LAPORAN TUGAS BESAR

IF3140/Manajemen Basis Data

Direktorat Sistem dan Teknologi Informasi ITB

Dipersiapkan oleh:

Catherine Pricilla – 13514004

Taufic Leonardo Sutejo - 13514022

Chalvin – 13514032

Evita Chandra – 13514034

Malvin Juanda – 13514044

Scarletta Julia Yapfrine - 13514074

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | Nomor Dokumen | | Halaman |
| *IF3140-TB-6-2* | | *56* |
| *Revisi* | *-* | *10/28/2016* |

Daftar Isi

[1 Ringkasan 3](#_Toc465448560)

[2 Studi Lapangan 3](#_Toc465448561)

[2.1 Latar Belakang 3](#_Toc465448562)

[2.2 Rumusan Masalah 3](#_Toc465448563)

[2.3 Tujuan 4](#_Toc465448564)

[2.4 Batasan Masalah 4](#_Toc465448565)

[2.5 Solusi Singkat 4](#_Toc465448566)

[3 Skema Basis Data 5](#_Toc465448567)

[3.1 Entity-Relational Diagram 5](#_Toc465448568)

[3.2 Diagram Skematik 6](#_Toc465448569)

[4 Implementasi 7](#_Toc465448571)

[4.1 Performance Tuning 7](#_Toc465448572)

[4.1.1 Proses Implementasi 7](#_Toc465448573)

[4.1.2 Skenario Pengujian 17](#_Toc465448574)

[4.1.3 Hasil Pengujian 21](#_Toc465448575)

[5 Kesimpulan 31](#_Toc465448576)

[6 Pembagian Kerja dalam Kelompok 55](#_Toc465448577)

[7 Lampiran 55](#_Toc465448578)

[7.1 Log Activity Anggota Kelompok 55](#_Toc465448579)

# Ringkasan

Penulis melakukan eksplorasi pada basis data Direktorat Sistem dan Teknologi Informasi Institut Teknologi Bandung (Ditsti ITB) yang dulu dikenal dengan nama Comlabs ITB. Eksplorasi basis data dilakukan dengan tujuan untuk mengubah desain basis data sehingga basis data bisa melakukan pemrosesan querydan transaksi dengan lebih cepat.

Hasil eksplorasi tersebut dicantumkan pada laporan ini. Secara garis besar, laporan ini berisi skema basis data yang digunakan oleh Ditsti ITB, cara-cara tuning yang dilakukan pada basis data Ditsti ITB, serta hasil dari tuning tersebut. Ada tiga cara tuning yang dilakukan, yaitu schema tuning, query tuning, dan index tuning. Tuning yang sudah dilakukan tersebut membuat basis data bekerja dengan lebih cepat dalam melakukan pemrosesan query.

# Studi Lapangan

## Latar Belakang

Saat ini, perkembangan teknologi semakin cepat menuntut semua perusahan-perusahaan dan instansi yang ada di Indonesia agar mampu memanfaatkan dan mengelola sumber daya yang ada secara efisien dan efektif. Salah satu sumber daya yang penting di era modernisasi ini adalah teknologi sistem informasi.

Dalam sebuah sistem informasi dibutuhkan sebuah basis data yang berguna untuk menyimpan dan mengolah data perusahaan/instansi. Setiap instansi memiliki proses bisnis, transaksi dan workload yang berbeda-beda untuk basis datanya. Setiap workload yang berbeda membutuhkan desain basis data tertentu agar performansinya meningkat. Oleh karena itu, dibutuhkan suatu proses tuning untuk meningkatkan performansi basis data tersebut.

Instansi Ditsti ITB merupakan direktorat yang melayani bidang teknologi dan informasi di Institut Teknologi Bandung. Salah satu bidang yang dilayani Ditsti-ITB adalah praktikum dari mata kuliah PTI (Pengantar Teknologi Informasi) untuk mahasiswa TPB (Tahap Persiapan Bersama). Praktikum PTI menggunakan website

Untuk menyimpan semua data mengenai praktikum yang tersedia digunakan sistem basis data Sisfo PTI.

Seiring dengan berjalannya waktu, basis data Sisfo PTI pun semakin besar, terlebih lagi dengan data yang berubah tiap tahunnya dan sistem basis data Sisfo PTI yang pada umumnya diakses setiap hari. Selain itu, server yang digunakan juga memiliki spek yang tergolong rendah sehingga tidak dapat melakukan proses transaksi yang terlalu banyak. Oleh karena itu, perlu dilakukan *performance tuning* pada basis data Sisfo PTI agar dapat meningkatkan performansi basis datanya.

## Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dikemukakan terdahulu maka dapat dirumuskan masalah yang akan di bahas pada tulisan ini adalah sebagai berikut:

1. Bagaimana proses bisnis di Ditsi ITB?
2. Apa saja workload pada basis data Ditsi ITB?
3. Bagaimana performansi dari basis data Ditsi ITB?
4. Bagaimana cara meningkatkan performansi basis data Ditsi ITB?
5. Apa solusi singkat yang ditawarkan?

## Tujuan

Tujuan yang ingin dicapai dalam pelaksanaan tugas besar ini adalah:

1. Menghasilkan desain basis data yang lebih baik
2. Menemukan *query-query* yang optimal untuk setiap eksekusi
3. Meningkatkan *performance database* Ditsti ITB

## Batasan Masalah

Adapun pembatasan masalah dalam tugas akhir ini adalah :

1. Query yang dipakai dalam Tugas Besar ini belum mencakup semua query yang digunakan di *web* Sisfo PTI

## Solusi Singkat

Solusi yang dapat diberikan untuk menyelesaikan persolan diatas yaitu dengan melakukan *performance tuning,* dalam hal ini dilakukan 3 cara *performance tuning*  yaitu *schema tuning, index tuning,* dan *query tuning.*

# Skema Basis Data

## Entity-Relational Diagram

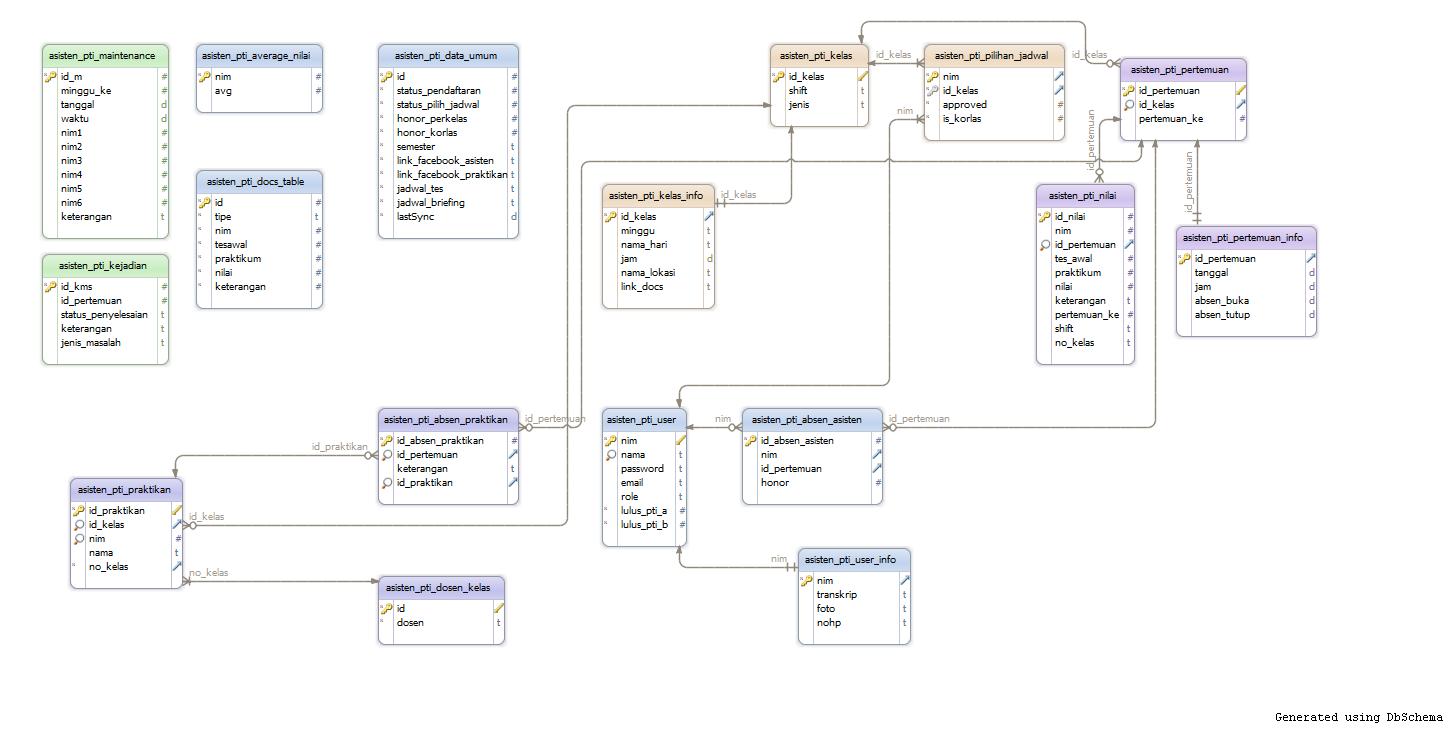


Gambar 1. Entity-Relational Diagram

## Diagram Skematik

## E:\skema sebelum.jpg

Gambar 2. Diagram Skematik Basis Data Sebelum dilakukan Tuning



Gambar 3. Diagram Skematik Basis Data Sesudah dilakukan Tuning

# Implementasi

## Performance Tuning

### Proses Implementasi

#### Schema Tuning

Skema tuning yang dilakukan pada basisdata SISFO PTI menggunakan 4 teknik, yaitu :

1. Collapsing Tables
2. Tabel **asisten\_pti\_kelas**, **asisten\_pti\_jam**, **asisten\_pti\_lokasi**, dan **asisten\_pti\_hari**

**Alasan**

Untuk mendapatkan nama\_lokasi, nama\_hari, dan jam praktikum, tabel asisten\_pti\_kelas perlu dilakukan join terhadap tabel asisten\_pti\_jam, asisten\_pti\_lokasi, dan asisten\_pti\_hari yang tentunya membutuhkan cost yang banyak apalagi digunakan dalam data yang besar. Selain itu, tabel asisten\_pti\_jam, asisten\_pti\_lokasi, dan asisten\_pti\_hari tidak memiliki relasi ke tabel lain selain tabel asisten\_pti\_kelas sehingga *collapsing* ketiga tabel tersebut ke tabel asisten\_pti\_kelas merupakan hal yang tepat.

**Implementasi**

Melakukan penggabungan tabel asisten\_pti\_hari, asisten\_pti\_lokasi, dan asisten\_pti\_jam pada tabel asisten\_pti\_kelas.

Asisten\_pti\_kelas(id\_kelas, shift, jenis, mingu, nama\_hari, jam, nama\_lokasi, link\_docs)

1. Tabel **asisten\_pti\_kejadian** dan tabel **asisten\_pti\_masalah\_teknis**

**Alasan**

Tabel asisten\_pti\_masalah\_teknis tidak memiliki relasi ke tabel lain dan hanya memiliki relasi ke tabel asisten\_pti\_kejadian. Selain itu, untuk mengetahui jenis masalah teknis tabel asisten\_pti\_kejadian perlu di-join terhadap tabel asisten\_pti\_masalah teknis yang memerlukan cost yang banyak. Oleh karena itu, untuk memperkecil cost, dilakukan *collapsing* terhadap kedua tabel diatas.

**Implementasi**

Melakukan penggabungan tabel asisten\_pti\_kejadian dengan tabel asisten\_pti\_masalah\_teknis.

Asisten\_pti\_kejadian(id\_kms, id\_pertemuan, status\_pembayaran, keterangan, jenis\_masalah)

1. Vertical Partitioning

Beberapa tabel pada skema basisdata SISFO PTI memiliki kolom(atribut) yang jarang diakses dalam penggunaannya. Berikut tabel yang dilakukan *vertical partitionong* :

1. Tabel **asisten\_pti\_user**

**Alasan**

Tabel asisten\_pti\_user memiliki kolom yang jarang diakes yaitu nohp, foto, dan transkrip. Ketiga kolom tersebut datanya baru akan ditampilkan ketika user mengklik suatu link di web asisten PTI. Data tersebut juga sangat jarang dilihat asisten karena biasanya asisten hanya melihat kelas yang akan dijaga maupun daftar asisten di suatu kelas. Sedangkan, atribut lainnya sering digunakan karena diutuhkan saat melakukan login asisten.

**Implementasi**

Memisahkan tabel asisten\_pti\_user menjadi tabel asisten\_pti\_user dan tabel asisten\_pti\_user\_info(tabel yang atributnya jrang diapakai).

Asisten\_pti\_user(nim, nama, password, email, role, lulus\_pti\_a, lulus\_pti\_b)

Assiten\_pti\_user\_info(nim, nama, transkrip, foto)

1. Tabel **asisten\_pti\_kelas**

**Alasan**

Tabel asisten\_pti\_kelas berisi informasi suatu kelas mulai dari shift, jenis PTI, sampai link docs penilaian. Informasi tersebut akan ditampilkan untuk melihat jadwal seorang praktikan. Akan tetapi, praktikan hanya melihat data tersebut ketika mengikuti awal praktikum dan seiring berjalannya praktikan akan lebih sering melihat tabel nilai mereka dibandingkan jadwal praktikum. Untuk itu, tuning terhadap tabel **asisten\_pti\_kelas** untuk kolom minggu, nama\_hari, nama\_lokasi, jam, dan link\_docs perlu dipartisi untuk mengurangi cost ketika proses karena aribut tersebut tidak dipakai di query untuk menampilkan absen ataupun nilai.

