MATERI 3: QUERY LANJUT PADA POSTGRESQL
Kuliah: Basisdata Lanjut

Tujuan Pembelajaran

Setelah menyelesaikan materi ini, mahasiswa diharapkan mampu:

1. Memahami konsep dasar operasi SELECT dalam PostgreSQL;
2. Menerapkan berbagai jenis JOIN untuk menggabungkan data dari multiple tables;
3. Menggunakan fungsi agregasi untuk analisis data;
4. Memahami perbedaan sintaks PostgreSQL dengan database lainnya;
5. Mengimplementasikan CTE (Common Table Expression) untuk menyederhanakan query kompleks.

Konsep Dasar

Materi ini membahas inti dari operasi pembacaan data (query) dalam SQL, yang bertujuan untuk mengambil, menggabungkan, dan meringkas informasi dari database relasional seperti PostgreSQL. Konsep utamanya meliputi:

- **SELECT**: Perintah fundamental untuk memilih dan mengambil data dari satu atau lebih tabel.
- **JOIN**: Teknik untuk menggabungkan baris dari dua atau lebih tabel berdasarkan kolom yang saling terkait (relationship).
- **Agregasi**: Metode untuk melakukan perhitungan pada sekelompok data untuk menghasilkan nilai ringkasan tunggal (seperti jumlah, rata-rata, maksimum).
- **CTE (Common Table Expression)**: Ekspresi query sementara yang meningkatkan keterbacaan dan struktur query kompleks, membuatnya mirip dengan membuat "tabel virtual" sementara dalam satu perintah.

Manfaat

Menguasai materi ini memberikan manfaat yang sangat besar, yaitu:

- **Efisiensi**: Dapat mengambil informasi yang tepat dan bermakna dari laut data dengan cepat dan akurat.
- **Kemampuan Analitis**: Mampu menjawab pertanyaan bisnis yang kompleks dengan menggabungkan data dari berbagai sumber dan meringkasnya.
- **Keterbacaan & Pemeliharaan**: Penggunaan CTE dan teknik query yang terstruktur membuat kode SQL lebih mudah dibaca, didebug, dan dipelihara oleh diri sendiri atau orang lain.
- **Portabilitas Skill**: Memahami perbedaan sintaks (seperti LIMIT/OFFSET) memudahkan adaptasi untuk bekerja dengan berbagai jenis database (PostgreSQL, MySQL, SQL Server, dll.).

TEORI DASAR

1. Konsep Dasar SELECT

Pengertian SELECT: SELECT adalah perintah fundamental SQL untuk mengambil data dari satu atau lebih tabel dalam database. Perintah ini memungkinkan kita untuk menyeleksi kolom tertentu, menerapkan filter, mengurutkan hasil, dan membatasi jumlah data yang ditampilkan.

Contoh dasar:

```
SELECT kolom1, kolom2, kolom3
FROM nama_tabel
WHERE kondisi;
```

2. Perbedaan LIMIT/OFFSET: PostgreSQL vs MySQL

PostgreSQL:

```
SELECT * FROM produk ORDER BY harga DESC LIMIT 10 OFFSET 20;
-- Mengambil 10 data, melewati 20 data pertama
```

-- Mengambil 10 data, melewati 20 data pertama

MySQL:

```
SELECT * FROM produk ORDER BY harga DESC LIMIT 20, 10;
-- Format: LIMIT offset, jumlah
-- Mengambil 10 data, melewati 20 data pertama
```

-- Format: LIMIT offset, jumlah
-- Mengambil 10 data, melewati 20 data pertama

Perbedaan utama:

- PostgreSQL menggunakan kata kunci terpisah: LIMIT dan OFFSET
- MySQL menggunakan sintaks gabungan: LIMIT offset, jumlah
- PostgreSQL lebih eksplisit dan mudah dibaca untuk pagination kompleks

3. Konsep JOIN

JOIN digunakan untuk menggabungkan baris dari dua atau lebih tabel berdasarkan kolom yang terkait antara tabel-tabel tersebut.

a. INNER JOIN

Menampilkan hanya baris yang memiliki nilai yang cocok di kedua tabel.

```
SELECT orders.id, customers.nama, orders.tanggal
FROM orders
INNER JOIN customers ON orders.customer_id = customers.id;
```

b. LEFT JOIN

Menampilkan semua baris dari tabel kiri (pertama), dan baris yang cocok dari tabel kanan. Jika tidak ada kecocokan, hasilnya NULL dari sisi kanan.

```
SELECT customers.nama, orders.id
FROM customers
LEFT JOIN orders ON customers.id = orders.customer_id;
```

c. RIGHT JOIN

Kebalikan dari LEFT JOIN. Menampilkan semua baris dari tabel kanan, dan baris yang cocok dari tabel kiri.

```
SELECT orders.id, customers.nama
FROM orders
RIGHT JOIN customers ON orders.customer_id = customers.id;
```

d. FULL OUTER JOIN

Menampilkan semua baris ketika ada kecocokan di salah satu tabel. Jika tidak ada kecocokan, kolom dari tabel tanpa kecocokan akan berisi NULL.

```
SELECT customers.nama, orders.id
FROM customers
FULL OUTER JOIN orders ON customers.id = orders.customer_id;
```

4. Fungsi Agregasi

Fungsi agregasi melakukan perhitungan pada sekumpulan nilai dan mengembalikan nilai tunggal. Sering digunakan dengan klausa GROUP BY.

