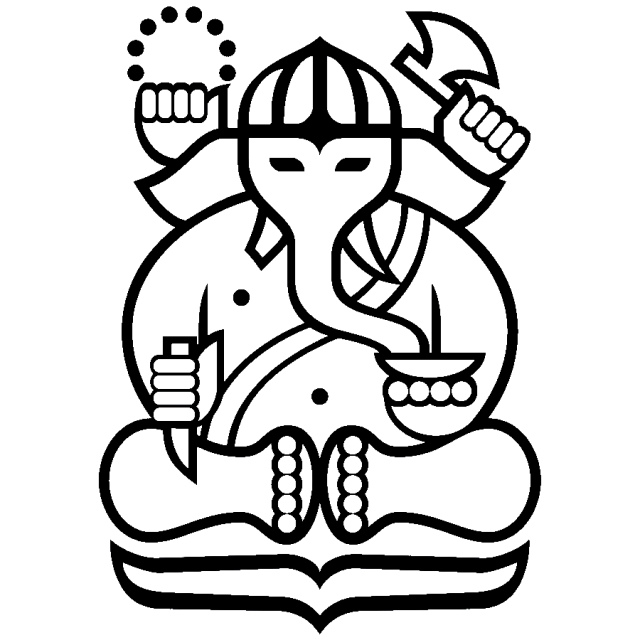
**IF3140 - Sistem Basis Data**

**Praktikum 0**

***Review* Basis Data Relasional dengan PostgreSQL**

27 September 2024

****

Erdianti Wiga Putri Andini

13522053

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2024**

**Petunjuk**

1. Praktikum ini bertujuan untuk membiasakan Anda dengan lingkungan DBMS PostgreSQL. PostgreSQL akan menjadi DBMS utama dalam rangkaian pembelajaran IF3140 Sistem Basis Data.
2. Setiap peserta kuliah **WAJIB** mengunduh, menginstal, menjalankan, dan mengerjakan praktikum ini pada server PostgreSQL pada perangkat masing masing.
3. Gunakan soal ini sebagai pembiasaan Anda dalam menggunakan PostgreSQL. Risiko tidak mengerjakan praktikum ini ditanggung oleh masing-masing. Soal praktikum akan memiliki tingkat kesulitan yang setara. Pengerjaan dengan batas waktu dapat menjadi opsi untuk berlatih.
4. Seluruh *keyword* yang ada diperbolehkan pada praktikum ini. Silahkan memilih *keyword* yang sesuai dengan kebutuhan soal.

**Bioskop**

| film = (idfilm, namafilm, tahun, durasi, imdbrating)  film\_has\_genre = (idfilm, genre)  film\_has\_kemampuanfilm = (idfilm, kemampuan)  genre = (genre)  kemampuanfilm = (kemampuan)  kursi = (idkursi, idstudio)  pelanggan = (idpelanggan, namapelanggan)  pelangganonline = (idpelanggan, email, passwordhash)  studio = (idstudio, namastudio)  studio\_has\_kemampuanfilm = (idstudio, kemampuan)  reservasi = (idfilm, idstudio, waktumulai, idkursi, idpelanggan)  penayangan = (idfilm, idstudio, waktumulai)  penayangan\_has\_kemampuanfilm = (idfilm, idstudio, waktumulai, kemampuan) |
| --- |

Silahkan eksplorasi *primary key*, *foreign key*, dan properti lainnya langsung melalui basis data.

**Bagian I. Operasi Dasar PostgreSQL**

# Nyalakan Server PostgreSQL

Jalankan file *docker* yang tersedia.

| Command Terminal | docker compose up |
| --- | --- |
| SS Terminal dan Hasil | |
|  | |

# Masuk ke PostgreSQL

Login ke PostgreSQL. Port yang digunakan adalah 5433, dengan **username praktikum dan password labdas**

| Command Terminal | psql -U praktikum -d postgres -h localhost -p 5433 |
| --- | --- |
| SS Terminal dan Hasil | |
|  | |

# Daftar Basis Data

Tampilkan daftar basis data yang ada.

| Query | \l |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Menuju Basis Data

Masuk ke basis data bioskop.

| Query | \c bioskop |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Daftar Tabel

Tampilkan daftar tabel yang ada pada basis data bioskop.

| Query | \dt |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Keyword EXPLAIN

Apa itu keyword EXPLAIN?

| EXPLAIN adalah keyword SQL yang digunakan untuk memberikan gambaran tentang rencana eksekusi query. EXPLAIN menunjukkan bagaimana PostgreSQL akan menjalankan suatu query, termasuk bagaimana data akan diambil, penggunaan indeks, biaya perkiraan, serta tahapan eksekusi yang akan dilakukan oleh query. |
| --- |

