Back End Development With Java Springboot Program

Sesi 10

Database

Introduction

Komponen utama penyusun suatu app yang terdiri dari sekumpulan data atau informasi yang tersimpan secara sistematis.

Ada banyak sekali jenis database, namun yang paling sering digunakan adalah Relational Database dan Non-Relational Database.

Relational Database

Pada relational database, data disimpan dalam sebuah skema yang ditampilkan seperti tabel (terdiri dari baris dan kolom).

Setiap data pada relational database diidentifikasi menggunakan key atau primary key.

Untuk memanipulasi data pada relational database digunakan sebuah bahasa yang disebut dengan Structured Query Language (SQL), oleh karena itu Relational Database juga disebut dengan SQL database.

Contoh Relational Database:

- MySQL
- MariaDB
- PostgreSQL
- SQL Server
- Oracle

Introduction

Non-relational Database

Non-relational database merupakan alternatif untuk relational database yang sering digunakan untuk menyimpan data dengan struktur yang tidak beraturan.

Non-relational database juga biasa disebut dengan NoSQL database.
Ada beberapa tipe non-relational database:

- Document Stores, data disimpan dalam bentuk JSON document. Contoh: MongoDB, Couchbase
- Key-value Stores, data disimpan dalam bentuk pasangan key-value. Contoh: Redis, Amazon DynamoDB
- Wide Column Stores, data disimpan dalam sebuah record yang bisa memiliki ribuan bahkan jutaan kolom. Contoh: Cassandra
- Graph Database, data disimpan dalam bentuk data struktur Graph. Contoh: Neo4j

SQL vs NoSQL

Bentuk Data

Pada SQL database seperti MySQL, sebuah data harus disimpan dalam bentuk tabel.
Data tersusun rapi dan mudah dibaca.

Sedangkan

pada NoSQL database, data dapat disimpan dalam bentuk document, key-value, wide column atau graph.
Data yang ditambahkan pada NoSQL database umumnya adalah data yang tidak terstruktur / beraturan.

SQL vs NoSQL

Read Operation

Operasi membaca data / Read pada SQL database bisa dibilang operasi yang cukup murah dan mudah (resource yang digunakan untuk operasi read tidak terlalu besar), hal ini dikarenakan sejak dari awal data harus di normalisasi terlebih dahulu sebelum ditambahkan ke database.

Yang dimaksud normalisasi adalah mengatur kolom pada tabel agar tidak terjadi data redundancy (duplikat data terjadi tanpa ada tujuan).

Operasi Read pada MySQL bisa menjadi mahal jika harus melakukan operasi join banyak table.

Sedangkan

operasi Read pada NoSQL database bisa menjadi mahal karena bentuk data yang ditambahkan bisa jadi tidak beraturan dari awal.

Untuk mendesain model data NoSQL database kita fokus pada query pattern dan bentuk data yang ingin didapatkan.

Hal ini berbeda dengan SQL database dimana fokus tertuju pada struktur dan normalisasi data.

Introduction

Meningkatkan skalabilitas dari SQL database lebih mudah dilakukan secara vertical, yang berarti peningkatan kemampuan dari SQL database dilakukan dengan cara meningkatkan spesifikasi mesin atau server.

Sedangkan

Skalabilitas dari NoSQL database lebih mudah dilakukan secara horizontal, yang dilakukan dengan cara menambah jumlah dari mesin atau server.

RDBMS - PostgreSQL

RDBMS

Relational Database Management System

Program yang digunakan untuk menampung basis data yang entitas utamanya terdiri dari **tabel-tabel yang mempunyai relasi dari satu tabel ke tabel yang lain**.

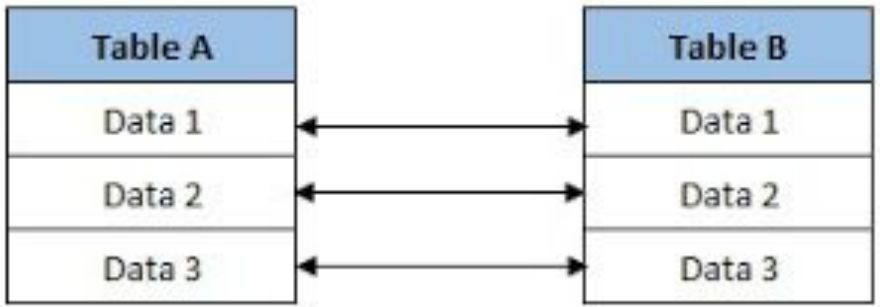
> **Suatu *database* terdiri dari banyak tabel. Tabel ini terdiri dari banyak *field* yang merupakan kolomnya. Isi tiap baris dari tabel inilah merupakan data.**

Untuk membuat sistem basis data yang terintegrasi maka antara satu tabel dengan tabel lain mempunyai hubungan yang harus selalu diperlihara. **Setiap tabel mempunyai sebuah *primary key*, *primary key* ini kemudian dihubungkan dengan tabel ke dua dan menjadi *foreign key* untuk tabel kedua.** Pembuatan *primary key* harus unik, artinya *primary key* baris pertama dengan *primary key* baris ke dua harus berbeda, dan begitu seterusnya.

