# LAPORAN

# PROJECT BASIS DATA LANJUT

Laporan ini dibuat guna memenuhi tugas Ujian Akhir Semester mata kuliah Basis Data Lanjut

Dosen Pengampu : Yuyun Umaidah S.Kom, M.Kom.


Disusun oleh:

Kelompok 3

| | |
|---|---|
| Ditha Alfariz | (2310631250046) |
| Aura Zahra Ramadhani | (2310631250007) |
| Firdha Nabila Subkhan | (2310631250016) |
| Gea Anindiya | (2310631250018) |
| Muhammad Fadllan Ziadh Ramadhan | (2010631250066) |

**PROGRAM STUDI SISTEM INFORMASI**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS SINGAPERBANGSA KARAWANG**


**2024**

# KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga laporan ini dapat diselesaikan tepat pada waktunya. Laporan ini disusun untuk memenuhi Ujian Akhir Semester mata kuliah Basis Data Lanjut yang kami tempuh pada semester 3 program studi Sistem Informasi.

Laporan ini membahas sebuah projek yang dimana kami menempatkan diri sebagai Software Engineering yang akan melakukan tugas membuat sebuah aplikasi sederhana untuk manajemen toko buku online dengan tujuan memberikan gambaran dan pemahaman yang lebih mendalam mengenai cara merancang skema database, mengimplementasikan skema tersebut dalam SQL, dan menulis beberapa kueri SQL untuk mengelola data sesuai dengan studi kasus. Kami telah berusaha menyusun laporan ini sebaik mungkin berdasarkan data yang telah diberikan dan pengetahuan yang telah diperoleh selama perkuliahan.

Dalam kesempatan ini, kami mengucapkan terima kasih kepada dosen pembimbing mata kuliah Yuyun Umaidah S.Kom, M.Kom, teman-teman mahasiswa, serta pihak-pihak lain yang telah memberikan bimbingan dan dukungan selama proses penyusunan laporan ini. Semoga laporan ini dapat memberikan manfaat dan menjadi tambahan wawasan bagi pembaca.

Kami menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang.

Akhir kata, kami berharap laporan ini dapat diterima dengan baik dan memberikan kontribusi positif dalam pengembangan ilmu pengetahuan di bidang Sistem Informasi.

Karawang, 24 November 2024

Penyusun

# DAFTAR ISI

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Dalam era digital, pengelolaan data secara manual sudah tidak relevan lagi bagi perusahaan yang ingin berkembang. Salah satu sektor yang membutuhkan digitalisasi adalah toko buku, di mana pengelolaan stok buku, data pelanggan, hingga proses pemesanan memerlukan sistem yang cepat, akurat, dan efisien. Sistem manajemen toko buku berbasis aplikasi mampu memberikan solusi modern dengan mengintegrasikan semua proses operasional dalam satu platform.

Dengan adanya aplikasi manajemen toko buku, toko dapat memantau stok buku secara real-time, mengelola pesanan pelanggan dengan lebih mudah, serta memastikan data pelanggan dan transaksi tersimpan secara aman. Hal ini juga membantu meningkatkan pengalaman pelanggan karena mereka dapat memperoleh layanan yang lebih cepat dan akurat. Oleh karena itu, pengembangan aplikasi sederhana untuk manajemen toko buku menjadi langkah awal dalam mendukung transformasi digital sektor ini.

Melalui pembuatan aplikasi ini, diharapkan toko buku dapat meminimalkan kesalahan manusia dalam pencatatan data, meningkatkan efisiensi operasional, dan memperluas jangkauan pasar dengan memanfaatkan teknologi berbasis database.

## 1.2 Tujuan

Adapun tujuan dari pembuatan laporan ini adalah:

1. Merancang Skema Database: Membuat desain skema database yang mencakup tabel-tabel untuk menyimpan informasi buku, pelanggan, pesanan, dan detail pesanan.
2. Mengimplementasikan Skema Database: Menuliskan dan menguji implementasi skema database menggunakan SQL untuk memastikan semua entitas saling terhubung dengan baik.

3. Mengelola Data dengan SQL: Mengembangkan kueri SQL untuk berbagai kebutuhan operasional, seperti menampilkan data, menambah pesanan baru, dan menghapus data lama.
4. Memastikan Transaksi Aman: Menyediakan skrip SQL untuk memastikan keandalan transaksi melalui mekanisme commit dan rollback.
5. Meningkatkan Performa Database: Mengidentifikasi dan menerapkan langkah-langkah optimasi untuk meningkatkan efisiensi kueri dan pengelolaan data.
6. Menjamin Keamanan Data: Merancang langkah-langkah keamanan untuk melindungi aplikasi dari serangan seperti SQL injection dan memastikan data tersimpan secara aman.
7. Memberikan Rekomendasi: Menyusun rekomendasi berbasis teori dan praktik terbaik untuk pengembangan dan pengelolaan aplikasi lebih lanjut.

# BAB II

# DASAR TEORI

## 2.1 Basis Data Relasional

Basis data relasional adalah model basis data yang mengorganisasi data ke dalam tabel-tabel (relasi) yang memiliki kolom (atribut) dan baris (tupel). Setiap tabel biasanya merepresentasikan sebuah entitas, seperti pelanggan, produk, atau transaksi, di mana setiap kolom menggambarkan atribut dari entitas tersebut, dan setiap baris merepresentasikan instansinya. Keunggulan utama basis data relasional adalah kemampuannya untuk menghubungkan data antar tabel melalui kunci primer (primary key) dan kunci asing (foreign key).

Konsep Utama:

- Tabel (Table): Struktur data berbentuk baris (record) dan kolom (field).
- Primary Key: Kolom yang secara unik mengidentifikasi setiap baris di dalam tabel. Misalnya, ID buku di tabel *Buku*.
- Foreign Key: Kolom yang digunakan untuk menghubungkan satu tabel dengan tabel lainnya. Misalnya, kolom Pelanggan_ID di tabel *Pesanan* mengacu pada ID pelanggan di tabel *Pelanggan*.

Relasi Antar Tabel:

- Satu ke Satu (1:1): Satu entitas di tabel A berhubungan dengan satu entitas di tabel B.
- Satu ke Banyak (1:N): Satu entitas di tabel A berhubungan dengan banyak entitas di tabel B (contoh: satu pelanggan memiliki banyak pesanan).
- Banyak ke Banyak (M:N): Banyak entitas di tabel A berhubungan dengan banyak entitas di tabel B, biasanya dijembatani oleh tabel relasi (contoh: *Pesanan* dan *Buku* dihubungkan melalui tabel *Detail_Pesanan*).

## 2.2 Desain Skema Database

Desain skema database adalah proses menyusun struktur basis data untuk memastikan bahwa data dapat diakses dan dikelola dengan efisien.

