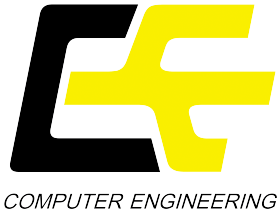

Modul Praktikum
Sistem Manajemen Basis Data

ACID dan High Concurrency PostgreSQL

Departemen Teknik Komputer
Institut Teknologi Sepuluh Nopember
Surabaya

Penulis: Arta Kusuma Hernanda

2025

Kata Pengantar

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga Modul Praktikum Basis Data: ACID dan High Concurrency dengan PostgreSQL ini dapat tersusun hingga selesai. Modul ini disusun untuk memenuhi kebutuhan mahasiswa dalam pembelajaran mata kuliah Basis Data di Departemen Teknik Komputer, Institut Teknologi Sepuluh Nopember Surabaya.

Modul praktikum ini bertujuan untuk memberikan pemahaman mendalam tentang konsep ACID (Atomicity, Consistency, Isolation, Durability) dan High Concurrency pada database PostgreSQL melalui empat skenario praktis. Melalui modul ini, mahasiswa diharapkan dapat memahami dan mempraktikkan berbagai teknik optimasi database seperti indexing, transactions, partitioning, serta strategi backup dan recovery.

Kami menyadari bahwa modul praktikum ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan demi perbaikan modul ini di masa yang akan datang.

Akhir kata, kami berharap modul praktikum ini dapat bermanfaat bagi mahasiswa dalam mengembangkan kemampuan dalam pengelolaan basis data dan dapat menjadi bekal yang berguna di dunia kerja nantinya.

Surabaya, April 2025
Tim Penyusun

Daftar Isi

Persiapan Praktikum

1 Deskripsi Praktikum

Praktikum ini bertujuan untuk memahami konsep ACID (Atomicity, Consistency, Isolation, Durability) dan High Concurrency pada database PostgreSQL melalui empat skenario praktis.

Praktikum ini terdiri dari empat modul yang saling berkaitan:

1. **Indexing**: Mempelajari cara mengoptimalkan performa query dengan index dan mengukur dampaknya pada konkurensi tinggi.
2. **Transaction**: Mempelajari konsep ACID, tingkat isolasi transaksi, dan menangani konkurensi tinggi dengan benar.
3. **Partitioning**: Mempelajari cara membagi tabel besar menjadi bagian yang lebih kecil dan dampaknya terhadap performa.
4. **Backup dan Recovery**: Mempelajari strategi backup dan recovery untuk menjamin ketersediaan dan integritas data.

2 Persiapan Environment

2.1 Menggunakan Docker

1. Install Docker dan Docker Compose di komputer Anda
2. Unduh file praktikum yang sudah disediakan oleh asisten praktikum
3. Jalankan docker-compose file untuk membuat container dengan perintah:

```
docker-compose up -d
```

4. Generate data dummy:

```
cd scripts
pip install faker
python generate_data.py
```

5. Import data ke PostgreSQL:

```
cd scripts
bash import_data.sh
```

6. Akses pgAdmin di browser:
   - URL: http://localhost:5050
   - Email: admin@example.com
   - Password: p4ssw0rd
7. Tambahkan server di pgAdmin:
   - Name: PostgreSQL Praktikum
   - Host: postgres
   - Port: 5432
   - Username: praktikan
   - Password: p4ssw0rd

3 Petunjuk Penggunaan Modul

Setiap modul praktikum memiliki:

- PDF Modul
- File README.md dengan penjelasan konsep dan langkah-langkah
- File SQL dengan query untuk latihan
- Tugas

4 Panduan Pengerjaan

1. Praktikum dikerjakan dalam kelompok
2. Setiap kelompok mengerjakan semua modul praktikum
3. Pelaksanaan praktikum dilakukan oleh asisten praktikum dengan pengawasan koordinator dosen praktikum
4. Setiap kelompok membuat laporan akhir berisi:
   - Jawaban tugas dari setiap modul
   - Analisis dan kesimpulan