**Implementasi**

Membagi tabel asisten\_pti\_kelas menjadi tabel asisten\_pti\_kelas dan tabel asisten\_pti\_kelas\_info(tabel yang atributnya jarang digunakan).

Asisten\_pti\_kelas(id\_kelas, shift, jenis)

Asisten\_pti\_kelas\_info(id\_kelas, minggu, nama\_hari, jam, nama\_lokasi, link\_docs)

1. Tabel **asisten\_pti\_pertemuan**

**Alasan**

Tabel asisten\_pti\_pertemuan memiliki atribut tanggal, jam, absen\_buka, dan absen\_tutup yang sangat jarang ditampilkan di web asisten maupun di web nilai. Untuk itu, ke-empat atribut tersebut akan dilakukan partisi untuk mempersingkat waktu *fetch* data.

**Implementasi**

Membagi tabel asisten\_pti\_pertemuan menjadi tabel asisten\_pti\_pertemuan dan tabel asisten\_pti\_pertemuan\_info(tabel yang atributnya jarang digunakan)

Asisten\_pti\_pertemuan(id\_pertemuan, id\_kelas, pertemuan\_ke)

Asisten\_pti\_pertemuan\_info(id\_pertemuan, tanggal, jam, absen\_buka, absen\_tutup)

1. Add Redundant Columns

Tabel **asisten\_pti\_nilai**

**Alasan**

Seperti diketahui, nilai merupakan salah satu data yang paling sering diakses oleh setiap praktikan. Akan tetapi, query untuk mendapatkan data tersebut memerlukan 3 kali natural join yaitu tabel asisten\_pti\_nilai, asisten\_pti\_pertemuan, dan asisten\_pti\_praktikan. Padahal, dari tabel asisten\_pti\_pertemuan atribut yang perlu diambil berupa pertemuan\_ke dan shift dan tabel asisten\_pti\_praktikan atribut yang perlu diambil berupa no\_kelas.

Oleh karena penggunaan natural join hanya mendapatkan atribut yang sedikit dan nilai sering diakses oleh praktikan maka untuk mengurangi cost ditambhkan *redundant* *columns* pada tabel asisten\_pti\_nilai.

**Implementasi**

Menambahkan atribut pertemuan\_ke, no\_kelas, dan sift ke tabel asisten\_pti\_nilai.

Asisten\_pti\_nilai(id\_nilai, nim, id\_pertemuan, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke, shift, no\_kelas)

1. Add Derived Columns

**Alasan**

Pada web nilai, terdapat rekapitulasi(rata-rata) nilai praktkan terhadap semua praktikum yang telah dijalani. Namun, perhitungan rata-rata tersebut menggunakan aggregasi SUM nilai semua praktikum dan COUNT jumlah praktikum.

Untuk itu, dibuat *derivated* *columns* asisten\_pti\_average\_nilai yang berisi nilai rata-rata praktikan sejauh ini. Akan tetapi, penambahan tabel baru tersebut memberatkan cost saat melakukan update data dimana diperlukan perhitungan ulang ketika praktikan mendapati nilai pada praktikum selanjutnya.

**Implementasi**

Menambahkan tabel asisten\_pti\_average\_nilai baru yang telah memproses hasil agregasi yang diperlukan pada query yang bersangkutan, yaitu mendapatan rata-rata mahasiswa per praktikum yang telah dijalani.

Asisten\_pti\_average\_nilai(nim,avg)

Skema tuning yang dilakukan pada basisdata SISFO PTI menggunakan 4 teknik, yaitu

1. Collapsing Tables
2. Tabel **asisten\_pti\_kelas**, **asisten\_pti\_jam**, **asisten\_pti\_lokasi**, dan **asisten\_pti\_hari**

**Alasan**

Untuk mendapatkan nama\_lokasi, nama\_hari, dan jam praktikum, tabel asisten\_pti\_kelas perlu dilakukan join terhadap tabel asisten\_pti\_jam, asisten\_pti\_lokasi, dan asisten\_pti\_hari yang tentunya membutuhkan cost yang banyak apalagi digunakan dalam data yang besar. Selain itu, tabel asisten\_pti\_jam, asisten\_pti\_lokasi, dan asisten\_pti\_hari tidak memiliki relasi ke tabel lain selain tabel asisten\_pti\_kelas sehingga *collapsing* ketiga tabel tersebut ke tabel asisten\_pti\_kelas merupakan hal yang tepat.

**Implementasi**

Melakukan penggabungan tabel asisten\_pti\_hari, asisten\_pti\_lokasi, dan asisten\_pti\_jam pada tabel asisten\_pti\_kelas.

Asisten\_pti\_kelas(id\_kelas, shift, jenis, mingu, nama\_hari, jam, nama\_lokasi, link\_docs)

1. Tabel **asisten\_pti\_kejadian** dan tabel **asisten\_pti\_masalah\_teknis**

**Alasan**

Tabel asisten\_pti\_masalah\_teknis tidak memiliki relasi ke tabel lain dan hanya memiliki relasi ke tabel asisten\_pti\_kejadian. Selain itu, untuk mengetahui jenis masalah teknis tabel asisten\_pti\_kejadian perlu di-join terhadap tabel asisten\_pti\_masalah teknis yang memerlukan cost yang banyak. Oleh karena itu, untuk memperkecil cost, dilakukan *collapsing* terhadap kedua tabel diatas.

**Implementasi**

Melakukan penggabungan tabel asisten\_pti\_kejadian dengan tabel asisten\_pti\_masalah\_teknis.

Asisten\_pti\_kejadian(id\_kms, id\_pertemuan, status\_pembayaran, keterangan, jenis\_masalah)

1. Vertical Partitioning

Beberapa tabel pada skema basisdata SISFO PTI memiliki kolom(atribut) yang jarang diakses dalam penggunaannya. Berikut tabel yang dilakukan *vertical partitionong* :

1. Tabel **asisten\_pti\_user**

**Alasan**

Tabel asisten\_pti\_user memiliki kolom yang jarang diakes yaitu nohp, foto, dan transkrip. Ketiga kolom tersebut datanya baru akan ditampilkan ketika user mengklik suatu link di web asisten PTI. Data tersebut juga sangat jarang dilihat asisten karena biasanya asisten hanya melihat kelas yang akan dijaga maupun daftar asisten di suatu kelas. Sedangkan, atribut lainnya sering digunakan karena diutuhkan saat melakukan login asisten.

**Implementasi**

Memisahkan tabel asisten\_pti\_user menjadi tabel asisten\_pti\_user dan tabel asisten\_pti\_user\_info(tabel yang atributnya jrang diapakai).

Asisten\_pti\_user(nim, nama, password, email, role, lulus\_pti\_a, lulus\_pti\_b)

Assiten\_pti\_user\_info(nim, nama, transkrip, foto)

1. Tabel **asisten\_pti\_kelas**

**Alasan**

Tabel asisten\_pti\_kelas berisi informasi suatu kelas mulai dari shift, jenis PTI, sampai link docs penilaian. Informasi tersebut akan ditampilkan untuk melihat jadwal seorang praktikan. Akan tetapi, praktikan hanya melihat data tersebut ketika mengikuti awal praktikum dan seiring berjalannya praktikan akan lebih sering melihat tabel nilai mereka dibandingkan jadwal praktikum. Untuk itu, tuning terhadap tabel **asisten\_pti\_kelas** untuk kolom minggu, nama\_hari, nama\_lokasi, jam, dan link\_docs perlu dipartisi untuk mengurangi cost ketika proses karena aribut tersebut tidak dipakai di query untuk menampilkan absen ataupun nilai.

**Implementasi**

Membagi tabel asisten\_pti\_kelas menjadi tabel asisten\_pti\_kelas dan tabel asisten\_pti\_kelas\_info(tabel yang atributnya jarang digunakan).

Asisten\_pti\_kelas(id\_kelas, shift, jenis)

Asisten\_pti\_kelas\_info(id\_kelas, minggu, nama\_hari, jam, nama\_lokasi, link\_docs)

1. Tabel **asisten\_pti\_pertemuan**

**Alasan**

Tabel asisten\_pti\_pertemuan memiliki atribut tanggal, jam, absen\_buka, dan absen\_tutup yang sangat jarang ditampilkan di web asisten maupun di web nilai. Untuk itu, ke-empat atribut tersebut akan dilakukan partisi untuk mempersingkat waktu *fetch* data.

**Implementasi**

Membagi tabel asisten\_pti\_pertemuan menjadi tabel asisten\_pti\_pertemuan dan tabel asisten\_pti\_pertemuan\_info(tabel yang atributnya jarang digunakan)

Asisten\_pti\_pertemuan(id\_pertemuan, id\_kelas, pertemuan\_ke)

Asisten\_pti\_pertemuan\_info(id\_pertemuan, tanggal, jam, absen\_buka, absen\_tutup)

1. Add Redundant Columns

Tabel **asisten\_pti\_nilai**

**Alasan**

Seperti diketahui, nilai merupakan salah satu data yang paling sering diakses oleh setiap praktikan. Akan tetapi, query untuk mendapatkan data tersebut memerlukan 3 kali natural join yaitu tabel asisten\_pti\_nilai, asisten\_pti\_pertemuan, dan asisten\_pti\_praktikan. Padahal, dari tabel asisten\_pti\_pertemuan atribut yang perlu diambil berupa pertemuan\_ke dan shift dan tabel asisten\_pti\_praktikan atribut yang perlu diambil berupa no\_kelas.

Oleh karena penggunaan natural join hanya mendapatkan atribut yang sedikit dan nilai sering diakses oleh praktikan maka untuk mengurangi cost ditambhkan *redundant* *columns* pada tabel asisten\_pti\_nilai.

**Implementasi**

Menambahkan atribut pertemuan\_ke, no\_kelas, dan sift ke tabel asisten\_pti\_nilai.

Asisten\_pti\_nilai(id\_nilai, nim, id\_pertemuan, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke, shift, no\_kelas)

1. Add Derivated Columns

**Alasan**

Pada web nilai, terdapat rekapitulasi(rata-rata) nilai praktkan terhadap semua praktikum yang telah dijalani. Namun, perhitungan rata-rata tersebut menggunakan aggregasi SUM nilai semua praktikum dan COUNT jumlah praktikum.

Untuk itu, dibuat *derivated* *columns* asisten\_pti\_average\_nilai yang berisi nilai rata-rata praktikan sejauh ini. Akan tetapi, penambahan tabel baru tersebut memberatkan cost saat melakukan update data dimana diperlukan perhitungan ulang ketika praktikan mendapati nilai pada praktikum selanjutnya.

**Implementasi**

Menambahkan tabel asisten\_pti\_average\_nilai baru yang telah memproses hasil agregasi yang diperlukan pada query yang bersangkutan, yaitu mendapatan rata-rata mahasiswa per praktikum yang telah dijalani.

Asisten\_pti\_average\_nilai(nim,avg)

#### Query Tuning

Query Tuning terhadap query-query yang sering digunakan pada basisdata SISFO PTI yaitu:

1. Pembagian kelas Praktikum

select nim, nama, no\_kelas, shift

from asisten\_pti\_praktikan natural join

asisten\_pti\_kelas where shift like 'SITH%'

**Alasan**

Penggunaan natural join dengan asisten\_pti\_kelas kurang efisien karena akan memerlukan waktu yang lebih lama. Oleh karena itu, untuk memperkecil waktu yang dibutuhkan, lebih baik melakukan penyeleksian shift terlebih dahulu pada table asisten\_pti\_kelas.

**Implementasi**

select nim, nama, no\_kelas, shift

from (select id\_kelas, shift from asisten\_pti\_kelas where shift like 'SITH%') as kelas

natural join asisten\_pti\_praktikan

Melakukan seleksi shift terlebih dahulu pada tabel asisten\_pti\_kelas baru di-natural joinkan dengan tabel asisten\_pti\_praktikan.

1. Pembagian Nilai per shift praktikum

select nim, shift, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke

from asisten\_pti\_nilai

where shift = 'SF A';

**Alasan**

Query diatas tidak perlu di*-tuning* lagi karena sudah baik.

**Implementasi**

-

1. Nilai per kelas kuliah

select nim, no\_kelas, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke

from asisten\_pti\_nilai

where no\_kelas = 'A01';

**Alasan**

Query diatas tidak perlu di*-tuning* lagi karena sudah baik.