Berbagai macam relasi dalam *database:*

- *one to one*
- *one to many*
- *many to many*

One to One



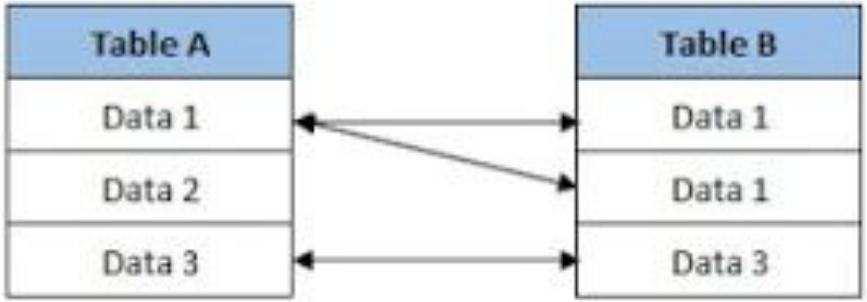
nim	nama	alamat	id_orangtua
1	A	JKT	1
2	B	JKT	2
3	C	JKT	3

Mahasiswa

id_orangtua	nama	umur
1	X	50
2	Y	60
3	Z	60

Orang-tua

One to Many



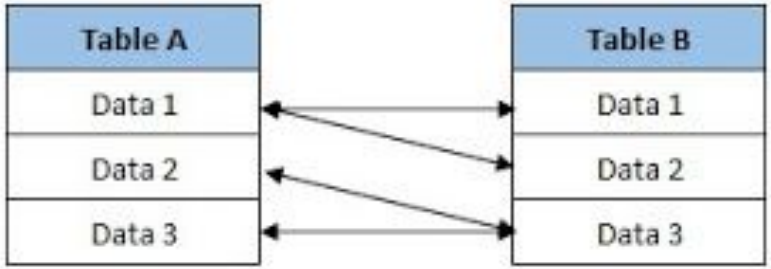
nim	nama	alamat	id_dosen
1	A	JKT	1
2	B	JKT	2
3	C	JKT	1

Mahasiswa

id_dosen	nama	umur
1	X	50
2	Y	60

Dosen

Many to Many



nim	nama	alamat	id_matkul
1	A	JKT	1
1	A	JKT	2
2	B	JKT	2
2	B	JKT	1

Mahasiswa

id_matkul	nama_matkul	sks	harga
1	basis data	3	450000
2	keamanan	3	450000

Matkul

RDBMS

Relational Database Management System

RDBMS akan menjaga agar data-data yang menjadi kunci relasi (*foreign key* dan *primary key*) berkaitan satu dengan yang lain. Jika ada data yang salah relasinya, maka RDMBS akan menolak data tersebut. Ini akan memudahkan pembuat program *(software developer)* dalam melakukan *coding* karena dibantu pengecekan secara otomatis oleh RDBMS.

Ada yang menarik, **jika menggunakan relasi *many to many* kita biasanya membuat satu buah tabel lagi** seperti contoh di bawah ini yaitu tabel **nilai** dari *database* "kampus", untuk menampung ***foreign key*** dari tabel-tabel yang memiliki hubungan *many to many* yaitu **tabel_mahasiswa** dan **tabel_matakuliah**.

Untuk relasi *many to many* kita membuat *database* baru yang bernama "kampus" yang terdiri dari **tabel_mahasiswa**, **tabel_matakuliah**, dan **nilai**.

Field (kolom) dari **tabel_mahasiswa** adalah `NIM`, `nama`, `alamat`, `tanggal_lahir` dengan ***primary key*** `NIM`.

Field (kolom) dari **tabel_matakuliah** adalah `kode_matakuliah`, `nama_matakuliah`, `sks` dengan ***primary key*** `kode_matakuliah`.

Untuk tabel **nilai** kita hanya perlu *field* `NIM`, `kode_matakuliah`, dan `nilai`. Di sini `NIM` dan `kode_matakuliah` merupakan ***foreign key*** untuk **tabel_mahasiswa** dan **tabel_matakuliah**.

PENGENALAN

sebuah *relational database manajemen system* (RDBMS) yang dikembangkan oleh tim relawan yang ada di seluruh dunia yang bersifat *open source*, artinya siapa saja bisa mengembangkannya. PostgreSQL tidak dikelola oleh perusahaan atau badan swasta lainnya, sehingga *source code* (kode program) yang tersedia bisa di dapatkan secara gratis.

Perbedaan yang paling mendasar antara postgres (sebutan untuk postgreSQL) dengan sistem relasional standar adalah, kemampuan postgres yang memungkinkan *user* untuk mendefinisikan SQL-nya sendiri, terutama untuk pembuatan *function*.

FITUR

Fitur-fitur dari PostgreSQL

Point-in-time recovery
mengizinkan server terus-menerus diback-up sehingga seandainya sebuah disk drive gagal bekerja, database dapat dikembalikan di titik dimana kegagalan itu terjadi.

Savepoints
berguna bagi database developer yang membutuhkan penanganan error dalam transaksi yang kompleks, yaitu suatu fitur yang mengizinkan suatu bagian tertentu dari transaksi database untuk dibatalkan tanpa mempengaruhi sisa transaksi yang lain.

Tablespaces
Berguna untuk memilih disk mana yang harus digunakan untuk menyimpan database, schema, table atau index. Sehingga kinerja PostgreSQL dalam menangani database raksasa berukuran ratusan gigabyte sampai puluhan terabyte dapat tetap terjaga.

Inheritance = Pewarisan
Mewariskan objek yang dimiliki ke pada objek yang diturunkan ,dan bersifat menyeluruh.
Kelas yang mewariskan biasa di sebut super class / class induk
Kelas yang diwariskan biasa di sebut sub class / kelas anak

Help
Digunkan untuk melakukan pencarian.Help pada fitur PostgreSQL memberikan hasil yang sangat akurat,selain itu fitur help-nya juga dilengkapi dengan berbagai screenshot yang sangat memudahkan.

