



# Back End Development With Java Springboot Program

+

Sesi 11



# NoSQL - MongoDB

## DB

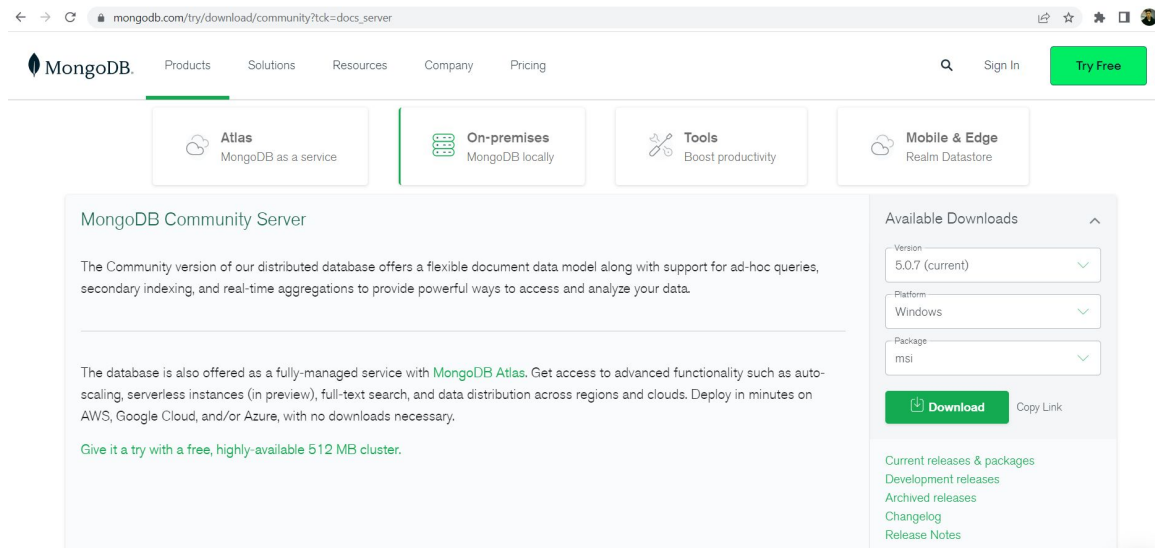
# Introduction

MongoDB dapat diinstall di hampir semua Operating System

Jenis MongoDB server yang dapat digunakan secara gratis adalah MongoDB Community Server.

### 1. Windows (Windows 7, 10)

Download [installernya](#) kemudian install sesuai instruksi.



# Introduction

### 2. Linux (Ubuntu, Debian)

- Import public key

Jalankan perintah berikut ini dari terminal

```
$ wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
```

- Tambahkan sourcelist

Ubuntu 20.0

```
$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
```

- Update package list dan Install MongoDB

```
$ sudo apt-get update
```

```
$ sudo apt-get install -y mongodb-org
```



- Start MongoDB

```
$ sudo systemctl start mongod
```

Untuk memeriksa apakah MongoDB server sudah berjalan dengan baik jalankan perintah berikut:

```
$ sudo systemctl status mongod
```

```
→ ~ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-11-09 11:03:27 WIB; 3s ago
     Docs: https://docs.mongodb.org/manual
    Main PID: 474058 (mongod)
      Memory: 201.8M
      CGroup: /system.slice/mongod.service
              └─474058 /usr/bin/mongod --config /etc/mongod.conf
```

## Introduction

### 3. Mac OS

Disarankan untuk menginstall MongoDB Server menggunakan package manager Homebrew.

```
$ brew update  
$ brew install mongodb
```

## Introduction of Mongo DB

Database SQL (MySQL)	Database MongoDB
Database	Database
Table	Collection
Field / Column	Field
Row	Document

## Konsep Dasar : JSON vs BSON

MongoDB menyimpan data dalam bentuk BSON document, lalu apa itu BSON dan adakah hubungannya dengan JSON?

JSON adalah singkatan dari JavaScript Object Notation, JSON menjadi salah satu format data yang paling banyak digunakan untuk transfer data antara browser dan server.

Pada dasarnya JSON hanya sebuah text yang memiliki format/syntax sebagai berikut:

```
{  
  "key" : "value"  
}
```

Sedangkan **BSON** adalah singkatan dari Binary JSON, BSON memiliki beberapa keunggulan dibandingkan dengan JSON diantaranya tingkat parsing yang lebih cepat dan mendukung lebih banyak tipe data native seperti :

dates dan binary data.



