

Day 1

Konsep Database dan Query Dasar

By Werdani S Hadi

1. Pengantar Database dan DBMS

Bayangkan Anda adalah seorang penjual buku di toko kecil. Awalnya, Anda bisa mengingat semua buku yang Anda jual, siapa saja pelanggannya, dan buku apa yang mereka beli. Namun, seiring waktu, toko Anda semakin besar. Buku bertambah, pelanggan makin banyak, dan pesanan menumpuk. Jika Anda hanya mengandalkan ingatan, pasti akan kacau.



Di sinilah **database** berperan. Database adalah **sistem terorganisir untuk menyimpan, mengelola, dan mengambil data secara efisien**. Jadi, database mirip seperti lemari arsip digital yang sangat rapi dan canggih. Ia memungkinkan Anda menyimpan semua informasi

(data) tentang buku, pelanggan, dan transaksi di satu tempat, sehingga Anda bisa menemukannya dengan cepat dan mudah.



DBMS adalah singkatan dari Database Management System. Secara sederhana, DBMS adalah perangkat lunak atau program yang berfungsi untuk mengelola, menyimpan, mengambil, dan memperbarui data dalam database.

Bayangkan jika database adalah perpustakaan digital, maka DBMS adalah pustakawan canggih yang mengatur semua buku di dalamnya. Anda tidak perlu tahu di rak mana buku A berada atau bagaimana cara mengambilnya. Anda hanya perlu memberitahu pustakawan (DBMS) bahwa Anda ingin buku A, dan dia akan mencarikannya untuk Anda.

Berikut merupakan fungsi utama DBMS :

1. **Mendefinisikan Data (Data Definition):** DBMS memungkinkan Anda untuk membuat dan memodifikasi struktur database, seperti membuat tabel baru, menentukan kolom, dan tipe datanya. Ini seperti membuat rak buku dan memberi label pada setiap slot.

2. **Memanipulasi Data (Data Manipulation)**: Ini adalah fungsi yang paling sering kita gunakan. DBMS memungkinkan Anda untuk:
 - **Menyimpan** data baru (**INSERT**).
 - **Mengambil** data yang ada (**SELECT**).
 - **Mengubah** data yang sudah ada (**UPDATE**).
 - **Menghapus** data (**DELETE**).
3. **Mengamankan Data (Data Security)**: DBMS menyediakan mekanisme keamanan untuk mengontrol siapa saja yang bisa mengakses atau mengubah data, serta tingkat aksesnya. Ini seperti memberikan kunci kepada orang tertentu untuk masuk ke area tertentu di perpustakaan.
4. **Mengelola Integritas Data (Data Integrity)**: DBMS memastikan bahwa data yang disimpan akurat dan konsisten. Contohnya, jika Anda menetapkan kolom **umur** sebagai angka, DBMS akan menolak jika Anda mencoba memasukkan teks ke dalamnya.

Jadi, Database adalah kumpulan data itu sendiri (isi perpustakaannya). DBMS adalah perangkat lunak yang mengelola kumpulan data tersebut (pustakawannya).

2. Mengapa Menggunakan Database?

Menggunakan database memberikan banyak keuntungan, seperti:

- **Penyimpanan Terstruktur**: Data disimpan dalam format tabel (seperti *spreadsheet*) yang terorganisir, membuatnya mudah dibaca dan dipahami.
- **Akses Cepat**: Anda bisa mencari dan mengambil data spesifik dalam hitungan detik, bahkan dari jutaan data.
- **Integritas Data**: Database memastikan data yang disimpan konsisten dan akurat, mengurangi risiko kesalahan.
- **Keamanan**: Anda dapat mengontrol siapa saja yang bisa mengakses atau mengubah data, menjaga informasi tetap aman.

3. Jenis Database dan Tools yang digunakan

Berikut adalah beberapa contoh **DBMS** (Database Management System) yang paling umum dan sering digunakan, dikategorikan berdasarkan model datanya:

a. Relational DBMS (RDBMS)

Ini adalah jenis DBMS yang paling populer. Mereka menyimpan data dalam tabel-tabel yang terstruktur dan saling terhubung melalui relasi.

- **MySQL**: Salah satu RDBMS *open-source* yang paling banyak digunakan, populer untuk aplikasi web seperti WordPress.
- **PostgreSQL**: RDBMS *open-source* yang kuat dan sering dianggap lebih unggul dari MySQL dalam hal fitur dan kepatuhan standar SQL.
- **Oracle Database**: DBMS komersial terkemuka yang sering digunakan di perusahaan besar untuk menangani data skala besar.
- **Microsoft SQL Server**: RDBMS yang dikembangkan oleh Microsoft, populer di lingkungan Windows dan .NET.
- **MariaDB**: Versi *fork* dari MySQL yang dibuat oleh pengembang aslinya, menawarkan fitur dan performa yang ditingkatkan.

b. NoSQL DBMS

Ini adalah jenis DBMS yang dirancang untuk menangani data tidak terstruktur atau semi-terstruktur dengan lebih fleksibel daripada RDBMS. Mereka tidak menggunakan model tabel tradisional.

