



# MongoDB Dasar

Eko Kurniawan Khannedy



# License

- Dokumen ini boleh Anda gunakan atau ubah untuk keperluan non komersial
- Tapi Anda wajib mencantumkan sumber dan pemilik dokumen ini
- Untuk keperluan komersial, silahkan hubungi pemilik dokumen ini

# Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- [youtube.com/c/ProgrammerZamanNow](https://youtube.com/c/ProgrammerZamanNow)



---

# Pengenalan MongoDB



# Pengenalan MongoDB

- MongoDB merupakan free dan opensource database management system
- MongoDB merupakan database management system berbasis document
- Dikembangkan oleh perusahaan bernama 10gen tahun 2007
- Dirilis ke public tahun 2009
- Saat ini perusahaan 10gen sudah berganti nama menjadi MongoDB Inc
- MongoDB hampir mendukung semua bahasa pemrograman sebagai client nya
- MongoDB tidak menggunakan SQL, namun menggunakan JavaScript sebagai bahasa utama untuk manipulasi document
- <https://github.com/mongodb/mongo>

# db-engines.com/en/ranking/document+store

☐ include secondary database models

47 systems in ranking, May 2020

Rank			DBMS	Database Model	Score		
May 2020	Apr 2020	May 2019			May 2020	Apr 2020	May 2019
1.	1.	1.	MongoDB	Document, Multi-model	438.99	+0.57	+30.92
2.	2.	2.	Amazon DynamoDB	Multi-model	64.72	+0.45	+8.78
3.	3.	4.	Microsoft Azure Cosmos DB	Multi-model	30.68	-1.37	+3.08
4.	4.	3.	Couchbase	Document, Multi-model	28.58	-1.83	-6.09
5.	5.	5.	CouchDB	Document	16.92	-0.85	-2.19
6.	6.	7.	Firebase Realtime Database	Document	13.12	+0.47	+1.84
7.	7.	6.	MarkLogic	Multi-model	10.96	-0.30	-3.09
8.	8.	8.	Realm	Document	8.38	-0.16	+0.74
9.	9.	10.	Google Cloud Firestore	Document	6.34	-0.27	+1.35
10.	10.	11.	ArangoDB	Multi-model	4.68	-0.20	-0.11



# Apa itu Document Oriented Database

- Document oriented database merupakan sistem database yang digunakan untuk memanipulasi data dalam bentuk document (semi structured data)
- Biasanya document disimpan dalam bentuk JSON atau XML
- Document oriented database biasanya bertolak belakang dengan relational database.
- Relational database biasanya menyimpan data dalam bentuk table, dan menyimpan relasinya di table lain.
- Document oriented database biasanya menyimpan data dalam bentuk JSON atau XML, dan menyimpan relasinya sebagai embedded object di dalam document yang sama.



## Istilah Relational DB vs Document DB

Relational DB	Document DB (MongoDB)
Database	Database
Table	Collection
Column	Field
Row, Record	Document (JSON, XML, dan lain-lain)
Join Table	Embedded Document, Reference
SQL	JavaScript (MongoDB)



---

# Menginstall MongoDB



# Menginstall MongoDB

- Linux : <https://docs.mongodb.com/manual/administration/install-on-linux/>
- Mac : <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>
- Windows : <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>



# Menginstall MongoDB Menggunakan Docker

- MongoDB Docker Image : [https://hub.docker.com/\\_/mongo](https://hub.docker.com/_/mongo)
- Docker Compose :  
<https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/mongodb/docker-compose.yml>

---

# MongoDB Client



# Mongo Shell

- MongoDB menyediakan aplikasi mongo client berupa command line interface untuk terkoneksi ke MongoDB Server dengan nama mongo shell.
- Mongo shell sangat bermanfaat saat kita tidak harus konek ke mongo server tanpa GUI
- Mongo shell menggunakan bahasa pemrograman JavaScript



# Menggunakan Mongo Shell

```
mongo --host localhost --port 27017
```



# MongoDB GUI Client

- Jika kita terbiasa menggunakan GUI, ada beberapa aplikasi yang bisa kita gunakan sebagai mongo client, seperti :
  - MongoDB Compass : <https://www.mongodb.com/products/compass>
  - JetBrains DataGrip : <https://www.jetbrains.com/datagrip/>
  - MongoDB for Visual Studio Code :  
<https://marketplace.visualstudio.com/items?itemName=mongodb.mongodb-vscode>
  - Robo 3T : <https://robomongo.org/>

---

# Database





# Database

- Database adalah tempat menyimpan collection
- Semua collection harus disimpan di database
- Biasanya database digunakan untuk memisahkan data secara logical per aplikasi, artinya biasanya satu aplikasi akan memiliki satu database
- Jarang sekali kita akan menggunakan satu database untuk beberapa aplikasi



# Membuat Database

- Kita tidak perlu secara eksplisit membuat database
- MongoDB akan secara otomatis membuatkan database sesuai dengan nama database yang kita pilih
- Untuk memilih nama database, kita bisa menggunakan perintah “use” diikuti nama database



# Memilih Database

```
> use belajar
switched to db belajar
> use tutorial
switched to db tutorial
> show databases
admin    0.000GB
config  0.000GB
local    0.000GB
test     0.000GB
> █
```



## Database Methods

Database Methods	Keterangan
db.dropDatabase()	Menghapus database
db.getName()	Mengambil nama database
db.hostInfo()	Mengambil informasi host tempat mongodb
db.version()	Mengambil versi database
db.stats()	Mengambil statistik penggunaan database