# Introduction

### **Table vs Collection**

Pada SQL database seperti MySQL, sebuah database adalah gabungan dari satu atau banyak tabel. Sedangkan pada MongoDB, sebuah database adalah gabungan dari satu atau banyak collection.

Yang perlu diperhatikan disini adalah kita bisa menyimpan data ke dalam collection dengan bentuk yang tidak terstruktur / beraturan berbeda dengan table pada MySQL.



# Operasi Dasar

Operasi dasar pada MongoDB mirip dengan operasi dasar pada SQL database pada umumnya. Untuk dapat berinteraksi dengan MongoDB server kita bisa gunakan MongoDB shell.

Buka terminal dan eksekusi perintah berikut:

```
$ mongo
```

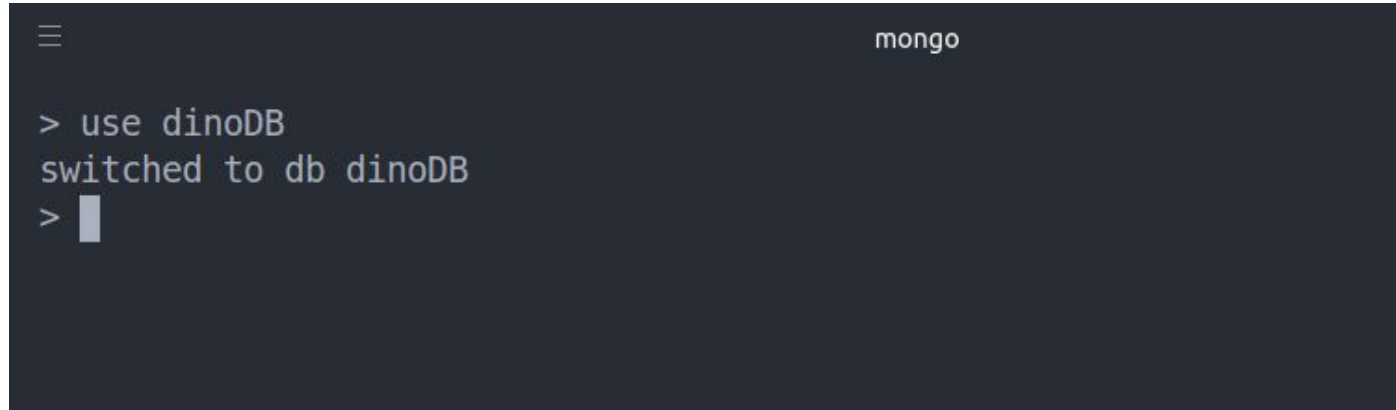
### 1. Membuat Database

Perintah yang digunakan untuk membuat database:

```
use DATABASE_NAME
```



Perintah diatas akan membuat sebuah database baru jika tidak ditemukan database dengan nama yang sama di server. Jika sudah ada database dengan nama yang sama, maka perintah tersebut akan menampilkan database yang sudah ada. Sebagai contoh kita akan buat sebuah database dengan nama dinoDB.

A terminal window with a dark background. The title bar at the top right says 'mongo'. On the left side of the terminal, there is a hamburger menu icon (three horizontal lines). The terminal shows the following text: '> use dinoDB', 'switched to db dinoDB', and '>' followed by a cursor (a small white rectangle).

```
mongo  
  
> use dinoDB  
switched to db dinoDB  
> 
```

Untuk melihat database yang sudah ada kita gunakan perintah

```
show dbs
```

## 2. Membuat Collection

Perintah atau method yang digunakan untuk membuat collection:

```
db.createCollection(name, options)
```

options disini adalah parameter yang bisa kita tambahkan dalam membuat sebuah collection dan bersifat optional.

```
≡ mongo

> db.createCollection("profile");
{ "ok" : 1 }
> █
```

# Operasi CRUD

Operasi CRUD (Create, Read, Update, Delete ) adalah operasi dasar di dunia database / storage.

### 3. Create

Create berarti menambahkan data(selanjutnya kita sebut document) ke dalam collection.

MongoDB menyediakan dua metode untuk menambahkan document, yaitu:

- `db.collection.insertOne()`, untuk menambahkan document tunggal/single
- `db.collection.insertMany()`, untuk menambahkan banyak document

Contoh menambahkan dokumen tunggal



```
≡ mongo
> db.profile.insertOne({ name: "t-rex", color: "red" });
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5fa8e9d62d3aec818293f0ed")
}
> █
```

Setiap document yang berhasil ditambahkan ke dalam database, MongoDB secara otomatis akan membuat sebuah id untuk document tersebut.

id ini memiliki fungsi untuk membedakan antara satu document dengan document yang lainnya, mirip dengan primary key pada SQL table.

Contoh menambahkan multi dokumen:

```
mongo
> db.profile.insertMany([ { name: "tricera" }, { name: "brachio" } ]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5fa8eb6f2d3aec818293f0ee"),
    ObjectId("5fa8eb6f2d3aec818293f0ef")
  ]
}
```

#### 4. Read

Read adalah membaca document yang ada di dalam MongoDB database. Perintah atau method yang digunakan adalah:

```
db.collection.find()
```

Contoh menampilkan semua data:

```
≡ mongo
> db.profile.find()
{ "_id" : ObjectId("5fa8e9d62d3aec818293f0ed"), "name" : "t-rex", "color" : "red" }
{ "_id" : ObjectId("5fa8eb6f2d3aec818293f0ee"), "name" : "tricera" }
{ "_id" : ObjectId("5fa8eb6f2d3aec818293f0ef"), "name" : "brachio" }
>
```



Kita bisa tambahkan filter untuk menampilkan data sesuai filter yang diberikan

```
≡ mongo  
  
> db.profile.find({ name: "t-rex" });  
{ "_id" : ObjectId("5fa8e9d62d3aec818293f0ed"), "name" : "t-rex", "color" : "red" }  
> █
```

## 5. Update

Untuk memperbaharui sebuah document kita gunakan method berikut:

- `db.collection.updateOne(filter, update)`, ubah satu document
- `db.collection.updateMany(filter, update)`, ubah banyak document sekaligus sesuai filter
- `db.collection.replaceOne(filter, update)`, ganti satu document dengan document yang baru

sesuai filter

Kita update document dengan filter color: "red" kemudian update name dari t-rex menjadi allo.

```

≡                                     mongo
> db.profile.updateOne( { color: "red" }, { $set: { name: "allo" } } )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.profile.find()
[ { "_id" : ObjectId("5fa8e9d62d3aec818293f0ed"), "name" : "allo", "color" : "red" }
  { "_id" : ObjectId("5fa8eb6f2d3aec818293f0ee"), "name" : "tricera" }
  { "_id" : ObjectId("5fa8eb6f2d3aec818293f0ef"), "name" : "brachio" }
> ]
```

## 6. Delete

Ada 2 method yang dapat digunakan untuk menghapus document.

- `db.collection.deleteOne(filter)`, hapus satu document
- `db.collection.deleteMany(filter)`, hapus banyak document sekaligus sesuai filter

Contoh:

Delete document dimana name="brachio".

```
≡ mongo
> db.profile.deleteOne({ name: "brachio" });
{ "acknowledged" : true, "deletedCount" : 1 }
> db.profile.find()
{ "_id" : ObjectId("5fa8e9d62d3aec818293f0ed"), "name" : "allo", "color" : "red" }
{ "_id" : ObjectId("5fa8eb6f2d3aec818293f0ee"), "name" : "tricera" }
> █
```

## Introduction of Relasi

Suatu hubungan antar tabel pada database dimana suatu tabel mempunyai data yang sama dengan data yang ada di tabel lainnya. Terdapat 3 macam relasi yaitu :

1. Relasi one to one

Hubungan antar tabel dimana tabel A adalah data master dan tabel B harus mempunyai data yang ada pada tabel A. Contohnya :

Tabel pasien

Nama : "Foxy"

Umur : 20

Penyakit : sakit\_0001

Tabel penyakit

\_id : sakit\_0001

Penyakit : flu

Disini tabel penyakit merupakan data master atau data inti ya dan tabel pasien disini mempunyai data yang ada pada tabel penyakit.

Coba di buat sebuah database bernama rumah\_sakit, dan masukkan data pasien dan penyakit.

```
> var penyakit_id = db.pasien.findOne().penyakit
```

Buat sebuah variabel baru (variabel berfungsi untuk penampungan data sementara sedangkan “db.pasien.findOne().penyakit” berfungsi untuk menampilkan data sakit\_0001 jadi variabel penyakit\_id berisikan sakit\_0001. Lalu ketikan seperti ini :

```
> db.penakit.findOne({_id : penyakit_id})  
{ "_id" : "sakit_0001", "penyakit" : "flu" }
```

Disini dengan menyimpan data penyakit pasien kita dapat mencari nama penyakitnya di tabel penyakit.

## Introduction of Relasi

### 2. Relasi one to many

Sebuah hubungan antara tabel dimana tabel A memiliki sebuah data yang bisa di pakai pada tabel B data tersebut bisa banyak data atau beberapa data saja. Contohnya :

Tabel pelanggan

`_id : "PL0001"`

`nama_pelanggan : "Brudi"`

Tabel transaksi

`_id : TR0001,`

`Tanggal_transaksi : New Date(),`

`Total_harga : 100000,`

`Id_pelanggan : "PL0001"`



Tabel pelanggan adalah tabel masternya sedangkan tabel transaksi adalah yang akan di relasikan.

Coba di buat sebuah database bernama transaksi\_nama-masingmasing, dan masukkan data pelanggan dan transaksi (tambahkan data transaksi minimal 3 data, untuk isi datanya bebas asalkan sesuai dengan format data di atas kecuali id\_pelanggan diisi sama dengan di atas). Kalau sudah coba ketika `“db.transaksi.find().pretty()”`

nah maka akan muncul semua datanya. Disini untuk id transaksinya berbeda – beda tetapi id\_pelanggannya tetap sama nah itu relasi dari one to many atau satu ke banyak ya (dari 1 data pelanggan bisa menghasilkan banyak data pada transaksi).



## Introduction of Relasi

### 3. Relasi many to many

Sebuah hubungan antara tabel dimana tabelnya ini ada banyak. Hubungannya itu bisa banyak tabel misalkan tabel A dengan tabel B dan tabel C dengan tabel B seperti itu. Contohnya :

```
Tabel detail transaksi
```

```
{
```

```
    no_trans : "TR0001",
```

```
    barang : "Gelas",
```

```
    jumlah : 3
```

```
}
```





Coba kita buat tabel detail transaksi pada tabel transaksi\_namamasing-masing yang sudah di buat sebelumnya. Selanjutnya coba di tambahkan datanya lagi selain gelas minimal 3 data dan untuk no\_trans nya jangan di rubah atau disamakan saja dengan data di atas

Disini tabel transaksi yang sebelumnya sudah berrelasi dengan tabel pelanggan bisa berrelasi lagi dengan tabel yang berbeda yaitu tabel detail transaksi itu relasi dari One to Many

## C. LOOKUP

Lookup adalah sebuah metode yang digunakan untuk menampilkan data seperti find tetapi di lookup ini kita dapat menampilkan beberapa tabel untuk di tampilkan (lookup ini bisa berjalan kalau tabelnya sudah berelasi)

Contohnya :

```
db.collection.aggregate({ $lookup : { from : collection2, localField : "_id", foreignField : "_id", as : "join" } })
```

Keterangan :

Aggregate : untuk mengelompokkan data

Lookup : untuk mengabungkan data

From : diisi tabel ke 2 yang akan di tampilkan

localField : diisi dengan nama data yang ada di tabel pertama (data ini harus data yang nanti sama dengan data yang ada di tabel ke 2)

foreignField : sama dengan localField tetapi untuk foreignField diisi dengan nama data yang ada di tabel kedua

