

MODUL 5

DATA MODEL DAN QUERY

5.1. Tujuan Praktikum

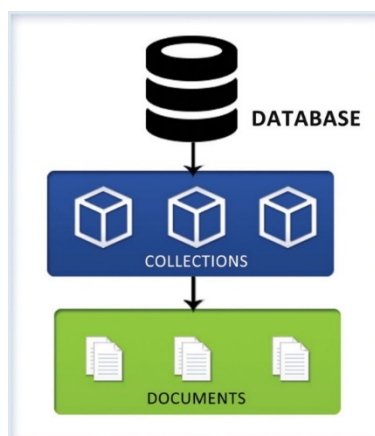
1. Mengenal data model pada mongodb
2. Mahasiswa Menguasai query dasar dan query kompleks

5.2. Dasar Teori

5.2.1. Data Model MongoDB

Pada bab sebelumnya, kita telah mempelajari bahwa MongoDB adalah sistem basis data berbasis dokumen di mana dokumen dapat memiliki skema yang fleksibel. Ini berarti bahwa dokumen dalam koleksi dapat memiliki rangkaian tipe data yang berbeda (atau sama). Ini memberi lebih banyak fleksibilitas ketika berhadapan dengan data.

MongoDB dapat memiliki banyak database. Setiap database adalah satu set koleksi. Koleksi mirip dengan konsep tabel di SQL; Namun, mereka tanpa skema. Setiap koleksi dapat memiliki beberapa dokumen. Pikirkan dokumen sebagai baris dalam SQL. Gambar 4.1 menggambarkan model database MongoDB.



Gambar 19. model database MongoDB

Dalam sistem RDBMS, karena struktur tabel dan tipe data untuk setiap kolom adalah tetap, Anda hanya dapat menambahkan data dari tipe data tertentu dalam kolom. Di MongoDB, koleksi adalah kumpulan dokumen tempat data disimpan sebagai pasangan nilai kunci. Mari kita pahami dengan contoh bagaimana data disimpan dalam dokumen. Dokumen berikut berisi nama dan nomor telepon pengguna:

```
{"Nama": "ABC", "Telepon": [ "1111111", "222222" ] }
```

Skema dinamis berarti bahwa dokumen dalam koleksi yang sama dapat memiliki kumpulan bidang atau struktur yang sama atau berbeda, dan bahkan bidang umum dapat menyimpan berbagai jenis nilai di seluruh dokumen. Tidak ada kekakuan dalam cara penyimpanan data dalam dokumen koleksi.

Mari kita lihat contoh koleksi Wilayah:

```
{ "R_ID" : "REG001", "Nama" : "Indonesia" }  
{ "R_ID" :1234, "Nama" : "Jakarta" , "Negara" : "Indonesia" }
```

Dalam kode ini, Anda memiliki dua dokumen di koleksi Wilayah. Meskipun kedua dokumen tersebut merupakan bagian dari satu koleksi, mereka memiliki struktur yang berbeda: koleksi kedua memiliki atribut informasi tambahan, yaitu **Negara**. Faktanya, jika Anda melihat atribut "R_ID", ini menyimpan nilai tipe data STRING di dokumen pertama, sedangkan dokumen kedua adalah angka. Jadi dokumen koleksi dapat memiliki skema yang sama sekali berbeda. Semua tergantung pada aplikasi mau menyimpan dokumen dalam koleksi tertentu bersama-sama atau memiliki banyak koleksi.

5.2.2. JSON dan BSON

MongoDB adalah database berbasis dokumen. MongoDB menggunakan Binary JSON untuk menyimpan datanya. Notasi adalah standar yang digunakan untuk pertukaran data di Web modern saat ini (bersama dengan XML). Formatnya dapat dibaca oleh manusia dan mesin. Ini bukan hanya cara yang bagus untuk bertukar data tetapi juga cara yang bagus untuk menyimpan data.

