
**JURNAL PRAKTIKUM
(LAB. ACTIVITY)
PEMROGRAMAN WEB LANJUT
SI087**

**Materi 5:
Database
Cookie dan Session**

Dosen:

**Lilis Dwi Farida, S.Kom., M.Eng
Irma Rofni Wulandari, S.Pd., M.Eng
Ika Nur Fajri, M.Kom
Aditya Rizki Yudiantika, S.T., M.Eng**

**S1 – SISTEM INFORMASI
UNIVERSITAS AMIKOM YOGYAKARTA
2020**

Pemrograman Web Lanjut

Pendahuluan

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Mengetahui konsep database
2. Mengetahui penggunaan MySQL
3. Memahami konsep DDL dan DML
4. Mampu menerapkan DDL dan DML

B. Peralatan

1. PC Desktop
2. Windows 10
3. XAMPP
4. Command Prompt

C. Teori

Basis data (*database*) dapat dikatakan sebagai sekumpulan data yang disimpan, saling berhubungan, dan diorganisasi secara bersama. Perangkat lunak yang dapat digunakan untuk mengolah basis data sudah cukup banyak, tetapi pada pembahasan kali ini akan menggunakan MySQL. MySQL dipilih karena cukup mudah dipelajari dan bersifat opensource, serta mendukung banyak platform sistem operasi. Selain itu, MySQL juga memiliki dukungan komunitas yang banyak.

1) SQL (*Structure Query Language*)

SQL adalah bahasa pemrograman yang digunakan untuk mengakses data dalam basis data relasional. SQL biasanya berupa perintah sederhana yang berisi instruksi manipulasi data. Perintah ini sering disebut dengan *query*.

SQL dikenalkan pertama kali dalam IBM pada tahun 1970 dan sebuah standar ISO dan ANSI ditetapkan untuk SQL. Standar ini tidak tergantung *software* yang digunakan. Beberapa perintah SQL yang digunakan adalah.

a) *Data Definition Language (DDL)*

Data Definition Language (DDL) adalah perintah SQL yang digunakan untuk mendefinisikan sebuah obyek di dalam *database* dan tabel. Beberapa perintah dasar dalam DDL antara lain **CREATE, ALTER, RENAME, DROP**.

b) *Data Modification Language (DML)*

Data Modification Language (DML) adalah perintah SQL yang digunakan untuk melakukan manipulasi terhadap obyek yang ada di dalam *database*. Perintah SQL yang termasuk di dalam DML antara lain **SELECT, INSERT, UPDATE, DELETE**.

c) *Data Control Language (DCL)*

Data Control Language (DCL) adalah perintah SQL yang berhubungan dengan manipulasi user dan hak akses(*priviledges*). Perintah SQL dalam DCL antara lain **GRANT, REVOKE**.

2) MySQL dengan *Command Prompt*

a) Masuk MySQL dengan *Command Prompt*

Untuk mengakses MySQL, kita dapat menggunakan *command prompt*. Akan tetapi, sebelum membuka MySQL pastikan terlebih dahulu servis MySQL sudah diaktifkan. Selanjutnya buka *command prompt* dan masuk ke dalam direktori tempat MySQL telah diinstall. Berikut langkah-langkahnya.

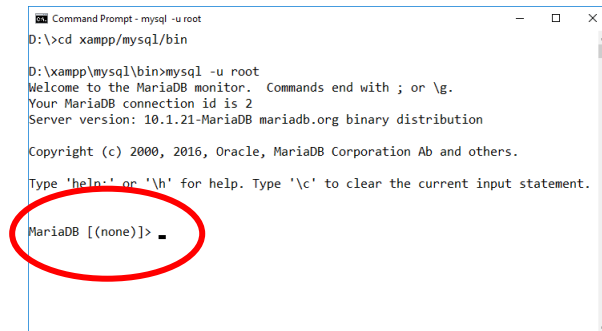
1. Ketikkan **cd xampp/mysql/bin** di tempat XAMPP telah telah diinstal. Pada contoh berikut, XAMPP diinstal di direktori D:\. Dan pastikan anda sudah berada di direktori D:\
2. Jika sudah masuk ke direktori MySQL, ketikkan perintah:

```
mysql -u "namauser" -p "password"
```

untuk dapat masuk ke MySQL. *Setting default* MySQL biasanya tanpa password.

Maka perintah masuk ke MySQL cukup sampai nama user saja. Contoh: **mysql -u root.**

Proses yang telah dilakukan di atas kurang lebih akan ditampilkan seperti gambar berikut.



```
Command Prompt - mysql -u root
D:\>cd xampp/mysql/bin

D:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.21-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> _
```

b) DDL Menggunakan *Command Prompt*

1. Membuat Database

Sintaks umum SQL dalam membuat sebuah *database* adalah sebagai berikut.

```
CREATE DATABASE [nama_database];
```

Aturan penamaan *database* sama dengan aturan penamaan pada variabel yang terdiri dari huruf, angka, dan *underscore*. Penamaan *database* tidak boleh sama dengan database yang sudah ada. Jika penamaan sama, maka akan dimunculkan pesan *error*.

2. Menampilkan Database

Sintaks umum SQL untuk menampilkan database yang telah dibuat adalah seperti berikut.

```
SHOW DATABASES;
```

3. Menghapus Database

Seorang *administrator* dapat menggunakan hak aksesnya untuk menghapus sebuah database yang memang sudah terpakai. Perintah umum untuk menghapus database adalah sebagai berikut.

```
DROP DATABASE [nama_database];
```

Perintah hapus *database* memiliki risiko yang tinggi. Karena sebuah *database* yang dihapus akan hilang beserta data yang ada di dalamnya. Sehingga seorang administrator harus sangat berhati-hati dalam menggunakan perintah **drop**. Perintah hapus *database* akan tidak berjalan ketika database yang dimaksud tidak ada.