Rule
Tindakan custom yang bisa kita definisikan dieksekusi saat sebuah tabel di-INSERT, UPDATE, atau DELETE.Selain itu sistem rule ini memungkinkan kita mengendalikan bagaimana data kita diubah atau diambil.

OO
Fitur OO seperti pewarisan tabel dan tipe data, atau tipe data array yang kadang praktis untuk menyimpan banyak item data di dalam satu record.Dengan adanya kemampuan OO ini maka di PostgreSQL, kita dapat mendefinisikan sebuah tabel yang mewarisi definisi tabel lain.

Installation

Selanjutnya kita butuh untuk mempersiapkan *database* postgreSQL yang akan kita gunakan.

Yang dibutuhkan:

- Chrome atau firefox
- Postgre sql
    - Masuk ke website https://www.postgresql.org
    - Klik *download*
    - Pilih OS
    - Pilih Postgres.app
    - Klik *download*, dan *install* versi yang paling stabil
      Postgres.app with PostgreSQL 9.5, 9.6, 10 and 11
    - Set *automation* di *preferences*
      "Security" -> "Privacy" -> "Automation" -> "Postgres"
    - Klik 2x salah satu *user*

Install masing-masing kebutuhan

INSTALLATION 2

Buka postgreSQL 11

Kita bisa menjalankan *query* lewat *console* atau terminal, seperti masuk ke *database* tertentu, melihat semua daftar *database* yang ada, membuat *database* baru, dll.

Jangan lupa nyalakan postgresnya terlebih dahulu dengan mengklik *start*. Jika sudah tampilannya menjadi seperti berikut:

Installation with Query Terminal

Jika masih tidak bisa, coba *update* postgress dengan sintaks:

```
brew upgrade postgres
```

Lalu coba jalankan sintaks di bawah:

*instruksi:

- Masuk sebagai postgres

```
psql postgres
```

- Melihat semua database yang ada

```
\l
```

- Membuat database baru

```
create database nama_db;
```

- Connect database

```
\c nama_db;
```

- Melihat semua table yang ada

```
\dt
```

- Menghapus database

```
drop database nama_db;
```

- Keluar dari console postgre

```
\q
```

PgAdmin

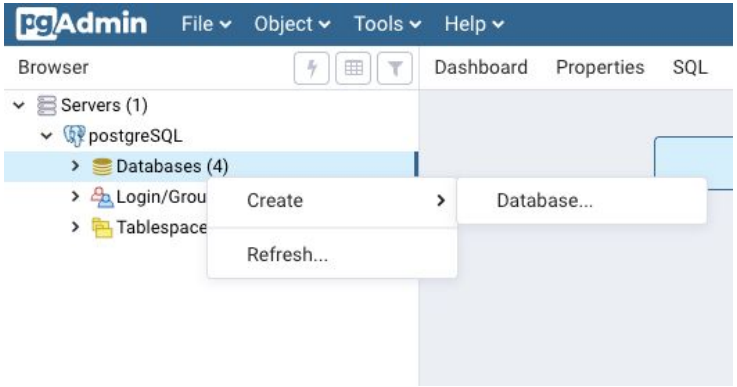
Selain menggunakan *console* kita juga bisa mengelola postgreSQL menggunakan *Grafical User Interface* (GUI). Salah satu GUI yang terkenal untuk postgreSQL adalah pgAdmin. pgAdmin adalah *platform opensource* yang digunakan untuk manajemen, pengembangan, serta manipulasi *database* posgreSQL.

***instruksi:**

- Kunjungi *website*-nya https://www.pgadmin.org/
- Klik install
- Pilih OS
- Setelah diinstal, *drag* ke desktop
- Klik *icon* 2x
- *Set password* (catat supaya tidak lupa)
- Buka *servers*
  Biasanya setiap kita buka server kita akan ditanyakan password yang sudah kita buat sebelumnya.
- Buat database baru melalui pgAdmin

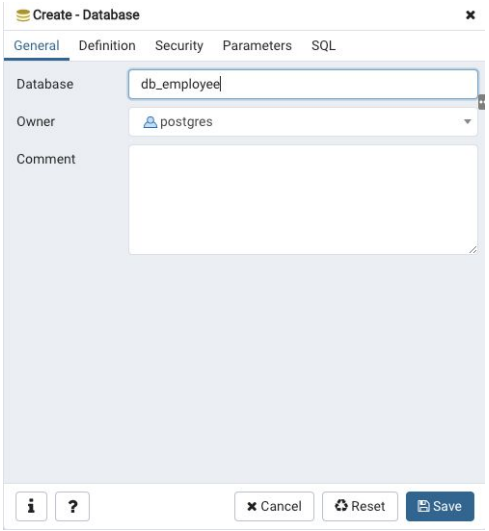
Create Database

1. Klik kanan di database
2. Pilih **Create > Database...**



3. Isikan nama *database*. Dalam gambar nama *database*nya adalah `db_employee`.

Nama *database* bebas, bisa disesuikan dengan keinginan teman-teman. Jika sudah diisikan namanya, selanjutnya tinggal kita save.



Integer

Kita akan membahas tentang tipe data yang akan sangat sering digunakan nanti yaitu *integer*. ***Integer* adalah tipe data yang hanya bisa dimasukan oleh angka bilangan bulat**. Jumlah maksimal *value* yang dapat ditampung oleh *integer* adalah 2147483647.

