Modul

Penyusun:

1. Dwi Marlina, M.Kom.
2. Rini Amalia, S.Kom., M.MSI.
3. Siti Khotijah, S.Kom., M.MSI.

PRAKTIKUM

SISTEM BASIS DATA

PRAKATA

Modul Praktikum Sistem Basis Data merupakan sarana penunjang perkuliahan yang digunakan mahasiswa untuk membantu dalam proses belajar yaitu pada mata kuliah praktikum struktur data. Dengan modul praktikum ini diharapkan mahasiswa dapat lebih mudah memahami materi yang dipelajari di dalam laboratorium komputer.

Modul praktikum ini disusun dengan memperhatikan kebutuhan dasar pengetahuan database yang perlu dipahami oleh setiap mahasiswa yang mempelajari mata kuliah Praktikum Sistem Basis Data.

Modul ini digunakan sebagai pegangan mahasiswa saat praktek di Laboratorium komputer agar dapat membantu mahasiswa saat praktek berlangsung karena pada modul ini terdapat contoh-contoh bahasa sql yang mudah untuk dipahami oleh mahasiswa. Semoga modul ini bermanfaat bagi pengtgunanya.

DAFTAR ISI

DAFTAR ISI ii
PRAKATA iii
BAB 1 PENGENALAN APLIKASI XAMPP 1
BAB 2 PENGENALAN SQL 5
BAB 3 DDL 9
BAB 4 DML Bagian 1..... 16
BAB 5 DML Bagian 2..... 21
BAB 6 FUNGSI AGREGAT 28
BAB 7 FUNGSI TANGGAL 34
BAB 8 FUNGSI STRING 42
BAB 9 DATABASE RELATION 47
BAB 10 RELASI 2 TABEL 52
BAB 11 RELASI 3 TABEL 57
BAB 12 RELASI DENGAN JOIN 59
BAB 13 UNION, INTERSECT, EXCEPT 64

BAB 1
MENGENAL APLIKASI XAMPP

A. Aplikasi Xampp

Xampp adalah perangkat yang menggabungkan tiga aplikasi kedalam satu paket, yaitu Apache, MySQL, dan PHPMyAdmin. Xampp adalah perangkat yang menggabungkan empat aplikasi kedalam satu paket, yaitu Apache, MySQL, PHPMyAdmin, dan Perl

1. Apache

   Apache adalah sebuah web server open source, jadi semua orang dapat menggunakannya secara gratis, dapat mengedit kode programnya. Fungsi utama dari Apache yakni menghasilkan halaman web yang benar sesuai dengan yang dibuat oleh seorang web programmer, dengan menggunakan kode PHP.

2. MySQL

   Apache adalah sebuah web server open source, jadi semua orang dapat menggunakannya secara gratis, dapat mengedit kode programnya. Fungsi utama dari Apache yakni menghasilkan halaman web yang benar sesuai dengan yang dibuat oleh seorang web programmer, dengan menggunakan kode PHP.

3. PHP

   PHP adalah bahasa pemograman untuk membuat web. PHP dapat membuat halaman web yang dinamis. Selain mendukung di sistem operasi Windows, PHP juga dapat di gunakan pada mac OS, Linux, dan sistem operasi yang lainnya.

4. Perl

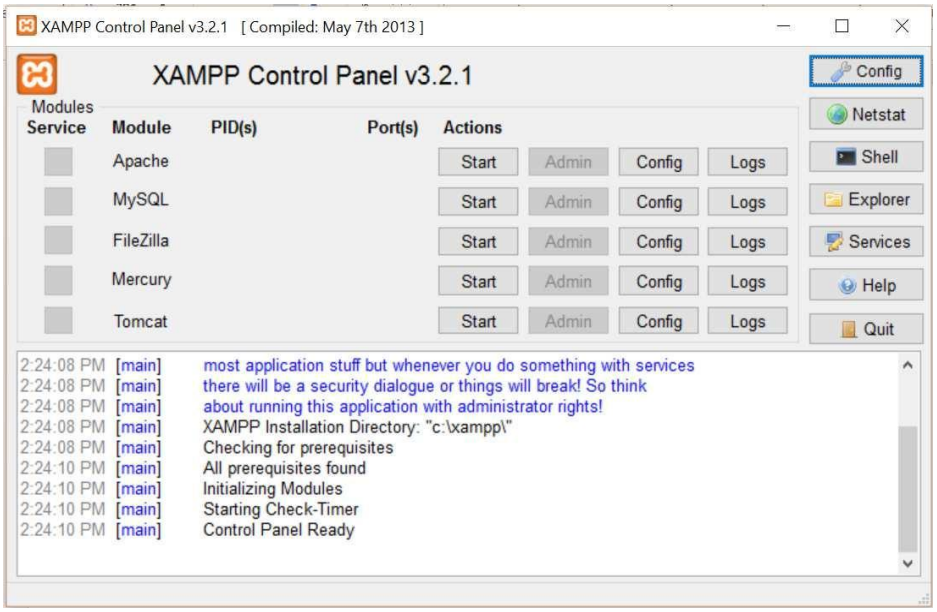
   Perl adalah bahasa pemrograman untuk semua tujuan, pertama kali dikembangkan oleh Larry Wall, mesin Unix. Perl dirilis pertama kali tanggal 18 Desember 1987 yang ditandai dengan keluarnya Perl 1. Pada versi-versi selanjutnya, Perl juga tersedia untuk berbagai sistem operasi

Unix (SunOS, Linux, BSD, HP-UX), juga tersedia untuk sistem operasi seperti DOS, Windows, PowerPC, BeOS, VMS, EBCDIC, dan PocketPC.

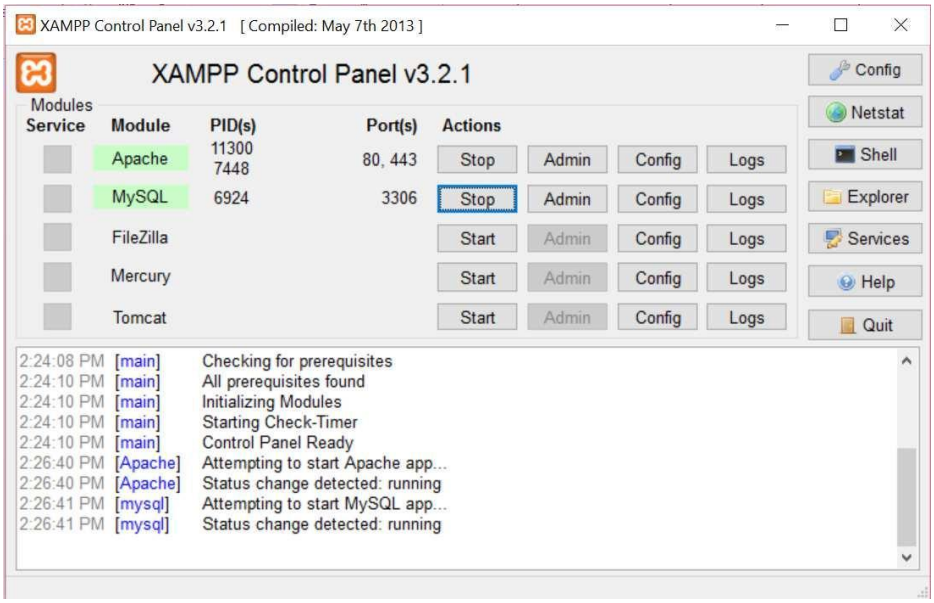
B. Langkah-langkah Praktikum Sistem Sistem Basis Data

Langkah-langkah yang harus dilakukan untuk memulai praktikum sistem basis data adalah sebagai berikut:

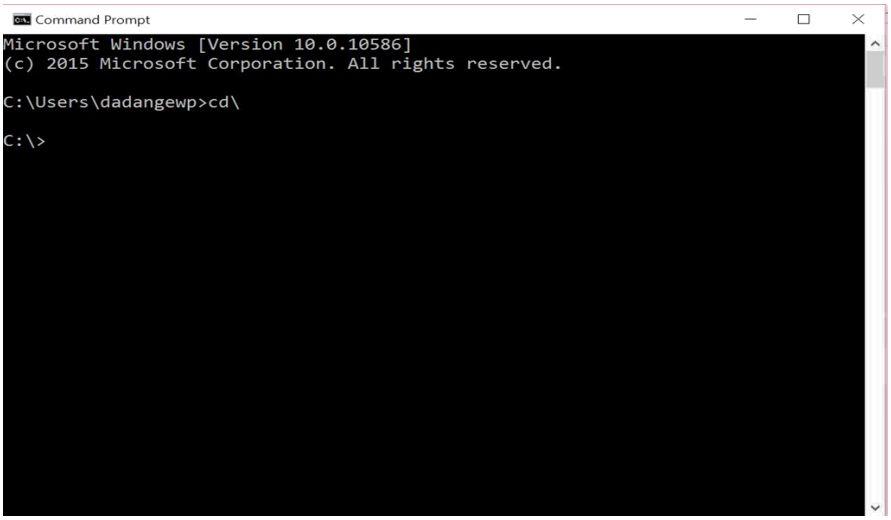
1. Mengakses PhpMyAdmin
2. Buka Xampp Control Panel



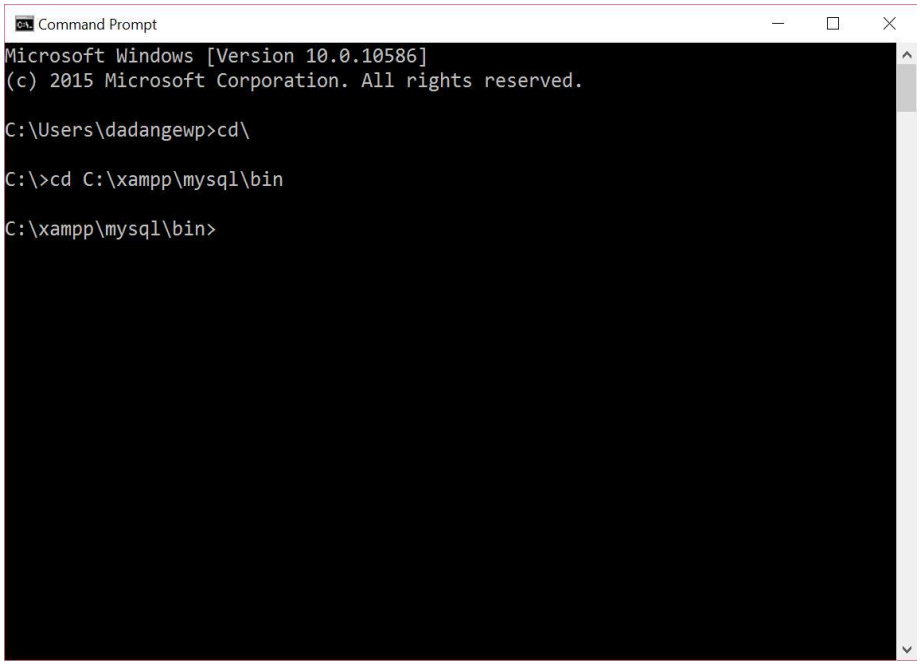
3. Jalankan Apache Server dan MySQL Server dengan menekan tombol "Start". Tunggu hingga muncul warna hijau pada nama Module.



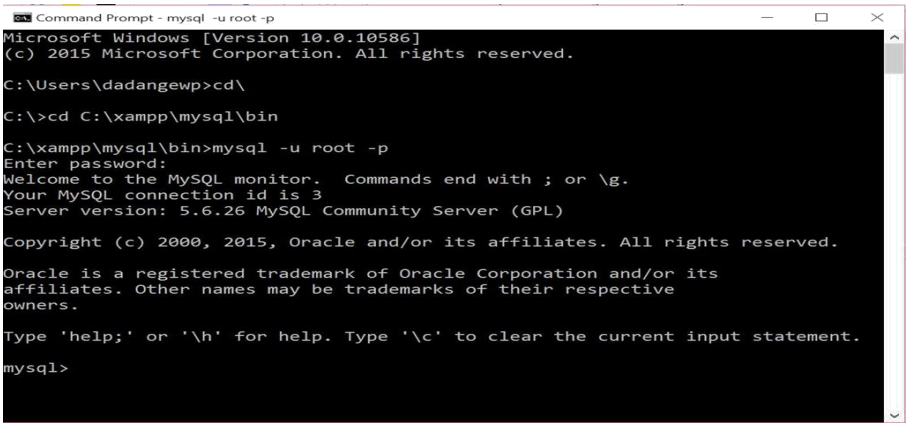
4. Pastikan bahwa server MySQL telah berjalan.
5. Buka command prompt dan ketik 'cd\' dan tekan "Enter". Sehingga anda akan berada di direktori (C:\)



6. Ketik C:\>xampp\mysql\bin, kemudian tekan tombol Enter.



7. Untuk dapat mengakses mysql, ketik : ‘mysql -u root -p’ (tanpa ‘ ’) kemudian tekan tombol Enter. Masukkan password (jika ada) kemudian klik tombol Enter lagi (secara default tidak ada password untuk root).



