DBMS

Not Just the Fundamentals

PRACTICUM MODULE

This report discusses the DBMS Practicum Course and Database SQL Implementation with the PHP Programming Language, creating a simple system natively to introduce how databases are used properly in the world of coding.

This report discusses the DBMS Practicum Course and Database Implementation with the PHP Programming Language, creating a simple system natively to introduce how databases are used properly in the world of coding.

DAFTAR ISI

1. PENDAHULUAN

1.1. TUJUAN

Pada bagian ini anda diharapkan mampu memahami bagaimana menggunakan aplikasi database server dan client, menjalankan, membuat database, tabel, dan data menggunakan aplikasi client dan command prompt. Database server yang digunakan yaitu MySQL server dan aplikasi client yang digunakan adalah HeidiSQL, PHP MyAdmin. MySQL server yang digunakan adalah yang terdapat pada aplikasi Laragon. Aplikasi Laragon dapat diunduh di https://laragon.org/download/index.html dan aplikasi PHP MyAdmin dapat diunduh di https://www.phpmyadmin.net/downloads/ untuk aplikasi HeidiSQL sudah include di dalam Laragon saat di install. Aplikasi-aplikasi tersebut bersifat gratis/Opern Source.

Pada tahapan ini saya mengasumsikan bahwa anda mampu mengintegrasikan aplikasi PHP MyAdmin kedalam aplikasi Laragon agar bisa digunakan, namun jika masih belum bisa anda dapat mencari tutorial di situs online mengenai cara mengintegrasikan hal yang kita sebutkan di atas. Setelah berhasil mengintegrasikan baru anda dapat melanjutkan modul ini.

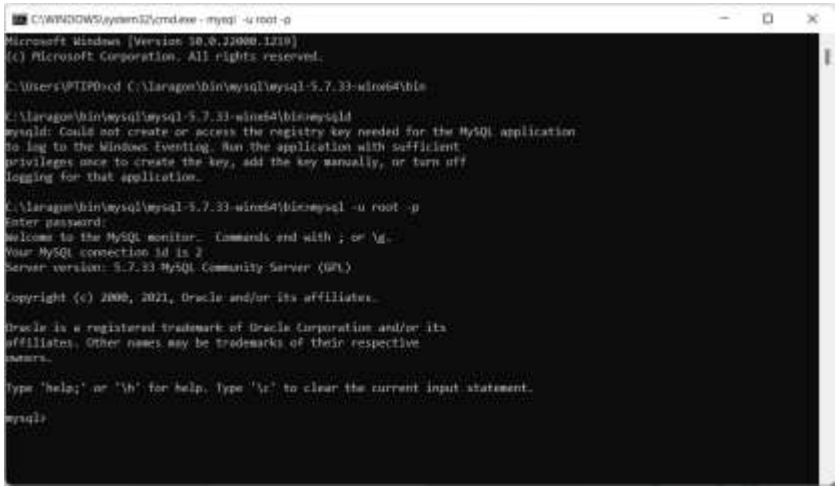
> Untuk mempelajari modul ini, anda diharapkan telah memahami konsep dasar basis data seperti konsep normalisasi data, konsep relasi, Data Definition Language (DDL) dasar, dan Data Manipulation Language (DML) dasar.

1.2. MENJALANKAN DATABASE SERVER

Setelah aplikasi Laragon telah selesai di unduh, silahkan pasang/install terlebih dahulu aplikasi tersebut. Untuk cara instalasinya sama seperti pada umumnya dan pada sesi instalasi anda akan diperintahkan untuk menentukan lokasi instalasi aplikasi tersebut, disini saya menyarankan untuk menginstal pada direktori D atau LocalDisk D, agar data tetap aman saat ada kebutuhan install ulang PC atau Laptop anda. setelah aplikasi berhasil terpasang anda dapat menjalankan aplikasi tersebut dan menjalankan servernya seperti pada gambar dibawah ini.

Gambar 1 : Menjalankan MYSQL Service dari Laragon

Selain itu, MySQL service juga dapat dijalankan (start) menggunakan command prompt dengan perintah “mysqld”. Lalu, untuk menghentikan (stop) MySQL Service nya menggunakan perintah “mysqladmin –u root shutdown” seperti pada gambar dibawah ini.



Gambar 2 : Menjalankan MySQL Service dari Command Prompt

1.3. PENGGUNAAN CMD (SEBAGAI MSQL CLIENT)

Berikut ini adalah cara penggunaan command prompt sebagai MySQL Client. MySQL Client digunakan sebagai antarmuka untuk login ke database server dan mengeksekusi query SQL. Langkah-langkahnya adalah sebagai berikut:

1. Jalankan command prompt (cmd.exe)
2. Masuk ke diroktori MySQL
3. Masukkan perintah :

```
mysql -u root -p
```

4. Selanjutnya membuat database dengan perintah query:

```
CREATE DATABASE latihan1;
```

5. Lalu pilih database yang ingin digunakan perintahnya :

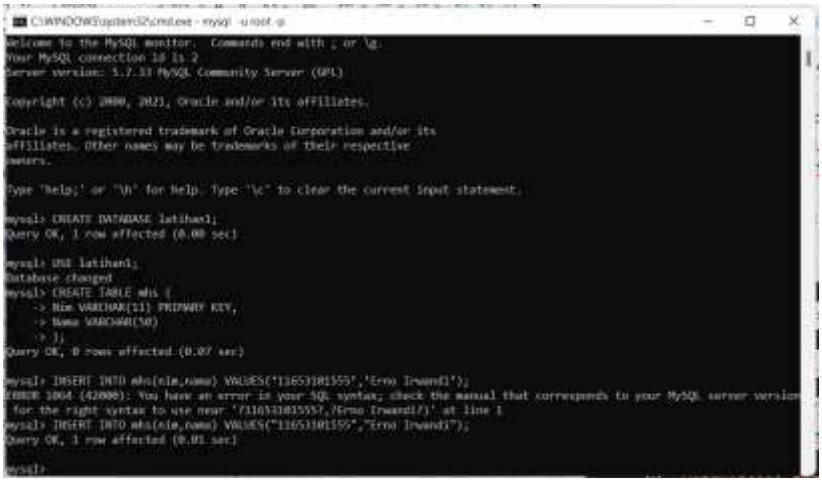
```
USE latihan1;
```

6. Selajutnya membuat tabel “mhs” yang terdiri dari nim dan nama, perintah query-nya seperti berikut ini:

```
CREATE TABLE mhs (
      Nim VARCHAR(11) PRIMARY KEY,
      Nama VARCHAR(50)
);
```

7. Lalu tambahkan satu contoh data mahasiswa dengan perintah query:

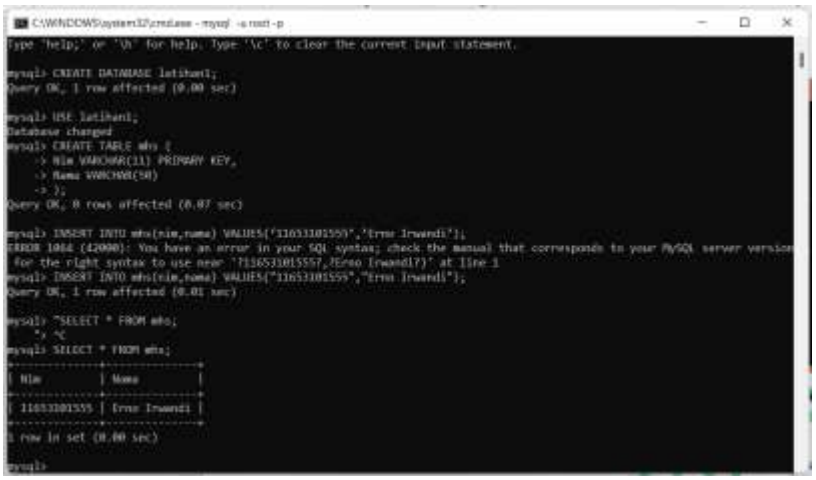
```
INSERT INTO mhs(nim,nama) VALUES(“11653101555”,”Erno
Irwandi”);
```



Gambar 3 : Membuat database menggunakan command prompt

8. Untuk melihat hasilnya, pilih database latihan1 dengan cara

```
SELECT * FROM mhs;
```



Gambar 4 : Melihat isi tabel mhs menggunakan command prompt

9.

Untuk melihat perintah-perintah (command line) pada MySQL bisa menggunakan perintah “help” setelah login ke MySQL seperti pada gambar dibawah ini.

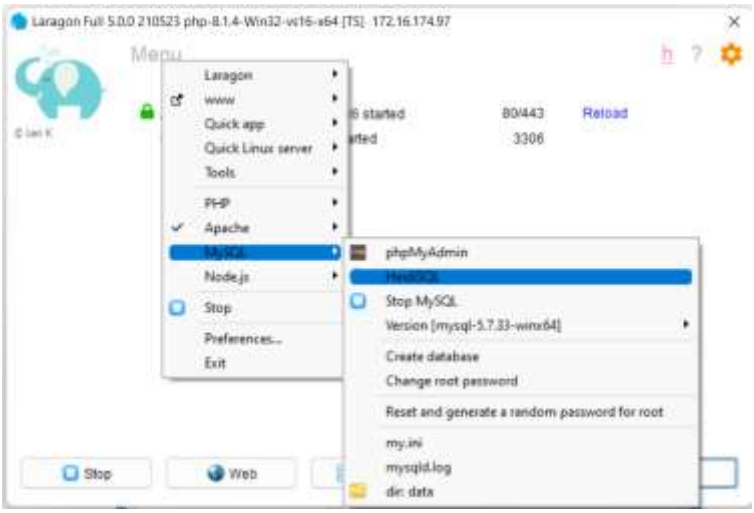
Gambar 5 : 5 MySQL help menggunakan command prompt

1.4. PENGGUNAAN PHP MYADMIN (SEBAGAI MSQL CLIENT)

1.5. PENGGUNAAN HEIDISQL (SEBAGAI MSQL CLIENT)

Berikut ini adalah langkah-langkah menggunakan HeidiSQL:

- Jalankan Aplikasi Laragon
- Pilih menu dan sorot bagian MySQL lalu pilih HeidiSQL



Gambar 6 : Memilih Database Client HeidiSQL

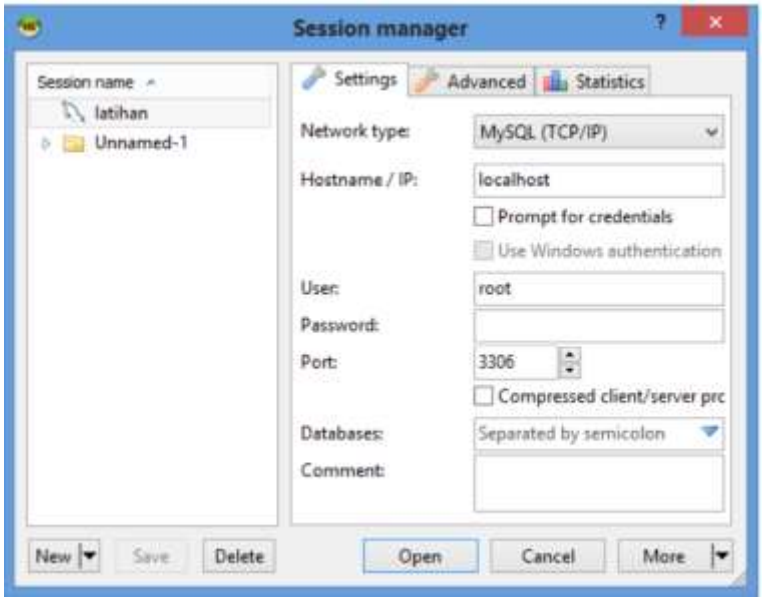
- Selanjutnya pilih menu database di laragon maka HeidiSQL akan otomatis muncul