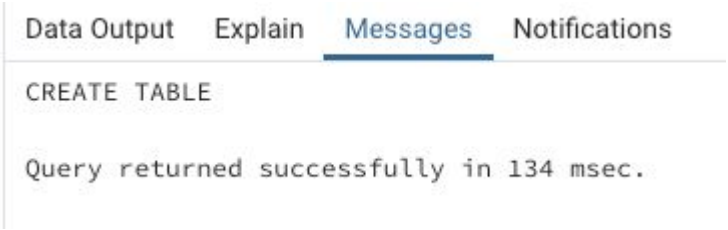
1. Tekan tombol petir dibagian kiri atas untuk memunculkan kolom query

2. Buat *table* baru menggunakan query dengan nama `tb_test` dengan kolom `testing`, tipe datanya *integer* di dalam *database* yang baru dibuat (`db_employee`) dan *execute*.
Query:

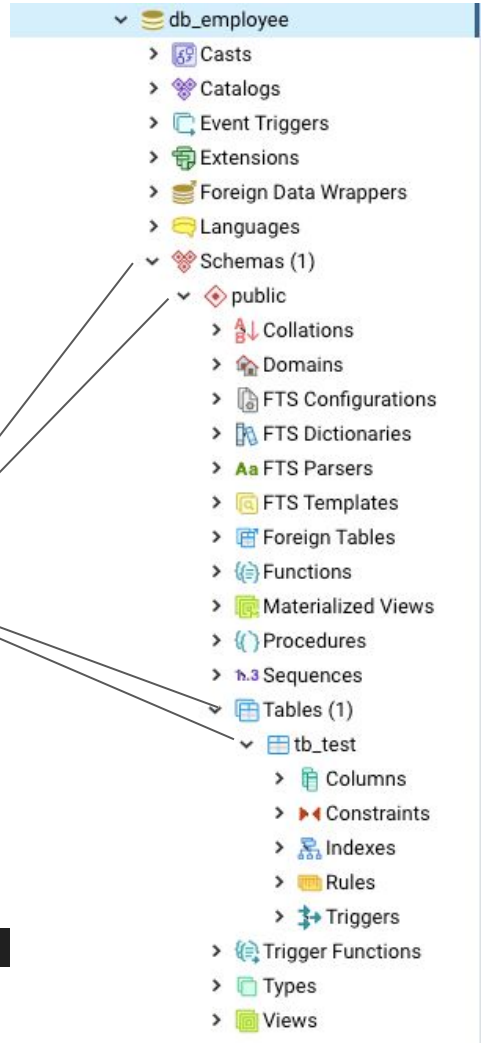
```
1 create table tb_test(
2 testing integer
3 );
```

3. Execute dengan mengklik tombol petir yang ada di bagian barisan atas kolom query:

Jika berhasil akan muncul tulisan seperti ini:



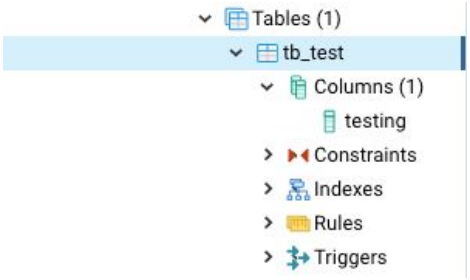
4. Lihat *table* yang barusan dibuat di
`db_employee` > `Schemas` > `Public` > `Tables` > `tb_test`



5. Coba masukkan *value* baru menggunakan query:

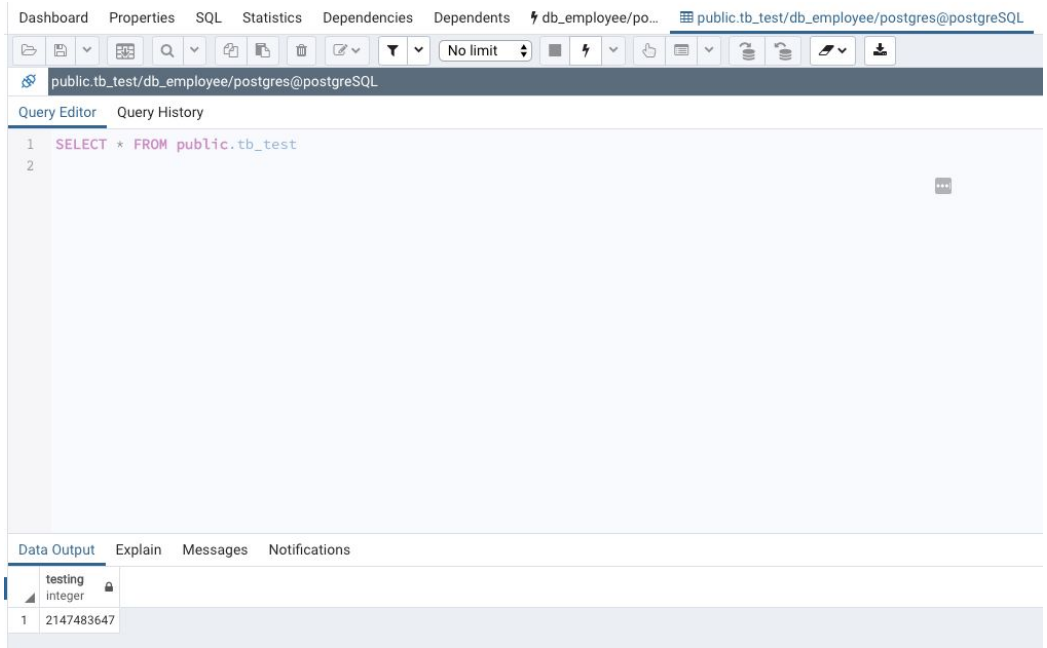
```
1 insert into tb_test values (2147483647);
```

5. Pilih tabel yang mau dilihat isinya, dalam hal ini tabel yang kita punya adalah `tb_test`



6. Lihat isi tabel, dengan klik "*button* tabel" disamping petir

Jika berhasil tampilannya akan seperti ini:



Character

Kita akan membahas tentang tipe data *character*. ***Character* adalah tipe data yang bisa dimasukan oleh huruf**. Jumlah maksimal *value* yang dapat ditampung oleh *character* adalah 255. Perlu diketahui juga pada saat membuat tipe data *character*, kita butuh mendefinisikan nilainya juga.

