Fondasi Basis Data: Structured Query Language dan Data Definition Language

Muhammad Riduwan Khafidi
Program Studi Teknik Rekayasa Perangkat Lunak
Politeknik Negeri Batam
4342611034@students.polibatam.ac.id

November 5, 2026

Abstrak

Structured Query Language (SQL) merupakan bahasa standar yang digunakan untuk berkomunikasi dengan sistem manajemen basis data relasional (RDBMS). Sejak distandarisasi oleh ANSI, SQL telah menjadi tulang punggung dalam pengelolaan data modern, terbagi menjadi tiga pilar utama: Data Definition Language (DDL), Data Manipulation Language (DML), dan Data Control Language (DCL). Makalah ini membedah secara mendalam konsep DDL, yang berperan sebagai fondasi dalam mendefinisikan struktur basis data. Pembahasan mencakup tipologi tipe data, pembuatan tabel (CREATE), penerapan batasan integritas (Integrity Constraints) seperti Primary Key dan Foreign Key, modifikasi struktur (ALTER), hingga optimalisasi pencarian data menggunakan Index. Selain itu, makalah ini juga menyentuh dasar-dasar manipulasi data (CRUD) untuk memberikan gambaran utuh mengenai siklus hidup data di dalam basis data.

1 Pendahuluan: Bahasa Universal Basis Data

Bayangkan sebuah perusahaan raksasa seperti Oracle, Microsoft, atau Meta. Mereka memiliki sistem basis data yang berbeda-beda, namun para *programmer* mereka dapat berbicara dalam satu bahasa yang sama: **SQL (Structured Query Language)**. Menurut American National Standards Institute (ANSI), SQL adalah bahasa standar untuk sistem manajemen basis data relasional.

Sejarah SQL tidak lepas dari peran Dr. Edgar F. Codd dari IBM pada tahun 1970 yang mencetuskan model relasional. Ide tersebut diwujudkan dalam bentuk bahasa pada tahun 1974, dan pada tahun 1986 distandarisasi oleh ANSI. Meskipun setiap RDBMS (seperti MySQL, PostgreSQL, Oracle, SQL Server) memiliki ekstensi tambahan (dialek) masing-masing, perintah standar seperti `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, dan `DROP` berlaku universal.

2 Tiga Pilar SQL

Dalam praktiknya, SQL tidak hanya digunakan untuk mengambil data, tetapi juga untuk membangun struktur dan mengatur keamanan. Secara garis besar, SQL terbagi menjadi tiga bagian yang saling melengkapi, seperti yang diilustrasikan pada Gambar 1.

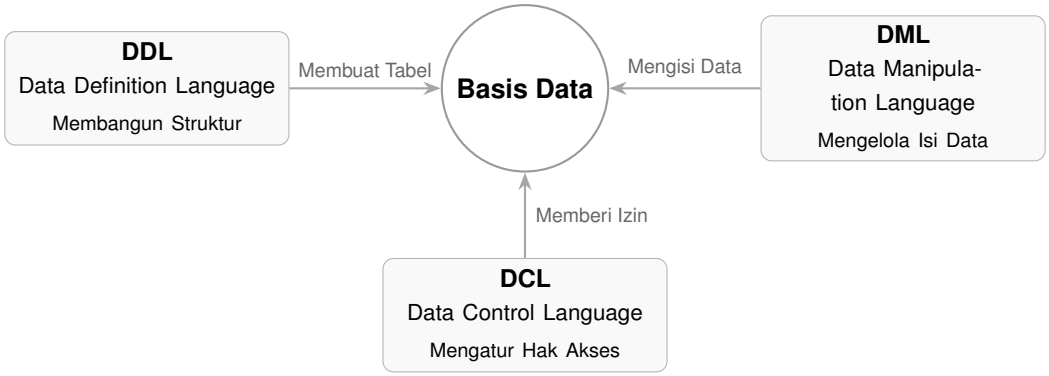


Figure 1. Ilustrasi Tiga Pilar Utama dalam SQL.

Pilar pertama adalah **DDL**, yang berfungsi seperti seorang arsitek. Ia mendefinisikan skema, domain, batasan integritas, hingga struktur penyimpanan fisik. Pilar kedua adalah **DML**, yang berfungsi seperti pekerja. Ia melakukan *query* dan pembaruan data (CRUD). Pilar ketiga adalah **DCL**, yang berfungsi seperti satpam. Ia mengatur siapa saja yang boleh mengakses dan memodifikasi data (misalnya melalui perintah `GRANT` dan `REVOKE`). Makalah ini akan berfokus pada pilar pertama, yaitu DDL.