**Implementasi**

-

1. Presensi Praktikum

select nim, shift, pertemuan\_ke from asisten\_pti\_absen\_praktikan

natural join asisten\_pti\_praktikan

natural join asisten\_pti\_pertemuan

natural join asisten\_pti\_kelas

where shift = 'FTMD A'

order by pertemuan\_ke;

**Alasan**

Penggunaan natural join dengan asisten\_pti\_kelas kurang efisien karena akan memerlukan waktu yang lebih lama. Oleh karena itu, untuk memperkecil waktu yang dibutuhkan, lebih baik melakukan penyeleksian shift terlebih dahulu pada table asisten\_pti\_kelas dan pengurutan pertemuan\_ke terlebih dahulu pada tabel asisten\_pti\_pertemuan

**Implementasi**

select nim, shift, pertemuan\_ke

from (select id\_pertemuan, pertemuan\_ke from asisten\_pti\_pertemuan order by pertemuan\_ke) as pertemuan

natural join (select shift, id\_kelas from asisten\_pti\_kelas

where shift = 'FTMD A') as kelas

natural join asisten\_pti\_absen\_praktikan

natural join asisten\_pti\_praktikan

Melakukan seleksi shift terlebih dahulu pada tabel asisten\_pti\_kelas dan pengurutan pertemuan\_ke pada tabel asisten\_pti\_pertemuan baru di-natural joinkan dengan tabel asisten\_pti\_praktikan dan asisten\_pti\_absen\_praktikan.

1. Presensi Kelas Kuliah

select nim, shift, pertemuan\_ke from asisten\_pti\_absen\_praktikan

natural join asisten\_pti\_praktikan

natural join asisten\_pti\_pertemuan

natural join asisten\_pti\_kelas

where no\_kelas=’A01’

order by pertemuan\_ke;

**Alasan**

Penggunaan natural join dengan asisten\_pti\_kelas kurang efisien karena akan memerlukan waktu yang lebih lama. Oleh karena itu, untuk memperkecil waktu yang dibutuhkan, lebih baik melakukan penyeleksian shift terlebih dahulu pada table asisten\_pti\_kelas dan pengurutan pertemuan\_ke terlebih dahulu pada tabel asisten\_pti\_pertemuan

**Implementasi**

SELECT nim, shift, pertemuan\_ke FROM (select id\_praktikan, id\_kelas, nim from asisten\_pti\_praktikan where no\_kelas='A01') as praktikan

NATURAL JOIN asisten\_pti\_absen\_praktikan

NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_kelas ORDER BY pertemuan\_ke

Melakukan seleksi no\_kelas terlebih dahulu pada tabel asisten\_pti\_praktikan dan pengurutan pertemuan\_ke pada tabel asisten\_pti\_pertemuan baru dilakukan natural join dengan tabel asisten\_pti\_kelas dan asisten\_pti\_absen\_praktikan.

1. Kelas yang Menjadi Tanggung Jawab Asisten Tertentu

select nama, shift, nama\_hari, minggu, jam

from asisten\_pti\_user

natural join asisten\_pti\_pilihan\_jadwal

natural join asisten\_pti\_kelas

natural join asisten\_pti\_kelas\_info

where approved = 1

AND nim = 13514032

**Alasan**

Penggunaan natural join dengan asisten\_pti\_user dan pilihan jadwal kurang efisien karena akan memerlukan waktu yang lebih lama. Oleh karena itu, untuk memperkecil waktu yang dibutuhkan, lebih baik melakukan penyeleksian nim terlebih dahulu pada table asisten\_pti\_user dan penyeleksian mahasiswa yang di-*approved* terlebih dahulu pada tabel asisten\_pti\_pilihan\_jadwal

**Implementasi**

select nama, shift, nama\_hari, minggu, jam

from (select nim, nama from asisten\_pti\_user where nim = 13514032) as pengguna

natural join (select id\_kelas, nim from asisten\_pti\_pilihan\_jadwal where approved = 1) as kelas

natural join asisten\_pti\_kelas natural join asisten\_pti\_kelas\_info

Melakukan seleksi nim terlebih dahulu pada table asisten\_pti\_user dan penyeleksian mahasiswa yang di-*approved* terlebih dahulu pada tabel asisten\_pti\_pilihan\_jadwal

baru di-natural joinkan dengan tabel asisten\_pti\_kelas dan asisten\_pti\_kelas\_info.

1. Pencarian jumlah honor yang didapat oleh asisten

select nama, pertemuan\_ke, honor

from asisten\_pti\_absen\_asisten

natural join asisten\_pti\_pertemuan

natural join asisten\_pti\_user

where nim = 13514032

**Alasan**

Penggunaan natural join dengan asisten\_pti\_user kurang efisien karena akan memerlukan waktu yang lebih lama. Oleh karena itu, untuk memperkecil waktu yang dibutuhkan, lebih baik melakukan penyeleksian nim terlebih dahulu pada table asisten\_pti\_user.

**Implementasi**

select nama, pertemuan\_ke, honor

from (select nim,nama from asisten\_pti\_user where nim = 13514032) as pengguna

natural join asisten\_pti\_pertemuan

natural join asisten\_pti\_absen\_asisten

Melakukan seleksi nim terlebih dahulu pada table asisten\_pti\_user baru di-natural joinkan dengan tabel asisten\_pti\_pertemuan dan asisten\_pti\_absen\_asisten.

1. Pencarian korlas

select nama, shift, is\_korlas

from asisten\_pti\_user

natural join asisten\_pti\_pilihan\_jadwal

natural join asisten\_pti\_kelas

where shift = 'FTSL E' AND approved = 1

**Alasan**

Penggunaan natural join dengan asisten\_pti\_kelas dan asisten\_pti\_pilihan jadwal kurang efisien karena akan memerlukan waktu yang lebih lama. Oleh karena itu, untuk memperkecil waktu yang dibutuhkan, lebih baik melakukan penyeleksian shift terlebih dahulu pada table asisten\_pti\_kelas dan penyeleksian mahasiswa yang di-*approved* terlebih dahulu pada tabel asisten\_pti\_pilihan\_jadwal.

**Implementasi**

select nama, shift, is\_korlas

from (select id\_kelas,nim , is\_korlas from asisten\_pti\_pilihan\_jadwal where approved = 1) as jadwal

natural join (select id\_kelas, shift from asisten\_pti\_kelas where shift = 'FTSL E') as kelas

natural join asisten\_pti\_user

Melakukan penyeleksian mahasiswa yang di-*approved* terlebih dahulu pada tabel asisten\_pti\_pilihan\_jadwal lalu melakukan penyeleksian shift terlebih dahulu pada tabel asisten\_pti\_kelas baru di-natural joinkan dengan tabel asisten\_pti\_kelas dan asisten\_pti\_kelas\_info.

1. Pencarian rata-rata nilai

SELECT \* FROM asisten\_pti\_average\_nilai

**Alasan**

Query diatas tidak perlu di*-tuning* lagi karena sudah baik.

**Implementasi**

-

#### Index Tuning

Index tuning dilakukan dengan cara penambahan index pada kolom-kolom yang sering diakses dan pada kolom-kolom yang merupakan *foreign key* pada query yang sering diproses. Ada dua jenis index yang digunakan, yaitu btree dan hash index. Kolom-kolom yang ditambahkan index adalah :

1. Kolom nim dari tabel asisten\_pti\_praktikan

Kolom ini diberi btree index karena semua data pada kolom tersebut unik.

1. Kolom nama dari tabel asisten\_pti\_user

Kolom ini diberi hash index karena data pada kolom tersebut ada yang sama.

1. Kolom id\_kelas dari tabel asisten\_pti\_praktikan

Kolom ini diberi hash index karena data pada kolom tersebut ada yang sama.

1. Kolom id\_kelas dari tabel asisten\_pti\_pertemuan

Kolom ini diberi hash index karena data pada kolom tersebut ada yang sama.

1. Kolom id\_pertemuan dari tabel asisten\_pti\_nilai

Kolom ini diberi hash index karena data pada kolom tersebut ada yang sama.

1. Kolom id\_pertemuan dari tabel asisten\_pti\_absen\_praktikan

Kolom ini diberi hash index karena data pada kolom tersebut ada yang sama.

1. Kolom id\_praktikan dari tabel asisten\_pti\_absen\_praktikan

Kolom ini diberi hash index karena data pada kolom tersebut ada yang sama.

### Skenario Pengujian

#### Schema Tuning

Pertama, dilakukan schema tuning seperti telah dituliskan pada proses implementasi di atas, maka dilakukan perubahan query untuk menyesuaikan dengan schema yang baru. Pengujian dilakukan dengan menggunakan kedua query ini untuk membandingkan hasil dari skema tuning dengan sebelumnya.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Query Awal | Query Akhir | Penjelasan |
| 1 | SELECT nim, nama, no\_kelas, shift  FROM asisten\_pti\_praktikan NATURAL JOIN  asisten\_pti\_kelas WHERE shift LIKE 'SITH%' | SELECT nim, nama, no\_kelas, shift  FROM asisten\_pti\_praktikan NATURAL JOIN  asisten\_pti\_kelas WHERE shift LIKE 'SITH%' | Query untuk melihat pembagian kelas praktikum |
| 2 | SELECT nim, shift, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai NATURAL JOIN asisten\_pti\_kelas NATURAL JOIN asisten\_pti\_pertemuan  WHERE shift = 'SF A' | SELECT nim, shift, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai  WHERE shift = 'SF A' | Query untuk melihat nilai per shift praktikum |
| 3 | SELECT nim, no\_kelas, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke FROM asisten\_pti\_nilai NATURAL JOIN asisten\_pti\_praktikan NATURAL JOIN asisten\_pti\_pertemuan WHERE no\_kelas = ‘A01’ | SELECT nim, no\_kelas, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai  WHERE no\_kelas = 'A01' | Query untuk melihat nilai per kelas kuliah |
| 4 | SELECT nim, shift, pertemuan\_ke FROM asisten\_pti\_absen\_praktikan NATURAL JOIN asisten\_pti\_praktikan NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_kelas WHERE shift = ‘FTMD A’ ORDER BY pertemuan\_ke | SELECT nim, shift, pertemuan\_ke FROM asisten\_pti\_absen\_praktikan  NATURAL JOIN asisten\_pti\_praktikan  NATURAL JOIN asisten\_pti\_pertemuan  NATURAL JOIN asisten\_pti\_kelas WHERE shift = 'FTMD A'  ORDER BY pertemuan\_ke | Query untuk melihat presensi berdasarkan kelas praktikum |
| 5 | SELECT nim, shift, pertemuan\_ke FROM asisten\_pti\_absen\_praktikan NATURAL JOIN  asisten\_pti\_praktikan  NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_kelas WHERE no\_kelas = ‘A01’ ORDER BY pertemuan\_ke | SELECT nim, shift, pertemuan\_ke FROM asisten\_pti\_absen\_praktikan  NATURAL JOIN asisten\_pti\_praktikan  NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_kelas WHERE no\_kelas='A01' ORDER BY pertemuan\_ke | Query untuk melihat presensi berdasarkan kelas kuliah |
| 6 | SELECT nim, avg(nilai) FROM asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_nilai GROUP BY nim | SELECT \* FROM asisten\_pti\_average\_nilai | Query untuk melihat nilai rata-rata praktikum dari masing-masing mahasiswa |
| 7 | SELECT nama, shift, nama\_hari, minggu, jam  FROM asisten\_pti\_user NATURAL JOIN asisten\_pti\_pilihan\_jadwal NATURAL JOIN asisten\_pti\_kelas NATURAL JOIN asisten\_pti\_hari NATURAL JOIN asisten\_pti\_jam NATURAL JOIN asisten\_pti\_lokasi  WHERE approved = 1 AND nim = 13514032 | SELECT nama, shift, nama\_hari, minggu, jam  FROM asisten\_pti\_user NATURAL JOIN asisten\_pti\_pilihan\_jadwal NATURAL JOIN asisten\_pti\_kelas NATURAL JOIN asisten\_pti\_kelas\_info  WHERE approved = 1  AND nim = 13514032 | Query untuk melihat data kelas yang menjadi tanggung jawab asisten tertentu |
| 8 | SELECT nama, pertemuan\_ke, honor FROM asisten\_pti\_absen\_asisten NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_user WHERE nim = 13514032 | SELECT nama, pertemuan\_ke, honor FROM asisten\_pti\_absen\_asisten NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_user WHERE nim = 13514032 | Query untuk melihat presensi asisten beserta honor |
| 9 | SELECT nama, shift, is\_korlas FROM asisten\_pti\_user NATURAL JOIN asisten\_pti\_pilihan\_jadwal NATURAL JOIN asisten\_pti\_kelas WHERE shift = ‘FTSL E’ AND approved = 1 | SELECT nama, shift, is\_korlas  FROM asisten\_pti\_user NATURAL JOIN asisten\_pti\_pilihan\_jadwal NATURAL JOIN asisten\_pti\_kelas WHERE shift = 'FTSL E' AND approved = 1 | Query untuk melihat korlas dari sebuah shift |

#### Index Tuning

Pada bagian ditambahkan index pada beberapa kolom yang sering diakses dan kolom yang merupakan foreign key. Query yang digunakan untuk pengujian masih sama dengan query akhir yang didapatkan dari schema tuning.