Hal yang sama juga dapat dilakukan jika anda hendak beralih Kembali menggunkana PHP myAdmin, hal itu merupakan alasan saya merekomendasikan Laragon alih alih XAMPP karena lebih Flexible

- Selanjutnya Lakukan konfigurasi dasar untuk login ke user root yaitu:

  ✓ Hostname/IP : localhost

- ✓ User : root
- ✓ Password : <kosong>
- ✓ Port : 3306

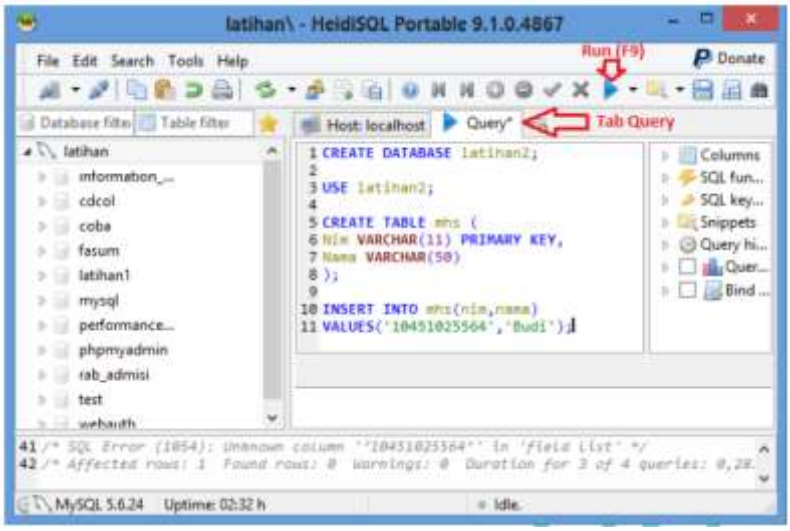


Gambar 7 : Mengatur konfigurasi dasar untuk login user root

- Lalu klik open, selanjutnya pilih tab Query dan masukkan perintah query berikut:

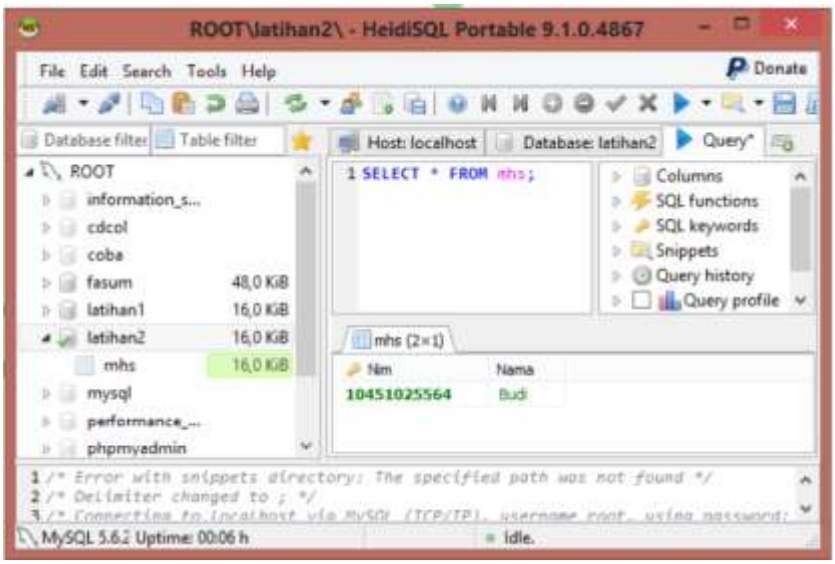
```
CREATE DATABASE latihan2;

USE latihan2;

CREATE TABLE mhs (
Nim VARCHAR(11) PRIMARY KEY,
Nama VARCHAR(50)
);

INSERT INTO mhs(nim,nama) VALUES("10451025564","Budi");
```

- Selanjutnya lakukan execute SQL / Run (F9)



Gambar 9 : Menjalankan Query

- Jika ingin melihat data menggunakan perintah Query, pilih database latihan2, selanjutnya masukkan query “SELECT * FROM mhs” pada tab Query, lalu klik Run (F9)



Gambar 8 : Melihat isi tabel mhs menggunakan HeidiSQL

1.6. MENAMPILAN OBYEK OBYEK DATABASE

Didalam database terdapat objek-objek database seperti database, table, column, index, variable, dan sebagainya. Perintah-perintah query yang digunakan untuk menampilkannya adalah sebagai berikut:

Tabel 1 : Perintah Query

No	**Query**
1	SHOW DATABASES;
2	SHOW TABLES;

No	Query
3	SHOW TABLES FROM NamaDatabase;
4	SHOW COLUMNS FROM NamaTabel;
5	SHOW INDEX FROM NamaTabel;
6	SHOW TABLE STATUS;
7	SHOW TABLE STATUS FROM NamaDatabase;
8	SHOW CREATE TABLE NamaTabel;
9	SHOW TABLES LIKE 'ma%';
10	SHOW VARIABLES;
11	SHOW VARIABLES LIKE 'have_isam';
12	SHOW VARIABLES LIKE 'have_bdb';
13	SHOW VARIABLES LIKE 'have_inno%';
14	SHOW TABLES LIKE 'NamaTabel';
15	SHOW TABLES FROM NamaDatabase LIKE 'NamaTabel';

2. DDL
(DATA DEFINITION LENGUAGE)

2.1. TUJUAN

Pada bab ini mahasiswa diharapkan mampu memahami perintah-perintah SQL pada Data Definition Languange (DDL). DDL merupakan kumpulan perintah SQL untuk membuat (create), mengubah (alter dan rename), dan menghapus (drop struktur dan definisi metadata pada objek-objek database. Contoh objek database seperti Database, Table, View, Index, Procedure, Function, dan Trigger. Perintah-perintah SQL setiap Database Management System (DBMS) kemungkinan memiliki perbedaan dalam penulisannya. Pada DBMS MySQL 5, perintah-perintah SQL pada DDL yang dapat digunakan dapat dilihat pada tabel berikut.

Tabel 2 : Pemetaan Perintah DDL

OBJEK	CREATE	ALTER	DROP	RENAME
DATABASE	√	√	√	
TABLE	√	√	√	√
VIEW	√	√	√	
INDEX	√		√	
PROCEDUR	√	√	√	
FUNCTION	√	√	√	
TRIGER	√		√	

Pada sesi ini yang akan dibahas adalah Database Table, View dan Index, sedangkan sisanya akan di bahas pada sesi Advance.

2.2. SQL CREATE

SQL Create digunakan untuk membuat objek-objek dalam database seperti untuk membuat database, table, view, index, procedure, function, dan trigger.

2.2.1. CREATE DATABASE

Perintah SQL untuk membuat Database :

```
CREATE DATABASE NamaDatabase;
```

Database yang telah dibuat dapat ditampilkan menggunakan perintah “SHOW DATABASES;”.

2.2.2. CREATE TABLE

Perintah SQL untuk membuat Table :

```
CREATE TABLE NamaTable (
  NamaField-1 TipeData(Ukuran),
  NamaField-2 TipeData(Ukuran),
  ...
  NamaField-n TipeData(Ukuran)
);
```

Biasanya setiap tabel harus memiliki primary key. Primary key dapat ditulis setelah tipedata sebelum koma(,) pada field yang dijadikan sebagai key. Tetapi ada juga yang menuliskan diakhir setelah nama field-nya ditulis semua.

Membuat primary key cara pertama :

```
CREATE TABLE NamaTable (
    NamaField-1 TipeData(Ukuran) PRIMARY KEY,
    NamaField-2 TipeData(Ukuran),
    ...
    NamaField-n TipeData(Ukuran)
);
```

Membuat primary key cara kedua :

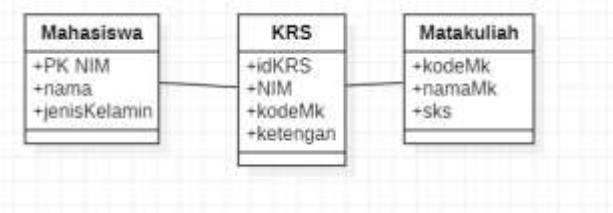
```
CREATE TABLE NamaTable (
    NamaField-1 TipeData(Ukuran),
    NamaField-2 TipeData(Ukuran),
    ...
    NamaField-n TipeData(Ukuran),
    PRIMARY KEY(NamaField-1)
);
```

Contoh:

```
CREATE TABLE MataKuliah (
    KodeMK VARCHAR(8),
    NamaMK VARCHAR(20),
    sks INT(11),
    PRIMARY KEY(KodeMK)
);
```

Satu table juga memungkinkan memiliki relasi dengan table lain. Pada SQL penulisan relasi dapat menggunakan FOREIGN KEY.

```
CREATE TABLE NamaTable1 (
    NamaField-1 TipeData(Ukuran),
    NamaField-2 TipeData(Ukuran),
    ...
    NamaField-n TipeData(Ukuran),
    PRIMARY KEY(NamaField-1),
    FOREIGN KEY(FieldForeignKey)
    REFERENCES NamaTable2(FieldPrimaryKey)
);
```



Gambar 10 : Contoh relasi antar tabel

Jika desain relasi database berbentuk seperti Gambar 2-1, maka penulisan SQLnya adalah sebagai berikut:

```
CREATE TABLE krs (
    IdKRS VARCHAR(20),
    IdMhs VARCHAR(11),
    KodeMK VARCHAR(8),
    Keterangan VARCHAR(50),
    PRIMARY KEY(IdKRS),
    FOREIGN KEY(IdMhs) REFERENCES Mahasiswa(Nim),
    FOREIGN KEY(KodeMK) REFERENCES MataKuliah(KodeMK)
);
```

Untuk menampilkan tabel-tabel yang telah dibuat dapat menggunakan perintah query:

```
USE NamaDatabase;
SHOW TABLES;
```

Contoh :

```
USE LatihanDB;
SHOW TABLES;
```

2.2.3. CREATE VIEW

View digunakan berfungsi sebagai tabel virtual yaitu tabel yang dibuat dari beberapa tabel atau view lain. Secara fisik, view bukan untuk menyimpan data tetapi hanya untuk menampilkan data dari beberapa tabel kedalam satu view. View biasanya digunakan untuk pembuatan laporan agar lebih mudah dalam pembuatan query. SQL view dibangun dari query SELECT baik dari satu atau beberapa tabel ataupun view.