1. Tekan tombol petir dibagian kiri atas pojok untuk memunculkan kolom query seperti biasa

2. Buat tabel baru dengan nama `tb barang`, dengan kolom `nama barang` tipe datanya *character*, dengan panjang karakternya 30, lalu *execute* dengan tombol petir yang di atas kolom query

```
1 create table tb_barang(
2 nama_barang character(30)
3 );
```

- Refresh table, dan lihat tabel `tb_barang` sudah ada sekarang
- Masukkan value baru (es krim) menggunakan query dan execute dengan tombol petir yang di atas kolom query

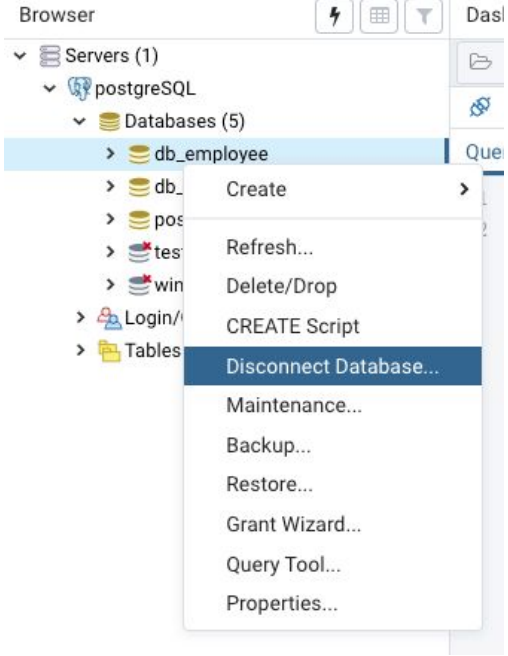
```
1 insert into tb_barang values ('es krim');
```

- Lihat isi baris barunya seperti biasa

CREATE & DELETE DATABASE

Kalau sebelumnya kita sudah membahas tentang bagaimana membuat *database* lewat pgAdmin. Sekarang kita akan coba gimana caranya membuat *database* dan menghapusnya menggunakan query di pgAdmin.

1. *Disconnect database* yang mau dihapus. Caranya dengan mengkilk kanan *database*, lalu Disconnect Database...



2. Tekan tombol petir dibagian kiri atas untuk memunculkan kolom query

3. Hapus *database* `db_employee` melalui query

```
1  drop database db_employee;
```

- Refresh database (klik kanan refresh)

Membuat database

- Membuat database `db_pegawai` melalui query

```
1  create database db_pegawai;
```

- Refresh database

CREATE & DELETE TABLE

Sebenernya kita sudah pernah membuat tabel. Sekarang kita akan coba gimana caranya membuat tabel baru dan menghapusnya menggunakan query di pgAdmin. Silahkan buat satu *database* baru dulu jika teman-teman belum mempunyai *database*.

- Membuat table company

```
CREATE TABLE COMPANY(
    ID INT PRIMARY KEY     NOT NULL,
    NAME           TEXT    NOT NULL,
    AGE            INT     NOT NULL,
    ADDRESS        CHAR(50),
    SALARY         REAL
);
```

- Refresh tables, dan lihat list tables bertambah, yaitu tabel company.

Menghapus table

- Hapus table company

```
drop table company;
```

- Refresh tables, dan lihat list tables berkurang, yaitu tabel company sudah tidak ada.

INSERT TABLE

Sebenernya kita sudah pernah membuat *insert*. Sekarang kita akan coba lagi bagaimana caranya mengisikan *table* dan memperbarui *table* yang sudah ada. Silahkan membuat satu tabel baru dengan nama *company*, jika teman-teman belum mempunyai tabel di *database*.

```
CREATE TABLE COMPANY(
    ID INT PRIMARY KEY     NOT NULL,
    NAME           TEXT    NOT NULL,
    AGE            INT     NOT NULL,
    ADDRESS        CHAR(50),
    SALARY         REAL
);
```

Insert table

- Kita akan memasukkan data baru ke dalam tabel company yang terdiri dari:

- `id` = 1
- `name` = fauzan
- `age` = 17
- `address` = jakarta
- `salary` = 2000

Buka kolom query dengan mengklik tombol petir, lalu execute query sebagai berikut:

```
insert into company values (1,'fauzan',17,'jakarta',2000);
```

Lihat isi tabel dengan mengklik tombol tabel disamping petir. Jika berhasil di-*insert*, *value*-nya akan bertambah seperti berikut:

Data Output | Explain | Messages | Notifications

	id [PK] integer	name text	age integer	address character (50)	salary real
1	1	fauzan	17	jakarta	2000

UPDATE TABLE

Update table company. Pada instruksi kali ini kita akan coba *update*
kolom `name` yang `id`-nya = 1, yaitu fauzan.
Kita akan coba ganti "fauzan" dengan "mantul".

```
1  update company set name = 'mantul' where id = 1;
```

