

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/376518272>

# Pakar Implementasi Sistem Basis Data UAS

Article · December 2023

CITATIONS

0

READS

59

5 authors, including:



**Jordi Irawan**

Universitas Palangka Raya

3 PUBLICATIONS 0 CITATIONS

SEE PROFILE



**Depro Winoto**

Universitas Palangka Raya

3 PUBLICATIONS 0 CITATIONS

SEE PROFILE



**Hariyadi Hariyadi**

Universitas Palangka Raya

3 PUBLICATIONS 0 CITATIONS

SEE PROFILE



**Lia Nelda**

Universitas Palangka Raya

1 PUBLICATION 0 CITATIONS

SEE PROFILE

**LAPORAN UJIAN AKHIR SEMESTER**  
**BASIS DATA I**  
**IMPLEMENTASI SISTEM BASIS DATA PADA APLIKASI PAKAR**



**Dosen Pengampu:**

Novera Kristianti, S.T., M.T.

NIDN: 0016119301

**Disusun oleh:**

Jordi Irawan	223010503002
Veven Desta Aditia	223010503004
Depro Winoto	223010503013
Hariyadi	223010503021
Lia Nelda	223010503012

**JURUSAN TEKNIK INFORMATIKA**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS PALANGKARAYA**

**2023**

# **BAB I**

## **TUJUAN DAN DASAR TEORI**

### **1.1. Tujuan**

- 1.1.1. Mahasiswa mampu menerapkan Bahasa Query formal dan terapan.
- 1.1.2. Mahasiswa mampu melakukan manipulasi data di database
- 1.1.3. Mahasiswa mampu melakukan optimasi pada query
- 1.1.4. Mahasiswa mampu mengimplementasikan database di sistem

### **1.2. Dasar Teori**

#### **1.2.1. Bahasa Query**

##### **A. Bahasa Query Formal**

Bahasa Query Formal merupakan bahasa Query yang diterjemahkan dengan menggunakan simbol-simbol matematis.

Bahasa Query Formal Prosedural (Aljabar Relasional)

Terdapat 5 operasi dasar dari Aljabar Relasional, yaitu :

##### **1. Select,**

Operasi select berfungsi untuk menyeleksi tuple-tuple yang memenuhi

predikat yang diberikan dari sebuah tabel relasi. Simbol sigma “ $\sigma$ ”

digunakan untuk menunjukkan operasi select

##### **2. Project,**

Operasi Project berfungsi untuk memilih nilai atribut-atribut tertentu saja dari

sebuah tabel relasi. Simbol Phi “ $\pi$ ” digunakan untuk menunjukkan operasi

project.

##### **3. Cartesian product,**

Operasi Cartesian Product berfungsi untuk mengkombinasikan informasi

yang ada dalam 2 tabel relasi yang baru. Simbol “x”.

4. Union,

Operasi Union berfungsi untuk mendapatkan gabungan nilai atribut dari sebuah table relasi dengan nilai atribut dari table relasi lainnya. Simbol “U”.

5. Set diferent,

Operasi Set Difference berfungsi untuk mendapatkan nilai yang ada dalam sebuah tabel relasi, tetapi tidak ada dalam table relasi lainnya. Simbol “-”.

B. Bahasa Query Terapan

Structured Query Language (SQL) merupakan bahasa query terapan yang banyak digunakan oleh berbagai DBMS, diterapkan dalam berbagai development tools dan program aplikasi untuk berinteraksi dengan basis data. SQL dibagi menjadi 2 yaitu:

1. Data Definition Language (DDL) Query-query ini digunakan untuk mendefinisikan struktur atau skema basis data.

Statemen SQL yang termasuk ke dalam DDL

CREATE DATABASE

CREATE TABEL

CREATE INDEX

CREATE VIEW

DROP DATABASE

DROP TABEL

DROP INDEX

DROP VIEW

ALTER TABLE

2. Data Manipulation Language (DML) Query-query ini digunakan untuk manajemen data dalam basis data.

Statemen SQL yang termasuk ke dalam DDL :

INSERT,  
SELECT,  
UPDATE,  
DELETE

### 1.2.2. Manipulasi Data Dengan Bahasa Query

Bahasa query adalah bahasa yang digunakan untuk mengakses data dalam basis data. Bahasa query dapat digunakan untuk melakukan berbagai operasi data, seperti menyisipkan, menghapus, memperbarui, dan mengambil data. Manipulasi data adalah proses mengubah data dalam basis data. Manipulasi data dapat dilakukan dengan menggunakan bahasa query.

Ada berbagai jenis operasi manipulasi data, antara lain:

- Penyisipan data (INSERT)
- Penghapusan data (DELETE)
- Pembaruan data (UPDATE)
- Pengambilan data (SELECT)

#### a. Penyisipan Data (INSERT)

Penyisipan data adalah proses menambahkan data baru ke dalam basis data. Perintah INSERT digunakan untuk melakukan operasi penyisipan data.

Sintak perintah INSERT adalah sebagai berikut:

INSERT INTO tabel (kolom1, kolom2, ...) VALUES (nilai1, nilai2, ...)
---

#### b. Penghapusan Data (DELETE)

Penghapusan data adalah proses menghapus data dari basis data. Perintah DELETE digunakan untuk melakukan operasi penghapusan data.

Sintak perintah DELETE adalah sebagai berikut:

DELETE FROM tabel [WHERE kondisi]
-----------------------------------

c. Pembaruan Data (UPDATE)

Pembaruan data adalah proses mengubah data dalam basis data. Perintah UPDATE digunakan untuk melakukan operasi pembaruan data.

Sintak perintah UPDATE adalah sebagai berikut:

UPDATE tabel SET kolom1 = nilai1, kolom2 = nilai2, ... [WHERE kondisi]
--

d. Pengambilan Data (SELECT)

Pengambilan data adalah proses mengambil data dari basis data. Perintah SELECT digunakan untuk melakukan operasi pengambilan data.

Sintak perintah SELECT adalah sebagai berikut:

SELECT kolom1, kolom2, ... FROM tabel [WHERE kondisi]
--

### 1.2.3. Proses dan Optimasi Query

Proses query adalah proses untuk mengeksekusi suatu query.