Prinsip Desain Skema:

- Normalisasi: Proses pengorganisasian data untuk mengurangi redundansi dan meningkatkan integritas. Ada tiga bentuk normal yang umum:
    - ➢ 1NF: Semua nilai di kolom harus atomic (tidak dapat dipecah lagi).
    - ➢ 2NF: Semua kolom non-kunci harus sepenuhnya bergantung pada primary key.
    - ➢ 3NF: Tidak ada ketergantungan transitif antara kolom non-kunci.
- Entity-Relationship Diagram (ERD):
  ERD digunakan untuk memvisualisasikan hubungan antar entitas dalam sistem. Misalnya:
    - ➢ Entitas: Buku, Pelanggan, Pesanan, Detail Pesanan.
    - ➢ Hubungan:
        - ➔ Pelanggan memiliki banyak Pesanan.
        - ➔ Pesanan memiliki banyak Buku (melalui tabel *Detail_Pesanan*).

Skema database dibagi menjadi 2, yaitu:

1. Skema Database Fisik

   Skema database fisik adalah representasi aktual dari bagaimana data disimpan di dalam perangkat penyimpanan, seperti hard drive atau SSD. Skema ini lebih bersifat teknis dan berfokus pada pengelolaan ruang penyimpanan, pengindeksan, partisi data, struktur file, serta pengaturan akses data. Berbeda dengan skema logis, skema fisik sangat bergantung pada perangkat lunak dan perangkat keras yang digunakan. Dalam skema fisik, desain mencakup:

   - Indeks: Digunakan untuk mempercepat akses data tertentu. Misalnya, membuat indeks pada kolom yang sering digunakan dalam pencarian.

- Partisi Data: Membagi tabel besar menjadi beberapa bagian yang lebih kecil berdasarkan kriteria tertentu, seperti rentang tanggal atau wilayah geografis, untuk meningkatkan efisiensi pengambilan data.
- Penyimpanan Data: Menentukan struktur file (seperti B-Tree atau Hashing) untuk menyimpan tabel dan indeks.
- Tuning Kinerja: Melibatkan pengaturan seperti pengelompokan tabel atau pengoptimalan cache untuk mempercepat proses akses data.

Skema fisik dibuat dengan mempertimbangkan kinerja, kapasitas penyimpanan, dan keamanan data. Misalnya, jika basis data diakses oleh banyak pengguna secara bersamaan, replikasi data dapat diterapkan untuk meningkatkan ketersediaan dan mengurangi beban server utama.

2. Skema Database Logis

Skema database logis adalah representasi konseptual dari struktur basis data yang berfokus pada cara data diorganisasi dan hubungan antar-data tanpa memperhatikan bagaimana data tersebut disimpan secara fisik. Skema ini menggambarkan tabel, kolom, tipe data, kunci primer, kunci asing, dan hubungan antar-entitas berdasarkan model data relasional. Skema logis bertujuan untuk menciptakan desain yang memenuhi kebutuhan pengguna dan aplikasi, sekaligus memastikan integritas dan efisiensi data.

Proses pengembangan skema logis biasanya dimulai dari model konseptual, seperti Entity-Relationship Diagram (ERD), yang kemudian diterjemahkan menjadi tabel-tabel dengan relasi tertentu. Dalam skema logis, konsep normalisasi sering diterapkan untuk mengurangi redundansi data dan mencegah inkonsistensi. Misalnya, tabel pelanggan dan tabel pesanan akan dihubungkan melalui kunci asing untuk menunjukkan hubungan antara pelanggan dan pesanan mereka.

Keunggulan skema logis adalah kemampuannya untuk menjembatani kebutuhan bisnis dengan implementasi teknis. Skema ini tidak bergantung pada

teknologi atau platform tertentu, sehingga mudah untuk dimodifikasi atau diadaptasi sesuai kebutuhan bisnis yang berubah.

3. Perbedaan Skema Database Fisik dan Logis

| Aspek | Skema Logis | Skema Fisik |
|---|---|---|
| Fokus | Struktur konseptual dan hubungan antar-data | Penyimpanan data secara teknis |
| Ketergantungan Platform | Tidak bergantung pada perangkat lunak atau hardware tertentu | Sangat bergantung pada perangkat lunak dan hardware |
| Elemen Utama | Tabel, kolom, tipe data, kunci, relasi | Indeks, partisi, struktur file, pengaturan penyimpanan |
| Tujuan | Memenuhi kebutuhan aplikasi dan pengguna | Mengoptimalkan kinerja dan penggunaan sumber daya |

## 2.3 Struktur Query Language (SQL)

Struktur Query Language (SQL) adalah bahasa standar yang digunakan untuk berinteraksi dengan basis data relasional. SQL memungkinkan pengguna untuk membuat, mengubah, dan menghapus struktur basis data serta melakukan manipulasi data melalui perintah-perintah seperti SELECT, INSERT, UPDATE, dan DELETE. SQL dibagi menjadi beberapa jenis:

- Data Definition Language (DDL): Perintah untuk membuat dan mengubah struktur basis data, seperti CREATE, ALTER, dan DROP.
- Data Manipulation Language (DML): Perintah untuk mengelola data dalam tabel, seperti INSERT, UPDATE, dan DELETE.
- Data Query Language (DQL): Perintah untuk mengambil data, seperti SELECT.
- Data Control Language (DCL): Perintah untuk mengontrol akses data, seperti GRANT dan REVOKE.

SQL memungkinkan penggunaan query kompleks, seperti penggabungan tabel dengan JOIN, penyaringan data menggunakan kondisi di WHERE, serta pengelompokan data dengan GROUP BY dan HAVING. Selain itu, SQL mendukung fungsi agregasi seperti SUM, COUNT, dan AVG untuk analisis data. Fleksibilitas dan kekuatannya menjadikan SQL sebagai alat utama dalam pengelolaan basis data relasional.

## 2.4 Optimasi Query dan Skalabilitas

Optimasi query adalah proses meningkatkan kinerja eksekusi query dengan meminimalkan sumber daya yang digunakan, seperti waktu dan memori. Hal ini sangat penting, terutama pada basis data dengan volume data yang besar atau sistem yang melayani banyak pengguna secara bersamaan. Optimasi query dilakukan melalui beberapa teknik:

1. Indeks

   Indeks berfungsi seperti daftar isi pada buku, memungkinkan DBMS untuk menemukan data tertentu dengan lebih cepat. Indeks dapat dibuat pada kolom yang sering digunakan dalam filter atau pengurutan (contoh: kolom yang muncul di klausa WHERE atau ORDER BY). Jenis indeks yang umum meliputi B-Tree, Hash Index, dan Full-Text Index.

2. Penggunaan Query Efisien

   Menulis query yang optimal juga berkontribusi pada kinerja. Contohnya, menghindari penggunaan wildcard (SELECT *) yang dapat memuat data berlebih, dan menggantinya dengan kolom spesifik yang dibutuhkan. Selain itu, query yang memanfaatkan subquery atau Common Table Expression (CTE) secara bijak dapat mengurangi beban kerja sistem.

   Statistik dan Rencana Eksekusi (Execution Plan): DBMS modern menyediakan alat untuk mempelajari rencana eksekusi query. Statistik seperti jumlah baris dalam tabel atau distribusi data digunakan untuk menentukan strategi terbaik dalam menjalankan query.

3. Materialized Views

   Materialized view adalah hasil dari query yang disimpan sebagai tabel fisik untuk mempercepat akses data yang sering digunakan. Namun, materialized view memerlukan mekanisme penyegaran untuk memastikan konsistensi data.