3 Memahami Tipe Data (Domain Types)

Sebelum membangun tabel, seorang perancang harus memahami tipe data apa yang akan digunakan. Salah satu pertanyaan klasik adalah perbedaan antara `CHAR(n)` dan `VARCHAR(n)`.

`CHAR(n)` adalah *fixed-length string* (panjang tetap). Jika kita mendeklarasikan `CHAR(10)` dan menyimpan kata "du", sistem tetap akan mengalokasikan 10 *byte* memori (kekurangannya diisi spasi). Sementara itu, `VARCHAR(n)` adalah *variable-length string*. Jika kita menyimpan "du" di `VARCHAR(10)`, sistem hanya akan mengalokasikan memori sesuai panjang karakternya (plus sedikit *overhead*). Perbedaan ini berdampak besar pada efisiensi ruang penyimpanan seperti yang terlihat pada tabel di bawah ini.

Table 1. Perbandingan Alokasi Memori `CHAR(10)` vs `VARCHAR(10)`

Data yang Disimpan	**Memori** `CHAR(10)`	**Memori** `VARCHAR(10)`
"du"	10 bytes	2 bytes
"dunia"	10 bytes	5 bytes
"duniaku"	10 bytes	7 bytes

Selain teks, SQL menyediakan tipe angka presisi (`INTEGER`, `NUMERIC(p,d)`), angka desimal mengambang (`REAL`, `FLOAT`), waktu (`DATE`, `TIME`, `TIMESTAMP`), serta tipe `NULL` yang merepresentasikan ketiadaan nilai. Jika sebuah atribut dideklarasikan `NOT NULL`, maka atribut tersebut wajib memiliki nilai.

4 Membangun Struktur: CREATE TABLE

Pilar utama DDL adalah perintah `CREATE TABLE`. Bayangkan kita ingin membuat sistem perbankan sederhana. Kita bisa membuat tabel `branch` (cabang bank) dengan sintaks berikut.

```
CREATE TABLE branch (
    branch_name VARCHAR(15) NOT NULL,
    branch_city VARCHAR(30),
    assets INTEGER,
    PRIMARY KEY (branch_name),
    CHECK (assets >= 0)
);
```

Dalam perintah di atas, kita tidak hanya mendefinisikan kolom dan tipe datanya, tetapi juga menerapkan **Integrity Constraints** (Batasan Integritas):

NOT NULL memastikan kolom `branch_name` tidak boleh kosong. **PRIMARY KEY** menetapkan bahwa `branch_name` harus unik dan menjadi identitas utama baris tersebut. **CHECK** memastikan bahwa nilai `assets` (aset) tidak boleh bernilai negatif. Jika ada upaya memasukkan aset minus, sistem akan menolak secara otomatis.

4.1 Relasi Antar Tabel: Foreign Key dan CASCADE

Dalam basis data relasional, tabel tidak berdiri sendiri. Kita sering menghubungkan tabel `borrower` (peminjam) dengan tabel `customer` dan `loan`.

```
CREATE TABLE borrower (
    customer_name VARCHAR(30),
    loan_number NUMBER(8),
    PRIMARY KEY (customer_name, loan_number),
    FOREIGN KEY (customer_name) REFERENCES customer(customer_name),
    FOREIGN KEY (loan_number) REFERENCES loan(loan_number) ON DELETE
   CASCADE
);
```

Perhatikan penggunaan **FOREIGN KEY** (Kunci Tamu). Ia memastikan bahwa data peminjam yang dimasukkan benar-benar terdaftar di tabel `customer`.

Konsep krusial di sini adalah **ON DELETE CASCADE**. Tanpa pengaturan ini, jika kita menghapus data `customer` dari tabel utama, sistem akan kebingungan karena datanya masih dipakai di tabel `borrower`. Dengan `CASCADE`, jika data `customer` dihapus, maka semua data pinjaman milik *customer* tersebut di tabel `borrower` akan ikut terhapus secara otomatis untuk menjaga integritas data.

5 Modifikasi Struktur: ALTER TABLE

Seiring berjalannya waktu, kebutuhan bisnis berubah. DDL menyediakan perintah `ALTER TABLE` untuk memodifikasi tabel yang sudah ada tanpa harus menghapus dan membuatnya ulang.

Jika kita ingin menambahkan kolom baru (misalnya tanggal lahir peminjam), kita gunakan `ALTER TABLE borrower ADD b_date DATE;`. Jika ingin mengganti nama kolom, gunakan `ALTER TABLE borrower CHANGE b_date birth_date DATE;`. Jika kolom tersebut sudah tidak relevan, hapus dengan `ALTER TABLE borrower DROP birth_date;`. Kita juga dapat menambahkan `DEFAULT` value, misalnya `Gender CHAR(1) DEFAULT '?'` jika gender tidak diisi.

