JOBSHEET

BASIS DATA LANJUT

Jurusan Teknologi Informasi

POLITEKNIK NEGERI MALANG

TAHUN AJARAN 2025/2026

PERTEMUAN 3

Query Lanjut (SELECT, JOIN, CTE) pada PostgreSQL

Team Teaching Basis Data Lanjut:

- Candra Bella Vista, S.Kom., MT.
- Moch Zawaruddin Abdullah, S.ST., M.Kom.
- Yan Watequlis Syaifudin, ST., MMT., PhD.
- Yoppy Yunhasnawa, S.ST., M.Sc.

Mata Kuliah	:	Basis Data Lanjut
Program Studi	:	D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester	:	3 (tiga)
Pertemuan ke-	:	3

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.

```
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:

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

Contoh Implementasi CTE

```
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

```
-- 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).

JOBSHEET PRAKTIKUM

SETUP AWAL

1. Install PostgreSQL

- Download dan install PostgreSQL dari https://www.postgresql.org/download/
- Install pgAdmin (GUI tool) atau gunakan psql (command line tool)

2. Login ke Database

```
psql -U username -d database_name
```

3. Buat Database Baru:

```
-- 2. Buat database baru
CREATE DATABASE company_db;
\c company_db;
```

4. Buat Tabel-tabel:

```
-- 3. Buat tabel-tabel
CREATE TABLE departments (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(100)
);

CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    salary NUMERIC(10,2),
    department_id INTEGER REFERENCES departments(id),
    hire_date DATE
);

CREATE TABLE projects (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    budget NUMERIC(15,2),
    department_id INTEGER REFERENCES departments(id)
);

CREATE TABLE employee_projects (
    employee_id INTEGER REFERENCES employees(id),
    project_id INTEGER REFERENCES projects(id),
    hours_worked NUMERIC(5,2),
    assignment_date DATE DEFAULT CURRENT_DATE,
    PRIMARY KEY (employee_id, project_id)
);
```

5. Insert Data Sample:

```sql
-- Data departments
INSERT INTO departments (name, location) VALUES
('IT', 'Jakarta'),
('HR', 'Bandung'),
('Finance', 'Surabaya'),
('Marketing', 'Medan'),
('Operations', 'Yogyakarta');

-- Data employees
INSERT INTO employees (name, salary, department_id, hire_date) VALUES
('Budi Santoso', 8000000, 1, '2022-01-15'),
('Siti Rahayu', 7500000, 1, '2022-03-20'),
('Ahmad Fauzi', 9000000, 2, '2021-11-10'),
('Dewi Anggraini', 8500000, 2, '2022-02-28'),
('Rudi Hermawan', 9500000, 3, '2021-08-05'),
('Maya Sari', 8800000, 3, '2022-04-15'),
('Hendra Pratama', 8200000, 4, '2022-05-20'),
('Lina Marlina', 7800000, 4, '2022-06-10'),
('Joko Widodo', 9200000, 1, '2021-12-01'),
('Ratna Dewi', 8700000, 3, '2022-03-15'),
('Fajar Nugroho', 0, NULL, '2023-01-01'),  -- Employee tanpa department
('Bambang Sutoyo', 6500000, 5, '2022-07-01');
```

```sql
-- Data projects
INSERT INTO projects (name, budget, department_id) VALUES
('Website Development', 50000000, 1),
('Mobile App', 75000000, 1),
('Recruitment System', 30000000, 2),
('Payroll System', 45000000, 3),
('Marketing Campaign', 60000000, 4),
('Database Migration', 35000000, 1),
('Inventory System', 40000000, NULL),  -- Project tanpa department
('CRM Implementation', 55000000, 5);