Skalabilitas adalah kemampuan sistem untuk menangani peningkatan beban kerja tanpa mengalami penurunan kinerja. Skalabilitas dibagi menjadi dua jenis:

1. Skalabilitas Vertikal

   Meningkatkan kapasitas server dengan menambah sumber daya, seperti CPU, RAM, atau penyimpanan. Meskipun lebih sederhana, skalabilitas vertikal memiliki keterbatasan fisik dan biaya.

2. Skalabilitas Horizontal

   Melibatkan penambahan server baru untuk mendistribusikan beban kerja. Contohnya adalah menggunakan teknik sharding, di mana data dibagi ke beberapa server berdasarkan kriteria tertentu, atau memanfaatkan replikasi untuk memastikan salinan data tersedia di beberapa lokasi.

   Kombinasi dari teknik optimasi query dan skalabilitas dapat digunakan untuk mencapai performa basis data yang optimal bahkan pada lingkungan dengan beban kerja yang sangat tinggi.

## 2.5 Teknologi dan Basis Data Lanjut

Teknologi yang digunakan dalam basis data terus berkembang untuk memenuhi kebutuhan pengelolaan data yang semakin kompleks. Berikut adalah beberapa teknologi terkini dalam Basis Data Lanjut:

- NoSQL **:** Basis data NoSQL seperti MongoDB, Cassandra, dan Redis dirancang untuk menangani data tidak terstruktur atau semi-terstruktur. NoSQL mendukung skalabilitas horizontal dengan baik dan sering digunakan dalam aplikasi dengan volume data yang besar dan kebutuhan akses data real-time.
- Data Warehouse dan Big Data**:** Data warehouse, seperti Amazon Redshift dan Google BigQuery, digunakan untuk menganalisis data dalam jumlah besar yang berasal dari berbagai sumber. Di sisi lain, teknologi big data seperti Apache Hadoop dan Apache Spark dirancang untuk memproses data skala petabyte atau lebih dalam waktu singkat.
- Cloud Database: Platform seperti Amazon RDS, Microsoft Azure SQL Database, dan Google Cloud Spanner menyediakan layanan basis data yang dikelola sepenuhnya di cloud. Keunggulan utama cloud database adalah skalabilitas yang hampir tanpa batas dan biaya operasional yang lebih fleksibel.
- In-Memory Database: Basis data seperti SAP HANA dan Redis menyimpan data sepenuhnya di memori utama untuk akses ultra-cepat. Teknologi ini cocok untuk aplikasi dengan kebutuhan latensi rendah, seperti perdagangan saham atau game online.

## 2.6 Keamanan Basis Data Lanjut

Keamanan adalah salah satu elemen penting dalam pengelolaan basis data. Serangan siber yang mengancam integritas, kerahasiaan, dan ketersediaan data menuntut implementasi langkah-langkah keamanan yang komprehensif.

Kontrol Akses Berbasis Peran (Role-Based Access Control - RBAC): Membatasi akses data berdasarkan peran pengguna. Contohnya, hanya administrator yang dapat menghapus tabel, sementara pengguna biasa hanya dapat melihat data tertentu.

1. Enkripsi Data:

   Enkripsi melindungi data dalam dua tahap utama:

   a. Data-at-Rest: Data yang disimpan di disk dienkripsi menggunakan algoritma seperti AES-256.

      Data-in-Transit: Data yang bergerak melalui jaringan diamankan dengan protokol seperti TLS/SSL.

      - Pemantauan dan Audit: Audit log mencatat semua aktivitas pada basis data, termasuk perubahan data dan login pengguna. Alat pemantauan modern dapat mendeteksi pola akses mencurigakan dan memberikan peringatan dini.
      - Proteksi Terhadap Serangan SQL Injection: SQL Injection adalah teknik penyerangan di mana peretas menyisipkan kode berbahaya dalam query SQL. Untuk melindungi sistem dari ancaman ini, disarankan menggunakan query parameterized atau ORM (Object Relational Mapping) yang aman.

   b. Backup dan Recovery: Backup berkala dan strategi pemulihan sangat penting untuk melindungi data dari kegagalan sistem atau serangan ransomware. Teknik seperti snapshot database atau replikasi multi-region dapat digunakan untuk meningkatkan ketahanan sistem.

# BAB III

# PEMBAHASAN

## 3.1 Studi Kasus

INSTRUKSI KERJA

Anda adalah seorang Software Engineering yang akan melakukan tugas membuat sebuah aplikasi sederhana untuk manajemen toko buku online. Aplikasi ini membutuhkan database untuk menyimpan informasi buku, pelanggan, dan pesanan. Anda diminta untuk merancang skema database, mengimplementasikan skema tersebut dalam SQL, dan menulis beberapa kueri SQL untuk mengelola data sesuai dengan studi kasus yang telah ditentukan sebagai berikut :

1. Skema Database: Buatlah skema database yang mencakup tabel-tabel berikut:

   A. Tabel "Buku":

   - ID (integer, primary key)
   - Judul (varchar)
   - Penulis (varchar)
   - Tahun_Terbit (integer)
   - Harga (decimal)

   B. Tabel "Pelanggan":

   - ID (integer, primary key)
   - Nama (varchar)
   - Alamat (varchar)
   - Email (varchar)
   - Telepon (varchar)

   C. Tabel "Pesanan":

   - ID (integer, primary key)
   - Tanggal_Pesanan (date)
   - Pelanggan_ID (integer, foreign key ke Pelanggan)
   - Total_Harga (decimal)

   D. Tabel "Detail_Pesanan":

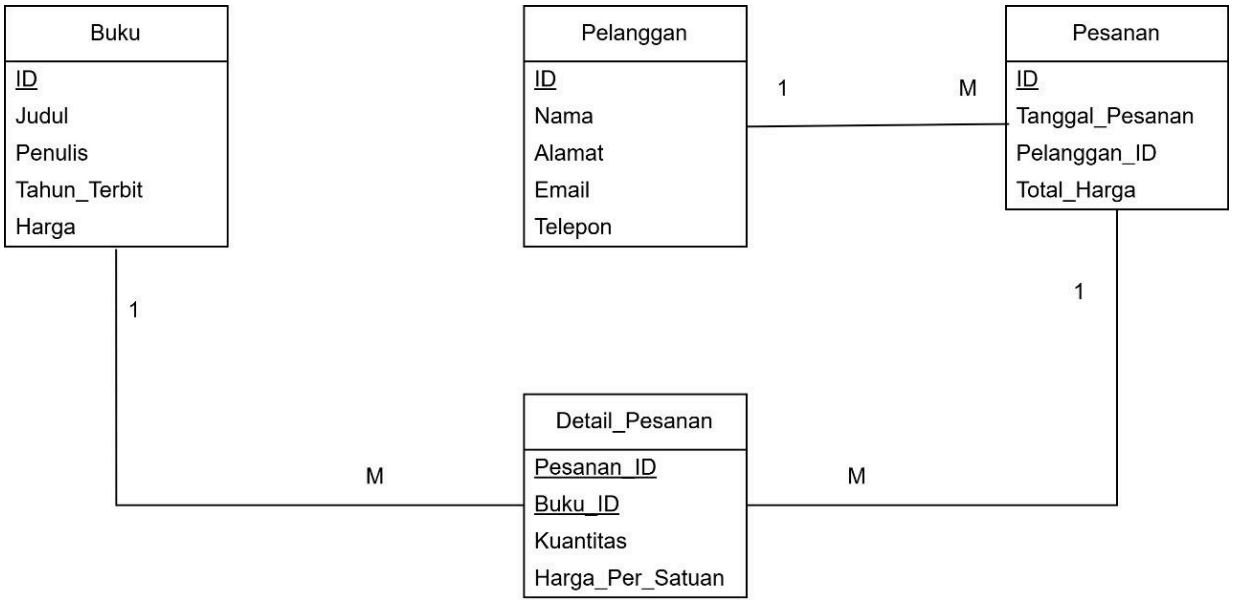
   - Pesanan_ID (integer, foreign key ke Pesanan)
   - Buku_ID (integer, foreign key ke Buku)
   - Kuantitas (integer)
   - Harga_Per_Satuan (decimal)