#### Query Tuning

Query dianalisis lalu dilakukan optimisasi agar lebih cepat. Berikut ini adalah perubahan query yang dilakukan:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Query Awal | Query Akhir | Penjelasan |
| 1 | SELECT nim, nama, no\_kelas, shift  FROM asisten\_pti\_praktikan NATURAL JOIN  asisten\_pti\_kelas WHERE shift LIKE 'SITH%' | SELECT nim, nama, no\_kelas, shift FROM (SELECT id\_kelas, shift FROM asisten\_pti\_kelas WHERE shift LIKE 'SITH%') AS kelas NATURAL JOIN asisten\_pti\_praktikan | Query untuk melihat pembagian kelas praktikum |
| 2 | SELECT nim, shift, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai  WHERE shift = 'SF A' | SELECT nim, shift, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai  WHERE shift = 'SF A' | Query untuk melihat nilai per shift praktikum |
| 3 | SELECT nim, no\_kelas, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai  WHERE no\_kelas = 'A01' | SELECT nim, no\_kelas, tes\_awal, praktikum, nilai, keterangan, pertemuan\_ke  FROM asisten\_pti\_nilai  WHERE no\_kelas = 'A01' | Query untuk melihat nilai per kelas kuliah |
| 4 | SELECT nim, shift, pertemuan\_ke FROM asisten\_pti\_absen\_praktikan  NATURAL JOIN asisten\_pti\_praktikan  NATURAL JOIN asisten\_pti\_pertemuan  NATURAL JOIN asisten\_pti\_kelas WHERE shift = 'FTMD A'  ORDER BY pertemuan\_ke | SELECT nim, shift, pertemuan\_ke  FROM (SELECT id\_pertemuan, pertemuan\_ke FROM asisten\_pti\_pertemuan ORDER BY pertemuan\_ke) AS pertemuan  NATURAL JOIN (SELECT shift, id\_kelas FROM asisten\_pti\_kelas  WHERE shift = 'FTMD A') AS kelas  NATURAL JOIN asisten\_pti\_absen\_praktikan  NATURAL JOIN asisten\_pti\_praktikan | Query untuk melihat presensi berdasarkan kelas praktikum |
| 5 | SELECT nim, shift, pertemuan\_ke FROM asisten\_pti\_absen\_praktikan  NATURAL JOIN asisten\_pti\_praktikan  NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_kelas WHERE no\_kelas='A01' ORDER BY pertemuan\_ke | SELECT nim, shift, pertemuan\_ke FROM (select id\_praktikan, id\_kelas, nim from asisten\_pti\_praktikan where no\_kelas='A01') as praktikan  NATURAL JOIN asisten\_pti\_absen\_praktikan  NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_kelas ORDER BY pertemuan\_ke | Query untuk melihat presensi berdasarkan kelas kuliah |
| 6 | SELECT \* FROM asisten\_pti\_average\_nilai | SELECT \* FROM asisten\_pti\_average\_nilai | Query untuk melihat nilai rata-rata praktikum dari masing-masing mahasiswa |
| 7 | SELECT nama, shift, nama\_hari, minggu, jam  FROM asisten\_pti\_user NATURAL JOIN asisten\_pti\_pilihan\_jadwal NATURAL JOIN asisten\_pti\_kelas NATURAL JOIN asisten\_pti\_kelas\_info  WHERE approved = 1  AND nim = 13514032 | SELECT nama, shift, nama\_hari, minggu, jam  FROM (SELECT nim, nama FROM asisten\_pti\_user WHERE nim = 13514032) AS pengguna NATURAL JOIN (SELECT id\_kelas, nim FROM asisten\_pti\_pilihan\_jadwal WHERE approved = 1) AS kelas NATURAL JOIN asisten\_pti\_kelas NATURAL JOIN asisten\_pti\_kelas\_info | Query untuk melihat data kelas yang menjadi tanggung jawab asisten tertentu |
| 8 | SELECT nama, pertemuan\_ke, honor FROM asisten\_pti\_absen\_asisten NATURAL JOIN asisten\_pti\_pertemuan NATURAL JOIN asisten\_pti\_user WHERE nim = 13514032 | SELECT nama, pertemuan\_ke, honor FROM (SELECT nim, nama FROM asisten\_pti\_user WHERE nim = 13514032) AS pengguna  NATURAL JOIN asisten\_pti\_pertemuan  NATURAL JOIN asisten\_pti\_absen\_asisten | Query untuk melihat presensi asisten beserta honor |
| 9 | SELECT nama, shift, is\_korlas  FROM asisten\_pti\_user NATURAL JOIN asisten\_pti\_pilihan\_jadwal NATURAL JOIN asisten\_pti\_kelas WHERE shift = 'FTSL E' AND approved = 1 | SELECT nama, shift, is\_korlas  FROM asisten\_pti\_user  NATURAL JOIN asisten\_pti\_pilihan\_jadwal  NATURAL JOIN asisten\_pti\_kelas  WHERE shift = 'FTSL E' AND approved = 1 | Query untuk melihat korlas dari sebuah shift |

### Hasil Pengujian

Performansi awal

|  |  |  |
| --- | --- | --- |
| Query | Time | Cost |
| 1 | 651 | 263 |
| 2 | 919 | 1605.89 |
| 3 | 281 | 1397.45 |
| 4 | 507 | 332.61 |
| 5 | 283 | 251.64 |
| 6 | 568 | 2470.54 |
| 7 | 536 | 42.61 |
| 8 | 994 | 372.58 |
| 9 | 984 | 270.6 |

Performansi setelah dilakukan Schema Tuning, Index Tuning dan Query Tuning

|  |  |  |
| --- | --- | --- |
| Query | Time | Cost |
| 1 | 492 | 225 |
| 2 | 622 | 1165.58 |
| 3 | 227 | 1165.58 |
| 4 | 268 | 1869.11 |
| 5 | 251 | 250.24 |
| 6 | 424 | 114.34 |
| 7 | 861 | 116.87 |
| 8 | 740 | 372.58 |
| 9 | 707 | 232.43 |

Setelah dituning, query nomor 7 menjadi lebih lama karena pada tabel asisten\_pti\_kelas dilakukan vertical tuning. Pada vertical tuning tersebut, nama\_hari, jam, dan minggu dipisahkan ke tabel lain. Hal ini karena query yang lebih sering diakses pada umumnya, yaitu query yang berhubungan dengan nilai, nama\_hari, jam, dan minggu tidak diakses. Proses tuning yang dilakukan lebih mengutamakan query yang sering diakses sehingga dilakukan vertical split pada tabel asisten\_pti\_kelas.

Proses tuning membuat pemrosesan query menjadi lebih cepat. Namun, proses tuning seperti menambahkan add derived column dan add redundant column bisa mempersulit saat ingin mengubah atau menambahkan data karena kolum yang perlu diubah datanya bisa lebih dari satu.

## Recovery

### Proses Implementasi

#### Write-Ahead Log

Pada *write-ahead logging*, perubahan-perubahan yang dilakukan pada arsip data (tabel dan indeks) di-*log* terlebih dahulu, lalu dieksekusi. Proses eksekusi perubahan dilakukan setelah *log record* disimpan pada media penyimpanan permanen. WAL memungkinkan perubahan yang terjadi pada basis data tidak hilang pada saat sistem mengalami *crash*. Pada PostgreSQL 9.6, WAL diimplementasikan dengan cara :

1. Membuka *file* postgresql.conf pada direktori PostgreSQL/9.6/data.
2. Pada *file* postgresql.conf, dilakukan perubahan berupa

wal\_level = replica

wal\_level dibuat menjadi replica karena pada replica merupakan level tertinggi untuk wal. Replica memungkinkan *query* UPDATE dan DELETE dieksekusi.

archive\_command berisi aksi yang dilakukan dan *path* tempat dikeluarkannya hasil *log*.

#### Continuous Archiving

*Continuous archiving* adalah salah satu metode *recovery* pada basis data. Pada *continuous archiving*, data pada basis data di-*back-up* terus menerus setiap satuan ukuran data. Pada PostgreSQL 9.6, *continuous archiving* diimplementasikan dengan cara :

1. Pada *file* postgresql.conf, dilakukan perubahan berupa

archive\_mode = on

archive\_command = ‘copy “%p” “C:\\archive\\%f”’

1. archive\_command berisi aksi yang dilakukan dan *path* tempat dikeluarkannya hasil *log*.
2. Pada *command prompt,* dilakukan *command*

select pg\_start\_backup(‘backup\_name’);

1. Lalu, dilakukan *command*

select pg\_stop\_backup();

#### Point-in-time Recovery

*Point-in-time recovery* adalah mengembalikan basis data pada kondisi pada waktu dan tanggal yang telah ditentukan. *Point-in-time* *recovery* dapat digunakan jika terjadi kesalahan saat pengelolaan basis data sehingga basis data dapat dikembalikan ke kondisi sebelum terjadi kesalahan. Pada PostgreSQL 9.6, *point-in-time recovery* diimplementasikan dengan cara :

1. *Server* PostgreSQL dimatikan dengan :

NET STOP postgresql-x64-9.6 (untuk 64-bit PostgreSQL 9.6)

1. Semua *file* pada folder PostgreSQL/9.6/data/pg\_xlog kecuali *sub folder* “archive\_status”.
2. Semua *file PARTIALLY FILLED WAL* yang sudah dibuat sebelumnya pada *folder* pg\_xlog.
3. Dibuat *file* dengan nama “recovery.conf” dan masukkan hal berikut :

restore\_command = ‘copy “C:\\archive\\%f” “%p”’

recovery\_target\_time = ‘timestamp’

recovery\_target\_xid = ‘’

recovery\_target\_inclusive = ‘true’

Timestamp adalah waktu target recovery dengan format seperti 2016-11-04 22:39:00 EST

1. *Server* PostgreSQL dihidupkan kembali dengan :

NET START postgresql-x64-9.6 (untuk 64-bit PostgreSQL 9.6)

1. *Server* akan masuk ke model *recovery* dan memroses pembacaan *file* WAL. Jika proses gagal karena *external error*, *server* cukup di-*restart* untuk melanjutkan *recovery*. Setelah proses selesai, *server* akan mengubah nama “recovery*.*conf” menjadi “recovery.done”.

### Skenario Pengujian

1. Pada postgresql.conf, lakukan perubahan seperti berikut:

wal\_level = replica

archive\_mode = on

archive\_command = ‘copy “%p” “C:\\archive\\%f”’

dengan C:\\archive\\ merupakan folder backup external. Lalu restart service postgresql

NET STOP postgresql-x64-9.6 (untuk 64-bit PostgreSQL 9.6)

NET START postgresql-x64-9.6 (untuk 64-bit PostgreSQL 9.6)

2. Membuat sebuah tabel bernama testPITR1. Usahakan tabel tersebut berukuran cukup besar untuk menjamin ada WAL yang masuk ke folder archive.

CREATE TABLE testPITR1 as select \* from asisten\_pti\_user , asisten\_pti\_absen\_praktikan;

jika perintah berhasil dieksekusi, akan muncul perintah seperti berikut

SELECT 833654

3. Sekarang, kita coba melakukan full backup database kita dengan perintah :

select pg\_start\_backup('Full Backup');

Pada folder data, lakukan proses zip pada seluruh file yang ada di sana. File-file tersebut merupakan backup keseluruhan database, termasuk konfigurasi. Kemudian lakukan perintah

select pg\_stop\_backup();

4. Untuk mengetes proses recovery, kita buat 2 buah tabel baru. Untuk masing-masing tabel, catat waktu pembuatannya.

Testpitr2 -- 2016-11-11 20:02:13

Testpitr3 -- 2016-11-11 20:04:47

5. Lakukan simulasi crash pada postgresql, bisa dengan meng-kill secara paksa proses postgresql, atau bisa dengan melakukan force shutdown pada komputer.

6. Ubah nama folder data pada tempat instalasi Postgres menjadi data.old. Pada tempat yang sama, buat folder baru bernama data, dan extract hasil backup database (hasil zip pada langkah 3). Pada langkah ini, database telah kembali pada state setelah backup dilakukan.