Perintah SQL untuk membuat View :

```
CREATE VIEW NamaView AS
SELECT NamaField-1, NamaField-2,..., NamaField-n
FROM NamaTable
WHERE Kondisi;
```

Contoh:

```
CREATE VIEW MhsPria AS
SELECT * FROM Mahasiswa WHERE JenisKelamin="L";
```

Perintah Query tersebut akan menghasilkan sebuah tabel virtual untuk menampilkan data mahasiswa yang berjenis kelamin laki-laki saja.

2.2.4. CREATE INDEX

Index sangat berguna untuk pencarian data. Apabila suatu table tidak memiliki index, maka proses pencarian data akan memakan waktu lebih lama jika jumlah datanya cukup besar. Analoginya kita mencari nomor telepon seseorang yang bernama "Budi" pada sebuah buku telepon. Jika dalam buku telepon tidak memiliki index, maka kita harus mencari disemua halaman, tetapi jika buku telepon tersebut memiliki index, maka kita cukup mencari lewat index dari pemilik nomor telepon yang berawalan "B". Pada sebuah tabel dapat memiliki lebih dari satu index.

Perintah SQL untuk membuat Index :

```
CREATE INDEX NamaIndex ON NamaTable(NamaField);
```

Contoh:

```
CREATE INDEX NamaMhs ON Mahasiswa(Nama);
```

Selain INDEX, ada juga UNIQUE dan PRIMARY KEY. Pada dasarnya, PRIMARY KEY dan UNIQUE sama-sama digunakan untuk menjamin tidak adanya redudansi data pada field tertentu. PRIMARY KEY dan UNIQUE juga digunakan untuk indexing. Dengan kata lain, jika sebuah field diset menjadi PRIMARY KEY atau UNIQUE, maka field tersebut secara otomatis diset juga sebagai index, sehingga tidak perlu lagi membuat INDEX secara manual. Perbedaan antara PRIMARY KEY dan UNIQUE adalah data record field yang diset sebagai PRIMARY KEY tidak boleh berisi NULL, sedangkan UNIQUEdiperbolehkan berisi NULL. Contoh PRIMARY KEY misalnya di tabel mahasiswa adalah field "NIM" karena setiap mahasiswa harus memiliki NIM dan tidak mungkin ada yang sama. Contoh UNIQUE misalnya field "NomorSIM" pada tabel mahasiswa yang dijadikan UNIQUE. Nomor SIM tidak mungkin ada yang sama, tetapi bisa jadi seorang mahasiswa tidak memiliki SIM.

Untuk membuat UNIQUE, perintah query yang digunakan adalah sebagai berikut:

```
CREATE UNIQUE INDEX NamaIndex ON NamaTable(NamaField);
```

Contoh:

```
CREATE UNIQUE INDEX UnikNoSIM ON Mahasiswa(NomorSIM);
```

2.3. SQL ALTER AND RENAME

SQL Alter digunakan untuk mengubah struktur objek-objek dalam database seperti untuk mengubah database, table, view, procedure, dan function, sedangkan Rename digunakan untuk mengubah nama table.

2.3.1. ALTER DATABASE

Fungsi perintah query ALTER DATABASE adalah untuk mengubah karakteristik suatu database, tetapi perintah ini jarang digunakan. Perintah query ALTER DATABASE adalah sebagai berikut:

```
ALTER DATABASE NamaDatabase
DEFAULT CHARACTER SET JenisKarakter
COLLATE DetailJenisKarakter
```

Contoh:

```
ALTER DATABASE `latihan1`
DEFAULT CHARACTER SET latin1
COLLATE latin1_bin
```

Beberapa contoh CHARACTER SET yang dapat digunakan seperti latin1, latin2, ascii, armscii8, binary, dan sebagainya. Sedangkan contoh COLLATE seperti latin1_bin, latin1_swedish_ci, latin2_general_ci, dan sebagainya.

2.3.2. ALTER TABLE

Fungsi ALTER TABLE adalah untuk mengubah struktur tabel, baik dari field, tipe data, primary key, dan sebagainya. Berikut ini beberapa bentuk ALTER TABLE

Tabel 3 : Query ALter Table

No	Query Alter Table	Fungsi
	ALTER TABLE NamaTable ADD NamaField TipeData;	Untuk menambah Field pada sebuah tabel
2	ALTER TABLE NamaTable ADD INDEX NamaIndex (FieldIndex-1, ..., FieldIndex-n);	Untuk menambah Index pada sebuah tabel
3	ALTER TABLE NamaTable ADD UNIQUE NamaUnique (FieldIndex-1, ..., FieldIndex-n);	Untuk menambah Unique pada sebuah tabel
4	ALTER TABLE NamaTable ADD PRIMARY KEY (FieldIndex-1, ..., FieldIndex-n);	ALTER TABLE NamaTable ADD PRIMARY KEY (FieldIndex-1, ..., FieldIndex-n);
5	ALTER TABLE NamaTable ADD CONSTRAINT NamaForeignKey FOREIGN KEY (FieldForeigKey) REFERENCES TabelReferensi(FieldPrimaryKey);	Untuk menambah foreign key pada sebuah tabel
6	ALTER TABLE NamaTable ALTER NamaField SET DEFAULT Nilai;	Untuk men-set nilai default pada sebuah field
7	ALTER TABLE NamaTable ALTER NamaField DROP DEFAULT;	Untuk menghapus nilai default pada sebuah field

No	Query Alter Table	Fungsi
8	ALTER TABLE NamaTable DROP COLUMN NamaField;	ALTER TABLE NamaTable DROP COLUMN NamaField;
9	ALTER TABLE NamaTable DROP PRIMARY KEY;	ALTER TABLE NamaTable DROP PRIMARY KEY;
10	ALTER TABLE NamaTable DROP PRIMARY KEY;	Untuk menghapus foreign key pada sebuah tabel
11	ALTER TABLE NamaTable DROP INDEX NamaIndex;	Untuk menghapus index pada sebuah tabel
12	ALTER TABLE NamaTable MODIFY COLUMN NamaField TipeData;	Untuk mengubah tipe data pada sebuah field
13	ALTER TABLE NamaTable CHANGE COLUMN NamaFieldLama NamaFieldBaru TipeData;	ALTER TABLE NamaTable CHANGE COLUMN NamaFieldLama NamaFieldBaru TipeData;

Contoh :

NO	CONTOH QUERY ALTER TABLE
1	ALTER TABLE NamaTable CHANGE COLUMN NamaFieldLama NamaFieldBaru TipeData;
2	ALTER TABLE Mahasiswa ADD INDEX IdxNama (NamaMhs);
3	ALTER TABLE Mahasiswa ADD UNIQUE UnikSIM (NomorSIM);
4	ALTER TABLE Mahasiswa ADD PRIMARY KEY (Nim);
5	ALTER TABLE krs ADD CONSTRAINT fkNIM FOREIGN KEY (mhs) REFERENCES Mahasiswa(Nim);
6	ALTER TABLE Mahasiswa ALTER JenisKelamin SET DEFAULT ‘L’;
7	ALTER TABLE Mahasiswa ALTER JenisKelamin DROP DEFAULT;
8	ALTER TABLE Mahasiswa DROP COLUMN JenisKelamin;
9	ALTER TABLE Mahasiswa DROP PRIMARY KEY;
10	ALTER TABLE krs DROP FOREIGN KEY fkNIM;
11	ALTER TABLE Mahasiswa DROP INDEX NamaMhs;
12	ALTER TABLE Mahasiswa MODIFY COLUMN NamaMhs VARCHAR(100);
13	ALTER TABLE Mahasiswa CHANGE COLUMN JenisKelamin JK CHAR(1);

2.3.3. ALTER VIEW

ALTER VIEW digunakan untuk mengubah sebuah view. Perintah query-nya adalah sebagai berikut:

```
ALTER VIEW NamaView AS QuerySelectYangDinginkan
```

Contoh:

```
ALTER VIEW MhsPria AS
SELECT Nim,Nama FROM Mahasiswa;
```

2.3.4. RENAME TABLE

RENAME TABLE digunakan untuk mengubah nama tabel pada sebuah database. Perintah query-nya adalah sebagai berikut:

```
ALTER TABLE NamaTable RENAME NamaTabelBaru;
```

Contoh:

```
ALTER TABLE Mahasiswa RENAME Mhs;
```

2.4. SQL DROP

SQL Drop digunakan untuk menghapus objek-objek fisik dalam database seperti untuk menghapus database, table, view, procedure, function, dan trigger.

2.4.1. DROP DATABASE

DROP DATABASE digunakan untuk menghapus database. Jika menggunakan perintah tersebut harus berhati-hati, karena apabila sudah terhapus, database tidak dapat dikembalikan lagi. Perintah query-nya adalah sebagai berikut:

```
DROP DATABASE NamaDatabase;
```

Contoh:

```
DROP DATABASE LatihanDB;
```

2.4.2. DROP TABLE

DROP TABLE digunakan untuk menghapus tabel dalam sebuah database. Jika menggunakan perintah tersebut juga harus berhati-hati, karena tabel telah dihapus tidak dapat dikembalikan lagi. Perintah query-nya adalah berikut:

```
USE NamaDatabase;
DROP TABLE NamaTabel;
```

Contoh:

```
USE LatihanDB;
DROP TABLE Mhs;
```

2.4.3. DROP VIEW

DROP VIEW digunakan untuk menghapus tabel virtual (view) dalam sebuah database. Perintah query-nya adalah sebagai berikut:

```
USE NamaDatabase;
DROP View NamaView;
```

Contoh:

```
USE LatihanDB;
DROP VIEW MhsPria;
```

3. DML
(DATA MANIPULATION LANGUAGE)

3.1. TUJUAN

Pada sesi ini mahasiswa diharapkan dapat memahami Data Manipulation Language (DML) lebih dalam. DML merupakan kumpulan perintah SQL yang berhubungan dengan manipulasi data di dalam tabel. DML tidak terkait dengan perubahan struktur dan definisi tipe data dari objek database. Query-query DML terdapat beberapa kategori seperti INSERT, SELECT, UPDATE, DELETE, TRUNCATE, CALL, DO, dan HANDLER. Tetapi pembahasan pada bab ini hanya membahas INSERT, SELECT, UPDATE, DELETE, dan TRUNCATE. Sedangkan CALL, DO, dan HANDLER akan dibahan pada sesi advance dan selanjutnya.