-- Data employee_projects
INSERT INTO employee_projects (employee_id, project_id, hours_worked, assignment_date) VALUES
(1, 1, 120.5, '2023-01-15'),
(1, 6, 80.0, '2023-02-01'),
(2, 1, 95.5, '2023-01-20'),
(2, 2, 150.0, '2023-02-15'),
(3, 3, 200.0, '2023-03-01'),
(4, 3, 180.5, '2023-03-05'),
(5, 4, 220.0, '2023-04-01'),
(6, 4, 190.5, '2023-04-05'),
(7, 5, 175.0, '2023-05-01'),
(8, 5, 160.5, '2023-05-10'),
(9, 2, 140.0, '2023-06-01'),
(9, 6, 90.5, '2023-06-15'),
(10, 4, 210.0, '2023-07-01'),
(11, 7, 100.0, '2023-07-15'),
(12, 8, 180.0, '2023-08-01');
```

Studi Kasus: Sistem Manajemen Perusahaan XYZ

Database perusahaan dengan berbagai entitas: departments, employees, projects, dan hubungan many-to-many antara employees dan projects.

Praktikum 1: SELECT dengan LIMIT/OFFSET

Konsep Dasar

- **LIMIT**: Membatasi jumlah baris yang ditampilkan
- **OFFSET**: Melewati sejumlah baris tertentu
- **Perbedaan dengan MySQL**: PostgreSQL menggunakan LIMIT x OFFSET y vs MySQL LIMIT y, x

Latihan Praktis

1. Basic SELECT dengan LIMIT

```
SELECT * FROM employees LIMIT 5;
```

2. SELECT dengan ORDER BY dan LIMIT/OFFSET

```
SELECT name, salary FROM employees
ORDER BY salary DESC
LIMIT 3 OFFSET 2;
```

3. SELECT dengan WHERE dan LIMIT

```
SELECT * FROM employees
WHERE salary > 8000000
LIMIT 4;
```

Soal Latihan Praktikum 1

1. Tampilkan 5 employee dengan gaji terendah
2. Tampilkan halaman 2 data projects (3 data per halaman)
3. Tampilkan 3 employee yang dihire paling akhir

Query Mahasiswa:

Hasil Eksekusi:

Praktikum 2: JOIN Operations

Jenis-Jenis JOIN

- **INNER JOIN**: Data yang match di kedua tabel
- **LEFT JOIN**: Semua data dari tabel kiri + match dari kanan
- **RIGHT JOIN**: Semua data dari tabel kanan + match dari kiri
- **FULL OUTER JOIN**: Semua data dari kedua tabel

Latihan Praktis

1. INNER JOIN

```
SELECT e.name, d.name as department, e.salary
FROM employees e
INNER JOIN departments d ON e.department_id = d.id;
```

2. LEFT JOIN

```
SELECT e.name, d.name as department
FROM employees e
LEFT JOIN departments d ON e.department_id = d.id;
```

3. RIGHT JOIN

```
SELECT d.name as department, COUNT(e.id) as employee_count
FROM departments d
RIGHT JOIN employees e ON d.id = e.department_id
GROUP BY d.name;
```

4. FULL OUTER JOIN

```
SELECT e.name as employee, d.name as department
FROM employees e
FULL OUTER JOIN departments d ON e.department_id = d.id;
```

5. Multiple JOIN

```
SELECT e.name, p.name as project, ep.hours_worked
FROM employees e
INNER JOIN employee_projects ep ON e.id = ep.employee_id
INNER JOIN projects p ON ep.project_id = p.id;
```

Soal Latihan Praktikum 2

1. Tampilkan semua projects beserta department penanggung jawabnya
2. Tampilkan employee yang tidak memiliki department
3. Tampilkan department yang tidak memiliki employee
4. Tampilkan employee yang bekerja di project 'Website Development'

Query Mahasiswa:

Hasil Eksekusi:

Praktikum 3: Fungsi Agregasi

Konsep Dasar

- **GROUP BY**: Mengelompokkan data berdasarkan kolom tertentu
- **HAVING**: Filter hasil agregasi (setelah GROUP BY)
- **Fungsi Agregasi**: COUNT, SUM, AVG, MAX, MIN

Latihan Praktis

1. Basic Agregasi

```sql
SELECT
    COUNT(*) as total_employees,
    AVG(salary) as avg_salary,
    MAX(salary) as max_salary
FROM employees;
```

2. GROUP BY

```sql
SELECT
    d.name as department,
    COUNT(e.id) as employee_count,
    AVG(e.salary) as avg_salary
FROM departments d
LEFT JOIN employees e ON d.id = e.department_id
GROUP BY d.name;
```

3. HAVING

```sql
SELECT
    d.name as department,
    COUNT(e.id) as employee_count
FROM departments d
LEFT JOIN employees e ON d.id = e.department_id
GROUP BY d.name
HAVING COUNT(e.id) > 2;
```

4. Multiple Agregasi

