MongoDB Query

rismayana@poltektedc.ac.id

Run MongoDB

- Run MongoDB Server
  - In command Prompt use syntax : mongod

C:\WINDOWS\system32\cmd.exe - mongod

```
C:\Users\risma>mongod
2019-04-14T20:42:12.584-0700 I CONTROL  [initandlisten] MongoDB starting : pid=1156 port=27017 dbpath=C:\data\db\ 64-bit host=rismayana
2019-04-14T20:42:12.584-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2019-04-14T20:42:12.585-0700 I CONTROL  [initandlisten] db version v3.6.2
2019-04-14T20:42:12.586-0700 I CONTROL  [initandlisten] git version: 489d177dbd0f0420a8ca04d39fd78d0a2c539420
2019-04-14T20:42:12.587-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips  22 Sep 2016
2019-04-14T20:42:12.587-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2019-04-14T20:42:12.588-0700 I CONTROL  [initandlisten] modules: none
2019-04-14T20:42:12.589-0700 I CONTROL  [initandlisten] build environment:
2019-04-14T20:42:12.589-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2019-04-14T20:42:12.590-0700 I CONTROL  [initandlisten]     distarch: x86_64
2019-04-14T20:42:12.591-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2019-04-14T20:42:12.591-0700 I CONTROL  [initandlisten] options: {}
2019-04-14T20:42:12.593-0700 I -        [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage eng
ine to 'wiredTiger'.
2019-04-14T20:42:12.594-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=3542M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_
base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recov
ery_progress),
2019-04-14T20:42:12.911-0700 I STORAGE  [initandlisten] WiredTiger message [1555299732:910911][1156:140734276717648], txn-recover: Main recovery loop: starting at 9/483
840
2019-04-14T20:42:13.201-0700 I STORAGE  [initandlisten] WiredTiger message [1555299733:200741][1156:140734276717648], txn-recover: Recovering log 9 through 10
2019-04-14T20:42:13.361-0700 I STORAGE  [initandlisten] WiredTiger message [1555299733:360222][1156:140734276717648], txn-recover: Recovering log 10 through 10
2019-04-14T20:42:13.544-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.544-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-04-14T20:42:13.546-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2019-04-14T20:42:13.548-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.549-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2019-04-14T20:42:13.550-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2019-04-14T20:42:13.550-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2019-04-14T20:42:13.551-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2019-04-14T20:42:13.552-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2019-04-14T20:42:13.553-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2019-04-14T20:42:13.553-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.554-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.559-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This
 can lead to increased memory pressure and poor performance.
2019-04-14T20:42:13.560-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2019-04-14T20:42:13.561-0700 I CONTROL  [initandlisten]
2019-04-15T10:42:13.917+0700 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2019-04-15T10:42:13.922+0700 I NETWORK  [initandlisten] waiting for connections on port 27017
2019-04-15T10:42:21.275+0700 I NETWORK  [listener] connection accepted from 127.0.0.1:54220 #1 (1 connection now open)
2019-04-15T10:42:21.391+0700 I STORAGE  [conn1] createCollection: dbfilm.payment with generated UUID: a23d6f43-5720-462b-9cee-ab8878362588
2019-04-15T10:42:26.217+0700 I NETWORK  [conn1] end connection 127.0.0.1:54220 (0 connections now open)
2019-04-15T10:42:42.351+0700 I NETWORK  [listener] connection accepted from 127.0.0.1:54228 #2 (1 connection now open)
```

MongoDB Query Editor

- By default, mongoDB such as MySQL, using command prompt to connect to server and using it to query and/or to edit it.
- In command prompt use syntax : mongo

C:\WINDOWS\system32\cmd.exe - mongo

```
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\risma>mongo
MongoDB shell version v3.6.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2019-04-14T20:42:13.544-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.544-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-04-14T20:42:13.546-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2019-04-14T20:42:13.548-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.549-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2019-04-14T20:42:13.550-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this ser
ver.
2019-04-14T20:42:13.550-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify
 which IP
2019-04-14T20:42:13.551-0700 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --
bind_ip_all to
2019-04-14T20:42:13.552-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired,
 start the
2019-04-14T20:42:13.553-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warn
ing.
2019-04-14T20:42:13.553-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.554-0700 I CONTROL  [initandlisten]
2019-04-14T20:42:13.559-0700 I CONTROL  [initandlisten] ** WARNING: The file system cache of this machine is configured
to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2019-04-14T20:42:13.560-0700 I CONTROL  [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
```

Basic Query

- To see how many database was restored in mongoDB server, use syntax : show dbs <enter>
- When we want to enter to database , use syntax : use dbasename <enter>
- To see how many collections in database where we use, we can use syntax : show collections <enter>

```
> show dbs
admin      0.000GB
config     0.000GB
dbfilm     0.010GB
dbmhs      0.000GB
local      0.000GB
penjualan  0.021GB
tokobuku   0.000GB
> use dbfilm
switched to db dbfilm
> show collections
actor
payment
```

The insert() Method

- To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()**method.
- **Syntax**
  - The basic syntax of **insert()** command is as follows

```
>db.COLLECTION_NAME.insert(document)
```

Example

```
>db.mycol.insert({
   _id: ObjectId(7df78ad8902c),
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100
})
```

- Inserting Documents Using Loop
- Documents can also be added to the collection using a for loop. The following code inserts users using for .

```
> for(var i=1;i<=20;i++) db.users.insert({"Name":"test user "+i,"age":10+i,"city":"bandung"})
WriteResult({ "nInserted" : 1 })
> db.users.find().pretty()
{
        "_id" : ObjectId("5cb402466db2ed1cfef6ac9a"),
        "Name" : "test user 1",
        "age" : 11,
        "city" : "bandung"
}
{
        "_id" : ObjectId("5cb402466db2ed1cfef6ac9b"),
        "Name" : "test user 2",
        "age" : 12,
        "city" : "bandung"
}
{
        "_id" : ObjectId("5cb402466db2ed1cfef6ac9c"),
        "Name" : "test user 3",
        "age" : 13,
        "city" : "bandung"
```

RDBMS Where Clause Equivalents in MongoDB

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

AND in MongoDB

- **Syntax**
- In the **find()** method, if you pass multiple keys by separating them by ',' then MongoDB treats it as **AND** condition. Following is the basic syntax of **AND**

```
>db.mycol.find({key1:value1, key2:value2}).pretty()
```

Example

Following example will show all the tutorials written by 'tutorials point' and whose title is 'MongoDB Overview'.

```
>db.mycol.find({"by":"tutorials point","title": "MongoDB Overview"}).pretty()
{
   "_id": ObjectId(7df78ad8902c),
   "title": "MongoDB Overview",
   "description": "MongoDB is no sql database",
   "by": "tutorials point",
   "url": "http://www.tutorialspoint.com",
```

OR in MongoDB

- **Syntax**
- To query documents based on the OR condition, you need to use **$or** keyword. Following is the basic syntax of **OR**

```
>db.mycol.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

Example

Following example will show all the tutorials written by 'tutorials point' or whose title is 'MongoDB Overview'.

```
>db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB
Overview"}]}).pretty()
{
   "_id": ObjectId(7df78ad8902c),
   "title": "MongoDB Overview",
```

Using AND and OR Together

- The following example will show the documents that have likes greater than 100 and whose title is either 'MongoDB Overview' or by is 'tutorials point'. Equivalent SQL where clause is **'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'**

```
>db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"},
   {"title": "MongoDB Overview"}]}).pretty()
{
   "_id": ObjectId(7df78ad8902c),
   "title": "MongoDB Overview",
   "description": "MongoDB is no sql database",
   "by": "tutorials point",
   "url": "http://www.tutorialspoint.com",
   "tags": ["mongodb", "database", "NoSQL"],
   "likes": "100" }
>
```

MongoDB Update() Method

- **Syntax**
- The basic syntax of **update()** method is as follows:

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'.

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB
Tutorial'}})
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

MongoDB Update() Method

- By default, MongoDB will update only a single document. To update multiple documents, you need to set a parameter 'multi' to true.

```
>db.mycol.update({'title':'MongoDB Overview'},
   {$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

MongoDB – Delete Document

- **The remove() Method**
- MongoDB's **remove()** method is used to remove a document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag.
- **deletion criteria**: (Optional) deletion criteria according to documents will be removed.
- **justOne**: (Optional) if set to true or 1, then remove only one document.

MongoDB – Delete Document

- **Syntax**
- Basic syntax of **remove()** method is as follows:

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

Following example will remove all the documents whose title is 'MongoDB Overview'.

```
>db.mycol.remove({'title':'MongoDB Overview'})
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

Remove Only One vs Remove All Documents

- **Remove Only One**
- If there are multiple records and you want to delete only the first record, then set **justOne** parameter in **remove()** method

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

- **Remove All Documents**
- If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. **This is equivalent of SQL's truncate command.**

```
>db.mycol.remove()
>db.mycol.find()
>
```

MongoDB – Projection

- In MongoDB, projection means selecting only the necessary data rather than selecting whole of the data of a document. If a document has 5 fields and you need to show only 3, then select only 3 fields from them
- **The find() Method**
- MongoDB's **find()** method, explained in MongoDB Query Document accepts second optional parameter that is list of fields that you want to retrieve. In MongoDB, when you execute **find()** method, then it displays all fields of a document. To limit this, you need to set a list of fields with value 1 or 0. 1 is used to show the field while 0 is used to hide the fields.
- **Syntax**
- The basic syntax of **find()** method with projection is as follows:

```
>db.COLLECTION_NAME.find({},{KEY:1})
```

Example

Consider the collection mycol has the following data

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

Following example will display the title of the document while querying the document.