2. Implementasikan skema database yang telah Anda rancang dalam SQL. Pastikan untuk menambahkan indeks yang diperlukan untuk meningkatkan performa kueri.

3. Tulislah kueri SQL untuk melakukan hal berikut:
    a. Menampilkan daftar buku yang diterbitkan setelah tahun 2010.
    b. Menampilkan total harga dari setiap pesanan bersama dengan nama pelanggan yang melakukan pesanan.
    c. Menghapus semua buku yang telah diterbitkan sebelum tahun 2000.
    d. Menambahkan sebuah pesanan baru ke dalam database bersama dengan detail pesanan yang sesuai.
4. Buatlah skrip SQL untuk memulai transaksi, menambahkan detail pesanan, mengurangi stok buku yang dipesan, menghitung total harga pesanan, dan menyelesaikan transaksi dengan melakukan commit atau rollback jika terjadi kesalahan.
5. Berikan pemikiran Anda tentang bagaimana Anda akan memantau kinerja database Anda dan mengatasi masalah kinerja yang mungkin timbul.
6. Jelaskan langkah-langkah yang akan Anda ambil untuk melindungi aplikasi Anda dari serangan SQL injection dan tindakan keamanan lainnya yang akan Anda terapkan pada basis data Anda.

## 3.2 Perancangan Database secara Logika

Perancangan database secara logika adalah tahap lanjutan dari desain konseptual, di mana fokusnya adalah menerjemahkan model konseptual (seperti ERD atau EER) menjadi struktur yang dapat diimplementasikan pada sistem manajemen basis data (*Database Management System*, DBMS) tertentu. Tahap ini memastikan bahwa data dapat disimpan, diakses, dan dimanipulasi dengan efisien sesuai dengan kebutuhan sistem.



## 3.3 Perancangan Database secara Fisik

Perancangan database secara fisik adalah tahap di mana rancangan logis diimplementasikan ke dalam spesifikasi teknis pada Sistem Manajemen Basis Data (*Database Management System*, DBMS). Tujuan dari tahap ini adalah untuk memastikan bahwa database dapat berfungsi dengan optimal dalam lingkungan yang sebenarnya, dengan mempertimbangkan aspek performa, penyimpanan, dan keamanan data.

**Tabel Buku**

| No | Field | Type | Size | Coinstraints | Key |
|---|---|---|---|---|---|
| 1 | ID | Int | | | Primary Key |
| 2 | Judul | Varchar | 250 | Not null | |
| 3 | Penulis | Varchar | 250 | Not null | |
| 4 | Tahun_terbit | Int | | Not null | |
| 5 | Harga | Decimal | 10,2 | Not null | |

**Tabel Pelanggan**

| No | Field | Type | Size | Coinstraints | Key |
|---|---|---|---|---|---|
| 1 | ID | Int | | Not null | Primary Key |
| 2 | Nama | Varchar | 200 | Not null | |
| 3 | Alamat | Varchar | 250 | Not null | |
| 4 | Email | Varchar | 200 | Not null | |
| 5 | Telepon | Varchar | 15 | Not null | |

**Tabel Pesanan**

| No | Field | Type | Size | Coinstraints | Key |
|---|---|---|---|---|---|
| 1 | ID | Int | | Not null | Primary Key |
| 2 | Tanggal_pesanan | Date | | Not null | |
| 3 | Pelanggan_ID | Int | | Not null | Foreign Key |
| 4 | Total_harga | Decimal | 10,2 | Not null | |

**Tabel Detail_pesanan**

| No | Field | Type | Size | Coinstraints | Key |
|---|---|---|---|---|---|
| 1 | Pesanan_ID | Int | | Not null | Foreign Key |
| 2 | Buku_ID | Int | | Not null | Foreign Key |
| 3 | Kuantitas | Int | | Not null | |
| 4 | Harga_per_satuan | Decimal | 10,2 | Not null | |

## 3.4 Implementasi Sistem Database

1. Skema Database

Membuat Database Toko Buku

```
MariaDB [(none)]> create database Toko_Buku;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use Toko_Buku;
Database changed
```

Membuat Tabel Buku, Pelanggan, Pesanan dan Detail Pesanan

```
MariaDB [Toko_Buku]> create table Buku (
    -> ID int primary key,
    -> Judul varchar (250) not null,
    -> Penulis varchar (250) not null,
    -> Tahun_terbit int not null,
    -> Harga decimal (10, 2) not null);
Query OK, 0 rows affected (0.346 sec)
```

```
MariaDB [Toko_Buku]> desc Buku;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| ID          | int(11)      | NO   | PRI | NULL    |       |
| Judul       | varchar(250) | NO   |     | NULL    |       |
| Penulis     | varchar(250) | NO   |     | NULL    |       |
| Tahun_terbit | int(11)     | NO   | MUL | NULL    |       |
| Harga       | decimal(10,2) | NO  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.097 sec)
```

```
MariaDB [Toko_Buku]> create table Pelanggan (
    -> ID int primary key,
    -> Nama varchar (200) not null,
    -> Alamat varchar (250) not null,
    -> Email varchar (200) not null,
    -> Telepon varchar (15) not null);
Query OK, 0 rows affected (0.229 sec)
```

```
MariaDB [Toko_Buku]> desc Pelanggan;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| ID      | int(11)      | NO   | PRI | NULL    |       |
| Nama    | varchar(200) | NO   |     | NULL    |       |
| Alamat  | varchar(250) | NO   |     | NULL    |       |
| Email   | varchar(200) | NO   | MUL | NULL    |       |
| Telepon | varchar(15)  | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.035 sec)
```

```
MariaDB [Toko_Buku]> create table Pesanan (
    -> ID int primary key,
    -> Tanggal_pesanan date not null,
    -> Pelanggan_ID int not null,
    -> Total_harga decimal (10, 2) not null,
    -> foreign key (Pelanggan_ID) references Pelanggan(ID));
Query OK, 0 rows affected (0.338 sec)
```

```
MariaDB [Toko_Buku]> desc Pesanan;
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| ID              | int(11)       | NO   | PRI | NULL    |       |
| Tanggal_pesanan | date          | NO   | MUL | NULL    |       |
| Pelanggan_ID    | int(11)       | NO   | MUL | NULL    |       |
| Total_harga     | decimal(10,2) | NO   |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
4 rows in set (0.086 sec)
```

```
MariaDB [Toko_Buku]> create table Detail_Pesanan (
    -> Pesanan_ID int not null,
    -> Buku_ID int not null,
    -> Kuantitas int not null,
    -> Harga_per_satuan decimal(10, 2) not null,
    -> primary key (Pesanan_ID, Buku_ID),
    -> foreign key (Pesanan_ID) REFERENCES Pesanan(ID) ON delete cascade on update cascade,
    -> foreign key (Buku_ID) REFERENCES Buku(ID) ON delete cascade on update cascade);
Query OK, 0 rows affected (0.771 sec)
```

```
MariaDB [Toko_Buku]> desc Detail_pesanan;
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| Pesanan_ID      | int(11)       | NO   | PRI | NULL    |       |
| Buku_ID         | int(11)       | NO   | PRI | NULL    |       |
| Kuantitas       | int(11)       | NO   |     | NULL    |       |
| Harga_per_satuan | decimal(10,2) | NO   |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
4 rows in set (0.016 sec)
```