3.2. SQL INSERT

SQL Insert digunakan untuk memasukkan data kedalam tabel. SQL Insert ada beberapa bentuk yang dapat digunakan seperti tabel berikut ini:

Tabel 4 : Variant Query Insert

NO	QUERY
1	INSERT INTO NamaTabel VALUES(NilaiField-1, ..., NilaiField-n);
2	INSERT INTO NamaTabel(NamaField-1, ..., NamaField-n) VALUES(NilaiField-1, ..., NilaiField-n);
3	INSERT INTO NamaTabel1 SELECT FieldTabel2-1, ..., FieldTabel2-n FROM NamaTabel2 [WHERE kondisi];
4	INSERT INTO NamaTabel1(FieldTabel1-1, ..., FieldTabel1-n) SELECT FieldTabel2-1, ..., FieldTabel2-n FROM NamaTabel2 [WHERE kondisi];

Query INSERT nomor 1 dan 2 merupakan query INSERT yang umum digunakan. Sedangkan, Query INSERT nomor 3 dan 4 biasanya digunakan ketika kita ingin menyalin (copy) dari tabel lain yang sudah ada. Tetapi perlu diingat bahwa jumlah field tabel 1 harus sama dengan jumlah field yang di SELECT dari tabel 2, serta tipe data dan ukurannya juga dianjurkan sama

Contoh 1:

Buat tabel Mahasiswa dengan field Nim, Nama dan JenisKelamin, kemudian masukkan data dengan perintah seperti berikut ini:

```
INSERT INTO Mahasiswa VALUES('11451022521', 'Budi Alamsyah', 'Laki-laki');
```

Atau:

```
INSERT INTO Mahasiswa(Nim, Nama, JenisKelamin) VALUES('11451022521', 'Aisyah', 'Perempuan');
```

Contoh 2:

Buat tabel MahasiswaLaki dengan field Nim dan Nama, kemudian masukkan data dari tabel lain yaitu tabel Mahasiswa dengan cara:

```
INSERT INTO MahasiswaLaki SELECT Nim, Nama FROM Mahasiswa WHERE
JenisKelamin='Laki-laki';
```

Contoh 3:
Buat tabel MahasiswaPerempuan dengan field Nim dan Nama, kemudian masukkan data dari tabel lain yaitu tabel Mahasiswa dengan cara

```
INSERT INTO MahasiswaPerempuan(Nim, Nama) SELECT Nim, Nama FROM
Mahasiswa WHERE JenisKelamin='Perempuan';
```

3.3. SQL SELECT

SQL Select digunakan untuk menampilkan/mengambil data dari tabel tertentu. SQL Select sangat banyak bentuknya. Perintah-perintah query SELECT dapat dilihat pada tabel berikut.

NO	QUERY	KETERANGAN
1	SELECT * FROM NamaTabel [WHERE Kondisi] [ORDER BY NamaField ASC|DESC];	Menampilkan seluruh field, jika memiliki kondisi tambahkan klausa WHERE, ORDER BY digunakan untuk mengurutkan data
2	SELECT NamaField-1,...,NamaField-n FROM NamaTabel [WHERE Kondisi] [ORDER BY NamaField ASC|DESC];	Menampilkan field yang dipilih
3	SELECT NamaField-1 AS NamaFieldAlias FROM NamaTabel;	Memberikan Nama alias pada field yang ditampilkan
4	SELECT * FROM NamaTabel AS NamaTabelAlias	Memberikan nama alias pada nama tabel
5	SELECT * FROM NamaTabel LIMIT NilaiBatas [OFFSET RecordAwal];	Untuk membatasi banyak data yang ditampilkan
6	SELECT * FROM NamaTabel WHERE NamaField LIKE Pola;	Untuk mencocokkan berdasarkan pola karakter. Contoh pola: '%...', '...%', '%...%', '_...'
7	SELECT DISTINCT(NamaField) FROM NamaTabel [WHERE Kondisi];	Record yang memiliki kesamaan nilai ditampilkan sekali
8	Record yang memiliki kesamaan nilai ditampilkan sekali	Equivalen dengan OR
9	SELECT * FROM NamaTabel WHERE NamaField BETWEEN Nilai-1 AND Nilai-2	Untuk menampilkan data berdasarkan rentang nilai
10	SELECT NamaField-1, FungsiAgregasi(NamaField) FROM NamaTabel WHERE NamaField Operator Nilai GROUP BY NamaField;	Untuk mengelompokkan data
11	SELECT NamaField-1, FungsiAgregasi(NamaField) FROM NamaTabel WHERE NamaField Operator Nilai GROUP BY NamaField HAVING FungsiAgregasi(NamaField) Operator Nilai;	Untuk mengelompokkan data berdasarkan syarat tertentu

NO	QUERY	KETERANGAN
12	SELECT NamaField-1,...,NamaField-n FROM NamaTabel1 INNER JOIN NamaTabel2 ON NamaTabel1.NamaField = NamaTabel2.NamaField;	Menampilkan data dari beberapa tabel menggunakan INNER JOIN
13	SELECT NamaField-1,...,NamaField-n FROM NamaTabel1 LEFT JOIN NamaTabel2 ON NamaTabel1.NamaField = NamaTabel2.NamaField;	Menampilkan data dari beberapa tabel menggunakan LEFT JOIN
14	SELECT NamaField-1,...,NamaField-n FROM NamaTabel1 RIGHT JOIN NamaTabel2 ON NamaTabel1.NamaField = NamaTabel2.NamaField;	Menampilkan data dari beberapa tabel menggunakan RIGHT JOIN
15	SELECT NamaField-1, ..., NamaField-n FROM NamaTable1 UNION SELECT NamaField-1, ..., NamaField-n FROM NamaTable2;	Untuk menggabungkan data dari beberapa tabel yang berbeda
16	SELECT AVG(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan rata-rata nilai
17	SELECT COUNT(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan banyak data
18	SELECT MAX(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan nilai maksimu
19	SELECT MIN(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan nilai minimum
20	SELECT SUM(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan jumlah/total data
21	SELECT UCASE(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan dalam huruf kapital
22	SELECT LCASE(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan dalam huruf kecil
23	SELECT MID(NamaField, Mulai, Panjang) AS NamaAlias FROM NamaTable;	Fungsi menampilkan sub karakter
24	SELECT LENGTH(NamaField) AS NamaAlias FROM NamaTable;	Fungsi menampilkan panjang karakter
25	SELECT ROUND(NamaField, Desimal) AS NamaAlias FROM NamaTable;	Fungsi pembulatan
26	SELECT NOW()	Fungsi menampilkan tanggal dan waktu saat ini

Contoh:

NO	QUERY	KETERANGAN
1	SELECT * FROM Mahasiswa;	Menampilkan seluruh data Mahasiswa
2	SELECT * FROM Mahasiswa WHERE JenisKelamin='Perempuan';	Menampilkan seluruh data Mahasiswa perempuan

3	SELECT * FROM Mahasiswa ORDER BY Nama DESC;	Mengurutkan data Mahasiswa dari Z – A (Descending)
4	SELECT * FROM Mahasiswa LIMIT 5 OFFSET 3;	Menampilkan Mahasiswa sebanyak 5 data dimulai dari record ke 3
5	SELECT * FROM Mahasiswa WHERE Nama LIKE '%di';	Menampilkan Mahasiswa yang namanya berakhiran 'di'
6	SELECT * FROM Mahasiswa WHERE Nama LIKE 'B%';	Menampilkan Mahasiswa yang namanya berawalan 'B'
7	SELECT * FROM Mahasiswa WHERE Nama LIKE '%ud%';	Menampilkan Mahasiswa yang namanya mengandung karakter 'ud'
8	SELECT * FROM Mahasiswa WHERE Nama LIKE '__s_';	Menampilkan Mahasiswa yang karakter ketiga dari namanya mengandung huruf 's'
9	9 SELECT DISTINCT(Nama) FROM Mahasiswa;	Menampilkan data nama Mahasiswa hanya satu record walaupun ada nama yang sama lebih dari satu record
10	SELECT * FROM Mahasiswa WHERE Nim IN ('114510025521', '114510025524', '114510025530');	SELECT * FROM Mahasiswa WHERE Nim IN ('114510025521', '114510025524', '114510025530');
11	SELECT * FROM Mahasiswa WHERE Tinggi BETWEEN 150 AND 170;	Menampilkan mahasiswa yang tingginya antara 150 sampai 170
12	SELECT SUM(harga) FROM Penjualan GROUP BY JenisBarang;	Menampilkan jumlah harga pada tabel Penjualan yang dikelompokkan berdasarkan JenisBarang
13	SELECT SUM(harga) FROM Penjualan GROUP BY JenisBarang HAVING MIN(harga) = '10000';	Menampilkan jumlah harga pada tabel Penjualan yang dikelompokkan berdasarkan JenisBarang dengan syarat harga yang dijumlahkan minimal berharga 10.000
14	SELECT M.Nim, M.Nama, A.NamaAgama FROM Mahasiswa AS M INNER JOIN Agama AS A ON M.IdAgama = A.IdAgama;	Menampilkan data dari dua tabel yaitu Mahasiswa dan Agama menggunakan INNER JOIN
15	SELECT ROUND(IPK, 2) AS ip FROM Mahasiswa;	Menampilkan IPK dengan pembulatan dua angka dibelakang koma
16	SELECT NOW();	Menampilkan tanggal dan waktu ketika dieksekusi

3.4. SQL UPDATE

SQL Update digunakan untuk memperbarui data. Perintah query-nya adalah sebagai berikut:

```
UPDATE NamaTabel SET NamaField-1=NilaiBaru, ..., NamaField-n=NilaiBaru
WHERE Kondisi;
```

Contoh:

```
UPDATE Mahasiswa SET Nama="Budiman", JenisKelamin="L" WHERE
NIM="11551025564";
```

3.5. SQL DELETE

SQL Delete digunakan untuk menghapus data. Perintah query-nya adalah sebagai berikut:

```
DELETE FROM NamaTabel WHERE Kondisi;
atau
DELETE * FROM NamaTabel WHERE Kondisi;
```

Contoh:

```
DELETE FROM Mahasiswa WHERE NIM="11551025564";
```

3.6. SQL TRUNCATE

SQL Truncate digunakan untuk menghapus semua data dalam satu tabel. Perintah ini sama dengan SQL Delete untuk semua data. Perintah query-nya seperti berikut ini:

```
TRUNCATE TABLE NamaTabel
```

Contoh:

```
TRUNCATE TABLE Mahasiswa;
```

Contoh tersebut digunakan untuk menghapus seluruh data yang terdapat pada tabel Mahasiswa.

4. STORE PROCEDUR

4.1. TUJUAN

Pada sesi ini anda diharapkan dapat memahami penggunaan stored procedure. Stored Procedure merupakan salah satu objek routine dalam database untuk menyimpan berbagai perintah yang sering digunakan.

4.2. MEMBUAT STORE PROCEDURE

Untuk membuat stored procedure dapat menggunakan perintah DDL yaitu CREATE PROCEDURE. Procedure ada beberapa bentuk, ada yang menggunakan parameter input dan parameter output atau kombinasinya, dan juga ada yang tidak menggunakan parameter sama sekali.

