MODUL 7
RELASI MYSQL DAN KONEKSI MYSQL DENGAN PHP

7.1 Tujuan Praktikum

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat:

1. Memahami konsep *database* MySQL dan operasi CRUD.
2. Mengimplementasikan *query* SQL untuk relasi antar tabel.
3. Membangun aplikasi web sederhana dengan integrasi PHP dan MySQL

7.2 Alat dan Bahan

Alat dan bahan yang digunakan dalam praktikum ini meliputi:

1. Sistem Operasi: Windows, Linux, atau MacOS.
2. PHP versi 8.x.
3. Perangkat lunak XAMPP.
4. Editor teks.

7.3 Dasar Teori

7.3.1 MySQL

MySQL adalah sebuah *Relational Database Management System* (RDBMS) sumber terbuka yang menggunakan *Structured Query Language* (SQL) untuk mengelola, mengorganisasi, dan mengambil data. MySQL dirilis di bawah lisensi ganda: GNU General Public License (GPL) versi 2 untuk penggunaan dalam ekosistem sumber terbuka, dan lisensi komersial untuk keperluan *proprietary*. Sebagai perangkat lunak bebas di bawah GPL, MySQL dapat digunakan secara gratis untuk keperluan pribadi maupun komersial, dengan syarat pengguna mematuhi ketentuan lisensi tersebut (misalnya, menyertakan kode sumber modifikasi jika didistribusikan). Untuk pengguna yang tidak ingin terikat dengan persyaratan GPL, tersedia opsi lisensi berbayar dari Oracle Corporation (pemilik MySQL saat ini). Dalam operasional basis data, SQL pada MySQL mendukung beberapa kategori perintah utama:

1. *Data Definition Language* (DDL)

   DDL adalah sub perintah dalam bahasa SQL yang digunakan untuk membangun struktur sebuah *database*, termasuk *database* dan tabel. Terdapat tiga perintah penting dalam DDL, yaitu:

Perintah	Deskripsi
CREATE	Perintah untuk membuat *database*, tabel, *view*, atau kolom baru.
DROP	Perintah untuk menghapus struktur dalam suatu *database*.
ALTER	Perintah untuk mengubah struktur tabel yang telah ada, termasuk mengubah nama tabel, menambah kolom, atau menghapus kolom.

2. *Data Manipulation Language* (DML)

   DML adalah sub perintah dalam bahasa SQL yang digunakan untuk memanipulasi data dalam *database* yang telah dibuat. Terdapat empat perintah utama dalam DML, yaitu:

Perintah	Deskripsi
INSERT	Perintah untuk memasukkan data baru ke dalam tabel.
SELECT	Perintah untuk memilih dan mengambil data dari tabel untuk ditampilkan.
UPDATE	Perintah untuk mengubah data yang ada pada tabel.
DELETE	Perintah untuk menghapus data dari tabel.

Contoh *query* untuk masing-masing perintah di atas adalah sebagai berikut:

- INSERT

```
INSERT INTO mahasiswa_t (nama, nim, email) VALUES
('rical', '607052300001', 'rical@telnet.net');
```

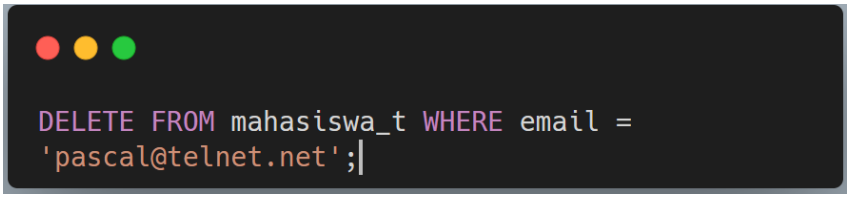
- SELECT

```
SELECT nama, nim, email FROM mahasiswa_t;
```

