**Eksperimen 4 – WEB DEVELOPMENT**

*Laporan ini disusun untuk memenuhi Tugas Mata Kuliah   
Web Development (Teori)*

A logo with a blue and orange design

Description automatically generated

**Disusun oleh:**

Nama : Ferdi Ahmad Ariesta

NIM : 221524041

Kelas : 3B – D4

Program Studi : D4 Teknik Informatika

**Mata Kuliah:**

Nama Mata Kuliah : Sistem Informasi

Kode Mata Kuliah: : 21TI3002

**Dosen Pengampu**:

KO007N-Joe Lian Min, M.Eng

**SARJANA TERAPAN PROGRAM STUDI TEKNIK INFORMATIKA**

**TEKNIK KOMPUTER DAN INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

Daftar Isi

[Pendahuluan 3](#_Toc176262442)

[Identifikasi Masalah 3](#_Toc176262443)

[Tujuan Eksperimen 3](#_Toc176262444)

[Metodologi Eksperimen 3](#_Toc176262445)

[Pelaksanaan Eksperimen 4](#_Toc176262446)

[Analisis Hasil Eksperimen 4](#_Toc176262447)

[Analisis Hasil Eksperimen 7](#_Toc176262448)

[Kesimpulan 7](#_Toc176262449)

# Pendahuluan

Dalam dunia pengelolaan database, migrasi data adalah proses yang kritis, terutama ketika melibatkan database yang sedang aktif digunakan. Salah satu tantangan utama adalah bagaimana menangani transaksi yang sedang berjalan selama proses migrasi. Transaksi aktif yang belum diselesaikan (belum di-COMMIT atau di-ROLLBACK) dapat mempengaruhi hasil migrasi dan integritas data. Eksperimen ini dilakukan untuk memahami dampak transaksi aktif terhadap proses migrasi database menggunakan PostgreSQL dengan alat pg\_dump, serta bagaimana PostgreSQL menangani transaksi tersebut untuk menjaga konsistensi dan integritas data selama migrasi.

# Identifikasi Masalah

Dalam proses migrasi database antar PostgreSQL instances, salah satu tantangan utama adalah bagaimana mengelola perubahan data secara real-time pada database sumber (source\_db) agar tetap konsisten dengan database target (target\_db). Hal ini terutama menjadi masalah ketika database masih aktif dan terdapat transaksi yang terjadi selama migrasi. Migrasi yang tidak dikelola dengan baik dapat menyebabkan kehilangan data atau inkonsistensi pada database target.

Teknologi **Change Data Capture (CDC)**, yang diimplementasikan melalui **logical replication** di PostgreSQL, memberikan solusi dengan memonitor dan mereplikasi perubahan data dari source\_db ke target\_db secara real-time. Namun, terdapat kendala-kendala seperti:

* Proses subscription yang terus berjalan tanpa menyelesaikan query.
* Replikasi yang tidak terjadi meskipun data baru dimasukkan di source\_db.
* Kesalahan konfigurasi terkait penggunaan pg\_hba.conf dan parameter wal\_level yang tidak sesuai.

# Tujuan Eksperimen

Tujuan dari eksperimen ini adalah:

1. Mengimplementasikan **Change Data Capture (CDC)** menggunakan PostgreSQL dengan logical replication.
2. Mengidentifikasi masalah yang menyebabkan replikasi tidak berjalan sebagaimana mestinya.
3. Memastikan bahwa perubahan yang terjadi pada source\_db dapat direplikasi ke target\_db secara real-time dan tanpa kehilangan data.
4. Memvalidasi keberhasilan replikasi dengan melakukan tes input data baru di source\_db dan memeriksa keberadaan data di target\_db.

# Metodologi Eksperimen

Metodologi yang digunakan dalam eksperimen ini meliputi beberapa langkah:

1. **Persiapan Lingkungan**:
   * Dua instance PostgreSQL (source\_db dan target\_db) diatur di lokal dengan logical replication.
   * Konfigurasi parameter wal\_level, max\_replication\_slots, dan max\_wal\_senders di postgresql.conf.
   * Pengaturan file pg\_hba.conf untuk mengizinkan koneksi replika antar database.
2. **Pembuatan Publikasi**:
   * Pada source\_db, dibuat publikasi menggunakan perintah SQL CREATE PUBLICATION, yang akan mempublikasikan perubahan dari tabel yang dipilih.
3. **Pembuatan Subscription**:
   * Pada target\_db, dibuat subscription untuk menerima perubahan dari publikasi source\_db dengan perintah SQL CREATE SUBSCRIPTION.
4. **Monitoring dan Validasi**:
   * Memasukkan data baru ke tabel yang telah dipublikasi di source\_db.
   * Memeriksa tabel di target\_db apakah perubahan data ter-replikasi dengan benar.
   * Melakukan troubleshooting jika replikasi tidak berjalan sesuai harapan, termasuk pemeriksaan pada log PostgreSQL dan pengaturan ulang konfigurasi.