7. Hapus semua file WAL pada folder data/pg\_xlog. Kemudian copy seluruh file WAL pada data.old/pg\_xlog ke data/pg\_xlog

8. Buat file bernama recovery.conf pada folder data. Isi file tersebut dengan perintah berikut :

restore\_command = 'copy "C:\\archive\\%f" "%p"'

recovery\_target\_time = '2016-11-11 20:02:13'

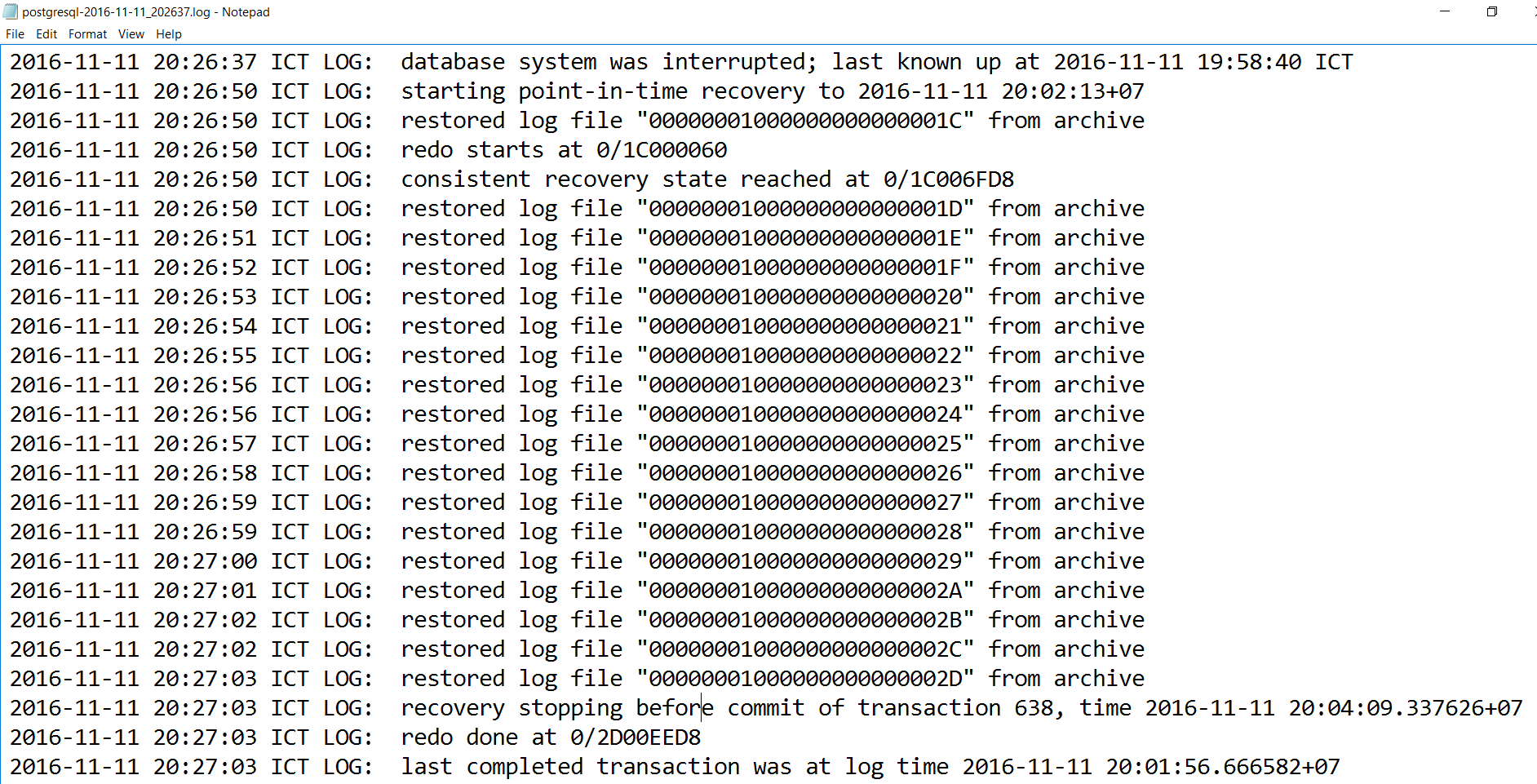
recovery\_target\_inclusive = 'false'

Perhatikan bahwa 2016-11-11 20:02:13 merupakan timestamp pembuatan tabel TestPITR2.

9. Untuk memulai proses recovery, jalankan postgresql

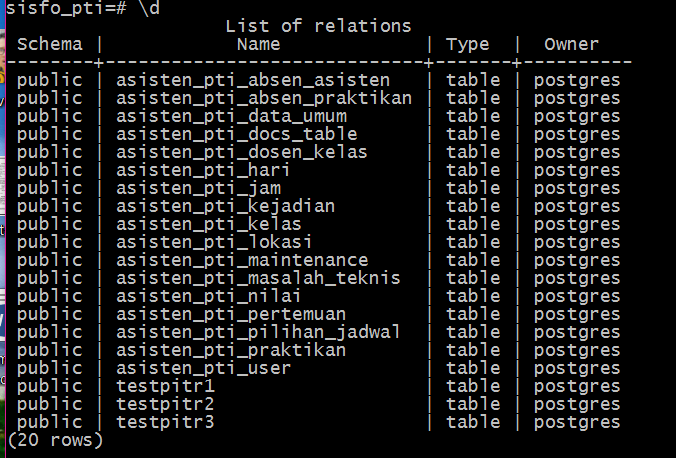
C:\Program Files\PostgreSQL\9.6>pg\_ctl start -D data

10. Jika proses recovery berhasil, file recovery.conf akan direname menjadi recovery.done, serta pada file log terbaru dalam pg\_log akan berisi keterangan seperti berikut :



### Hasil Pengujian

Sebelum *recovery*



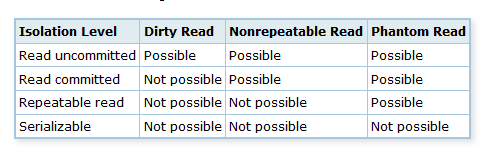
Setelah *recovery*



## Concurrency

### Proses Implementasi

Teknik konkurensi yang digunakan adalah teknik Transaction Isolation Level, pada teknik ini terdapat 4 macam level yang digunakan yaitu read uncommitted, read committed, repeatable read, dan serializable. Setiap isolation level memiliki behaviour yang berbeda-beda yang dapat dilihat pada tabel di bawah ini.



Berikut ini adalah penjelasan dan detil proses implementasi dari setiap teknik yang digunakan:

#### Read Uncommitted

Pada level ini, dimungkinkan untuk melakukan dirty read, nonrepeatable read, phantom read. Teknik ini memungkinkan untuk melakukan transaksi membaca data yang mungkin belum selesai di-commit pada transaksi lain. Proses ini harus dilakukan karena web DITSI ITB memiliki server yang kurang optimal sehingga teknik transaction isolation level ini digunakan pada query yang membutuhkan waktu yang lama dikarenakan level read uncommitted memastikan performansi yang paling cepat.

#### Read Committed

Pada level ini, dimungkinkan untuk melakukan nonrepeatable read, dan phantom read. Akan tetapi pada level ini, tidak dimungkinkan untuk melakukan dirty read sehingga pengguna tidak dapat melihat perubahan yang dilakukan oleh pengguna lain pada waktu yang bersamaan. Proses ini digunakan karena pada web DITSI ITB untuk melakukan update data seperti input nilai mahasiswa dibutuhkan validasi untuk memastikan data yang dimasukkan sudah benar.

#### Repeatable Read

Pada level ini, dimungkinkan untuk melakukan phantom read, tetapi tidak dimungkinkan untuk melakukan dirty read, dan norepeatable read sehingga transaksi update data tidak dapat dilakukan secara bersamaan dengan transaksi pembacaan data. Akan tetapi, transaksi pemasukkan data baru dapat dilakukan bersamaan dengan transaksi pembacaan data. Proses ini perlu dilakukan pada data absensi dimana tidak boleh dilakukan penggantian data, sedangkan boleh dilakukan penambahan data.

#### Serializable

Pada level ini, tidak dapat dilakukan dirty read, nonrepeatable read, dan phantom read sehingga jumlah konkurensi yang dapat dilakukan terbatas. Proses ini perlu dilakukan untuk memastikan konsistensi data karena setiap transaksi harus divalidasi terlebih dahulu sebelum melakukan transaksi berikutnya.

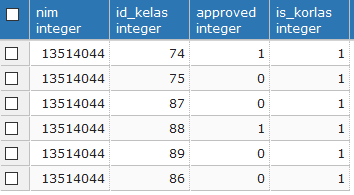
### Skenario Pengujian

|  |  |  |
| --- | --- | --- |
| Pengujian | Transaksi-1 | Transaksi-2 |
| Meng-*update* table asisten\_pti\_pilihan\_jadwal yang memiliki nim 13514044 dengan mengganti is\_korlas = 1, kemudian memilih tabel asisten\_pti\_pilihan\_jadwal dengan nim 13514044 | BEGIN; update asisten\_pti\_pilihan\_jadwal set is\_korlas = 0 where nim = '13514044'; COMMIT; | BEGIN; SELECT pg\_sleep(2); select \* from asisten\_pti\_pilihan\_jadwal where nim = '13514044'; COMMIT; |
| Meng-*update* table asisten\_pti\_pilihan\_jadwal dengan *isolation repetable read* | BEGIN ISOLATION LEVEL REPEATABLE READ; SELECT pg\_sleep(2); update asisten\_pti\_pilihan\_jadwal set is\_korlas = 0 where nim = '13514044'; COMMIT; | BEGIN ISOLATION LEVEL REPEATABLE READ; select \* from asisten\_pti\_pilihan\_jadwal where nim = '13514044'; COMMIT; |
| Memasikkan data ke asisten\_pti\_pilihan\_jadwal kemudian transaksi lain *read* tabel yang sama dengan *isolation repeatable reas* | BEGIN ISOLATION LEVEL REPEATABLE READ;  select pg\_sleep(2);  insert into asisten\_pti\_pilihan\_jadwal values (13514044, 90, 0, 0);  COMMIT; | BEGIN ISOLATION LEVEL REPEATABLE READ;  select \* from asisten\_pti\_pilihan\_jadwal where nim = 13514044;  COMMIT; |
| Melakukan update data pada tabel asisten\_pti\_pilihan\_jadwal kemudian dilakukan *read* pada tabel yang sama untuk mengecek *isolaton read commited* | BEGIN ISOLATION LEVEL READ UNCOMMITTED;  SELECT pg\_sleep(2);  update asisten\_pti\_pilihan\_jadwal set is\_korlas = 1 where nim = '13514044';  COMMIT; | BEGIN ISOLATION LEVEL READ UNCOMMITTED;  select \* from asisten\_pti\_pilihan\_jadwal where nim = '13514044';  COMMIT; |
| Melakukan update data pada tabel asisten\_pti\_pilihan\_jadwal kemudian dilakukan *read* pada tabel yang sama untuk mengecek *isolaton serializable* | BEGIN ISOLATION LEVEL SERIALIZABLE;  SELECT pg\_sleep(2);  update asisten\_pti\_pilihan\_jadwal set is\_korlas = 0 where nim = '13514044';  COMMIT; | BEGIN ISOLATION LEVEL SERIALIZABLE;  select \* from asisten\_pti\_pilihan\_jadwal where nim = '13514044';  COMMIT; |

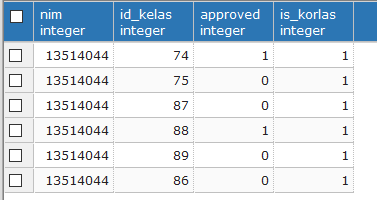
### Hasil Pengujian

#### Pengujian 1

Sebelum

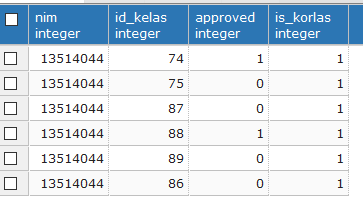


Sesudah

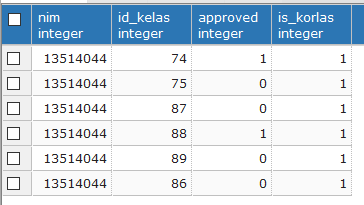


#### Pengujian 2

Sebelum

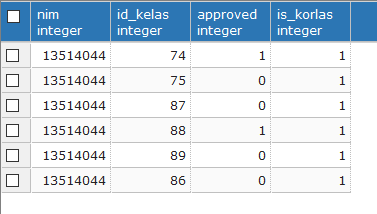


Sesudah

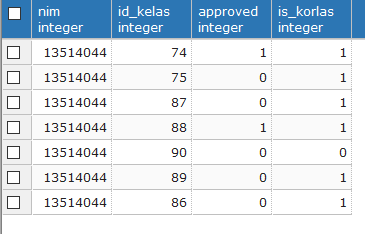


#### Pengujian 3

Sebelum

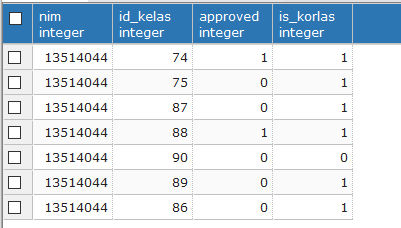


Sesudah

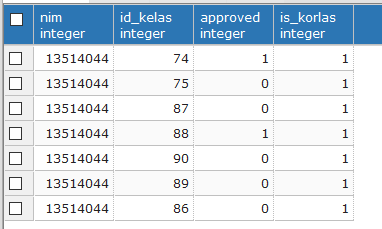


#### Pengujian 4

Sebelum

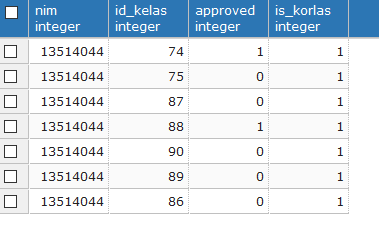


Sesudah

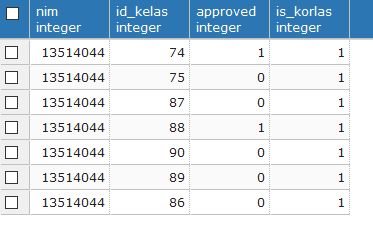


#### Pengujian 5

Sebelum



Sesudah



Kelebihan dari proses implementasi konkurensi yang telah dilakukan adalah konsistensi dari data lebih terjaga dan error dapat diminimalisir. Minimalisasi error juga dapat mengakibatkan minimalisasi rollback yang dilakukan saat error terjadi. Masalah yang didapatkan adalah pada beberapa proses konkurensi seperti serializable mengakibatkan waktu menjadi lebih lama.

## Security and User Management

### Proses Implementasi

#### Discretionary Access Control : Account Level

*Discretionary access control* pada level *account* adalah suatu model untuk mengontrol akses basis data di mana pemilik basis data menentukan *user* mana saja yang bisa mengakses basis data serta jenis *authorization* yang diberikan pada masing-masing *user* (jenis-jenis *authorization* : *read* (*select*), *insert*, *update*, dan *delete*). Pada PostgreSQL, proses ini diimplementasikan denga menggunakan GRANT. Pemilik basis data bisa memilih pengguna mana yang diberi akses beserta *authorization* dengan metode tersebut.