4.3. PROCEDUR TANPA PARAMETER

Perintah query untuk membuat procedure tanpa parameter adalah sebagai berikut:

```
DELIMITER $$
CREATE PROCEDURE NamaProsedur()
BEGIN
...Perintah-perintah...
END $$
DELIMITER;
```

Delimiter “$$” digunakan sebagai pembatas antar procedure. Delimiter “$$” dapat diganti dengan simbol lain. Delimiter “;” digunakan untuk mengembalikan delimiter default. Cara memanggil procedure menggunakan perintah:

```
CALL NamaProsedur();
```

Contoh:

```
DELIMITER $$
CREATE PROCEDURE HapusSemuaData()
BEGIN
TRUNCATE TABLE Nilai;
TRUNCATE TABLE Mahasiswa;
END$$
DELIMITER;
```

Cara memanggil prosedur HapusSemuaData adalah:

```
CALL HapusSemuaData();
```

4.3.1. DEKLARASI VARIABEL

Variabel dideklarsikan didalam store procedure ataupun store function. Cara penulisan variabel adalah sebagai berikut:

```
DECLARE namaVariabel TipeData;
```

Contoh:

```
DECLARE Nama VARCHAR(30);
```

4.3.2. PERCABANGAN

Percabangan digunakan untuk membuat perintah yang membutuhkan kondisi tertentu. Percabangan di MySQL memiliki beberapa bentuk yaitu menggunakan IF dan CASE.

Perintah IF dengan satu kondisi:

```
IF condition THEN
statement_list
[statement_list]...
END IF;
```

Perintah IF dengan kondisi lebih dari satu:

```
IF condition THEN
[statement_list]
[ELSEIF condition THEN]
[statement_list]
[ELSE]
[statement_list]
END IF;
```

Perintah CASE:

```
CASE case_value
WHEN when_value THEN statement_list
[WHEN when_value THEN statement_list] ...
[ELSE statement_list]
END CASE;
```

Contoh Penggunaan Perintah IF:

```
DELIMITER $$
CREATE PROCEDURE JK()
    BEGIN
        DECLARE nilai VARCHAR(1) DEFAULT 'L';
        IF nilai = 'L' THEN
            SELECT 'LAKI-LAKI';
        ELSE IF nilai = 'P' THEN
            SELECT 'PEREMPUAN';
        ELSE
            BEGIN
        SELECT 'TIDAK TERDEFINISI';
        END;
    END IF;
 END$$
```

Contoh Penggunaan Perintah CASE:

```
DELIMITER $$
CREATE PROCEDURE JK()
    BEGIN
        DECLARE nilai VARCHAR(1) DEFAULT ‘L’;
            CASE nilai
                WHEN ‘L’ THEN SELECT ‘LAKI-LAKI’;
                WHEN ‘P’ THEN SELECT ‘PERENPUAN’;
        ELSE
        BEGIN
            SELECT ‘TIDAK TERDEFINISI’;
        END;
    END CASE;
 END$$
```

4.3.3. PERULANGAN

Perulangan digunakan untuk membuat perintah yang membutuhkan iterasi tertentu. Perulangan di MySQL memiliki beberapa bentuk yaitu menggunakan LOOP, WHILE, dan REPEAT.

Perintah LOOP

```
[begin_label:] LOOP
statement_list
END LOOP [end_label];
```

Perintah WHILE

```
[begin_label:] WHILE search_condition DO
statement_list
END WHILE [end_label];
```

Perintah REPEAT

```
[begin_label:] REPEAT
statement_list
UNTIL search_condition
END REPEAT [end_label];
```

4.4. PROCEDUR MENGGUNAKAN PARAMETER

Sama halnya dengan membuat procedure tanpa parameter, perintah yang digunakan juga sama. Perbedaannya, procedure nya menggunakan paramater input atau output. Parameter input ditandai dengan perintah IN parameternya dan perintah output ditandai dengan perintah OUT pada parameternya.

Perintah query untuk membuat procedure menggunakan parameter IN adalah sebagai berikut:

```
DELIMITER $$
CREATE PROCEDURE NamaProsedur(IN NamaParam1 TypeData, ...)
BEGIN
```

```
...Perintah-perintah...
END $$
DELIMITER ;
```

Contoh:

```
DELIMITER $$
CREATE PROCEDURE HapusMahasiswa(IN paramNim VARCHAR(11))
BEGIN
DELETE FROM Mahasiswa WHERE Nim = paramNim;
END $$
DELIMITER;
```

Cara pemanggilannya adalah sebagai berikut:

```
CALL HapusMahasiswa('11351025531');
```

Perintah query untuk membuat procedure menggunakan parameter OUT adalah sebagai berikut:

```
DELIMITER $$
CREATE PROCEDURE NamaProsedur(OUT NamaParam1 TypeData, ...)
BEGIN
...Perintah-perintah...
END $$
DELIMITER;
```

Parameter OUT menghasilkan satu nilai, sehingga parameter out tidak digunakan untuk menyimpan hasil proses yang lebih dari satu nilai.

Cara pemanggilannya adalah sebagai berikut:

```
CALL NamaProsedur(@NamaParam, ...)
SELECT @NamaParam;
```

Contoh:

```
DELIMITER $$
CREATE PROCEDURE BanyakMahasiswa(OUT Banyak INT)
BEGIN
DECLARE hitungMhs INT;
SELECT COUNT(Nim) INTO hitungMhs
FROM Mahasiswa
WHERE JenisKelamin = 'L';
SET Banyak = hitungMhs;
END $$
DELIMITER;
```

Cara Pemanggilannya adalah sebagai berikut:

```
CALL BanyakMahasiswa(@MhsLaki);
SELECT @MhsLaki;
```

4.5. MENGUBAH STORE PROCEDUR

Stored procedure dapat diubah menggunakan perintah DROP dan CREATE. Perintah query nya seperti berikut:

```
DELIMITER $$
    DROP PROCEDURE IF EXISTS NamaProsedur$$
    CREATE PROCEDURE NamaProsedur()
    BEGIN
    ...Perintah-perintah baru...
    END $$
DELIMITER;
```

Contoh:

```
DELIMITER $$
    DROP PROCEDURE IF EXISTS BanyakMahasiswa$$
    CREATE PROCEDURE BanyakMahasiswa(OUT Banyak INT)
    BEGIN
    DECLARE hitungMhs INT;
    SELECT COUNT(Nim) INTO hitungMhs
    FROM Mahasiswa
    WHERE JenisKelamin = 'P';
    SET Banyak = hitungMhs;
    END $$
DELIMITER;
```

4.6. MENGHAPUS STORE PROCEDUR

Untuk menghapus stored procedure dapat menggunakan perintah DROP. Perintah query nya adalah sebagai berikut:

```
DROP PROCEDURE IF EXISTS NamaProsedur;
```

Contoh:

```
DROP PROCEDURE IF EXISTS BanyakMahasiswa;
```

4.7. MENAMPILKAN DAFTAR PROCEDUR

Untuk menampilkan daftar procedure dapat menggunakan perintah berikut:

```
SHOW PROCEDURE STATUS WHERE Db="NamaDatabase";
```

Contoh:

```
SHOW PROCEDURE STATUS WHERE Db="simak";
```

5. ADVANCED

5.1. TUJUAN

Pada bab ini mahasiswa diharapkan dapat memahami penggunaan stored Function, Trigger dan Event.

5.2. STORED FUNCTION

Stored Stored Function hampir sama dengan Stored Procedure, bedanya stored function memiliki nilai yang dikembalikan (return). Untuk membuat stored function dapat menggunakan perintah DDL yaitu CREATE FUNCTION. Function ada beberapa bentuk, ada yang menggunakan parameter input dan ada yang tidak menggunakan.

5.2.1. MEMBUAT STORED FUNCTION

Perintah query untuk membuat function tanpa parameter adalah sebagai berikut:

```
DELIMITER $$
    CREATE FUNCTION NamaFungsi() RETURNS TypeData
    BEGIN
    ...Perintah-perintah...
    RETURN NilaiAkhir;
    END $$
DELIMITER;
```

Contoh:

```
DELIMITER $$
    CREATE FUNCTION BanyakMahasiswaLaki() RETURNS INT
    BEGIN
    DECLARE hitungMhs INT;
    SELECT COUNT(Nim) INTO hitungMhs
    FROM Mahasiswa
    WHERE JenisKelamin = "L";
    RETURN hitungMhs;
    END $$
DELIMITER;
```

Cara pemanggilannya adalah sebagai berikut:

```
SELECT BanyakMahasiswaLaki;
```

Perintah query untuk membuat function yang menggunakan parameter adala sebagai berikut:

```
DELIMITER $$
    CREATE FUNCTION NamaFungsi(NamaParam TypeData, ...)
    RETURNS TypeData
    BEGIN
    ...Perintah-perintah...
    RETURN NilaiAkhir;
    END $$
DELIMITER;
```

Contoh:

```
DELIMITER $$
    CREATE FUNCTION BanyakMhs(jk CHAR(1)) RETURNS INT
    BEGIN
    DECLARE hitungMhs INT;
    SELECT COUNT(Nim) INTO hitungMhs
    FROM Mahasiswa
    WHERE JenisKelamin = jk;
    RETURN hitungMhs;
    END $$
DELIMITER;
```

Cara pemanggilannya adalah sebagai berikut:

```
SELECT BanyakMhs("L");
```

Contoh 2:

```
DELIMITER $$
    CREATE FUNCTION LuasLingkaran(jari FLOAT) RETURNS FLOAT
    BEGIN
    DECLARE Luas FLOAT;
    SET Luas = 3.14*jari*jari;
    RETURN Luas;
    END $$
DELIMITER;
```

Cara pemanggilannya adalah sebagai berikut:

```
SELECT LuasLingkaran(7);
```

5.2.2. MENGUBAH STORED FUNCTION

Stored function dapat diubah menggunakan perintah DROP dan CREATE. Perintah query nya seperti berikut:

```
DELIMITER $$
    DROP FUNCTION IF EXISTS NamaFungsi$$
    CREATE FUNCTION NamaFungsi() RETURNS TypeData
    BEGIN
    ...Perintah-perintah baru...
    RETURN NilaiAkhir;
    END $$
DELIMITER;
```

Contoh:

```
DELIMITER $$
    DROP FUNCTION IF EXISTS LuasLingkaran$$
    CREATE FUNCTION LuasLingkaran(r FLOAT) RETURNS FLOAT
    BEGIN
    DECLARE L FLOAT;
```

```
    SET L = 3.14*r*r;
    RETURN L;
    END $$
DELIMITER;
```

5.2.3. MENGHAPUS STORED FUNCTION

Untuk menghapus stored function dapat menggunakan perintah DROP. Perintah query nya adalah sebagai berikut:

```
DROP FUNCTION IF EXISTS NamaFungsi;
```

Contoh:

```
DROP FUNCTION IF EXISTS LuasLingkaran;
```

5.3. TRIGER

Trigger merupakan sekumpulan perintah atau sintaks SQL yang akan secara otomatis dijalankan apabila ada perintah INSERT, DELETE, atau UPDATE yang dijalankan didalam tabel. Contohnya adalah dalam sistem penjualan, jika dientri barang baru maka stock akan bertambah secara otomatis.

5.3.1. MEMBUAT TRIGER

Perintah query untuk membuat trigger adalah sebagai berikut:

```
DELIMITER $$
    CREATE TRIGGER NamaTrigger
    [BEFORE|AFTER] [INSERT|UPDATE|DELETE] ON NamaTabel
    FOR EACH ROW
    BEGIN
    ...perintah-perintah...
    END$$
DELIMITER;
```

BEFORE digunakan jika trigger ingin diaktifkan sebelum dihubungkan dengan suatu operasi. AFTER digunakan jika trigger ingin diaktifkan setelah dihubungkan dengan suatu operasi.