Menambahkan Data ke Tabel Buku, Pelanggan, Pesanan dan Detail Pesanan

```
MariaDB [Toko_Buku]> insert into Buku values
    -> (1, 'Harry Potter and the Sorcerer\'s Stone', 'J.K. Rowling', 1997, 150000),
    -> (2, 'Layangan Putus', 'Mommy ASF', 2020, 120000),
    -> (3, 'Indistractable', 'Nir Eyal & Julie Li', 2019, 116000),
    -> (4, 'To Kill a Mockingbird', 'Harper Lee', 1960, 100000),
    -> (5, 'Filosofi Teras', 'Henry Manampiring', 2018, 90000);
Query OK, 5 rows affected (0.062 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [Toko_Buku]> insert into Pelanggan values
    -> (101, 'James', 'Jl. Merdeka No. 10, Jakarta', 'james@gmail.com', '081234567890'),
    -> (102, 'Karin', 'Jl. Sutan Syahrir No. 5, Bandung', 'kariin@yahoo.com', '082345678901'),
    -> (103, 'Adam', 'Jl. Sudirman No. 12, Surabaya', 'adamj@outlook.com', '083456789012'),
    -> (104, 'Emily', 'Jl. Pemuda No. 8, Yogyakarta', 'emily.kh@mail.com', '084567890123'),
    -> (105, 'David', 'Jl. Raya No. 15, Bali', 'davidle@hotmail.com', '085678901234');
Query OK, 5 rows affected (1.541 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [Toko_Buku]> insert into Pesanan values
    -> (221, '2024-11-18', 101, 270000),
    -> (222, '2024-11-19', 102, 232000),
    -> (223, '2024-11-20', 103, 100000),
    -> (224, '2024-11-21', 104, 90000),
    -> (225, '2024-11-22', 105, 240000);
Query OK, 5 rows affected (0.041 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [Toko_Buku]> insert into Detail_pesanan values
    -> (221, 1, 1, 150000),
    -> (221, 2, 1, 120000),
    -> (222, 3, 2, 232000),
    -> (223, 4, 1, 100000),
    -> (224, 5, 1, 90000),
    -> (225, 2, 2, 240000);
Query OK, 6 rows affected (0.064 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Menampilkan Data Tabel Buku, Pelanggan, Pesanan dan Detail Pesanan

```
MariaDB [Toko_Buku]> select *from Buku;
+----+--------------------------------------+--------------------+--------------+-----------+
| ID | Judul                                | Penulis            | Tahun_terbit | Harga     |
+----+--------------------------------------+--------------------+--------------+-----------+
|  1 | Harry Potter and the Sorcerer's Stone | J.K. Rowling       |         1997 | 150000.00 |
|  2 | Layangan Putus                       | Mommy ASF          |         2020 | 120000.00 |
|  3 | Indistractable                       | Nir Eyal & Julie Li |         2019 | 116000.00 |
|  4 | To Kill a Mockingbird                | Harper Lee         |         1960 | 100000.00 |
|  5 | Filosofi Teras                       | Henry Manampiring  |         2018 |  90000.00 |
+----+--------------------------------------+--------------------+--------------+-----------+
5 rows in set (0.000 sec)
```

```
MariaDB [Toko_Buku]> select *from Pelanggan;
+-----+-------+---------------------------------+---------------------+--------------+
| ID  | Nama  | Alamat                          | Email               | Telepon      |
+-----+-------+---------------------------------+---------------------+--------------+
| 101 | James | Jl. Merdeka No. 10, Jakarta     | james@gmail.com     | 081234567890 |
| 102 | Karin | Jl. Sutan Syahrir No. 5, Bandung | kariin@yahoo.com    | 082345678901 |
| 103 | Adam  | Jl. Sudirman No. 12, Surabaya   | adamj@outlook.com   | 083456789012 |
| 104 | Emily | Jl. Pemuda No. 8, Yogyakarta    | emily.kh@mail.com   | 084567890123 |
| 105 | David | Jl. Raya No. 15, Bali           | davidle@hotmail.com | 085678901234 |
+-----+-------+---------------------------------+---------------------+--------------+
5 rows in set (0.000 sec)
```

```
MariaDB [Toko_Buku]> select *from Pesanan;
+-----+-----------------+--------------+-------------+
| ID  | Tanggal_pesanan | Pelanggan_ID | Total_harga |
+-----+-----------------+--------------+-------------+
| 221 | 2024-11-18      |          101 |   270000.00 |
| 222 | 2024-11-19      |          102 |   232000.00 |
| 223 | 2024-11-20      |          103 |   100000.00 |
| 224 | 2024-11-21      |          104 |    90000.00 |
| 225 | 2024-11-22      |          105 |   240000.00 |
+-----+-----------------+--------------+-------------+
5 rows in set (0.001 sec)
```

```
MariaDB [Toko_Buku]> select *from Detail_pesanan;
+------------+---------+-----------+------------------+
| Pesanan_ID | Buku_ID | Kuantitas | Harga_per_satuan |
+------------+---------+-----------+------------------+
|        221 |       1 |         1 |        150000.00 |
|        221 |       2 |         1 |        120000.00 |
|        222 |       3 |         2 |        232000.00 |
|        223 |       4 |         1 |        100000.00 |
|        224 |       5 |         1 |         90000.00 |
|        225 |       2 |         2 |        240000.00 |
+------------+---------+-----------+------------------+
6 rows in set (0.000 sec)
```

2. Menambahkan Indeks

Indeks pada Tabel Buku: Menambahkan indeks pada kolom Tahun_terbit agar pencarian berdasarkan tahun lebih cepat

```
MariaDB [Toko_Buku]> create INDEX idx_buku_tahun_terbit on Buku(Tahun_terbit);
Query OK, 0 rows affected (0.297 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [Toko_Buku]> select * from Buku where Tahun_Terbit > 2018;
+----+---------------+--------------------+--------------+-----------+
| ID | Judul         | Penulis            | Tahun_terbit | Harga     |
+----+---------------+--------------------+--------------+-----------+
|  3 | Indistractable | Nir Eyal & Julie Li |         2019 | 116000.00 |
|  2 | Layangan Putus | Mommy ASF          |         2020 | 120000.00 |
+----+---------------+--------------------+--------------+-----------+
2 rows in set (0.000 sec)
```

Indeks pada Tabel Pelanggan: Menambahkan indeks pada kolom Email untuk mempercepat pencarian pelanggan berdasarkan email