Proses query terdiri dari beberapa tahap, yaitu:

- Analisis query
- Pemilihan rencana query
- Eksekusi rencana query

Optimasi Query adalah sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu query untuk membuat evaluasi

tersebut menjadi lebih efektif. Tujuan dari optimasi query adalah menemukan jalan akses yang termurah untuk meminimumkan total waktu pada saat proses sebuah query. Untuk mencapai tujuan tersebut, maka diperlukan optimizer untuk melakukan analisa query dan untuk melakukan pencarian jalan akses. Optimasi Query dapat dilakukan dengan mengoptimalkan ekspresi Aljabar Relasional seperti : UNION, INTERSECTION, SET DIFFERENCE dan CARTESIAN PRODUCT. Kelompok yang kedua terdiri dari operasi-operasi yang khusus dibuat untuk database-database relasional. Yang termasuk dalam kelompok yang kedua adalah SELECT, PROJECT dan JOIN

#### **1.2.4. Proses dan Optimasi Query**

Manajemen Basis Data (Database Management System, DBMS) adalah sistem perangkat lunak yang digunakan untuk mengelola basis data. DBMS menyediakan berbagai fitur untuk mengelola basis data, seperti:

- Membuat dan mengelola table
- Menyimpan dan mengambil data
- Memperbarui dan menghapus data
- Melakukan pencarian data
- Mendukung keamanan dan integritas data

Fungsi utama manajemen basis data adalah untuk mengelola basis data secara efektif dan efisien. Fungsi-fungsi manajemen basis data meliputi:

- Perancangan basis data adalah proses untuk menentukan struktur dan data dalam basis data.
- Pemeliharaan basis data adalah proses untuk menjaga basis data agar tetap konsisten dan akurat.

- Keamanan basis data adalah proses untuk melindungi basis data dari akses yang tidak sah.
- Integritas basis data adalah proses untuk memastikan bahwa data dalam basis data tetap konsisten.
- Performa basis data adalah proses untuk meningkatkan kinerja dari basis data.

Sistem manajemen basis data (DBMS) adalah perangkat lunak yang digunakan untuk mengelola basis data. DBMS menyediakan berbagai fitur untuk mengelola basis data, seperti:

- Membuat dan mengelola table
- Menyimpan dan mengambil data
- Memperbarui dan menghapus data
- Melakukan pencarian data
- Mendukung keamanan dan integritas data

#### **1.2.5. Implementasi Sistem**

Sistem adalah sekumpulan elemen yang saling berkaitan dan saling mempengaruhi dalam rangka mencapai tujuan tertentu. Elemen-elemen tersebut dapat berupa komponen fisik, komponen nonfisik, atau kombinasi keduanya.

Syarat-syarat Sistem :

- Sistem harus dibentuk untuk menyelesaikan masalah.
- Elemen sistem harus mempunyai rencana yang ditetapkan.
- Adanya hubungan diantara elemen sistem.
- Unsur dasar dari proses (arus informasi, energi dan material) lebih penting dari pada elemen sistem.
- Tujuan organisasi lebih penting dari pada tujuan elemen.

Tahap Pengembangan Sistem

- Perencanaan Sistem
- Analisis Sistem
- Perancangan Sistem



- Implementasi dan Pemeliharaan Sistem

Implementasi sistem adalah tahap meletakkan sistem informasi baru ke dalam sistem yang sudah ada (sistem lama). Tahap ini merupakan tahap yang paling kritis dalam pengembangan sistem, karena berkaitan dengan keberhasilan penerapan sistem baru.

#### Tujuan Implementasi Sistem

- Untuk mengoperasikan sistem baru.
- Untuk memenuhi kebutuhan pemakai sistem.
- Untuk memastikan bahwa sistem baru memenuhi persyaratan yang telah ditetapkan.

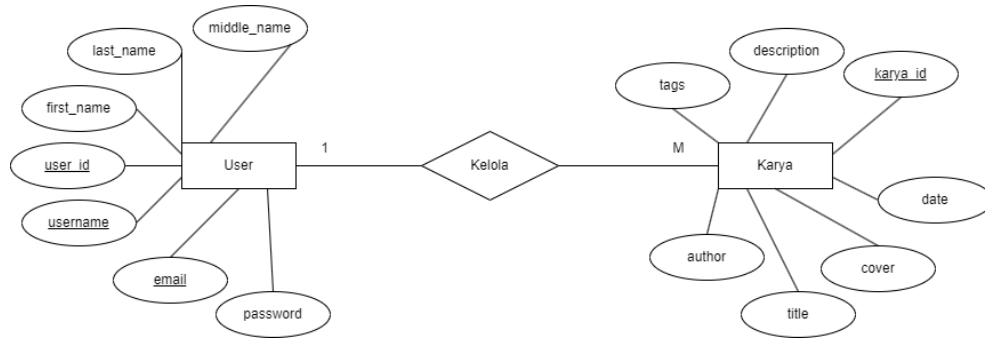
#### Langkah-langkah Implementasi Sistem

- Membuat dan menguji basis data dan jaringan
- Membuat dan menguji program
- Memasang dan menguji sistem baru
- Mengirim sistem baru ke dalam sistem lama

## BAB II

### PEMBAHASAN

#### 2.1. Entity Relational Diagram (ERD)



**Gambar 2.1.** Entity Relation Diagram

Pada **Gambar 2.1.** Merupakan sebuah ERD dari Aplikasi Pakar yang mana pengguna dapat membuat, melihat, menghapus, mengedit setiap karya yang mereka buat dan setiap karya hanya bisa dikelola oleh seorang pengguna pemilih karya tersebut (*One-To-Many*).