Di dalam statement trigger, kita dapat mengakses record tabel sebelum atau sesudah proses dengan menggunakan NEW dan OLD. NEW digunakan untuk mengambil record yang akan diproses (INSERT atau UPDATE), sedangkan OLD digunakan untuk mengakses record yang sudah diproses (UPDATE atau DELETE).

Tabel 5 : Event Triger

NO	EVENT	NEW	OLD
1	BEFORE INSERT	Semua field yang akan di insert	Tidak support “OLD”
2	AFTER INSERT	Jika menggunakan AFTER tidak bisa mengakses NEW	Tidak ada “OLD” field pada AFTER INSERT (setelah diinsert)
3	BEFORE UPDATE	Semua field yang akan diupdate	Tidak support “OLD”
4	BEFORE DELETE	Tidak ada baris baru saat Di delete	BEFORE tidak support delete
5	AFTER DELETE	AFTER tidak support New	Semua field yang sebelumnya yang telah di-delete

Contoh:

```
DELIMITER $$
    CREATE TRIGGER InsertLog BEFORE DELETE ON Mahasiswa
    FOR EACH ROW
    BEGIN
    INSERT INTO Log(Deskripsi, Waktu, Pengguna)
    VALUES( CONCAT('Menghapus NIM. ',OLD.Nim), Now(), User() );
    END$$
DELIMITER;
```

Trigger tidak ada cara pemanggilannya, tetapi trigger akan dijalankan atau dieksekusi ketika ada perintah INSERT, UPDATE atau DELETE. Pada contoh tersebut, trigger InsertLog akan dijalankan jika ada perintah DELETE pada tabel Mahasiswa.

5.3.2. MENGUBAH TRIGER

Trigger dapat diubah menggunakan perintah DROP dan CREATE. Perintah query nya seperti berikut:

```
DELIMITER $$
    DROP TRIGGER IF EXISTS NamaTrigger$$
    CREATE TRIGGER NamaTrigger
    [BEFORE|AFTER] [INSERT|UPDATE|DELETE] ON NamaTabel
    FOR EACH ROW
    BEGIN
    ...perintah-perintah baru...
    END$$
DELIMITER ;
```

Contoh:

```
DELIMITER $$
    DROP TRIGGER IF EXISTS InsertLog$$
```

```
    CREATE TRIGGER InsertLog BEFORE DELETE ON Mahasiswa
    FOR EACH ROW
    BEGIN
    INSERT INTO Log(Deskripsi, Waktu, Pengguna)
    VALUES(
    CONCAT('Menghapus Mahasiswa dengan NIM. ',OLD.Nim),
    Now(),
    User()
    );
    END$$
DELIMITER;
```

5.3.3. MENGHAPUS TRIGER

Untuk menghapus trigger dapat menggunakan perintah DROP. Perintah query nya adalah sebagai berikut:

```
DROP TRIGGER IF EXISTS NamaTrigger;
```

Contoh:

```
DROP TRIGGER IF EXISTS InsertLog;
```

5.4. EVENT

Event digunakan untuk mengelola penjadwalan untuk menjalankan perintah tertentu secara otomatis. Contohnya adalah setiap 6 bulan sekali data-data penjualan akan dipindahkan ketabel history secara otomatis.

5.4.1. MEMBUAT EVENT

Perintah query untuk membuat event adalah sebagai berikut:

```
CREATE EVENT NamaEvent
    ON SCHEDULE EVERY lama [SECOND|MINUTE|...]
    STARTS waktuMulaiEvent
    ENDS waktuAkhirEvent
    DO
    ...perintah-perintah...
```

Contoh:

```
CREATE EVENT event1
    ON SCHEDULE EVERY 5 SECOND
    STARTS '2017-02-21 00:00:00'
    ENDS '2017-02-21 22:00:00'
    DO
INSERT INTO tes(waktu) VALUES(NOW());
```

Maksud contoh tersebut adalah setiap lima detik dimulai dari '2017-02-21 00:00:00' sampai dengan '2017-02-21 00:00:00' akan dilakukan proses INSERT kedalam tabel 'tes'. Apabila waktu akhir event sudah sama dengan atau melebihi waktu sistem, maka event akan otomatis terhapus.

5.4.2. MENGAKTIFKAN EVENT

Perintah query untuk mengaktifkan event adalah sebagai berikut:

```
SET GLOBAL event_scheduler = ON;
```

Apabila event sudah dibuat, tetapi event_scheduler belum diaktifkan, maka event tidak akan dijalankan.

5.4.3. MENAMPILKAN EVENT

Perintah query untuk menampilkan daftar event yang telah dibuat sebagai berikut:

```
SHOW EVENTS;
```

5.4.4. MENGHAPUS EVENT

Perintah query untuk menghapus event yang telah dibuat adalah sebagai berikut:

```
DROP EVENT namaEventYangAkanDihapus;
```

5.4.5. MENGUBAH EVENT

Perintah query untuk mengubah event yang telah dibuat adalah berikut:

```
ALTER EVENT NamaEventYangAkanDiubah
    ON SCHEDULE EVERY lama [SECOND|MINUTE|...]
    STARTS waktuMulaiEvent
    ENDS waktuAkhirEvent
    DO
    ...perintah-perintah...
```

Contoh:

```
ALTER EVENT event1
    ON SCHEDULE EVERY 1 MINUTE
    STARTS '2017-02-21 00:00:00'
    ENDS '2017-02-22 06:00:00'
    DO
INSERT INTO tes(waktu) VALUES(NOW());
```

Maksud contoh tersebut adalah setiap satu menit dimulai dari ‘2017-02-21 00:00:00’ sampai dengan ‘2017-02-22 06:00:00’ akan dilakukan proses INSERT kedalam tabel ‘tes’.

6. DCL
(DATA CONTROL LANGUAGE)

6.1. TUJUAN

Pada bab ini mahasiswa diharapkan dapat memahami penggunaan query Data Control Language (DCL). DCL adalah jenis instruksi SQL yang berkaitan dengan manajemen hak akses dan pengguna (user).

6.2. MANAJEMEN USER

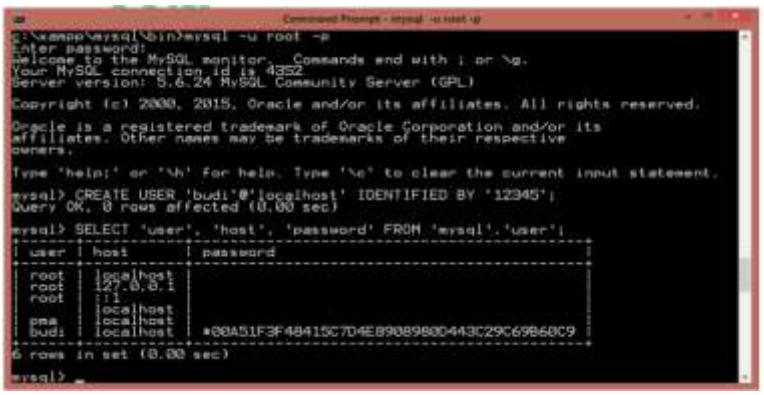
User adalah akun database yang digunakan untuk mengakses suatu database tertentu. Manajemen user biasanya digunakan untuk keamanan database agar hanya orang yang memiliki hak akses yang dapat mengakses database. “root” adalah user tertinggi dalam database.

6.2.1. MEMBUAT USER

User dapat dibuat dengan perintah query seperti berikut ini:

```
CREATE USER ‘NamaUser’@‘LokasiUser’ IDENTIFIED BY "Passwordnya";
```

“LokasiUser” adalah alamat IP atau domain dari database server. Contohnya, ‘localhost’, ‘127.0.0.1’, dan sebagainya.



Gambar 11 : 1 Membuat user menggunakan command prompt

6.2.2. MENGUBAH NAMA USER

Untuk mengubah nama user dapat menggunakan perintah query sebagai berikut:

```
RENAME USER NamaUserLama TO NamaUserBaru;
```

User yang bisa menjalankan perintah tersebut adalah “root” atau user yang diberikan hak akses (privileges) untuk mengelola user.
Contoh:

```
RENAME USER "budi"@"localhost" TO "anto"@"localhost";
```

6.2.3. MENGUBAH PASSWORD USER

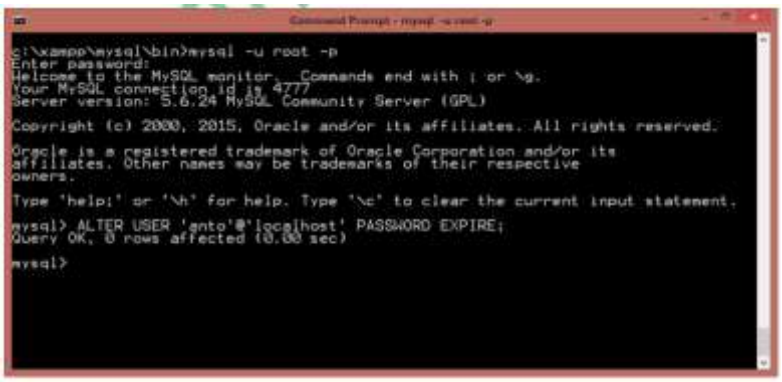
Untuk mengubah password user dapat menggunakan perintah query sebagai berikut:

```
ALTER USER 'NamaUser'@'LokasiUser' PASSWORD EXPIRE;
```

Lalu login lagi menggunakan NamaUser yang password-nya sudah diubah dan jalankan query:

```
SET PASSWORD = PASSWORD("password baru");
```

Contoh:
Login menggunakan root.



Gambar 12 : Login root untuk me-reset password menggunakan command prompt

6.2.4. MENGHAPUS USER

Untuk menghapus user dapat menggunakan perintah query seperti berikut:

```
DROP USER 'NamaUser'@'LokasiUser';
```

Contoh:

```
DROP USER "anto"@"localhost";
```

6.3. MANAJEMEN HAK AKSES

Hak akses digunakan untuk memberikan akses kepada pengguna database. Akses yang diberikan bermacam-macam tergantung hak akses yang diberikan kepada pengguna tersebut.

6.3.1. MEMBUAT HAK AKSES

Untuk memberikan hak akses kepada user yang diinginkan dapat menggunakan perintah query seperti berikut ini:

```
GRANT HakAkses ON objek TO 'NamaUser'@'LokasiUser';
```

- ✓ HakAkses adalah privileges yang akan berikan kepada user.
- ✓ HakAkses tersebut berisi query yang diperbolehkan, seperti: SELECT, INSERT, UPDATE, DELETE, atau query lainnya (EXECUTE, CREATE, ALTER, DROP, dsb).
- ✓ Objek maksudnya adalah objek dari sebuah database, dapat berupa tabel atau view.