```
MariaDB [Toko_Buku]> create INDEX idx_pelanggan_email ON Pelanggan(Email);
Query OK, 0 rows affected (0.298 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [Toko_Buku]> select * from Pelanggan where Email = 'kariin@yahoo.com';
+-----+-------+---------------------------------+-----------------+--------------+
| ID  | Nama  | Alamat                          | Email           | Telepon      |
+-----+-------+---------------------------------+-----------------+--------------+
| 102 | Karin | Jl. Sutan Syahrir No. 5, Bandung | kariin@yahoo.com | 082345678901 |
+-----+-------+---------------------------------+-----------------+--------------+
1 row in set (0.001 sec)
```

Indeks pada Tabel Pesanan: Menambahkan indeks pada kolom Tanggal_pesanan untuk memudahkan pencarian pesanan berdasarkan tanggal

```
MariaDB [Toko_Buku]> create INDEX idx_pesanan_tanggal ON Pesanan(Tanggal_pesanan);
Query OK, 0 rows affected (0.232 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [Toko_Buku]> select * from Pesanan where Tanggal_Pesanan BETWEEN '2024-11-18' AND '2024-11-20';
+-----+-----------------+--------------+-------------+
| ID  | Tanggal_pesanan | Pelanggan_ID | Total_harga |
+-----+-----------------+--------------+-------------+
| 221 | 2024-11-18      |          101 |   270000.00 |
| 222 | 2024-11-19      |          102 |   232000.00 |
| 223 | 2024-11-20      |          103 |   100000.00 |
+-----+-----------------+--------------+-------------+
3 rows in set (0.000 sec)
```

Indeks pada Tabel Detail_pesanan: Menambahkan indeks pada kolom Pesanan_ID dan Buku_ID untuk mempercepat pencarian data detail pesanan

```
MariaDB [Toko_Buku]> create INDEX idx_detail_pesanan ON Detail_Pesanan(Pesanan_ID, Buku_ID);
Query OK, 0 rows affected (0.258 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [Toko_Buku]> select * from Detail_Pesanan where Pesanan_ID = 221;
+------------+---------+-----------+------------------+
| Pesanan_ID | Buku_ID | Kuantitas | Harga_per_satuan |
+------------+---------+-----------+------------------+
|        221 |       1 |         1 |        150000.00 |
|        221 |       2 |         1 |        120000.00 |
+------------+---------+-----------+------------------+
2 rows in set (0.000 sec)
```

3. Kueri SQL
    a. Menampilkan daftar buku yang diterbitkan setelah tahun 2010.

```
MariaDB [Toko_Buku]> select * from Buku
    -> where Tahun_Terbit > 2010;
+----+----------------+---------------------+--------------+-----------+
| ID | Judul          | Penulis             | Tahun_terbit | Harga     |
+----+----------------+---------------------+--------------+-----------+
|  2 | Layangan Putus | Mommy ASF           |         2020 | 120000.00 |
|  3 | Indistractable | Nir Eyal & Julie Li |         2019 | 116000.00 |
|  5 | Filosofi Teras | Henry Manampiring   |         2018 |  90000.00 |
+----+----------------+---------------------+--------------+-----------+
3 rows in set (0.026 sec)
```

    b. Menampilkan total harga dari setiap pesanan bersama dengan nama pelanggan yang melakukan pesanan.

```
MariaDB [Toko_Buku]> select p.Nama, pes.Total_Harga
    -> from Pesanan pes
    -> JOIN Pelanggan p ON pes.Pelanggan_ID = p.ID;
+-------+-------------+
| Nama  | Total_Harga |
+-------+-------------+
| James |   270000.00 |
| Karin |   232000.00 |
| Adam  |   100000.00 |
| Emily |    90000.00 |
| David |   240000.00 |
+-------+-------------+
5 rows in set (0.001 sec)
```

    c. Menghapus semua buku yang telah diterbitkan sebelum tahun 2000.

```
MariaDB [Toko_Buku]> delete from Buku
    -> where Tahun_Terbit < 2000;
Query OK, 2 rows affected (0.077 sec)

MariaDB [Toko_Buku]> select *from Buku;
+----+----------------+---------------------+--------------+-----------+
| ID | Judul          | Penulis             | Tahun_terbit | Harga     |
+----+----------------+---------------------+--------------+-----------+
|  2 | Layangan Putus | Mommy ASF           |         2020 | 120000.00 |
|  3 | Indistractable | Nir Eyal & Julie Li |         2019 | 116000.00 |
|  5 | Filosofi Teras | Henry Manampiring   |         2018 |  90000.00 |
+----+----------------+---------------------+--------------+-----------+
3 rows in set (0.000 sec)
```

    d. Menambahkan sebuah pesanan baru ke dalam database bersama dengan detail pesanan yang sesuai.

```
MariaDB [Toko_Buku]> insert into Pesanan values
    -> (226,'2024-11-23', 105, 420000);
Query OK, 1 row affected (0.069 sec)

MariaDB [Toko_Buku]> select *from Pesanan;
+-----+-----------------+--------------+-------------+
| ID  | Tanggal_pesanan | Pelanggan_ID | Total_harga |
+-----+-----------------+--------------+-------------+
| 221 | 2024-11-18      |          101 |   270000.00 |
| 222 | 2024-11-19      |          102 |   232000.00 |
| 223 | 2024-11-20      |          103 |   100000.00 |
| 224 | 2024-11-21      |          104 |    90000.00 |
| 225 | 2024-11-22      |          105 |   240000.00 |
| 226 | 2024-11-23      |          105 |   420000.00 |
+-----+-----------------+--------------+-------------+
6 rows in set (0.000 sec)
```

```
MariaDB [Toko_Buku]> insert into Detail_Pesanan values
    -> (226, 5, 2, 90000),
    -> (226, 2, 1, 240000);
Query OK, 2 rows affected (0.085 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [Toko_Buku]> select *from Detail_pesanan;
+------------+---------+-----------+------------------+
| Pesanan_ID | Buku_ID | Kuantitas | Harga_per_satuan |
+------------+---------+-----------+------------------+
|        221 |       2 |         1 |        120000.00 |
|        222 |       3 |         2 |        232000.00 |
|        224 |       5 |         1 |         90000.00 |
|        225 |       2 |         2 |        240000.00 |
|        226 |       2 |         1 |        240000.00 |
|        226 |       5 |         2 |         90000.00 |
+------------+---------+-----------+------------------+
6 rows in set (0.000 sec)
```

4. Skrip SQL

   Menambahkan Kolom Stok di Tabel Buku

```
MariaDB [Toko_Buku]> alter table Buku
    -> add column Stok int not null default 0;
Query OK, 0 rows affected (0.099 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [Toko_Buku]> update Buku
    -> set Stok = 10;
Query OK, 3 rows affected (0.068 sec)
Rows matched: 3  Changed: 3  Warnings: 0