#### 2.2. Skema Tabel

Berikut ini perencanaan Skema dari tabel yang akan dibuat dari **Gambar**

##### 2.1.

```
CREATE TABLE user
(
    user_id          INT(11)          NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    username         VARCHAR(191)     NOT NULL UNIQUE,
    password         VARCHAR(191)     NOT NULL,
    first_name       VARCHAR(191)     NOT NULL,
    last_name        VARCHAR(191),
    middle_name      VARCHAR(191),
    email            VARCHAR(191)     NOT NULL UNIQUE,
);
```

**Gambar 2.2.** Skema Tabel User

```
CREATE TABLE karya (
    karya_id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    tag VARCHAR(191) NOT NULL,
    description VARCHAR(191),
    title VARCHAR(191),
    date DATETIME(3) default current_timestamp() NOT NULL ,
    cover VARCHAR(191),
```

```
author INT(11) NOT NULL,
FOREIGN KEY (author) REFERENCES user(user_id)
);
```

**Gambar 2.3.** Skema Tabel Karya

### 2.3. Bahasa Query

Berikut ini beberapa Fitur dari Aplikasi Pakar yang menggunakan bahasa Query

#### 2.3.1. *Login*

```
SELECT user_id FROM user
WHERE email = 'email' AND password = 'password';
```

**Gambar 2.4.** Bahasa Query Terapan – *Login*

```
 $\pi$  user_id ( $\sigma$  email='email'  $\wedge$  password='password'(user))
```

**Gambar 2.5.** Bahasa Query Formal – *Login*

Query ini mencari *user\_id* dari tabel *user* di mana nilai kolom *email* sama dengan 'email' dan nilai kolom *password* sama dengan 'password'.

#### 2.3.2. *FindOneKarya*

```
SELECT k.karya_id, k.title, k.description, k.date,
k.cover, k.tag, u.user_id, u.username, u.first_name,
u.middle_name, u.last_name
FROM karya as k
JOIN user as u ON k.author = u.user_id
WHERE k.karya_id = 'nilai_karyaid';
```

**Gambar 2.6.** Bahasa Query Terapan – *FindOneKarya*

```
 $\pi$ karya_id, title, description, date, cover, tag, user_id
, username, first_name, middle_name, last_name
( $\sigma$ karya_id=nilai_karyaid( $karya \bowtie_{author=user\_id}$ user))
```

**Gambar 2.7.** Bahasa Query Formal – *FindOneKarya*

Pada Query ini k dan u adalah alias untuk tabel karya dan user masing-masing, membuatnya lebih mudah dibaca.

JOIN digunakan untuk menggabungkan dua tabel berdasarkan kondisi yang diberikan. Di sini, menggabungkan tabel karya dengan tabel user menggunakan kondisi karya.author = user.user\_id.

WHERE digunakan untuk memfilter baris-baris yang sesuai dengan kriteria tertentu. Dalam hal ini, kita memilih baris di mana karya\_id sama dengan nilai dari variabel karya\_id

### 2.3.3. *FilterKarya*

#### 1. Filter hanya pemilik karya

```
SELECT k.karya_id, k.title, k.description, k.date,  
k.cover, k.tag, u.user_id, u.username, u.first_name,  
u.middle_name, u.last_name  
FROM karya as k  
JOIN user as u ON k.author = u.user_id  
WHERE k.author = 'nilai_user_id';
```

**Gambar 2.8.** Bahasa Query Terapan – *Filter hanya pemilik karya*

```
nkarya_id, title, description, date, cover, tag, use  
r_id, username, first_name, middle_name, last_name  
(okarya.author=nilai_user_id(karya&user))
```

**Gambar 2.9.** Bahasa Query Formal – *Filter hanya pemilik karya*

Pada Query ini k dan u adalah alias untuk tabel karya dan user masing-masing, membuatnya lebih mudah dibaca.

JOIN digunakan untuk menggabungkan dua tabel berdasarkan kondisi yang diberikan. Di sini, menggabungkan tabel karya dengan tabel user menggunakan kondisi karya.author = user.user\_id. WHERE digunakan untuk memfilter baris-baris di mana nilai karya.author sama dengan nilai user\_id.

#### 2. Filter untuk menampilkan semua karya

```
SELECT k.karya_id, k.title, k.description, k.date,  
k.cover, k.tag, u.user_id, u.username, u.first_name,  
u.middle_name, u.last_name  
FROM karya k  
JOIN user u ON k.author = u.user_id;
```

**Gambar 2.10.** Bahasa Query Terapan – *Filter untuk menampilkan semua karya*

```
πkarya_id, title, description, date, cover, tag, use  
r_id, username, first_name, middle_name, last_name  
(karya⋈user)
```

**Gambar 2.11.** Bahasa Query Formal – *Filter untuk menampilkan semua karya*

Pada Query ini k dan u adalah alias untuk tabel karya dan user masing-masing, membuatnya lebih mudah dibaca.

JOIN digunakan untuk menggabungkan dua tabel berdasarkan kondisi yang diberikan. Di sini, menggabungkan tabel karya dengan tabel user menggunakan kondisi karya.author = user.user\_id

### 3. Filter untuk menampilkan salah satu tag

```
SELECT k.karya_id, k.title, k.description, k.date,  
k.cover, k.tag, u.user_id, u.username, u.first_name,  
u.middle_name, u.last_name  
FROM karya as k  
JOIN user as u ON k.author = u.user_id  
WHERE k.tag = 'nilai_tag';
```

**Gambar 2.12.** Bahasa Query Terapan – *Filter untuk menampilkan salah satu tag*

```
πkarya_id, title, description, date, cover, tag, use  
r_id, username, first_name, middle_name, last_name  
(σkarya.tag=nilai_tag(karya⋈user))
```

**Gambar 2.13.** Bahasa Query Formal – *Filter untuk menampilkan salah satu tag*

Pada Query ini k dan u adalah alias untuk tabel karya dan user masing-masing, membuatnya lebih mudah dibaca.

JOIN digunakan untuk menggabungkan dua tabel berdasarkan kondisi yang diberikan. Di sini, menggabungkan tabel karya dengan tabel user menggunakan kondisi karya.author = user.user\_id. WHERE digunakan untuk memfilter baris-baris di mana nilai karya.tag sama dengan nilai tag.