4. Mengaktifkan Database

Database yang telah dibuat harus diaktifkan terlebih dahulu sebelum digunakan.

Sintaks umum untuk mengaktifkan *database* adalah sebagai berikut.

```
USE [nama_database];
```

Setelah mengaktifkan *database* yang akan digunakan, langkah selanjutnya adalah membuat tabel yang akan disimpan di dalam *database*.

5. Membuat Tabel

Bentuk umum SQL untuk membuat sebuah tabel adalah sebagai berikut.

```
CREATE TABLE [nama_tabel] (  
  Nama_field_1 tippedata_1 (lebardata_1),  
  Nama_field_2 tippedata_2 (lebardata_2),  
  ....  
  Nama_field_n tippedata_n (lebardata_n),  
  PRIMARY KEY (field_key)  
);
```

Aturan penamaan tabel dan kolom(field) sama dengan aturan penamaan database.

Contoh tabel **berita**:

No	Nama Kolom (<i>field</i>)	Tipe Data	Lebar Data
1	berita_id *	Int	5
2	berita_judul	Varchar	50
3	berita_tanggal	Date	
4	berita_isi	text	

```
CREATE TABLE berita(  
  berita_id int(5),  
  berita_judul varchar(200),  
  berita_tanggal date,  
  berita_isi);
```

Selanjutnya untuk melihat tabel **berita** sudah ada atau belum ketikkan perintah “**SHOW TABLES;**”. Sedangkan untuk melihat struktur tabel **berita** gunakan perintah “**DESC [nama_tabel];**” atau “**DESC berita;**”.

6. Mengubah Struktur Tabel dengan Alter

ALTER TABLE digunakan untuk melakukan perubahan pada tabel, antara lain mengubah nama tabel, struktur tabel, menambah field, dll. Berikut beberapa perintah yang dapat digunakan untuk mengubah tabel dengan menggunakan perintah ALTER.

a. Tambah field_baru

```
ALTER TABLE [nama_tabel] ADD [nama_kolom] [tipe_data] ([lebar_data]);
```

b. **Menambah primary key (kolom kunci)**

```
ALTER TABLE [nama_tabel] ADD PRIMARY KEY([nama_field]);
```

c. **Mengubah field**

```
ALTER TABLE [nama_tabel] CHANGE [nama_field_lama] [nama_field_baru]  
[tipe_data] ([lebar_data]);
```

d. **Menghapus nama field**

```
ALTER TABLE [nama_tabel] DROP [nama_field];
```

e. **RENAME nama tabel**

```
ALTER TABLE [nama_tabel] RENAME TO [nama_tabel_baru];
```

7. Menghapus Tabel

Untuk menghapus sebuah tabel dapat menggunakan perintah SQL sebagai berikut.

```
DROP TABLE [nama_tabel];
```

Perintah drop harus digunakan dengan sangat hati-hati karena terkait dengan ketersediaan data yang bisa jadi bersifat sangat penting.

c) DML Menggunakan *Command Prompt*

1. Menambah Record dengan INSERT

Bentuk umum perintah SQL untuk input data ke dalam sebuah tabel adalah sebagai berikut.

```
INSERT INTO [nama_tabel] VALUES ('nilai1', 'nilai2', ..., 'nilai_n');
```

Atau dapat menyebutkan nama *field*-nya seperti pada perintah berikut.

```
INSERT INTO [nama_tabel] ('field1', 'field2') VALUES  
('nilai1', 'nilai2');
```

2. Mengubah Record dengan UPDATE

Perintah update cukup penting dalam pengolahan database, karena biasanya banyak data yang perlu diperbaiki. Proses update adalah mengubah data lama dengan *record* yang baru. Perubahan dengan perintah update bersifat permanen. Bentuk umum perintah update adalah sebagai berikut.

```
UPDATE [nama_tabel] SET field1='recordbaru' WHERE kondisi;
```

3. Menghapus Record dengan DELETE

Proses *delete* adalah proses penghapusan *record* di dalam sebuah tabel. Perintah *delete* yang dilakukan bersifat permanen, sehingga perintah ini perlu dijalankan dengan sangat hati-hati. Perintah SQL untuk delete data adalah sebagai berikut.

```
DELETE FROM [nama_tabel] WHERE kondisi;
```

4. Menampilkan Record dengan SELECT

Perintah *select* berfungsi untuk menampilkan data pada sebuah tabel. Dengan menggunakan perintah *select* kita dapat mengatur tampilan atau keluaran data sesuai dengan yang diinginkan. Bentuk dasar perintah SELECT adalah sebagai berikut.

```
SELECT [field | *] FROM [nama_tabel] WHERE kondisi;
```

d) Backup dan Restore

Database yang telah dibuat dapat di *backup* atau dapat juga melakukan *restore* dari file *database* yang lain.

1. Backup

Proses *backup database* dapat dilakukan dengan menuliskan perintah sebagai berikut.

```
C:\xampp\mysql\bin>mysqldump -u root -p pw10001 > D:\backup\pw1.sql
```

Dengan melakukan proses *backup* di atas, database **pw10001** akan di-*backup* dan disimpan di direktori **D:/** pada folder **backup**.