- **MongoDB**: Contoh terkemuka dari basis data berbasis dokumen (*document-based*). Sangat fleksibel dan sering digunakan untuk aplikasi yang membutuhkan skema dinamis.
- **Redis**: Basis data berbasis *key-value* yang sangat cepat dan sering digunakan untuk *caching* dan menyimpan data sesi.
- **Cassandra**: Basis data berbasis kolom (*column-family*) yang dirancang untuk menangani data dalam jumlah besar dengan ketersediaan tinggi.
- **Neo4j**: Contoh basis data berbasis grafik (*graph-based*) yang ideal untuk data dengan banyak relasi kompleks, seperti jejaring sosial.

Pilihan DBMS tergantung pada kebutuhan spesifik proyek Anda, seperti jenis data yang akan disimpan, skala aplikasi, dan anggaran.



Mengapa banyak orang suka MySQL?

- **Gratis dan Open-Source:** Anda bisa menggunakannya tanpa biaya dan komunitasnya sangat aktif.
- **Populer:** Banyak sekali tutorial, forum, dan dokumentasi yang bisa Anda temukan *online*.
- **Terintegrasi dengan Baik:** Sangat cocok digunakan bersama bahasa pemrograman web seperti PHP, Python, dan JavaScript.

Untuk tools yang dipakai, bisa menggunakan XAMPP sebagai server di komputer masing-masing.

<https://www.apachefriends.org/>

4. Tips Menguasai Database

Untuk bisa mahir menggunakan database, ada beberapa hal dasar yang perlu Anda kuasai:

- **Pahami Konsep Dasar:** Sebelum langsung *coding*, pahami dulu konsep seperti **tabel** (*table*), **kolom** (*column*), **baris** (*row*), relasi, dan **kunci utama** (*primary key*).
- **Kuasi SQL:** **SQL** (*Structured Query Language*) adalah bahasa untuk berkomunikasi dengan database. Anda harus familiar dengan perintah dasar seperti **SELECT**, **INSERT**, **UPDATE**, dan **DELETE**.

- **Latihan:** Praktik adalah kunci. Buatlah studi kasus sederhana seperti membuat database untuk toko buku, perpustakaan, atau data mahasiswa.

5. Konsep Penting dalam Database Relasional

Sebelum membuat tabel, sangat penting untuk memahami cara data diorganisir.

a. Diagram Entitas-Relasi (ERD)

ERD atau **Entity-Relationship Diagram** adalah "peta" atau rancangan database Anda. ERD membantu kita memvisualisasikan bagaimana tabel-tabel (disebut **entitas**) dan kolom-kolomnya saling terhubung.

- **Entitas (Entity):** Merujuk pada objek atau konsep yang ingin kita simpan datanya, seperti **mahasiswa**, **dosen**, atau **mata_kuliah**. Setiap entitas akan menjadi sebuah tabel.
- **Atribut (Attribute):** Adalah karakteristik dari sebuah entitas. Contohnya, entitas **mahasiswa** memiliki atribut **nim**, **nama**, **jurusan**, dan **tanggal_lahir**. Setiap atribut akan menjadi kolom dalam tabel.
- **Relasi (Relationship):** Adalah hubungan antara entitas-entitas. Misalnya, seorang **mahasiswa** mengambil (**mengambil**) sebuah **mata_kuliah**.

b. Jenis-Jenis Relasi (Relationships)

Ada tiga jenis relasi utama yang perlu Anda ketahui:

- **One-to-One (1:1):** Satu baris di tabel A hanya bisa berhubungan dengan satu baris di tabel B. Contoh: Satu **mahasiswa** memiliki satu **kartu_mahasiswa**.
- **One-to-Many (1:M):** Satu baris di tabel A bisa berhubungan dengan banyak baris di tabel B. Contoh: Satu **dosen** bisa mengajar banyak **mata_kuliah**. Ini adalah jenis relasi yang paling umum.
- **Many-to-Many (M:N):** Banyak baris di tabel A bisa berhubungan dengan banyak baris di tabel B. Contoh: Banyak **mahasiswa** bisa mengambil banyak **mata_kuliah**. Relasi ini biasanya diselesaikan dengan membuat tabel ketiga (tabel penghubung).

c. Kunci (Keys)

Kunci adalah atribut atau kombinasi atribut yang mengidentifikasi baris data secara unik dalam sebuah tabel.