#### 2.3.4. *GetProfile*

```
SELECT user_id, username, first_name, middle_name,  
last_name, email  
FROM user  
WHERE user_id = 'nilai_user_id';
```

**Gambar 2.14.** Bahasa Query Terapan – *GetProfile*

```
 $\pi$ user_id, username, first_name, middle_name, last_name,  
email( $\sigma$ user_id=nilai_user_id(user))
```

**Gambar 2.15.** Bahasa Query Formal – *GetProfile*

WHERE digunakan untuk memfilter baris di mana nilai user\_id sama dengan nilai user\_id yang diberikan.

## 2.4. Manipulasi Data

### 2.4.1. *Register*

```
INSERT INTO user (first_name, middle_name, last_name,  
password, email, username)  
VALUES ('nilai_first_name', 'nilai_middle_name',  
'nilai_last_name', 'nilai_password', 'nilai_email',  
'nilai_username');
```

**Gambar 2.16.** Manipulasi Data – *Register*

Di sini, user adalah nama tabel yang akan diisi, dan kolom-kolom yang disebutkan di dalam tanda kurung adalah kolom-kolom yang akan diisi.

### 2.4.2. *UpdateKarya*

```
UPDATE karya  
SET title = 'nilai_title', cover = 'nilai_cover',  
description = 'nilai_description', tag = 'nilai_tag'  
WHERE karya_id = 'nilai_karyaid';
```

**Gambar 2.17.** Manipulasi Data – *UpdateKarya*

Di sini, karya adalah nama tabel yang akan diperbarui, dan kolom-kolom yang akan diubah diikuti oleh nilai-nilai yang baru.

### 2.4.3. *DeleteKarya*

```
DELETE FROM karya WHERE karya_id = 'nilai_karyaid';
```

**Gambar 2.18.** Manipulasi Data – *DeleteKarya*

Di sini, karya adalah nama tabel yang akan dihapus dari, dan pengguna menentukan kondisi bahwa hanya baris dengan nilai karya\_id yang sama dengan 'nilai\_karyaid' yang akan dihapus.

#### 2.4.4. *TambahKarya*

```
INSERT INTO karya (title, cover, description, tag,  
author)  
VALUES ('nilai_title', 'nilai_cover',  
'nilai_description', 'nilai_tag', 'nilai_author');
```

**Gambar 2.19.** Manipulasi Data – *TambahKarya*

Di sini, karya adalah nama tabel yang akan diisi, dan kolom-kolom yang disebutkan di dalam tanda kurung adalah kolom-kolom yang akan diisi.

#### 2.4.5. *UpdateProfile*

```
INSERT INTO karya (title, cover, description, tag,  
author)  
VALUES ('nilai_title', 'nilai_cover',  
'nilai_description', 'nilai_tag', 'nilai_author');
```

**Gambar 2.20.** Manipulasi Data – *UpdateProfile*

Di sini, user adalah nama tabel yang akan diperbarui, dan kolom-kolom yang akan diubah diikuti oleh nilai-nilai yang baru.

### 2.5. Optimasi Query

Percobaan optimasi dilakukan pada tabel user yang diuji coba dengan 1juta baris data. Berikut ini persiapan untuk memasukkan 1juta data ke tabel user.

```

DELIMITER //

CREATE PROCEDURE InsertSampleUsers()
BEGIN
    DECLARE i INT DEFAULT 1;

    WHILE i <= 1000000 DO
        INSERT INTO user (username, password, first_name,
last_name, middle_name, email)
        VALUES
            (CONCAT('user', i), CONCAT('pass', i), CONCAT('FirstName',
i), CONCAT('LastName', i), CONCAT('M', i), CONCAT('user', i,
'@example.com'));

        SET i = i + 1;
    END WHILE;
END //

DELIMITER ;

```

**Gambar 2.21.** *Generator Dummy Data Function*

```

pakar> CALL InsertSampleUsers()
[2023-12-15 06:33:06] 1,000,000 rows affected in 12 m 56 s 756 ms

```

**Gambar 2.22.** Pemanggilan Fungsi Generator

```

MariaDB [pakar]> select count(*) from user;
+-----+
| count(*) |
+-----+
| 1000000 |
+-----+
1 row in set (0.428 sec)

```

**Gambar 2.23.** Menguji jumlah data

```

MariaDB [pakar]> explain select count(*) from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | user | index | NULL | username | 766 | NULL | 995747 | Using index |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)

```

**Gambar 2.24.** *Execution Plan*

Pada **Gambar 2.24.** Menunjukkan tipe EP yang dihasilkan secara default pada MySQL Penulis adalah INDEX. Ini berarti MySQL tidak melakukan *full load* data pada saat membaca sebuah data, namun berbanding terbalik dengan proses penulisan yang mana membuat proses ini jadi lebih lama.

## 2.6. Implementasi Sistem

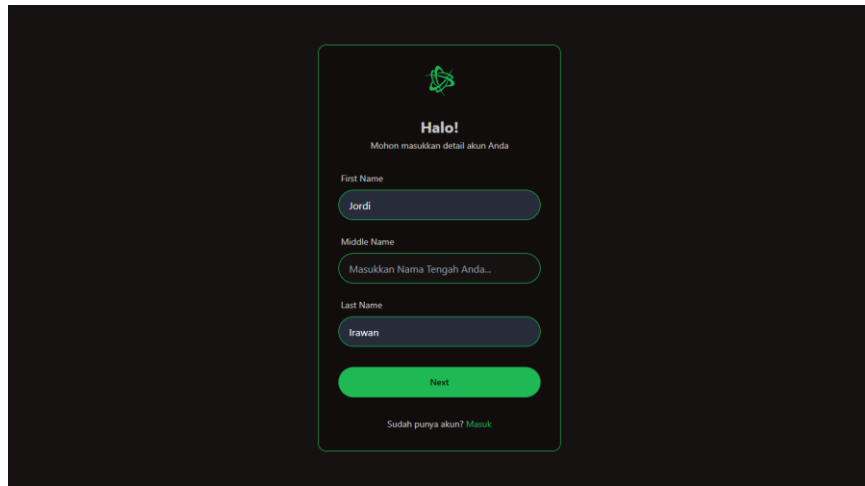


Database ini diimplementasikan ke dalam aplikasi Pakar berbasis web dengan bantuan NextJS sebagai *Layouting Website* dan Prisma sebagai *connector* ke database MySQL yang disambung ke NextJS