Contoh kasus (akan digunakan juga untuk model kontrol akses yang lain beserta pengujian) :

Pihak Ditsti selaku pemilik basis data SISFO PTI mempekerjakan Chalvin untuk membantu mengelola basis data. Chalvin diberi hak akses untuk melihat, menambah, memperbarui, serta menghapus data pada basis data. SISFO PTI juga mempekerjakan beberapa orang asisten untuk membantu mengawasi praktikum. Salah satu asisten yang dipekerjakan oleh Diststi adalah Kristianto. Dalam melaksanakan tugasnya, asisten juga perlu melihat, menambah, memperbarui, serta menghapus data pada basis data sehingga diberi hak akses untuk melakukan hal-hal tersebut. Ditsti juga memberikan hak akses basis datanya pada mahasiswa peserta praktikum PTI. Salah satu mahasiswa peserta praktikum PTI adalah Bella. Hak akses yang diberikan pada mahasiswa peserta praktikum adalah hak akses untuk melihat basis data, tanpa hak akses lainnya.

Query sebelum implementasi :

CREATE USER Chalvin;

CREATE USER Kristianto;

CREATE USER Bella;

1. Chalvin

Chalvin mendapatkan *privilege* tertinggi sehingga dapat melakukan semua operasi read/write/append pada basis data.

Implementasi

GRANT SELECT, INSERT, UPDATE, DELETE ON all tables IN schema public TO Chalvin;

Alasan

Karena Chalvin dapat melakukan semua operasi maka diberikan GRANT select, insert, dan update.

1. Kristianto

Kristianto dalam konteks *Account Level* memiliki *privileges* yang sama dengan Chalvin karena dalam sistem SISFO PTI, asisten dapat melakukan *update/insert/delete* data (mis: nilai dan absen praktikan).

Implementasi

GRANT SELECT, INSERT, UPDATE, DELETE ON all tables IN schema public TO Kristianto;

Alasan

Karena Kristianto dapat melakukan semua operasi maka diberikan GRANT select, insert, delete, dan update.

1. Bella

Praktikan merupakan user dengan *privilege* terendah yang dapat dilihat secara umum (*public*).

Implementasi

GRANT SELECT ON all tables IN schema public TO Bella;

Alasan

Karena praktikan hanya dapat melihat (tidak bisa melakukan perubahan nilai) melihat tabel maka diberi GRANT select saja.

#### Discretionary Access Control : Relation Level

*Discretionary access control* pada level relasi adalah suatu model untuk mengontrol akses basis data dengan cara menentukan relasi mana saja (beserta kolomnya) yang bisa diakses oleh suatu *user* beserta bentuk aksesnya (*select, insert, update,* dan *delete*). Pada PostgreSQL, proses ini juga diimplementasikan dengan menggunakan GRANT.

Query sebelum implementasi :

DROP OWNED BY Kristianto;

DROP OWNED BY Bella;

1. Chalvin

Chalvin bisa melakukan semua operasi pada semua tabel pada basis data sehingga tidak ada perubahan hak akses dari sebelumnya.

1. Kristianto

Kristianto hanya dapat melakukan operasi INSERT, UPDATE, dan DELETE pada beberapa tabel tertentu yang merupakan haknya. Berikut tabel yang dibatasi haknya :

* Tabel asisten\_pti\_absen\_asisten, hanya dapat melakukan operasi SELECT dan INSERT. Asisten yang merupakan koordinator kelas dapat memasukkan absen asisten pada suatu kelas namun tidak dapat melakukan perubahan lagi apabila sudah memasukkan data.
* Tabel asisten\_pti\_data\_umum, hanya dapat melakukan operasi SELECT karena data\_umum memiliki kolum honor, link facebook, dsb yang hanya dapat diubah oleh seorang admin.
* Tabel asisten\_pti\_kelas dan tabel asisten\_pti\_kelas\_info, yang memasukkan kelas adalah seorang admin sehingga asisten hanya dapat melakukan operasi SELECT.
* Tabel asisten\_pti\_pilihan\_jadwal, yang menetapkan seorang asisten menjadi asisten pada sebuah kelas adalah admin sehingga asisten hanya dapat melakukan operasi SELECT.
* Tabel asisten\_pti\_praktikan, yang memasukkan data praktikan adalah seorang admin sehingga asisten hanya dapat melakukan operasi SELECT.
* Tabel asisten\_pti\_user, tabel asisten\_pti\_user memiliki kolom lulus\_pti\_a, lulus\_pti\_b, dan role. Kolom tersebut tidak boleh diubah oleh asisten sehingga asisten hanya mendapat *privilege* SELECT pada kolom tersebut.

Selain itu, asisten juga tidak diberikan GRANT pada Tabel asisten\_pti\_docs\_table dan Tabel asisten\_pti\_maintenance karena tabel tersebut hanya dapat diubah oleh seorang admin. Untuk tabel lainnya, asisten dapat melakukan operasi SELECT, INSERT, UPDATE, dan DELETE karena berhubungan dengan perannya seperti menambahkan absen praktikan.

Implementasi

GRANT SELECT, INSERT ON TABLE public.asisten\_pti\_absen\_asisten TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_absen\_praktikan TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_average\_nilai TO Kristianto;

GRANT SELECT ON TABLE public.asisten\_pti\_data\_umum TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_kejadian TO Kristianto;

GRANT SELECT ON TABLE public.asisten\_pti\_kelas TO Kristianto;

GRANT SELECT ON TABLE public.asisten\_pti\_kelas\_info TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_masalah\_teknis TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_nilai TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_pertemuan TO Kristianto;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_pertemuan\_info TO Kristianto;

GRANT SELECT ON TABLE public.asisten\_pti\_pilihan\_jadwal TO Kristianto;

GRANT SELECT ON TABLE public.asisten\_pti\_praktikan TO Kristianto;

GRANT SELECT, INSERT(nim, nama, password, email), UPDATE(nim, nama, password, email) ON TABLE public.asisten\_pti\_user TO Kristianto;

GRANT SELECT, INSERT, UPDATE ON TABLE public.asisten\_pti\_user\_info TO Kristianto;

1. Bella

Praktikan hanya dapat melihat beberapa tabel sehingga hanya dapat melakukan SELECT.

Beberapa tabel tidak berhubungan dengan role praktikan sehingga tidak diberikan role GRANT. Berikut tabel yang tidak diberikan GRANT pada praktikan:

* Tabel asisten\_pti\_masalah\_teknis, masalah teknis hanya dapat diisi dan dillihat oleh asisten/admin.
* Tabel asisten\_pti\_user dan asisten\_pti\_user\_info, merupakan tabel informasi user asisten sehingga praktikan tidak boleh melihatnya
* Tabel asisten\_pti\_data\_umum
* Tabel asisten\_pti\_docs\_table
* Tabel asisten\_pti\_maintenance

Selain itu, pada Tabel asisten\_pti\_kelas\_info, public tidak boleh mengakses link docs karena merupakan link docs ke sheet penilaian.

Implementasi

GRANT SELECT ON TABLE public.asisten\_pti\_absen\_praktikan TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_pilihan\_jadwal TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_average\_nilai TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_kelas TO Bella;

GRANT SELECT (id\_kelas, minggu, nama\_hari, jam, nama\_lokasi) ON TABLE public.asisten\_pti\_kelas\_info TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_nilai TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_pertemuan TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_pertemuan\_info TO Bella;

GRANT SELECT ON TABLE public.asisten\_pti\_praktikan TO Bella;

#### Role-based Access Control

*Role-based access control* adalah suatu model untuk mengontrol akses basis data dengan cara memberi *role* pada masing-masing pengguna dan mengatur hak akses dari masing-masing *role* sehingga hak akses dari suatu pengguna bergantung pada *role*-nya.

Ada tiga *role* dalam SISFO PTI, yaitu admin, asisten, dan praktikan. Hak akses dari admin sama dengan Chalvin pada DAC, hak akses dari asisten sama dengan Kristianto pada DAC , dan hak akses praktikan sama dengan hak akses Bella pada DAC. Perbedaannya adalah dalam RBAC, suatu pengguna diberi hak akses dalam bentuk *role*, sehingga masing-masing pengguna cukup ditentukan *role*-nya, tanpa perlu diberi hak akses satu per satu. RBAC sangat cocok digunakan pada basis data SISFO PTI karena pada dasarnya, pengakses basis data SISFO PTI yang digunakan untuk praktikum PTI, sudah terbagi menjadi tiga, yaitu para asisten, admin, dan para praktikan. Misalkan asisten lain yang bisa mengakses basis data SISFO PTI selain Kristianto adalah Scarletta dan Malvin, dan praktikan lain yang bisa mengakses basis data SISFO PTI selain Bella adalah Mario dan Vincent. Selain Chalvin, SISFO PTI juga mempekerjakan Golfin untuk mengelola basis data.

Query sebelum implementasi :

CREATE ROLE Admin;

CREATE ROLE Asisten;

CREATE ROLE Praktikan;

CREATE USER Golfin;

CREATE USER Scarletta;

CREATE USER Malvin;

CREATE USER Mario;

CREATE USER Vincent;

1. Admin

Implementasi

GRANT SELECT, INSERT, UPDATE, DELETE ON all tables IN schema public TO Admin;

1. Asisten

Implementasi

GRANT SELECT, INSERT ON TABLE public.asisten\_pti\_absen\_asisten TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_absen\_praktikan TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_average\_nilai TO Asisten;

GRANT SELECT ON TABLE public.asisten\_pti\_data\_umum TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_kejadian TO Asisten;

GRANT SELECT ON TABLE public.asisten\_pti\_kelas TO Asisten;

GRANT SELECT ON TABLE public.asisten\_pti\_kelas\_info TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_kejadian TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_nilai TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_pertemuan TO Asisten;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE public.asisten\_pti\_pertemuan\_info TO Asisten;

GRANT SELECT ON TABLE public.asisten\_pti\_pilihan\_jadwal TO Asisten;

GRANT SELECT ON TABLE public.asisten\_pti\_praktikan TO Asisten;

GRANT SELECT, INSERT(nim, nama, password, email), UPDATE(nim, nama, password, email) ON TABLE public.asisten\_pti\_user TO Asisten;

GRANT SELECT, INSERT, UPDATE ON TABLE public.asisten\_pti\_user\_info TO Asisten;

1. Praktikan

Implementasi

GRANT SELECT ON TABLE public.asisten\_pti\_absen\_praktikan TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_pilihan\_jadwal TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_average\_nilai TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_kelas TO Praktikan;

GRANT SELECT (id\_kelas, minggu, nama\_hari, jam, nama\_lokasi) ON TABLE public.asisten\_pti\_kelas\_info TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_nilai TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_pertemuan TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_pertemuan\_info TO Praktikan;

GRANT SELECT ON TABLE public.asisten\_pti\_praktikan TO Praktikan;

Query setelah implementasi :

GRANT Admin TO Golfin;

GRANT Asisten TO Malvin;

GRANT Asisten TO Scarletta;

GRANT Praktikan TO Mario;

GRANT Praktikan TO Vincent;

### Skenario Pengujian

#### Discretionary Access Control : Account Level

|  |  |  |
| --- | --- | --- |
| **USER : Chalvin** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 1 | Insert into asisten\_pti\_absen\_asisten (id\_absen\_asisten, nim, id\_pertemuan, honor) values (100001, ’13514044’, ’30’, ’45000’); | Menguji hak INSERT |
| 2 | Update asisten\_pti\_kelas\_info set link\_docs=’sss’ where id\_kelas = 72; | Menguji hak UPDATE |
| 3 | Select \* from asisten\_pti\_pertemuan natural join asisten\_pti\_kelas; | Menguji hak SELECT |
| 4 | DELETE FROM asisten\_pti\_absen\_asisten where id\_absen\_asisten = 2 | Menguji hak DELETE |
| **USER : Kristianto** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 5 | Insert into asisten\_pti\_absen\_asisten (id\_absen\_asisten, nim, id\_pertemuan, honor) values (100001, ’13514044’, ’30’, ’45000’); | Menguji hak INSERT |
| 6 | Update asisten\_pti\_kelas\_info set link\_docs=’sss’ where id\_kelas = 72; | Menguji hak UPDATE |
| 7 | Select \* from asisten\_pti\_kelas\_info; | Menguji hak SELECT |
| 8 | DELETE FROM asisten\_pti\_kelas\_info where id\_kelas = 2; | Menguji hak DELETE |
| **USER : Bella** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 9 | Select \* from asisten\_pti\_user natural join asisten\_pti\_user\_info; | Menguji hak SELECT; |
| 10 | INSERT INTO asisten\_pti\_absen\_praktikan values (100000, 10, ’a’, 1); | Menguji hak INSERT pada asisten\_pti\_absen\_praktikan yang seharusnya tidak bisa dimasukkan oleh praktikan |
| 11 | Select \* from asisten\_pti\_average\_nilai; | Menguji hak SELECT |