```
>db.mycol.find({},{"title":1,_id:0})
{"title":"MongoDB Overview"}
{"title":"NoSQL Overview"}
{"title":"Tutorials Point Overview"}
>
```

Please note **_id** field is always displayed while executing **find()** method, if you don't want this field, then you need to set it as 0.

MongoDB – Limit Records

- **The Limit() Method**
- To limit the records in MongoDB, you need to use **limit()** method. The method accepts one number type argument, which is the number of documents that you want to be displayed.
- **Syntax**
- The basic syntax of **limit()** method is as follows:

```
>db.COLLECTION_NAME.find().limit(NUMBER)
```

Example

Consider the collection myycol has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

Following example will display only two documents while querying the document.

```
>db.mycol.find({},{"title":1,_id:0}).limit(2)
{"title":"MongoDB Overview"}
{"title":"NoSQL Overview"}
>
```

If you don't specify the number argument in **limit()** method then it will display all documents from the collection.

MongoDB – Sort Records

- **The sort() Method**
- To sort documents in MongoDB, you need to use **sort()** method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.
- **Syntax**
- The basic syntax of **sort()** method is as follows:

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

Example

Consider the collection myycol has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

Following example will display the documents sorted by title in the descending order.

```
>db.mycol.find({},{"title":1,_id:0}).sort({"title":-1})
{"title":"Tutorials Point Overview"}
{"title":"NoSQL Overview"}
{"title":"MongoDB Overview"}
>
```

Please note, if you don't specify the sorting preference, then **sort()** method will display the documents in ascending order.

Importing Data To MongoDB

Importing a CSV file in MongoDB (1)

- **Step 1 :** Go to the directory where MongoDB is kept.
- **Step 2 :** Go to its Bin folder.
- **Step 3 :** Save or keep your CSV or TXT file here which you want to import. The image below shows exactly what you need to do.

1	title	description	first_name	last_name	actor_id
2	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist	PENELOPE	GUINESS	1
3	ANACONDA CONFESSIONS	A Lacklusture Display of a Dentist And a Dentist	PENELOPE	GUINESS	1
4	ANGELS LIFE	A Thoughtful Display of a Woman And a Astron	PENELOPE	GUINESS	1
5	BULWORTH COMMANDMENTS	A Amazing Display of a Mad Cow And a Pioneer	PENELOPE	GUINESS	1
6	CHEAPER CLYDE	A Emotional Character Study of a Pioneer And a	PENELOPE	GUINESS	1
7	COLOR PHILADELPHIA	A Thoughtful Panorama of a Car And a Crocodil	PENELOPE	GUINESS	1
8	ELEPHANT TROJAN	A Beautiful Panorama of a Lumberjack And a Fc	PENELOPE	GUINESS	1
9	GLEAMING JAWBREAKER	A Amazing Display of a Composer And a Forens	PENELOPE	GUINESS	1
10	HUMAN GRAFFITI	A Beautiful Reflection of a Womanizer And a Su	PENELOPE	GUINESS	1
11	KING EVOLUTION	A Action-Packed Tale of a Boy And a Lumberjac	PENELOPE	GUINESS	1
12	LADY STAGE	A Beautiful Character Study of a Woman And a	PENELOPE	GUINESS	1
13	LANGUAGE COWBOY	A Epic Yarn of a Cat And a Madman who must V	PENELOPE	GUINESS	1
14	MULHOLLAND BEAST	A Awe-Inspiring Display of a Husband And a Sq	PENELOPE	GUINESS	1
15	OKLAHOMA JUMANJI	A Thoughtful Drama of a Dentist And a Woman	PENELOPE	GUINESS	1
16	RULES HUMAN	A Beautiful Epistle of a Astronaut And a Studen	PENELOPE	GUINESS	1
17	SPLASH GUMP	A Taut Saga of a Crocodile And a Boat who must	PENELOPE	GUINESS	1
18	VERTIGO NORTHWEST	A Unbelieveable Display of a Mad Scientist And	PENELOPE	GUINESS	1

Importing a CSV file in MongoDB (2)

- **Step 4 :** Now into the same folder open the ***cmd*** by pressing **ctrl+L and** typing ***cmd*** there. Press enter and you will see a ***cmd*** window now. (You can open this cmd window at this place by right-clicking the mouse along with shift key pressed – here you will see **open command window here**)
- **Step 5 :** Type the undermentioned command:

```
\bin>mongoimport -d dbfilm -c payment --type CSV --file payment.csv --headerline
```

- Sample

```
C:\Program Files\MongoDB\Server\3.6\bin>mongoimport -d dbfilm -c payment --type CSV --file payment.csv --headerline
2019-04-15T10:42:21.282+0700    connected to: localhost
2019-04-15T10:42:23.265+0700    [#########.............] dbfilm.payment       11.9MB/28.8MB (41.2%)
2019-04-15T10:42:26.205+0700    [########################] dbfilm.payment       28.8MB/28.8MB (100.0%)
2019-04-15T10:42:26.205+0700    imported 87980 documents
```

IMPORTANT NOTE:

- **dbfilm** is the database name
- **payment** is the collection name.
- **–type CSV** denotes that file type that is being imported is of type CSV.
- **payment.csv** is the file name that we are importing.
- **–headerline** is the header of the CSV file, below which listed data is mentioned.