- Lihat isi tabel dengan mengklik tombol tabel disamping petir. Jika berhasil di-update, `name` -nya akan berubah, seperti berikut:

Data Output | Explain | Messages | Notifications

	id [PK] integer	name text	age integer	address character (50)	salary real
1	1	mantul	17	jakarta	2000

Update banyak kolom

- Update lebih dari 1 kolom. Pada contoh sebelumnya kita hanya meng-update 1 kolom saja, yaitu kolom `name` . Selanjutnya, kita akan mencoba meng-update kolom `name` , `age` , dan `salary` yang memiliki `id` = 1 menggunakan query sebagai berikut:

```
1  update company set name = 'zaki', age = 25, salary = 5000 where id = 1;
```

- Lihat isi tabel dengan mengklik tombol tabel disamping petir. Jika berhasil di-update, `name` , `age` , dan `salary` akan berubah, seperti berikut:

Best Practice DDL & DML

Introduction of DDL

Data Definition Language(DDL)merupakan sintaks-sintaks yang berfungsi untuk melakukan manipulasi struktur dari basis data.

Secara umum, DDL digunakan untuk membuat tabel dan view.

Secara khusus dalam DBMS tertentu, DDL digunakan untuk membuat trigger, membuat stored procedure, juga membuat database, index, rule, schema, dan lainlain tergantung DBMS.

Introduction

Beberapa sintaks yang sering dijumpai dalam DDL.

CREATE DATABASE: membuat basis data.

CREATE TABLE : membuat tabel.

ALTER TABLE : merubah struktur suatu tabel.

DROP TABLE : menghapus suatu tabel.

CREATE INDEX : membuat suatu index dalam tabel.

DROP INDEX : menghapus suatu index dalam tabel

Introduction

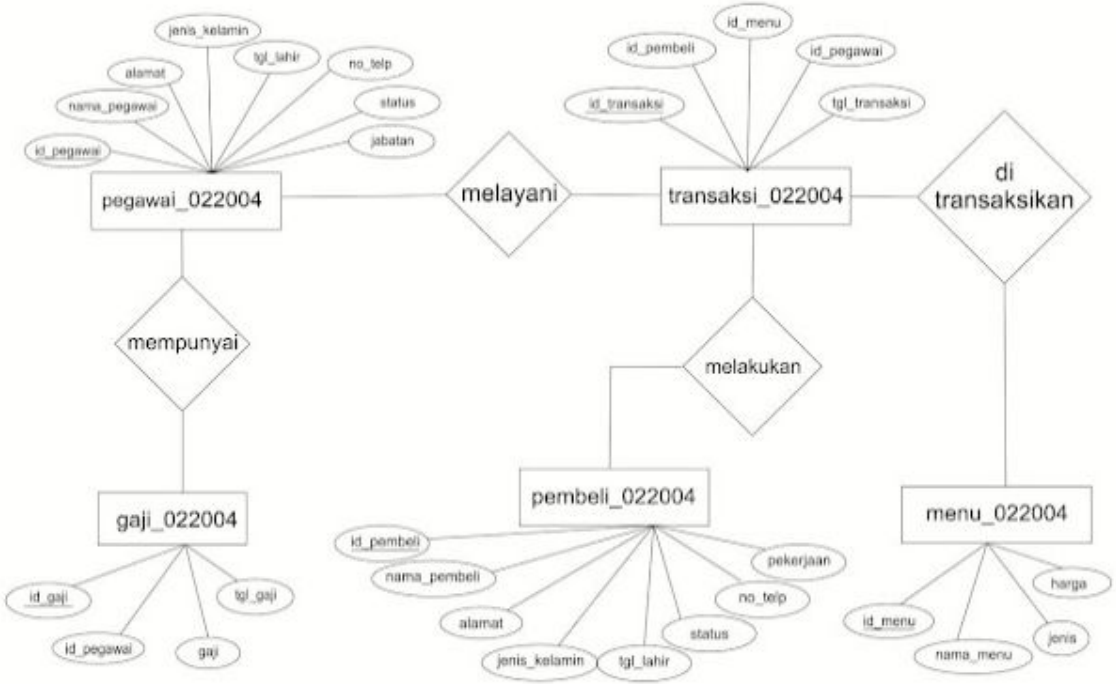
Data Manipulation Language (DML)merupakan sintaks-sintaks yang berfungsi untuk melakukan manipulasi data ataupun objek-objek yang ada di dalam tabel.

Berikut merupakan penjelasan singkat dari sintaks-sintaks DML.
SELECT, mengakses data dari suatu tabel dalam basis data.
UPDATE, melakukan update data dalam suatu tabel pada basis data.
DELETE, menghapus data dari suatu tabel dalam basis data.
INSERT, menambahkan data ke dalam suatu tabel dalam basis data.