#### Discretionary Access Control : Relation Level

|  |  |  |
| --- | --- | --- |
| **USER : Chalvin** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 1 | Insert into asisten\_pti\_absen\_asisten (id\_absen\_asisten, nim, id\_pertemuan, honor) values (100001, ’13514044’, ’30’, ’45000’); | Menguji hak INSERT |
| 2 | Update asisten\_pti\_kelas\_info set link\_docs=’sss’ where id\_kelas = 72; | Menguji hak UPDATE |
| 3 | Select \* from asisten\_pti\_pertemuan natural join asisten\_pti\_kelas; | Menguji hak SELECT |
| 4 | DELETE FROM asisten\_pti\_absen\_asisten where id\_absen\_asisten = 2 | Menguji hak DELETE |
| **USER : Kristianto** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 5 | Select \* from asisten\_pti\_maintenance; | Menguji hak pada asisten\_pti\_maintenance yang seharusnya tidak bisa diakses asisten |
| 6 | Update asisten\_pti\_user set role=’admin’ where nim = ’13514044’; | Menguji hak UPDATE pada asisten\_pti\_user pada kolum role yang seharusnya tidak bisa diakses asisten |
| 7 | Update asisten\_pti\_user set nama=’maju’ where nim = ’13514044’; | Menguji hak UPDATE |
| 8 | Insert into asisten\_pti\_kelas values (800, ’FTI’, ’PTI A’); | Menguji hak INSERT pada asisten\_pti\_kelas yang seharusnya tidak bisa ditambahkan oleh asisten |
| 9 | SELECT \* from asisten\_pti\_kelas; | Menguji hak SELECT |
| 10 | DELETE FROM asisten\_pti\_absen\_asisten where id\_absen\_asisten = 1 | Menguji hak DELETE pada asisten\_pti\_absen\_asisten yang seharusnya tidak bisa dihapus oleh asisten |
| 11 | DELETE FROM asisten\_pti\_absen\_praktikan where id\_praktikan = 11 | Menguji hak DELETE |
| **USER : Bella** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 12 | Select \* from asisten\_pti\_user natural join asisten\_pti\_user\_info; | Menguji hak pada asisten\_pti\_user yang seharusnya tidak bisa diakses praktikan |
| 13 | INSERT INTO asisten\_pti\_absen\_praktikan values (100000, 10, ’a’, 1); | Menguji hak INSERT pada asisten\_pti\_absen\_praktikan yang seharusnya tidak bisa dimasukkan oleh praktikan |
| 14 | Select \* from asisten\_pti\_average\_nilai; | Menguji hak SELECT |
| 15 | Update asisten\_pti\_praktikan set nama=’piki’ where id\_praktikan = 10; | Menguji hak UPDATE pada asistenpti\_praktikan yang seharusnya tidak bisa diakses oleh praktikan |
| 16 | Select link\_docs from asisten\_pti\_kelas\_info; | Menguji apakah kolum link\_docs bisa diakses praktikan |
| 17 | Select id\_kelas, minggu, nama\_hari from asisten\_pti\_kelas\_info; | Menguji hak SELECT; |

#### Role-based Access Control

|  |  |  |
| --- | --- | --- |
| **USER : Golfin, ROLE : Admin** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 1 | Insert into asisten\_pti\_absen\_asisten (id\_absen\_asisten, nim, id\_pertemuan, honor) values (500789, ’13514044’, ’30’, ’45000’); | Menguji hak INSERT |
| 2 | Update asisten\_pti\_pilihan\_jadwal set approved=’1’ where nim=’10115066’; | Menguji hak UPDATE |
| 3 | Select \* from asisten\_pti\_pertemuan natural join asisten\_pti\_kelas; | Menguji hak SELECT |
| 4 | DELETE FROM asisten\_pti\_absen\_asisten where id\_absen\_asisten = 10; | Menguji hak DELETE |
| **USER : Malvin, ROLE : Asisten** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 5 | Select \* from asisten\_pti\_docs\_table; | Menguji hak pada asisten\_pti\_docs\_table yang seharusnya tidak bisa diakses asisten |
| 6 | Update asisten\_pti\_pilihan\_jadwal set approved=’1’ where nim = ’10113001’; | Menguji hak UPDATE pada asisten\_pti\_pilihan\_jadwal pada kolum approved yang seharusnya tidak bisa diakses asisten |
| 7 | Update asisten\_pti\_average\_nilai set avg=56 where nim = ’16516274’; | Menguji hak UPDATE |
| 8 | Insert into asisten\_pti\_kelas (id\_kelas, shift, jenis) values (‘2600’,’FMIPA AB’,’PTI B’); | Menguji hak INSERT pada asisten\_pti\_kelas yang seharusnya tidak bisa ditambahkan oleh asisten |
| 9 | Select \* from asisten\_pti\_pertemuan; | Menguji hak SELECT |
| 10 | Delete from asisten\_pti\_praktikan where nim=’16116046’; | Menguji hak DELETE pada asisten\_pti\_absen\_praktikan yang seharusnya tidak bisa dihapus oleh asisten |
| 11 | Delete from asisten\_pti\_kejadian where id\_kms=’387’; | Menguji hak DELETE |
| **USER : Mario, ROLE : Praktikan** | | |
| **Skenario ke – i** | **Skenario** | **Penjelasan** |
| 12 | Select \* from asisten\_pti\_kejadian; | Menguji hak pada asisten\_pti\_user yang seharusnya tidak bisa diakses praktikan |
| 13 | INSERT INTO asisten\_pti\_average\_nilai (nim, avg) values (‘16514288’,’100’); | Menguji hak INSERT pada asisten\_pti\_average\_nilai yang seharusnya tidak bisa dimasukkan oleh praktikan |
| 14 | Select \* from asisten\_pti\_nilai; | Menguji hak SELECT |
| 15 | Update asisten\_pti\_praktikan set nama=’steffi’ where id\_praktikan = 10; | Menguji hak UPDATE pada asisten\_pti\_praktikan yang seharusnya tidak bisa diakses oleh praktikan |
| 16 | Select link\_docs from asisten\_pti\_kelas\_info; | Menguji apakah kolum link\_docs bisa diakses praktikan |
| 17 | Delete from asisten\_pti\_nilai where nim=’16616284’ | Menguji hak DELETE |

### Hasil Pengujian

#### Discretionary Access Control : Account Level

|  |
| --- |
| **USER : Chalvin** |
| **Skenario ke – 1** |
|  |
| **Skenario ke – 2** |
|  |
| **Skenario ke – 3** |
|  |
| **Skenario ke-4** |
|  |
| **USER : Kristianto** |
| **Skenario ke-5** |
|  |
| **Skenario ke-6** |
|  |
| **Skenario ke-7** |
|  |
| **Skenario ke-8** |
|  |
| **USER : Bella** |
| **Skenario ke-9** |
|  |
| **Skenario ke-10** |
|  |
| **Skenario ke-11** |
|  |

#### Discretionary Access Control : Relation Level

|  |
| --- |
| **USER : Chalvin** |
| **Skenario ke – 1** |
|  |
| **Skenario ke – 2** |
|  |
| **Skenario ke – 3** |
|  |
| **Skenario ke-4** |
|  |
| **USER : Kristianto** |
| **Skenario ke-5** |
|  |
| **Skenario ke-6** |
|  |
| **Skenario ke-7** |
|  |
| **Skenario ke-8** |
|  |
| **Skenario ke-9** |
|  |
| **Skenario ke-10** |
|  |
| **Skenario ke-11** |
|  |
| **USER : Bella** |
| **Skenario ke-12** |
|  |
| **Skenario ke-13** |
|  |
| **Skenario ke-14** |
|  |
| **Skenario ke-15** |
|  |
| **Skenario ke-16** |
|  |
| **Skenario ke-17** |
|  |

#### Role-based Access Control

|  |
| --- |
| **USER : Golfin, ROLE : Admin** |
| **Skenario ke – 1** |
|  |
| **Skenario ke – 2** |
|  |
| **Skenario ke – 3** |
|  |
| **Skenario ke-4** |
|  |
| **USER : Malvin, ROLE : Admin** |
| **Skenario ke-5** |
|  |
| **Skenario ke-6** |
|  |
| **Skenario ke-7** |
|  |
| **Skenario ke-8** |
|  |
| **Skenario ke-9** |
|  |
| **Skenario ke-10** |
|  |
| **Skenario ke-11** |
|  |
| **USER : Mario, ROLE : Praktikan** |
| **Skenario ke-12** |
|  |
| **Skenario ke-13** |
|  |
| **Skenario ke-14** |
|  |
| **Skenario ke-15** |
|  |
| **Skenario ke-16** |
|  |
| **Skenario ke-17** |
|  |

## Stored Procedure / Function

### Proses Implementasi

#### Stored Procedure / Function 1 : Penambahan Kelas

Implementasi :

CREATE OR REPLACE FUNCTION insert\_kelas(id\_kelas int, shift VARCHAR(20), jenis VARCHAR(20))

RETURNS void AS $$

BEGIN

Insert into asisten\_pti\_kelas values (id\_kelas, shift, jenis);

END;

$$ LANGUAGE plpgsql;

Contoh cara pemanggilan :

SELECT insert\_kelas(10003, ‘FTI’, ‘PTI A’);

Alasan :

Fungsi diatas perlu dibuat di level database karena query diatas sering diakses, sehingga untuk mempermudah dan menghindari kesalahan penulisan *query* dan proses *recompile* berulang-ulang maka dibuat dalam bentuk *stored procedure.*

#### Stored Procedure / Function 2 : Peng-*approve*-an Asisten

Implementasi :

CREATE OR REPLACE FUNCTION approve\_asisten(nim\_asisten integer)

RETURNS void AS $$

BEGIN

Update asisten\_pti\_pilihan\_jadwal set approved=’1’ where nim = nim\_asisten;

END;

$$ LANGUAGE plpgsql;

Contoh cara pemanggilan :

SELECT approve\_asisten(13514044);

Alasan :

Fungsi diatas perlu dibuat di level database karena query diatas sering diakses, sehingga untuk mempermudah dan menghindari kesalahan penulisan *query* dan proses *recompile* berulang-ulang maka dibuat dalam bentuk *stored procedure.*

#### Stored Procedure / Function 3 : Pembagian Kelas

Implementasi :

CREATE OR REPLACE FUNCTION pembagian\_kelas(s VARCHAR(20))

RETURNS TABLE( nim int, nama VARCHAR(50), no\_kelas VARCHAR(20), shift VARCHAR(20) ) AS

$$ SELECT nim, nama, no\_kelas, shift FROM (SELECT id\_kelas, shift FROM asisten\_pti\_kelas WHERE shift LIKE s) AS kelas NATURAL JOIN asisten\_pti\_praktikan

$$ LANGUAGE sql;

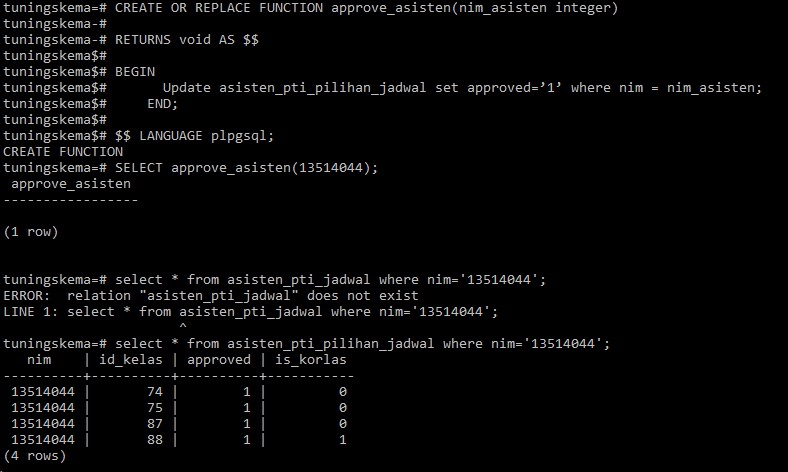
Contoh cara pemanggilan :

Select pembagian\_kelas(’SITH%’);

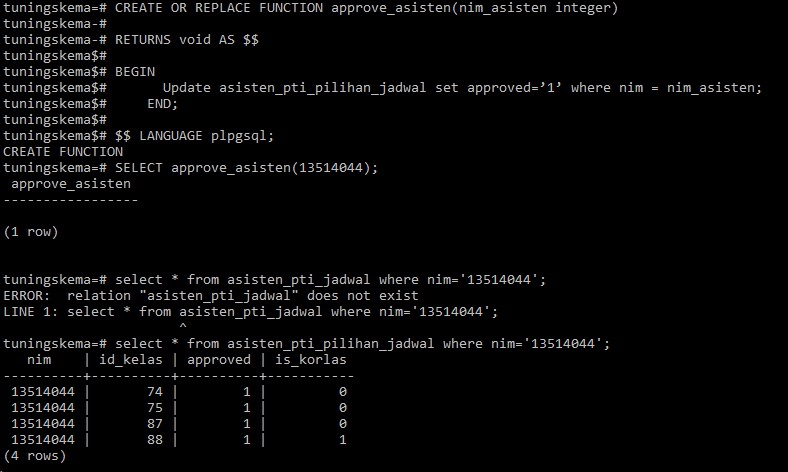
Alasan :

Fungsi diatas perlu dibuat di level database karena query diatas sering diakses, sehingga untuk mempermudah dan menghindari kesalahan penulisan *query* dan proses *recompile* berulang-ulang maka dibuat dalam bentuk *stored procedure.*

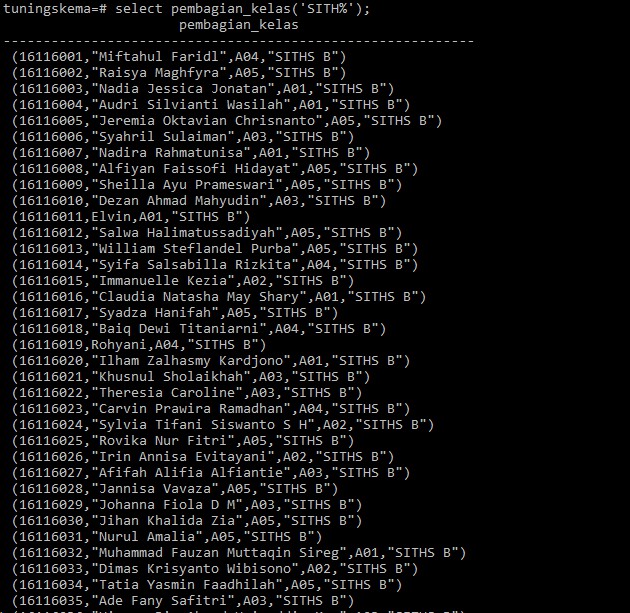
### Hasil Implementasi



Gambar 4.5.2.1 Hasil Implementasi Stored Procedure 1



Gambar 4.5.2.2 Hasil Implementasi Stored Procedure 2



Gambar 4.5.2.3 Hasil Implementasi Stored Procedure 3

*Stored procedure* hanya perlu dikompilasi saat pertama kali dibuat. Untuk selanjutnya, query yang dieksekusi dengan *stored procedure* langsung dijalankan tanpa perlu dilakukan pengecekan dan kompilasi ulang sehingga waktunya lebih cepat. Query yang kompleks juga bisa disederhanakan eksekusinya dengan dijadikan *stored procedure*.