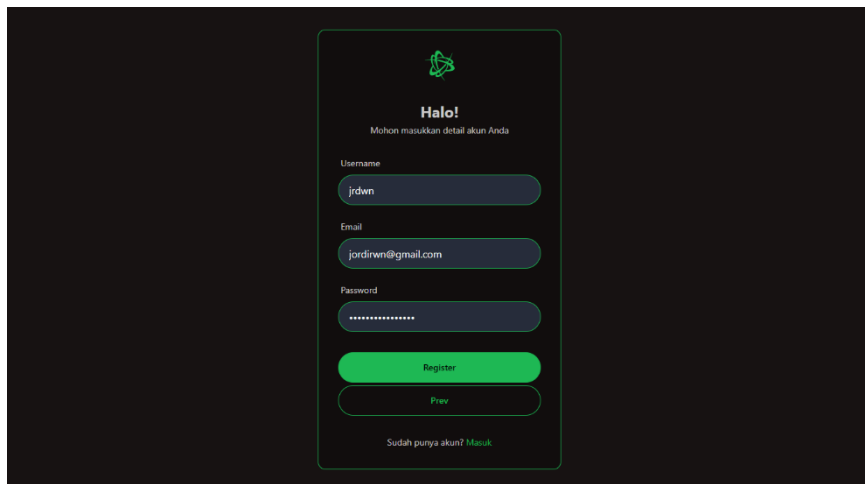
### 2.6.1. Register

```
1 import { NextResponse } from "next/server";
2 import prisma from "../../lib/prisma";
3
4 export async function POST(request) {
5   const url = request.nextUrl.clone();
6   url.pathname = "/login";
7   //prettier-ignore
8   const { first_name, middle_name, last_name, password, email, username } = await request.json();
9   try {
10    // register user
11    await prisma.$executeRaw`
12      INSERT INTO user (first_name, middle_name, last_name, password, email, username)
13      VALUES (${first_name}, ${middle_name}, ${last_name}, ${password}, ${email}, ${username})`;
14    return NextResponse.redirect(url);
15  } catch (error) {
16    console.log(error);
17    return NextResponse.json({ success: false }, { status: 400 });
18  }
19 }
```

Gambar 2.25. POST /register



Gambar 2.26. Register User tahap 1



### Gambar 2.27. Register User tahap 2

Pada Tahap Register Pengguna akan mengisi data Nama, Username, Email, dan Password yang nantinya dikirim ke server untuk dilakukan pengecekan terhadap data tersebut apakah sesuai dengan format atau tidak *duplicate* dengan data seseorang yang telah dibuat sebelumnya untuk kemudian disimpan ke database MySQL.

Jika proses ini berhasil pengguna akan diarahkan langsung ke halaman Login.

#### 2.6.2. Login

```
1 import { NextResponse } from "next/server";
2 import prisma from "../../lib/prisma";
3
4 export async function POST(request) {
5   const url = request.nextUrl.clone();
6   url.pathname = "/explore";
7   //prettier-ignore
8   const { password, email } = await request.json();
9   try {
10    // login a user
11    const user =
12      await prisma.$queryRaw `SELECT user_id FROM user WHERE email = ${email} AND password = ${password}`;
13    if (user.length) {
14      return NextResponse.redirect(url, {
15        headers: { "Set-Cookie": `user_id=${user[0].user_id};path=/` },
16      });
17    }
18    return NextResponse.json({ success: false }, { status: 401 });
19  } catch (error) {
20    return NextResponse.json({ success: false }, { status: 401 });
21  }
22 }
```

### Gambar 2.28. GET /login

### Gambar 2.29. Halaman Login

Pada Halaman Login pengguna hanya perlu memasukkan Email dan Password yang nantinya akan dilakukan pengecekan kembali di database terkait kebenaran data tersebut.

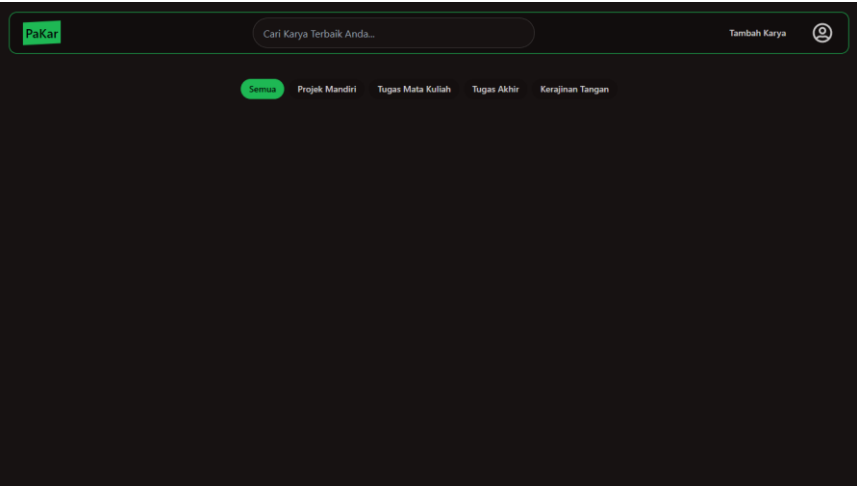
#### 2.6.3. Explore

```

1 import { NextResponse } from "next/server";
2 import { cookies } from "next/headers";
3 import prisma from "../../lib/prisma";
4
5 export async function GET(request) {
6   const tag = request.nextUrl.searchParams.get("tag");
7   try {
8     if (tag === "me") {
9       const user_id = cookies().get("user_id").value;
10      const karya = await prisma.$queryRaw
11        SELECT karya_id, title, description, date, cover, tag, user_id, username, first_name, middle_name, last_name
12      FROM karya
13      JOIN user ON karya.author = user.user_id
14      WHERE karya.author = ${user_id};
15      return NextResponse.json(karya);
16    } else if (
17      tag === "" ||
18      tag === "all" ||
19      tag === "null" ||
20      tag === "other"
21    ) {
22      const karya = await prisma.$queryRaw
23        SELECT karya_id, title, description, date, cover, tag, user_id, username, first_name, middle_name, last_name
24      FROM karya
25      JOIN user ON karya.author = user.user_id;
26      return NextResponse.json(karya);
27    } else {
28      const karya = await prisma.$queryRaw
29        SELECT karya_id, title, description, date, cover, tag, user_id, username, first_name, middle_name, last_name
30      FROM karya
31      JOIN user ON karya.author = user.user_id
32      WHERE karya.tag = ${tag};
33      return NextResponse.json(karya);
34    }
35  } catch (error) {
36    console.log(error);
37    return NextResponse.json({ success: false }, { status: 400 });
38  }
39 }

```

**Gambar 2.30.** GET /karya



**Gambar 2.31.** Halaman Explore Belum Ada Data

Karena Data karya belum ada maka dilakukan penambahan beberapa data yang bisa dilakukan pada tombol yang berada di pojok kanan atas.