- UPDATE

```
UPDATE mahasiswa_t SET nama = 'Pascal', nim =
'607052300002', email = 'pascal@telnet.net' WHERE
nama = 'rical';
```

- DELETE



```
DELETE FROM mahasiswa_t WHERE email =
'pascal@telnet.net';
```

3. *Data Control Language* (DCL)

   DCL adalah sub bahasa SQL yang berfungsi untuk mengontrol data dan server *database*, termasuk manipulasi pengguna dan hak aksesnya. Terdapat empat perintah dalam DCL, yaitu:

Perintah	Deskripsi
GRANT	Digunakan untuk memberikan izin kepada pengguna untuk mengakses tabel dalam *database*.
REVOKE	Digunakan untuk mencabut izin hak akses pengguna.
COMMIT	Digunakan untuk menetapkan penyimpanan *database*.
ROLLBACK	Digunakan untuk membatalkan penyimpanan *database*.

7.3.2 Struktur Database

A. Tabel

Tabel adalah struktur dasar dalam basis data relasional yang menyimpan data dalam format baris (*row*) dan kolom (*column*). Setiap kolom merepresentasikan atribut data tertentu (misalnya: NIM, nama, kelas), sedangkan setiap baris mewakili satu entitas atau rekaman data lengkap (contoh: data individu mahasiswa).

B. *Field*

Kolom (atau *field*) adalah unit data spesifik yang mendefinisikan atribut dari suatu entitas dalam tabel. Contoh kolom pada tabel mahasiswa meliputi NIM, nama, kelas, dan jurusan. Setiap kolom memiliki tipe data tertentu (misalnya: integer, varchar, date) yang membatasi nilai yang dapat dimasukkan.

C. *Record*

Baris (atau *record*) adalah kumpulan nilai kolom yang saling terkait dan merepresentasikan satu entitas utuh dalam tabel. Satu baris pada tabel mahasiswa,

misalnya, berisi data lengkap seorang mahasiswa seperti NIM: 6070523, nama: "Risnanda", kelas: "4701", dan jurusan: "D3TT".

D. *Primary Key* (PK)

Primary Key adalah kolom atau kombinasi kolom yang secara unik mengidentifikasi setiap baris dalam tabel. Syarat utama *Primary Key* adalah:

1. Nilainya tidak boleh duplikat di dalam tabel.
2. Nilai harus selalu terisi (tidak boleh *null*).
3. Nilainya tidak boleh berubah sepanjang siklus hidup data.

E. *Foreign Key* (FK)

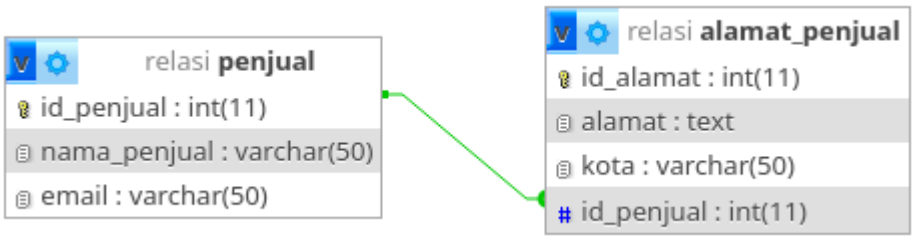
Foreign Key adalah kolom atau kombinasi kolom dalam suatu tabel yang merujuk ke *Primary Key* di tabel lain, membentuk hubungan relasional (*relationship*) antar tabel. Contoh: Kolom **mahasiswa_id** di tabel **kelas** adalah *Foreign Key* yang merujuk ke *Primary Key* **mahasiswa_id** di tabel **mahasiswa**. Sifat *Foreign Key*:

1. Dapat berisi nilai *null* (jika tidak diatur sebagai wajib).
2. Dapat memiliki nilai duplikat (kecuali jika diindeks secara unik).
3. Nilainya harus sesuai dengan nilai *Primary Key* yang dirujuk (*referential integrity*).
4. Perubahan atau penghapusan nilai *Primary Key* yang dirujuk dapat dibatasi oleh aturan *ON UPDATE* atau *ON DELETE* (misalnya: *CASCADE*, *RESTRICT*, *SET NULL*).