BAB 2

SQL (STRUCTURED QUERY LANGUAGE)

A. Sejarah SQL

Tahun 1970 E.F.Codd memperkenalkan database relational dalam sebuah artikel "*A Relational Model of Data For Large Shared Data Bank*". Tahun 1979 dikembangkan menjadi sebuah database relational. Bahasa dari database Relational tersebut adalah SQL.

B. Perintah pada SQL

Menurut penggunaannya, perintah-perintah SQL dapat dikelompokkan menjadi 2 bagian, yaitu :

1. Secara Interpretasi (Interactive SQL), yaitu dengan cara memasukkan perintah-perintah SQL melalui console atau mikrokomputer dan secara langsung diproses sehingga dapat langsung dilihat.
2. Secara Sisip (Embedded SQL), yaitu dengan cara menyisipkan perintah-perintah SQL ke dalam bahasa pemrogram tertentu sehingga untuk melihatnya dibutuhkan media khusus yang dirancang oleh seorang programmer.

C. Statement pada SQL

Statement SQl terbagi menjadi 3 bagian, yaitu:

1. DDL (Data Definition Language), yaitu sebuah perintah SQL yang berorientasi pada pembentukan atau penghapusan database, tabel dan index.
2. DML (Data Manipulation Language), yaitu perintah-perintah SQL yang berhubungan dengan data atau record, di antaranya menampilkan data, menghapus data, dan meng-update data.
3. DCL (Data Control Language), merupakan kumpulan perintah SQL yang berfungsi untuk melakukan pendefinisian pemakai yang boleh atau tidak mengakses database dan apa saja privilegenya.

Perintah-perintah yang ada pada DDL yaitu:

- CREATE DATABASE
- DROP DATABASE
- CREATE TABLE
- DROP TABLE
- ALTER TABLE
- CREATE INDEX
- DROP INDEX
- CREATE VIEW

Perintah-perintah yang ada pada DML yaitu:

- INSERTS
- SELECT
- UPDATED
- DELETE

Perintah-perintah yang ada pada DCL yaitu:

- COMMIT
- ROLLBACK
- GRANT
- REVOKE

D. Tipe Data pada MySql

Tipe data pada Mysql terdiri dari :

- Tipe data angka (numerik)
- Tipe data teks (string)
- Tipe data date
- Tipe data blob

Tipe data angka (numerik)

No	Nama	Fungsi	Jangkauan	Ukuran
1	TINYINT	Menyimpan data bilangan bulat positif dan negatif.	-128 s/d 127	1 byte (8 bit).
2	SMALLINT	Menyimpan data bilangan bulat positif dan negatif.	-32.768 s/d 32.767	2 byte (16 bit).
3	MEDIUMINT	Menyimpan data bilangan bulat positif dan negatif.	-8.388.608 s/d 8.388.607	Ukuran : 3 byte (24 bit).
4	INT	Menyimpan data bilangan bulat positif dan negatif.	-2.147.483.648 s/d 2.147.483.647	4 byte (32 bit).
5	BIGINT	Menyimpan data bilangan bulat positif dan negatif.	± 9,22 x 1018	8 byte (64 bit).
6	FLOAT	menyimpan data bilangan pecahan positif dan negatif presisi tunggal	-3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38.	4 byte (32 bit)
7	DOUBLE	menyimpan data bilangan pecahan positif dan negatif presisi ganda.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	5 byte (64 bit).
8	REAL	menyimpan data bilangan pecahan positif dan negatif presisi ganda.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	6 byte (64 bit).
9	DECIMAL	menyimpan data bilangan pecahan positif dan negatif.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	7 byte (64 bit).
10	NUMERIC	menyimpan data bilangan pecahan positif dan negatif.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	8 byte (64 bit).

Tipe data teks (String)

No	Nama	Fungsi	Jangkauan
1	CHAR	menyimpan data string ukuran tetap.	0 s/d 255 karakter
2	VARCHAR	menyimpan data string ukuran dinamis.	0 s/d 255 karakter (versi 4.1), 0 s/d 65.535
3	TINYTEXT	menyimpan data text.	1 s/d 255 karakter (versi 4.1), 0 s/d 65.535
4	TEXT	menyimpan data text.	0 s/d 65.535
5	MEDIUMTEXT	menyimpan data text.	0 s/d 224 - 1 karakter
6	LONGTEXT	menyimpan data text.	1 s/d 224 - 1 karakter

Tipe data date

No	Nama	Fungsi	Jangkauan	Ukuran
1	DATE	menyimpan data tanggal	1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)	3 byte.
2	TIME	menyimpan data waktu	-838:59:59 s/d +838:59:59 (HH:MM:SS)	3 byte.
3	DATETIME	menyimpan data tanggal dan waktu.	1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'	8 byte
4	YEAR	menyimpan data tahun dari tanggal	1900 s/d 2155	1 byte

Tipe data blob

No	Nama	Fungsi	Jangkauan
1	BIT	Menyimpan data biner.	64 digit biner
2	TINYBLOB	menyimpan data biner/ Gambar ukuran kecil	255 byte
3	BLOB	Menyimpan data biner/ Gambar	4
4	MEDIUMBLOB	Menyimpan data biner/ Gambar kuran sedang	224-1 byte
5	LONGBLOB	Menyimpan data biner/ Gambar ukuran besar	232- 1 byte

BAB 3
DDL (DATA DEFINITION LANGUAGE)

A. Pengertian DDL

DDL (Data Definition Language), DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, tabel, atribut (kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar tabel. Yang termasuk dalam kelompok DDL ini adalah **CREATE**, **ALTER**, dan **DROP**.

B. Perintah pada DDL

1. CREATE

   Syntaks membuat database :

```
CREATE DATABASE namadatabase;
```

   Namadatabase tidak boleh mengandung spasi dan tidak boleh memiliki nama yang sama antar database.

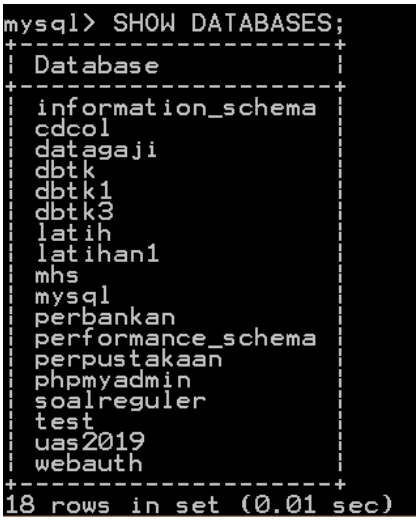
   Contoh:

   Membuat database perpustakaan:

```
CREATE DATABASE Perpustakaan;
```

```
mysql> CREATE DATABASE Perpustakaan;
Query OK, 1 row affected (0.02 sec)
```

   untuk menampilkan daftar nama database yang ada pada mysql menggunakan perintah : SHOW DATABASES;



2. Memilih database/menggunakan database yang telah dibuat

   Sebelum membuat suatu tabel, terlebih dahulu harus memilih salah satu database sebagai database aktif yang akan digunakan untuk menyimpan tabel-tabel yang akan dibuat. Berikut ini perintah untuk menggunakan database dengan nama Perpustakaan: `USE Perpustakaan;`

3. Menghapus database: DROP DATABASE namadatabase

   Database yang akan dihapus harus sesuai dengan namadatabase. Berikut ini perintah untuk menghapus database dengan nama Perpustakaan : `DROP DATABASE Perpustakaan;`

4. Membuat Tabel : CREATE TABLE namatabel (Field1 TipeData1, Field2 TipeData2);

Nama tabel tidak boleh mengandung spasi (space). Field1 dan TipeData1 merupakan nama kolom pertama dan tipe data untuk kolom pertama. Jika ingin membuat tabel dengan kolom lebih dari satu, maka setelah pendefinisian tipe data sebelumnya diberikan tanda koma (,).

Berikut ini perintah untuk membuat tabel dengan nama Anggota :

```
mysql> CREATE TABLE ANGGOTA (Id_Anggota Varchar(5),
    -> Nama_Anggota Varchar(25),
    -> Jns_Kel char(1),
    -> Pekerjaan varchar(20),
    -> Alamat text,
    -> Telepon varchar(12));
Query OK, 0 rows affected (0.38 sec)
```

5. Menampilkan tabel

   Untuk menampilkan daftar nama tabel yang ada pada database yang sedang aktif/digunakan (dalam hal ini database Perpustakaan) : `SHOW TABLES;`

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_perpustakaan |
+------------------------+
| anggota                |
+------------------------+
1 row in set (0.00 sec)
```

6. Menampilkan struktur tabel/field pada tabel

   Untuk menampilkan struktur tabel (dalam hal ini Anggota) syntaxnya adalah : `DESC Anggota;`

```
mysql> DESC Anggota;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Id_Anggota   | varchar(5)  | YES  |     | NULL    |       |
| Nama_Anggota | varchar(25) | YES  |     | NULL    |       |
| Jns_Kel      | char(1)     | YES  |     | NULL    |       |
| Pekerjaan    | varchar(20) | YES  |     | NULL    |       |
| Alamat       | text        | YES  |     | NULL    |       |
| Telepon      | varchar(12) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.16 sec)
```

7. Menghapus Tabel : DROP TABLE namatabel;

   Tabel yang akan dihapus sesuai dengan namatabel, berikut ini perintah untuk menghapus tabel dengan nama jenisfilm : DROP TABLE Anggota;

```
mysql> DROP TABLE Anggota;
Query OK, 0 rows affected (0.09 sec)
```

8. Mendefinisikan Null/Not Null : CREATE TABLE namatabel (Field1 TipeData1 NOT NULL, Field2 TipeData2);

```
mysql> CREATE TABLE ANGGOTA (Id_Anggota varchar(6) NOT NULL,
    -> Nama_Anggota varchar(25),
    -> Jns_Kel char(1),
    -> Pekerjaan varchar(20),
    -> Alamat text,
    -> Telepon varchar(12));
Query OK, 0 rows affected (0.19 sec)
```

```
mysql> DESC ANGGOTA;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Id_Anggota   | varchar(6)  | NO   |     | NULL    |       |
| Nama_Anggota | varchar(25) | YES  |     | NULL    |       |
| Jns_Kel      | char(1)     | YES  |     | NULL    |       |
| Pekerjaan    | varchar(20) | YES  |     | NULL    |       |
| Alamat       | text        | YES  |     | NULL    |       |
| Telepon      | varchar(12) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.02 sec)
```

9. Mendefinisikan Primary Key Pada Tabel

   Terdapat tiga cara untuk mendefinisikan primary key. Berikut ini adalah syntax mendefinisikan primary key untuk Field1:

```
CREATE TABLE namatabel (Field1 TipeData1 NOT NULL
PRIMARY KEY, Field2 TipeData2);
```

```
mysql> CREATE TABLE ANGGOTA (Id_Anggota varchar(6) NOT NULL PRIMARY KEY,
    -> Nama_Anggota varchar(25),
    -> Jns_Kel char(1),
    -> Pekerjaan varchar(20),
    -> Alamat text,
    -> Telepon varchar(12));
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> DESC ANGGOTA;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Id_Anggota   | varchar(6)  | NO   | PRI | NULL    |       |
| Nama_Anggota | varchar(25) | YES  |     | NULL    |       |
| Jns_Kel      | char(1)     | YES  |     | NULL    |       |
| Pekerjaan    | varchar(20) | YES  |     | NULL    |       |
| Alamat       | text        | YES  |     | NULL    |       |
| Telepon      | varchar(12) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.05 sec)
```

atau

```
CREATE TABLE namatabel (Field1 TipeData1, Field2
TipeData2, PRIMARY KEY (Field1));
```

atau

```
ALTER TABLE namatabel ADD CONSTRAINT
namaconstraint PRIMARY KEY (namakolom);
```

10. Menghapus Primary Key pada tabel

    Cara 1: Jika primary key dibuat dengan menggunakan alter table :

    ```
    ALTER TABLE namatabel DROP CONSTRAINT
    namaconstraint;
    ```

    Cara 2: Jika primary key dibuat melalui create table :

    ```
    ALTER TABLE namatabel DROP PRIMARY KEY;
    ```

```
mysql> ALTER TABLE ANGGOTA DROP PRIMARY KEY;
Query OK, 0 rows affected (0.34 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

11. Membuat INDEX

    Index berfungsi mempercepat proses pencarian data dalam suatu tabel. Adanya index pada suatu field tabel, menyebabkan proses pencarian otomatis akan dilakukan terlebih dahulu ke dalam index, apabila ditemukan baru akan diambilkan data yang sesungguhnya dari tabel. Apabila tidak ditemukan dalam index, sudah dapat dipastikan bahwa data tersebut memang tidak ada dalam tabel.