- ✓ Jika ingin memberikan hak penuh untuk semua query dasar tersebut, HakAkses ini bisa diisi dengan ALL.
- ✓ User yang dapat memberikan hak akses adalah “root” atau user yang diberikan hak untuk memberikan hak akses kepada user lain.

Contoh:
Buat database simak dan tabel mahasiswa dengan field Nim, Nama, dan JenisKelamin. Lalu jalankan query berikut ini:

```
GRANT SELECT ON "simak"."mahasiswa" TO "anto"@"localhost";
```

- ✓ SELECT adalah privileges yang diberikan kepada use ‘anto’@’localhost’.
- ✓ ‘simak’ adalah nama database yang ingin diberikan hak akses. Jika ingin mengizinkan user dapat mengakses semua database yang ada maka nama database dapat ditulis dengan tanda bintang (*).
- ✓ ‘mahasiswa’ adalah nama tabel yang ingin diberikan hak akses. Jika ingin mengizinkan user dapat menggunakan semua tabel maka nama dapat ditulis dengan tanda bintang (*).

Untuk pengujian, coba login menggunakan user ‘anto’ lalu jalankan perintah selain query SELECT mahasiswa pada database simak misalnya INSERT, UPDATE atau yang lainnya.

Contoh lainnya:

```
GRANT ALL ON simak.mahasiswa TO 'anto'@'localhost';
GRANT SELECT (Nama, JenisKelamin) ON simak.mahasiswa TO 'anto'@'localhost';
GRANT INSERT ON simak.mahasiswa TO 'anto'@'localhost';
GRANT UPDATE ON simak.mahasiswa TO 'anto'@'localhost';
GRANT DELETE ON simak.mahasiswa TO 'anto'@'localhost';
GRANT DELETE ON simak.* TO 'anto'@'localhost';
GRANT SELECT, INSERT, DELETE ON simak.mahasiswa TO 'anto'@'localhost';
dll...
```

6.3.2. MENGHAPUS HAK AKSES

Untuk menghapus hak akses pada user tertentu dapat menggunakan perintah query seperi berikut ini:

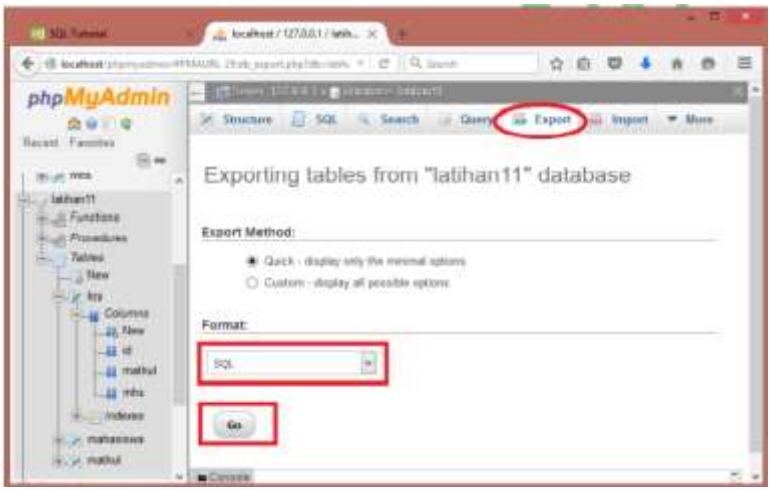
```
REVOKE HakAkses ON objek FROM 'NamaUser'@'LokasiUser';
```

Contoh:

```
REVOKE SELECT ON simak.mahasiswa FROM 'anto'@'localhost';
```

6.3.3. BACKUP DAN RESTORE

Cara yang digunakan untuk backup database cukup banyak. Jika menggunakan “phpmyadmin” caranya sangat mudah yaitu memilih database yang dibackup lalu pilih menu export lalu ikuti perintah-perintah selanjutnya



Gambar 13 : Backup database menggunakan phpmyadmin

Atau jika menggunakan command prompt, dapat menggunakan perintah seperti berikut ini

```
mysqldump -u [uname] -p[pass] [dbname] > [backupfile.sql]
```

Contoh:

```
C:\laragon\bin\mysql\mysql-5.7.33-winx64\bin>mysqldump –u root -p
simak > E:\file_backup.sql
```

Perintah tersebut untuk backup satu database. Jika kita ingin mem-backup lebih spesifik yaitu tabel tertentu saja, maka dapat menggunakan perintah seperti berikut ini:

```
C:\laragon\bin\mysql\mysql-5.7.33-winx64\bin>mysqldump simak -u
root –p mahasiswa > E:\file_
backup.sql
```

Untuk restore dapat menggunakan perintah seperti berikut ini:

```
mysql -u [uname] -p[pass] [db_to_restore] < [backupfile.sql]
```

Contoh:

```
C:\laragon\bin\mysql\mysql-5.7.33-winx64\bin>mysql -u root -p
simak_baru < E:\file_backup.sql
```

7. TCL
(TRANSACTION CONTROL LANGUAGE)

7.1. TUJUAN

Pada sesi ini anda diharapkan dapat memahami penggunaan beberapa query Transaction Control Language (TCL). TCL merupakan perintah SQL untuk proses transaksi. Proses transaksi digunakan untuk perintah yang lebih dari satu, namun harus berjalan semua, atau tidak sama sekali.

Misalnya untuk aplikasi critical seperti transfer uang dalam sistem database perbankan. Setidaknya akan ada 2 perintah, yaitu mengurangi uang nasabah A, dan menambah uang nasabah B. Namun jika terjadi kesalahan sistem, kedua ini harus dibatalkan, tidak bisa hanya satu perintah saja.

7.2. TRANSACTION AND COMMIT

Transaction adalah segala macam proses DML, DDL, maupun DCL yang terjadi dalam database. START TRANSACTION marupakan awal dari blok perintah untuk melakukan transakasi. COMMIT digunakan untuk mengakhiri transaksi yang sedang terjadi dengan merubah semua pending data menjadi permanen. Untuk membuat transaksi dapat menggunakan perintah query seperti berikut ini:

```
START TRANSACTION;
    Perintah-perintah query dan non query yang ingin di eksekusi....
COMMIT;
```

Contoh:

```
START TRANSACTION;
    UPDATE matakuliah SET SKS=1 WHERE idMK='TIF3301';
    UPDATE matakuliah SET SKS=3 WHERE idMK='TIF3301';
COMMIT;
```

Berdasarkan perintah diatas, jika belum ada perintah COMMIT maka eksekusi UPDATE masih dipending.

7.3. TRANSACTION COMMIT AND ROLLBACK

ROLLBACK digunakan untuk mengembalikan transaksi kebentuk semula sampai bertemu COMMIT terakhir kali. Berikut ini adalah penggunaan perintah ROLLBACK:

```
START TRANSACTION;
    Perintah-perintah query dan non query yang ingin di eksekusi....
    Jika Perintah-perintahnya tidak ada yang Error Maka:
    Jika ada yang error Maka:
ROLLBACK;
COMMIT;
```

Contoh:

```
START TRANSACTION;
```

```
    UPDATE matakuliah SET SKS=1 WHERE idMK='TIF3301';
    UPDATE matakuliah SET SKS=3 WHERE idMK='TIF3301';
ROLLBACK;
COMMIT;
```

Contoh tersebut akan membatalkan semua transaksi lalu dilakukan COMMIT. Sehingga sama saja tidak ada perintah yang dieksekusi.

7.4. ROLEBACK AND SAVE POINT

Savepoint merupakan penanda pada baris tertentu pada sebuah kumpulan query. Perintah yang digunakan untuk SAVEPOINT adalah sebagai berikut:

```
START TRANSACTION;
    Perintah-perintah query dan non query yang ingin di eksekusi 1
    SAVEPOINT penanda1;
    Perintah-perintah query dan non query yang ingin di eksekusi 2
    SAVEPOINT penanda2;
    Perintah-perintah query dan non query yang ingin di eksekusi-n
    ...
    ...
    Jika Perintah-perintahnya tidak ada yang Error Maka:
ROLLBACK TO SAVEPOINT penanda;
COMMIT;
```

Contoh:

```
START TRANSACTION;
    UPDATE matakuliah SET SKS=1 WHERE idMK='TIF3301';
    SAVEPOINT titikA;
    UPDATE matakuliah SET SKS=2 WHERE idMK='TIF3301';
    SAVEPOINT titikB;
    UPDATE matakuliah SET SKS=3 WHERE idMK='TIF3301';
    SAVEPOINT titikC;
    UPDATE matakuliah SET SKS=4 WHERE idMK='TIF3301';
ROLLBACK TO SAVEPOINT titikB;
COMMIT;
```

Berdasarkan contoh tersebut, jika belum ada perintah COMMIT maka eksekusi UPDATE masih di-pending. Ketika bertemu perintah ROLLBACK TO SAVEPOINT titikB, maka query terakhir yang dibaca adalah sebelum SAVEPOINT titikB. Sehingga jika dilakukan COMMIT, maka SKS akan berubah menjadi 2.

7.5. AUTO COMMIT

Pada dasarnya DBMS MySQL, konfigurasi AUTOCOMMIT nya sudah dalam keadaan ON, maksudnya perintah-perintah query yang dijalankan akan otomatis langsung dieksekusi atau tidak dipending terlebih dahulu. Apabila ingin membuat setiap transaksi harus dilakukan COMMIT terlebih dahulu, maka harus menonaktifkan AUTOCOMMIT. Cara yang digunakan untuk menonaktifkan AUTOCOMMIT pada DBMS MySQL yaitu:

```
SET AUTOCOMMIT = 0;
```

Untuk mengaktifkan kembali, perintahnya yaitu:

```
SET AUTOCOMMIT = 1;
```

Contoh, ketika AUTOCOMMIT sudah dinonaktifkan, jalankanlah perintah:

```
UPDATE matakuliah SET SKS=3 WHERE idMK='TIF3301';
```

- ✓ Selanjutnya coba cek isi data matakuliah dengan idMK 'TIF3301', apakah ada yang berubah atau tidak?
- ✓ Jawabannya adalah tidak ada perubahan, karena belum dilakukan perintah COMMIT
- ✓ Tetapi jika setelah perintah UPDATE tersebut dilakukan perintah COMMIT maka data matakuliah akan berubah.

7.6. HANDLER

Handler digunakan untuk menangani eksekusi query jika terjadi error atau warning. Handler digunakan dalam prosedur atau fungsi.