5 Struktur Folder Praktikum

Folder praktikum memiliki struktur sebagai berikut:

```
postgres-praktikum/
|-- docker-compose.yml
|-- init/
|   |-- 01-schema.sql
|   `-- 02-functions.sql
|-- data/
|   |-- products.csv
|   |-- customers.csv
|   |-- orders.csv
|   `-- order_items.csv
|-- scripts/
|   |-- generate_data.py
|   |-- import_data.sh
|   |-- benchmark.sh
|   `-- README.md
`-- praktikum/
|   |-- praktikum1_indexing/
|   |-- praktikum2_transaction/
|   |-- praktikum3_partitioning/
`-- |-- praktikum4_backup/
```

6 Dataset Praktikum

Dataset yang digunakan adalah data e-commerce dummy yang berisi:

- 10.000 produk
- 5.000 pelanggan
- 20.000 pesanan
- 50.000 item pesanan

7 Referensi

- PostgreSQL Documentation
- PostgreSQL Concurrency
- PostgreSQL Indexing
- PostgreSQL Partitioning
- PostgreSQL Backup and Restore

Bab 1

Implementasi Indexing pada PostgreSQL

1 Pendahuluan

Index pada database adalah struktur data khusus yang mempercepat operasi pencarian dan pengambilan data. Analoginya seperti indeks pada buku yang membantu kita menemukan konten tertentu dengan cepat tanpa perlu membaca seluruh buku. Dalam PostgreSQL, index disimpan dalam struktur khusus yang mengoptimalkan pencarian berdasarkan kolom tertentu.

Implementasi indexing yang tepat merupakan salah satu strategi utama untuk meningkatkan performa database, terutama pada sistem dengan data besar dan tingkat konkurensi tinggi.

1.1 Tujuan Praktikum

Dilaksanakannya praktikum indexing, praktikan diharapkan mampu:

1. Memahami konsep dasar indexing pada database PostgreSQL
2. Mempelajari tipe-tipe index di PostgreSQL
3. Mengimplementasikan index yang tepat untuk sebuah skenario
4. Menganalisis dampak index terhadap performa query konkuren

2 Konsep Dasar Indexing

2.1 Apa itu Index?

Index pada database bekerja mirip dengan indeks buku: membantu sistem menemukan data dengan cepat tanpa memeriksa seluruh tabel (table scan). Index menyimpan pointer ke baris data dalam tabel berdasarkan nilai kolom yang diindeks.

2.2 Bagaimana Index Bekerja

Ketika query dijalankan, PostgreSQL menganalisis apakah menggunakan index akan lebih efisien daripada melakukan table scan. Jika menggunakan index lebih efisien, PostgreSQL akan:

1. Mencari data di index berdasarkan kondisi WHERE
2. Menemukan referensi (pointer) ke baris data yang relevan
3. Mengambil data dari tabel menggunakan pointer tersebut

2.3 Kapan Menggunakan Index

Index sangat berguna pada:

1. Kolom yang sering digunakan dalam klausa WHERE
2. Kolom yang sering digunakan untuk JOIN antar tabel
3. Kolom yang sering digunakan dalam klausa ORDER BY atau GROUP BY
4. Tabel berukuran besar dengan query yang hanya mengambil sebagian kecil data

2.4 Kapan Tidak Menggunakan Index

Index tidak selalu bermanfaat pada:

1. Tabel kecil yang lebih efisien dilakukan sequential scan
2. Kolom dengan kardinalitas rendah (nilai yang berbeda sedikit)
3. Kolom yang jarang digunakan dalam query
4. Tabel yang sering diupdate/dihapus (overhead pemeliharaan index)

2.5 Dampak Index pada Performa

Index memberikan dampak pada:

1. **SELECT**: Mempercepat query dengan mengurangi jumlah baris yang perlu diperiksa
2. **INSERT**: Memerlukan overhead tambahan untuk memperbarui index
3. **UPDATE**: Memerlukan overhead untuk memperbarui index jika kolom yang diindex berubah
4. **DELETE**: Memerlukan overhead untuk memperbarui index

3 Jenis-jenis Index di PostgreSQL

PostgreSQL menyediakan beberapa jenis index yang dapat dipilih sesuai dengan kebutuhan:

3.1 B-tree Index

1. Index default di PostgreSQL
2. Efisien untuk perbandingan dengan operator equality (=) dan range ($<, >, <=, >=, BETWEEN$)
3. Mendukung urutan untuk ORDER BY
4. Cocok untuk kebanyakan kasus penggunaan

3.2 Hash Index

1. Optimal hanya untuk operator equality (=)
2. Lebih cepat dari B-tree untuk operasi equality sederhana
3. Tidak mendukung range queries atau sorting
4. Dirancang untuk tabel hash di memori

3.3 GiST Index (Generalized Search Tree)

1. Index untuk data geometri, text-search, dan data kompleks lainnya
2. Fleksibel, dapat digunakan untuk data custom
3. Mendukung nearest-neighbor searches
4. Digunakan untuk full-text search, data geografis, dll

3.4 GIN Index (Generalized Inverted Index)

1. Dirancang untuk nilai yang memiliki multiple components (arrays, jsonb, text-search)
2. Efisien untuk pencarian yang membutuhkan matching multiple values
3. Cocok untuk kolom yang menyimpan data semi-structured
4. Lebih lambat untuk operasi insert dibanding GiST

3.5 BRIN Index (Block Range INdex)

1. Dirancang untuk tabel sangat besar dengan data yang terurut secara natural
2. Sangat kecil dan efisien untuk kolom seperti timestamp, ID sequential, dll
3. Kinerja query lebih rendah dari B-tree tapi overhead penyimpanan jauh lebih kecil

3.6 SP-GiST Index (Space-Partitioned GiST)

1. Untuk data yang bisa dipartisi secara non-overlapping
2. Baik untuk data hierarchical seperti rentang IP, geo-data
3. Mendukung nearest-neighbor searches

Jenis Index	Karakteristik dan Penggunaan
B-tree	Index default PostgreSQL. Efisien untuk operasi perbandingan (=, ¡, ¿, BETWEEN) dan mendukung pengurutan (ORDER BY).
Hash	Khusus untuk operasi equality (=). Lebih cepat dari B-tree untuk lookup sederhana, tidak mendukung range.
GiST/GIN	Untuk data kompleks seperti full-text search, data spasial, array, dan JSON. GIN lebih lambat untuk insert tapi lebih cepat untuk search.
BRIN	Untuk tabel besar dengan data terurut secara natural (seperti timestamp). Ukuran kecil, overhead rendah.

Gambar 1.1: Jenis-jenis Index di PostgreSQL dan Penggunaannya

3.7 Benchmark pada Konkurensi Tinggi

Pada situasi konkurensi tinggi, performa index dapat dipengaruhi oleh beberapa faktor:

- **Contention**: Kompetisi antar koneksi untuk mengakses data yang sama
- **Lock contention**: Waktu tunggu karena lock pada baris atau tabel
- **Index bloat**: Overhead karena index yang jarang di-maintenance

4 Best Practices Implementasi Indexing

1. **Index kolom yang sering digunakan dalam WHERE, JOIN, ORDER BY**
2. **Hindari over-indexing**: Terlalu banyak index berdampak negatif pada performa INSERT/UPDATE/DELETE
3. **Gunakan composite index** untuk query dengan multiple conditions
4. **Perhatikan urutan kolom** pada composite index (most selective first)
5. **Pertimbangkan partial index** untuk subset data yang sering diakses
6. **Gunakan EXPLAIN ANALYZE** untuk validasi penggunaan index

7. **Jalankan VACUUM ANALYZE** secara berkala untuk memperbarui statistik
8. **Monitor ukuran dan penggunaan index** menggunakan pg_stat_*
9. **Lakukan REINDEX** pada index yang terfragmentasi
10. **Evaluasi trade-off** antara kecepatan query vs overhead pemeliharaan

5 Tahapan Praktikum

5.1 Persiapan

1. Pastikan container Docker PostgreSQL sudah berjalan
2. Gunakan pgAdmin atau PSQL untuk mengakses database
3. Pastikan data sudah diimpor menggunakan script yang disediakan

5.2 Mengeksplorasi Execution Plan

Pertama, Anda akan mempelajari cara melihat execution plan query menggunakan EXPLAIN dan EXPLAIN ANALYZE:

```
-- Menampilkan execution plan
EXPLAIN SELECT * FROM products WHERE category = '
 Elektronik';

-- Menampilkan execution plan beserta runtime
EXPLAIN ANALYZE SELECT * FROM products WHERE category = '
 Elektronik';
```