Index juga dapat dibuat untuk setiap kolom yang akan dijadikan kriteria tertentu untuk pencarian data, sehingga proses pencariannya akan lebih cepat. Pada index terdapat perintah pembuatan dan penghapusan index, namun tidak terdapat perintah perubahan nama index. Syntax penulisan :

```
CREATE INDEX namaindex ON nama namatable(field);
```

atau

```
ALTER TABLE namatable ADD INDEX namaindex(field);
```

```
mysql> CREATE INDEX mhsx ON Mahasiswa(NPM);
Query OK, 0 rows affected (5.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

12. Menghapus INDEX

    Penghapusan nama index tidak akan menghapus field table atau tabel, namun hanya memperlambat proses pencarian saja. Syntax penulisan:

```
Drop index nama_index on nama_table;
    Atau
Alter table nama_table drop index nama_index;
```

```
mysql> DROP INDEX MHSX ON Mahasiswa;
Query OK, 0 rows affected (0.50 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

13. Menambah Kolom Baru Pada Tabel : ALTER TABLE namatabel ADD fieldbaru tipe;

    Contoh:

```
mysql> ALTER TABLE ANGGOTA ADD TELEPON varchar(12);
Query OK, 3 rows affected (1.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Untuk meletakkan field diawal, tambahkan sintaks first : ALTER TABLE namatabel ADD COLUMN namafield FIRST;

Untuk menyisipkan field setelah field tertentu, tambahkan sintaks after : ALTER TABLE namatabel ADD COLUMN Fieldsisip CHAR(5) AFTER field

14. Mengubah Tipe Data atau Lebar Kolom Pada Tabel : `ALTER TABLE Namatabel MODIFY COLUMN FIELD Tipedata;`
    Namatabel adalah nama tabel yang akan diubah tipe data atau lebar kolomnya. Field adalah kolom yang akan diubah tipe data atau lebarnya. Tipe adalah tipe data baru atau tipe data lama dengan lebar kolom yang berbeda. Berikut ini contoh perintah untuk mengubah tipe data untuk kolom nama dengan char(20) :

```
mysql> ALTER TABLE MAHASISWA MODIFY COLUMN Nama varchar(20);
Query OK, 0 rows affected (0.39 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

15. Mengubah Nama Field/Kolom : `ALTER TABLE namatabel CHANGE COLUMN namalamakolom namabarukolom tipedatabaru;`

```
mysql> ALTER TABLE ANGGOTA CHANGE COLUMN Nama NamaAnggota varchar(20);
Query OK, 3 rows affected (0.43 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

16. Menghapus field/kolom pada tabel : `ALTER TABLE namatabel DROP COLUMN namakolom;`

```
mysql> ALTER TABLE ANGGOTA DROP COLUMN Telepon;
Query OK, 3 rows affected (0.53 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Latihan

1. Buatlah database dengan nama Latihan1!
2. Buatlah tabel Mahasiswa dengan field NPM, Nama, Alamat, Jen_Kel, TglLahir, dan Prodi!
3. Rubahlah field Jen_Kel menjadi JnsKel pada Tabel Mahasiswa!
4. Hapuslah field Prodi pada tabel mahasiswa!
5. Rubahlan size kolom nama menjadi size 25 pada tabel Mahasiswa!
6. Tambahkan kolom kota setelah kolom alamat pada tabel Mahasiswa!
7. Tambahkan kolom Telepon pada tabel mahasiswa!
8. Jadikanh field NPM pada tabel mahasiswa sebagai primary key!
9. Rubahlah Tabel Mahasiswa menjadi tabel mhs!
10. Rubahlah database Latihan1 menjadi database Latihan!

BAB 4

DML (DATA MANIPULATION LANGUAGE) Bag.1

A. Definisi DML

DML (Data Manipulation Language) DML adalah kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data, misalnya untuk pengambilan, penyisipan, pengubahan dan penghapusan data.

B. Perintah Pada DML

Perintah yang termasuk katagori DML sebagai berikut :

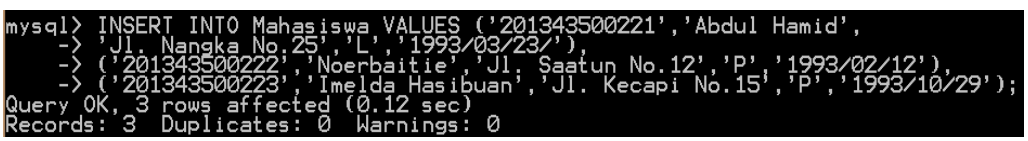
PERINTAH	KETERANGAN
SELECT	menampilkan data dari tabel
INSERT	menyisipkan baris pada tabel
DELETE	menghapus baris pada tabel
UPDATE	mengubah isi kolom pada tabel
COMMIT	menuliskan perubahan pada disk
ROOLLBACK	membatalkan perubahan dari perintah COMMIT

1. INSERT

   Perintah INSERT digunakan untuk menyisipkan baris pada tabel/menambahkan baris pada suatu tabel. Terdapat dua cara untuk menambah baris, yaitu:

   - Menambah baris dengan mengisi data pada setiap kolom

     ```
     INSERT INTO namatabel VALUES
     (nilai1,nilai2,nilai-n);
     ```



   - Menambah baris dengan hanya mengisi data pada kolom

tertentu :

```
INSERT INTO namatabel (kolom1,kolom2,kolom-n)
VALUES (nilai1,nilai2,nilai-n);
```

```
mysql> INSERT INTO Mahasiswa (NPM,Nama,Alamat,Jen_Kel,TglLahir)
    -> VALUES ('201343500224','Rizki Fauzi','Jl. Merdeka No.55','L','1992/10/21');
Query OK, 1 row affected (0.09 sec)
```

Ket : Jika data bertipe string, date atau time maka pemberian nilainya diapit dengan tanda petik tunggal ('XXX') atau petik ganda ("XXX"). Jika data bertipe numerik (90, 2700) maka pemberian nilainya tidak diapit tanda petik tunggal maupun ganda.

2. DELETE

Perintah DELETE digunakan untuk menghapus satu baris dengan kondisi tertentu atau seluruh baris tanpa kondisi

- Menghapus satu baris dengan kondisi:

```
DELETE FROM namatabel WHERE kondisi;
```

```
mysql> DELETE FROM Mahasiswa where alamat = 'Jl. Merdeka No.55';
Query OK, 1 row affected (0.13 sec)
```

- Menghapus seluruh baris tanpa kondisi;

```
DELETE FROM namatabel;
```

```
mysql> DELETE FROM Mahasiswa;
Query OK, 3 rows affected (0.10 sec)
```

3. UPDATE

Perintah UPDATE digunakan untuk mengubah isi data pada satu baris dengan kondisi atau beberapa kolom tanpa kondisi pada suatu tabel.

- Mengubah isi data pada satu baris dengan kondisi:
  `UPDATE namatabel SET kolom1 = nilai1, kolom2 = nilai2 WHERE kondisi;`

```
mysql> UPDATE Anggota SET Alamat = 'Jl. Wilis No.22'
    -> WHERE NamaAnggota = 'Siti Sarah';
Query OK, 1 row affected (0.36 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- Mengubah isis data pada beberapa kolom dengan kondisi
  `UPDATE namatabel SET kolom1 = nilai1;`

```
mysql> UPDATE Anggota SET Pekerjaan = 'Mahasiswa';
Query OK, 2 rows affected (0.08 sec)
Rows matched: 3  Changed: 2  Warnings: 0
```

4. SELECT

   Perintah SELECT digunakan untuk menampilkan isi dari suatu tabel yang dapat dihubungkan dengan tabel yang lainnya.

   a. Menampilkan data untuk semua field/kolom menggunakan * (asterik) : `SELECT * From namatabel;`

```
mysql> SELECT * FROM Anggota;
+------------+------------------+----------------------+-----------+
| ID_ANGGOTA | NamaAnggota      | ALAMAT               | PEKERJAAN |
+------------+------------------+----------------------+-----------+
| 012015001  | Siti Sarah       | Jl. Wilis No.22      | Karyawan  |
| 032015002  | Khairunnisa Salma| Jl. Sulirang No.21 D | Mahasiswa |
| 022015003  | Rianto Budianto  | Jl. Cagar Alam No.56 | Pelajar   |
| 032015004  | Lilis Khairunisa | Jl. Bandeng No.60    | Mahasiswa |
| 032013005  | Candra Kurniawan | Jl. Limau No.34      | Mahasiswa |
+------------+------------------+----------------------+-----------+
5 rows in set (0.00 sec)
```

   b. Menampilkan data untuk field/kolom tertentu :

```
mysql> SELECT ID_ANGGOTA, NamaAnggota FROM Anggota;
+------------+------------------+
| ID_ANGGOTA | NamaAnggota      |
+------------+------------------+
| 012015001  | Siti Sarah       |
| 032015002  | Khairunnisa Salma|
| 022015003  | Rianto Budianto  |
| 032015004  | Lilis Khairunisa |
| 032013005  | Candra Kurniawan |
+------------+------------------+
5 rows in set (0.31 sec)
```

c. Menampilkan seluruh data untuk kondisi data tertentu: `SELECT * FROM namatabel WHERE kondisi;`

```
mysql> SELECT * FROM Anggota WHERE Pekerjaan = 'Mahasiswa';
+------------+------------------+----------------------+-----------+
| ID_ANGGOTA | NamaAnggota      | ALAMAT               | PEKERJAAN |
+------------+------------------+----------------------+-----------+
| 032015002  | Khairunnisa Salma | Jl. Sulirang No.21 D | Mahasiswa |
| 032015004  | Lilis Khairunisa  | Jl. Bandeng No.60    | Mahasiswa |
| 032013005  | Candra Kurniawan  | Jl. Limau No.34      | Mahasiswa |
+------------+------------------+----------------------+-----------+
3 rows in set (0.00 sec)
```

d. Menampilkan beberapa field dengan kondisi : `SELECT field1, field2 FROM namatabel WHERE kondisi;`

```
mysql> SELECT NamaAnggota, Alamat FROM Anggota WHERE Pekerjaan = 'Mahasiswa';
+------------------+----------------------+
| NamaAnggota      | Alamat               |
+------------------+----------------------+
| Khairunnisa Salma | Jl. Sulirang No.21 D |
| Lilis Khairunisa  | Jl. Bandeng No.60    |
| Candra Kurniawan  | Jl. Limau No.34      |
+------------------+----------------------+
3 rows in set (0.02 sec)
```

e. Ekspresi dan operator pada SQL
   - + (positif), - (negatif), ~ (bitwise NOT)
   - * (perkalian), / (pembagian), % (modulus)
   - + (penjumlahan), + (penggabungan), - (pengurangan)
   - +, >, <, >=, <=, <>, !=, !>, !<
   - ^ (bitwise exclusive OR), & (bitwise AND), | (bitwise OR)
   - NOT
   - AND
   - ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
   - = (penugasan)

Latihan

1. Buatlah database dengan nama Anda!
2. Buatlah tabel dengan nama Buku pada database nama Anda sesuai dengan struktur tabel dibawah ini!

```
mysql> desc buku;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| KdBuku     | varchar(4)  | YES  |     | NULL    |       |
| NamaBuku   | varchar(30) | YES  |     | NULL    |       |
| Pengarang  | varchar(20) | YES  |     | NULL    |       |
| JmlBuku    | int(3)      | YES  |     | NULL    |       |
| KdPenerbit | varchar(6)  | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.13 sec)
```

3. Isilah data pada tabel buku!

```
mysql> Select * from Buku;
+--------+------------------+----------+---------+------------+
| KdBuku | NamaBuku         | Pengarang | JmlBuku | KdPenerbit |
+--------+------------------+----------+---------+------------+
| B001   | Bawang Merah     | Mira     |      25 | PIM201     |
| B002   | Anak Asuh        | Mira     |      14 | PIM201     |
| B003   | Buah Hati        | Nita     |      12 | PBK101     |
| B004   | Pendekar Sakti   | Anwar    |      21 | PBK101     |
| B005   | Si Penolong Cilik | Selly    |      18 | PIM201     |
+--------+------------------+----------+---------+------------+
5 rows in set (0.00 sec)
```

4. Jadikan field KbBuku pada tabel buku sebagai primary key!
5. Rubahlah field NamaBuku menjadi field Judul pada tabel Buku!
6. Rubahlah nama pengarang untuk buku "Pendekar Sakti" menjadi "Anwar Sanusi"!
7. Tambahkah data pada tabel buku dengan data: Kode buku B006, Judul: Si kancil yang cerdik, pengarang: Iriawati, jumlah=28, kode penerbit : PBK101!
8. Tampilkan data buku dengan kode penerbit 'PBK101'!
9. Tampilkan Judul dan JmlBuku untuk buku yang pengarangnya bernama 'Mira'!
10. Hapuslah data untuk buku yang berjudul 'Anak Asuh'!

BAB 5

DML (DATA MANIPULATION LANGUAGE) Bag.2

A. Ekspresi dan operator pada SQL

- + (positif), - (negatif), ~ (bitwise NOT)
- * (perkalian), / (pembagian), % (modulus)
- + (penjumlahan), + (penggabungan), - (pengurangan)
- +, >, <, >=, <=, <>, !=, !>, !<
- ^ (bitwise exclusive OR), & (bitwise AND), | (bitwise OR)
- NOT
- AND
- ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
- = (penugasan)

B. Perintah SELECT dengan Operator dan Klausa

1. Operator BETWEEN ... AND ...

   Digunakan untuk menampilkan data antara dua nilai. Syntax: `SELECT namafield FROM namatabel WHERE namafield BETWEEN nilai1 AND nilai2;`

   Contoh: menampilkan nilai UTS antara 60 dan 75 pada tabel Nilai