Jika ingin mem-*backup* struktur tabelnya saja, dapat menggunakan perintah sebagai berikut.

```
C:\xampp\mysql\bin>mysqldump -u root -p -no-data pw10001 > D:\backup\pw1.sql
```

Atau jika ingin mem-*backup* datanya saja tanpa mengikutsertakan strukturnya, dapat menggunakan perintah sebagai berikut.

```
C:\xampp\mysql\bin>mysqldump -u root -p -no-create-info pw10001 > D:\backup\pw1.sql
```

2. Restore

Untuk me-*restore* sebuah *database*, terlebih dahulu siapkan sebuah database, misal **pw1_baru**. Kemudian tuliskan perintah seperti berikut ini.

```
C:\xampp\mysql\bin>mysqldump -u root -p pw1_baru < D:\backup\pw1.sql
```

3) SQL dengan “phpmyadmin”

Menurut wikipedia, **PhpMyAdmin** adalah perangkat lunak bebas / open source yang ditulis dalam bahasa pemrograman php yang digunakan untuk menangani administrasi MySQL melalui WWW (*World Wide Web*). Dengan menggunakan phpmyadmin, anda dapat membuat database, membuat tabel, menginsert, menghapus

dan mengupdate data dengan GUI dan terasa lebih mudah, tanpa perlu mengetikkan perintah SQL secara manual.

a) Primary Key

Primary Key adalah *field* kunci/utama dari sebuah tabel yang menunjukkan bahwa *field* tersebut tidak bisa diisi dengan data yang sama. Dapat dikatakan juga bahwa *primary key* menjadikan tiap *record* memiliki identitas sendiri-sendiri yang membedakan satu dengan lainnya (unik). *Primary key* akan berguna ketika menampilkan *record* hasil pencarian(*searching*), pengurutan(*sorting*), dan berbagai operasi query lainnya. Nilai dari *primary key* tidak boleh kosong, harus unik (tidak boleh sama).

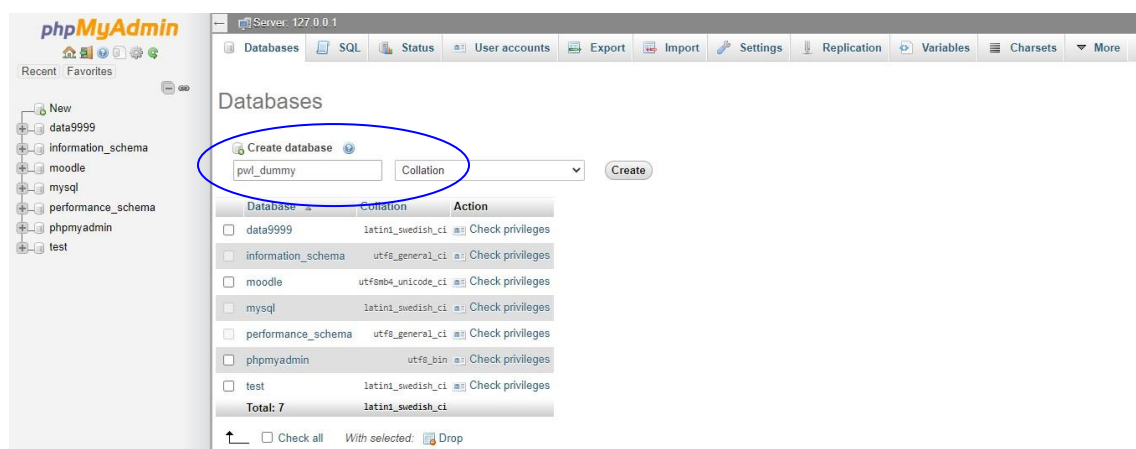
b) Foreign Key

Foreign Key adalah *field* pada sebuah tabel yang berfungsi sebagai kunci tamu dari tabel yang lain. *Foreign key* sangat berguna ketika bekerja dengan banyak tabel dan tabel tersebut memiliki relasi satu sama lain. *Field* yang menjadi *foreign key* di sebuah tabel, harus memiliki penamaan yang sama dengan tabel induknya.

c) Relasi Antar Tabel

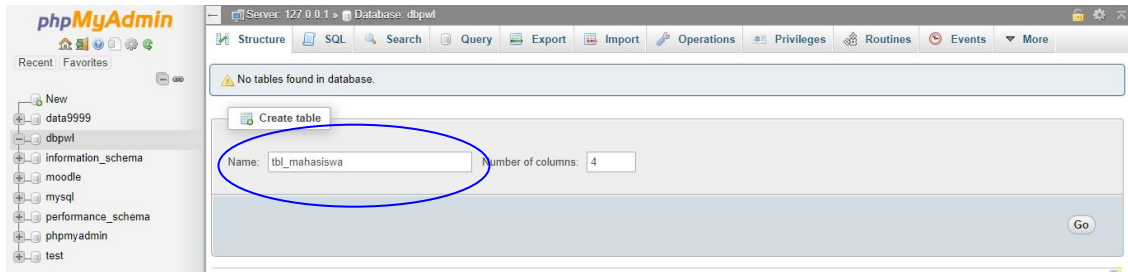
Akan dijelaskan pada proses praktikum.

Untuk membuat database di PHPMyAdmin, terlebih dahulu yang perlu dilakukan adalah menjalankan servis MySQL yang ada di XAMPP. Selanjutnya buka browser dan tuliskan localhost/phpmyadmin. Halaman yang dimunculkan adalah sebagai berikut:

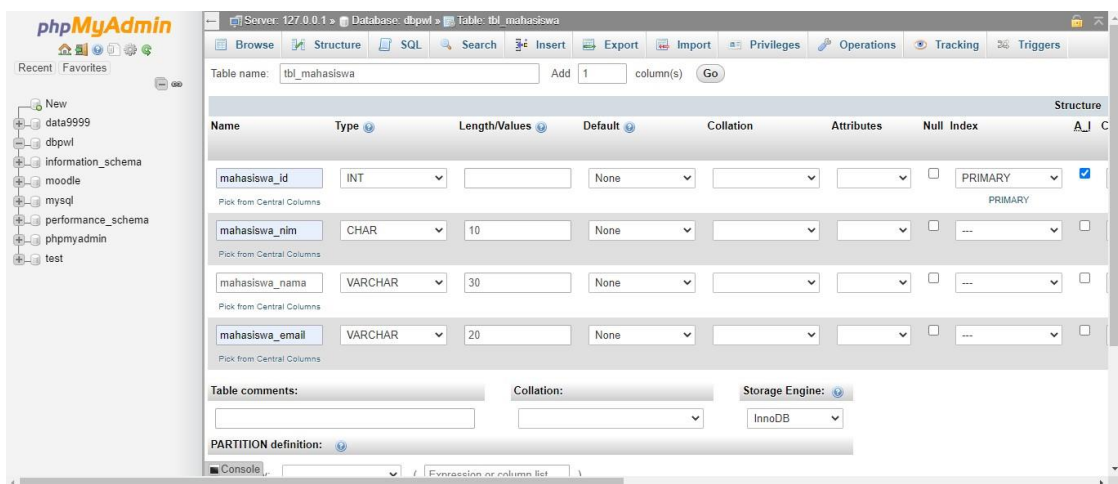


Bagian yang dilingkari digunakan untuk mendeklarasikan nama database yang akan dibuat. Penamaan database tanpa menggunakan spasi. Setelah database terbentuk,

langkah selanjutnya adalah membuat tabel sesuai dengan kebutuhan. Tabel dibuat dengan penamaan yang informatif untuk memberikan info isi dari tabel yang dibuat. Selain itu, jumlah kolom dideklarasikan ketika tabel dibuat.



Selanjutnya, untuk membuat *field* dari tabel yang telah dibuat, cukup mengisi nama *field* dan tipe data sesuai dengan kebutuhan seperti gambar berikut:



4) Koneksi Database

MySQLi atau **MySQL Improved** merupakan komponen yang digunakan untuk mengakses / berkomunikasi dengan database MySQL melalui PHP. Dengan menggunakan MySQLi, sebagai programmer PHP Anda dapat mengakses seluruh fasilitas terbaru yang ada pada MySQL versi 4.1 atau yang lebih baru.

a) Koneksi PHP dan MySQLi

Bentuk koneksi dengan menggunakan mysqli adalah sebagai berikut.

```
mysqli_connect("hostname","user","password","database_name");
```

b) Menampilkan Data

Setelah melakukan koneksi terhadap basis data yang akan dioperasikan, selanjutnya kita bisa memulai untuk melakukan manipulasi terhadap data yang

terkandung dalam basis data tersebut. Operasi yang akan kita lakukan kali ini adalah menampilkan data.

Agar proses “*migrasi*” dari **mysql** ke **mysqli** tidak terlalu menyusahkan, PHP memberikan 2 alternatif cara penulisan **mysqli**.

1. *Procedural Style*, cara ini mirip dengan *extention* mysql. Kita dapat menggunakan fungsi-fungsi untuk mengakses database MySQL

a. mysqli_query()

Digunakan untuk melakukan eksekusi perintah SQL untuk memanipulasi database yang berhasil dilakukan koneksinya.

b. mysqli_fetch_array()

Digunakan untuk melakukan pemrosesan hasil query yang dilakukan dengan perintah `mysqli_query()`, dan memasukkannya ke dalam array asosiatif, array numeris atau keduanya.

c. mysqli_fetch_assoc()

Fungsi ini hampir sama dengan fungsi `mysqli_fetch_array()`, hanya saja array yang dihasilkan hanya array asosiatif.

d. mysqli_fetch_row()

Fungsi ini hampir sama dengan fungsi `mysqli_fetch_array()`, hanya saja array yang dihasilkan hanya array numeris.

e. mysqli_num_rows()

Fungsi ini digunakan untuk menghitung jumlah record yang ada pada database.

2. *Object Oriented Style*, cara ini lebih dekat dengan aturan penulisan pemrograman berorientasi object. Kedua jenis style ini menggunakan nama fungsi dan method yang kurang lebih sama. Sebagai contoh, pada **procedural style mysqli** terdapat fungsi `mysqli_query()`, sedangkan dalam **Object oriented style** dapat menggunakan method `$mysqli->query()`.

Pada praktikum ini, kita akan menggunakan Procedural style, untuk OOP style dapat dipelajari secara mandiri.

5) *Cookie dan Session*

a) *Cookie*

Cookie adalah sebuah nilai yang dikirimkan dan ditanamkan server pada komputer *client*. Biasanya informasi-informasi yang disimpan dalam *cookie* ini

adalah informasi yang berkaitan dengan user. *Cookie* diletakkan di sisi *client*, sehingga pengguna dapat melihat bahkan memodifikasi dan menghapus *cookie* tersebut. Hal tersebut seringkali membuat penyimpanan data menggunakan *cookie* menjadi tidak efektif, apalagi user dapat menonaktifkan penggunaan *cookie* melalui *setting* browser yang mengakibatkan penggunaan *cookie* menjadi sia-sia.

Setelah sebuah variabel *cookie* dideklarasikan, ia akan disimpan di sisi *client* dan selalu tersedia saat browser mengakses website (selama masa kadaluarsa *cookie* belum habis). Berdasarkan sifat yang telah diuraikan diatas, *cookie* dapat digunakan antara lain.

1. Menyimpan nama pengunjung.
2. Merekam daftar barang yang ingin dibeli pengunjung.
3. Menyimpan pilihan-pilihan yang diatur oleh pengunjung.
4. Menciptakan suatu sesi yang memungkinkan seseorang dapat masuk ke halaman-halaman lain tanpa perlu melakukan login kembali.

Untuk membuat sebuah *cookie* PHP telah menyediakan fungsi sebagai berikut.
`setcookie(name, value, expire);`

Keterangan:

1. **Name**, untuk nama *cookie*. Digunakan sebagai pengenal *cookie*.
2. **Value**, berisi nilai yang akan disimpan dalam *cookie*.
3. **Expire**, merupakan batas waktu ketika *cookie* akan terhapus otomatis.

b) Session

Session adalah salah satu fasilitas yang ada pada PHP yang digunakan untuk menyimpan data sementara ke dalam variabel (variabel *session*) sehingga data tadi dapat di akses oleh *client* selama variabel *session* tadi tidak dikosongkan atau dihilangkan. Nilai variabel di dalam *session* di simpan di sisi server (web server). Berbeda dengan *cookies* yang nilai variabelnya disimpan di sisi *client* (browser). Jadi *session* relatif lebih aman digunakan untuk menyimpan variabel nilai yang bersifat rahasia seperti *username* dan *password* pada saat login.

1. Mengawali Session

Untuk menunjukkan bahwa suatu halaman menggunakan *session* maka pada awal halaman harus ada awal *session* yaitu dengan **`session_start ()`**

2. Penggunaan Variabel Session

Setelah *session* dimulai maka variabel *session* sudah dapat mulai digunakan. Penggunaannya menggunakan format **`$_SESSION['nama_variabel']`**

3. Menghapus Session

Setelah variabel session digunakan, variabel tersebut dapat dihapus. Banyak cara untuk menghapus session di antaranya sebagai berikut:

- a. `$_SESSION['nama_variabel'] = ""` untuk memberikan atau mengganti nilai dari variabel session menjadi null atau kosong.
- b. `unset ($_SESSION['nama_variabel'])` untuk menghapus sebuah variabel session.
- c. `session_destroy()` untuk menghapus semua variabel session yang mungkin ada banyak variabel session yang dibuat.

Contoh: latsession1.php

```
<?php
session_start();
$_SESSION['username']="farida";
echo $_SESSION['username'];
?>
```

Contoh: latsession2.php

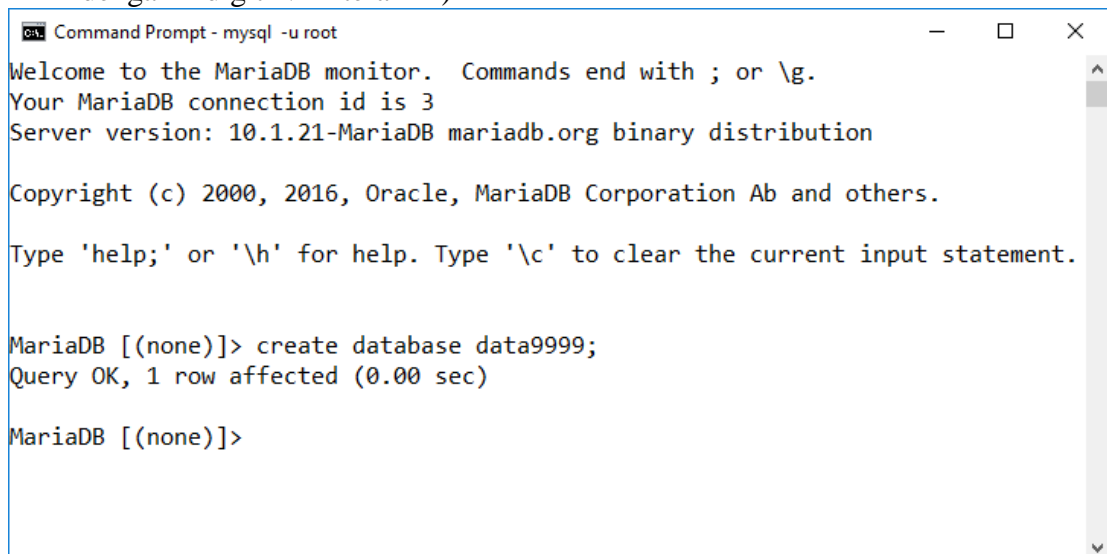
```
<?php
session_start();

echo $_SESSION['username'];
?>
```

Pada contoh kedua, meskipun variabel `$_SESSION['password']` tidak dideklarasikan, tetapi masih tetap dapat dikenali dan dioperasikan pada file yang lain dengan melakukan inisialisasi menggunakan fungsi `session_start()`.

D. Praktikum

1. Buatlah sebuah database menggunakan perintah DDL dengan nama **dataxxxx** (isi xxxx dengan 4 digit NIM terakhir)



```
Command Prompt - mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.1.21-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database data9999;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]>
```

Untuk melihat database yang anda buat, anda dapat menggunakan perintah berikut:

```
MariaDB [(none)]> show databases;
```

```
+-----+
| Database |
+-----+
| data9999 |
| information_schema |
| moodle   |
| mysql    |
| performance_schema |
| phpmyadmin |
| test     |
+-----+
7 rows in set (0.20 sec)
```

Setelah berhasil membuat database, masuk ke dalam database tersebut dengan perintah **USE**

```
MariaDB [(none)]> use data9999;
```

```
Database changed
MariaDB [data9999]>
```

Jika perintah berhasil, seperti pada gambar di atas, maka anda sudah masuk ke dalam database yang anda buat, dan database dapat digunakan untuk proses selanjutnya.

2. Buat tabel mahasiswa dengan perintah DDL(CREATE TABLE) dengan ketentuan sebagai berikut

No	Nama Tabel	Tipe Data	Lebar data	Keterangan
1	mahasiswa_id	Integer		Primary key
2	mahasiswa_nama	Varchar	50	
3	mahasiswa_nim	Varchar	10	

```
MariaDB [data9999]> create table mahasiswa(
-> mahasiswa_id int(5) primary key,
-> mahasiswa_nama varchar(50),
-> mahasiswa_nim varchar(10));
Query OK, 0 rows affected (1.65 sec)
```

```
MariaDB [data9999]>
```