---

# Collection



# Collection

- Collection adalah tempat menyimpan document
- Maximum per document yang bisa disimpan adalah 16MB
- Maximum level nested document yang bisa disimpan adalah 100 level



## Database Methods untuk Collection

Database Methods untuk Collection	Keterangan
<code>db.getCollectionNames()</code>	Mengambil semua nama collection
<code>db.createCollection(name)</code>	Membuat collection baru
<code>db.getCollection(name)</code>	Mendapatkan object collection
<code>db.&lt;name&gt;</code>	Sama dengan <code>db.getCollection(&lt;name&gt;)</code>
<code>db.getCollectionInfos()</code>	Mendapat informasi semua collection



## Collection Methods

Database Methods untuk Collection	Keterangan
db.<collection>.find()	Mengambil semua document
db.<collection>.count()	Mengambil jumlah document
db.<collection>.drop()	Menghapus collection
db.<collection>.totalSize()	Mengambil total ukuran collection
db.<collection>.stats()	Mengambil informasi statistik collection





# Kode Program : Collection

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/collection.js>

---

# Data Model



# Kenapa Perlu Mengerti Data Modeling

- Pindah dari relational database ke document database bukanlah hal yang sederhana hanya dengan memindahkan semua table ke collection
- Penggunaan document database tidak akan mendatangkan manfaat besar jika kita tidak mengerti cara memodelkan data untuk kebutuhan aplikasi kita
- Saat memodelkan data menggunakan relational database, biasanya kita mengacu ke database normalization
- Saat memodelkan data menggunakan document database, kita harus mengacu ke penggunaan aplikasi dalam melakukan query, update dan memproses data



# Schema yang Flexible

- Tidak seperti di relational database, di MongoDB kita bisa memasukkan data ke collection secara langsung tanpa mendefinisikan schema collection nya.
- Schema untuk collection di MongoDB sangat flexible, tiap document bisa berbeda. Tidak seperti table di relational database yang harus sama tiap record.
- Namun pada prakteknya, sangat direkomendasikan menggunakan jenis data yang sama untuk tiap collection, walaupun bisa berbeda-beda di collection yang sama



# Primary Key

- Saat membuat dokumen di MongoDB, kita wajib menambahkan primary key
- Tidak seperti relational database yang bebas membuat column untuk primary key, di MongoDB, primary key wajib menggunakan field `_id`
- Selain itu primary key tidak bisa lebih dari 1 field, hanya bisa field `_id`, jadi jika kita ingin membuat composite primary key, maka kita hanya bisa melakukan dengan menggunakan 1 field `_id`



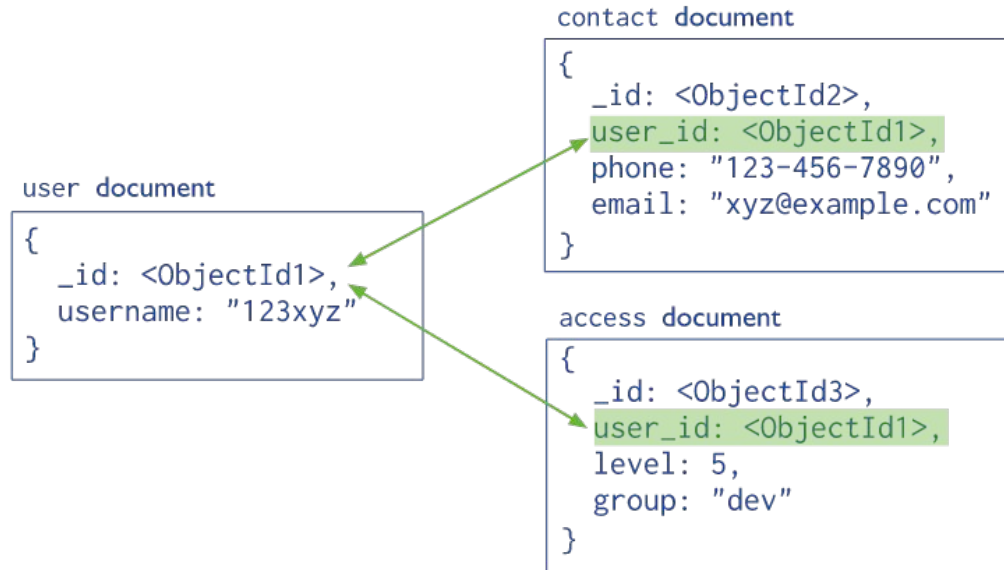
## Struktur Dokumenten - Embedded

```
{
  _id: <ObjectId>,
  username: "123xyz",
  contact: {
    phone: "123-456-7890",
    email: "xyz@example.com"
  },
  access: {
    level: 5,
    group: "dev"
  }
}
```

Embedded sub-  
document

Embedded sub-  
document

# Struktur Dokumenten - Reference





# Embedded vs Reference

Gunakan Embedded jika :

- Antar document saling ketergantungan
- Tidak bisa langsung melakukan perubahan ke embedded document
- Embedded document selalu dibutuhkan ketika mengambil data document

Gunakan Reference jika :

- Antar document bisa berdiri sendiri dan tidak terlalu ketergantungan satu sama lain
- Bisa melakukan manipulasi data langsung terhadap reference document
- Reference document tidak selalu dibutuhkan saat mengambil document



---

**BSO**N



# BSON

- BSON singkatan dari Binary JSON, yaitu binary-encoded serialization dokumen seperti JSON
- Sama seperti JSON, di BSON juga bisa kita bisa menggunakan embedded object, array dan lain-lain
- <http://bsonspec.org/>
- <https://docs.mongodb.com/manual/reference/bson-types/>