```
mysql> SELECT UTS FROM Nilai WHERE UTS BETWEEN 60 AND 75;
+------+
| UTS  |
+------+
|   60 |
|   70 |
|   75 |
+------+
3 rows in set (0.00 sec)
```

2. Operator ANY

   Operator ANY digunakan berkaitan dengan subquery. Operator ini menghasilkan TRUE (benar) jika paling tidak salah satu perbandingan dengan hasil subquery menghasilkan nilai TRUE. Ilustrasinya jika:

   Gaji > ANY (S)

Jika subquery S menghasilkan G1, G2, ..., Gn, maka kondisi di atas identik dengan:

(gaji > G1) OR (gaji > G2) OR ... OR (gaji > Gn)

Contoh: perintah untuk menampilkan semua data Nilai yang Nilai UTSnya bukan yang terkecil:

```
mysql> SELECT * FROM Nilai WHERE UTS > ANY (SELECT UTS FROM Nilai);
+--------------+-------+------+------+
| NPM          | KD_MK | UTS  | UAS  |
+--------------+-------+------+------+
| 201143500439 | KK021 |   60 |   75 |
| 201143500121 | KD123 |   70 |   90 |
| 201143500234 | KK021 |   50 |   40 |
| 201143500165 | KU122 |   90 |   80 |
| 201143500228 | KU122 |   75 |   75 |
| 201143500326 | KD123 |   80 |    0 |
+--------------+-------+------+------+
6 rows in set (0.00 sec)
```

3. Operator ALL

   Operator ALL digunakan untuk melakukan perbandingan dengan subquery. Kondisi dengan ALL menghasilkan nilai TRUE (benar) jika subquery tidak menghasilkan apapun atau jika perbandingan menghasilkan TRUE untuk setiap nilai query terhadap hasil subquery.

   Contoh : perintah untuk menampilkan data jenisfilm yang harganya paling tinggi:

```
mysql> SELECT * FROM Nilai WHERE UAS >= ALL (SELECT UAS From Nilai);
+--------------+-------+------+------+
| NPM          | KD_MK | UTS  | UAS  |
+--------------+-------+------+------+
| 201143500121 | KD123 |   70 |   90 |
+--------------+-------+------+------+
1 row in set (0.00 sec)
```

4. Klausa LIKE

   Klausa LIKE digunakan untuk melakukan pencarian berdasarkan pola tertentu pada suatu kolom. Syntax: SELECT `namafield` FROM `namatabel` WHERE `namafield` LIKE `pola;`

   Pada pola dapat menggunakan tanda %, baik sebelum ataupun sesudah pola.

   Contoh1 : Menampilkan Nama Mahasiswa yang diawali huruf "D"

```
mysql> Select Nama From Mahasiswa Where Nama LIKE "D%";
+--------------+
| Nama         |
+--------------+
| Desi Maryani |
+--------------+
1 row in set (0.00 sec)
```

Contoh2 : Menampilkan Nama, alamat dan jenis kelamin Mahasiswa yang namanya mengandung huruf 'R'!

```
mysql> SELECT Nama, Alamat, JnsKel FROM Mahasiswa
    -> ORDER BY Nama LIKE '%R%';
+----------------+---------+--------+
| Nama           | Alamat  | JnsKel |
+----------------+---------+--------+
| Muhammad Gandi | Depok   | L      |
| Andi           | Jakarta | L      |
| Desi Maryani   | Bekasi  | P      |
| Firdaus        | Jakarta | L      |
| Endah Haryati  | Depok   | P      |
| Riswandi       | Bekasi  | L      |
| Indar Hari     | Bogor   | L      |
+----------------+---------+--------+
7 rows in set (0.05 sec)
```

5. Klausa ORDER BY

   Klausa ORDER BY digunakan untuk mengurutkan data berdasarkan kolom tertentu sesuai dengan tipe data yang dimiliki. Syntax : `SELECT namafield FROM namatabel ORDER By namafield yang akan diurutkan;`

   Contoh1 : perintah untuk mengurutkan data mahasiswa berdasarkan kolom nama yang diurutkan secara ascending (menaik):

```
mysql> SELECT * From Mahasiswa ORDER BY Nama ASC;
+-------------+---------------+---------+--------+------------+
| NPM         | Nama          | Alamat  | JnsKel | TglLahir   |
+-------------+---------------+---------+--------+------------+
| 201143500439 | Andi          | Jakarta | L      | 1991-06-12 |
| 201143500121 | Desi Maryani  | Bekasi  | P      | 1991-08-24 |
| 201143500234 | Endah Haryati | Depok   | P      | 1991-05-15 |
| 201143500165 | Firdaus       | Jakarta | L      | 1991-05-18 |
| 201143500228 | Gandi         | Depok   | L      | 1991-07-10 |
| 201143500326 | Hilda         | Bogor   | P      | 1991-02-08 |
| 201443500694 | Indar Hari    | Bogor   | L      | 1990-12-30 |
| 201443500693 | Riswandi      | Bekasi  | L      | 1990-10-08 |
+-------------+---------------+---------+--------+------------+
8 rows in set (0.03 sec)
```

Contoh2 : perintah untuk mengurutkan data mahasiswa berdasarkan kolom nama yang diurutkan secara descending (menurun):

```
mysql> SELECT * From Mahasiswa ORDER BY Nama Desc;
+--------------+---------------+---------+--------+------------+
| NPM          | Nama          | Alamat  | JnsKel | TglLahir   |
+--------------+---------------+---------+--------+------------+
| 201443500693 | Riswandi      | Bekasi  | L      | 1990-10-08 |
| 201443500694 | Indar Hari    | Bogor   | L      | 1990-12-30 |
| 201143500326 | Hilda         | Bogor   | P      | 1991-02-08 |
| 201143500228 | Gandi         | Depok   | L      | 1991-07-10 |
| 201143500165 | Firdaus       | Jakarta | L      | 1991-05-18 |
| 201143500234 | Endah Haryati | Depok   | P      | 1991-05-15 |
| 201143500121 | Desi Maryani  | Bekasi  | P      | 1991-08-24 |
| 201143500439 | Andi          | Jakarta | L      | 1991-06-12 |
+--------------+---------------+---------+--------+------------+
8 rows in set (0.00 sec)
```

6. Klausa Subquery IN, NOT IN, EXISTS, NOT EXISTS

   Subquery berarti query di dalam query. Dengan menggunakan subquery, hasil dariquery akan menjadi bagian dari query di atasnya. Subquery terletak di dalam klausa WHERE atau HAVING. Pada klausa WHERE, subquery digunakan untuk memilih baris-baris tertentu yang kemudian digunakan oleh query. Sedangkan pada klausa HAVING, subquery digunakan untuk memilih kelompok baris yang kemudian digunakan oleh query.

   Contoh IN/EXISTS:

   Perintah untuk menampilkan data pada tabel Matakuliah yang mana data pada kolomkodematakuliah-nya tercantum pada tabel Nilai menggunakan IN :

```
mysql>
mysql> SELECT * FROM Matakuliah WHERE KD_MK IN (SELECT KD_MK FROM Nilai);
+-------+------------------+------+
| KD_MK | Nama_MataKuliah  | SKS  |
+-------+------------------+------+
| KK021 | Sistem Basis Data |    2 |
| KU122 | Ilmu Budaya Dasar |    2 |
+-------+------------------+------+
2 rows in set (0.05 sec)
```

atau menggunakan EXISTS:

```
mysql> SELECT * FROM Matakuliah WHERE EXISTS (SELECT * FROM Nilai WHERE UTS > 65
);
+-------+-------------------------------+------+
| KD_MK | Nama_MataKuliah               | SKS  |
+-------+-------------------------------+------+
| KD132 | Interaksi Manusia & Komputer |    3 |
| KK021 | Sistem Basis Data             |    2 |
| KU122 | Ilmu Budaya Dasar             |    2 |
+-------+-------------------------------+------+
3 rows in set (0.00 sec)
```

Contoh NOT IN/NOT EXISTS:

Perintah untuk menampilkan data pada tabel Matakuliah yang mana data pada kolom KD_MK-nya tidak tercantum pada tabel Nilai menggunakan NOT IN:

```
mysql> SELECT * FROM Matakuliah WHERE KD_MK NOT IN (SELECT KD_MK FROM Nilai);
+-------+-------------------------------+------+
| KD_MK | Nama_MataKuliah               | SKS  |
+-------+-------------------------------+------+
| KD132 | Interaksi Manusia & Komputer |    3 |
+-------+-------------------------------+------+
1 row in set (0.00 sec)
```

Atau menggunakan NOT EXISTS

```
mysql> SELECT * FROM Matakuliah WHERE NOT EXISTS
    -> (SELECT * FROM Nilai WHERE KD_MK > 65);
+-------+-------------------------------+------+
| KD_MK | Nama_MataKuliah               | SKS  |
+-------+-------------------------------+------+
| KD132 | Interaksi Manusia & Komputer |    3 |
| KK021 | Sistem Basis Data             |    2 |
| KU122 | Ilmu Budaya Dasar             |    2 |
+-------+-------------------------------+------+
3 rows in set, 7 warnings (0.00 sec)
```

7. Perintah DISTINCT

   Distinct adalah kata kunci ini untuk menghilangan duplikasi.

   Contoh: Tampilkan kolom stok pada tabel barang:

```
mysql> SELECT DISTINCT Stok From Barang;
+------+
| Stok |
+------+
|   50 |
|  120 |
|   60 |
|  100 |
|   80 |
+------+
5 rows in set (0.00 sec)
```

Latihan:

1. Buatlah Tabel di bawah ini:

   Mahasiswa

```
+-------------+----------------+---------+--------+------------+
| NPM         | Nama           | Alamat  | JnsKel | TglLahir   |
+-------------+----------------+---------+--------+------------+
| 201143500121 | Desi Maryani   | Bekasi  | P      | 1991-08-24 |
| 201143500165 | Firdaus        | Jakarta | L      | 1991-05-18 |
| 201143500228 | Muhammad Gandi | Depok   | L      | 1991-07-10 |
| 201143500234 | Endah Haryati  | Depok   | P      | 1991-05-15 |
| 201143500439 | Andi Hermawan  | Jakarta | L      | 1991-06-12 |
| 201443500693 | Arafindy       | Bekasi  | L      | 1990-10-08 |
| 201443500694 | Indar Triaswati | Bogor  | L      | 1990-12-30 |
+-------------+----------------+---------+--------+------------+
7 rows in set (0.00 sec)
```

   Matakuliah

```
+-------+-------------------------------+------+
| KD_MK | Nama_MataKuliah               | SKS  |
+-------+-------------------------------+------+
| KD132 | Interaksi Manusia & Komputer  |    3 |
| KK020 | Praktikum Sistem Basis Data   |    1 |
| KK021 | Sistem Basis Data             |    2 |
| KK034 | Manajemen Proyek              |    3 |
| KK077 | Metode Numerik                |    3 |
| KU122 | Ilmu Budaya Dasar             |    2 |
+-------+-------------------------------+------+
6 rows in set (0.05 sec)
```

   Nilai

```
+--------------+-------+------+------+
| NPM          | KD_MK | UTS  | UAS  |
+--------------+-------+------+------+
| 201143500439 | KK021 |   60 |   75 |
| 201143500121 | KD123 |   70 |   90 |
| 201143500234 | KK021 |   50 |   40 |
| 201143500165 | KU122 |   90 |   80 |
| 201143500228 | KU122 |   75 |   75 |
| 201143500326 | KD123 |   80 |    0 |
| 201143500439 | KD123 |   40 |   30 |
+--------------+-------+------+------+
7 rows in set (0.03 sec)
```

2. Tampilkan NPM dan Nama untuk mahasiswa berjenis kelamin ‘Perempuan’!
3. Tampilkan data mahasiswa dengan namamahasiswa diurutkan secara Descending!
4. Tampilkan Nama matakuliah dan SKS untuk nama matakuliah yang mengandung karakter ‘P’!
5. Tampilkan Nama, alamat dan jenis kelamin untuk mahasiswa yang namanya mengandung 8 karakter!
6. Tampilkan kolom SKS pada tabel matakuliah dengan tidak ada pengulangan data!
7. Tampilkan NPM dan UTS mahasiswa untuk nilai UTS diatas 60!
8. Tampilkan NPM dan UAS mahasiswa untuk nilai UAS antara 70 sampai 90!
9. Tampilkan data pada tabel Mahasiswa yang mana data pada kolom NPMnya tercantum pada tabel Nilai!
10. Tampilkan data pada tabel Mahasiswa yang mana data pada kolom NPMnya tidak tercantum pada tabel Nilai!

BAB 6
FUNGSI AGREGAT

Fungsi agregat (aggregate function) adalah fungsi yang menerima koleksi nilai dan mengembalikan nilai tunggal sebagai hasilnya. Seperti jumlah data, nilai minimun, nilai maksimun, dan nialai rata-rata.

Jenis-jenis Fungsi Agregat:

Fungsi	Deskripsi
COUNT	Mengembalikan jumlah (banyaknya atau kemunculannya) nilai suatu kolom
SUM	Mengembalikan jumlah (total atau sum) nilai di suatu kolom
AVG	Mengembalikan rata-rata (average) nilai di suatu kolom
MIN	Mengembalikan nilai terkecil (minimal) di suatu kolom
MAX	Mengembalikan nilai terbesar (maximal) di suatu kolom

1. COUNT

   Perintah yang digunakan untuk menghitung jumlah baris suatu kolom pada tabel. Sintaks: `SELECT COUNT(namafield) FROM nama_tabel;`

   Contoh :

Kd_Barang	Nama_Barang	Stok	Harga_Sat
A0001	Buku Gambar	50	25000
A0002	Buku Tulis	120	28000
B0005	Pulpen	120	36000
B0006	Pensil	60	54000
B0007	Spidol	60	15000
C0003	Penggaris	100	22000
C0004	Penghapus	80	28000

   Perintah untuk menghitung jumlah baris kolom nama barang pada tabel barang:

```
mysql> SELECT COUNT(nama_barang) FROM Barang;
+--------------------+
| COUNT(nama_barang) |
+--------------------+
|                  7 |
+--------------------+
1 row in set (0.00 sec)
```

2. SUM

Perintah yang digunakan untuk menghitung jumlah nilai suatu kolom pada tabel. Sintaks: `SELECT SUM(namafield) FROM nama_tabel;`

Contoh :

perintah untuk menghitung jumlah nilai kolom stok pada tabel barang :

```
mysql> SELECT SUM(Stok) TotalStok FROM Barang;
+------------+
| TotalStok  |
+------------+
|        590 |
+------------+
1 row in set (0.02 sec)
```

atau

```
mysql> SELECT SUM(Stok) FROM Barang;
+------------+
| SUM(Stok)  |
+------------+
|        590 |
+------------+
1 row in set (0.01 sec)
```

3. AVG

Perintah yang digunakan untuk menghitung rata- rata dari nilai suatu kolom pada tabel. Sintaks: `SELECT AVG(namafield) FROM nama_tabel;`

Contoh :

perintah untuk menghitung rata-rata dari kolom Harga_Sat pada tabel Barang:

```
mysql> SELECT AVG(Harga_Sat)Rata_Rata_HargaSatuan FROM Barang;
+-----------------------+
| Rata_Rata_HargaSatuan |
+-----------------------+
|    29714.285714285714 |
+-----------------------+
1 row in set (0.05 sec)
```

4. Min

Perintah yang digunakan untuk menampilkan nilai terkecil dari suatu kolom pada tabel. Sintaks: `SELECT MIN(namafield) FROM nama_tabel;`

Contoh :

perintah untuk menampilkan harga terendah dari kolom harga satuan pada tabel barang;

```
mysql> SELECT MIN(Harga_Sat)Harga_Terendah FROM Barang;
+-----------------+
| Harga_Terendah  |
+-----------------+
|           15000 |
+-----------------+
1 row in set (0.06 sec)
```

5. MAX

Perintah yang digunakan untuk menampilkan nilai terbesar dari suatu kolom pada tabel. Sintaks: `SELECT MAX(namafield) FROM nama_tabel;`

Contoh :

perintah untuk menampilkan harga terbesar dari kolom harga satuan pada tabel barang:

```
mysql> SELECT MAX(Harga_Sat)Harga_Tertinggi FROM Barang;
+-----------------+
| Harga_Tertinggi |
+-----------------+
|           54000 |
+-----------------+
1 row in set (0.00 sec)
```

6. Keyword DISTINCT

DISTINCT dapat dimanfaatkan untuk mengeliminasi kemunculan data yang sama. Sintaks: SELECT DISTINCT field1,field2,...,fieldn FROM namatabel

Contoh:

```
mysql> SELECT DISTINCT Stok From Barang;
+-------+
| Stok  |
+-------+
|    50 |
|   120 |
|    60 |
|   100 |
|    80 |
+-------+
5 rows in set (0.03 sec)
```

atau

```
mysql> SELECT SUM(DISTINCT Stok) TotalStok FROM Barang;
+------------+
| TotalStok  |
+------------+
|        410 |
+------------+
1 row in set (0.03 sec)
```

7. Klausa Group By

Digunakan untuk menampilkan atau memilih sekumpulan data berdasarkan kelompok data tertentu.

1) Pengelompokan nya biasa nya di sertai oleh Aggregat Fuction
2) Dalam Implementasi nya Aggregat Function harus di ikuti oleh Group by bila terdapat Field lain yang dijadikan kriteria pengelompokan

```
+-----------+------------+-----------+--------+----------+
| No_faktur | Tanggal    | Kd_Barang | Jumlah | Kd_Kasir |
+-----------+------------+-----------+--------+----------+
| N26111    | 2017-06-05 | A0001     |     10 | ST153    |
| N26111    | 2017-06-05 | A0006     |     10 | ST153    |
| N26111    | 2017-06-05 | C0004     |     25 | ST153    |
| N26112    | 2017-06-06 | A0002     |     12 | SV152    |
| N26112    | 2017-06-06 | B0007     |     12 | SV152    |
| N26113    | 2017-06-07 | A0002     |     20 | ST153    |
| N26114    | 2017-06-08 | A0002     |     15 | SV152    |
| N26114    | 2017-06-08 | A0001     |      6 | SV152    |
+-----------+------------+-----------+--------+----------+
```

Contoh:

Perintah untuk menampilkan jumlah terendah dari kode barang pada tabel jual

```
mysql> SELECT Kd_Barang, Min(Jumlah) AS Jumlah
    -> FROM Jual
    -> GROUP BY (Kd_Barang);
+-----------+--------+
| Kd_Barang | Jumlah |
+-----------+--------+
| A0001     |      6 |
| A0002     |     12 |
| A0006     |     10 |
| B0007     |     12 |
| C0004     |     25 |
+-----------+--------+
5 rows in set (0.06 sec)
```

Latihan:

Mahasiswa

```
+-------------+-----------------+---------+--------+------------+
| NPM         | Nama            | Alamat  | JnsKel | TglLahir   |
+-------------+-----------------+---------+--------+------------+
| 201143500121 | Desi Maryani    | Bekasi  | P      | 1991-08-24 |
| 201143500165 | Firdaus         | Jakarta | L      | 1991-05-18 |
| 201143500228 | Muhammad Gandi  | Depok   | L      | 1991-07-10 |
| 201143500234 | Endah Haryati   | Depok   | P      | 1991-05-15 |
| 201143500439 | Andi Hermawan   | Jakarta | L      | 1991-06-12 |
| 201443500693 | Arafindy        | Bekasi  | L      | 1990-10-08 |
| 201443500694 | Indar Triaswati | Bogor   | L      | 1990-12-30 |
+-------------+-----------------+---------+--------+------------+
7 rows in set (0.00 sec)
```

Matakuliah

```
+-------+-------------------------------+------+
| KD_MK | Nama_MataKuliah               | SKS  |
+-------+-------------------------------+------+
| KD123 | Interaksi Manusia & Komputer  |    3 |
| KK020 | Praktikum Sistem Basis Data   |    1 |
| KK021 | Sistem Basis Data             |    2 |
| KK034 | Manajemen Proyek              |    3 |
| KK077 | Metode Numerik                |    3 |
| KU122 | Ilmu Budaya Dasar             |    2 |
+-------+-------------------------------+------+
```

Nilai

```
+--------------+-------+------+------+
| NPM          | KD_MK | UTS  | UAS  |
+--------------+-------+------+------+
| 201143500439 | KK021 |   60 |   75 |
| 201143500121 | KD123 |   70 |   90 |
| 201143500234 | KK021 |   50 |   40 |
| 201143500165 | KU122 |   90 |   80 |
| 201143500228 | KU122 |   75 |   75 |
| 201143500326 | KD123 |   80 |    0 |
| 201143500439 | KD123 |   40 |   30 |
+--------------+-------+------+------+
7 rows in set (0.03 sec)
```

1. Tampilkan nilai Rata-rata UTS pada tabel Nilai!

2. Tampilkan jumlah total nilai UAS pada tabel Nilai!
3. Tampilkan nilai terendah dari nilai UAS pada tabel Nilai!
4. Tampilkan rata-rata dan jumlah nilai UAS untuk mata kuliah 'KD123'!
5. Tampilkan KD_MK dan jumlah KD_MK dari masing-masing KD_MK matakuliah!
6. Tampilkan KD_MK dan jumlah KD_MK dari masing-masing KD_MK yang mempunyai jumlah lebih dari 2!
7. Tampilkan KD_MK dan jumlah total nilai UTS dari masing-masing KD_MK!
8. Tampilkan NPM dan nilai UTS pada tabel nilai yang nilai UTS-nya 50 sampai dengan 70!
9. Tampilkan NPM dan nilai UAS pada tabel nilai yang nilai UAS-nya 60 sampai dengan 80!
10. Tampilkan NPM dan nilai UTS pada tabel nilai yang nilai UTS-nya bukan 50 sampai dengan 75!

BAB 7
FUNGSI TANGGAL

1. **ADDATE** (x, Interval nilai_interval)

   Berfungsi untuk mendapatkan tanggal baru karena proses dari penjumlahan nilai interval.

Tipe Nilai	Keterangan
Second	Satuan Detik
Minute	Satuan Menit
Hour	Satuan Jam
Day	Satuan Hari
Month	Satuan Bulan
Year	Satuan Tahun
Minute_Second	"Menit : Detik"
Hour_Minute	"Jam : Menit"
Day_Hour	"Hari : Jam"
Year_Month	"Tahun : Bulan"
Hour_Second	"Jam : Detik"

   Contoh:

   Menampilkan tanggal setalah ditambah 10 hari:

   Sintaks : `SELECT ADDDATE('2016-09-21', Interval 10 Day);`

   (Hasilnya setelah ditambah 10 hari)

```
mysql> SELECT ADDDATE('2016-09-21', Interval 10 day);
+-----------------------------------------+
| ADDDATE('2016-09-21', Interval 10 day) |
+-----------------------------------------+
| 2016-10-01                              |
+-----------------------------------------+
```

2. **CURDATE()**

   Menghasilkan tanggal saat ini, namun tidak seperti halnya fungsi now() yang disertai dengan waktu.

Contoh:

Menampilkan tanggal hari ini.

Sintaks: `SELECT CURDATE()Tanggal_Hari_Ini;`

```
mysql> SELECT CURDATE()Tanggal_Hari_Ini;
+-------------------+
| Tanggal_Hari_Ini  |
+-------------------+
| 2019-07-18        |
+-------------------+
```

3. CURTIME()

Menghasilkan waktu terkini

Contoh:

Menampilkan waktu saat ini.

Sintaks: SELECT CURTIME

```
mysql> SELECT CURTIME()Waktu_Saat_Ini;
+-----------------+
| Waktu_Saat_Ini  |
+-----------------+
| 12:45:44        |
+-----------------+
```

4. CURRENT_TIMESTAMP()

Menampilkan tanggal saat ini berikut dengan jam, menit dan detik.

Contoh:

Menampilkan tanggal dan waktu saat ini.

Sintaks: SELECT CURRENT_TIMESTAMP()

Tanggal_Dan_Tanggal_Saat_Ini;

```
mysql> SELECT CURRENT_TIMESTAMP()Tanggal_Dan_Tanggal_Saat_Ini;
+----------------------------+
| Tanggal_Dan_Tanggal_Saat_Ini |
+----------------------------+
| 2019-07-18 13:40:07        |
+----------------------------+
```

5. DAYNAME

Berfungsi untuk menampilkan nama hari sesuai dengan tanggal saat itu.

Contoh:

Menampilkan nama hari untuk tanggal "2000-08-12".

Sintaks: SELECT DAYNAME('2008-08-12')
TANGGAL_TERSEBUT_HARI;

```
mysql> SELECT DAYNAME('2000-08-12')Tanggal_Tersebut_Hari;
+-----------------------+
| Tanggal_Tersebut_Hari |
+-----------------------+
| Saturday              |
+-----------------------+
```

6. DAYOFMONTH

Berfungsi untuk menampilkan tanggal pada suatu format penanggalan dari sebulan.

Contoh:

Menampilkan tanggal untuk '2000-08-12'

Sintaks: SELECT DAYOFMONTH('2000-08-12')Tanggal;

```
mysql> SELECT DAYOFMONTH('2000-08-12')Tanggal;
+----------+
| Tanggal  |
+----------+
|       12 |
+----------+
```

7. DAYOFWEEK

Berfungsi untuk menampilkan hari dari seminggu dengan menggunakan kode angka.

Daftar Kode Angka

Nama Hari	Kode Angka
Sunday	1
Monday	2
Tuesday	3
Wednesday	4
Thursday	5
Friday	6
Saturday	7

Contoh:

Menampilkan kode tanggal untuk tanggal "2000-08-12".

Sintaks: SELECT DAYOFWEEK("2000-08-12")Kode_Tanggalnya;

8. DAYOFYEAR

Berfungsi untuk menampilkan hari ke berapa dalam setahun.

Contoh:

Menampilkan hari ke pada tanggal '2000-08-12'.

Sintaks:SELECT DAYOFYEAR("2000-08-12")Saat_Ini_Hari_Ke;

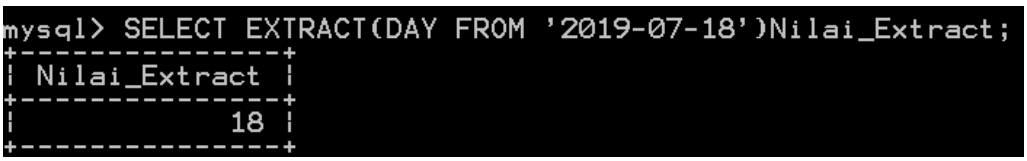
9. EXTRACT

Fungsi extract ini dapat mengambil bagian dari tanggal, bulan, atau tahun saja dari suatu penanggalan. Juga dapat mengambil bagian dari jam, menit, atau detik dari suatu pengaturan waktu.

Contoh

Menampilkan extract dari penanggalan '2019-07-18').

Sintaks: select Extract(Day From"2006-05-07")Nilai_Extractnya;



10. FROM_DAYS

Fungsi ini berguna dalam mengubah nilai menjadi bentuk penanggalan. Namun nilai tersebut harus hanya terdiri dari 6 angka. Contoh: 123456, 275830.