MariaDB [Toko_Buku]> select *from Buku;
+----+---------------+--------------------+--------------+-----------+------+
| ID | Judul         | Penulis            | Tahun_terbit | Harga     | Stok |
+----+---------------+--------------------+--------------+-----------+------+
|  2 | Layangan Putus | Mommy ASF          |         2020 | 120000.00 |   10 |
|  3 | Indistractable | Nir Eyal & Julie Li |         2019 | 116000.00 |   10 |
|  5 | Filosofi Teras | Henry Manampiring  |         2018 |  90000.00 |   10 |
+----+---------------+--------------------+--------------+-----------+------+
3 rows in set (0.000 sec)
```

   Memulai Transaksi

```
MariaDB [Toko_Buku]> START TRANSACTION;
Query OK, 0 rows affected (0.000 sec)
```

Melakukan Transaksi (Menambahkan detail pesanan, mengurangi stok buku dan menghitung total harga pesanan)

```
MariaDB [Toko_Buku]> insert into Detail_pesanan values
    -> (224,5,1,90000);
Query OK, 1 row affected (0.129 sec)

MariaDB [Toko_Buku]> update Buku
    -> set Stok = Stok - 1
    -> where ID = 5;
Query OK, 1 row affected (0.001 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Toko_Buku]> select Pesanan_ID, SUM(Kuantitas * Harga_per_satuan) AS Total_harga
    -> from Detail_pesanan
    -> where Pesanan_ID = 224
    -> group by Pesanan_ID;
+------------+-------------+
| Pesanan_ID | Total_harga |
+------------+-------------+
|        224 |    90000.00 |
+------------+-------------+
1 row in set (0.001 sec)
```

```
MariaDB [Toko_Buku]> COMMIT;
Query OK, 0 rows affected (0.049 sec)

MariaDB [Toko_Buku]> select 'Transaksi Berhasil' AS pesan;
+--------------------+
| pesan              |
+--------------------+
| Transaksi Berhasil |
+--------------------+
1 row in set (0.000 sec)

MariaDB [Toko_Buku]> ROLLBACK;
Query OK, 0 rows affected (0.000 sec)