Berikut adalah perintah untuk melihat tabel yang anda buat (**SHOW**) dan deskripsinya (**DESC**)

```
MariaDB [data9999]> show tables;
```

```
+-----+  
| Tables_in_data9999 |  
+-----+  
| mahasiswa          |  
+-----+  
1 row in set (0.00 sec)
```

```
MariaDB [data9999]> desc mahasiswa;
```

```
+-----+-----+-----+-----+-----+-----+  
| Field          | Type          | Null | Key | Default | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| mahasiswa_id   | int(5)        | NO   | PRI | NULL    | auto_increment |  
| mahasiswa_nama | varchar(50)   | YES  |     | NULL    |                |  
| mahasiswa_nim  | varchar(10)   | YES  |     | NULL    |                |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.04 sec)
```

3. Tambahkan *auto increment* untuk *field* mahasiswa_id agar penomoran mahasiswa_id menjadi otomatis terurut.

```
MariaDB [data9999]> alter table mahasiswa change mahasiswa_id mahasiswa_id int(5)  
auto_increment;
```

```
Query OK, 0 rows affected (0.75 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [data9999]> █
```

4. Tambahkan field untuk tabel mahasiswa seperti berikut:

No	Nama Tabel	Tipe Data	Lebar data	Keterangan
1	mahasiswa_email	Varchar	20	
2	mahasiswa_tglahir	Date		

```
MariaDB [data9999]> alter table mahasiswa add mahasiswa_email varchar(30);  
Query OK, 0 rows affected (0.38 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [data9999]> alter table mahasiswa add mahasiswa_tglahir date;  
Query OK, 0 rows affected (0.32 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Jika ingin melihat struktur tabel yang baru, gunakan perintah **desc mahasiswa**;

5. Inputkan data mahasiswa dengan perintah **insert** seperti berikut berikut:

```
MariaDB [data9999]> insert into mahasiswa(mahasiswa_nama, mahasiswa_nim, mahasiswa_email) va  
lues("Yumna", "16.12.1111", "yumna@amikom.ac.id");  
Query OK, 1 row affected (0.12 sec)
```

```
MariaDB [data9999]> insert into mahasiswa(mahasiswa_nama, mahasiswa_nim, mahasiswa_tglahir)  
values("Musa", "16.12.1122", "1999-02-25");  
Query OK, 1 row affected (0.12 sec)
```

```
MariaDB [data9999]> █
```

6. Tampilkan data dengan perintah **select** seperti berikut:

a. Menampilkan semua data dengan menggunakan tanda asterisk (*)

```
MariaDB [data9999]> select * from mahasiswa;
```

mahasiswa_id	mahasiswa_nama	mahasiswa_nim	mahasiswa_email	mahasiswa_tgl_lahir
1	Yumna	16.12.1111	yumna@amikom.ac.id	NULL
2	Musa	16.12.1122	NULL	1999-02-25

```
2 rows in set (0.00 sec)
```

b. Menampilkan sebagian field dengan menyebutkan field yang ingin ditampilkan

```
MariaDB [data9999]> select mahasiswa_nama, mahasiswa_nim from mahasiswa;
```

mahasiswa_nama	mahasiswa_nim
Yumna	16.12.1111
Musa	16.12.1122

```
2 rows in set (0.00 sec)
```

c. Menampilkan dengan kondisi

- Menampilkan nama yang emailnya kosong

```
MariaDB [data9999]> select mahasiswa_nama from mahasiswa where mahasiswa_email is null;
```

mahasiswa_nama
Musa

```
1 row in set (0.10 sec)
```

- Menampilkan data mahasiswa yang bernama "YUMNA"

```
MariaDB [data9999]> select * from mahasiswa where mahasiswa_nama like "yumna";
```

mahasiswa_id	mahasiswa_nama	mahasiswa_nim	mahasiswa_email	mahasiswa_tgl_lahir
1	Yumna	16.12.1111	yumna@amikom.ac.id	NULL

```
1 row in set (0.10 sec)
```

- Menampilkan data mahasiswa yang namanya mengandung huruf "S"

```
MariaDB [data9999]> select * from mahasiswa where mahasiswa_nama like "%s%";
```

mahasiswa_id	mahasiswa_nama	mahasiswa_nim	mahasiswa_email	mahasiswa_tgl_lahir
2	Musa	16.12.1122	NULL	1999-02-25

```
1 row in set (0.00 sec)
```

7. Ubah data di dalam tabel mahasiswa dengan menggunakan perintah **update**

```
MariaDB [data9999]> update mahasiswa set mahasiswa_email="musa@amikom.ac.id";
Query OK, 2 rows affected (0.12 sec)
Rows matched: 2 Changed: 2 Warnings: 0
```

Pada proses update di atas, terdapat dua record yang berubah. Hal ini dikarenakan tidak ada kondisi untuk memilih record mana yang akan berubah. Ketiadaan kondisi akan mengubah semua record yang ada (misal jika terdapat data 100 record, maka semua data akan berubah). Maka gunakan kondisi (**where**) ketika mengubah data.

Misal:

Update mahasiswa set mahasiswa_email="yumna@amikom.ac.id" where mahasiswa_id = 1;

8. Hapus data dengan perintah **delete**.

```
MariaDB [data9999]> delete from mahasiswa where mahasiswa_nama like "Y%";
Query OK, 1 row affected (0.14 sec)
```