5.3 Mengukur Performa Query Tanpa Index

1. Jalankan query berikut tanpa index dan catat waktu eksekusinya:

```
    -- Query 1: Filter berdasarkan kategori
    SELECT COUNT(*) FROM products WHERE category = '
   Elektronik';

    -- Query 2: Filter berdasarkan range harga
    SELECT * FROM products WHERE price BETWEEN 1000000
    AND 5000000;

```

```
    -- Query 3: Filter berdasarkan tanggal pesanan
    SELECT o.order_id, o.customer_id, o.order_date
    FROM orders o
    WHERE o.order_date BETWEEN '2023-06-01' AND '
   2023-06-30';

    -- Query 4: Join tanpa index
    SELECT c.first_name, c.last_name, o.order_id, o.
   order_date, o.total_amount
    FROM customers c
    JOIN orders o ON c.customer_id = o.customer_id
    WHERE c.city = 'Jakarta';

    -- Query 5: Aggregate yang berat
    SELECT p.category, COUNT(*) as total_products, AVG
   (p.price) as avg_price
    FROM products p
    GROUP BY p.category
    ORDER BY total_products DESC;

```

2. Lakukan benchmark konkuren tanpa index:

```
    # Di terminal
    bash query_before.sql 20 100 10

```

5.4 Implementasi Index yang Tepat

Sekarang, tambahkan index yang sesuai untuk setiap query:

```
-- Hapus index yang sudah ada (untuk tujuan praktikum)
DROP INDEX IF EXISTS idx_products_category;
DROP INDEX IF EXISTS idx_products_price;
DROP INDEX IF EXISTS idx_orders_date;
DROP INDEX IF EXISTS idx_customers_city;

-- Index 1: B-Tree index untuk kategori produk
CREATE INDEX idx_products_category ON products(category);

-- Index 2: B-Tree index untuk range harga
CREATE INDEX idx_products_price ON products(price);

-- Index 3: B-Tree index untuk tanggal pesanan
```

```
CREATE INDEX idx_orders_date ON orders(order_date);

-- Index 4: Index untuk city pada tabel customers
CREATE INDEX idx_customers_city ON customers(city);

-- Index 5: Composite index untuk JOIN operation
CREATE INDEX idx_orders_customer_id ON orders(customer_id
 );
```

5.5 Mengukur Performa Setelah Indexing

1. Jalankan kembali query yang sama dan bandingkan waktu eksekusinya:

```
    -- Jalankan semua query yang sama seperti
   sebelumnya dan bandingkan execution plan
    EXPLAIN ANALYZE SELECT COUNT(*) FROM products
   WHERE category = 'Elektronik';
    -- dan seterusnya untuk query lainnya

```

2. Lakukan benchmark konkuren dengan index:

```
    # Di terminal
    cd scripts
    bash benchmark.sh ../praktikum/praktikum1_indexing
   /query_after.sql 20 100 10

```

5.6 Mempelajari Tipe Index Lain

PostgreSQL mendukung beberapa tipe index. Cobalah implementasi berikut:

```
-- Index Partial untuk produk dengan stok rendah
CREATE INDEX idx_products_low_stock ON products(
 product_id, stock_quantity)
WHERE stock_quantity < 10;

-- Index menggunakan ekspresi untuk pencarian case-
 insensitive
CREATE INDEX idx_products_name_lower ON products(LOWER(
 name));

-- BRIN Index untuk data berurutan (seperti timestamp)
```

```
CREATE INDEX idx_orders_date_brin ON orders USING BRIN(order_date);

-- GIN Index untuk pencarian full-text (jika menggunakan PostgreSQL >= 9.6)
CREATE INDEX idx_products_description_gin ON products
USING GIN(to_tsvector('english', description));
```

6 Referensi

- PostgreSQL Documentation: Indexes
- PostgreSQL Documentation: EXPLAIN
- PostgreSQL Documentation: Performance Tips
- PostgreSQL Documentation: Index Types
- PostgreSQL Documentation: pgbench