## Type Data di BSON (1)

Type Data	Alias
Double	double
String	string
Object	object
Array	arrat
Binary Data	binData
ObjectId	objectId



## Type Data di BSON (2)

Type Data	Alias
Boolean	bool
Date	date
Null	null
Regular Expression	regex
JavaScript	javascript
JavaScript with Scope	javascriptWithScope



## Tipe Data di BSON (3)

Tipe Data	Alias
32 Bit Integer	int
Timestamp	timestamp
64 Bit Integer	long
Decimal 128	decimal
Min Key	minKey
Max key	maxKey



# ObjectId

- ObjectId adalah random data yang unik, cepat untuk digenerate dan terurut.
- Nilai ObjectId memiliki ukuran panjang 12 byte, konsisten terdiri dari informasi 4 byte timestamp, 5 byte random value, dan 3 byte incrementing counter
- ObjectId digunakan sebagai default `_id` (primary key) di document jika kita tidak secara eksplisit menyebutkan `_id` document nya



# Date dan ISODate

- BSON Date adalah 64 bit integer yang merepresentasikan angka milisecond sejak Unix epoch (1 Januari 1970).
- Nilai ini bisa merepresentasikan waktu dengan jarak 290 juta tahun sebelum dan setelah unix epoch.
- ISODate merupakan representasi waktu yang digunakan oleh MongoDB
- Date ini kompatibel dengan Date di JavaScript
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date)

---

# Insert Document





# Insert Document

- Untuk menyimpan data ke MongoDB, kita perlu membuat document dalam bentuk JSON
- Field `_id` tidak wajib dimasukkan, jika kita tidak memasukkan field `_id`, maka secara otomatis MongoDB akan membuat `_id` baru secara random dengan tipe data `ObjectId`
- Atau kita juga bisa secara eksplisit membuat `ObjectId` baru dengan menggunakan perintah “new `ObjectId()`”



# Insert Document Function

Function	Keterangan
<code>db.&lt;collection&gt;.insertOne(document)</code>	Menambah dokumen ke collection
<code>db.&lt;collection&gt;.insertMany(array&lt;document&gt;)</code>	Menambah semua dokumen di array ke collection
<code>db.&lt;collection&gt;.insert(document / array)</code>	Menambah satu document atau banyak dokumen



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/insert.js>

---

# Query Document



# Query Document

- Sama seperti di relational database, di MongoDB pun kita bisa melakukan query atau pencarian document yang sudah kita simpan di collection



# Query Document Function

Function	Keterangan
<code>db.&lt;collection&gt;.find(query)</code>	Mencari document dengan query



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query.js>

---

# Comparison Query Operator





## Comparison Operator (1)

Operator	Keterangan
\$eq	Membandingkan value dengan value lain
\$gt	Membandingkan value lebih besar dari value lain
\$gte	Membandingkan value lebih besar atau sama dengan value lain
\$lt	Membandingkan value lebih kecil dari value lain
\$lte	Membandingkan value lebih kecil atau sama dengan value lain



## Comparison Operator (2)

Operator	Keterangan
\$in	Membandingkan value dengan value yang ada di array
\$nin	Membandingkan value tidak ada dalam value yang ada di array
\$ne	Membandingkan value tidak sama dengan value lain



# Syntax Comparison Operator

```
1
2
3
4 db.customer.find({
5   |   field: {
6   |     |   $operator: "value"
7   |     |   }
8   |   })
9
10
11
12
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-comparison.js>

---

# Logical Query Operator



## Logical Operator

Operator	Keterangan
\$and	Menggabungkan query dengan operasi AND, mengembalikan document jika semua kondisi benar
\$or	Menggabungkan query dengan operasi OR, mengembalikan document jika salah satu kondisi benar
\$nor	Menggabungkan query dengan operasi NOR, mengembalikan document yang gagal di semua kondisi
\$not	Membalikkan kondisi, mengembalikan document yang tidak sesuai kondisi



## Syntax Logical Operator (1)

```
4  // $and, $or, $nor
5  db.collection.find({
6      $operator : [
7          {
8              // expression
9          },
10         {
11             // expression
12         }
13     ]
14 })
```



## Syntax Logical Operator (2)

```
26
27 // $not
28 db.collection.find({
29     field: {
30         $not: {
31             // operator expression
32         }
33     }
34 })
```





# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-logical.js>

---

# Element Query Operator



## Element Operator

Operator	Keterangan
\$exists	Mencocokkan document yang memiliki field tersebut
\$type	Mencocokkan document yang memiliki type field tersebut



## Syntax Element Operator

```
7
8
9
10 db.collection.find({
11     field: {
12         $operator: value
13     }
14 })
15
16
17
18
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-element.js>

---

# Evaluation Query Operator



## Evaluation Operator

Operator	Keterangan
\$expr	Menggunakan aggregation operation
\$jsonSchema	Validasi document sesuai dengan JSON schema
\$mod	Melakukan operasi modulo
\$regex	Mengambil document sesuai dengan regular expression (PCRE)
\$text	Melakukan pencarian menggunakan text
\$where	Mengambil document dengan JavaScript Function



## Syntax \$expr Operator

```
17
18
19 db.collection.find({
20     $expr: {
21         // aggregation expression
22     }
23 })
24
25
26
27
```





## Syntax \$jsonSchema Operator

```
17
18
19 db.collection.find({
20     $jsonSchema: {
21         // JSON Schema Object
22     }
23 })
24
25
26
27
28
```



