



STUDI KASUS PROYEK BASIS DATA KELOMPOK 4

PERANCANGAN DAN IMPLEMENTASI BASIS DATA SISTEM APLIKASI "REMILIS"

Kelompok 4

- Muhammad Luthfi Taufikurrahman (2022071053)
- Arellia Agustin (2022071060)
- Indra Setiawan (2022071067)
- Muhammad Faris Hafizh (2022071068)

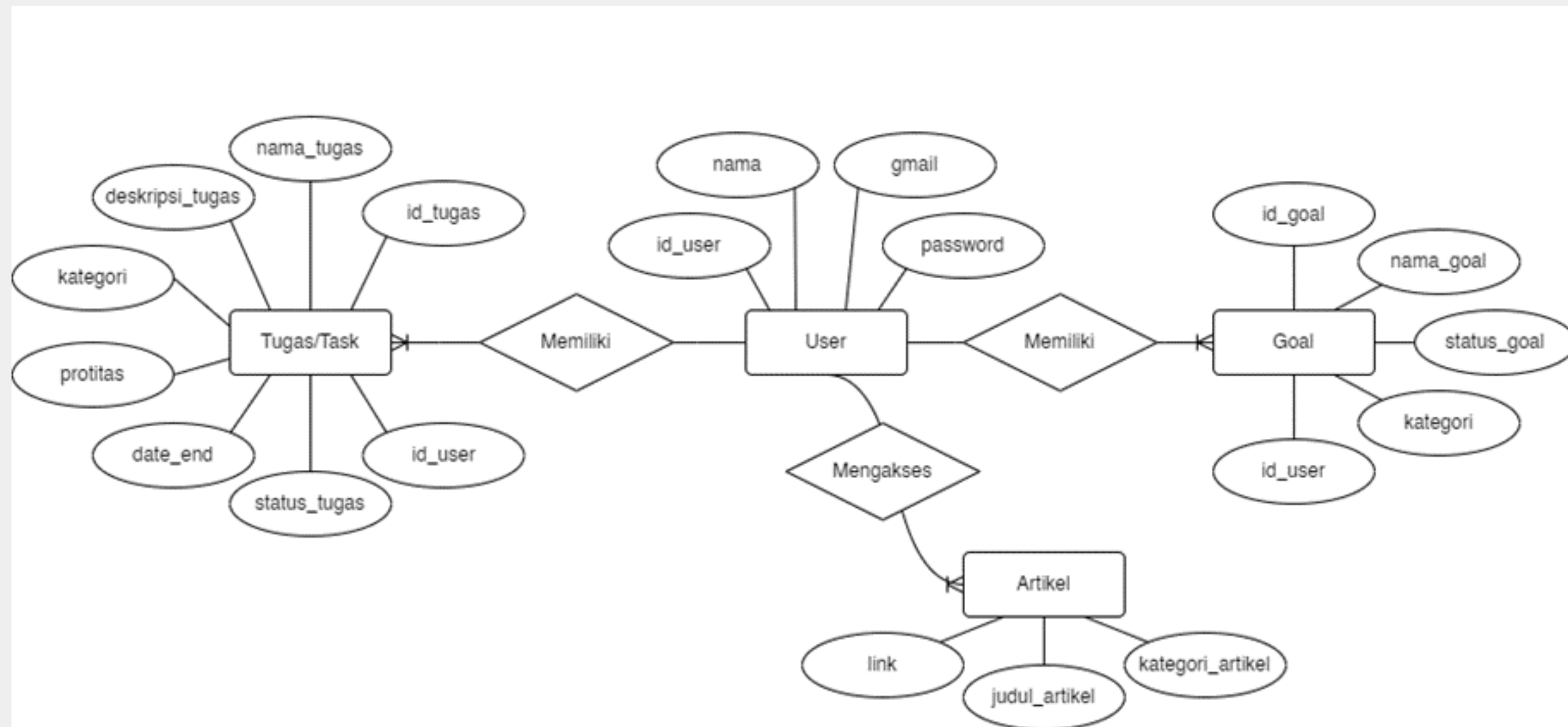
PENDAHULUAN

Aplikasi "Remilis" merupakan sebuah solusi manajemen tugas dan tujuan yang dirancang untuk membantu pengguna dalam mengatur aktivitas harian mereka secara efektif. Aplikasi ini menyediakan berbagai fitur, termasuk pengelolaan tugas, pencapaian tujuan, penyajian artikel terkait kesehatan mental dan resep makanan, serta fitur login dan registrasi pengguna. Studi kasus ini bertujuan untuk merancang dan mengimplementasikan basis data relasional menggunakan MySQL yang mendukung fungsi-fungsi inti aplikasi "Remilis".

Dokumen ini berisi laporan lengkap untuk pembuatan database, tabel dengan constraint, operasi CRUD (Create, Read, Update, Delete), stored procedure, trigger, view, serta teknik normalisasi dan optimasi query. Dengan rancangan ini, diharapkan basis data dapat mendukung operasional aplikasi "Remilis" dengan efisien dan aman.

Rancangan Basis Data

- Terdiri dari empat tabel utama: user, tugas, goal, dan artikel.
- Diagram ERD (Entity-Relationship Diagram)





MEMBUAT DATABASE

```
CREATE DATABASE project_db;  
USE project_db;
```



TABEL USER

```
CREATE TABLE project_db.user (  
  id_user VARCHAR(5) PRIMARY KEY,  
  nama VARCHAR(32),  
  gmail VARCHAR(255),  
  sandi VARCHAR(32),  
  status ENUM("active", "not_active"),  
  CONSTRAINT chck_status CHECK (status = "active" OR status = "not_active")  
);
```

TABEL TUGAS

```
CREATE TABLE project_db.tugas (  
  id_tugas VARCHAR(5) PRIMARY KEY,  
  nama_tugas VARCHAR(32),  
  deskripsi_tugas TEXT,  
  kategori VARCHAR(32),  