```
MariaDB [data9999]> select * from mahasiswa;
```

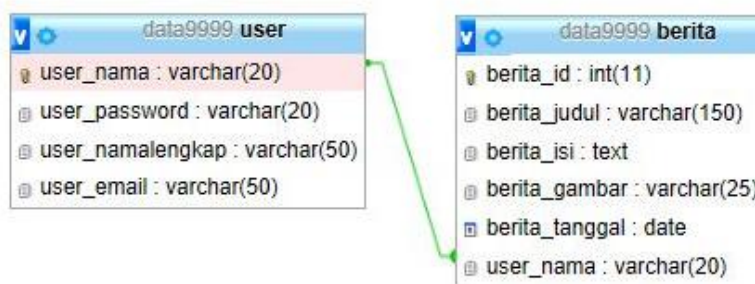
mahasiswa_id	mahasiswa_nama	mahasiswa_nim	mahasiswa_email	mahasiswa_tgl_lahir
2	Musa	16.12.1122	musa@amikom.ac.id	1999-02-25

1 row in set (0.00 sec)

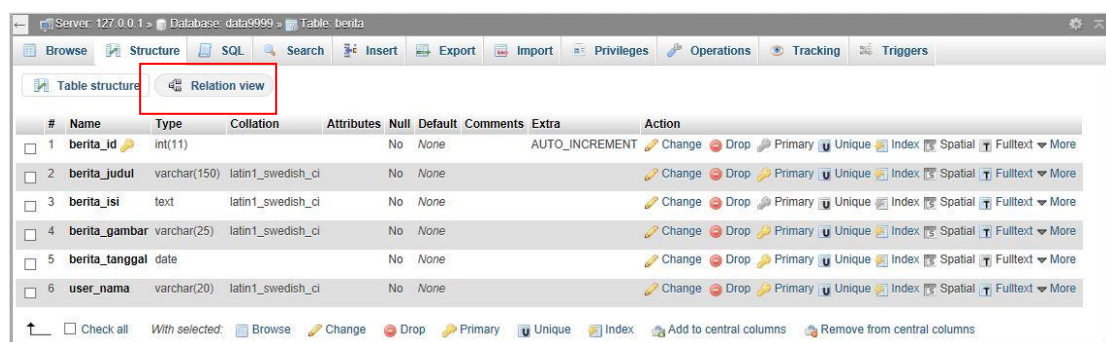
Perintah di atas adalah untuk menghapus data mahasiswa yang namanya dimulai dengan huruf "Y"

Database dengan phpMyAdmin

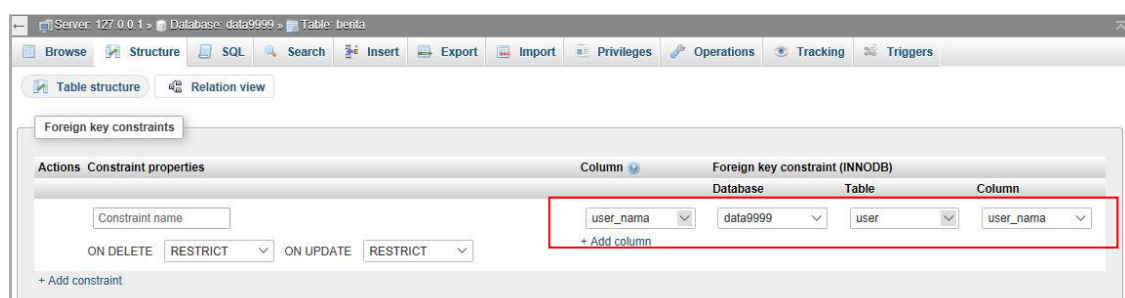
9. Buatlah sebuah database menggunakan phpMyAdmin sesuai struktur di bawah ini



10. Buatlah koneksi antar tabel untuk membuat hubungan antara tabel user dan tabel berita. Gunakan langkah berikut.



11. Klik tombol relation view pada tabel yang di dalamnya terdapat *foreign key* yang akan dibuat relasi dengan tabel induknya.



Selanjutnya isikan/set pengaturan seperti berikut:

- a. **Column** berisi field *foreign key* (**user_nama**) dari tabel berita
- b. **Database** berisi nama database yang saat ini digunakan
- c. **Table** adalah pilihan untuk tabel induk (tabel user)
- d. **Column** adalah pilihan field *primary key* (**user_nama**) dari tabel user

12. Isikan masing-masing satu data untuk tabel user dan berita melalui phpMyadmin

13. Buatlah file koneksi di PHP dan file PHP untuk menampilkan data

- a. Membuat Koneksi Database

config.php

```
<?php
$config =
mysqli_connect("localhost","root","","data9999");
if (!$config) {
    die('Gagal terhubung ke MySQL:
'.mysqli_connect_error());
}
?>
```

- b. Menampilkan Data Tunggal

username.php

```
<?php
include "config.php"; //memanggil file koneksi database ke PHP

$sql = "Select * from user";
$query = mysqli_query($config,$sql);
$row = mysqli_fetch_assoc($query);

//data ditampilkan dalam bentuk array dalam bentuk pertama
echo "<pre>";
print_r($row);
echo "<pre>";

//atau bentuk kedua
echo "Username :".$row['user_nama']."<br />";
echo "Nama Lengkap :".$row['user_namalengkap']."<br />";
echo "Email :".$row['user_email']."<br />";

?>
```

Keterangan:

- Fungsi **include** bertugas untuk memasukkan file config.php ke dalam file username.php. Tugasnya membuat koneksi ke server database.
- Query “SELECT * from user” disimpan di dalam variabel **\$sql**. Kemudian dijalankan dengan perintah `mysqli_query($config,$sql)` yang disimpan di dalam variabel **\$query**.

- Hasil *query* yang tersimpan di variabel **\$row** adalah data mentah yang akan diolah dengan fungsi `mysqli_fetch_assoc($query)` yang berguna untuk mengolah data mentah tersebut ke dalam bentuk array. Selanjutnya disimpan di dalam variabel **\$row**.
- Selanjutnya adalah perintah untuk menampilkan data. Variabel **\$query** disimpan dalam bentuk array sehingga dapat memanfaatkan fungsi `print_r` untuk menampilkan struktur array tersebut. Atau dalam bentuk kedua array **\$query** yang ditampilkan merupakan kolom yang sebelumnya didefinisikan pada *query* (username, nama lengkap, dan email), sehingga untuk menampilkan kita perlu menyebut array dengan index asosiatifnya.

c. Menampilkan data jamak

Untuk menampilkan data yang banyak perlu dideskripsikan perintah **`mysqli_fetch_assoc($query)`** yang dilakukan secara berulang hingga data yang diinginkan telah mencukupi atau sampai kondisi dimana data yang dihasilkan dari *query* telah habis. Oleh sebab itu, perlu sebuah perulangan yang akan menampilkan data yang jumlahnya banyak.