Semua tipe data dasar (seperti string, angka, nilai Boolean, dan array) didukung oleh JSON. Kode berikut menunjukkan seperti apa dokumen JSON:




```
// "Document collections" - "HTMLPage" document  
{  
  _id: 1,  
  title: "Hello",  
  type: "HTMLpage",  
  text: "<html>Hi..Welcome to my world</html>"  
}  
...  
// Document collection also bisa menyimpan dokumen "Gambar"  
{  
  _id: 3,  
  title: "Family Photo",  
  type: "JPEG",  
  sizeInMB: 10,.....  
}
```

Skema ini tidak hanya memungkinkan untuk menyimpan data terkait dengan struktur berbeda secara bersamaan dalam **Collection** yang sama, tetapi juga menyederhanakan query. Collection yang sama dapat digunakan untuk melakukan query pada atribut umum seperti mengambil semua konten yang diunggah pada tanggal dan waktu tertentu serta query pada atribut tertentu seperti menemukan gambar dengan ukuran lebih besar dari X MB. Jadi pemrograman berorientasi objek adalah salah satu contoh kasus penggunaan database dengan skema polimorfik.

5.2.3. Konsep Method, Filter, Operator

Untuk penulisan rumus pada mongodb dibutuhkan ketiga script ini yaitu Method, Filter, dan Operator. Method adalah metode yang di pakai sebagai aksi yang akan digunakan misalkan `find()`, `insertOne()`, `deleteOne()`, dll. Untuk filter adalah sebuah fungsi untuk menyaring sebuah data agar datanya tersebut bisa di kelompokkan dan tidak tampil semua pada saat kita mau menampilkannya contoh `find({ umur : 20 })`. Sedangkan untuk operator adalah sebuah fungsi yang digunakan agar menyaringan data pada filter bisa lebih spesifik lagi misalkan seperti contoh pada filter disitu hanya menyaring data umur yang 20 tahun saja sedangkan ketika kita memkasi operator disini bisa menyaring bedasarkan umur yang lebih dari 20 atau di bawah 20 tahun dan dll. Contohnya :

```
db.mahasiswa.find( { umur : { $gt : 20 } } )
```

	= Method
	= Filter
	= Operator

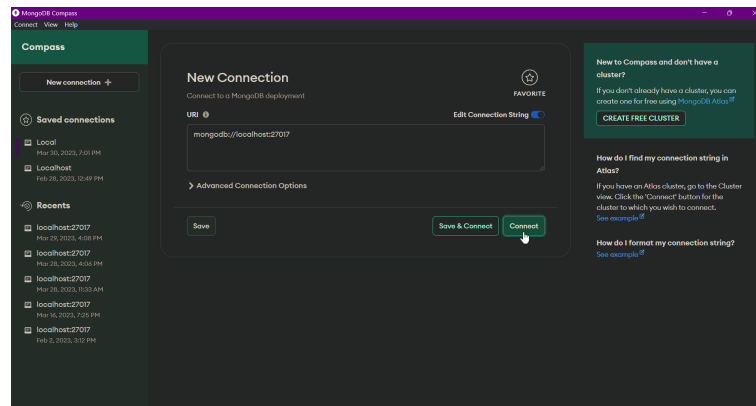
5.4. Software

1. MongoDB
2. Compass
3. MongoDB Shell

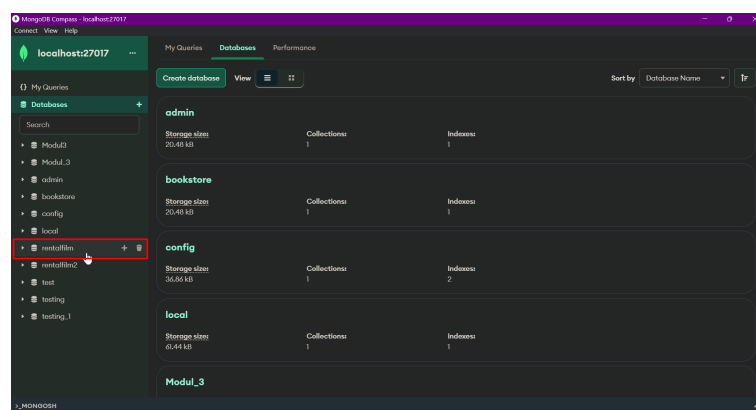
5.5. Tahapan Kerja

5.4.1. Menjalankan MongoDB di Compass dan Command Prompt

1. Buka Aplikasi MongoDB Compass kemudian klik “Connect”



2. Untuk melanjutkan ketahapan kerja selanjutnya pastikan database “Rentalfilm” sudah ada di MongoDB.



3. Buka MongoShell dengan melakukan perintah “mongosh” di dalam command prompt.

```
C:\Users\Denny>mongosh
```

4. Lalu ganti ke database rental film dengan menggunakan perintah “use”.

```
test> use rentalfilm
switched to db rentalfilm
rentalfilm>
```