Fungsi Agregasi Umum:

- COUNT(): Menghitung jumlah baris
- SUM(): Menjumlahkan nilai
- AVG(): Menghitung rata-rata
- MAX(): Mencari nilai tertinggi
- MIN(): Mencari nilai terendah

GROUP BY: Mengelompokkan baris yang memiliki nilai yang sama ke dalam summary rows.

```
SELECT department_id, COUNT(*) as jumlah_karyawan
FROM employees
GROUP BY department_id;
```

HAVING: Memfilter hasil agregasi (setelah GROUP BY), berbeda dengan WHERE yang memfilter sebelum agregasi.

```sql
SELECT department_id, AVG(gaji) as rata_gaji
FROM employees
GROUP BY department_id
HAVING AVG(gaji) > 5000000;
```

Perbedaan WHERE vs HAVING:

- WHERE memfilter baris sebelum pengelompokan
- HAVING memfilter hasil setelah pengelompokan

5. CTE (Common Table Expression)

Konsep WITH Clause

CTE memungkinkan kita membuat query sementara yang dapat dirujuk dalam query utama, membuat query kompleks menjadi lebih modular dan mudah dibaca.

Struktur dasar:

```sql
WITH nama_cte AS (
    SELECT ... FROM ... WHERE ...
)
SELECT * FROM nama_cte;
```

Contoh Implementasi CTE

```sql
WITH department_stats AS (
    SELECT
        department_id,
        COUNT(*) as total_karyawan,
        AVG(gaji) as rata_gaji
    FROM employees
    GROUP BY department_id
)
SELECT
    d.nama_department,
    ds.total_karyawan,
    ds.rata_gaji
FROM department_stats ds
JOIN departments d ON ds.department_id = d.id
WHERE ds.rata_gaji > 5000000;
```

Perbandingan CTE vs Subquery di MySQL/PostgreSQL

CTE:

-- Lebih mudah dibaca dan dimaintenance

```sql
-- Lebih mudah dibaca dan dimaintenance
WITH high_salary_employees AS (
    SELECT * FROM employees WHERE gaji > 8000000
)
SELECT * FROM high_salary_employees WHERE department_id = 5;
```

Subquery:

-- Lebih kompleks untuk query bertingkat

```sql
-- Lebih kompleks untuk query bertingkat
SELECT * FROM (
    SELECT * FROM employees WHERE gaji > 8000000
) AS high_salary_employees
WHERE department_id = 5;
```

Keunggulan CTE:

1. **Keterbacaan**: Struktur lebih jelas dan terorganisir
2. **Reusability**: Dapat dirujuk multiple times dalam query yang sama
3. **Maintainability**: Lebih mudah di-debug dan dimodifikasi
4. **Recursive Queries**: Mendukung query rekursif (hanya di CTE)

Menggunakan CTE untuk Query Kompleks

```sql
WITH
sales_per_month AS (
    SELECT
        EXTRACT(YEAR FROM tanggal) as tahun,
        EXTRACT(MONTH FROM tanggal) as bulan,
        SUM(jumlah) as total_penjualan
    FROM penjualan
    GROUP BY tahun, bulan
),
employee_performance AS (
    SELECT
        karyawan_id,
        COUNT(*) as total_transaksi,
        SUM(jumlah) as total_penjualan
    FROM penjualan
    GROUP BY karyawan_id
)
SELECT
    spm.tahun,
    spm.bulan,
    spm.total_penjualan,
    ep.nama_karyawan,
    ep.total_transaksi
FROM sales_per_month spm
JOIN (
    SELECT
        e.id,
        e.nama as nama_karyawan,
        ep.total_transaksi
    FROM employee_performance ep
    JOIN karyawan e ON ep.karyawan_id = e.id
) ep ON 1=1
ORDER BY spm.tahun, spm.bulan;
```

Best Practices

1. **SELECT Spesifik**: Hindari SELECT *, sebutkan kolom explicitly
2. **Indexing**: Pastikan kolom yang di-JOIN dan di-WHERE ter-index
3. **LIMIT untuk Testing**: Gunakan LIMIT saat mengembangkan query kompleks
4. **CTE untuk Kompleksitas**: Gunakan CTE untuk query yang memiliki multiple subqueries
5. **EXPLAIN ANALYZE**: Gunakan untuk menganalisis performance query

Dengan memahami konsep-konsep dasar ini, mahasiswa akan memiliki fondasi yang kuat untuk melakukan query data yang efektif dan efisien dalam PostgreSQL.

Konsep-konsep ini adalah tulang punggung dari hampir semua sistem yang berurusan dengan data:

1. **Laporan dan Dashboard**: Membuat laporan penjualan per bulan, rata-rata nilai pelanggan, atau jumlah produk yang terjual per kategori (menggunakan **Agregasi** dan **GROUP BY**).

2. **Sistem E-commerce**: Menampilkan daftar pesanan beserta detail informasi pelanggan dan produknya (menggunakan **JOIN** antara tabel orders, customers, dan products).
3. **Media Sosial**: Menampilkan feed berita yang berisi postingan dari orang yang diikuti (menggunakan **JOIN** pada tabel users, posts, dan follows).
4. **Analisis Data Kompleks**: Membersihkan data, melakukan transformasi multi-tahap, dan menghitung metrik bisnis yang rumit sebelum ditampilkan (menggunakan **CTE** untuk menyederhanakan setiap tahapan).