**Menambahkan Karya Baru!**  
Mohon masukkan detail karya Anda

Judul  
Projek Basis Data 1

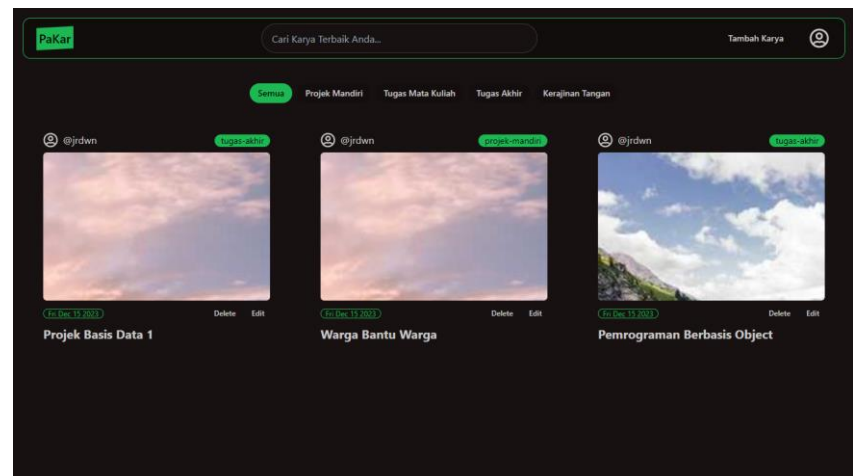
Link Cover  
<https://picsum.photos/seed/picum/200>

Tag  
Tugas Akhir

Deskripsi  
Hello World

Simpan

**Gambar 2.32.** Proses Tambah Data



**Gambar 2.33.** Setelah Proses Tambah Data

### **BAB III**

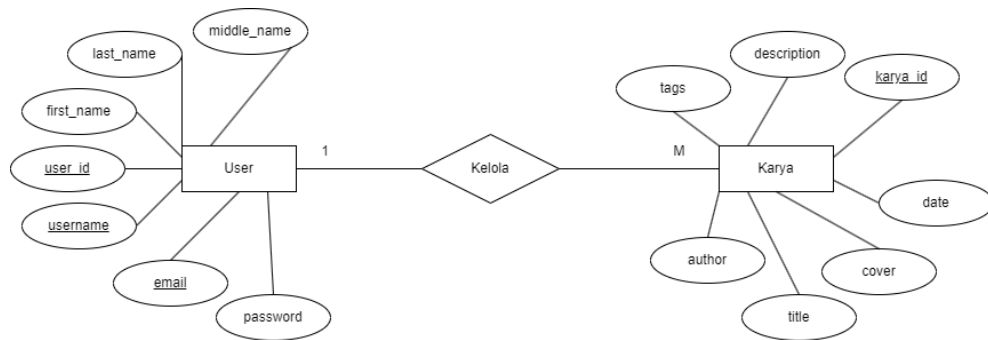
### **KESIMPULAN**

Melalui integrasi yang baik antara bahasa query, manipulasi data, dan optimasi query, implementasi sistem dapat menciptakan pengalaman pengguna yang lebih efisien dan responsif. Selain itu, penerapan prinsip-prinsip desain database yang baik memberikan dasar yang kuat untuk keandalan dan konsistensi data dalam aplikasi.

## DAFTAR PUSTAKA

- Dosen Teknik Informatika. 2023. Modul Basis Data I. Palangka Raya. Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya.
- Tavares, O. M. I., Rangkoly, S. M., Bawan, S. B. D., Ahmad, K. B., Utami, E., & Mustafa, M. S. (2021). Analysis of Query Restructurization Models in Optimizing Data Execution Response Time on Mysql Php 7.2. 27 Relational Databases. *J-ICON: Jurnal Komputer dan Informatika*, 9(1), 1-10.
- Rizaldi, A., Dewi, N. P., Rizal, M., & Alamsyah, M. F. R. (2021). Web-based Exhibition for Students' Works in Architecture Department. *JISAMAR (Journal of Information System, Applied, Management, Accounting and Research)*, 5(4), 975-986.
- Pratama, A. (2013). Pengenalan MySQL. *Diambil Dari*.
- Faradilla A. (2023). Apa Itu Query? Pengertian Query Database dan Contohnya. Hostinger. <https://www.hostinger.co.id/tutorial/apa-itu-query> (Diakses pada tanggal 14 Desember 2023).
- Andy Maulana Yusuf. (2019) Apa Sih Query Lanjut Itu?. <https://fit.labs.telkomuniversity.ac.id/apa-sih-query-lanjut-itu-penasaran-yuk-kita-lihat/> (Diakses pada tanggal 14 Desember 2023).
- (n.d.). Tips Optimasi Query untuk Mempercepat Pengaksesan Data pada Oracle Database. <https://i-3.co.id/8-tips-optimasi-query-pada-oracle-database/> (Diakses pada tanggal 14 Desember 2023).
- Raden Budiarto Hadiprakoso. (2021). Sistem Basis Data: Perancangan dan Implementasi.

