Fondasi Basis Data: Data Manipulation Language (DML)

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

November 5, 2026

Abstrak

Jika *Data Definition Language* (DDL) berfungsi membangun struktur atau kerangka tabel, maka *Data Manipulation Language* (DML) adalah pilar yang bertugas mengelola isi dari kerangka tersebut. Makalah ini membahas secara komprehensif perintah-perintah dasar DML dalam SQL, dengan fokus utama pada klausa `SELECT`, `FROM`, dan `WHERE`. Pembahasan mencakup teknik penyaringan data (*filtering*), penggunaan operator logika, pencarian berbasis pola teks (`LIKE`), penggunaan fungsi agregat untuk meringkas data, hingga konsep penggabungan dua tabel (*join*) menggunakan klausa `WHERE`. Pemahaman mendalam mengenai DML sangat krusial bagi seorang *software engineer* untuk dapat berinteraksi secara dinamis dengan basis data.

1 Pendahuluan: Struktur Dasar Query

Dalam SQL, mengambil atau memanipulasi data dilakukan melalui sebuah *query* (kueri). Struktur dasar dari kueri SQL dapat dirumuskan sebagai berikut:

$$\texttt{SELECT}\ A_1, A_2, \ldots, A_n\ \texttt{FROM}\ R_1, R_2, \ldots, R_m\ \texttt{WHERE}\ P$$

Di mana:

- A_i adalah *selected field* (kolom yang ingin ditampilkan). Tanda bintang (*) berarti menampilkan semua kolom.
- R_i adalah nama tabel yang akan diambil datanya.
- P adalah *requirement condition* (kondisi yang harus dipenuhi oleh data).

Sebagai contoh, kita memiliki beberapa tabel perbankan: `Account`, `Deposit`, `Check`, `ATMWithdrawal`, dan `Customer`. Misalkan kita memiliki tabel `ATMWithdrawal` seperti di bawah ini:

Table 1. Tabel `ATMWithdrawal`

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/1/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/1/2000 10:05:00
4	2	100	$40.00	11/1/2000 10:07:00
5	2	100	$200.00	11/8/2000 14:14:00

Jika kita ingin menampilkan kolom `AcctNo` dan `Amount` saja, kita menggunakan kueri dasar:

```
SELECT AcctNo, Amount
FROM ATMWithdrawal;
```

2 Filtering Data dengan WHERE

Klausa `WHERE` digunakan untuk menyaring data sehingga hanya baris yang memenuhi kondisi tertentu yang akan ditampilkan. Contohnya, jika kita ingin menampilkan semua atribut dari tabel `ATMWithdrawal` di mana nilai `Amount` kurang dari 50:

```
SELECT *
FROM ATMWithdrawal
WHERE Amount < 50;
```

Kueri di atas akan memeriksa setiap baris. Baris dengan Amount $25.00 dan $40.00 akan diloloskan, sedangkan baris $150.00 dan $200.00 akan diabaikan.

3 Ekstensi Klausa SELECT

SQL menyediakan berbagai ekstensi untuk memperkaya hasil kueri `SELECT`:

3.1 DISTINCT dan ALL

Secara default, SQL memperbolehkan adanya duplikasi data pada hasil kueri. Untuk menghilangkan duplikasi, gunakan kata kunci `DISTINCT` setelah `SELECT`. Sebaliknya, `ALL` secara eksplisit mempertahankan duplikasi (meskipun ini adalah perilaku *default*).

```
SELECT DISTINCT AcctNo, Amount FROM Deposit;
```

3.2 Ekspresi Aritmatika dan Alias (AS)

Kita dapat melakukan operasi matematika (+, -, *, /) pada kolom yang bertipe numerik. Selain itu, kita bisa memberikan nama baru pada kolom hasil (alias) menggunakan kata kunci `AS`.

```sql
SELECT AcctNo AS Number, Amount * 10 AS Amt
FROM ATMWithdrawal
WHERE Amount < 50;
```

Pada kueri di atas, nilai `Amount` akan dikalikan 10, dan judul kolom pada hasilnya akan berubah menjadi `Number` dan `Amt`.

3.3 Fungsi Agregat (Aggregate Functions)

Fungsi agregat digunakan untuk melakukan perhitungan terhadap sekumpulan nilai dan mengembalikan satu nilai tunggal. Nilai yang diolah harus numerik.

- `avg`: Nilai rata-rata
- `min`: Nilai minimum
- `max`: Nilai maksimum
- `sum`: Jumlah total nilai
- `count`: Jumlah banyaknya data

Contoh penggunaan:

```sql
SELECT MIN(Balance), MAX(Balance), AVG(Balance) FROM Account;

SELECT count(*) FROM Customer;

SELECT avg(Amount) FROM Check WHERE Amount < 50;
```

4 Ekstensi Klausa WHERE (Operator Lanjutan)

Selain operator perbandingan standar (`=`, `>`, `>=`, `<`, `<=`, `<>`), klausa `WHERE` mendukung berbagai operator logika dan pencocokan string.