5.4.2. Query Dasar

1. Untuk mencari data dan menampilkan data tersebut kita dapat menggunakan query:

```
db.<nama collection>.find()
```

Untuk melihat semua isi customers maka kita ketikkan

```
db.customers.find()
```

Ketik query tersebut pada cmd

```
test> use rentalfilm
switched to db rentalfilm
rentalfilm> db.customers.find()
```

```

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

...
> use rentalfilm
switched to db rentalfilm
> db.customers.find()
{ "_id" : ObjectId("6066273448ee35c9edf4e1ad"), "id" : "Astrid", "fullname" : "Astrid Gruber", "email" : "astrid.gruber@apple.at" }
{ "_id" : ObjectId("60662af48b14a546b91b8e17"), "id" : "Fernanda", "fullname" : "Fernanda Ramos", "email" : "fernadaramos4@uol.com.br", "age" : 24 }
{ "_id" : ObjectId("60662af48b14a546b91b8e17"), "id" : "Mark", "fullname" : "Mark Phillips", "email" : "mphilips12@shaw.ca", "city" : "San Francisco" }

```

2. Untuk mencari nilai tertentu kita bisa mengisikan pasangan key dan value di dalam kurung. Misalkan kita ingin mencari data dengan usia 24

```
db.customers.find( {age:24} ).pretty()
```

```

> use rentalfilm
switched to db rentalfilm
> db.customers.find()
{ "_id" : ObjectId("6066273448ee35c9edf4e1ad"), "id" : "Astrid", "fullname" : "Astrid Gruber", "email" : "astrid.gruber@apple.at" }
{ "_id" : ObjectId("60662af48b14a546b91b8e17"), "id" : "Fernanda", "fullname" : "Fernanda Ramos", "email" : "fernadaramos4@uol.com.br", "age" : 24 }
{ "_id" : ObjectId("60662af48b14a546b91b8e17"), "id" : "Mark", "fullname" : "Mark Phillips", "email" : "mphilips12@shaw.ca", "city" : "San Francisco" }
> db.customers.find( {age:24} )
{ "_id" : ObjectId("60662af48b14a546b91b8e17"), "id" : "Fernanda", "fullname" : "Fernanda Ramos", "email" : "fernadaramos4@uol.com.br", "age" : 24 }
> db.customers.find( {age:24} ).pretty()
  "_id" : ObjectId("60662af48b14a546b91b8e17"),
    "id" : "Fernanda",
    "fullname" : "Fernanda Ramos",
    "email" : "fernadaramos4@uol.com.br",
    "age" : 24
}

```

method pretty() membuat tampilan dari data kita menjadi lebih rapi layaknya penulisan dictionary pada python. Seperti disebutkan di awal, tidak ada konsep join pada MongoDB setidaknya secara langsung. Akan tetapi, query-query yang kita gunakan lebih sederhana dan mudah dimengerti layaknya saat kita bermain dengan object pada python.

5.4.3. Query Kompleks

1. \$in & \$nin

\$in adalah sebuah operator yang berfungsi hampir menyerupai sama dengan (=) tetapi kalau dalam operator ini kita bisa menampilkan data lebih dari 1 data. Perintah dasarnya adalah seperti ini :

```
db.tabel.find( { data : { $in : [ nama1,nama2 ] } } ).pretty()
```

Contohnya pada command prompt :

```
Rentalfilm> db.Customer.find({Active : {$in : [ "Y", "N" ]}})
```

```
rentalfilm> db.Customer.find({Active : {$in : ["Y", "N"]}})
[
  {
    _id: 1,
    AddressId: 1,
    AddressColumn: 2,
    FirstName: 'Denny',
    LastName: 'Partala',
    Email: 'email@gmail.com',
    Active: 'Y',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 2,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Otosu',
    LastName: 'Bearu',
    Email: 'email2@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 3,
    AddressId: 3,
    AddressColumn: 3,
    FirstName: 'Bjorn',
    LastName: 'Mato',
    Email: 'email3@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  }
]
```

Sedangkan untuk \$nin adalah kebalikan dari \$in yang berfungsi untuk menampilkan data yang tidak di tampilkan pada saat di \$in di jalankan. Perintah dasarnya adalah seperti ini:

```
db.tabel.find({ data : { $in : [ nama1,nama2] } } ).pretty()
```

Contohnya pada command prompt :

```
rentalfilm> db.Customer.find({Active : {$nin : ["Y", "N"]}}).pretty()
[
  {
    _id: 4,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Mitch',
    LastName: 'Benjamin',
    Email: 'email4@gmail.com',
    Active: 'T',
    CreateDate: '2023-03-15T10:22:52.525+00:00',
    LastUpdate: '2023-03-15T10:22:52.525+00:00'
  }
]
```

2. \$and, \$or, \$not

And, or, not adalah sebuah operator yang digunakan untuk menyaring data dengan syarat tertentu agar datanya bisa ditampilkan. \$and adalah sebuah operator yang digunakan untuk menyaring data dengan 2 atau lebih kriteria dimana semua kriterianya itu harus benar untuk menampilkan datanya. \$or adalah sebuah operator yang digunakan untuk menyaring data dengan 2 atau lebih kriteria dimana jika salah satu kriteria terpenuhi maka akan menampilkan datanya. \$not adalah sebuah operator yang digunakan untuk menyaring data dengan menampilkan kebalikan dari hasil yang seharusnya. Ketentuan dari \$and dan \$or adalah seperti ini :

Kriteria 1	Kriteria 2	\$and	\$or
Benar	Benar	Benar	Benar
Salah	Salah	Salah	Salah
Salah	Benar	Salah	Benar
Benar	Salah	Salah	Benar

Perintah dasar \$and seperti ini:

```
db.tabel.find( { $and : [ { data1 : isi1 }, { data2 : isi2 } ] } ).pretty()
```

```
rentalfilm> db.Customer.find( { $and : [ { AddressId : 2, Active : "T" } ] } )
[
  {
    _id: 4,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Mitch',
    LastName: 'Benjamin',
    Email: 'email4@gmail.com',
    Active: 'T',
    CreateDate: '2023-03-15T10:22:52.525+00:00',
    LastUpdate: '2023-03-15T10:22:52.525+00:00'
  }
]
```

Perintah dasar \$or seperti ini:

```
db.tabel.find( { $or : [ { data1 : isi1 }, { data2 : isi2 } ] } ).pretty()
```

```
rentalfilm> db.Customer.find( { $or : [ { Active : "Y" }, { AddressId : 3 } ] } )
[
  {
    _id: 1,
    AddressId: 1,
    AddressColumn: 2,
    FirstName: 'Denny',
    LastName: 'Partala',
    Email: 'email@gmail.com',
    Active: 'Y',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 3,
    AddressId: 3,
    AddressColumn: 3,
    FirstName: 'Bjorn',
    LastName: 'Mato',
    Email: 'email3@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  }
]
```

Perintah dasar \$not seperti ini:

```
db.tabel.find( { data : { $not : { $eq : isi } } } ).pretty()
```

```
rentalfilm> db.Customer.find( { AddressId: { $not : { $eq : 1 } } } )
```

```
rentalfilm> db.Customer.find( {AddressId: { $not : { $eq : 1}}} )
[
  {
    _id: 2,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Otoso',
    LastName: 'Bearu',
    Email: 'email2@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 3,
    AddressId: 3,
    AddressColumn: 3,
    FirstName: 'Bjorn',
    LastName: 'Mato',
    Email: 'email3@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 4,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Mitch',
    LastName: 'Benjamin',
    Email: 'email4@gmail.com',
    Active: 'T',
    CreateDate: '2023-03-15T10:22:52.525+00:00',
    LastUpdate: '2023-03-15T10:22:52.525+00:00'
  }
]
```

Keterangan :

- \$eq adalah equal atau sama dengan(=)
- \$ne adalah not equal atau tidak sama dengan(!=)
- \$eq atau \$ne rumus yang harus digunakan dalam menggunakan \$not kalau tidak di gunakan salah satu rumus ini maka akan error.