MariaDB [Toko_Buku]> select 'Terjadi Kesalahan, Transaksi Dibatalkan' AS pesan;
+----------------------------------------+
| pesan                                  |
+----------------------------------------+
| Terjadi Kesalahan, Transaksi Dibatalkan |
+----------------------------------------+
1 row in set (0.000 sec)
```

Menyelesaikan Transaksi

5. Memantau Kinerja Database dan Mengatasi Masalah Kinerja

   Fokus pemantauan kinerja database aplikasi manajemen toko buku online yaitu pada:

   a. Query Performance

      Memantau berapa lama setiap query membutuhkan waktu untuk diproses. Query yang lambat dapat menjadi indikasi adanya masalah pada indeks, join, atau struktur tabel. Jumlah baris yang dikembalikan oleh setiap query juga perlu dipantau. Query yang mengembalikan terlalu banyak baris dapat menyebabkan kinerja menurun.

   b. Resource Utilization

Memantau penggunaan CPU oleh database server. CPU yang terlalu tinggi dapat mengindikasikan adanya query yang intensif atau masalah hardware, penggunaan memori oleh database server. Memori yang tidak mencukupi dapat menyebabkan swapping dan memperlambat kinerja, dan aktivitas I/O pada disk. I/O yang tinggi dapat mengindikasikan masalah pada disk atau konfigurasi storage.

c. Error Logs

Mencari pesan error yang terkait dengan kinerja, seperti deadlock, timeout, atau kesalahan pada query.

Alat yang dapat digunakan untuk memantau kinerja database MySQL yaitu alat bawaan seperti 'EXPLAIN' untuk menganalisis query, 'SHOW STATUS' untuk melihat status server, dan 'SLOW QUERY LOG' untuk melacak query lambat. Bisa juga menggunakan alat pihak ketiga yaitu Percona Monitoring and Management (PMM) yang merupakan suite alat yang komprehensif untuk memantau kinerja MySQL.

Permasalahan kinerja databse yang biasa terjadi seperti query lambat, deadlock tabel yang terlalu besar, indeks yang tidak optimal, dan lain sebagainya dapat diatasi dengan beberapa cara yaitu sebagai berikut:

a. Analisis query: Identifikasi query lambat dan optimalkan dengan menambahkan indeks, mengubah struktur tabel, atau menggunakan teknik query yang lebih efisien.
b. Normalisasi data: Memastikan data ternormalisasi dengan baik untuk menghindari redundansi dan meningkatkan efisiensi query.
c. Meningkatkan konfigurasi database: Menyesuaikan konfigurasi database untuk meningkatkan kinerja, seperti mengubah ukuran buffer pool, jumlah koneksi maksimum, atau pengaturan caching.
d. Sharding: Membagi database menjadi beberapa shard yang lebih kecil jika tabel sudah sangat besar.
e. Caching: Menggunakan caching untuk menyimpan hasil query yang sering digunakan, sehingga tidak perlu dihitung ulang setiap kali.

6. Langkah-langkah untuk Melindungi Aplikasi

Serangan SQL injection merupakan ancaman serius terhadap keamanan data, yang mengeksploitasi kerentanan dalam aplikasi berbasis web untuk mengakses dan memodifikasi data sensitif. Langkah-langkah penting untuk melindungi aplikasi, memulihkan dan memperkuat keamanan server yang telah

menjadi sasaran serangan SQL injection. Untuk melindungi aplikasi dari serangan SQL Injection, berikut adalah beberapa langkah yang dapat diambil:

a. Gunakan Prepared Statements dan Parameterized Queries

   Prepared statements dan parameterized queries adalah metode yang paling efektif untuk mencegah SQL Injection. Dengan teknik ini, input pengguna tidak akan pernah digabungkan langsung ke dalam kueri SQL, sehingga menghilangkan peluang bagi penyerang untuk menyisipkan kode berbahaya.

   Kode SQL : SELECT * FROM users WHERE username = ? AND password = ?;

b. Validasi dan Sanitasi Input Pengguna

   Pastikan semua input pengguna divalidasi dan disanitasi sebelum digunakan dalam kueri SQL. Batasi input hanya pada karakter atau format yang diharapkan dan hindari penggunaan karakter khusus yang dapat digunakan dalam injeksi SQL.

c. Batasi Hak Akses Database

   Batasi hak akses pengguna database hanya pada operasi yang diperlukan. Jangan memberikan hak administratif pada akun yang digunakan oleh aplikasi web. Ini akan membatasi dampak serangan jika terjadi SQL Injection.

d. Gunakan ORM (Object-Relational Mapping) Frameworks

   Menggunakan ORM frameworks seperti Hibernate atau Entity Framework dapat membantu mengurangi risiko SQL Injection dengan menyediakan lapisan abstraksi antara aplikasi dan database, yang secara otomatis menangani query parameterization.

e. Monitor dan Audit Aktivitas Database

   Monitoring dan auditing aktivitas database dapat membantu mendeteksi aktivitas mencurigakan. Dengan memantau log kueri SQL yang dijalankan, Anda dapat mengidentifikasi pola serangan dan mengambil tindakan pencegahan lebih lanjut.

f. Pentingnya Pengujian Keamanan Aplikasi

   Pengujian keamanan aplikasi secara berkala sangat penting untuk memastikan bahwa perlindungan terhadap SQL Injection sudah cukup kuat. Gunakan alat-alat pengujian keamanan otomatis dan manual untuk mengidentifikasi potensi celah dan segera perbaiki.

Untuk melakukan pemulihan aplikasi dari serangan SQL Injection, berikut adalah beberapa langkah yang dapat diambil:

a. Identifikasi dan Isolasi

   Langkah pertama adalah mengidentifikasi dan mengisolasi server yang terdampak. Hentikan semua proses yang berjalan, cabut dari jaringan, dan cadangkan data penting.

b. Pemeriksaan dan Analisis

   Lakukan pemeriksaan menyeluruh pada server untuk mengidentifikasi bagaimana serangan terjadi. Cari kode berbahaya, pola akses tidak sah, dan aktivitas mencurigakan lainnya.

c. Perbaikan Kerentanan

   Perbaiki semua kerentanan yang memungkinkan serangan terjadi. Ini mungkin termasuk memperbarui perangkat lunak, menambal celah keamanan, dan menerapkan praktik pengkodean yang aman.

d. Pemulihan Data

   Jika memungkinkan, pulihkan data yang rusak atau hilang dari cadangan terbaru. Verifikasi integritas data dan pastikan tidak ada data yang disusupi atau dimodifikasi.

Untuk melakukan penguatan keamanan aplikasi dari serangan SQL Injection, berikut adalah beberapa langkah yang dapat diambil:

a. Validasi Input

   Terapkan validasi input yang ketat untuk semua input pengguna, memastikan bahwa hanya data yang diharapkan dan valid yang diproses.

b. Pencegahan SQL Injection

   Gunakan teknik pencegahan SQL injection seperti pernyataan terparameter, pelarian karakter, dan pemfilteran kata kunci.

c. Pemantauan dan Logging

Aktifkan pemantauan dan logging yang komprehensif untuk mendeteksi dan merespons aktivitas mencurigakan secara cepat.

d. Pelatihan dan Kesadaran

Berikan pelatihan dan tingkatkan kesadaran tentang serangan SQL injection kepada tim pengembangan dan administrasi.

e. Cadangan dan Pemulihan Bencana

Pastikan rencana pencadangan dan pemulihan bencana yang komprehensif diterapkan untuk meminimalkan dampak serangan di masa mendatang.

# BAB IV

# PENUTUP

## 3.5 Kesimpulan

Berdasarkan data yang kami simpulkan di atas menunjukkan bahwa pengelolaan data yang efektif serta efisien sangat penting bagi sebuah toko buku, terlebih yang dimana pengelolaan stok buku, data pelanggan, hingga proses pemesanan dibutuhkan nya sistem yang cepat dan akurat. oleh karena itu, dengan adanya aplikasi manajemen toko buku, toko dapat memantau stok buku secara real-time, dan memastikan data pelanggan serta transaksi tersimpan secara aman. maka dari itu, pengembangan aplikasi untuk manajemen toko buku menjadi langkah yang tepat untuk mendukung transformasi digital di sektor ini.

## 3.6 Saran

**1. Peningkatan Penerapan Teknologi Terkini**

Menyarankan untuk terus mengikuti perkembangan teknologi basis data, seperti penggunaan NoSQL untuk data tidak terstruktur, atau bisa di sebut sebagai cloud database untuk fleksibilitas dan skalabilitas yang jauh lebih baik. Hal ini akan sangat membantu dalam transformasi disektor toko buku dalam menghadapi pertumbuhan data dan permintaan yang akan meningkat.

**2. Pelatihan dan Kesadaran Keamanan**

Ini adalah hal yang sangat penting untuk memberikan pelatihan kepada tim pengembangan dan administrasi mengenai keamanan aplikasi dan praktik terbaik dalam pengkodean. Kesadaran ini akan membuat potensi ancaman, seperti SQL injection, harus di tingkatkan agar semua anggota tim dapat berkontribusi dalam menjaga keamanan aplikasi

**3. Monitoring dan Pemeliharaan Rutin**

Lakukan pemantauan secara berkala terhadap performa database dan aktivitas pada pengguna. Audit log dan analisis performa dapat membantu mendeteksi masalah sebelum menjadi kritis. Pemilihan rutin ini seperti pembersihan data yang tidak relevan, dan juga penting untuk menjaga kinerja database

**4. Pengembangan Berkelanjutan**

Aplikasi manajemen toko buku harus terus dikembangkan dan di perbaharui untuk memenuhi kebutuhan user / pengguna yang dapat berubah. Mendengarkan umpan balik dari pelanggan dan karyawan dapat memberikan wawasan berharga untuk perbaikan aplikasi.

**5. Backup dan Pemulihan Data**

Mengimplementasikan strategi cadangan yang kuat untuk melindungi data dari kehilangan. Rencana pemulihan bencana harus disiapkan untuk memastikan bahwa data dapat di pulihkan dengan cepat jika terjadi suatu kegagalan sistem atau serangan cyber.

**6. Penggunaan Framework ORM**

Mempertimbangkan untuk menggunakan framework Object - Relational Mapping ( ORM ) untuk meningkatkan produktivitas pengembangan dan mengurangi resiko SQL injection. ORM dapat menyederhanakan interaksi dengan database dan mengurangi jumlah kode yang harus ditulis

## DAFTAR PUSTAKA


Amazon Web Services. (n.d.). *Basis data relasional*. Diakses pada 24 November 2024, dari https://aws.amazon.com/id/relational-database/

AppMaster. (n.d.). *Panduan skema database beserta contohnya*. Diakses pada 24 November 2024, dari https://appmaster.io/id/blog/panduan-skema-database-beserta-contohnya

Hostinger. (n.d.). *Apa itu query?*. Diakses pada 25 November 2024, dari https://www.hostinger.co.id/tutorial/apa-itu-query

Unmaha. (2023, November 1). *Cara optimasi query di database manajemen sistem*. Diakses pada 25 November 2024, dari https://blog.unmaha.ac.id/cara-optimasi-query-di-database-manajemen-sistem/

Binus University. (2023). *Inovasi teknologi basis data dari SQL ke NoSQL dan seterusnya*. Diakses pada 25 November 2024, dari https://sis.binus.ac.id/2023/11/01/inovasi-teknologi-basis-data-dari-sql-ke-nosql-dan-seterusnya/

Puskomedia. (n.d.). *Menerapkan strategi keamanan basis data untuk melindungi data pelanggan dan informasi penting*. Diakses pada 26 November 2024, dari https://www.puskomedia.id/blog/menerapkan-strategi-keamanan-basis-data-untuk-melindungi-data-pelanggan-dan-informasi-penting/

UTI-TTIS. (n.d.). *Perlindungan terhadap serangan SQL injection: Panduan lengkap untuk keamanan database*. Diakses pada 26 November 2024, dari Perlindungan Terhadap Serangan SQL Injection: Panduan Lengkap untuk Keamanan Database - UTI-TTIS.


**Link Video Presentasi: https://youtu.be/CwpLP027y3g?si=FSPEatdZTJJnGwoK**