```
mysql> SELECT FROM_DAYS(123456) Penanggalannya_Adalah;
+-----------------------+
| Penanggalannya_Adalah |
+-----------------------+
| 0338-01-05            |
+-----------------------+
```

```
mysql> SELECT FROM_DAYS(275830) Penanggalannya_Adalah;
+-----------------------+
| Penanggalannya_Adalah |
+-----------------------+
| 0755-03-14            |
+-----------------------+
```

11. HOUR

Berfungsi untuk mengambil nilai jam dari suatu pengaturan waktu.

Contoh:

```
mysql> SELECT HOUR('08:30:15')Jamnya_Adalah;
+----------------+
| Jamnya_Adalah  |
+----------------+
|              8 |
+----------------+
```

12. MINUTE

Berfungsi untuk mengambil nilai menit dari suatu pengaturan waktu.

Contoh:

```
mysql> SELECT MINUTE('08:30:15')Menitnya_Adalah;
+-----------------+
| Menitnya_Adalah |
+-----------------+
|              30 |
+-----------------+
```

13. SECOND

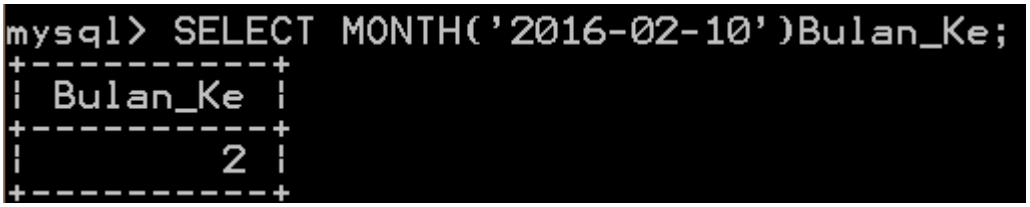
Berfungsi untuk mengambil nilai detik dari suatu pengaturan waktu.

Contoh:

```
mysql> SELECT SECOND('08:30:15')Menitnya_Adalah;
+-----------------+
| Menitnya_Adalah |
+-----------------+
|              15 |
+-----------------+
```

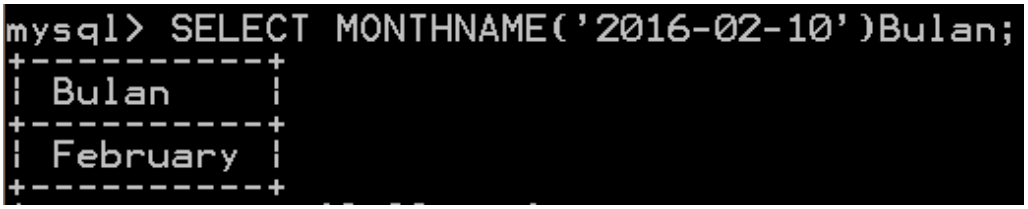
14. MONTH

Berfungsi untuk mengambil nilai bulan dari suatu pengaturan tanggal.



15. MONTHNAME

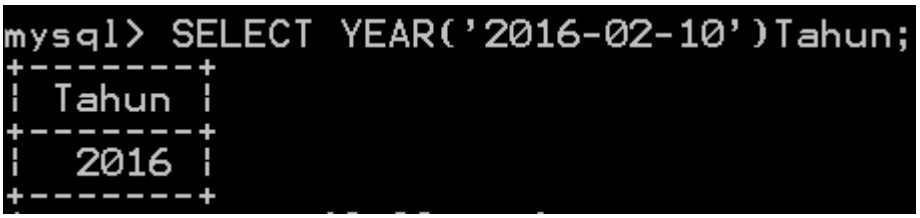
Berfungsi untuk mengambil nilai bulan dari suatu pengaturan tanggal namun diubah menjadi nama bulan.



16. YEAR

Menampilkan tahun dalam penanggalan.

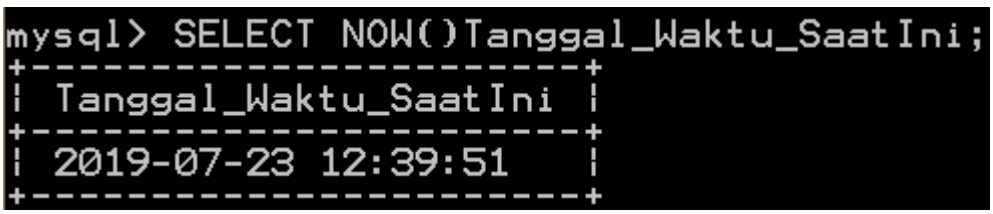
Contoh :



17. NOW()

Menampilkan penanggalan saat ini berikut dengan waktu saaat ini. Fungsi now ( ) sama dengan fungsi current_timestamp( ).

Contoh:



18. TO_DAYS

Menampilkan nilai penanggalan dari suatu penanggalan.

Contoh:

```
mysql> SELECT TO_DAYS('2016-02-10')Nilainya;
+----------+
| Nilainya |
+----------+
|   736369 |
+----------+
```

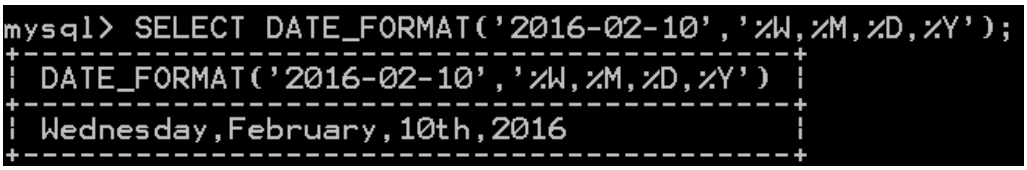
19. DATE_FORMAT

Berfungsi untuk merubah nilai penanggalan atau pengaturan waktu dengan format yang diinginkan. Berikut ini merupakan daftar simbol format:

SIMBOL FORMAT	KETERANGAN
%M	Menampilkan nama bulan bukan dalam bentuk singkatan
%m	Menampilkan bulan dengan angka
%b	Menampilkan nama bulan dalam bentuk singkatan.
%W	Menampilkan nama hari bukan dalam bentuk singkatan.
%D	Menampilkan nilai hari ke dalam sebulan.
%Y	Menampilkan tahun dengan format 4 digit.
%y	Menampilkan tahun dengan format 2 digit.
%j	Menampilkan nomor hari dalam setahun.
%a	Menampilkan nama hari dalam bentuk singkatan.
%d	Menampilkan nomor hari dalam sebulan.
%r	Menampilkan jam dalam format 12 jam.

%T	Menampilkan jam dalam format 24 jam.
%H	Menampilkan jam dalam format 24 jam : 00-23.
%h	Menampilkan jam dalam format 12 jam : 00-12.
%S	Menampilkan detik

Contoh :



Latihan

1. Apa perintah untuk menampilkan tanggal setelah ditambah 2 bulan ("2012-05-06")
2. Apa perintah menampilkan tanggal & waktu sekarang
3. Tampilkan nama hari dari tanggal "2012-05-06"
4. Tampilkan nama bulan dari tanggal "2012-05-06"
5. Tampilkan nilai penanggalan "2010-05-06" dengan format (nama hari, nama bulan, nilai hari, nilai tahun dengan 4 digit)

BAB 8

FUNGSI STRING

1. ASCII(X)

Berfungsi untuk mencari nilai Ascii dari suatu string.

Contoh:

```
mysql> SELECT ASCII('%')Nilai_Ascii;
+-------------+
| Nilai_Ascii |
+-------------+
|          37 |
+-------------+
```

2. CHAR(X1,X2,...)

Fungsi ini merupakan kebalikan dari fungsi ascii(x), jika pada fungsi ascii (x) mengembalikan string menjadi nilai ascii. Fungsi char(x1,…) adalah mengubah fungsi ascii menjadi karakter.

Contoh:

```
mysql> SELECT CHAR(37,65,43,37,66)Karakter;
+----------+
| Karakter |
+----------+
| %A+%B    |
+----------+
```

3. CHAR_LENGTH(STRING) ATAU LENGTH(STRING)

Char_length(string) memiliki fungsi yng sama dengan length(string), yaitu untuk menghitung jumlah karakter pada sebuah string.

Contoh:

```
mysql> SELECT CHAR_LENGTH("Semangat Belajar")
    -> Jumlah_Karakter;
+----------------+
| Jumlah_Karakter |
+----------------+
|              16 |
+----------------+
```

Atau

```
mysql> SELECT LENGTH("Semangat Belajar Database")
    -> Jumlah_Karakter;
+-----------------+
| Jumlah_Karakter |
+-----------------+
|              25 |
+-----------------+
```

4. ENCODE(STRING,STRING_ENKRIPSI)

Berfungsi untuk merubah nilai string menjadi kode-kode tertentu. Dalam merubah string menjadi kode harus terdapat string dan string_enkripsi. Jika hanya terdapat string saja maka proses tidak dapat dilakukan.

Contoh:

```
mysql> SELECT ENCODE("Database","MySql")Kode;
+-----------+
| Kode      |
+-----------+
| dL┤®kÈùq |
+-----------+
```

5. LEFT(STRING,Nilai pengambilan dari kiri)

Berfungsi untuk mengambil karakter terkiri dari suatu string dengan jumlah pengambilan karakter dari kiri.

Contoh:

```
mysql> SELECT LEFT("INDONESIA",3)Tiga_Digit_Dari_Kiri;
+----------------------+
| Tiga_Digit_Dari_Kiri |
+----------------------+
| IND                  |
+----------------------+
```

6. MID(string,posisi,nilai pengambilan dari posisi)

Berfungsi untuk mengambil karakter dari suatu string, sebelumnya harus menentukan terlebih dahulu posisi pengambilan. Setelah itu baru menentukan jumlah karakter yang akan diambil dari posisi yang telah ditentukan.

Contoh:

```
mysql> SELECT MID("INDONESIA",4,3)3Digit_Dari_DIGIT_KE4;
+-----------------------+
| 3Digit_Dari_DIGIT_KE4 |
+-----------------------+
| ONE                   |
+-----------------------+
```

7. RIGHT(string,nilai pengambilan dari kanan)

Berfungsi untuk mengambil karakter terkanan dari suatu string dengan jumlah pengambilan karakter dari kanan.

Contoh:

```
mysql> SELECT RIGHT("INDONESIA",3)Tiga_Digit_Dari_KANAN;
+-----------------------+
| Tiga_Digit_Dari_KANAN |
+-----------------------+
| SIA                   |
+-----------------------+
```

8. LCASE(string) Atau LOWER(string)

Berfungsi untuk merubah tipe karakter dari suatu string dari huruf kapital menjadi huruf kecil.

Contoh:

```
mysql> SELECT LCASE("INDAHNYA KEBERSAMAAN")Hasilnya;
+-----------------------+
| Hasilnya              |
+-----------------------+
| indahnya kebersamaan  |
+-----------------------+
```

9. UCASE(string) Atau UPPER(string)

Fungsi ucase(string) atau upper(string) adalah merupakan kebalikan dari fungsi lcase(string), yaitu untuk merubah huruf kecil menjadi huruf kapital atau besar.

Contoh:

```
mysql> SELECT UPPER("indahnya kebersamaan")Hasilnya;
+-----------------------+
| Hasilnya              |
+-----------------------+
| INDAHNYA KEBERSAMAAN  |
+-----------------------+
```

10. LTRIM(string)

Berfungsi untuk menghilangkan atau menghapus spasi dari bagian kiri suatu string.

Contoh:

```
mysql> SELECT LTRIM("        Meraih Impian")Hasilnya;
+----------------+
| Hasilnya       |
+----------------+
| Meraih Impian  |
+----------------+
```

11. RTRIM(string)

Berfungsi untuk menghilangkan atau menghapus spasi dari bagian kanan suatu string.

Contoh:

```
mysql> SELECT RTRIM("Meraih Mimpi         ")Hasilnya;
+---------------+
| Hasilnya      |
+---------------+
| Meraih Mimpi  |
+---------------+
```

12. TRIM(string)

Berfungsi untuk menghilangkan atau menghapus spasi dari bagian kiri dan kanan suatu string.

Contoh:

```
mysql> SELECT TRIM("        Meraih Mimpi         ")Hasilnya;
+---------------+
| Hasilnya      |
+---------------+
| Meraih Mimpi  |
+---------------+
```

13. PASSWORD(string)

Berfungsi untuk merubah suatu string menjadi kode sandi yang telah dienkripsi.

Contoh:

```
mysql> SELECT PASSWORD("DATABASE")PASSWORDNYA;
+-------------------------------------------+
| PASSWORDNYA                               |
+-------------------------------------------+
| *5A2F344EB2B7CBD5636DE81FD6B4D0A28FFF2B10 |
+-------------------------------------------+
```

LATIHAN :

1. Tampilkan nilai ASCII dari "A"!
2. Tampilkan jumlah karakter dari string "Praktikum Sistem Basis Data"!

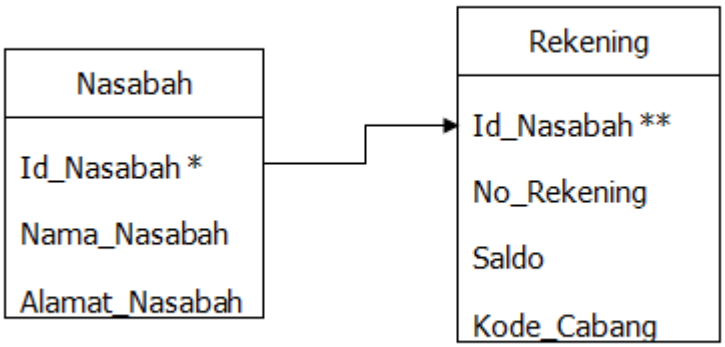
Diket : S1 ="ALBY PUTRA FAHRI"

3. Tampilkan pengambilan string dari S1 dengan hasil PUTRA!
4. Tampilkan pengambilan string dari S1 dengan hasil AL!
5. Tampilkan pengambilan string dari S1 dengan hasil FAHRI!

BAB 9
DATABASE RELASIONAL

A. Pengertian Database Relasional

Database Relasional merupakan suatu konsep penyimpanan data terstruktur. Teori database relasional di kemukakan pertamakali oleh Dr. E.F. Codd. Dalam database relasional, data disimpan dalam bentuk relasi atau tabel dua dimensi, dan antara tabel satu dengan tabel lainnya terdapat hubungan atau relationship sehingga dapat di simpulkan, database adalah kumpulan dari sejumlah tabel yang saling hubungan atau saling keterkaitan.



Ada 3 macam relasi tabel, diantaranya:

1. One to One (1-1)

   Mempunyai pengertian "Setiap baris data pada tabel pertama dihubungkan hanya ke satu baris data pada tabel ke dua". Contohnya: relasi antara tabel mahasiswa dan tabel orang tua. Satu baris mahasiswa hanya berhubungan dengan satu baris orang tua begitu juga sebaliknya.

2. One to Many (1-M)

   Mempunyai pengertian "Setiap baris data dari tabel pertama dapat dihubungkan ke satu baris atau lebih data pada tabel ke dua". Contohnya : relasi perwalian antara tabel dosen dan tabel mahasiswa.

Satu baris dosen atau satu dosen bisa berhubungan dengan satu baris atau lebih mahasiswa

3. Many to Many (M-N)

   Mempunyai pengertian "Satu baris atau lebih data pada tabel pertama bisa dihubungkan ke satu atau lebih baris data pada tabel ke dua". Artinya ada banyak baris di tabel satu dan tabel dua yang saling berhubungan satu sama lain. Contohnya : relasi antar tabel mahasiswa dan tabel mata kuliah. Satu baris mahasiswa bisa berhubungan dengan banyak baris mata kuliah begitu juga sebaliknya.

Database relational atau normalisasi adalah suatu proses untuk merelasikan field dari tabel yang satu dengan tabel lainnya, di mana dalam salah satu tabel terdapat field yang bersifat primary key atau foreign key.

Model relasional digunakan untuk mengatasi kesulitan dalam pengelolaan dan pengaksesan data. Alasan yang melandasi mengapa dibutuhkan relasi tabel, yaitu karena terdapatnya anomali-anomali (error atau inkonsistensi data) yang harus dihindari, agar kebutuhan data dan kepastian data terjamin. Anomali-anomali itu meliputi anomali update, insert, dan delete.

1. Anomali INSERT

   Kesalahan yang terjadi di saat proses penyisipan record baru.

   Contoh:

NPM	Kode_MK	Semester	SKS
201343001	KU001	1	2
201343001	KU002	1	2
201343002	KK001	1	2
201343002	KK002	1	2
201343002	KU003	1	2

   Jika kampus akan mengadakan matakuliah baru dengan kode matakuliah KK005, maka proses penyisipan untuk kode matakuliah

KK005 tidak dapat dilakukan sampai ada mahasiswa yang mengambil matakuliah tersebut.

2. Anomali DELETE

   Kesalahan yang terjadi di saat proses penghapusan record atau tuple.

   Contoh:

NPM	Kode_MK	Semester	SKS
201343001	KU001	1	2
201343001	KU002	1	2
201343002	KK001	1	2
201343002	KK002	1	2
201343002	KU003	1	2

   Mahasiswa yang memiliki NPM 201343002 memutuskan untuk membatalkan mengambil matakuliah KU003, maka dengan demikian jika didelete record tersebut, akan berakibat hilangnya informasi tentang kode matakuliah KU003.

3. Anomali UPDATE

   Anomali atau kesalahan yang terjadi pada saat proses suatu record.

   Contoh:

NPM	Kode_MK	Semester	SKS
201343001	KU001	1	2
201343001	KU002	1	2
201343002	KU001	1	2
201343002	KU002	1	2
201343002	KU003	1	2
201343003	KU002	1	2
201343003	KU003	1	2

   Jika Anda perhatikan pada tabel kuliahan, terdapat banyak kode matakuliah yang diambil oleh mahasiswa dengan NPM yang berbeda. Misalnya akan dilakukan perubahan SKS untuk kode matakuliah

KU002, maka akan dilakukan proses update beberapa kali sesuai dengan banyaknya jumlah yang mengambil kode matakuliah tersebut.

B. Normaliasasi

Teknik/pendekatan yang digunakan dalam membangun disain logik database relasional melalui organisasi himpunan data dengan tingkat ketergantungan fungsional dan keterkaitan yang tinggi sedemikian sehingga menghasilkan struktur tabel yang normal.

Proses pembentukan Normalisasi terdiri dari: Bentuk tidak normal (Unnormalized Form), Bentuk 1NF (First Normal Form), Bentuk 2NF (Second Normal Form), Bentuk 3NF (Third Normal Form).

1. Bentuk Tidak Normal (unnormalized form)

Kobuk	**Judul**	**Pengarang**	**RK1**	**RK2**	**RK3**
101-K	Belajar Sendiri VB. 6	M. Bakri	001	003	001
103-C	Kisah Cinta Sang Penyair	Minolsta	003	001	
110-S	Belajar Cepat Berhitung	Cokro S.	002	002	001
106-A	Bertauhid Yang Benar	Ust. Soleh	003		003

2. Bentuk Normal Pertama (INF)
   a. Tiap field harus bernilai "atomik, yaitu setiap recordnya hanya terdiri dari satu nilai.
   b. Setiap field harus hanya memiliki satu pengertian dan memiliki nama yang unik.
   c. Tidak terdapat record yang sama atau bernilai ganda.

Kobuk	Judul	Pengarang	Kode_RK
101-K	Belajar Sendiri VB. 6	M. Bakri	001
101-K	Belajar Sendiri VB. 6	M. Bakri	003
103-C	Kisah Cinta Sang Penyair	Minolsta	001
103-C	Kisah Cinta Sang Penyair	Minolsta	003
110-S	Belajar Cepat Berhitung	Cokro S.	001
110-S	Belajar Cepat Berhitung	Cokro S.	002
106-A	Cara Cepat Belajar Database	Lina Marlina	003

3. Bentuk Normal Kedua (2NF)

   Apabila bentuk normal pertama atau 1NF terpenuhi, maka dapat menuju ke bentuk normal kedua atau 2NF. Pada tahap ini tabel yang terdapat pada bentuk normal pertama atau 1NF akan dibagi menjadi 2 bagian

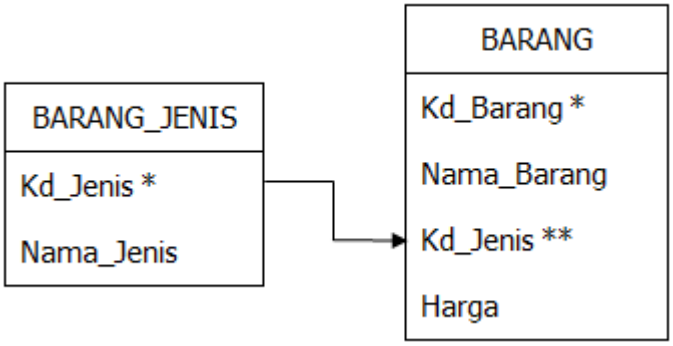
Tabel Buku

Kobuk	Judul	Pengarang
101-K	Belajar Sendiri VB. 6	M. Bakri
103-C	Kisah Cinta Sang Penyair	Minolsta
110-S	Belajar Cepat Berhitung	Cokro S.
106-A	Cara Cepat Belajar Database	Lina Marlina

Tabel Rak Buku

Kobuk	Kode_RK
101-K	001
101-K	003
103-C	001
103-C	003
110-S	001
110-S	002
106-A	003

BAB 10
RELASI DUA TABEL

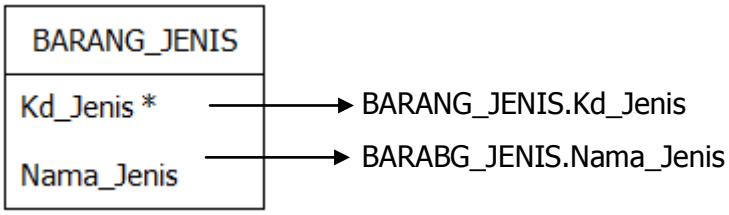


Gambar 10.1 relasi antara table Barang dan Jenis

A. Teori Pengambilan Kolom Tabel Relasi

Dari gambar 10.1 hubungan data dari satu ke banyak (one to Many). Artinya satu jenis barang memungkinkan memiliki anggota atau tipe barang lebih dari satu. Misalnya hardisk memiliki tipe hardisk seageate 40 GB, seageate 60 GB, WDC 40 GB dan lain sebagainya. Jadi satu jenis barang bisa memiliki anggota banyak barang.

Pada saat kita membuat query pada beberapa tabel yang berelasi, anda harus menyebutkan nama tabel dan kolom yang hendak diambil. Teknik yan paling mudah adalah dengan menghubungkan antara nama tabel dan kolom dengan titik (**.**). Cara penulisan query untuk table **BARANG_JENIS**, lihat gambar dibawah ini :



Gambar 10.2 Analogi pengambilan kolom table BARANG_JENIS

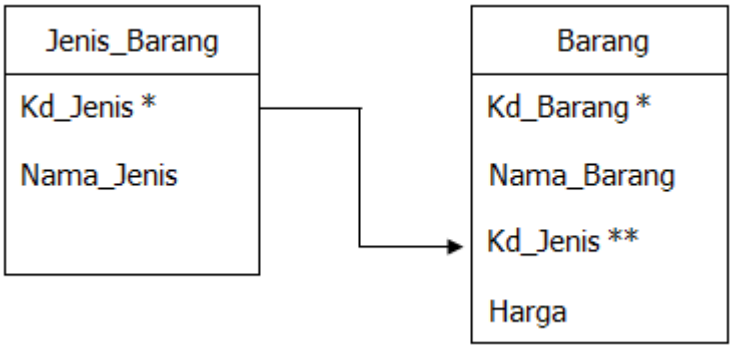
B. Teknik Dasar Relasi Dua Tabel

```
SINTAKS : (Relasi 2 tabel dengan kondisi)
SELECT tabelA.kolom1, tabelB.kolom1, tabelA.kolom2
FROM tabelA, tabelB
WHERE tabelA.kolom1(primary key) = tabelB.(foreign key);
```

Keterangan :

- tabelA.kolom1 : perintah untuk mengambil data didalam kolom bernama **kolom1** pada **tableA**
- tabelA.kolom2 : perintah untuk mengambil data didalam kolom bernama **kolom2** pada **tableA**
- tabelB.kolom1 : perintah untuk mengambil data didalam kolom bernama **kolom1** pada **tableB**
- tabelA.kunciA : nama kolom yang menjadi kunci utama(primer) pada table A
- tabelB.kunciA : nama kolom tamu yang berasal dari **tabelA**, kolom ini menjadi tamu pada **tabelB** karena adanya relasi.