Berikan satu contoh penggunaan keyword EXPLAIN. Lalu, jelaskan apa yang terjadi!

| Query | EXPLAIN SELECT \* FROM \_film WHERE tahun > 2000; |
| --- | --- |
| SS Query dan Hasil | |
|  | |
| Penjelasan | |
| PostgreSQL memutuskan menggunakan Sequential Scan untuk memindai semua baris di tabel \_film, karena tidak ada indeks yang tersedia pada kolom tahun. Total query cost adalah 2.56, yang merupakan nilai kecil dan menunjukkan bahwa eksekusi query ini tidak terlalu mahal. PostgreSQL memperkirakan bahwa 73 baris akan dikembalikan sebagai hasil dari query ini.  Setiap baris yang dihasilkan diperkirakan memiliki lebar 28 byte. | |

# Properti \timing

Apa itu properti \timing?

| Properti \timing di PostgreSQL adalah perintah yang digunakan untuk menampilkan waktu eksekusi query. Dengan mengaktifkan \timing, akan ditampilkan berapa lama waktu yang diperlukan untuk mengeksekusi query setelah setiap query selesai dijalankan. |
| --- |

Berikan satu contoh penggunaan properti \timing. Lalu, jelaskan apa yang terjadi!

| Query | \timing  bioskop=# SELECT \* FROM \_film WHERE durasi > 115; |
| --- | --- |
| SS Query dan Hasil | |
|  | |
| Penjelasan | |
| Setelah mengaktifkan properti \timing, query SELECT \* FROM \_film WHERE durasi > 115; dieksekusi untuk mengambil semua kolom dari tabel \_film dengan durasi film lebih dari 115 menit. PostgreSQL akan melakukan pemindaian pada tabel tersebut, menggunakan Sequential Scan jika tidak ada indeks pada kolom durasi, atau Index Scan jika ada indeks yang relevan. Setelah query dieksekusi, PostgreSQL menampilkan hasilnya bersama dengan waktu eksekusi yaitu Time: 1.012 ms, yang menunjukkan berapa lama query tersebut berjalan. Penggunaan \timing memungkinkan untuk mengukur efisiensi query dan melakukan analisis kinerja. Jika waktu eksekusi tinggi, ini dapat menunjukkan perlunya optimasi. | |

# Eksekusi Query secara Paralel

Kita akan menjalankan suatu query menggunakan kakas query-executor.jar. Kakas ini akan menjalankan 30 thread untuk sebuah query. Silakan mengunduh kakas tersebut. Taruh satu saja pada query.sql pada direktori yang sama kemudian jalankan kakas tersebut dengan perintah “java -jar query-executor.jar”. Pastikan pada komputer Anda, sudah terpasang Java

| Query Bebas | SELECT \* FROM \_film WHERE durasi > 115; |
| --- | --- |
| SS Hasil Kakas | |
|  | |

**Bagian II. PostgreSQL DDL & DML**

*Kerjakan No 1-5 terlebih dahulu. Selanjutnya, kerjakan No 6-10 secara berurutan. Untuk seluruh soal, aktifkan \timing dan masukkan hasil perhitungan waktu dalam tangkapan layar.*

# Rekap Penjualan Tiket

Tampilkan berapa jumlah kursi yang direservasi per bulan, urutkan dari bulan yang memiliki reservasi kursi paling banyak.

| Query | SELECT EXTRACT(MONTH from TO\_TIMESTAMP(waktumulai, 'YYYY-MM-DD HH24:MI:SS')) AS bulan, COUNT(idkursi) AS jumlah\_kursi  FROM \_reservasi  GROUP BY bulan  ORDER BY jumlah\_kursi DESC; |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Genre Film Terfavorit

Tampilkan genre terfavorit (ditunjukkan dengan penjualan tiket yang paling tinggi) dari seluruh film yang dirilis setelah tahun 1995. Berikan pula total jumlah tiket yang terjual

| Query | SELECT fg.genre, COUNT(r.idkursi) AS total\_tiket  FROM \_film f  JOIN \_film\_has\_genre fg ON f.idfilm = fg.idfilm  JOIN \_reservasi r ON f.idfilm = r.idfilm  WHERE f.tahun > 1995  GROUP BY fg.genre  ORDER BY total\_tiket DESC  LIMIT 1; |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Studio Elite

Studio Elite didefinisikan sebagai studio yang memiliki kemampuan 3D, dan memiliki jumlah kursi lebih dari 150. Tampilkan id, nama, dan jumlah kursi dari seluruh Studio Elite yang ada.