## Syntax \$mod Operator

```
39
40
41 db.collection.find({
42     field: {
43         $mod: [ divisor, remainder ]
44     }
45 })
46
47
48
49
50
```



## Syntax \$regex Operator

```
64
65
66 db.collection.find({
67     field: {
68         $regex: /regex/,
69         $options: "<option>"
70     }
71 })
72
73
74
```



## Syntax \$text Operator

```
65
66 db.collection.find({
67     $text: {
68         $search: "string",
69         $language: "string",
70         $caseSensitive: "boolean",
71         $diacriticSensitive: "boolean"
72     }
73 })
74
75
76
```



## Syntax \$where Operator

```
84 |
85 |
86 | db.collection.find({
87 |     $where: function(){
88 |         return true;
89 |     }
90 | });
91 |
92 |
93 |
94 |
95 |
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-evaluation.js>

---

# Array Query Operator



## Array Operator

Operator	Keterangan
\$all	Mencocokkan array yang mengandung elemen-elemen tertentu
\$elemMatch	Mengambil document jika tiap element di array memenuhi kondisi tertentu
\$size	Mengambil document jika ukuran array sesuai





## Syntax \$all Operator

```
40
41
42 db.collection.find({
43     field: {
44         $all: ["value"]
45     }
46 })
```



## Syntax \$elemMatch Operator

```
48
49 db.collection.find({
50     fields: {
51         $elemMatch: {
52             // query1,
53             // query2
54         }
55     }
56 })
57
58
```



## Syntax \$size Operator

```
48
49 db.collection.find({
50     fields: {
51         $size: 1 // length
52     }
53 })
54
55
56
57
58
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-array.js>

---

# Projection Operator



# Projection

- Pada function find, terdapat parameter kedua setelah query, yaitu projection
- Projection adalah memilih field mana yang ingin kita ambil atau hide
- `db.<collection>.find(query, projection)`



# Syntax Projection

```
8
9
10 db.collection.find({
11     // query
12 }, {
13     field1: 1, // include,
14     field2: 0 // hide
15 })
16
17
18
```



## Projection Operator

Operator	Keterangan
\$	Limit array hanya mengembalikan data pertama yang match dengan array operator
\$elemMatch	Limit array hanya mengembalikan data pertama yang match dengan kondisi query
\$meta	Mengembalikan informasi metadata yang didapat dari setiap matching document
\$slice	Mengontrol jumlah data yang ditampilkan pada array





## Syntax \$ Operator

```
38
39 db.products.find({
40     field: {
41         $elemMatch: {
42             // query
43         }
44     }
45 }, {
46     "field.$": 1
47 });
48
49
```



## Syntax \$elemMatch Operator

```
38
39 db.collection.find({}, {
40     field: {
41         $elemMatch: {
42             // query
43         }
44     }
45 })
```



## Syntax \$meta Operator

```
50
51 db.collection.find({
52     $text: {
53         $search: "query"
54     }
55 }, {
56     score: {
57         $meta: "textScore"
58     }
59 })
60
61
```



## Syntax \$slice Operator

```
56
57
58 db.products.find({
59     // query
60 }, {
61     field: {
62         $slice: 2 // slice size
63     }
64 });
65
66
67
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-projection.js>

---

# Query Modifier



# Query Modifier

- Query Modifier adalah memodifikasi hasil query yang telah kita lakukan
- Contoh yang sering kita lakukan seperti, mengubah query menjadi jumlah data, membatasi jumlah data dengan paging, dan lain-lain
- Untuk memodifikasi hasil query, kita bisa menambahkan function query modifier setelah menggunakan function find



## Query Modifier Function

Operator	Keterangan
count()	Mengambil jumlah data hasil query
limit(size)	Membatasi jumlah data yang didapat dari query
skip(size)	Menghiraukan data pertama hasil query sejumlah yang ditentukan
sort(query)	Mengurutkan hasil data query





# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/query-modifier.js>

---

# Update Document



# Update Document

- Sama seperti database lainnya, di MongoDB juga kita bisa mengubah document yang sudah kita insert ke collection
- Namun berbeda dengan perintah SQL, di MongoDB, untuk mengubah document, kita diberikan beberapa function
- Untuk update document, kita bisa menggunakan collection : `db.<collection>.<updateFunction>()`



## Update Document Function

Operator	Keterangan
updateOne()	Mengubah satu document
updateMany()	Mengubah banyak document sekaligus
replaceOne()	Mengubah total satu document dengan document baru



## Syntax updateOne() Function

```
1
2
3
4 db.collection.updateOne(
5     {}, // filter
6     {}, // update
7     {} // option
8 )
9
10
11
12
```



## Syntax updateMany() Function

```
1
2
3
4 db.collection.updateMany(
5     {}, // filter
6     {}, // update
7     {} // option
8 )
9
10
11
12
```



## Syntax replaceOne() Function

```
1
2
3
4 db.collection.replaceOne(
5     {}, // filter
6     {}, // replacement
7     {} // option
8 )
9
10
11
12
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/update.js>



---

# Field Update Operator



# Field Update Operator

- Sebelumnya kita sudah tau kalo untuk update document di MongoDB kita bisa menggunakan operator \$set dan \$unset
- Namun sebenarnya masih banyak operator yang bisa kita gunakan



## Update Document Function

Operator	Keterangan
\$set	Mengubah nilai field di document
\$unset	Menghapus field di document
\$rename	Mengubah nama field di document
\$inc	Menaikan nilai number di field sesuai dengan jumlah tertentu
\$currentDate	Mengubah field menjadi waktu saat ini