## LAMPIRAN



**Gambar 2.1.** Entity Relation Diagram

```

CREATE TABLE user
(
    user_id          INT(11)          NOT NULL AUTO_INCREMENT
PRIMARY KEY,
    username         VARCHAR(191)     NOT NULL UNIQUE,
    password         VARCHAR(191)     NOT NULL,
    first_name       VARCHAR(191)     NOT NULL,
    last_name        VARCHAR(191),
    middle_name      VARCHAR(191),
    email            VARCHAR(191)     NOT NULL UNIQUE,
);
  
```

**Gambar 2.2.** Skema Tabel User

```

CREATE TABLE karya (
    karya_id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    tag VARCHAR(191) NOT NULL,
    description VARCHAR(191),
    title VARCHAR(191),
    date DATETIME(3) default current_timestamp() NOT NULL ,
    cover VARCHAR(191),
    author INT(11) NOT NULL,
    FOREIGN KEY (author) REFERENCES user(user_id)
);
  
```

**Gambar 2.3.** Skema Tabel Karya

```

SELECT user_id FROM user
WHERE email = 'email' AND password = 'passsword';
  
```

**Gambar 2.4.** Bahasa Query Terapan – Login

```

 $\pi$  user_id ( $\sigma$  email='email'  $\wedge$  password='password'(user))
  
```

**Gambar 2.5.** Bahasa Query Formal – Login

```
SELECT k.karya_id, k.title, k.description, k.date,
k.cover, k.tag, u.user_id, u.username, u.first_name,
u.middle_name, u.last_name
FROM karya as k
JOIN user as u ON k.author = u.user_id
WHERE k.karya_id = 'nilai_karyaid';
```

**Gambar 2.6.** Bahasa Query Terapan – *FindOneKarya*

```
 $\pi$ karya_id, title, description, date, cover, tag, user_id
, username, first_name, middle_name, last_name
( $\sigma$ karya_id=nilai_karyaid( $karya \bowtie author=user_id$ user))
```

**Gambar 2.7.** Bahasa Query Formal – *FindOneKarya*

```
SELECT k.karya_id, k.title, k.description, k.date,
k.cover, k.tag, u.user_id, u.username, u.first_name,
u.middle_name, u.last_name
FROM karya as k
JOIN user as u ON k.author = u.user_id
WHERE k.author = 'nilai_user_id';
```

**Gambar 2.8.** Bahasa Query Terapan – *Filter hanya pemilik karya*

```
 $\pi$ karya_id, title, description, date, cover, tag, use
r_id, username, first_name, middle_name, last_name
( $\sigma$ karya.author=nilai_user_id( $karya \bowtie user$ ))
```

**Gambar 2.9.** Bahasa Query Formal – *Filter hanya pemilik karya*

```
SELECT k.karya_id, k.title, k.description, k.date,
k.cover, k.tag, u.user_id, u.username, u.first_name,
u.middle_name, u.last_name
FROM karya k
JOIN user u ON k.author = u.user_id;
```

**Gambar 2.10.** Bahasa Query Terapan – *Filter untuk menampilkan semua karya*

```
 $\pi$ karya_id, title, description, date, cover, tag, use
r_id, username, first_name, middle_name, last_name
( $karya \bowtie user$ )
```

**Gambar 2.11.** Bahasa Query Formal – *Filter untuk menampilkan semua karya*



```
SELECT k.karya_id, k.title, k.description, k.date,
k.cover, k.tag, u.user_id, u.username, u.first_name,
u.middle_name, u.last_name
FROM karya as k
JOIN user as u ON k.author = u.user_id
WHERE k.tag = 'nilai_tag';
```

**Gambar 2.12.** Bahasa Query Terapan – *Filter untuk menampilkan salah satu tag*

```
 $\pi_{karya\_id, title, description, date, cover, tag, user\_id, username, first\_name, middle\_name, last\_name}(\sigma_{karya.tag=nilai\_tag}(karya \bowtie user))$ 
```

**Gambar 2.13.** Bahasa Query Formal – *Filter untuk menampilkan salah satu tag*

```
SELECT user_id, username, first_name, middle_name,
last_name, email
FROM user
WHERE user_id = 'nilai_user_id';
```

**Gambar 2.14.** Bahasa Query Terapan – *GetProfile*

```
 $\pi_{user\_id, username, first\_name, middle\_name, last\_name, email}(\sigma_{user\_id=nilai\_user\_id}(user))$ 
```

**Gambar 2.15.** Bahasa Query Formal – *GetProfile*

```
INSERT INTO user (first_name, middle_name, last_name,
password, email, username)
VALUES ('nilai_first_name', 'nilai_middle_name',
'nilai_last_name', 'nilai_password', 'nilai_email',
'nilai_username');
```

**Gambar 2.16.** Manipulasi Data – *Register*

```
UPDATE karya
SET title = 'nilai_title', cover = 'nilai_cover',
description = 'nilai_description', tag = 'nilai_tag'
WHERE karya_id = 'nilai_karyaid';
```

**Gambar 2.17.** Manipulasi Data – *UpdateKarya*

```
DELETE FROM karya WHERE karya_id = 'nilai_karyaid';
```

**Gambar 2.18.** Manipulasi Data – *DeleteKarya*

```
INSERT INTO karya (title, cover, description, tag,
author)
VALUES ('nilai_title', 'nilai_cover',
'nilai_description', 'nilai_tag', 'nilai_author');
```

**Gambar 2.19.** Manipulasi Data – *TambahKarya*

```
INSERT INTO karya (title, cover, description, tag,
author)
VALUES ('nilai_title', 'nilai_cover',
'nilai_description', 'nilai_tag', 'nilai_author');
```

**Gambar 2.20.** Manipulasi Data – *UpdateProfile*

```
DELIMITER //

CREATE PROCEDURE InsertSampleUsers()
BEGIN
    DECLARE i INT DEFAULT 1;

    WHILE i <= 1000000 DO
        INSERT INTO user (username, password, first_name,
last_name, middle_name, email)
        VALUES
            (CONCAT('user', i), CONCAT('pass', i), CONCAT('FirstName',
i), CONCAT('LastName', i), CONCAT('M', i), CONCAT('user', i,
'@example.com'));

        SET i = i + 1;
    END WHILE;
END //

DELIMITER ;
```