- **Primary Key (Kunci Utama):** Atribut yang secara unik mengidentifikasi setiap baris data di dalam tabel. Nilainya tidak boleh kosong (`NULL`) dan harus unik. Contoh: `id` pada tabel `mahasiswa`.
- **Foreign Key (Kunci Asing):** Atribut di satu tabel yang menjadi *Primary Key* di tabel lain. Kunci asing digunakan untuk membuat relasi antar tabel. Contoh: Jika tabel `mata_kuliah` memiliki kolom `id_dosen` untuk menunjukkan siapa dosen pengajarnya, maka `id_dosen` adalah *Foreign Key* yang merujuk pada *Primary Key* di tabel `dosen`.

6. Tipe Data Penting dalam MySQL

Setiap kali Anda membuat kolom dalam tabel, Anda harus menentukan jenis data yang akan disimpan di dalamnya. Memilih tipe data yang tepat sangat penting untuk efisiensi dan integritas data.

Berikut adalah beberapa tipe data yang paling sering digunakan di MySQL:

Kategori	Tipe Data	Deskripsi	Contoh Penggunaan
Numerik	<code>INT</code>	Bilangan bulat (<i>integer</i>).	<code>id, jumlah_barang</code>
	<code>DECIMAL(M,D)</code>	Bilangan desimal dengan presisi tetap. <code>M</code> adalah total digit, <code>D</code> adalah digit di belakang koma.	<code>harga_produk</code>
String	<code>VARCHAR(N)</code>	Teks dengan panjang bervariasi. <code>N</code> adalah jumlah karakter maksimum.	<code>nama, alamat, jurusan</code>

	TEXT	Teks panjang tanpa batasan panjang yang spesifik.	deskripsi_produk, isi_artikel
Tanggal & Waktu	DATE	Tanggal dalam format YYYY-MM-DD.	tanggal_lahir, tanggal_transaksi
	DATETIME	Tanggal dan waktu dalam format YYYY-MM-DD HH:MM:SS.	waktu_login

6. Praktik Hari Ini : Membuat Database dan Tabel

Mari kita coba praktik sederhana untuk memulai. Pastikan Anda sudah menginstal MySQL di komputer Anda.

Langkah 1: Membuat Database Baru

Sebelum membuat tabel, kita harus membuat wadahnya, yaitu database.

```
CREATE DATABASE aa_kampus;
```

Penjelasan: Perintah **CREATE DATABASE** digunakan untuk membuat database baru.

Langkah 2: Menggunakan Database

Setelah dibuat, kita harus memberitahu MySQL database mana yang akan kita gunakan.

```
USE aa_kampus;
```

Penjelasan: Perintah **USE** digunakan untuk memilih database yang akan kita operasikan.

Langkah 3: Membuat Tabel

Sekarang kita buat tabel untuk menyimpan data mahasiswa.

```
CREATE TABLE mahasiswa (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nim VARCHAR(50),
    nama VARCHAR(200),
    jurusan VARCHAR(50),
    tanggal_lahir DATE,
    updated_at DATETIME NULL,
    created_at TIMESTAMP NULL
);
```

Penjelasan:

- **CREATE TABLE mahasiswa**: Perintah untuk membuat tabel bernama **mahasiswa**.
- **id INT PRIMARY KEY AUTO_INCREMENT**: Membuat kolom **id** sebagai kunci utama yang akan otomatis bertambah.
- **nama VARCHAR(50)**: Membuat kolom **nama** dengan tipe data teks dan panjang maksimal 50 karakter.
- **jurusan VARCHAR(50)**: Sama seperti **nama**, untuk menyimpan nama jurusan.
- **tanggal_lahir DATE**: Membuat kolom untuk menyimpan tanggal lahir.

Langkah 4: Memasukkan Data

Kita masukkan data beberapa mahasiswa ke dalam tabel.

```
INSERT INTO mahasiswa (nama, jurusan, tanggal_lahir)
VALUES ('Budi Santoso', 'Teknik Informatika', '2004-03-15');

INSERT INTO mahasiswa (nama, jurusan, tanggal_lahir)
VALUES ('Siti Aminah', 'Sistem Informasi', '2005-08-22');
```

Penjelasan: Perintah **INSERT INTO** digunakan untuk menambahkan baris data ke dalam tabel.

Langkah 5: Melihat Data yang Telah Dimasukkan

Terakhir, mari kita cek apakah data sudah tersimpan dengan benar.

```
SELECT * FROM mahasiswa;
```

Penjelasan: Perintah **SELECT** digunakan untuk mengambil data. Tanda ***** berarti kita ingin menampilkan semua kolom dari tabel **mahasiswa**.