## Syntax \$set Operator

```
6
7
8  db.collection.update({
9      // query
10 }, {
11     $set: {
12         field1: "value",
13         field2: "value"
14     }
15 })
16
17
```



## Syntax \$unset Operator

```
6
7
8 db.collection.update({
9     // query
10 }, {
11     $unset: {
12         field1: "",
13         field2: ""
14     }
15 })
16
17
```



## Syntax \$rename Operator

```
6
7
8  db.collection.update({
9      // query
10 }, {
11     $rename: {
12         field1: "newName1",
13         field2: "newName2"
14     }
15 })
16
17
```



## Syntax \$incr Operator

```
24
25 db.collection.updateMany({
26     // query
27 }, {
28     $inc: {
29         field: 1, // increment
30         field: -1 // decrement
31     }
32 });
33
34
```



## Syntax \$currentDate Operator

```
8  db.collection.update({
9    // query
10 }, {
11   $currentDate: {
12     field1: {
13       $type: "date"
14     },
15     field2: {
16       $type: "timestamp"
17     }
18   },
```





# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/update-field.js>

---

# Array Update Operator



# Array Update Operator

- Secara default, saat kita mengubah field dengan tipe array, maka seluruh array akan diubah
- Kadang kita ingin menambah, atau hanya mengubah data array tanpa harus mengubah seluruh field array
- Hal ini bisa dilakukan di MongoDB



## Array Update Operator (1)

Operator	Keterangan
\$	Mengupdate data array pertama sesuai kondisi query
\$[]	Mengupdate semua data array sesuai kondisi query
\$[<identifier>]	Mengupdate semua data array yang sesuai kondisi arrayFilters
<index>	Mengupdate data array sesuai dengan nomor index



## Syntax \$ Operator

```
54
55
56 db.collection.updateMany({
57   field: "value"
58 }, {
59   $operator: {
60     "field.$": "value"
61   }
62 })
```



## Syntax \$[] Operator

```
54
55
56 db.collection.updateMany({
57     // query
58 }, {
59     $operator: {
60         "field.$[]": "value"
61     }
62 })
63
64
65
```



## Syntax \$[<identifier>] Operator

```
56 db.collection.updateMany({
57   // query
58 }, {
59   $operator: {
60     "field.$[element]" : "value"
61   }
62 }, {
63   arrayFilters: [
64     {
65       element: {
66         $operator: "value"
67       }
68     }
69   ]
70 })
```



## Syntax <index> Operator

```
54
55
56 db.collection.updateMany({
57     // query
58 }, {
59     $operator: {
60         "field.<index>": "value"
61     }
62 });
63
64
65
```





## Array Update Operator (2)

Operator	Keterangan
\$addToSet	Menambahkan value ke array, dihiraukan jika sudah ada
\$pop	Menghapus element pertama (-1) atau terakhir (1) array
\$pull	Menghapus semua element di array yang sesuai kondisi
\$push	Menambahkan element ke array
\$pullAll	Menghapus semua element di array



## Syntax \$addToSet Operator

```
69
70
71
72 db.products.updateOne({
73     // query
74 }, {
75     $addToSet: {
76         field: "value"
77     }
78 });
79
80
```



## Syntax \$pop Operator

```
69
70
71
72 db.products.updateMany({
73     // query
74 }, {
75     $pop: {
76         ratings: -1
77     }
78 });
79
80
```



## Syntax \$pull Operator

```
87
88 db.products.updateMany({
89     // query
90 }, {
91     $pull: {
92         field: {
93             $operator: "value"
94         }
95     }
96 })
97
98
```



## Syntax \$push Operator

```
95
96 db.products.updateMany({
97   // query
98 }, {
99   $push: {
100     field: "value"
101   }
102 })
103
104
105
```



## Syntax \$pullAll Operator

```
99
100
101 db.products.updateMany({
102     // query
103 }, {
104     $pullAll: {
105         field: ["value"]
106     }
107 })
108
109
110
```



## Array Update Operator Modifier

Operator	Keterangan
\$each	Digunakan di \$addToSet dan \$push, untuk menambahkan multiple element
\$position	Digunakan di \$push untuk mengubah posisi menambahkan data
\$slice	Digunakan di \$push untuk menentukan jumlah maksimal data array
\$sort	Digunakan untuk mengurutkan array setelah operasi \$push



## Syntax \$each Operator

```
120
121 db.products.updateMany({
122     // query
123 }, {
124     $operator: {
125         field: {
126             $each: ["value1", "value2"]
127         }
128     }
129 })
130
131
```





## Syntax \$position Operator

```
138
139 db.products.updateMany({
140     // query
141 }, {
142     $push: {
143         field: {
144             $each: ["value"],
145             $position: 1
146         }
147     }
148 })
149
```



## Syntax \$slice Operator

```
128
129 db.products.updateMany({
130     // query
131 }, {
132     $operator: {
133         field: {
134             $each: ["value", "value"],
135             $slice: 1
136         }
137     }
138 })
139
```