4.1 Operator Logika

Gunakan `AND`, `OR`, dan `NOT` untuk menggabungkan beberapa kondisi.

```sql
SELECT Number, CustID
FROM Account
WHERE Balance > 1000 AND TYPE = 'checking';
```

4.2 Pengecekan Nilai Kosong (NULL)

Gunakan `IS NULL` atau `IS NOT NULL` untuk mencari data yang kosong atau tidak kosong. (Catatan: Jangan gunakan `= NULL`).

```
SELECT * FROM Customer WHERE Address IS NULL;
SELECT * FROM Customer WHERE Phone IS NOT NULL;
```

4.3 Operator String: LIKE

Operator `LIKE` digunakan untuk pencarian berbasis pola. Terdapat dua *wildcard* utama:

- `%` : Mewakili nol atau beberapa karakter.
- `_` : Mewakili tepat satu karakter.

```
-- Nama berawalan 'Fr'
SELECT * FROM Customer WHERE Name LIKE 'Fr%';

-- Nama yang mengandung 're'
SELECT * FROM Customer WHERE Name LIKE '%re%';

-- Nama berawalan 'Fr', diikuti 2 karakter, lalu 're'
SELECT * FROM Customer WHERE Name LIKE 'Fr__re';
```

4.4 Operator IN dan BETWEEN

`IN` digunakan untuk mempersingkat penulisan beberapa kondisi `OR`. `BETWEEN` digunakan untuk mencari nilai dalam sebuah rentang (inklusif).

```
-- Menggunakan OR (Panjang)
SELECT * FROM Deposit WHERE AcctNo = '102' OR AcctNo = '104';
-- Menggunakan IN (Ringkas)
SELECT * FROM Deposit WHERE AcctNo IN ('102', '104');

-- Menggunakan BETWEEN untuk rentang tanggal
SELECT * FROM ATMwithdrawal
WHERE WithdrawDate BETWEEN '2008-04-01' AND '2008-04-15';
```

5 Query dengan Dua Tabel (FROM Extension)

Dalam basis data relasional, data sering kali tersebar di beberapa tabel. Kita bisa menggabungkan dua tabel di dalam klausa `FROM`. Jika kita menuliskan `SELECT * FROM Account, Deposit`, sistem akan menghasilkan **Kombinasi Kartesian** (Cartesian Product), yaitu semua kemungkinan pasangan baris dari kedua tabel.

Agar hasilnya masuk akal, kita harus memfilter hasil perkawinan silang tersebut menggunakan klausa `WHERE` sebagai jembatan penghubung (yang saat ini kita kenal sebagai *JOIN*).

Misalkan kita memiliki tabel `Account` dan `Deposit`:

Table 2. Tabel `Account`

Number	Owner	Balance
101	J. Smith	1000.00
102	W. Wei	2000.00
103	J. Smith	5000.00
104	M. Jones	1000.00
105	H. Martin	10000.00

Table 3. Tabel `Deposit`

Account	T-id	Amount
102	1	500.00
102	2	200.00
104	3	1000.00
105	4	10000.00

Jika kita ingin mencari nama pemilik akun (`Owner`) dan saldonya (`Balance`) yang pernah melakukan deposit dengan nominal di atas 1000, kita gunakan kueri berikut:

```
SELECT A.Owner, A.Balance
FROM Account A, Deposit D
WHERE D.Account = A.Number AND D.Amount > 1000;
```

Penjelasan:

1. `A` dan `D` adalah *alias* untuk tabel `Account` dan `Deposit` agar penulisan lebih singkat.
2. `D.Account = A.Number` adalah kunci penghubung (relasi) antara kedua tabel.
3. `D.Amount > 1000` adalah filter untuk nominal deposit.

Mesin basis data akan memeriksa setiap kombinasi baris. Jika nomor akun cocok dan nominal deposit lebih dari 1000, baris tersebut akan ditampilkan. Jika tidak cocok (misalnya akun 101 tidak ada di tabel deposit), maka kombinasi tersebut diabaikan (*Ignore*). Hasil akhirnya adalah data milik W. Wei (Deposit $10.000) dan H. Martin (Deposit $10.000).

6 Kesimpulan

Data Manipulation Language (DML) memberikan kekuatan bagi pengguna untuk mengekstraksi informasi berharga dari tumpukan data mentah. Penguasaan klausa `SELECT`, `FROM`, dan `WHERE` adalah fondasi mutlak. Dengan memadukan operator logika, pencarian pola, fungsi agregat, dan penggabungan antar tabel, seorang *software engineer* dapat membangun kueri yang kompleks namun efisien untuk memenuhi kebutuhan bisnis atau aplikasi yang sedang dikembangkan.

Referensi

References

[1] Lubis, A. I. (2026). *SQL: Data Manipulation Language (DML)* [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.