| Query | SELECT s.idstudio, s.namastudio, COUNT(k.idkursi) AS jumlah\_kursi  FROM \_studio s  JOIN \_kursi k ON s.idstudio = k.idstudio  JOIN \_studio\_has\_kemampuanfilm sk ON s.idstudio = sk.idstudio  WHERE sk.kemampuan = '3D'  GROUP BY s.idstudio, s.namastudio  HAVING COUNT(k.idkursi) > 150; |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Film Jadul Bagus

Tampilkan nama, tahun rilis, durasi, rating, dan jumlah penonton dari film yang dirilis sebelum tahun 1998, memiliki rating di atas 7, dan telah ditonton lebih dari 350 orang. Urutkan dari film yang memiliki rating paling tinggi. **Khusus untuk soal ini coba jalankan pula selain dengan perintah timing, eksekusi secara bersamaan menggunakan kakas query-executor.jar!**

| Query | SELECT f.namafilm, f.tahun, f.durasi, f.imdbrating, COUNT(idkursi) as jumlah\_penonton  FROM \_film f  JOIN \_reservasi r ON f.idfilm = r.idfilm  WHERE f.tahun < 1998 AND f.imdbrating > 7  GROUP BY f.namafilm, f.tahun, f.durasi, f.imdbrating  HAVING COUNT(idkursi) > 350  ORDER BY f.imdbrating DESC; |
| --- | --- |
| SS Query dan Hasil | |
| Timing    Jar | |

# 

# Studio Jauh

Tampilkan id film yang ditayangkan di studio 10 hingga studio 19 pada bulan Januari hingga September tahun 2015 dan memiliki rating lebih dari 6.

| Query | SELECT DISTINCT p.idfilm  FROM \_penayangan p  JOIN \_film f ON p.idfilm = f.idfilm  WHERE p.idstudio BETWEEN 10 AND 19 AND p.waktumulai >= '2015-01-01' AND p.waktumulai < '2015-10-01' AND f.imdbrating > 6; |
| --- | --- |
| SS Query dan Hasil | |
|  | |

# Komunitas Marcus

Buatlah sebuah view bernama “marcus\_movie” yang menampilkan judul film sebagai “Judul”, rating IMDB sebagai “Rating Film”, dan tahun rilis sebagai “Tahun” dari film-film yang pernah dilihat oleh pelanggan yang memiliki nama depan Marcus! Tunjukkan juga bahwa view berhasil dibuat!

| Query Jawaban | CREATE VIEW "marcus\_movie" AS  SELECT DISTINCT f.namafilm AS "Judul", f.imdbrating AS "Rating Film", f.tahun AS "Tahun"  FROM \_film f  JOIN \_reservasi r ON f.idfilm = r.idfilm  JOIN \_pelanggan p ON r.idpelanggan = p.idpelanggan  WHERE p.namapelanggan LIKE 'Marcus%'; |
| --- | --- |
| Query Testing | SELECT \* FROM marcus\_movie; |
| SS Query Sebelum Manipulasi Menggunakan Query Testing | |
|  | |
| SS Query Saat Manipulasi Menggunakan Query Jawaban | |
|  | |
| SS Query Setelah Manipulasi Menggunakan Query Testing | |
|  | |

# Populer di Genrenya

Satu film dapat memiliki lebih dari satu genre. Buatlah View dengan nama ‘above\_avg\_genre\_penayangan’ yang menampilkan genre sebagai ‘nama\_genre’ dan jumlah penayangannya sebagai ‘jumlah penayangan’ untuk genre film yang memiliki jumlah penayangan di atas rata-rata. Tampilkan hasil akhir viewnya!

| Query Jawaban | CREATE VIEW "above\_avg\_genre\_penayangan" AS  SELECT g.genre AS "nama\_genre", COUNT(p.idfilm) AS "jumlah penayangan"  FROM \_genre g  JOIN \_film\_has\_genre fg ON g.genre = fg.genre  JOIN \_penayangan p ON fg.idfilm = p.idfilm  GROUP BY g.genre  HAVING COUNT(p.idfilm) > (SELECT AVG(jumlah\_genre) FROM  (SELECT COUNT(p2.idfilm) as jumlah\_genre  FROM \_film\_has\_genre as fg NATURAL JOIN \_penayangan as p2  GROUP BY fg.genre )); |
| --- | --- |
| Query Testing | SELECT \* FROM above\_avg\_genre\_penayangan; |
| SS Query Sebelum Manipulasi Menggunakan Query Testing | |
|  | |
| SS Query Saat Manipulasi Menggunakan Query Jawaban | |
|  | |
| SS Query Setelah Manipulasi Menggunakan Query Testing | |
|  | |