```sql
SELECT
    p.name as project,
    SUM(ep.hours_worked) as total_hours,
    COUNT(ep.employee_id) as total_employees,
    AVG(ep.hours_worked) as avg_hours_per_employee
FROM projects p
LEFT JOIN employee_projects ep ON p.id = ep.project_id
GROUP BY p.name;
```

Soal Latihan Praktikum 3

1. Tampilkan total budget projects per department
2. Tampilkan department dengan rata-rata gaji di atas 8.5 juta
3. Tampilkan project dengan total jam kerja lebih dari 200 jam
4. Tampilkan employee dengan total jam kerja terbanyak

Query Mahasiswa:

Hasil Eksekusi:

Praktikum 4: Common Table Expression (CTE)

Konsep Dasar

- **CTE**: Tabel sementara yang hanya ada selama eksekusi query
- **Keunggulan**: Lebih mudah dibaca dan dimaintain dibanding subquery
- **Syntax**: WITH cte_name AS (SELECT ...) SELECT * FROM cte_name

Latihan Praktis

1. Basic CTE

```sql
WITH high_salary_employees AS (
    SELECT * FROM employees WHERE salary > 8500000
)
SELECT * FROM high_salary_employees ORDER BY salary DESC;
```

2. Multiple CTEs

```sql
WITH dept_stats AS (
    SELECT
        department_id,
        COUNT(*) as emp_count,
        AVG(salary) as avg_salary
    FROM employees
    GROUP BY department_id
),
project_stats AS (
    SELECT
        department_id,
        COUNT(*) as project_count
    FROM projects
    GROUP BY department_id
)
SELECT
    d.name as department,
    ds.emp_count,
    ds.avg_salary,
    ps.project_count
FROM departments d
LEFT JOIN dept_stats ds ON d.id = ds.department_id
LEFT JOIN project_stats ps ON d.id = ps.department_id;
```

3. CTE vs Subquery

```sql
-- Subquery (lebih rumit)
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

-- CTE (lebih mudah dibaca)
WITH avg_salary AS (
    SELECT AVG(salary) as avg_sal FROM employees
)
SELECT name, salary
FROM employees, avg_salary
WHERE salary > avg_sal;
```

4. CTE Complex Query

```sql
WITH employee_project_stats AS (
    SELECT
        e.name as employee_name,
        COUNT(ep.project_id) as total_projects,
        SUM(ep.hours_worked) as total_hours
    FROM employees e
    LEFT JOIN employee_projects ep ON e.id = ep.employee_id
    GROUP BY e.name
)
SELECT
    employee_name,
    total_projects,
    total_hours,
    CASE
        WHEN total_hours > 200 THEN 'High'
        WHEN total_hours > 100 THEN 'Medium'
        ELSE 'Low'
    END as workload_level
FROM employee_project_stats
ORDER BY total_hours DESC;
```

Soal Latihan Praktikum 4

1. Buat CTE untuk menampilkan top 3 employee dengan gaji tertinggi
2. Buat CTE untuk menghitung rata-rata jam kerja per department
3. Buat CTE untuk menampilkan project dengan budget di atas rata-rata
4. Buat CTE berjenjang (multiple CTEs) untuk analisis department lengkap

Query Mahasiswa:

Hasil Eksekusi:

Tugas Akhir Integrasi

Buat query lengkap yang menggabungkan semua konsep untuk menjawab pertanyaan berikut:

1. Tampilkan 5 department dengan total gaji tertinggi
2. Tampilkan project dengan rata-rata jam kerja per employee tertinggi

3. Tampilkan employee yang bekerja di lebih dari 1 project
4. Buat laporan performance department: nama department, jumlah employee, jumlah project, total budget projects, rata-rata gaji, total jam kerja

Query Mahasiswa:

Hasil Eksekusi:

Refleksi Pembelajaran

Nama Mahasiswa: _____
NIM: _____

Hal yang sudah dipahami:

- SELECT dengan LIMIT/OFFSET
- Berbagai jenis JOIN
- Fungsi agregasi dengan GROUP BY dan HAVING
- Penggunaan CTE untuk query kompleks

Kesulitan yang dialami: