



# Database

**Gold** - Chapter 5 - Topic 3

---

**Selamat datang di Chapter 5 Topik 3 online  
course Fullstack Web dari Binar Academy!**





**Yes, kita udah berhasil setengah jalan melewati chapter ini!**

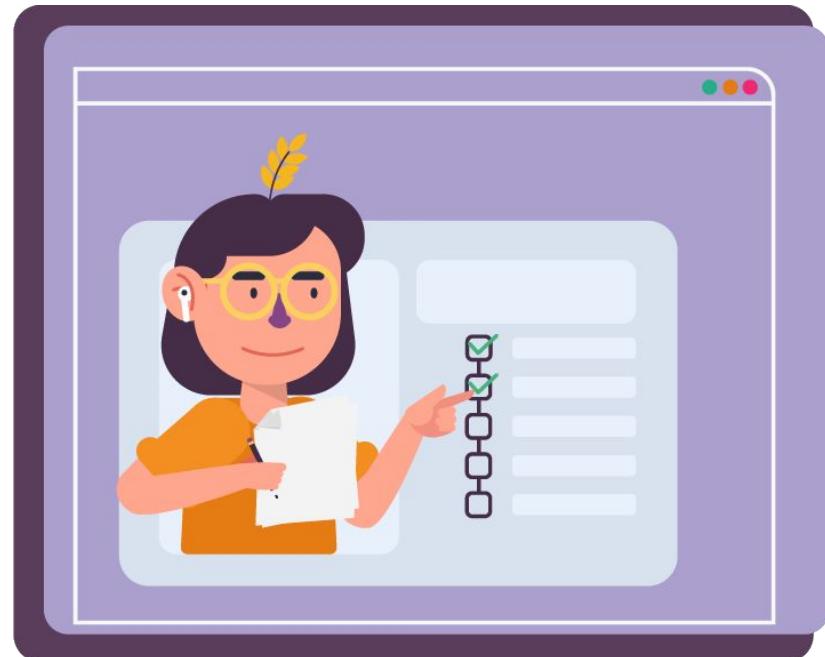
Di topik sebelumnya kita bahas tentang RestfulAPI dan bagaimana implementasinya pada Express. Selanjutnya, di topik ketiga ini kita akan bahas mengenai **database**. Markicuss~





### Detailnya, kita bakal bahas hal-hal berikut:

- Memahami apa itu database
- Mengenal jenis-jenis DBMS
- Mengenal SQL dan NoSQL
- Cara melakukan query pada konsol DBMS
- Mengenal ERD





Kalian pasti sering banget kan mendengar kata “data”. Tapi, kalau basis data atau yang sering disebut dengan **database** itu apa ya?

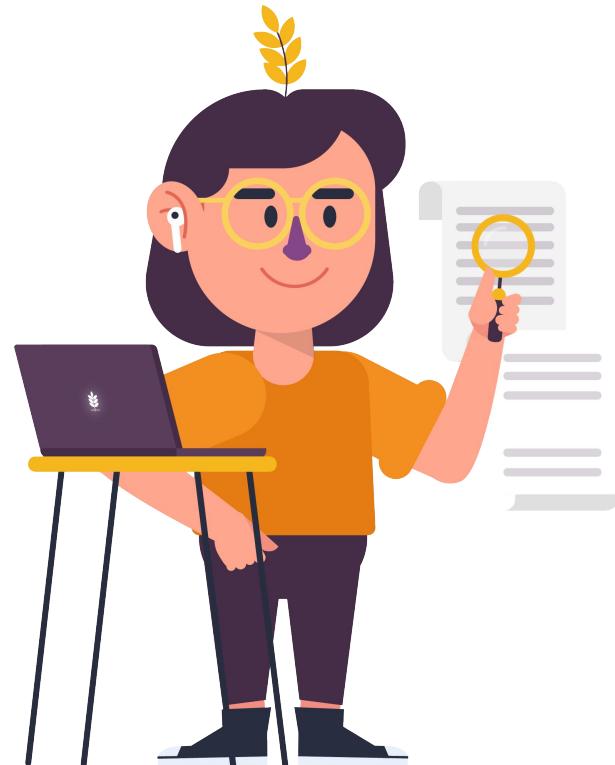


# Database

Di chapter-chapter sebelumnya kita udah mengenal berbagai tipe data dan struktur data. Nah, sekarang kita mau belajar tentang basis data nih. Apakah kedua materi tersebut berhubungan?

Yap! Basis data atau database ini meliputi spesifikasi berupa tipe data, struktur data, juga batasan-batasan yang ada pada data dan kemudian disimpan.

Jadi, database dapat diartikan sebagai **kumpulan data yang disimpan secara sistematis di dalam komputer. Data tersebut dapat diolah atau dimanipulasi dengan perangkat lunak (software)/program/aplikasi untuk menghasilkan informasi.**





Kalian pasti pernah belanja online kan?

Nah, buat memahami konsep database, kita bisa bayangan ketika mau belanja online. Barang-barang yang disajikan di halaman lapak penjual merupakan sebuah data yang disimpan dan nantinya bisa dipanggil lagi untuk ditampilkan, sehingga pembeli bisa lihat foto barang, deskripsi, dan lain sebagainya.

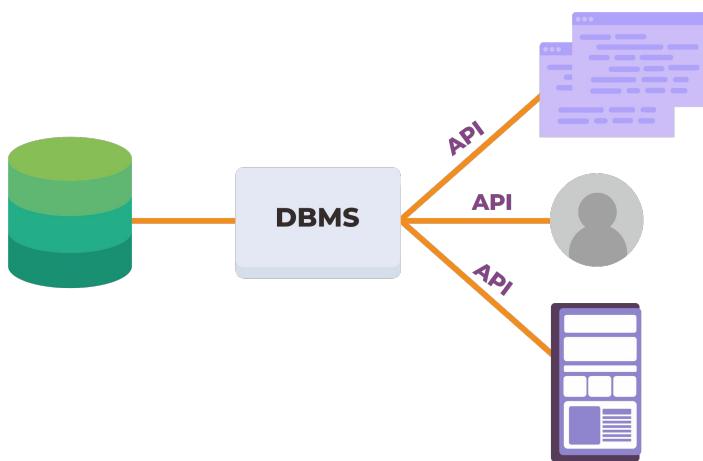
Coba bayangan kalau lapak pedagang online nggak pakai konsep database?! Sudah pasti pengolahan datanya jadi ribet banget. Ada satu barang, kita cari dulu fotonya, cari lagi deskripsinya, cari ini, cari itu. Meh... bakalan jadi lama dan akhirnya gak efisien deh.



Nah, dengan konsep database, hal-hal yang gak efisien seperti itu bisa dikurangi. Karena database bisa dianalogikan seperti sebuah rak buku.

Buku-buku itu bisa diibaratkan seperti data-data yang kita simpan. Jadi, kalau kita perlu beberapa data, maka kita nggak perlu repot lagi cari kesana-kemari. Tinggal cari rak bukunya, lalu cari mau buku yang mana, beres deh!





Kamu tahu nggak sih kenapa database itu aspek yang sangat penting dalam sistem informasi?

Faktor utamanya karena **database berfungsi sebagai gudang penyimpanan data untuk diolah lebih lanjut**. Kita dapat mengorganisasi data, menghindari duplikasi data, menghindari hubungan antar data yang tidak jelas dan update data yang rumit.

Nah, proses untuk memasukkan dan mengambil data yang berhubungan dengan media penyimpanan data, tentunya memerlukan suatu alat atau perangkat lunak, yang biasanya disebut dengan sistem manajemen basis data (Database Management System/DBMS). **DBMS memungkinkan kita untuk memelihara, mengontrol, juga mengakses data secara praktis dan efisien**.



Dari penjelasan sebelumnya, kita tahu bahwa kita membutuhkan DBMS untuk memproses database.

Oleh karena itu, kita perlu mengetahui dulu beberapa **tipe database**, yaa!



# Tipe database

## Relational Database

Semua data harus **mengikuti struktur yang sama**.

Ibaratnya, jika di wilayah A, semua orang bicara dengan bahasa yang sama, dan bahasa itu menjadi satu-satunya cara agar warga bisa “interaksi” dengan dunia luar. Ketika ada satu rumah yang mengganti bahasa itu maka akan bikin pusing dan mengganggu semua orang.

Nah, ini berarti jika ada perubahan struktur pada database kita, maka akan sulit dan mengganggu keseluruhan sistem. **Bahasa khusus yang digunakan untuk membuat dan mengolah relational database ini adalah SQL (Structured Query Language)**.





## Non-Relational Database

**Skema data kita dinamis, yang berarti datanya tidak terstruktur.**

Ibaratnya, jika di suatu wilayah B, semua orang bisa “interaksi” ke dunia luar dengan bahasa dan cara yang berbeda. Ketika ada satu rumah memakai bahasa yang berbeda, gak akan berefek dengan orang lain.

Nah, fleksibilitas ini akan memudahkan kita jika ingin membuat dokumen tanpa harus mendefinisikan strukturnya. Mekanisme untuk menyimpan dan mengambil data yang **non-relational database ini dengan menggunakan NoSQL (Not Only SQL atau Not SQL).**

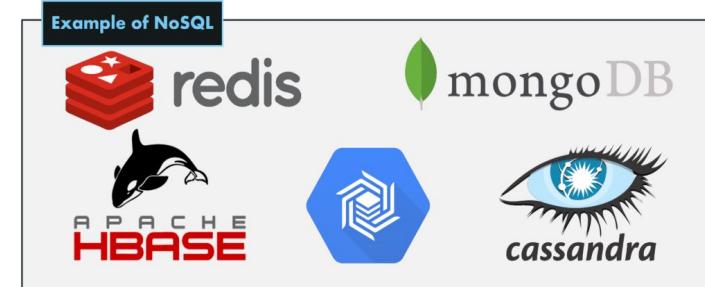




Di slide sebelumnya, kamu sudah paham kan tentang database? Selanjutnya, kita akan mengenal beberapa **DBMS** yang dapat kita gunakan untuk kedua tipe database itu. Cussss~



Ada banyak jenis DBMS yang dapat kita gunakan untuk memanipulasi *database*, baik itu yang berbasis SQL maupun NoSQL. Berikut ini beberapa contoh DBMS yang sering digunakan :

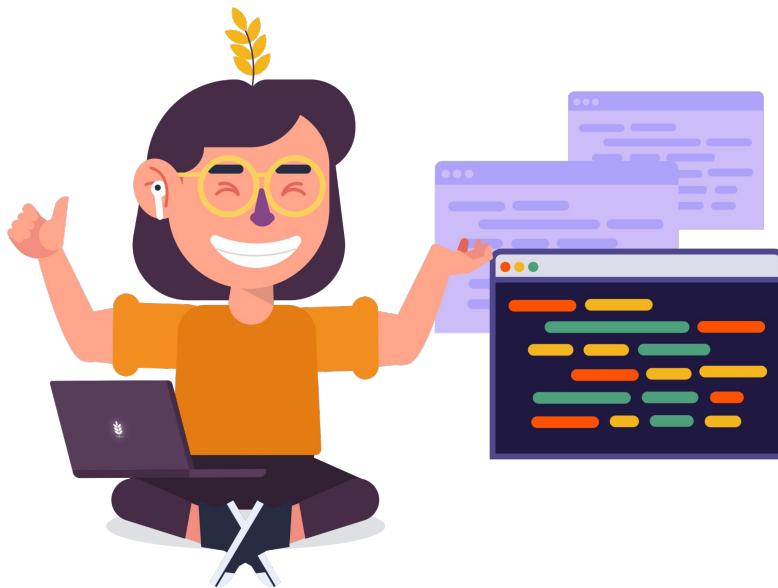


Berikut contoh DBMS yang berbasis **Structured Query Language**:

- MySQL
- PostgreSQL
- Microsoft SQL Server
- SQLite
- Oracle RDBMS
- MariaDB

Berikut contoh DBMS yang berbasis **NoSQL**:

- MongoDB
- Redis
- Apache Cassandra
- Apache HBase
- Bigtable



## SQL

DBMS yang berbasis SQL, secara sederhana menggunakan SQL sebagai bahasanya. Jadi, ketika kita ingin memanipulasi database di dalam DBMS tersebut, maka kita **harus pakai bahasa SQL**.

SQL adalah **akronim dari Structured Query Language, suatu bahasa yang kita gunakan untuk melakukan query**.

Nah, kalau Query adalah suatu pernyataan (statement) yang kita kirimkan ke dalam suatu DBMS untuk mencari/membuat/memanipulasi data yang ada di database. Ini berarti query di dalam SQL sangat terstruktur. Karena itulah akan ada banyak aturan yang harus kita patuhi.



Berikut ini adalah contoh query dengan menggunakan bahasa SQL:

```
SELECT * FROM articles;
```

Maksudnya, kita ingin menampilkan semua data yang ada di dalam **table** yang bernama articles.

**Apa itu table?** Di dalam SQL, kita menyimpan data dalam bentuk table, yang dalamnya terdapat row dan column. Untuk setiap datanya kita sebut dengan **row**, sedangkan informasi-informasi yang terdapat pada suatu data kita sebut dengan **column**.





Berikut ini contoh suatu table bernama **articles**, yang mana di dalam table tersebut memiliki beberapa column, yaitu **id**, **title**, dan **body**. Dari table ini, kita bisa tahu bahwa "Hello World" adalah judul dari table articles di baris pertama.

Articles

| <b>id</b> | <b>title</b>      | <b>body</b>  |
|-----------|-------------------|--|
| 1         | "Hello World"     | "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum."   |
| 2         | "Bye World"       | "Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32" |
| 3         | "Edit Your World" | "There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet. It uses a dictionary of over 200 Latin words, combined with a handful of model sentence structures, to generate Lorem Ipsum which looks reasonable. The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-characteristic words etc."             |

Kalau kita ingin menambahkan artikel baru, kita harus mengikuti kaidah table tersebut ya! Maksudnya, kita hanya boleh memasukkan data sesuai column-nya aja, yaitu **id**, **title**, dan **body**.



Perlu kita ingat juga kalau di setiap column pasti sudah ditentukan tipe datanya seperti apa. Contohnya, column title di table articles itu udah ditentukan kalau tipe datanya berupa string. Nah, kalau kita mau input data, maka tipenya harus string. Kalau tipe datanya gak sesuai bakalan error.

Terus gimana ya kalau kita mau menambah data baru, misalnya ada author di table articles?

Tenang! Kita bisa lakukan itu kok. Tapi, kita harus modifikasi table-nya dulu. Kita akan bahas gimana caranya di topik selanjutnya ya!



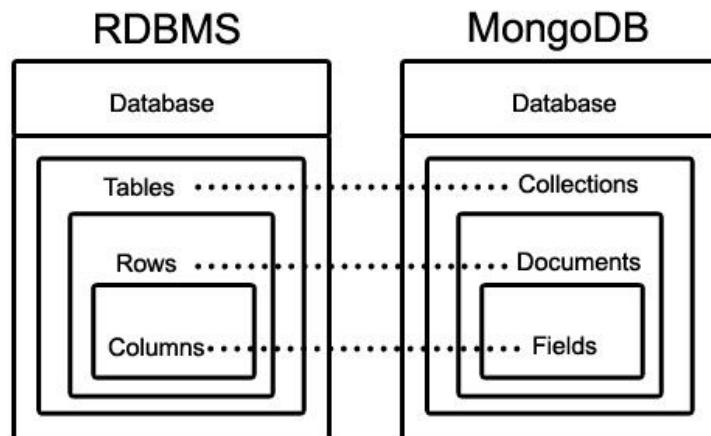


## NoSQL

NoSQL (Not Only SQL atau Not SQL) adalah **sebuah DBMS yang tidak menggunakan bahasa SQL dalam melaksanakan query**. Ini berarti setiap database NoSQL memiliki bahasanya sendiri. Misalnya, redis dan mongodb, kedua DBMS tersebut memiliki bahasa yang berbeda.

NoSQL juga bisa memiliki struktur dan kaidah yang berbeda dari DBMS NoSQL yang lainnya. Tapi, untuk topik ini kita akan bahas cara pakai **MongoDB** aja karena sangat populer digunakan.

| TYPE              | EXAMPLE   |
|-------------------|---|
| Key-Value Store   |  redis  riak  |
| Wide Column Store |  apache hbase  cassandra                                |
| Document Store    |  mongoDB  CouchDB relax                                 |
| Graph Store       |  neo4j  InfiniteGraph<br>The Distributed Graph Database |



**MongoDB menyimpan datanya dalam bentuk documents (kalau di SQL istilahnya rows). Dokumen tersebut berbentuk objek seperti JSON**, artinya nggak ada kaidah baku dalam pembuatan dokumen dan setiap dokumen bisa memiliki bentuk data yang berbeda-beda.

Di dalam dokumen itu terdapat beberapa **field** (kalau di SQL istilahnya columns). Fields tersebut formatnya berpasangan, berupa key dan value. Dokumen akan disimpan di dalam sebuah **collections** (kalau di SQL istilahnya tables).



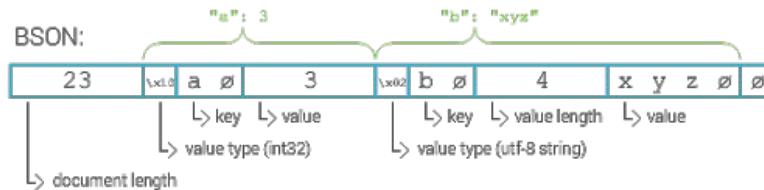
Biar lebih jelas, yuk kita lihat contoh implementasi NoSQL di samping ini! Berdasarkan table SQL yang sebelumnya, kita ubah ke dalam format MongoDB. Dalam contoh ini, kita ingin menyimpan artikel yang memiliki title dan body.

```
[  
 {  
     "_id": ObjectId("5ed86aa6a319c6b0a4b1c29e"),  
     "title": "Hello World",  
     "body": "Lorem Ipsum is simply dummy text of the  
printing and typesetting industry."  
 },  
 {  
     "_id": ObjectId("5ed86b03a319c6b0a4b1c29f"),  
     "title": "Bye World",  
     "body": "Contrary to popular belief, Lorem Ipsum is  
not simply random text."  
 },  
 {  
     "_id": ObjectId("5ed86b03a319c6b0a4b1c29f"),  
     "title": "Edit Your World",  
     "body": "There are many variations of passages of  
Lorem Ipsum available, but the majority have suffered  
alteration."  
 }  
 ]
```

JSON:

```
{  
  "a": 3,  
  "b": "xyz"  
}
```

BSON:



ObjectId('5b7d297cc718bc133212aa94')

5b7d297c  
UNIX Timestamp  
4 Bytes

c718bc1332  
Random Value  
5 Bytes

12aa94  
Count  
3 bytes

Dari contoh di slide sebelumnya, kita bisa lihat kalau format MongoDB mirip format JSON ya?

Yap, memang mirip! Tapi, di MongoDB nggak pakai format JSON dalam menyimpan data. **MongoDB menggunakan format Binary Structured Object Notation (BSON)**. Simpelnya, BSON itu JSON yang bisa kita isi beberapa syntax JavaScript, misalnya function.

Kita bisa lihat contoh sebelumnya terdapat function **ObjectId()**, function ini cuma bisa kita tulis dengan format BSON aja. Kalau kalian masih kepo tentang BSON bisa buka link [ini](#).



Oh iya, di dalam MongoDB, setiap dokumen juga bisa memiliki bentuk yang berbeda dari dokumen lain yang berada di dalam koleksi yang sama.

Misalnya, kita ingin menambahkan artikel baru yang memiliki field approved dan author, maka kita dapat langsung menambahkannya tanpa harus mengubah struktur table terlebih dulu. Contohnya seperti di samping ini.

```
[...dataTadi,
{
  "_id": ObjectId("5ed86b44a319c6b0a4b1c2a2"),
  "title": "Oof World",
  "body": "The standard chunk of Lorem Ipsum used
since the 1500s is reproduced below for those
interested. Sections 1.10.32 and 1.10.33 from \"de
Finibus Bonorum et Malorum\" by Cicero.",
  "approved": true,
  "author": "Fikri Rahmat Nurhidayat"
}]
```



### SQL Vs NoSQL

Untuk menyederhanakan konsep, perbandingan antara SQL dan MongoDB dapat didefinisikan di dalam tabel di samping

| SQL        | NoSQL             |
|------------|-------------------|
| Database   | Database          |
| Table      | Collection        |
| Column     | Field             |
| Row        | Document          |
| Join Table | Embedded Document |



Oke, kita kan udah tahu konsep DBMS dan contoh DBMS yang bisa digunakan untuk memanipulasi database.

Nah, kita juga perlu tahu dong gimana caranya **implementasi DBMS**, baik yang berbasis SQL (PostgreSQL) dan NoSQL (MongoDB).

Yes, kita udah tau tentang SQL dan DBMS. Sekarang kita akan coba menggunakannya ya! Tapi, pertama-tama kita harus **install PostgreSQL**-nya dulu nih. Untuk petunjuk instalasinya bisa dilihat pada link di samping, disesuaikan aja dengan sistem operasi pada perangkat yang kamu gunakan ya.

Kalau kamu mau download, langsung aja klik link di bawah ini:

- [Linux](#)
- [Mac](#)
- [Windows](#)





Nah, setelah selesai dengan proses instalasi, maka kita akan mendapatkan aplikasi baru di dalam terminal kita, yaitu psql. **Psql ini adalah konsol yang kita gunakan untuk melakukan query ke dalam suatu database.** Untuk masuk ke konsol tersebut, kita hanya perlu membuka terminal dan ketik:



```
sudo -i -u postgres psql
```

Selanjutnya kita akan ditanya *password system* kita, dan kalau kita isi dengan benar maka hasilnya akan seperti ini:

```
~> sudo -i -u postgres psql
[sudo] password for fnurhidayat:
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
Type "help" for help.

postgres=#
```



Setelah kita masuk ke dalam konsol psql, maka kita bisa menjalankan beberapa query. Tapi, kita gak akan membahas secara detail penggunaan query di topik ini. Kita hanya perlu melakukan beberapa **basic setup untuk postgresql**, yaitu:

- Create user
- Login ke database
- Membuat database baru, baik melalui konsol maupun dari terminal





## Create user

Hal krusial yang harus kita lakukan pertama kali ketika kita menginstal postgres adalah **membuat user yang akan kita pakai sebagai admin** di database kita. Untuk nama user sangat direkomendasikan bila menggunakan nama yang sama ketika menjalankan perintah “whoami” di terminal, maksudnya nama user di sistem kita. Dengan menjalankan query tersebut, kita akan membuat user baru bernama “fnurhidayat”, dan user tersebut memiliki password “123456”.



```
CREATE USER fnurhidayat WITH PASSWORD '123456';
```



Di sini kita juga harus memberi otoritas kepada user tersebut agar dapat membuat database dan mengakses database. Kita bisa memberi role **SUPERUSER**, tapi ini gak disarankan kalau server beneran ya! Tapi, kalau kita belajar ini akan memudahkan kita nantinya dalam me-manage database di PC kita.

Setelah itu, kita bisa exit dengan mengetikkan **\q** atau menggunakan shortcut tombol keyboard **CTRL+D**. Kemudian kita dapat mengetikkan **exit** maka kita akan benar-benar logout atau keluar dari konsol postgres.



```
ALTER USER fnurhidayat WITH SUPERUSER;
```



## Login ke database

Sebenarnya ketika tadi kita menjalankan perintah ini:

```
sudo -i -u postgres psql
```

Berarti kita akan masuk ke database bernama **postgres** dengan user bernama **postgres**. Ini merupakan user dan database default dari Postgres. Nah, kalau kita ingin login dengan user yang kita buat tadi, kita bisa menulis perintah ini dengan simpel kaya contoh di samping!

```
# psql <nama-database>
# Pastikan login di terminal sebagai user-mu
psql postgres
```

```
~> psql postgres
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
Type "help" for help.

postgres=#
```

## Membuat database baru

Kita bisa membuat database baru melalui konsol, maupun dari terminal langsung.



### 1. Membuat Database dari Konsol

Pertama-tama kita harus masuk dulu ke konsol `psql` dengan *database* `postgres`, lalu kita tuliskan *query* seperti ini:

```
# Untuk penamaan database standar yang  
dipakai adalah menggunakan kaidah  
snake_case  
  
CREATE DATABASE nama_database;
```

Nah, ketika sudah muncul seperti gambar di sebelah kanan, berarti kita udah berhasil nih membuat *database* baru.

```
postgres=# CREATE DATABASE chapter_6;  
CREATE DATABASE  
postgres=#
```



Nah, untuk cek apakah database-nya benar-benar terbuat, kita bisa meminta list database yang ada di sistem kita dengan perintah simpel berikut ini di Postgresql:

```
\l
```

Maka nanti *output*-nya akan seperti gambar di samping.

| List of databases                   |             |          |             |             |                       |
|-------------------------------------|-------------|----------|-------------|-------------|-----------------------|
| Name                                | Owner       | Encoding | Collate     | Ctype       | Access privileges     |
| 30m_development                     | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| armand_test                         | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| bukak-sithik-jest_development       | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| bukak-sithik-jest_test              | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| bukak-sithik_production             | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| cc4-day_development                 | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| cc4_day_development                 | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| chapter_6                           | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| day-class-session-15                | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| day-class-session-15_development    | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| day-session-16                      | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| deploy-latihan                      | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| extra_development                   | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| jest-22_development                 | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| jest-22_test                        | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| jest-22d_development                | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| jest-22d_test                       | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| jest-pug_development                | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| latihan-relationship                | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| passport_development                | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| postgres                            | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| relationship-session-19_development | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| relationship_tutorial               | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| sendgrid_development                | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| sendgrid_test                       | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| session-14                          | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| session-15_development              | fnurhidayat | UTF8     | en_US.UTF-8 | en_US.UTF-8 |                       |
| template0                           | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres           |
|                                     |             |          |             |             | postgres=CTc/postgres |



## 2. Membuat Database dari Terminal

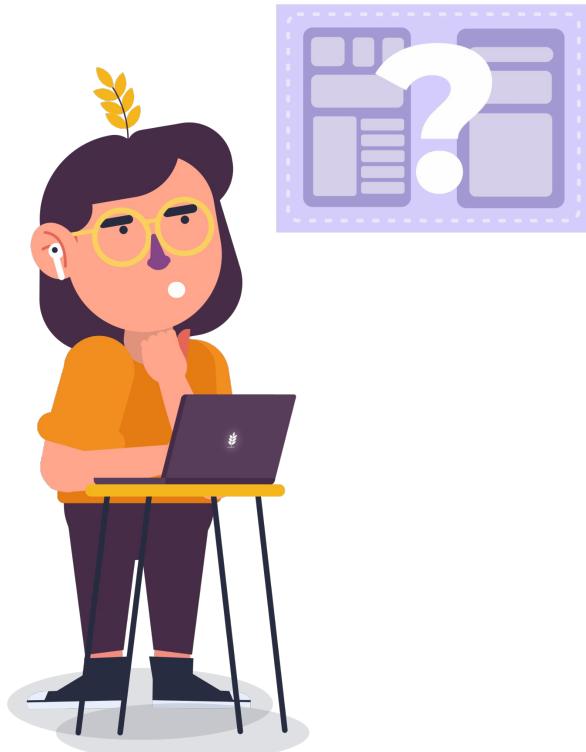
Untuk membuat *database* melalui terminal langsung, kita hanya perlu *login* dulu sebagai *user* yang kita pakai di dalam psql. Lalu, kita bisa jalankan perintah:

```
createdb nama_database
```

```
~> createdb chapter_6_terminal  
~>
```

Untuk mengecek apakah *database*-nya benar-benar sudah dibuat, kita bisa jalankan perintah ini:

```
psql postgres -c "\l"
```



Sama halnya kayak DBMS berbasis SQL, **untuk implementasi DBMS NoSQL ini kita perlu menginstall MongoDB** terlebih dulu.

Kalau kamu bingung gimana caranya, kamu bisa ikuti petunjuk instalasi pada link di bawah ini ya! Kamu bisa memilih link yang sesuai dengan sistem operasi di komputer/laptop yang kamu gunakan:

- [Linux](#)
- [Mac](#)
- [Windows](#)



Setelah menginstal MongoDB, maka akan ada aplikasi baru di terminal kita bernama **mongo**. Mongo ini adalah sebuah shell seperti psql, yang digunakan untuk melakukan query di MongoDB.

Untuk masuk ke Mongo Shell, kita hanya perlu menjalankan perintah sederhana di bawah ini:

```
mongosh
```

Setelah itu, kita akan masuk ke **mongo shell**. Kita akan langsung masuk ke database bernama test. Di MongoDB kita gak perlu melakukan setup untuk user, karena secara default akan langsung membuat user dari terminal kita.

```
~> mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
Server has startup warnings:
2020-07-06T09:09:04.942+0700 I STORAGE [initandlisten]
2020-07-06T09:09:04.942+0700 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-07-06T09:09:04.942+0700 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-07-06T09:09:06.101+0700 I CONTROL [initandlisten]
2020-07-06T09:09:06.101+0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-07-06T09:09:06.101+0700 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-07-06T09:09:06.101+0700 I CONTROL [initandlisten]
```



Selanjutnya, kita akan coba melakukan beberapa **basic setup untuk MongoDB**, yaitu:

- Membuat database baru
- Membuat data artikel di MongoDB (insert, update, dan delete data)
- Mencari data sesuai kondisi





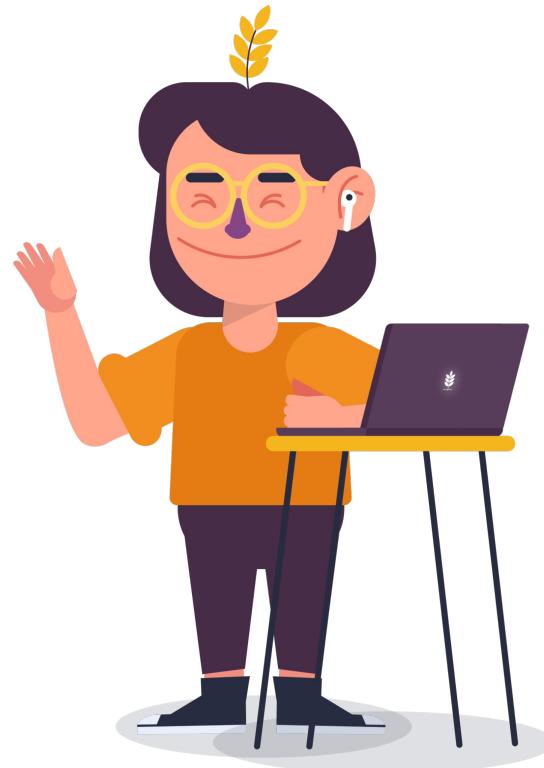
## Membuat *database* baru

Untuk membuat *database* di MongoDB, kita gak perlu menulis *query* yang kompleks seperti di SQL tadi, kita hanya perlu menulis perintah ini di dalam Mongo Shell.

```
use nama_database
```

Ketika kita menggunakan operator **use**, maka Mongo Shell akan langsung mencari *database*. Kalau gak ada, maka MongoDB akan otomatis membuat *database* tersebut. Untuk mengecek kita berada di *database* mana, bisa gunakan perintah ini, lalu akan muncul nama *database*-nya.

```
db
```





## Membuat data artikel di MongoDB

Di topik ini kita gak akan membahas lebih lanjut tentang MongoDB, kita hanya akan coba pelajari gimana caranya melakukan *insert*, *update*, dan *delete* data.

### Insert Data

Misalnya, kita ingin membuat data artikel yang memiliki properti seperti berikut ini:

- Title
- Body
- Approved

Karena data ini akan kita masukkan ke dalam koleksi bernama artikel, maka collection kita nanti akan bernama articles.





Untuk melakukan insert data, secara default perintahnya seperti ini:



```
db.nama_collection.insertOne({  
    "namaField": "Nilai",  
    "iniNamaField": true  
})
```

Nah, untuk percobaan kita membuat artikel, perintahnya akan seperti pada gambar di samping ini.

```
> db.articles.insertOne({  
... title: "Hello World",  
... body: "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has  
been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of  
type and scrambled it to make a type specimen book. It has survived not only five centuries, but also  
the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960  
s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop  
publishing software like Aldus PageMaker including versions of Lorem Ipsum.",  
... approved: true  
... })  
{  
    "acknowledged" : true,  
    "insertedId" : ObjectId("5f02c4c72cfbb948d4ff899e")  
}  
> |
```



Untuk mengecek apakah datanya sudah masuk, kita bisa lakukan *query* dengan perintah ini:



```
db.articles.find({}).pretty()
```

Hasilnya akan seperti pada gambar di bawah ini.

```
> db.articles.find({}).pretty()
{
    "_id" : ObjectId("5f02c4c72cfbb948d4ff899e"),
    "title" : "Hello World",
    "body" : "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ips
um has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a ga
lley of type and scrambled it to make a type specimen book. It has survived not only five centuries, b
ut also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in t
he 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with d
esktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.",
    "approved" : true
}
> █
```



## Update Data

Untuk melakukan *update* data, kita hanya perlu informasi dari data yang akan kita *update*, dan datanya. Sebagai contoh, artikel yang udah kita buat tadi, memiliki properti *approved* yang nilainya **false**. Nah, kita akan coba *update* nilai tersebut jadi **true**.

Tapi, kita juga perlu informasi unik dari artikel tersebut. Nah, data yang unik dari collection *articles* adalah **\_id**, maka kita harus copy nilai id tersebut dari *article* yang barusan kita buat tadi. Nanti perintah dan hasilnya akan tampak seperti contoh di bawah ini:

```
> db.articles.update( { _id: ObjectId("5f02c4c72cfbb948d4ff899e") }, { $set: { approved: true } })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```



## Delete Data

Untuk menghapus data sangat mudah, kita cuma butuh kondisi data apa aja yang akan dihapus. Misalnya, kita ingin hapus data dengan `_id` yang bernilai

`ObjectId("5f02c4c72cfbb948d4ff899e")`.

Gunakan perintah seperti ini:



```
db.articles.deleteOne({ _id: ObjectId("5f02c4c72cfbb948d4ff899e") })
```

## Mencari data sesuai kondisi

Misalnya, kita mau cari artikel yang belum di-*approve*, kita bisa menjalankan *query* seperti ini:



```
db.articles.find({ approved: false }).pretty()
```

Ok, sekarang kalian udah tau nih, gimana sih cara pake DBMS. Nah, ketika kita ngomongin soal database, biasanya ga jauh-jauh nih dengan yang namanya skema data.

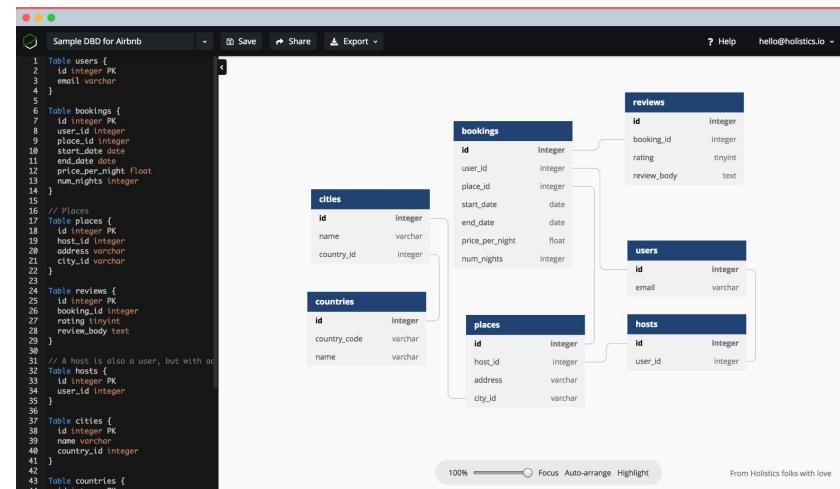
Untuk mendokumentasikan skema atau definisi data, kita bisa menggunakan yang namanya **Entity Relationship Diagram**.





## Entity Relationship Diagram (ERD)

Sebuah diagram yang digunakan untuk mendokumentasikan bentuk data atau skema dalam sebuah database, dan juga hubungannya satu dengan yang lain. Salah satu tools yang umum digunakan dalam membuat ERD adalah [dbdiagram.io](https://www.dbdiagram.io)



### Referensi buat kamu yang suka kepo-kepo~

- <https://www.edureka.co/blog/what-is-dbms>
- <https://www.edureka.co/blog/sql-vs-nosql-db/#:~:text=The%20SQL%20databases%20have%20a,varies%20from%20database%20to%20database>
- <https://www.guru99.com/nosql-tutorial.html>
- <https://www.dicoding.com/blog/manfaat-database-untuk-pemrograman-web/>
- <https://medium.com/@muhamadenrinal/the-sql-vs-nosql-1b5a2778374e>



Nahhh sekarang sudah tuntas kita bahas tentang database, selanjutnya kita bakal bahas lebih dalam tentang Database SQL.

Letsgow !



# Terima Kasih!



Next Topic

loading...