  prioritas ENUM("high", "mid", "low"),  
  date_end DATETIME,  
  status_tugas ENUM("done", "not done"),  
  user_id VARCHAR(5),  
  FOREIGN KEY (user_id) REFERENCES project_db.user(id_user),  
  CONSTRAINT chck_priority CHECK (prioritas = 'high' OR prioritas = 'mid' OR prioritas = 'low')  
);
```

TABEL GOAL

```
CREATE TABLE project_db.goal (  
  id_goal VARCHAR(5) PRIMARY KEY,  
  nama_goal VARCHAR(32),  
  status_goal ENUM("completed", "uncompleted"),  
  kategori VARCHAR(32),  
  id_user VARCHAR(5),  
  FOREIGN KEY (id_user) REFERENCES project_db.user(id_user),  
  CONSTRAINT chck_goal_status CHECK (status_goal = 'completed' OR status_goal = 'uncompleted')  
);
```

Tabel Artikel

```
CREATE TABLE project_db.artikel (  
    judul_artikel VARCHAR(255) PRIMARY KEY,  
    kategori_artikel VARCHAR(32),  
    link VARCHAR(32)  
);
```


Stored Procedure

```
-- Update login status
CREATE PROCEDURE sp_update_login(
    IN in_gmail VARCHAR(255),
    IN in_sandi VARCHAR(32)
)
BEGIN
    UPDATE project_db.user SET status = 'active' WHERE gmail = in_gmail AND sandi = in_sandi;
END //

-- Validate login
CREATE PROCEDURE sp_validate_login(
    IN in_gmail VARCHAR(255),
    IN in_sandi VARCHAR(32)
)
BEGIN
    DECLARE user_count INT;
    SELECT COUNT(*) INTO user_count FROM project_db.user WHERE gmail = in_gmail AND sandi = in_sandi;
    IF user_count > 0 THEN
        CALL sp_update_login(in_gmail, in_sandi);
    END IF;
END //

)
```

```
-- Logout
CREATE PROCEDURE sp_logout(
    IN in_id VARCHAR(5)
)
BEGIN
    UPDATE project_db.user SET status = 'not_active' WHERE id_user = in_id;
END //

-- Validate registration
CREATE PROCEDURE sp_validate_register(
    IN in_id VARCHAR(5),
    IN in_nama VARCHAR(32),
    IN in_gmail VARCHAR(255),
    IN in_sandi VARCHAR(32),
    IN in_status ENUM("not_active")
)
```