# Pelaksanaan Eksperimen

Tahap 1: Pembuatan Database Sumber dan Target

|  |
| --- |
| CREATE DATABASE source\_db; |
| CREATE DATABASE target\_db; |

Tahap 2 : Konfigurasi Database Sumber

1. Buat table di database sumber :

|  |
| --- |
| CREATE TABLE public.customers (  id SERIAL PRIMARY KEY,  name VARCHAR(100),  email VARCHAR(100),  created\_at TIMESTAMP  ); |

1. Masukkan Data ke dalam Tabel Sumber:

|  |
| --- |
| INSERT INTO public.customers (name, email, created\_at)  VALUES ('Alice', 'alice@example.com', NOW()),  ('Bob', 'bob@example.com', NOW()); |

1. Aktifkan Logical Replication di PostgreSQL:

Dalam file konfigurasi postgresql.conf, aktifkan logical replication dengan mengubah pengaturan wal\_level. Buka file postgresql.conf dan atur:

wal\_level = logical

Restart PostgreSQL setelah mengubah file konfigurasi.

1. Cek Pengaturan max\_replication\_slots dan max\_wal\_senders:

Pastikan parameter max\_replication\_slots dan max\_wal\_senders diatur cukup besar untuk menangani replikasi. Contoh konfigurasi:

max\_replication\_slots = 4

max\_wal\_senders = 4

1. Buat Publication di Database Sumber:

Publication adalah entitas yang mempublikasikan perubahan tabel dari database sumber. Buat publication untuk tabel customers:

|  |
| --- |
| CREATE PUBLICATION my\_publication FOR TABLE public.customers; |

Periksa publication untuk memastikan bahwa tabel telah masuk:

|  |
| --- |
| SELECT \* FROM pg\_publication\_tables WHERE pubname = 'my\_publication'; |

Tahap 3 : Konfigurasi Database Target

1. Pastikan Tabel Ada di Database Target:

Sebelum Anda membuat subscription, tabel customers harus ada di database target dengan skema yang sama seperti di database sumber. Jika tabel belum ada, buat dengan skema yang sama:

|  |
| --- |
| CREATE TABLE public.customers (  id SERIAL PRIMARY KEY,  name VARCHAR(100),  email VARCHAR(100),  created\_at TIMESTAMP  ); |

1. Buat Subscription di Database Target:

Setelah tabel ada di database target, buat subscription untuk mulai menerima perubahan dari database sumber. Subscription adalah entitas yang berlangganan perubahan dari publication di database sumber:

|  |
| --- |
| CREATE SUBSCRIPTION my\_subscription  CONNECTION 'host=localhost dbname=source\_db user=your\_user password=your\_password'  PUBLICATION my\_publication; |

1. Periksa Status Subscription:

Untuk memeriksa apakah subscription berjalan dengan baik, jalankan query berikut di database target:

|  |
| --- |
| SELECT \* FROM pg\_stat\_subscription; |

Anda akan melihat detail tentang subscription, termasuk statusnya (harus enabled) dan LSN (Log Sequence Number) terakhir yang diterima.

Tahap 4 : Verifikasi Replikasi Data

1. Uji Replikasi:

Coba masukkan data baru ke tabel customers di database sumber untuk melihat apakah data direplikasi ke database target:

|  |
| --- |
| INSERT INTO public.customers (name, email, created\_at)  VALUES ('Charlie', 'charlie@example.com', NOW()); |

1. Periksa Data di Database Target:

Lihat apakah data baru tersebut muncul di database target dengan query berikut:

|  |
| --- |
| SELECT \* FROM public.customers; |

# Analisis Hasil Eksperimen

|  |
| --- |
| **Menghentikan Sementara (Pause):**  ALTER SUBSCRIPTION my\_subscription DISABLE;  **Mengaktifkan Kembali:**  ALTER SUBSCRIPTION my\_subscription ENABLE;  **Menghapus Subscription:**  DROP SUBSCRIPTION my\_subscription; |

1. Kondisi awal database pada source\_db

A screenshot of a computer

Description automatically generated

# Kesimpulan