6 Optimalisasi Pencarian: Index

Bayangkan sebuah tabel berisi jutaan data nasabah. Jika kita sering mencari data berdasarkan nama cabang (`branch_name`), proses pencarian akan memakan waktu lama. Di sinilah **INDEX** berperan. Index bekerja seperti indeks di bagian belakang sebuah buku. Sistem tidak perlu membaca seluruh halaman, cukup melihat indeks untuk langsung melompat ke halaman yang dituju.

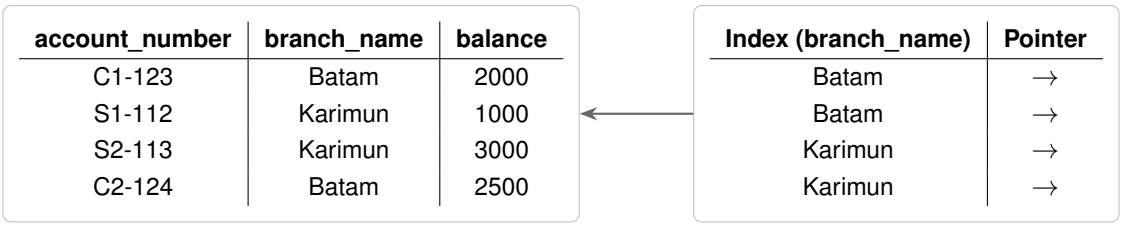


Figure 2. Ilustrasi Index yang Menunjuk Langsung ke Baris Data.

Seperti yang terlihat pada Gambar 2, dengan membuat `CREATE INDEX branch_idx ON Account (branch_name)`, sistem akan membuat daftar terurut. Pencarian kata "Karimun" akan langsung menunjuk ke barisnya tanpa memindai seluruh tabel.

Namun, Index memiliki harga. **Jangan gunakan Index** pada tabel yang sangat kecil, pada kolom yang isinya didominasi oleh nilai `NULL`, atau pada tabel yang sangat sering mengalami operasi `UPDATE` dan `INSERT` secara masif. Setiap kali data ditambahkan atau diubah, sistem harus memperbarui Index-nya juga, yang justru memperlambat proses penulisan.

7 Data Manipulation (CRUD) sebagai Pelengkap

Setelah struktur (DDL) selesai dibangun, data di dalamnya diisi dan dikelola menggunakan DML. Operasi dasar CRUD (*Create, Read, Update, Delete*) dijalankan melalui perintah berikut.

Untuk menambah data baru, digunakan `INSERT INTO account VALUES ('A-9732',`

'Perryridge', 1200);. Jika urutan kolom tidak diingat, kita bisa spesifikasikan: INSERT INTO account (branch_name, balance, account_number) VALUES ('Perryridge', 1200, 'A-9732');.

Untuk menghapus data spesifik, perintah DELETE FROM account WHERE branch_name = 'Perryridge'; akan menghapus baris dengan kondisi tertentu. Namun, perintah DELETE FROM account; tanpa klausa WHERE akan mengosongkan seluruh tabel.

Adapun untuk mengubah data, digunakan UPDATE. Misalnya, jika bank ingin memberikan bunga 5% untuk nasabah dengan saldo di atas 1000, perintahnya adalah UPDATE account SET balance = balance * 1.05 WHERE balance > 1000;.

Menariknya, SQL juga mendukung logika percabangan menggunakan CASE. Jika bank ingin memberikan bunga 6% untuk saldo di atas 1000, dan 5% untuk saldo di bawahnya, hal ini bisa dilakukan dalam satu perintah *update*:

```
UPDATE account
SET balance = CASE
    WHEN balance <= 1000 THEN balance * 1.05
    ELSE balance * 1.06
END;
```

8 Kesimpulan

Data Definition Language (DDL) adalah fondasi yang menentukan seberapa kokoh sebuah sistem basis data. Melalui pemilihan tipe data yang tepat (seperti CHAR vs VARCHAR), penerapan *constraints* yang ketat (seperti PRIMARY KEY, FOREIGN KEY, dan CASCADE), serta strategi *Indexing* yang bijak, seorang perancang dapat memastikan bahwa data tidak hanya tersimpan dengan aman, tetapi juga dapat diakses dengan sangat cepat. Penguasaan DDL, yang dilengkapi dengan pemahaman dasar DML, adalah bekal mutlak bagi setiap *software engineer* dalam membangun aplikasi berskala besar.

Referensi

References

[1] Lubis, A. I. (2026). *Data Definition Language* [Slide Presentasi Mata Kuliah Pengantar Basis Data]. Politeknik Negeri Batam.

[2] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.