3. \$exists, \$type, \$regex

Operator exists adalah operator yang digunakan untuk pengecekan data apakah data di suatu tabel ada atau tidak. Contohnya pada suatu tabel barang terdapat data gelas dan piring. Pada data gelas terdapat sebuah data diskon tetapi pada data piring tidak ada data diskon nah pada kasus ini kita dapat memakai \$exists untuk mengecek keberadaan suatu datanya. Perintah dasarnya adalah sebagai berikut :

```
db.tabel.find( {data : { $exists : true } } ).pretty()
```

```
rentalfilm> db.Customer.find({Email: {$exists : true }})
```



```

rentalfilm> db.Customer.find( {Email : { $exists: true}})
[
  {
    _id: 1,
    AddressId: 1,
    AddressColumn: 2,
    FirstName: 'Denny',
    LastName: 'Partala',
    Email: 'email@gmail.com',
    Active: 'Y',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 2,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Otoso',
    LastName: 'Bearu',
    Email: 'email2@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 3,
    AddressId: 3,
    AddressColumn: 3,
    FirstName: 'Bjorn',
    LastName: 'Mato',
    Email: 'email3@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 4,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Mitch',
    LastName: 'Benjamin',
  }
]

```

Contoh diatas merupakan perintah untuk mencari semua data yang dimana field “Email” ada.

\$type

Operator type adalah operator yang digunakan untuk menyaring atau mensortir data berdasarkan suatu tipe data pada sebuah data tertentu. Contohnya pada tabel barang terdapat price yang bertipe data number lalu kita akan mencarinya berdasarkan type data number maka akan muncul data barang apabila kita mencarinya berdasarkan type data string maka hasilnya tidak akan muncul. Perintah dasarnya adalah sebagai berikut:

```
db.tabel.find( {data : { $type : "tipe-datanya" } } ).pretty()
```

```
rentalfilm> db.Customer.find({FirstName: {$type : "string" }})
```

```
rentalfilm> db.Customer.find( {FirstName: {$type : "string"}} )
[
  {
    _id: 1,
    AddressId: 1,
    AddressColumn: 2,
    FirstName: 'Denny',
    LastName: 'Partala',
    Email: 'email@gmail.com',
    Active: 'Y',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 2,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Otoso',
    LastName: 'Bearu',
    Email: 'email2@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 3,
    AddressId: 3,
    AddressColumn: 3,
    FirstName: 'Bjorn',
    LastName: 'Mato',
    Email: 'email3@gmail.com',
    Active: 'N',
    CreateDate: ISODate("2023-03-15T10:22:52.525Z"),
    LastUpdate: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 4,
    AddressId: 2,
    AddressColumn: 2,
    FirstName: 'Mitch',
    LastName: 'Benjamin',
    Email: 'email4@gmail.com',
  }
]
```

\$regex

Operator regex adalah operator yang digunakan untuk mencari sebuah data berdasarkan kata atau huruf tertentu. Misalkan kita mempunyai data mata kuliah dengan nama “Basis Data Lanjut” lalu kita mau mencari data yang Namanya Basis Data maka kita bisa menggunakan operator ini. Contoh penulisan operator regex :

```
db.tabel.find( {data : { $regex : /isi-yang-mau-dicari/ , $options : '<options>' } } ).pretty()
```

```
rentalfilm> db.users.find( {name : {$regex : /^B/, $options : 'm'}} )
[
  {
    _id: 5,
    name: 'Bosco',
    status: 1,
    dob: ISODate("2023-04-04T04:09:26.587Z"),
    gender: 'M',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z"),
    nama: 'Dendol'
  }
]
```

Keterangan :

- \$options bisa di isi dengan berbagai jenis value contohnya : ‘m’ ketika ingin menggunakan simbol anchor seperti ^, &.
- isi-yang-mau-dicari bisa diisi dengan nama atau huruf yang dicari misalkan mau mencari basisdatalog kita bisa menulis /basisdatalog/.