7.3.3 Relasi Database

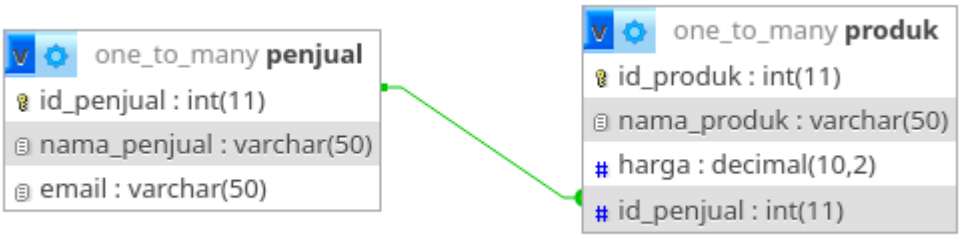
A. Relasi One-to-One (1:1)

Relasi One-to-One adalah hubungan di mana satu baris dalam tabel A hanya terkait dengan satu baris dalam tabel B, dan sebaliknya. Relasi ini jarang digunakan dalam desain basis data konvensional dan biasanya diterapkan untuk alasan optimasi atau keamanan (misalnya, memisahkan data sensitif ke tabel terpisah).



B. Relasi One-to-Many (1:M)

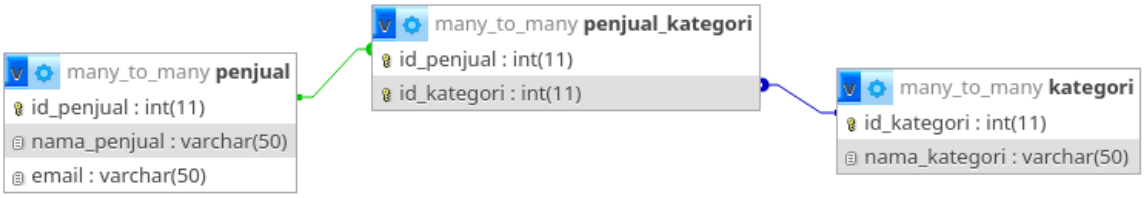
Relasi One-to-Many adalah hubungan di mana satu baris dalam tabel A dapat terkait dengan banyak baris dalam tabel B, tetapi satu baris dalam tabel B hanya merujuk ke satu baris dalam tabel A. Ini adalah jenis relasi paling umum dalam basis data relasional.



Tabel penjual dengan *Primary Key* id_penjual berelasi 1:N dengan tabel produk. Satu penjual dapat menjual banyak produk, tetapi setiap produk hanya terdaftar di bawah satu penjual.

C. Relasi Many-to-Many (M:M)

Relasi Many-to-Many adalah hubungan di mana banyak baris dalam tabel A dapat terkait dengan banyak baris dalam tabel B. Untuk mengimplementasikan relasi ini, diperlukan tabel perantara (*junction table*) yang menyimpan *Foreign Key* dari kedua tabel.



Tabel penjual dan kategori dihubungkan melalui tabel perantara penjual_kategori. Satu penjual dapat menjual produk dalam banyak kategori (misalnya: makanan dan minuman), sementara satu kategori dapat dimiliki oleh banyak penjual.