Handler exception (error)

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    ROLLBACK;
END;
```

Handler warning

```
DECLARE EXIT HANDLER FOR SQLWARNING
BEGIN
    ROLLBACK;
END;
```

Selain EXIT ada juga CONTINUE

Contoh 1:

```
DELIMITER $$
CREATE PROCEDURE ContohTransaksi1()
BEGIN
    DECLARE pesan VARCHAR(30);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SET pesan = 'DI ROLLBACK KARENA EXCEPTION';
        SELECT pesan;
    END;
```

```
    DECLARE EXIT HANDLER FOR SQLWARNING
    BEGIN
        ROLLBACK;
        SET pesan = 'DI ROLLBACK KARENA WARNING';
        SELECT pesan;
    END;

    START TRANSACTION;
        INSERT INTO mhs VALUES('11380', 'Miki', 'Pku', '1989-02-25');
        INSERT INTO matakuliah VALUES('TIF3310', 'Basis Data', 3);
        SET pesan = 'SEMUA QUERY BERHASIL DIEKSEKUSI';
        SELECT pesan;
    COMMIT;
END$$
DELIMITER;
```

Contoh2:

```
DELIMITER $$
CREATE FUNCTION input_mhs(pNim VARCHAR(11), pNama VARCHAR(30),
pTempat VARCHAR(30), pTgl DATE) RETURNS BOOLEAN
BEGIN
    DECLARE eksekusi BOOLEAN DEFAULT FALSE;
    DECLARE isError BOOLEAN DEFAULT FALSE;
    DECLARE isWarning BOOLEAN DEFAULT FALSE;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET isError := TRUE;
    DECLARE CONTINUE HANDLER FOR SQLWARNING SET isWarning := TRUE;
    INSERT INTO mhs VALUES(pNim, pNama, pTempat, pTgl);
    IF isError = FALSE AND isWarning = FALSE THEN
        SET eksekusi := TRUE;
    END IF;
    RETURN eksekusi;
END$$

CREATE FUNCTION input_matakuliah(pIdmk VARCHAR(11), pNamaMK
VARCHAR(30), pSKS INT) RETURNS BOOLEAN
BEGIN
    DECLARE eksekusi BOOLEAN DEFAULT FALSE;
    DECLARE isErr BOOLEAN DEFAULT FALSE;
    DECLARE isWarn BOOLEAN DEFAULT FALSE;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET isErr := TRUE;
    DECLARE CONTINUE HANDLER FOR SQLWARNING SET isWarn := TRUE;

    INSERT INTO matakuliah VALUES(pIdmk, pNamaMK, pSKS);
    IF isErr = FALSE AND isWarn = FALSE THEN
        SET eksekusi := TRUE;
```

```
    END IF;

    RETURN eksekusi;
END$$
DELIMITER ;
```

Contoh:

```
DELIMITER $$
    CREATE PROCEDURE ContohTransaksi2()
    BEGIN
        DECLARE isBerhsl1 BOOLEAN DEFAULT FALSE;
        DECLARE isBerhsl2 BOOLEAN DEFAULT FALSE;

        START TRANSACTION;

        SELECT 'STATE SEBELUM EKSEKUSI' AS pesan;
        SAVEPOINT sebelum_eksekusi;
        SET isBerhsl1 = input_mhs('11380', 'Miki', 'Pku', '1989-02-
25');
        SET isBerhsl2 = input_matakuliah('TIF3310', 'Basis Data', 3);
        SELECT isBerhsl1, isBerhsl2;

        IF isBerhsl1 = TRUE AND isBerhsl2 = TRUE THEN
            COMMIT;
            SELECT 'STATE EKSEKUSI BERHASIL DILAKUKAN' AS pesan;
        ELSE
            SELECT 'STATE EKSEKUSI GAGAL DILAKUKAN' AS pesan;
            ROLLBACK TO sebelum_eksekusi;
        END IF;
    END$$
DELIMITER;
```

8. PHP
(SIMPLE CRUD)

8.1. TUJUAN

CRUD adalah akronim untuk Create, Read, Update, dan Delete. Operasi CRUD adalah manipulasi data dasar untuk database. Dalam tutorial ini kita akan membuat aplikasi PHP sederhana untuk melakukan semua operasi ini pada tabel database MySQL di satu tempat.

Kita akan membuat sebuah aplikasi dengan tampilan seperti pada gambar di bawah dengan sebuah tabel yang menampilkan data dari database di sisi frontend. Terdapat beberapa tombol untuk mengubah data yang ada di database lewat antarmuka yang kita buat. Tabel pengguna yang kita buat akan berisi informasi pengguna seperti nama, nomor telepon, email, dll.

Saya asumsikan sekarang ini anda sudah bisa membuat database, table dan beberapa atribut lain nya untuk dapat mengikuti tutorial ini.

8.2. KONEKSI

Untuk memlulai membuat project PHP anda diharusnkan untuk membuat folder di dalam direktori www dilokasi anda menginstal aplikasi laragon. Jika menggunakan Xampp maka folder dibuat didalam direktori htdocs.

Selanjutnya, setelah selesai membuat folder nama projek anda, maka anda harus membuat file config.php dan mengisi code seperti pada contoh di bawah ini.

```php
<?php
    $databaseHost = 'localhost';
    $databaseName = 'crud_db';
    $databaseUsername = 'root';
    $databasePassword = '';
$mysqli = mysqli_connect($databaseHost, $databaseUsername,
$databasePassword, $databaseName);
```

File **config.php ** menyimpan informasi tentang database host, username dan password. Sebagian besar server lokal bekerja dengan detail yang diberikan. Anda dapat mengubahnya sesuai dengan detail host dan database Anda.

Untuk menghubungkan PHP dengan MySQL, kita menggunakan fungsi mysqli_connect() dengan alamat server sebagai parameter pertama, user database sebagai parameter kedua, password user sebagai parameter ketiga, dan nama database sebagai parameter keempat.

Silahkan isi password sesuai dengan yang dibuat di servernya. Jika tidak menggunakan passsword, maka isikan dengan String kosong saja.

8.3. VIEW

Tahap selanjutnya adalah membuat file index.php dan tulislah codenya seperti dibawah ini.

```
<?php
// Create database connection using config file
include_once("config.php");

// Fetch all users data from database
$result = mysqli_query($mysqli, "SELECT * FROM users ORDER BY id
DESC");
?>

<html>
<head>
    <title>Homepage</title>
</head>
<body>
<a href="add.php">Add New User</a><br/><br/>
    <table width='80%' border=1>
    <tr>
        <th>Name</th> <th>Mobile</th> <th>Email</th> <th>Update</th>
    </tr>
    <?php
    while($user_data = mysqli_fetch_array($result)) {
        echo "<tr>";
        echo "<td>".$user_data['name']."</td>";
        echo "<td>".$user_data['mobile']."</td>";
        echo "<td>".$user_data['email']."</td>";
        echo "<td><a href='edit.php?id=$user_data[id]'>Edit</a> | <a
href='delete.php?id=$user_data[id]'>Delete</a></td></tr>";
    }
    ?>
    </table>
</body>
</html>
```

File index.php merupakan file utama yang menyertakan file konfigurasi untuk koneksi database. Kemudian menampilkan semua daftar pengguna menggunakan MySQL Select Query. Pengguna yang akan ditampilkan di dalam daftar perlu menambahkan terlebih dahulu menggunakan tautan 'Tambahkan Pengguna Baru'.

8.4. ADD

Selanjutnya membuat file add.php

```
<html>
<head>
```

```
    <title>Add Users</title>
</head>
<body>
    <a href="index.php">Go to Home</a>
    <br/><br/>

    <form action="add.php" method="post" name="form1">
        <table width="25%" border="0">
            <tr>
                <td>Name</td>
                <td><input type="text" name="name"></td>
            </tr>
            <tr>
                <td>Email</td>
                <td><input type="text" name="email"></td>
            </tr>
            <tr>
                <td>Mobile</td>
                <td><input type="text" name="mobile"></td>
            </tr>
            <tr>
                <td></td>
                <td><input type="submit" name="Submit"
value="Add"></td>
            </tr>
        </table>
    </form>

    <?php

    // Check If form submitted, insert form data into users table.
    if(isset($_POST['Submit'])) {
        $name = $_POST['name'];
        $email = $_POST['email'];
        $mobile = $_POST['mobile'];

        // include database connection file
        include_once("config.php");

        // Insert user data into table
        $result = mysqli_query($mysqli, "INSERT INTO
users(name,email,mobile) VALUES('$name','$email','$mobile')");

        // Show message when user added
        echo "User added successfully. <a href='index.php'>View
Users</a>";
    }
    ?>
```

```
</body>
</html>
```

File add.php berfungsi untuk menambahkan pengguna baru. Formulir HTML digunakan untuk menerima masukan data pengguna. Setelah data pengguna diserahkan, MySQL INSERT Query digunakan untuk memasukkan data pengguna ke dalam database.

8.5. EDIT

Membuat file edit.php

```
<?php
// include database connection file
include_once("config.php");
// Check if form is submitted for user update, then redirect to
homepage after update
if(isset($_POST['update']))
{
    $id = $_POST['id'];

    $name=$_POST['name'];
    $mobile=$_POST['mobile'];
    $email=$_POST['email'];
    // update user data
    $result = mysqli_query($mysqli, "UPDATE users SET
name='$name',email='$email',mobile='$mobile' WHERE id=$id");
    // Redirect to homepage to display updated user in list
    header("Location: index.php");
}
?>
<?php
// Display selected user data based on id
// Getting id from url
$id = $_GET['id'];

// Fetech user data based on id
$result = mysqli_query($mysqli, "SELECT * FROM users WHERE id=$id");

while($user_data = mysqli_fetch_array($result))
{
    $name = $user_data['name'];
    $email = $user_data['email'];
    $mobile = $user_data['mobile'];
}
?>
<html>
<head>
    <title>Edit User Data</title>
```

```
</head>

<body>
    <a href="index.php">Home</a>
    <br/><br/>

    <form name="update_user" method="post" action="edit.php">
        <table border="0">
            <tr>
                <td>Name</td>
                <td><input type="text" name="name" value=<?php echo
$name;?>></td>
            </tr>
            <tr>
                <td>Email</td>
                <td><input type="text" name="email" value=<?php echo
$email;?>></td>
            </tr>
            <tr>
                <td>Mobile</td>
                <td><input type="text" name="mobile" value=<?php echo
$mobile;?>></td>
            </tr>
            <tr>
                <td><input type="hidden" name="id" value=<?php echo
$_GET['id'];?>></td>
                <td><input type="submit" name="update"
value="Update"></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Edit.php digunakan untuk mengedit / update data pengguna. Anda dapat mengubah data pengguna dan memperbaruinya. File ini akan mengarahkan pengguna kembali ke homepage, setelah update sukses.

8.6. DELETE

Membuat file delete.php

```
<?php
// include database connection file
include_once("config.php");

// Get id from URL to delete that user
$id = $_GET['id'];
```

```php
// Delete user row from table based on given id
$result = mysqli_query($mysqli, "DELETE FROM users WHERE id=$id");

// After delete redirect to Home, so that latest user list will be
displayed.
header("Location:index.php");
?>
```

REFERENSI

Derek J. Balling, Jeremy Zawodny. 2004. High Performance MySQL. O'Reilly Media Inc. United States of America.

MySQL Tutorial. 2015. MySQL 5.1 Reference Manual. [online] http://dev.mysql.com/doc/refman/5.1/en/index.html.

Nixon R. 2014. Learning PHP, MySQL, JavaScript, CSS & HTML5, 3rd Ed. O'Reilly Media Inc. United States of America.

Prasetyo D.D. 2003. Administrasi Database Server MySQL. PT. Elex Media Komputindo. Jakarta.

W3Schools. 2015. SQL Quick Reference From W3Schools. [online] http://www.w3schools.com/sql/sql_quickref.asp