Selain // kita juga dapat menggunakan :

/^ isi/ = untuk pencarian nama sesuai huruf depannya misalkan pada pencarian di atas kita menulisnya /^Basis/

/isi\$/ = untuk pencarian nama sesuai huruf belakang misalnya /Lanjut&/ untuk pencarian “Basis Data”

4. . \$expr

Operator expr adalah sebuah operator yang berfungsi untuk membandingkan suatu data tertentu. Misalkan kita mempunyai data harga dan data modal pada sebuah tabel, Lalu kita cek apakah harga melebihi dari modal yang diberikan kalau iya maka datanya akan muncul tetapi jika tidak maka dia tidak akan muncul. Perintah dasarnya adalah seperti ini:

```
db.tabel.find( { $expr : { $gt : [ "$harga", "$modal" ] } } ).pretty()
```

```
rentalfilm> db.Film.find({ $expr : { $gt : [ "$Rental_Duration", "$Replacement_Cost" ] } })
[
  {
    _id: 1,
    LanguageID: 1,
    Title: 'Country Bear',
    Description: '-',
    Release_Year: 2023,
    Rental_Duration: 1608,
    Rental_Rate: 10.2,
    Length: 90,
    Replacement_Cost: 1200,
    Rating: 9,
    Last_Update: '2023-03-15T10:22:52.525+00:00',
    Special_Features: 'Bears',
    Fulltext: 'Bear Bear Bear'
  }
]
```

Keterangan :

- \$gt adalah operator disini kita dapat menggunakan operator lain sesuai kebutuhan.
- \$harga dan \$modal adalah sebuah nama di kolom tabel dan disini dapat kita panggil dengan menggunakan dolar (\$)

5. \$Next

Operator next adalah sebuah operator yang digunakan untuk menampilkan data di mulai data awal terlebih dahulu sampai data terakhir. Perintah dasar dari next adalah seperti ini:

```
db.tabel.find().next()
```

```
rentalfilm> db.users.find().next()
{
  _id: 1,
  name: 'Denny',
  status: 1,
  dob: ISODate("1970-01-01T00:00:00.000Z"),
  gender: 'M',
  created_at: ISODate("2023-03-15T10:22:52.525Z"),
  updated_at: ISODate("2023-03-15T10:22:52.525Z")
}
```

Jika dijalankan perintah di atas pada command prompt keluar data yang pertama tetapi jika kita jalankan lagi maka datanya tersebut akan memunculkan data pertama lagi.

Untuk memunculkan data berikutnya kita memerlukan satu fungsi lagi yaitu menggunakan “const”, Const adalah sebuah penampung data yang apabila datanya sudah diisi maka tidak bisa dirubah kembali rumus pembuatan const adalah :

```
const nama-variabel = isi
```

```
rentalfilm> const FindNext = db.users.find()  
rentalfilm> FindNext
```

Keterangan:

- Untuk nama-variabel bebas kita mau memberikan nama apa asal kita ingat nanti saat pemanggilannya
- Isi ini bisa kita isikan apa saja bebas asal sesuai dengan kebutuhan contoh kalau pada kasus next kita bisa mengisikan db.tabel.find().

Kalau sudah dibuat sebuah variabelnya sekarang coba ketika “nama- variabel.next()” pada command prompt maka akan keluar data yang pertama, jika kita menuliskannya lagi maka akan muncul data seterusnya sampai data terakhir dan apabila sudah sampai data terakhir ingin menuliskannya lagi maka data tersebut tidak tampil.

6. Sort

Operator sort adalah sebuah operator yang digunakan untuk memilah data contoh kita data mengurutkan data dari A sampai Z atau sebaliknya, bisa juga untuk mengurutkan angka dari yang terkecil terlebih dahulu atau yang terbesar terlebih dahulu. Rumus dari sort adalah seperti ini :

```
db.tabel.find().sort({data:-1}).pretty()
```

```
rentalfilm> db.users.find().sort({_id : -1})
```

```
rentalfilm> db.users.find().sort({_id : -1})
[
  {
    _id: 5,
    name: 'Bosco',
    status: 1,
    dob: ISODate("1970-01-01T00:00:00.000Z"),
    gender: 'M',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 4,
    name: 'Ursidae',
    status: 1,
    dob: ISODate("2000-01-12T00:00:00.000Z"),
    gender: 'F',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 3,
    name: 'Fredrica',
    status: 1,
    dob: ISODate("1999-10-11T00:00:00.000Z"),
    gender: 'F',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 2,
    name: 'Wen',
    status: 1,
    dob: ISODate("2001-08-16T00:00:00.000Z"),
    gender: 'M',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 1,
    name: 'Denny',
    status: 1,
    dob: ISODate("1970-01-01T00:00:00.000Z"),
  }
]
```