as : as atau alias dapat diisi bebas karena disini jika nanti data kita mau di tampilkan kita cukup memanggil nama yang sudah di aliaskan saja.

```
> db.transaksi.aggregate({ $lookup : { from : "detail_transaksi", localField : "_id", foreignField : "no_trans", as : "detail" } } )
{ "_id" : "TR0001", "tanggal_transaksi" : ISODate("2018-11-21T13:29:04.413Z"), "total_harga" : 100000, "id_pelanggan" : "PL0001", "detail" : [ { "_id" : ObjectId("5bf7fb6d80c75c064d3e09ee"), "no_trans" : "TR0001", "barang" : "Gelas", "jumlah" : 3 }, { "_id" : ObjectId("5bf7fc1100c75c064d3e09ef"), "no_trans" : "TR0001", "barang" : "Piring", "jumlah" : 12 } ] }
{ "_id" : "TR0002", "tanggal_transaksi" : ISODate("2018-11-21T13:29:26.507Z"), "total_harga" : 100000, "id_pelanggan" : "PL0001", "detail" : [ ] }
{ "_id" : "TR0003", "tanggal_transaksi" : ISODate("2018-11-21T13:29:32.855Z"), "total_harga" : 100000, "id_pelanggan" : "PL0001", "detail" : [ ] }
```



Untuk menampilkan 3 tabel atau lebih sekaligus kita dapat menggunakan cara ini:

```
db.collection.aggregate( [ { $lookup : { from : collection2, localField : "_id", foreignField : "_id", as : "join" } }, { $lookup : { from : collection3, localField : "_id", foreignField : "_id", as : "join" } } ] )
```

Keterangan : Pembedanya adalah pada kurung [ ] nya ya

```
> db.transaksi.aggregate([ { $lookup : { from : "detail_transaksi", localField : "_id", foreignField : "no_trans", as : "detail" } }, { $lookup : { from : "pelanggan", localField : "id_pelanggan", foreignField : "_id", as : "customer" } } ] )
```



Validasi adalah sebuah metode untuk pengecekan suatu data yang di masukkan, misalkan pada saat kita login pada suatu website kita ketikan asal – asalan maka akan muncul peringatan.

Coba buat database terlebih dahulu dengan nama “barang\_nama-masing-masing”.

```
db.createCollection("barang", {  
  validator : {  
    $jsonSchema : {  
      }  
    }  
  }  
})
```

Lalu kita masukan ini pada json schema

```
$jsonSchema : {  
  bsonType : "object",  
  Required : ["namabarang", "hargabarang", "jenisbarang", "description"]  
}
```

## Selanjutnya masukan propertiesnya

```
namaBarang : {  
  bsonType : "string",  
  Description : "Nama Barang harus berupa karakter dan tidak boleh kosong!"  
},  
  
hargaBarang : {  
  bsonType : "number",  
  Description : "Harga Barang harus berupa number dan tidak boleh kosong!"  
},  
  
jenisBarang : {  
  bsonType : "objectId",  
  Description : "Jenis Barang harus berupa objectID dan tidak boleh kosong!"  
},  
  
description : {  
  bsonType : "array",  
  Description : "Deskripsi harus berupa karakter dan tidak boleh kosong!"  
},
```



Pada description karna di berupa array coba di tambahkan lagi seperti ini :

```
description : {  
  bsonType : "array",  
  Description : "Deskripsi harus berupa karakter dan tidak boleh kosong!",  
  Items : {  
    Merk : {  
      bsonType : "string",  
      description : "Ukuran harus berupa karakter dan tidak boleh kosong!"  
    },  
    ukuran : {  
      bsonType : "string",  
      description : "Ukuran harus berupa karakter dan tidak boleh kosong!"  
    },  
    stok : {  
      bsonType : "number",  
      description : "Stok harus berupa angka dan tidak boleh kosong!"  
    }  
  }  
},
```



Lalu tambahkan script ini di bawah items

```
description : {  
  bsonType : "array",  
  description : "Deskripsi harus berupa karakter dan tidak boleh kosong!",  
  items : {  
    bsonType : "object",  
    required : ["merk", "ukuran", "stok"],  
    Properties : [  
      Merk : {  
        bsonType : "string",  
        description : "Ukuran harus berupa karakter dan tidak boleh kosong!"  
      },  
    ],  
  },  
}
```



Jangan lupa di kasih ) di bagian paling bawah untuk menutup createCollection.

Keterangan :

createCollection : untuk membuat tabel

validator : untuk membuat validasi pada json

\$jsonSchema : untuk masuk ke dalam fungsi json

bsonType : tipe data

required : data apa saja yang pada saat di tambahkan datanya itu harus diisi

properties : peraturan – peraturan yang akan dibuat untuk validasi data tersebut

items : untuk data bertipe array

Terakhir coba di jalankan.