**Gambar 2.21.** *Generator Dummy Data Function*

```
pakar> CALL InsertSampleUsers()
[2023-12-15 06:33:06] 1,000,000 rows affected in 12 m 56 s 756 ms
```

**Gambar 2.22.** Pemanggilan Fungsi Generator

```
MariaDB [pakar]> select count(*) from user;
+-----+
| count(*) |
+-----+
| 1000000 |
+-----+
1 row in set (0.428 sec)
```

**Gambar 2.23.** Menguji jumlah data

```
MariaDB [pakar]> explain select count(*) from user;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | user | index | NULL | username | 766 | NULL | 995747 | Using index |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)
```

**Gambar 2.24.** *Execution Plan*

```

1 import { NextResponse } from "next/server";
2 import prisma from "../../lib/prisma";
3
4 export async function POST(request) {
5   const url = request.nextUrl.clone();
6   url.pathname = "/login";
7   //prettier-ignore
8   const { first_name, middle_name, last_name, password, email, username } = await request.json();
9   try {
10    // register user
11    await prisma.$executeRaw`
12      INSERT INTO user (first_name, middle_name, last_name, password, email, username)
13      VALUES (${first_name}, ${middle_name}, ${last_name}, ${password}, ${email}, ${username})`;
14    return NextResponse.redirect(url);
15  } catch (error) {
16    console.log(error);
17    return NextResponse.json({ success: false }, { status: 400 });
18  }
19 }

```

**Gambar 2.25.** POST /register

**Halo!**  
Mohon masukkan detail akun Anda

First Name  
Jordi

Middle Name  
Masukkan Nama Tengah Anda...

Last Name  
Irawan

Next

Sudah punya akun? [Masuk](#)

**Gambar 2.26.** Register User tahap 1

**Halo!**  
Mohon masukkan detail akun Anda

Username  
jrdwn

Email  
jordirwn@gmail.com

Password  
\*\*\*\*\*

Register

Prev

Sudah punya akun? [Masuk](#)

**Gambar 2.27.** Register User tahap 2

```

1 import { NextResponse } from "next/server";
2 import prisma from "../../lib/prisma";
3
4 export async function POST(request) {
5   const url = request.nextUrl.clone();
6   url.pathname = "/explore";
7   //prettier-ignore
8   const { password, email } = await request.json();
9   try {
10    // login a user
11    const user =
12      await prisma.$queryRaw `SELECT user_id FROM user WHERE email = ${email} AND password = ${password}`;
13    if (user.length) {
14      return NextResponse.redirect(url, {
15        headers: { "Set-Cookie": `user_id=${user[0].user_id};path=/` },
16      });
17    }
18    return NextResponse.json({ success: false }, { status: 401 });
19  } catch (error) {
20    return NextResponse.json({ success: false }, { status: 401 });
21  }
22 }

```

Gambar 2.26. GET /login

Selamat Datang Kembali!

Mohon masukkan detail akun Anda

Email

jordrwn@gmail.com

Password

\*\*\*\*\*

☒ Remember me [Forgot Password?](#)

Login

[Belum punya akun? Daftar](#)

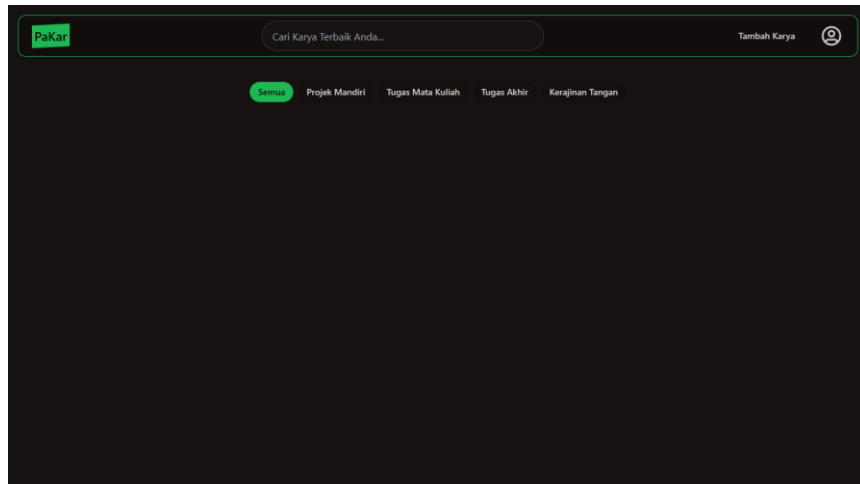
Gambar 2.27. Halaman Login

```

1 import { NextResponse } from "next/server";
2 import { cookies } from "next/headers";
3 import prisma from "../../lib/prisma";
4
5 export async function GET(request) {
6   const tag = request.nextUrl.searchParams.get("tag");
7   try {
8     if (tag === "me") {
9       const user_id = cookies().get("user_id").value;
10      const karya = await prisma.$queryRaw `
11        SELECT karya_id, title, description, date, cover, tag, user_id, username, first_name, middle_name, last_name
12        FROM karya
13        JOIN user ON karya.author = user.user_id
14        WHERE karya.author = ${user_id}`;
15      return NextResponse.json(karya);
16    } else if (
17      tag === "" ||
18      tag === "all" ||
19      tag === "null" ||
20      tag === "other"
21    ) {
22      const karya = await prisma.$queryRaw `
23        SELECT karya_id, title, description, date, cover, tag, user_id, username, first_name, middle_name, last_name
24        FROM karya
25        JOIN user ON karya.author = user.user_id`;
26      return NextResponse.json(karya);
27    } else {
28      const karya = await prisma.$queryRaw `
29        SELECT karya_id, title, description, date, cover, tag, user_id, username, first_name, middle_name, last_name
30        FROM karya
31        JOIN user ON karya.author = user.user_id
32        WHERE karya.tag = ${tag}`;
33      return NextResponse.json(karya);
34    }
35  } catch (error) {
36    console.log(error);
37    return NextResponse.json({ success: false }, { status: 400 });
38  }
39 }

```

Gambar 2.28. GET /karya



**Gambar 2.29.** Halaman Explore Belum Ada Data