7.3.4 Query SQL Relasi

Untuk mengambil data dari tabel yang saling berelasi dalam SQL, digunakan operasi JOIN. Operasi ini menggabungkan baris dari dua atau lebih tabel berdasarkan kondisi relasi kolom terkait. Berikut jenis-jenis utama JOIN dalam SQL standar:

1. INNER JOIN

   INNER JOIN menggabungkan baris dari dua tabel hanya jika nilai kolom penghubung (*key*) memenuhi kondisi yang ditentukan. Baris tanpa pasangan di kedua tabel tidak akan ditampilkan.

```sql
SELECT m.*, k.*
FROM mahasiswa_t m
INNER JOIN kelas_t k ON m.kelas_id = k.id;
```

2. LEFT JOIN

   LEFT JOIN menampilkan semua baris dari tabel kiri (tabel pertama), dan baris yang cocok dari tabel kanan (tabel kedua). Jika tidak ada pasangan, kolom dari tabel kanan akan berisi nilai NULL.

```sql
SELECT m.*, k.*
FROM mahasiswa_t m
LEFT JOIN kelas_t k ON m.kelas_id = k.id;
```

3. RIGHT JOIN

   RIGHT JOIN menampilkan semua baris dari tabel kanan (tabel kedua), dan baris yang cocok dari tabel kiri. Jika tidak ada pasangan, kolom dari tabel kiri akan berisi nilai NULL.

```sql
SELECT m.*, k.*
FROM mahasiswa_t m
RIGHT JOIN kelas_t k ON m.kelas_id = k.id;
```

4. FULL JOIN

   FULL JOIN menampilkan semua baris dari kedua tabel, dengan nilai NULL di kolom yang tidak memiliki pasangan. Catatan: MySQL tidak mendukung sintaks FULL JOIN secara *native*. Untuk meniru perilaku ini, gunakan kombinasi LEFT JOIN dan RIGHT JOIN dengan klausa UNION:

```sql
SELECT m.*, k.*
FROM mahasiswa_t m
LEFT JOIN kelas_t k ON m.kelas_id = k.id
UNION
SELECT m.*, k.*
FROM mahasiswa_t m
RIGHT JOIN kelas_t k ON m.kelas_id = k.id;
```

7.3.5 Perintah SQL pada PHP

1. Membuat Koneksi ke Server MySQL dengan mysqli_connect()

Perintah mysqli_connect() digunakan untuk menguji dan membuat koneksi ke server database MySQL. Penulisan perintahnya adalah sebagai berikut:

- $object = new mysqli("localhost", "root", "", "prakmod7_db");
- $conn = mysqli_connect("localhost", "root", "", "prakmod7_db");

Untuk menutup koneksi, dapat menggunakan perintah:

- $object->close();
- mysqli_close($conn);

2. Menjalankan Query SQL dengan mysqli_query()

Perintah mysqli_query() digunakan untuk menjalankan perintah-perintah SQL agar dapat memanipulasi data pada tabel. Dalam perintah ini, parameter "*request*" berisi *query* SQL seperti *SELECT*, *INSERT*, *UPDATE*, *DELETE*, dan lain-lain. Penulisan perintahnya adalah sebagai berikut:

- $object->query($sql);
- mysqli_query($conn, $sql);

3. Mengambil Record dari Database

Terdapat beberapa perintah yang dapat digunakan untuk mengambil *record* dari suatu database. Umumnya, *record* tersebut akan memiliki tipe data array.

1. Tipe Data Array: Perintah mysqli_fetch_array() dapat digunakan untuk menyimpan record dalam bentuk array asosiatif, array numerik, atau keduanya. Penulisan perintahnya adalah:
    - $object->query($sql)->fetch_array();
    - mysqli_fetch_array($result);
2. Tipe Data Array Asosiatif: Perintah mysqli_fetch_assoc() digunakan untuk menyimpan record dalam bentuk array asosiatif saja. Penulisan perintahnya adalah:
    - $object->query($sql)->fetch_assoc();
    - mysqli_fetch_assoc($result);
3. Tipe Data Array Numerik: Perintah mysqli_fetch_row() digunakan untuk menyimpan record dalam bentuk array numerik saja. Penulisan perintahnya adalah:
    - $object->query($sql)->fetch_row();

- mysqli_fetch_row($result);