Stored Procedure

```
BEGIN
  DECLARE user_count INT;
  SELECT COUNT(*) INTO user_count FROM project_db.user WHERE gmail = in_gmail;
  IF user_count = 0 THEN
    CALL sp_tambah_user(in_id, in_nama, in_gmail, in_sandi, in_status);
  ELSE
    SELECT 'error gmail have been used';
  END IF;
END //

-- Add user
CREATE PROCEDURE sp_tambah_user(
  IN in_id VARCHAR(5),
  IN in_nama VARCHAR(32),
  IN in_gmail VARCHAR(255),
  IN in_sandi VARCHAR(32),
  IN in_status ENUM("not_active")
)
```

```
BEGIN
  INSERT INTO project_db.user(id_user, nama, gmail, sandi, status)
  VALUES (in_id, in_nama, in_gmail, in_sandi, in_status);
END //

-- Update password
CREATE PROCEDURE sp_update_password(
  IN in_gmail VARCHAR(255),
  IN in_sandi VARCHAR(32)
)
BEGIN
  UPDATE project_db.user SET sandi = in_sandi WHERE gmail = in_gmail;
END //

DELIMITER ;
```

Trigger

```
DELIMITER //  
CREATE TRIGGER after_task_done  
AFTER UPDATE ON project_db.tugas  
FOR EACH ROW  
BEGIN  
    DELETE FROM project_db.tugas WHERE status_tugas = 'done';  
END //  
DELIMITER ;
```

View

View untuk menyajikan data tugas dan goal berdasarkan pengguna yang aktif.

```
CREATE VIEW tugas_view AS  
SELECT nama_tugas, deskripsi_tugas, kategori, prioritas, date_end, status_tugas  
FROM project_db.tugas  
WHERE user_id = (SELECT id_user FROM project_db.user WHERE status = 'active');
```

```
CREATE VIEW goal_view AS  
SELECT nama_goal, status_goal, kategori  
FROM project_db.goal  
WHERE id_user = (SELECT id_user FROM project_db.user WHERE status = 'active');
```

NORMALISASI DAN INDEXING

- **Basis data telah dinormalisasi hingga bentuk normal ketiga (3NF).**
- **Contoh indexing:**

```
CREATE index idx_user on user(id_user);
```

```
CREATE index idx_tugas on tugas(id_tugas);
```

```
CREATE index idx_goal on goal(id_goal);
```

```
CREATE index idx_artikel on artikel(judul_artikel);
```

PENYAJIAN LAPORAN

Contoh query untuk menyajikan laporan jumlah tugas per pengguna.

```
SELECT u.nama, COUNT(t.id_tugas) AS total_tugas  
FROM project_db.user u  
JOIN project_db.tugas t ON u.id_user = t.user_id  
GROUP BY u.nama;
```

TRANSACTIONAL CONTROL COMMAND

Contoh penggunaan perintah kontrol transaksi:

```
set autocommit = 0;  
start transaction;  
call sp_tambah_user("A0001","Ibu Maya","maya@gmail.com","123",1);  
call sp_validate_login("maya@gmail.com","123");  
commit;
```



OPTIMASI QUERY

Penggunaan indexing pada kolom yang sering digunakan dalam klausa
WHERE untuk meningkatkan kinerja query.