Keterangan :

- Data adalah sebuah data yang akan di sortir
- -1 adalah pengurutan data berdasarkan descending atau menurun (selain -1 terdapat juga 1 yaitu ascending atau menaik)

7. Limit

Operator limit adalah sebuah operator yang berfungsi untuk memfilter data yang kita ingin tampilkan dengan jumlah yang telah di tentukan sebelumnya. Rumus dari sort adalah seperti ini :

```
db.tabel.find().limit(5).pretty()
```

```
rentalfilm> db.users.find().limit(2).pretty()
[
  {
    _id: 1,
    name: 'Denny',
    status: 1,
    dob: ISODate("1970-01-01T00:00:00.000Z"),
    gender: 'M',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z")
  },
  {
    _id: 2,
    name: 'Wen',
    status: 1,
    dob: ISODate("2001-08-16T00:00:00.000Z"),
    gender: 'M',
    created_at: ISODate("2023-03-15T10:22:52.525Z"),
    updated_at: ISODate("2023-03-15T10:22:52.525Z")
  }
]
rentalfilm>
```

Keterangan :

Jumlah data yang ingin ditampilkan diisi di dalam kurung pada limit misalkan contoh di atas adalah 5.

8. \$Unset

Operator unset adalah sebuah operator yang berfungsi untuk menghapus data pada tabel. Misalkan kita salah memasukkan sebuah data contoh pada suatu tabel barang data pertama berisi nama-barang lalu di data yang kedua berisi nama-barang, tipe. Untuk menghapus tipe di data yang ke dua agar data bisa sama dengan data yang pertama. Disini kita bisa menggunakan unset, untuk unset rumusnya adalah seperti ini :

```
db.tabel.updateMany({kriteria}, {$unset : {kriteria2} } )
```

```
rentalfilm> db.users.updateMany({_id : 1}, {$unset : {gender: ""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Keterangan :

- Kriteria pertama adalah kriteria yang dipakai untuk update data.
- Kriteria2 adalah kriteria yang dipakai untuk menghapus data tersebut.

9. \$Rename

Operator rename adalah sebuah operator yang berfungsi untuk mengubah nama data pada tabel. Misalkan kita salah memasukkan sebuah data contoh pada suatu tabel barang data pertama berisi nama-barang lalu di data yang kedua berisi nama-barang (salah pengetikan) lalu kita bisa merubah nama-barang yang salah pengetika menjadi nama-barang menggunakan operator rename. Untuk perintah dasarnya adalah seperti ini:

```
db.tabel.updateMany({kriteria}, {$rename : {data: "data-baru"} } )
```

```
rentalfilm> db.users.updateMany({_id : 1}, {$rename : {name : "nama"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Keterangan :

- Data adalah sebuah nama data yang ada pada suatu tabel
- Data-baru adalah sebuah nama baru untuk merubah nama data awal

10. \$Upsert

Operator upsert adalah sebuah operator yang digunakan untuk menambahkan suatu data jika data yang terupdate tidak di temukan, misalkan pada update disitu ada sebuah kriteria jika kriterianya itu tidak sesuai maka data tersebut akan disimpan secara otomatis. Untuk rumus upsert adalah seperti ini :

```
db.tabel.updateMany({kriteria}, {$set : {data : "isi"} }, { upsert : true } )
```

```
rentalfilm> db.users.updateOne({_id : 10}, {nama : "Dendol", dob: ISODate()}}, {upsert: true})
```

```
rentalfilm> db.users.updateOne({_id : 10}, {$set : {nama : "Dendol", dob: ISODate()}}, {upsert: true})
{
  acknowledged: true,
  insertedId: 10,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

11. \$ElemMatch

Operator elemMatch adalah sebuah operator yang digunakan untuk mencari data pada suatu data array. Misalkan kita mempunyai data array 82,85,88 dan data kedua 75, 88, 89 lalu kita ingin mencari data yang lebih dari 80 dan kurang dari 85 kita disini bisa menggunakan operator elemMatch. Untuk perintah elemMatch seperti ini:

```
db.tabel.find( { data : { $elemMatch : { $gt : 80, $lt : 85 } } }).pretty()
```

```
db.Elem.find( { results : { $elemMatch : { $gte : 80, $lt : 85 } } })
```

```
testing> db.Elem.find(
...   { results: { $elemMatch: { $gte: 80, $lt: 85 } } }
... )
[ { _id: 1, results: [ 82, 85, 88 ] } ]
testing> ■
```

Keterangan :

Diatas adalah rumus untuk mencari data array yang datanya lebih dari 80 dan kurang dari 85 pada data sebelumnya kan 82,85,88 dan data kedua 75, 88, 89 untuk 75 dan 89 tidak masuk di kriteria tetapi 90 masuk ke dalam kriteria sehinggal data array ini bisa dapat di tampilkan.

12. \$Push

Operator push adalah sebuah operator yang digunakan untuk menambahkan data pada sebuah data array. Misalkan kita mempunyai data array 70,80,90 lalu kita ingin

menambahkannya menjadi 70,80,90,100 maka kita bisa menggunakan operator push.

Untuk perintah push 1 data array seperti ini :

```
db.tabel.updateOne({kriteria}, {$push : {data : isi} } )
```

```
db.Elem.updateOne({_id : 1}, {$push : {results : 100} } )
```

```
testing> db.Elem.updateOne({_id : 1}, {$push : {results: 100}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
testing>
```

Untuk menambahkan lebih dari 1 data array, seperti ini:

```
db.tabel.updateOne({kriteria}, {$push : {data : { $each : [isi,isi,isi]}}} } )
```

```
db.Elem.updateOne({_id : 1}, {$push : {results : { $each : [ 160, 1600]}}} } )
```

```
testing> db.Elem.updateOne({_id : 1}, {$push : {results: {$each : [ 160, 1600]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
testing>
```

13. \$Pull

Operator pull adalah sebuah operator yang digunakan untuk menghapus data pada sebuah data array. Misalkan kita mempunyai data array 70,80,90 lalu kita ingin menghapus salah satunya menjadi 70,80 maka kita bisa menggunakan operator pull. Untuk perintah dasar pull seperti ini :

```
db.tabel.updateOne({kriteria}, {$pull : {data : isi-yang-dihapus} } )
```

```
db.Elem.updateOne({_id : 1}, {$pull : {results : 1600} } )
```

```
testing> db.Elem.updateOne({_id : 1 }, {$pull : { results : 1600}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
testing>
```


14. \$addToSet

Operator addToSet adalah sebuah operator yang digunakan untuk menambahkan data pada sebuah data array tetapi jika datanya sudah ada pada array tersebut maka datanya tidak akan ditambahkan. Untuk perintah dasar addToSet seperti ini :

```
db.tabel.updateOne({kriteria}, {$addToSet : {data : isi} } )
```

```
db.Elem.updateOne({_id : 1}, {$addToSet : {results : 160} } )
```

```
testing> db.Elem.updateOne({_id : 1 }, {$addToSet : { results : 160}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
testing> _
```

5.5. Tugas dan Latihan

1. Buat dan jalankan perintah untuk mencari data dari collection Customer menggunakan minimal 2 buah variasi dari perintah \$in dan \$nin
2. Buatlah dan Jalankan perintah untuk mencari data dari collection Film untuk menampilkan data Title yang diakhiri dengan huruf G.
3. Buatlah dan jalankan perintah untuk menampilkan 2 buah data dari collection Customer.
4. Buatlah dan jalankan perintah untuk menampilkan semua data di film berdasarkan urutan alfabet.
5. Buatlah collection baru dengan nama ArrTest masukan beberapa data yang memiliki setidaknya 1 array kemudian buat dan jalankan perintah Push, Pull dan addToSet secara bergiliran.