Beberapa perintah telah kita pelajari pada pertemuan sebelumnya, selanjutnya silahkan kerjakan challenge terkait DDL dan DML Postgres pada slide selanjutnya

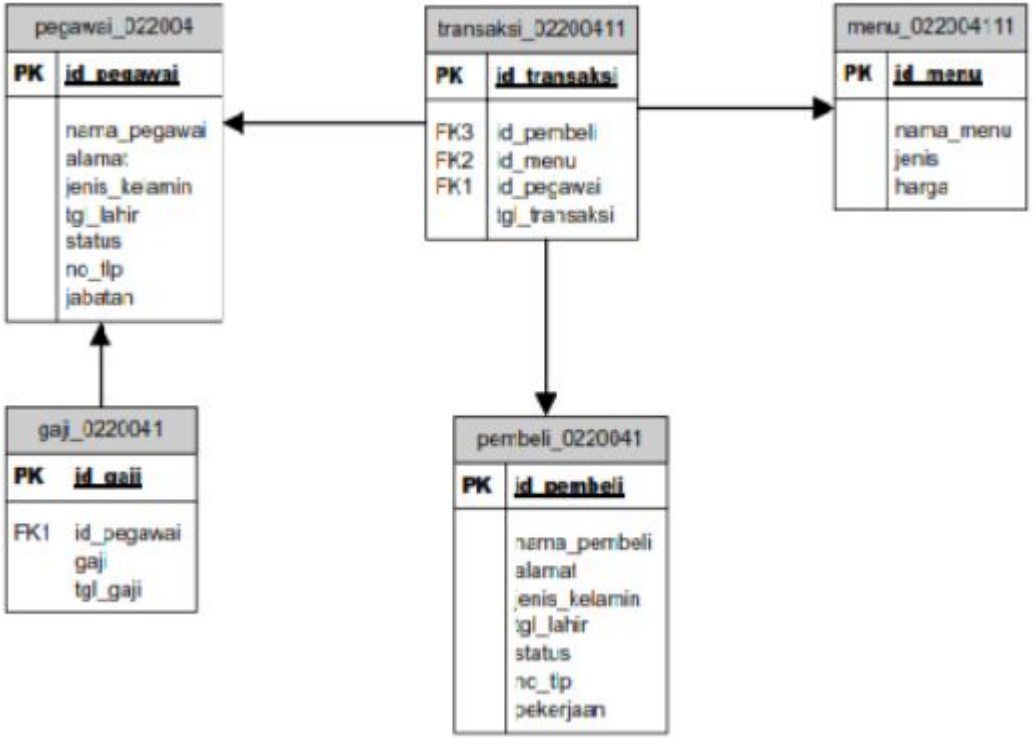
Challenge 1 : DDL & DML Postgre

Challenge 1 : DDL dan DML Postgre



HACKTIV8

Challenge 1 : DDL dan DML Postgre



Challenge 1

Dengan Penerapan DDL dan DML buatlah database dari ERD yang ada pada slide sebelumnya dengan mekanisme query sql yang sudah dipelajari baik DDL dan DML.

Expect : DML

Details :
a. Insert
b. Select
c. View
d. Delete

Lakukan secara Praktik dan tunjukkan kepada Instructor untuk menampilkan masing2 Perintah DML sesuai dengan expected hasil di slide selanjutnya.

Insert

A. Insert Tabel Pegawai

```
('1','yeni','salakan','perempuan','1992-10-14','lajang','5678094','cook asisten'),
('2','ikri','tamansiswa','laki-laki','1992-04-13','lajang','677908','supervisor'),
('3','martini','bengkulu','perempuan','1989-05-09','menikah','89345','operation manager'
('4','hartini','bengkulu','perempuan','1990-03-05','menikah','12367','manager'),
('5','yuni','bengkulu','laki-laki','1989-09-04','lajang','98076','general manager'),
('6','santoso','jogja','laki-laki','1989-09-08','lajang','123480','waiter'),
('7','piniarty','bengkulu','perempuan','1992-04-09','lajang','34467','cleaning servis'),
('8','astuti','jogja','perempuan','1991-03-06','lajang','9876','kasir'),
('9','tumini','bantul','perempuan','1989-03-02','lajang','945678','kasir'),
('10','yovi','salakan','Perempuan','1991-07-25','lajang','899981928','kepala koki');
```

B. Insert Tabel Menu

```
('111','javanes tofu','indonesian food','50000'),
('112','sirloin steack','western food','70000'),
('113','tanderloin steack','western food','90000'),
('114','fish ball sup','western food','75000'),
('115','mie rebus','indonesian food','15000'),
('116','macaroni chicken sup','western food','80000')
('117','fruit punch','beverage','15000'),
('118','mocacino ice','beverage','15000'),
('119','strwbery juice','beverage','10000'),
('120','milkshake','beverage','12000');
```

Insert

C. Insert Tabel Pembeli

```
('211','wiwit','bengkulu','perempuan','1989-03-01','belum menikah','567890','apoteker'),
('212','ayu','malang','perempuan','1988-06-03','menikah','5690235','dosen'),
('213','nengsih','solo','perempuan','1980-07-09','menikah','765489','asisten'),
('214','albert','australia','perempuan','1989-09-08','belum menikah','346789','analis'),
('215','salmi','bengkulu','perempuan','1989-09-07','menikah','907654','perawat'),
('216','rahmat','lombok','laki-laki','1988-08-06','menikah','987654','kontraktor'),
('217','budiana','bali','perempuan','1987-03-01','menikah','12468','analis'),
('218','wati','bengkulu','perempuan','1980-07-06','menikah','987766','programer'),
('219','rayan nazriel','banglades','laki-laki','1989-09-10','belum menikah','6790007','perawat'),
('220','alexs','papua','laki-laki','1990-03-04','belum menikah','9988664','dokter');
```

D. Insert Tabel Gaji

```
('2001','1','2000000','2012-12-01'),
('2002','2','1800000','2012-12-01'),
('2003','3','1500000','2012-12-01'),
('2004','4','3500000','2012-12-01'),
('2005','5','4000000','2012-12-01'),
('2006','6','2500000','2012-12-01'),
('2007','7','1500000','2012-12-01'),
('2008','8','1000000','2012-12-01'),
('2009','9','3500000','2012-12-01'),
('2010','10','3500000','2012-12-01');
```