## Syntax \$sort Operator

```
148
149 db.products.updateMany({
150     // query
151 }, {
152     $push: {
153         field: {
154             $each: ["value"],
155             $sort: 1
156         }
157     }
158 })
159
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/update-array.js>

---

**Delete Document**



# Delete Document

- MongoDB memiliki function yang bisa kita gunakan untuk menghapus document di collection secara permanen
- Document yang sudah kita hapus, tidak akan bisa dikembalikan lagi



## Delete Document Function

Function	Keterangan
<code>db.&lt;collection&gt;.deleteOne(query)</code>	Menghapus satu document yang sesuai dengan kondisi query
<code>db.&lt;collection&gt;.deleteMany(query)</code>	Menghapus banyak document yang sesuai dengan kondisi query



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/delete.js>



---

# Bulk Write Operations



# Bulk Write Operation

- Komunikasi antara aplikasi dengan database biasanya dilakukan secara request-response
- Artinya tiap perintah yang ingin kita kirimkan dari aplikasi ke database, akan diresponse secara langsung oleh database
- Proses tersebut akan sangat lambat, jika kita menghadapi kasus harus memanipulasi data besar secara langsung. Misal upload data dari file dengan jumlah jutaan ke dalam database.
- MongoDB mendukung Bulk Write Operation, yaitu operasi bulk yang dalam satu request, kita bisa mengirim banyak perintah
- Fitur ini cocok pada kasus jika kita ingin melakukan operasi bulk atau batch secara banyak sekaligus



## Bulk Write Function

Function	Keterangan
<code>db.&lt;collection&gt;.insertMany()</code>	Insert document secara banyak sekaligus
<code>db.&lt;collection&gt;.updateMany()</code>	Update document secara banyak sekaligus
<code>db.&lt;collection&gt;.deleteMany()</code>	Delete document secara banyak sekaligus
<code>db.&lt;collection&gt;.bulkWrite()</code>	Melakukan operasi write (insert, update, delete) banyak secara sekaligus



# Supported Bulk Write Operation

- insertOne
- updateOne
- updateMany
- replaceOne
- deleteOne
- deleteMany



## Syntax bulkWrite() Function

```
5 db.customers.bulkWrite([
6     {
7         // operation 1
8     },
9     {
10        // operation 2
11    },
12    {
13        // operation n
14    }
15 ])
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/bulk-write.js>

---

# Schema Validation



# Schema Validation

- Pada Relational DB, kita biasanya menambahkan constraint terhadap data yang ada di tabel
- Misal, maksimal karakter, Enum string, Not Null, dan lain-lain
- Di MongoDB, fitur untuk validasi data lebih canggih dibanding constraint di Relational DB
- MongoDB mendukung Schema Validation menggunakan JSON Schema





# JSON Schema

- JSON Schema adalah standar resmi untuk memvalidasi data JSON
- Dengan menggunakan JSON Schema, kita bisa memberi batasan, data JSON apa yang valid, sehingga bisa dimasukkan ke dalam collection
- <http://json-schema.org/>



## Syntax Create Collection dengan Validator

```
4
5
6 db.createCollection("collection", {
7     validator: {
8         $jsonSchema: {
9             // json schema
10         }
11     }
12 })
13
14
15
```



## Syntax Update Collection dengan Validator

```
99
100 db.runCommand({
101     collMod: "collection",
102     validationAction: "error",
103     validator: {
104         $jsonSchema: {
105             // json schema
106         }
107     }
108 })
109
110
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/schema-validation.js>

---

# Indexes



# Indexes

- Index adalah fitur di MongoDB untuk mengefisienkan proses query. Tanpa Index, MongoDB harus melakukan proses query dengan cara collection scan (mencari keseluruhan data dari awal sampai akhir)
- Jika terdapat Index pada collection MongoDB, MongoDB bisa menggunakan Index untuk mendapatkan data, tanpa harus melakukan pencarian keseluruhan document
- Index adalah struktur data khusus yang menyimpan data dalam struktur yang mudah untuk dicari.
- Index menyimpan spesifik field, lalu mengurutkan data field tersebut. Hal ini tidak hanya mempermudah ketika proses pencarian, namun mempermudah ketika kita butuh melakukan pencarian dalam bentuk range document (seperti paging).
- Secara dasar, Index di MongoDB, cara kerjanya sama seperti Index di Relational DB



## Create Index Function

Function	Keterangan
<code>db.&lt;collection&gt;.createIndex()</code>	Membuat index di collection
<code>db.&lt;collection&gt;.getIndexes()</code>	Melihat semua index di collection
<code>db.&lt;collection&gt;.dropIndex()</code>	Menghapus index di collection



# Single Field Index

- Secara default, field `_id` di MongoDB sudah di index secara otomatis, jadi kita tidak perlu membuat index lagi secara manual untuk field `_id`
- MongoDB mendukung penuh pembuatan index di semua field yang ada di document. Dengan begitu, ini bisa mempercepat proses query di MongoDB untuk query terhadap field tersebut





## Syntax Single Field Index

```
3
4  db.products.createIndex({
5    |   field: 1 // ascending
6  | })
7
8  db.products.createIndex({
9    |   field: -1 // descending
10 | })
11
12
13
```



# Compound Indexes

- Jika kita butuh melakukan query terhadap lebih dari satu field, kita juga bisa membuat index terhadap lebih dari satu field, atau disebut Compound Index
- MongoDB membatasi pembuatan maksimal field yang bisa dibuat di compound index adalah 32 field



## Syntax Compound Field Index

```
46
47
48
49
50 db.collection.createIndex({
51     field1: 1,
52     field2: -1
53 });
54
55
56
57
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/index.js>



# Indexing Strategy

- Buat index untuk mendukung performa query
  - Gunakan single index, jika kita hanya melakukan query terhadap satu field saja
  - Gunakan compound index, jika kita sering melakukan query ke field pertama, atau kombinasi field pertama dan kedua, atau pertama dan kedua dan seterusnya
- Buat index untuk mengurutkan hasil query
- Sering-seringlah menggunakan function explain() untuk mengecek apakah query kita sudah di optimize dengan index atau belum