SISTEM KEAMANAN BASIS DATA

Penggunaan user management dan pemberian hak akses.

```
create user 'user1'@'localhost' identified by "123";
```

```
grant insert,select,update on project_db.user to 'user1'@'localhost' with grant option;  
grant insert,select,update on project_db.tugas to 'user1'@'localhost' with grant option;  
grant insert,select,update on project_db.goal to 'user1'@'localhost' with grant option;
```



KESIMPULAN

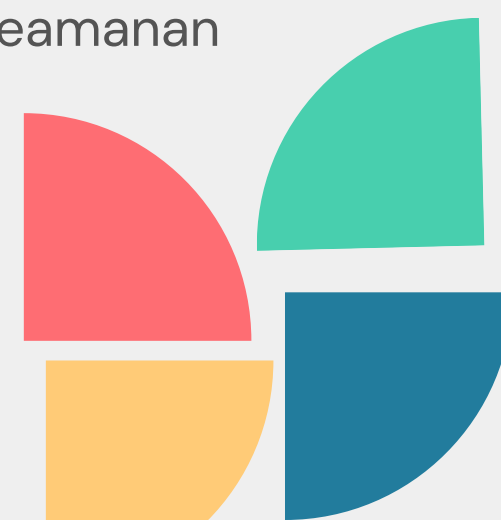

Proyek basis data "Remilis" yang dirancang dan diimplementasikan menggunakan MySQL ini mencakup seluruh aspek penting yang diperlukan untuk mendukung operasional aplikasi manajemen tugas dan tujuan secara efisien. Dengan mengimplementasikan teknik-teknik normalisasi hingga bentuk normal ketiga (3NF), penggunaan indexing, serta optimasi query, basis data ini dapat memastikan performa yang optimal dalam pengelolaan data.

Berbagai fitur telah berhasil diimplementasikan, termasuk pembuatan database, tabel dengan constraint, operasi CRUD (Create, Read, Update, Delete), stored procedure, trigger, view, dan teknik transactional control. Hal ini memungkinkan pengguna untuk melakukan berbagai operasi seperti login, registrasi, pengelolaan tugas, pencapaian tujuan, dan penyajian artikel dengan mudah dan aman.

Penyajian laporan yang tepat waktu dan akurat juga dapat dicapai melalui query-query yang telah dirancang, memastikan bahwa data dapat diakses dan dianalisis dengan cepat. Sistem keamanan basis data juga telah diperkuat dengan manajemen pengguna dan pemberian hak akses yang sesuai, sehingga data pengguna tetap terlindungi.

Dengan laporan ini, tim proyek telah menyediakan panduan yang komprehensif untuk merancang, mengimplementasikan, dan mengelola basis data relasional menggunakan MySQL. Implementasi ini diharapkan dapat mendukung operasional aplikasi "Remilis" dengan efisien dan aman, serta memberikan pengalaman pengguna yang lebih baik dalam mengatur aktivitas harian mereka.

Laporan ini mencakup seluruh aspek yang dibutuhkan untuk merancang, mengimplementasikan, dan mengelola basis data relasional menggunakan MySQL. Dengan menerapkan normalisasi, indexing, dan optimasi query, serta menjaga keamanan basis data, sistem ini diharapkan dapat berjalan dengan efisien dan aman.



The background features four decorative geometric patterns in the corners. Top-left: A series of parallel diagonal lines in a light blue-grey color, with a quarter-circle arc in the same color to its right. Top-right: A cluster of overlapping quarter-circles in yellow, red, teal, and dark blue. Bottom-left: A cluster of overlapping quarter-circles in red, teal, and dark blue. Bottom-right: A series of parallel diagonal lines in a light blue-grey color, with a large quarter-circle arc in the same color to its left.

IMPLEMENTASI QUERY

The background features four decorative geometric patterns in the corners. The top-left corner has a series of parallel diagonal lines. The top-right corner contains a cluster of overlapping semi-circles in yellow, red, teal, and blue. The bottom-left corner features a similar cluster of overlapping semi-circles in red, teal, blue, and red. The bottom-right corner has a large, faint semi-circle outline with several parallel diagonal lines inside it.

TERIMA KASIH