# Tahun Baru Bersama Pacar

Veronica Doheny selalu membuat sebuah reservasi dengan pacarnya dengan menonton film yang tayang di tahun baru. Tetapi, mereka tidak akan menonton film yang memiliki rating dibawah 7. Namun, pada pada malam tahun baru, selalu saja terdapat bug sehingga transaksi yang ia lakukan tidak tercatat. Bantu pegawai untuk menambahkan pemesanan tiket pada tahun baru 2011. (Kerjakan hanya dengan 1 query saja, untuk letak kursi dibebaskan asalkan bersebelahan dengan asumsi seluruh kursi masih kosong)

| Query Jawaban | WITH film\_yang\_tayang AS (  SELECT p.idfilm, p.idstudio, p.waktumulai  FROM \_penayangan p  JOIN \_film f ON p.idfilm = f.idfilm  WHERE p.waktumulai::date = '2011-01-01'  AND f.imdbrating >= 7  LIMIT 1  ),  veronica\_id AS (  SELECT idpelanggan FROM \_pelanggan WHERE namapelanggan = 'Veronica Doheny'  ),  insert\_reservasi AS (  INSERT INTO \_reservasi (idfilm, idstudio, waktumulai, idkursi, idpelanggan)  SELECT fy.idfilm, fy.idstudio, fy.waktumulai, 'B6' AS idkursi, vi.idpelanggan  FROM film\_yang\_tayang fy, veronica\_id vi  RETURNING idfilm, idstudio, waktumulai, idpelanggan  )  INSERT INTO \_reservasi (idfilm, idstudio, waktumulai, idkursi, idpelanggan)  SELECT ir.idfilm, ir.idstudio, ir.waktumulai, 'B7' AS idkursi, ir.idpelanggan  FROM insert\_reservasi ir; |
| --- | --- |
| Query Testing | SELECT r.idfilm, r.idstudio, r.waktumulai, r.idkursi, p.namapelanggan  FROM \_reservasi r  JOIN \_pelanggan p ON r.idpelanggan = p.idpelanggan  WHERE p.namapelanggan = 'Veronica Doheny' AND r.waktumulai = '2011-01-01 00:00:00'; |
| SS Query Sebelum Manipulasi Menggunakan Query Testing | |
|  | |
| SS Query Saat Manipulasi Menggunakan Query Jawaban | |
|  | |
| SS Query Setelah Manipulasi Menggunakan Query Testing | |
|  | |

# *Rating Masking*

Pihak bioskop ingin memperbarui rating beberapa film berdasarkan jumlah penontonnya. Buatlah query untuk mengupdate rating film yang memiliki jumlah penonton diatas 500 dan rating lebih kecil dari 6.00 menjadi 6.00. Tunjukkan bahwa tidak ada lagi film yang memiliki penonton diatas 500 yang memiliki rating dibawah 6.00.

| Query Jawaban | UPDATE \_film  SET imdbrating = 6.00  WHERE imdbrating < 6.00  AND idfilm IN (  SELECT r.idfilm  FROM \_reservasi r  GROUP BY r.idfilm  HAVING COUNT(r.idkursi) > 500); |
| --- | --- |
| Query Testing | SELECT f.idfilm, f.imdbrating, COUNT(r.idkursi) AS jumlah\_penonton FROM \_film f  JOIN \_reservasi r ON r.idfilm = f.idfilm  WHERE f.imdbrating < 6.00  GROUP BY f.idfilm, f.imdbrating  HAVING COUNT(r.idkursi) > 500; |
| SS Query Sebelum Manipulasi Menggunakan Query Testing | |
|  | |
| SS Query Saat Manipulasi Menggunakan Query Jawaban | |
|  | |
| SS Query Setelah Manipulasi Menggunakan Query Testing | |
|  | |

# SPG Studio

Buatlah sebuah tabel dengan nama SPG yang berisi id karyawan, id studio, id film yang sedang ditayangkan. Perhatikan kolom yang harus direferensikan ke table lain.

| Query Jawaban | CREATE TABLE karyawan (  idkaryawan INT PRIMARY KEY,  namakaryawan VARCHAR(100) NOT NULL  );  ALTER TABLE \_studio  ADD CONSTRAINT unique\_studio UNIQUE (idstudio);  ALTER TABLE \_film  ADD CONSTRAINT unique\_film UNIQUE (idfilm);  CREATE TABLE SPG (  idkaryawan INT REFERENCES karyawan(idkaryawan),  idstudio INT REFERENCES \_studio(idstudio),  idfilm INT REFERENCES \_film(idfilm),  PRIMARY KEY (idkaryawan, idstudio, idfilm)  ); |
| --- | --- |
| Query Testing | \d SPG; |
| SS Query Sebelum Manipulasi Menggunakan Query Testing | |
|  | |
| SS Query Saat Manipulasi Menggunakan Query Jawaban | |
|  | |
| SS Query Setelah Manipulasi Menggunakan Query Testing | |
|  | |