---

# Text Indexes



# Text Indexes

- MongoDB menyediakan text index untuk mendukung pencarian text di tipe data string.
- Text index tidak hanya bisa digunakan pada field dengan tipe data string, namun juga pada array berisi tipe data string



## Syntax Text Index

```
19
20
21 db.products.createIndex({
22     field1: "text",
23     field2: "text"
24 }, {
25     weights: {
26         field1: 10,
27         field2: 5
28     }
29 });
30
```





# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/index-text.js>

---

# Wildcard Indexes



# Wildcard Indexes

- MongoDB mendukung wildcard index, dimana ini digunakan untuk membuat index terhadap field yang belum diketahui atau field yang sering berubah-ubah
- Misal contoh kita punya sebuah embedded document dengan tipe field customFields, dimana isinya bisa bebas sesuai dengan data yang dimasukkan.
- Agar bisa mendukung proses query yang cepat pada field tersebut, kita bisa menggunakan wildcard index



## Syntax Wildcard Index

```
46
47
48
49 db.customers.createIndex({
50   "field.$**" : 1
51 });
52
53
54
55
56
57
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/index-wildcard.js>

---

# Index Properties



# Index Properties

- MongoDB mendukung properties di index
- Istilah properties di Index mungkin agak sedikit membingungkan, sederhananya adalah menambahkan behaviour atau kemampuan tertentu terhadap index yang kita buat



# TTL Index

- TTL singkatan dari Time To Live, yaitu waktu untuk hidup
- TTL Index digunakan sebagai waktu hidup document di collection, artinya data akan hilang dalam kurun waktu tertentu secara otomatis
- TTL Index hanya dapat digunakan di field dengan tipe data Date
- Background proses di MongoDB akan secara regular melakukan penghapusan data secara otomatis
- Sayangnya, proses background tersebut berjalan setiap 60 detik sekali





## Syntax TTL Index

```
17
18
19
20 db.collection.createIndex({
21   |   field: 1
22   }, {
23   |   expireAfterSeconds: 10
24   })
25
26
27
28
```



# Unique Index

- Unique Index memastikan bahwa field-field di index tersebut tidak menyimpan data duplicate.
- Contohnya, di MongoDB, field `_id` secara otomatis merupakan unique index, sehingga kita tidak bisa membuat document dengan field `_id` yang sama



## Syntax Unique Index

```
35
36
37
38 db.collection.createIndex({
39     field: 1
40 }, {
41     unique: true
42 });
43
44
45
46
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/index-properties.js>

---

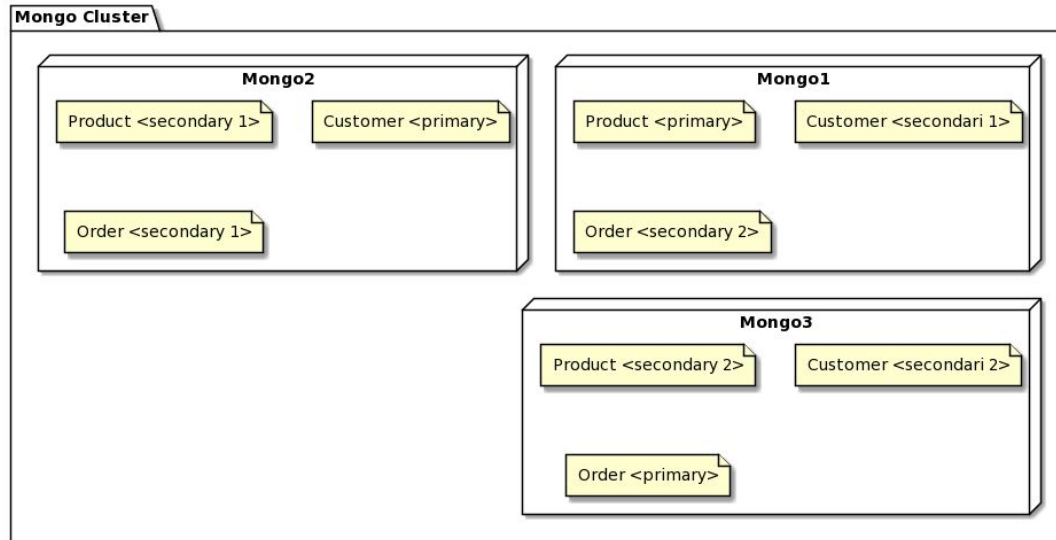
# Transactions



# Transaction

- Di Relational DB, fitur yang sangat berguna dan banyak orang gunakan adalah fitur transaction
- Fitur transaction secara sederhana adalah menggabungkan beberapa operasi database dalam satu transaction, dimana transaction akan dianggap sukses jika semua operasi sukses, dan transaction akan dianggap gagal jika ada salah satu operasi yang gagal
- Dan jika transaction gagal, maka seluruh operasi yang sukses sebelumnya harus dibatalkan (rollback)
- Fitur transaction di MongoDB hanya bisa jalan di cluster (replica-set), tidak di single node
- Dalam cluster, Database di MongoDB akan memiliki primary data dan secondary data

# MongoDB Cluster Replica Set





# Transaction Function

Function	Keterangan
<code>session.startTransaction()</code>	Memulai transaksi
<code>session.commitTransaction()</code>	Commit transaksi
<code>session.abortTransaction()</code>	Membatalkan transaksi