## Trigger

### Proses Implementasi

#### Trigger 1 : Penambahan Data Log Asisten

Membuat tabel log asisten :

CREATE TABLE logasisten (nim INT NOT NULL, ENTRY\_DATE TEXT NOT NULL);

Membuat prosedur penambahan data waktu log asisten pada tabel logasisten :

CREATE OR REPLACE FUNCTION insert\_logasisten() RETURNS TRIGGER AS $$ BEGIN insert into logasisten(nim, ENTRY\_DATE) VALUES (new.nim, current\_timestamp); RETURN NEW; END; $$ LANGUAGE plpgsql;

Membuat prosedur trigger :

CREATE TRIGGER trigger\_logasisten AFTER INSERT ON asisten\_pti\_absen\_asisten FOR EACH ROW EXECUTE PROCEDURE insert\_logasisten();

Contoh pemanggilan :

INSERT INTO asisten\_pti\_absen\_asisten VALUES (100000, 10113073, 484, 45000);

Alasan :

*Trigger* dilakukan pada level database agar proses menjadi lebih cepat dan setiap dilakukan perubahan pada suatu tabel, tabel lain yang datanya bergantung pada tabel yang diubah bisa berubah juga sehingga meminimalisir kesalahan akibat terlalu banyak tabel yang perlu diperbarui.

#### Trigger 2 : Penambahan Data Log Praktikan

Membuat tabel log praktikan :

CREATE TABLE logpraktikan2 (id\_praktikan INT NOT NULL, ENTRY\_DATE TEXT NOT NULL);

Membuat prosedur penambahan data waktu log praktikan pada tabel logpraktikan :

CREATE OR REPLACE FUNCTION insert\_logpraktikan2() RETURNS TRIGGER AS $$ BEGIN insert into logpraktikan2(id\_praktikan, ENTRY\_DATE) VALUES (new.id\_praktikan, current\_timestamp); RETURN NEW; END; $$ LANGUAGE plpgsql;

Membuat prosedur trigger :

CREATE TRIGGER trigger\_logpraktikan3 AFTER INSERT ON asisten\_pti\_absen\_praktikan FOR EACH ROW EXECUTE PROCEDURE insert\_logpraktikan2();

Contoh pemanggilan :  
INSERT INTO asisten\_pti\_absen\_praktikan VALUES (100000, 1000, '', 750);

Alasan :

*Trigger* dilakukan pada level database agar proses menjadi lebih cepat dan setiap dilakukan perubahan pada suatu tabel, tabel lain yang datanya bergantung pada tabel yang diubah bisa berubah juga sehingga meminimalisir kesalahan akibat terlalu banyak tabel yang perlu diperbarui.

#### Trigger 3 : Perbaruan Tabel Nilai Rata-rata Praktikan

Membuat prosedur update data nilIai avergae pada tabel asisten\_pti\_average\_nilai :

CREATE OR REPLACE FUNCTION change\_average() RETURNS TRIGGER AS $$

BEGIN with nilai\_average as (

select avg(nilai) from asisten\_pti\_nilai where nim= new.nim)

UPDATE

asisten\_pti\_average\_nilai

SET avg = na.avg

from nilai\_average as na

WHERE nim = new.nim;

RETURN NEW;

END; $$ LANGUAGE plpgsql;

Membuat prosedur trigger :

CREATE TRIGGER trigger\_change\_average AFTER INSERT ON asisten\_pti\_nilai FOR EACH ROW EXECUTE PROCEDURE change\_average();

Contoh pemanggilan :

INSERT INTO asisten\_pti\_nilai VALUES (46831,'16617424', 17424, 100, 100, 100, '’, 7, 'FTSL B', 'B12');

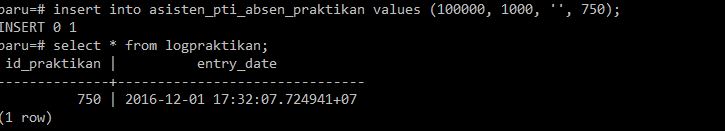
Alasan :

*Trigger* dilakukan pada level database agar proses menjadi lebih cepat dan setiap dilakukan perubahan pada suatu tabel, tabel lain yang datanya bergantung pada tabel yang diubah bisa berubah juga sehingga meminimalisir kesalahan akibat terlalu banyak tabel yang perlu diperbarui.

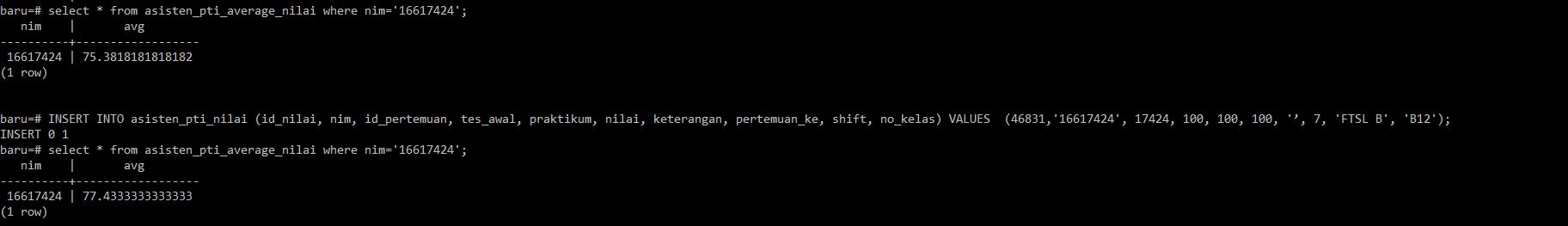
### Hasil Implementasi

## 

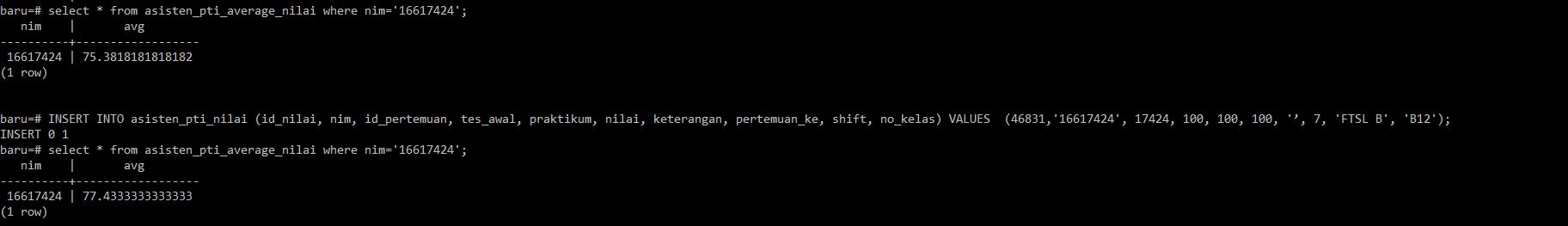
Gambar 4.6.2.1 Hasil Implementasi Trigger 1



Gambar 4.6.2.2 Hasil Implementasi Trigger 2



Gambar 4.6.2.3 Hasil Implementasi Trigger 3 (Nilai rata-rata sebelum tabel nilai diperbarui)



Gambar 4.6.2.4 Hasil Implementasi Trigger 3 (Nilai rata-rata setelah tabel nilai diperbarui)

Dengan adanya implementasi trigger, jika ingin dilakukan perubahan pada suatu tabel, maka tabel lain yang berhubungan (tanpa *key*) bisa berubah juga dengan sendirinya sehingga kesalahan akibat terlalu banyak tabel yang perlu diperbarui bisa diminimalisir.

# Kesimpulan

Tuning yang dilakukan pada sistem basisdata SISFO PTI secara umum membuat waktu dan cost untuk mem-fetch data lebih cepat. Tuning yang dilakukan membuat beberapa query menghasilkan waktu dan cost mendapatkan data lebih lambat. Namum, query tersebut jarang diakses sehingga implikasinya tidak terlalu signifikan.

Hal yang paling berpengaruh adalah ketika dilakukan schema tuning terdapat penambahan kolum maupun tabel baru yang merupakan hasil agregasi sehingga mempersulit saat melakukan update data. Selain itu, terdapat tabel agregasi sehingga tiap kali update data diperlukan proses yang lebih untuk mengupdate tabel tersebut.

Harapan kami, web SISFO PTI dapat beroperasi lebih optimal dengan waktu eksekusi yang lebih cepat. Namun, laporan ini hanya dapat menjadi referensi karena kami tidak menggunakan semua workload yang ada di web tersebut, hanya yang paling sering karena kami tidak mengetahui workload di belakang sistem.

# Pembagian Kerja dalam Kelompok

|  |  |
| --- | --- |
| **Nama** | **Tugas** |
| Catherine Pricilla | Membuat data *dummy, schema tuning, concurrency, DAC: account level*, *stored procedure* 1,laporan |
| Taufic Leonardo Sutejo | Membuat data *dummy, schema tuning, recovery*, *DAC: account level*, *stored procedure* 2 |
| Chalvin Polpan | Membuat *workload, recovery*, *DAC: relation level.* *stored procedure* 3 |
| Evita Chandra | *Query tuning, concurrency, DAC: relation level, trigger* 1,laporan |
| Malvin Juanda | *Query tuning, schema tuning, concurrency, RBAC, trigger* 2,laporan |
| Scarletta Julia Yapfrine | *Workload, index tuning, recovery, RBAC*, *trigger* 3,laporan |

# Lampiran

## Log Activity Anggota Kelompok

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama Anggota** | **Tanggal** | **Kegiatan** |
| 1. | Catherine Pricilla, Taufic Leonardo | 20 Oktober 2016 | Membuat data *dummy* |
| 2. | Chalvin Polpan, Scarletta Julia | 22 Oktober 2016 | Membuat *workload* |
| 3. | Malvin Juanda, Catherine Pricilla, Taufic Leonardo | 23 Oktober 2016 | Melakukan *Schema tuning* |
| 4. | Evita Chandra, Malvin Juanda | 25 Oktober 2016 | Melakukan *Query tuning* |
| 5. | Scarletta Julia Yapfrine | 27 Oktober 2016 | Melakukan *Index tuning* |
| 6. | Evita Chandra, Scarletta Julia Yapfrine, Catherine Pricilla, Malvin Juanda | 28 Oktober 2016 | Membuat laporan |
| 7 | Chalvin, Scarletta Julia Yapfrine, Taufic Leonardo | 5 November 2016 | Melakukan eksplorasi tentang recovery |
| 8 | Catherine Pricilla, Evita Chandra, Malvin Juanda | 5 November 2016 | Melakukan eksplorasi tentang concurrency |
| 9 | Chalvin, Catherine Pricilla, Evita Chandra, Malvin Juanda, Scarletta Julia Yapfrine, Taufic Leonardo | 6 November 2016 | Membahas hasil eksplorasi |
| 10 | Chalvin, Scarletta Julia Yapfrine, Taufic Leonardo | 10 November 2016 | Implementasi recovery |
| 11 | Catherine Pricilla, Evita Chandra, Malvin Juanda | 10 November 2016 | Implementasi concurrency |
| 12 | Catherine Pricilla, Taufic Leonardo Sutejo | 22 November 2016 | DAC: Account Level |
| 13 | Chalvin, Evita Chandra | 22 November 2016 | DAC : Relation Level |
| 14 | Malvin Juanda, Scarletta Julia Yapfrine | 22 November 2016 | RBAC |
| 15 | Catherine Pricilla, Taufic Leonardo Sutejo, Chalvin | 30 November 2016 | Stored Procedure |
| 16 | Evita Chandra, Malvin Juanda, Scarletta Julia Yapfrine | 30 November 2016 | Trigger |