Insert

E. Insert Tabel Transaksi

```
('1001','211','111','9','2012-09-09'),
('1002','211','112','9','2012-11-08'),
('1003','212','113','10','2012-11-10'),
('1004','213','114','10','2012-11-28'),
('1005','213','115','9','2012-11-28'),
('1006','214','116','9','2012-11-29'),
('1007','215','117','10','2012-12-08'),
('1008','216','118','9','2012-12-08'),
('1009','217','119','9','2012-12-18'),
('1010','218','119','9','2012-12-18'),
('1011','219','120','10','2012-12-28'),
('1012','220','120','10','2012-12-29');
```

Select

F. Select Tabel Pegawai

id_pegawai	nama_pegawai	alamat	jenis_kelamin	tgl_lahir	status	no_telp	jabatan
1	yeni	salakan	perempuan	1992-10-14	lajang	5678094	cook asisten
2	ikri	taman siswa	laki-laki	1992-04-13	lajang	677908	supervisor
3	martini	bengkulu	perempuan	1989-05-09	menikah	89345	operation manager
4	hartini	bengkulu	perempuan	1990-03-05	menikah	12367	manager
5	yuni	bengkulu	laki-laki	1989-09-04	lajang	98076	general manager
6	santoso	jogja	laki-laki	1989-09-08	lajang	123480	waiter
7	piniarty	bengkulu	perempuan	1992-04-09	lajang	34467	cleaning servis
8	astuti	jogja	perempuan	1991-03-06	lajang	9876	kasir
9	yovi	salakan	Perempuan	1991-07-25	lajang	899981928	kepala koki
10	tumini	bantul	perempuan	1989-03-02	lajang	945678	kasir

(10 rows)

Select

G. Select Tabel Menu

id_menu	nama_menu	jenis	harga
111	javanes tofu	indonesian food	50000
112	sirloin steack	western food	70000
113	tanderloin steack	western food	90000
114	fish ball sup	western food	75000
115	mie rebus	indonesian food	15000
116	macaroni chicken sup	western food	80000
117	fruit punch	beverage	15000
118	mocacino ice	beverage	15000
119	strwbery juice	beverage	10000
120	milkshake	beverage	12000

(10 rows)

Select

H. Select Tabel Pembeli

id_pembeli	nama_pembeli	alamat	jenis_kelamin	tgl_lahir	status	no_telp	pekerjaan
211	wiwit	bengkulu	perempuan	1989-03-01	belum menikah	567890	apoteker
212	ayu	malang	perempuan	1988-06-03	menikah	5690235	dosen
213	nengsih	solo	perempuan	1980-07-09	menikah	765489	asisten
214	albert	australia	perempuan	1989-09-08	belum menikah	346789	analis
215	salmi	bengkulu	perempuan	1989-09-07	menikah	907654	perawat
216	rahmat	lombok	laki-laki	1988-08-06	menikah	987654	kontrakto
217	budiana	bali	perempuan	1987-03-01	menikah	12468	analis
218	wati	bengkulu	perempuan	1980-07-06	menikah	987766	programer
219	rayan nazriel	banglades	laki-laki	1989-09-10	belum menikah	6790007	perawat
220	alexs	papua	laki-laki	1990-03-04	belum menikah	9988664	dokter

(10 rows)

Select

I. Select Tabel Gaji

id_gaji	id_pegawai	gaji	tgl_gaji
2001	1	2000000	2012-12-01
2002	2	1800000	2012-12-01
2003	3	1500000	2012-12-01
2004	4	3500000	2012-12-01
2005	5	4000000	2012-12-01
2006	6	2500000	2012-12-01
2007	7	1500000	2012-12-01
2008	8	1000000	2012-12-01
2009	9	3500000	2012-12-01
2010	10	3500000	2012-12-01

(10 rows)

Select

J. Select Tabel Transaksi

id_transaksi	id_pembeli	id_menu	id_pegawai	tgl_transaksi
1001	211	111	9	2012-09-09
1002	211	112	9	2012-11-08
1003	212	113	10	2012-11-10
1004	213	114	10	2012-11-28
1005	213	115	9	2012-11-28
1006	214	116	9	2012-11-29
1007	215	117	10	2012-12-08
1008	216	118	9	2012-12-08
1009	217	119	9	2012-12-18
1010	218	119	9	2012-12-18
1011	219	120	10	2012-12-28
1012	220	120	10	2012-12-29

(12 rows)

View

K. View : Nama Pegawai, Jabatan, Gaji

id_pegawai	nama_pegawai	jabatan	gaji
2	ikri	supervisor	1500000
3	martini	operation manager	2500000
4	hartini	manager	3000000
5	yuni	general manager	2500000
6	santoso	waiter	1500000
7	piniarty	cleaning servis	1000000
8	astuti	kasir	2000000
9	yovi	kepala koki	2000000
10	tumini	kasir	2500000
1	astuti	cook asisten	2000000

(10 rows)

View

L. Delete : Transaksi

id_transaksi	id_pembeli	id_menu	id_pegawai	tgl_transaksi
1001	211	111	9	2012-09-08
1002	211	112	9	2012-09-08
1003	212	113	10	2012-09-08
1004	213	115	9	2012-10-08
1005	213	116	9	2012-10-08
1006	214	117	10	2012-11-08
1007	215	118	10	2012-11-09
1008	216	118	10	2012-11-10
1009	217	119	9	2012-12-01
1010	218	119	9	2012-12-10
1011	219	120	10	2012-12-11

(11 rows)