```
<?php
include "config.php"; //memanggil file koneksi database ke PHP

$sql = "Select * from user";
$query = mysqli_query($config,$sql);

echo "<pre>";
while($row = mysqli_fetch_assoc($query)){ // fungsi perulangan untuk menampilkan data jamak
    print_r($row);
}
echo "<pre>";
?>
```

14. Latihan Cookie dan Session

a. Latihan Cookie

latcookie.php

```
<?php
$value="12";
setcookie("barang",$value, time()+180);
echo "Cookie telah dibuat";
?>
```

Untuk membaca/mengakses cookie:

Latihan: latcookie2.php

```
<?php
echo "Jumlah Barang: ";
if(isset($_COOKIE['barang'])){
    echo $_COOKIE['barang'];
}
?>
```

Untuk menghapus nilai cookie gunakan **`setcookie()`** dengan menyebutkan nama cookie pada argumen pertama dan string kosong pada argumen kedua dan mengisikan

nilai dari parameter *expire*-nya dengan satu jam yang lalu. Cara lain, argumen kedua tidak disebutkan.

Latihan:hapuscookie.php

```
<?php
setcookie("barang","",time()-180);
?>
```

b. Latihan Session untuk Program Autentikasi welcome.php

```
<?php
session_start();
if(isset($_SESSION['username'])){ ?>
    <h2>Control Panel</h2>
    <p>Selamat datang "<?php echo $_SESSION['username']; ?>".
    Klik <a href="logout.php">disini</a> untuk logout.</p>
<?php
} else { ?>
    <h2>Maaf..</h2>
    <p>Anda tidak berhak mengakses halaman ini.
    Silakan <a href="login.php">login</a> terlebih dahulu.</p>
<?php
}
?>
```

Pada halaman welcome.php di atas, akan dilakukan pemeriksaan terlebih dahulu apakah seorang *user* telah terautentikasi melalui fungsi **isset(\$_SESSION['username'])**. Asumsi dari contoh tersebut adalah user telah melakukan login dengan user dan password. Jika belum terautentikasi user akan dibawa ke halaman login.

login.php

```
<html>
<head>
    <title>Form Login</title>
</head>
<body>
    <form action="login_action.php" method="post">
        Username : <input type="text" name="txtUsername" />
        <br />
        Password : <input type="password" name="txtPassword" />
        <br />
        <input type="submit" value="Login" />
    </form>
</body>
</html>
```

login_action.php

```
<?php
session_start();
include("config.php");

$sql = "SELECT user_nama FROM user ";
$sql.= "WHERE user_nama='".$$_POST['txtUsername']."' ";
$sql.= "AND user_password='".$$_POST['txtPassword']."' ";
$hasil = mysqli_query($config,$sql) or exit("Error query :
<b>".$sql."</b>.");
if(mysqli_num_rows($hasil)>0){
    $data = mysqli_fetch_array($hasil);
    $_SESSION['username'] = $data['username'];
    header("Location:welcome.php");
    exit();
} else { ?>
    <h2>Maaf..</h2>
    <p>
        Username atau password salah.
        Klik <a href="login.php">disini</a> untuk kembali login.
    </p>
    <?php
}
?>
```

Fungsi **mysql_num_rows()** berguna untuk menghitung baris data yang dihasilkan dari sebuah query. Dalam kasus ini, fungsi tersebut akan mengembalikan nilai 0, jika data username dan password yang diinputkan tidak cocok dengan data manapun pada basis data. Jika proses autentikasi berhasil, selanjutnya user akan dibawa (*redirect*) ke halaman welcome.php melalui statement **header("Location:welcome.php");**.

15. Tantangan:

Menambah Data di Database

- Tambahkan 3 data mahasiswa dengan isian yang lengkap, dengan ketentuan:
 - bulan lahir Mei, Juni, Juli
 - mahasiswa_nim angkatan 16.12.xxxx, 17.12.xxxx, 17.62.xxxx (isikan xxxx dengan angka sembarang)
- Ubah data email mahasiswa dengan kondisi bulan lahirnya adalah Juli
- Hapus data mahasiswa dengan kondisi angkatan 17 dan program studi SI (kode 12)
- Print screen masing-masing perintah dan beri keterangan.
- Simpan dalam bentuk PDF dan beri keterangan Nama dan NIM anda, lalu laporkan kepada Dosen/Asisten

Cookie dan Session

- Tambahkan script berikut untuk membuat halaman logout

```
<?php
....
.... dst.
?>

<h2>Terima Kasih</h2>
<p>
Anda telah logout dari system.
Klik <a href="....">disini</a> untuk kembali login.
</p>
```

E. Tugas

1. Jika kalian diminta untuk membuat sebuah data presensi, tabel apa yang diperlukan
2. Buatlah rancangan tabel dan tentukan fieldnya
3. Buat tabel dan field dengan menggunakan PHPMyAdmin

F. Referensi

Arief, M. R. 2011. *Pemrograman Web Dinamis menggunakan PHP dan MySQL*. Yogyakarta: Andi Offset.

Hakim, Lukmanul. 2014. *Rahasia Inti Master PHP & MySQLi(improved)*. Yogyakarta, Lokomedia

Paranginan, Kasiman. 2012. *Aplikasi Web dengan PHP dan MySQL*. Yogyakarta: Andi Offset.

Raharjo, Budi., dkk. 2014. *Modul Pemrograman Web HTML, PHP, & MySQL*. Bandung: Modula.

Farida, Lilis Dwi. 2017. *Modul Pemrograman Web Lanjut*. Yogyakarta: Universitas AMIKOM Yogyakarta