```
SINTAKS : (Relasi 2 tabel dengan kondisi)
SELECT tabelA.kolom1, tabelB.kolom1, tabelA.kolom2
FROM tabelA, tabelB
WHERE tabelA.kolom1(primary key) = tabelB.(foreign key)
AND Kondisi;
```



Gambar 10.3 Model Relasional 2 Tabel

Contoh:

Tampilkan kode barang, jenis barang dan nama barang

Langkah-langkah yang perlu diperhatikan:

1. Tentukan tabel yang dibutuhkan yaitu tabel Jenis_Barang dan tabel Barang
2. Untuk menampilkan kode barang terdapat pada ditabel Barang, untuk menampilkan jenis barang terdapat ditabel Jenis_Barang dan untuk menampilkan nama barang terdapat pada tabel Barang
3. Tentukan relasi dari kedua tabel tersebut, yaitu pada tabel Jenis_Barang id relasinya Kd_Jenis dan pada tabel Barang id relasinya Kd_Jenis.
4. Buat perintah SQLnya: **SELECT Barang.Kd_Barang, Jenis_Barang.Nama_Jenis, Barang.Nama_Barang FROM Jenis_Barang, Barang WHERE Jenis_Barang.Kd_Jenis = Barang.Kd_Jenis;**

LATIHAN :

STRUKTUR TABEL :

Fields

Name	Type	Primary	Not Null	Default
Departement_id	char(2)	☑	☑	
Departement_nan	varchar(20)	☐	☑	
Manager_id	char(3)	☐	☑	

Fields

Name	Type	Primary	Not Null	Default
Employee_id	char(3)	☑	☑	
Name	varchar(20)	☐	☑	
Salary	integer	☐	☑	
Manager_id	char(3)	☐	☑	
Departement_id	char(2)	☐	☑	

ISI TABEL :

Departement_id	Departement_name	Manager_id
Click here to define a filter		
10	Administration	101
20	IT	103

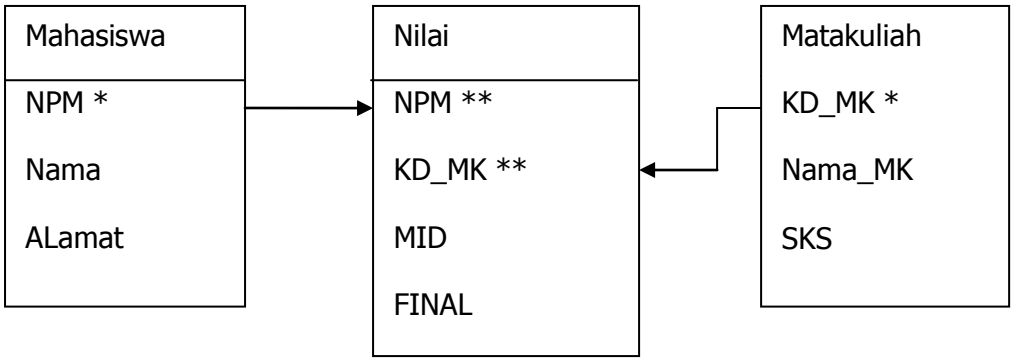
Employee_id	Name	Salary	Manager_id	Departement_id
Click here to define a filter				
100	Steven	8000	101	10
101	Lexa	10000	101	10
102	Bruce	9000	103	20
103	Diana	11000	103	20
104	Bruce	8500	103	20

Pertanyaan :

1. Tampilkan nama employee yan memiliki salary lebih dari 9000
2. Tampilkan id dan nama dari employee yang memiliki nama diakhiri dengan huruf n
3. Hitung jumlah employee yang memilki nama Bruce

4. Tampilkan id department dan nama departement dari employee yang bernama Lexa
5. Tampilkan id department yang total salary semua employee yang bekerja pada department tersebut lebih dari 20000

BAB 11
RELASI 3 TABEL



Contoh :

Untuk menampilkan nama mahasiswa, nama matakuliah dan nilai MID, perintah sqlnya adalah **SELECT Mahasiswa.Nama, Matakuliah.Nama_MK, Nilai.MID FROM Mahasiswa, Nilai, Matakuliah WHERE Mahasiswa.NPM = NILAI.NPM and Nilai.KD_MK = Matakuliah.KD_MK;**

Latihan

Buatlah 3Tabel yaitu Tabel Mahasiswa, Matakuliah dan Nilai!

Mahasiswa

NPM	Nama	Alamat
12196076	Alya	Bogor
11196779	Didi	Jakarta
12196324	Tata	Depok
10196839	Vinka	Bekasi
12196999	Sely	Jakarta
10196778	Dhani	Bogor

Matakuliah

KD_MK	Nama_MK	SKS	Semester
KK021	Sistem Basis Data	2	3
KD132	SIM	3	3
KU122	Pancasila	2	2

Nilai

NPM	KD_MK	MID	FINAL
12196076	KK021	60	70
11196779	KK021	80	90
12196324	KK021	50	40
10196839	KU122	90	80
12196999	KD132	75	75
10196778	KD132	80	0
12196076	KD132	40	30

Pertanyaan:

1. Tampilkan nama mahasiwa dan nama matakuliah yang nilai midnya antara 70 sampai 80
2. Tampilkan nama mahasiswa yang mengambil matakuliah Sistem Basis Data
3. Tampilkan nama mahasiswa, nilai final dan nama matakuliah
4. Tampilkan nama mahasiswa yang nilai mid kurang dari 80 dan mengambil matakuliah sistem basis data

BAB 12
RELASI DENGAN JOIN

Join adalah penggabungan data yang berasal dari beberapa table. Operator yang biasa digunakan adalah sama dengan(=), maka sering disebut dengan *equality join* atau *equijoin.*

Equijoin dikelompokkan ke dalam dua bagian yaitu iiner equijoin (inner join) dan outer equijoin (outer join). Yang termasuk dalam outer join adalah left jon dan right join.

Bentuk penggabungan data yang terakhir yang akan dibahas adalah perkalian Cartesian (Cartesian product) atau disebut juga dengan cross join atau full join.

1. JOIN/INNER JOIN

Akan menghasilkan baris-baris yang cocok antar kedua table paling tidak satu baris.

Sintaks :

```
SELECT column_name(s)
From table_name1
INNER JOIN table_name2
ON table_name1.column_name = table_name2.column_name;
```

2. LEFT JOIN

Akan menampilkan seluruh baris dari table di sebelah kiri (tabel1), walaupun tidak ada cocok dengan table di sebelah kanan (tabel2).

Sintaks :

```
SELECT column_name(s)
From table_name1
LEFT JOIN table_name2
ON table_name1.column_name = table_name2.column_name;
```

3. RIGHT JOIN

Akan menampilkan seluruh baris dari table di sebelah kanan (tabel2), walaupun tidak ada cocok dengan table di sebelah kiri (tabel1).

Sintaks :

```
SELECT column_name(s)
From table_name1
RIGHT JOIN table_name2
ON table_name1.column_name = table_name2.column_name;
```

4. FULL JOIN

Akan menampilkan baris-baris yang cocok dari salah satu tabel

Sintaks :

```
SELECT column_name(s)
From table_name1
FULL JOIN table_name2
ON table_name1.column_name = table_name2.column_name;
```

CONTOH :

Database Penjualan

Pelanggan

P_id	Nama	Alamat	Kota
1	Hani	Jl. Bunga	Jakarta Timur
2	Stefan	Jl. Ikan	Jakarta Barat
3	Pipit	Jl. Buah	Jakarta Selatan

Pesan

O_id	noOrder	P_id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	1

1. JOIN/INNER JOIN

Menampilkan nama dan nomor order yang diurutkan berdasarkan nama

Sintaks :

```
SELECT pelanggan.nama, pesan.noOrder
From Pelanggan
INNER JOIN pesan
ON pelanggan.p-id = pesan.p_id
ORDER BY pelanggan.nama;
```

Hasil :

Nama	noOrder
Hani	22456
Hani	24562
Hani	34764
Pipit	77895
Pipit	44678

2. LEFT JOIN

Menampilkan nama dan nomo r order yang diurutkan berdasarkan nama

Sintaks :

```
SELECT pelanggan.nama, pesan.noOrder
From Pelanggan
LEFT JOIN pesan
ON pelanggan.p-id = pesan.p_id
ORDER BY pelanggan.nama;
```

Hasil :

Nama	noOrder
Hani	22456
Hani	24562
Pipit	77895

Pipit	44678
Stefan	Null

3. RIGHT JOIN

Menampilkan nama dan nomo r order yang diurutkan berdasarkan nama

Sintaks :

```
SELECT pelanggan.nama, pesan.noOrder
From Pelanggan
RIGHT JOIN pesan
ON pelanggan.p-id = pesan.p_id
ORDER BY pelanggan.nama;
```

Hasil :

Nama	noOrder
Hani	22456
Hani	24562
Hani	34764
Pipit	77895
Pipit	44678

4. FULL JOIN/CROSS JOIN

Menampilkan nama dan nomo r order yang diurutkan berdasarkan nama

Sintaks :

```
SELECT pelanggan.nama, pesan.noOrder
From Pelanggan
CROSS JOIN pesan;
```

Hasil :

Nama	No_order
Hani	77895
Hani	44678
Hani	22456
Hani	24562
Hani	34764
Stefan	77895
Stefan	44678
Stefan	22456
Stefan	24562
Stefan	34764
Pipit	77895
Pipit	44678
Pipit	22456
Pipit	24562
Pipit	34764

LATIHAN :

Buat database Animal

Buat table berikut dan cari JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN

Animal

Id	animal
1	Cat
2	Dog
3	cow

Food

Id	Food
1	Milk
2	Bone
3	Grass

BAB 13
UNION, INTERSECT, EXCEPT

Pada pembahasan ini menggunakan tabel sebagai berikut :

Alumni

ID	Nama	Kota_Id
1	Anugrah	5
2	Diki Atmaja	8
3	Kresna	4

Mahasiswa

ID	Nama	Kelas	Kota_Id
1	Diana	A	3
2	Budi	B	2
3	Sari	C	1

Kelas A

ID	Nama	Kelas	Kota_Id
1	Diana	A	3
2	Liliana	A	2
4	Lulu	A	2

UNION

UNION merupakan operator yang digunakan untuk menggabungkan hasil query, dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama. Pemakaian UNION ada 2 jenis yaitu UNION dan UNION All. UNION ALL akan menampilkan seluruh data dari kedua tabel walaupun terdapat data yang sama. Syntax: **SELECT kondisi-1 UNION ALL kondisi-2;**

```
mysql> Select * From KelasA UNION ALL Select * From Mahasiswa;
+------+---------+-------+---------+
| ID   | Nama    | Kelas | Kota_Id |
+------+---------+-------+---------+
| 1    | Diana   | A     | 3       |
| 2    | Liliana | A     | 2       |
| 3    | Lulu    | A     | 2       |
| 1    | Diana   | A     | 3       |
| 2    | Budi    | B     | 2       |
| 3    | Sari    | C     | 1       |
+------+---------+-------+---------+
6 rows in set (0.00 sec)
```

UNION akan menampilkan seluruh data dari kedua tabel tetapi tidak menampilkan data yang sama. Data yang sama dianggap satu data (nilai dari setiap field sama). Syntax : **SELECT kondisi-1 UNION kondisi-2;**

```
mysql> Select * From Mahasiswa UNION Select * From KelasA;
+------+---------+-------+---------+
| ID   | Nama    | Kelas | Kota_Id |
+------+---------+-------+---------+
| 1    | Diana   | A     | 3       |
| 2    | Budi    | B     | 2       |
| 3    | Sari    | C     | 1       |
| 2    | Liliana | A     | 2       |
| 3    | Lulu    | A     | 2       |
+------+---------+-------+---------+
5 rows in set (0.00 sec)
```

INTERECT

INTERSECT merupakan operator yang digunakan untuk memperoleh data dari dua buah query dimana data yang ditampilkan adalah yang memenuhi kedua query tersebut dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama.

Syntax:

SELECT kondisi-1 INTERSECT kondisi-2;

SELECT kondisi-1 IN kondisi-2;

SELECT fields FROM table-1 where fields IN SELECT fields FROM table-2;

```
mysql> Select * From Mahasiswa where Kelas IN (Select Kelas From KelasA);
+------+-------+-------+---------+
| ID   | Nama  | Kelas | Kota_Id |
+------+-------+-------+---------+
| 1    | Diana | A     | 3       |
+------+-------+-------+---------+
1 row in set (0.30 sec)
```

EXCEPT

EXCEPT bertujuan untuk menampilkan data hasil pengurangan dari dua query atau sub query.

Syntax:

SELECT kondisi-1 EXCEPT kondisi-2;

SELECT kondisi-1 NOT IN kondisi-2;

SELECT fields FROM table-1 where fields NOT IN SELECT fields FROM table-2;

```
mysql> Select ID,Nama,Kelas From Mahasiswa Where Kelas NOT IN
    -> (Select Kelas From KelasA);
+------+------+-------+
| ID   | Nama | Kelas |
+------+------+-------+
| 2    | Budi | B     |
| 3    | Sari | C     |
+------+------+-------+
2 rows in set (0.00 sec)
```

Daftar Pustaka

Modul Praktikum Sistem Basis Data (MySql), Unindra Press Februari 2019

http://jenibastari.blogspot.com/2012/06/makalah-xampp-teori-sistem-basis-data.html?m=1

http://www.sicily-news.com/technology/xampp-pengertian-dan-bagian-serta-fungsinya/

https://ayooindonesia.wordpress.com/2015/04/12/pengertian-database-relasional/

https://www.academia.edu/12063129/BASIS_DATA_RELASIONAL