_kabupaten) as jml
FROM kabupaten
GROUP BY kode_provinsi ORDER BY jml DESC;
```

```
MariaDB [kampus]> SELECT kode_provinsi, COUNT(kode_kabupaten) as jml
    -> FROM kabupaten
    -> GROUP BY kode_provinsi ORDER BY jml DESC;
+---------------+-----+
| kode_provinsi | jml |
+---------------+-----+
|             1 |   3 |
|          NULL |   2 |
|             3 |   2 |
|             4 |   1 |
+---------------+-----+
4 rows in set (0.001 sec)
```

```
SELECT kode_provinsi, COUNT(kode_kabupaten) as jml

FROM kabupaten

GROUP BY kode_provinsi

HAVING COUNT(kode_kabupaten) < 5;
```

```
MariaDB [kampus]> SELECT kode_provinsi, COUNT(kode_kabupaten) as jml
    -> FROM kabupaten
    -> GROUP BY kode_provinsi
    -> HAVING COUNT(kode_kabupaten) < 5;
+---------------+-----+
| kode_provinsi | jml |
+---------------+-----+
|          NULL |   2 |
|             1 |   3 |
|             3 |   2 |
|             4 |   1 |
+---------------+-----+
4 rows in set (0.001 sec)
```

4.4.4 Join dan Union

```
SELECT nama_kabupaten, nama_provinsi FROM kabupaten
INNER JOIN provinsi
on kabupaten.kode_provinsi = provinsi.kode_provinsi;
```

```
MariaDB [kampus]> SELECT nama_kabupaten, nama_provinsi FROM kabupaten INNER JOIN provinsi
    -> on kabupaten.kode_provinsi = provinsi.kode_provinsi;
+----------------+---------------+
| nama_kabupaten | nama_provinsi |
+----------------+---------------+
| Malang         | Jawa Timur    |
| Probolinggi    | Jawa Timur    |
| Madiun         | Jawa Timur    |
| Indramayu      | Jawa Barat    |
| Cianjur        | Jawa Barat    |
| Boyolali       | Jawa Tengah   |
+----------------+---------------+
6 rows in set (0.001 sec)
```

```
SELECT * FROM kabupaten LEFT OUTER JOIN provinsi
on kabupaten.kode_provinsi = provinsi.kode_provinsi;
```

```
MariaDB [kampus]> SELECT * FROM kabupaten LEFT OUTER JOIN provinsi
    -> on kabupaten.kode_provinsi = provinsi.kode_provinsi;
+---------------+----------------+-----------------+---------------+---------------+---------------+------------+
| kode_kabupaten | nama_kabupaten | jumlah_penduduk | kode_provinsi | kode_provinsi | nama_provinsi | nilai_aset |
+---------------+----------------+-----------------+---------------+---------------+---------------+------------+
|              1 | Malang         |             600 |             1 |             1 | Jawa Timur    |       3000 |
|              2 | Probolinggi    |             560 |             1 |             1 | Jawa Timur    |       3000 |
|              3 | Madiun         |             300 |             1 |             1 | Jawa Timur    |       3000 |
|              4 | Indramayu      |             700 |             3 |             3 | Jawa Barat    |        900 |
|              5 | Cianjur        |             400 |             3 |             3 | Jawa Barat    |        900 |
|              6 | Boyolali       |             550 |             4 |             4 | Jawa Tengah   |       1000 |
|              7 | Blora          |             350 |          NULL |          NULL | NULL          |       NULL |
|              8 | Cirebon        |             650 |          NULL |          NULL | NULL          |       NULL |
+---------------+----------------+-----------------+---------------+---------------+---------------+------------+
8 rows in set (0.001 sec)
```

```
SELECT * FROM kabupaten LEFT OUTER JOIN provinsi
on kabupaten.kode_provinsi = provinsi.kode_provinsi
WHERE provinsi.kode_provinsi = 1;
```

```
MariaDB [kampus]> SELECT * FROM kabupaten LEFT OUTER JOIN provinsi
    -> on kabupaten.kode_provinsi = provinsi.kode_provinsi
    -> WHERE provinsi.kode_provinsi = 1;
+---------------+---------------+----------------+--------------+--------------+--------------+------------+
| kode_kabupaten | nama_kabupaten | jumlah_penduduk | kode_provinsi | kode_provinsi | nama_provinsi | nilai_aset |
+---------------+---------------+----------------+--------------+--------------+--------------+------------+
|             1 | Malang         |            600 |            1 |            1 | Jawa Timur    |       3000 |
|             2 | Probolinggi    |            560 |            1 |            1 | Jawa Timur    |       3000 |
|             3 | Madiun         |            300 |            1 |            1 | Jawa Timur    |       3000 |
+---------------+---------------+----------------+--------------+--------------+--------------+------------+
3 rows in set (0.001 sec)
```

```
SELECT nama_kabupaten FROM kabupaten
UNION
SELECT nama_provinsi FROM provinsi;
```

```
MariaDB [kampus]> SELECT nama_kabupaten FROM kabupaten
    -> UNION
    -> SELECT nama_provinsi FROM provinsi;
+------------------+
| nama_kabupaten   |
+------------------+
| Malang           |
| Probolinggi      |
| Madiun           |
| Indramayu        |
| Cianjur          |
| Boyolali         |
| Blora            |
| Cirebon          |
| Jawa Timur       |
| Banten           |
| Jawa Barat       |
| Jawa Tengah      |
| DI Yogyakarta    |
| DKI Jakarta      |
| Kalimantan Timur |
+------------------+
15 rows in set (0.001 sec)
```

4.4.5 Sub Query

```
SELECT * FROM provinsi WHERE nilai_aset = (SELECT
MIN(nilai_aset) FROM provinsi);
```

```
MariaDB [kampus]> SELECT * FROM provinsi WHERE nilai_aset = (SELECT MIN(nilai_aset) FROM provinsi);
+--------------+--------------+------------+
| kode_provinsi | nama_provinsi | nilai_aset |
+--------------+--------------+------------+
|            2 | Banten       |        500 |
|            5 | DI Yogyakarta |        500 |
+--------------+--------------+------------+
2 rows in set (0.000 sec)
```

```
SELECT * FROM kabupaten
WHERE jumlah_penduduk >
(SELECT AVG(jumlah_penduduk) FROM kabupaten);
```

```
MariaDB [kampus]> SELECT * FROM kabupaten
    -> WHERE jumlah_penduduk >
    -> (SELECT AVG(jumlah_penduduk) FROM kabupaten);
+----------------+----------------+-----------------+---------------+
| kode_kabupaten | nama_kabupaten | jumlah_penduduk | kode_provinsi |
+----------------+----------------+-----------------+---------------+
|              1 | Malang         |             600 |             1 |
|              2 | Probolinggi    |             560 |             1 |
|              4 | Indramayu      |             700 |             3 |
|              6 | Boyolali       |             550 |             4 |
|              8 | Cirebon        |             650 |          NULL |
+----------------+----------------+-----------------+---------------+
5 rows in set (0.001 sec)
```

```
SELECT COUNT(*) FROM Kabupaten WHERE
jumlah_penduduk > (SELECT AVG(jumlah_penduduk) FROM
kabupaten);
```

```
MariaDB [kampus]> SELECT COUNT(*) FROM Kabupaten WHERE
    -> jumlah_penduduk > (SELECT AVG(jumlah_penduduk) FROM kabupaten);
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.001 sec)
```