# Read Preferences

Read preferences adalah bagaimana MongoDB mengontrol dari mana kita membaca data

- primary : Semua query diambil dari primary replica set
- primaryPreferred : Semua query diambil dari primary replica set, namun jika tidak ada primary replica set, maka diambil dari secondary replica set
- secondary: Semua query diambil dari secondary replica set
- secondaryPreferred : Semua query diambil dari secondary replica set, namun jika tidak ada secondary replica set, maka diambil dari primary replica set
- nearest : Semua query diambil dari replica set paling murah network latency nya



# Read Concern

Read Concern adalah bagaimana MongoDB mengontrol data yang kita dapatkan

- local: Data akan didapatkan di local node
- available : Data akan didapatkan dimanapun (tidak peduli node mana)
- majority : Data akan didapatkan di mayoritas data di semua node
- linearizable : Data akan dipastikan data paling terbaru di semua node
- snapshot : Data akan diambil dari mayoritas data snapshot (data yang telah di commit) di semua node



# Write Concern

Write Concern adalah bagaimana MongoDB mengontrol operasi write (insert, update, delete)

- `<number>` : Operasi dianggap sukses jika sudah berhasil melakukan operasi write di node sejumlah `<number>`
- `majority` : Operasi dianggap sukses jika sudah berhasil melakukan operasi write di mayoritas node



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/transaction.js>

---

# Security



# Security

- Secara default, jika kita menjalankan MongoDB, mode yang dijalankan tidaklah aman
- Tidak ada Authentication dan tidak ada Authorization
- Agar aman, kita harus mengaktifkan fitur access control di MongoDB



# User Admin

- User admin harus ada terlebih dahulu sebelum kita mengaktifkan access control
- User admin adalah user yang memiliki role `userAdminAnyDatabase` dan `readWriteAnyDatabase`
- Setelah membuat user admin, kita bisa menjalankan ulang MongoDB dengan perintah `--auth`



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/security.js>



---

# Authentication



# Authentication

- Authentication adalah proses memverifikasi identitas pengguna ketika mengakses MongoDB
- Saat menggunakan authentication, maka client wajib menggunakan username dan password untuk terkoneksi ke MongoDB server
- MongoDB mendukung banyak mekanisme authentication, seperti :
  - SCRAM : <https://tools.ietf.org/html/rfc5802>
  - Certificate Authentication
  - LDAP
  - Kerberos, dan lain-lain



# User

- Di MongoDB, kita bisa menambahkan user, dan juga menambahkan role ke user tersebut
- Saat kita membuat user, kita harus menentukan database sebagai authentication database
- Namun bukan berarti user hanya bisa mengakses database itu saja, tapi user juga bisa mengakses database lain jika mau
- Nama user harus unik per database, namun jika database nya berbeda, kita bisa membuat user dengan nama yang sama



## User Function

Function	Keterangan
db.createUser()	Membuat user
db.getUsers()	Mendapatkan semua user
db.dropUser()	Menghapus user
db.updateUser()	Mengupdate user
db.changeUserPassword()	Mengubah user password



## Syntax User

```
53
54
55 db.createUser(
56     {
57         user: "user",
58         pwd: "password",
59         roles: [
60             {
61                 role: "role",
62                 db: "database"
63             },
64             "other role"
65         ]
66     }
67 )
68
69
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/authentication.js>

---

# Authorization



# Authorization

- Authorization adalah proses yang dilakukan setelah proses Authentication sukses
- Authorization dilakukan untuk melakukan pengecekan apakah user memiliki hak akses untuk melakukan sebuah action
- Hak akses di MongoDB disimpan dalam bentuk role





## Database Roles

Role	Keterangan
read	Bisa membaca data di semua collection yang bukan sistem collection
readWrite	Bisa membaca dan mengubah data di semua collection yang bukan sistem collection
dbAdmin	Bisa melakukan kemampuan administrasi database
userAdmin	Mampu membuat user dan role
dbOwner	Kombinasi readWrite, dbAdmin dan userAdmin



## All Database Roles

Role	Keterangan
readAnyDatabase	Seperti read role, tapi untuk semua database
readWriteAnyDatabase	Seperti readWrite role, tapi untuk semua database
userAdminAnyDatabase	Seperti userAdmin, tapi untuk semua database
dbAdminAnyDatabase	Seperti dbAdmin, tapi untuk semua database



## Backup & Restore Roles

Role	Keterangan
backup	Kemampuan untuk melakukan backup database
restore	Kemampuan untuk melakukan restore database



# Superuser Roles

Role	Keterangan
root	Bisa melakukan apapun



# Privileges

- Role membatasi hak akses di level database
- Kadang kita ingin membatasi di level collection
- Untuk melakukan itu, kita bisa menggunakan privileges



## Role Function

Role	Keterangan
db.createRole()	Membuat role baru
db.getRoles()	Mendapatkan role
db.deleteRole()	Menghapus role
db.updateRole()	Mengubah role



## Syntax Role

```
47 db.createRole({
48     role: "name",
49     privileges: [
50         {
51             resource: {
52                 db: "databaes",
53                 collection: "collection"
54             },
55             actions: [ "action" ]
56         }
57     ],
58     roles: [
59         {
60             role: "role",
61             db: "database"
62         }
63     ]
64 });
65
```



# Kode Program

- <https://github.com/ProgrammerZamanNow/belajar-mongodb/blob/master/scripts/authorization.js>



---

# Materi Selanjutnya



# Materi Selanjutnya

- MongoDB Aggregation
- MongoDB Geospatial
- MongoDB Scalability



# Eko Kurniawan Khannedy

- Telegram : @khannedy
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : <https://t.me/ProgrammerZamanNow>
- Email : echo.khannedy@gmail.com