

MONGO DB TUTORIAL

Hands-on Session

by Suchitra Jayaprakash
suchitra@cmi.ac.in

MONGO DB

- Document Oriented Database.
- Data is stored as documents - JSON like syntax - javascript object notation.
- Database : Physical container for collection.
- Collection: Group of MongoDB documents.
- Document: Set of key-value pairs.
- Dynamic schema.

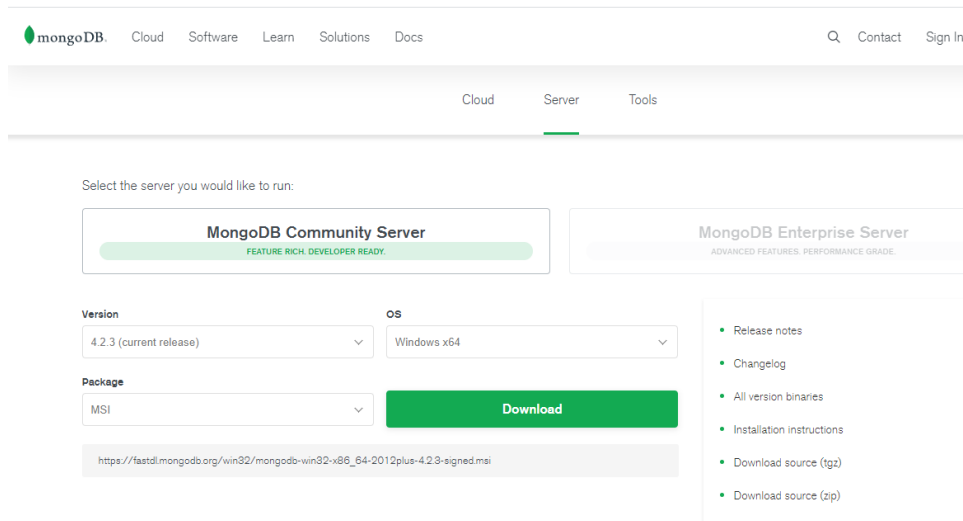
MONGO DB

| RDBMS | MONGODB |
|-------------|-----------------|
| Database | Database |
| Table | Collection |
| Row | Document |
| Column | Field |
| Primary Key | Default key _id |

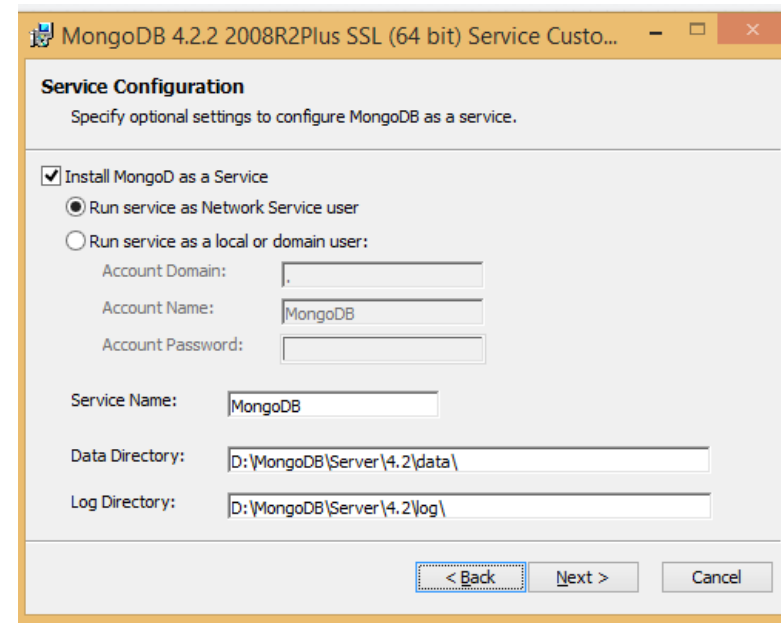
- **Some advantages :**
 - Schema less.
 - Conversion of application objects to database objects is not required.
 - No complex joins

MONGO DB Installation

- Download the latest release of MongoDB from <https://www.mongodb.com/download-center>



- Run installation file
- Select complete install



- Go to " C:\Program Files\MongoDB\Server\4.2\bin"
- Double click on mongo.exe.

MONGO DB COMMANDS

- Create Database

use <dbname>

to know current database , type db

db.stats()

shows database metadata

- Create user

db.createUser({

user:"suchi",

pwd:"123",

roles:["readWrite","dbAdmin"]

})

```
> use myfirst
switched to db myfirst
> db
myfirst
> db.stats()
{
  "db" : "myfirst",
  "collections" : 0,
  "views" : 0,
  "objects" : 0,
  "avgObjSize" : 0,
  "dataSize" : 0,
  "storageSize" : 0,
  "numExtents" : 0,
  "indexes" : 0,
  "indexSize" : 0,
  "scaleFactor" : 1,
  "fileSize" : 0,
  "fsUsedSize" : 0,
  "fsTotalSize" : 0,
  "ok" : 1
}
> db.createUser({
...   user:"suchi",
...   pwd:"123",
...   roles:["readWrite","dbAdmin"]
... });
Successfully added user: { "user" : "suchi", "roles" : [ "readWrite", "dbAdmin" ] }
```

MONGO DB COMMANDS

- **Create Collections**

```
db.createCollection('students');
```

- **Displays all collections**

```
show collections
```

- **Insert Data**

```
db.students.insert({
    First_Name:"John",
    Last_Name:"Dicken"
})
```

```
db.students.insert({
    First_Name:"Mary",
    Last_Name:"Kate",
    Gender:"Female",
    age:23})
```

- **Bulk Insert Data**

```
db.students.insert([
    {
        First_Name:"Sam",
        Last_Name:"Jackson",
        email:["sam@cmi.ac.in","sam@gmail.com"]},
    { First_Name:"Chris",
      Last_Name:"Jack",
      Gender:"Male"}
])
```

```
> db.students.insert<<
... First_Name:"John",
... Last_Name:"Dicken"
... >>;
WriteResult<< "nInserted" : 1 >>
> db.students.insert<<
... First_Name:"Mary",
... Last_Name:"Kate",
... Gender:"Female",
... age:23
... >>;
WriteResult<< "nInserted" : 1 >>
> db.students.insert([
... <
... First_Name:"Sam",
... Last_Name:"Jackson",
... email:["sam@cmi.ac.in","sam@gmail.com"]>,
... < First_Name:"Chris",
... Last_Name:"Jack",
... Gender:"Male">
... ]>];
BulkWriteResult<<
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 2,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "nUpserted" : [ ]
>>
> █
```

MONGO DB COMMANDS

- List of records:

```
db.students.find()
```

```
db.students.find().pretty()
```

```
db.students.insert([
  {First_Name:"Jim",
   Last_Name:"Carry",
   email:["Jim@cmi.ac.in","Jim@gmail.com"],
   Address:{
     houseno:"123/1",
     street:"1st cross st",
     locality:"chennai",
     pincode:"600073"}
  }
])
```

```
db.students.find( { 'Address.pincode': "600073" })
```

```
db.students.find()
{ "_id" : ObjectId("5e5a7b6494d2a62f45369245"), "First_Name" : "John", "Last_Name" : "Dicken" }
{ "_id" : ObjectId("5e5a7b7594d2a62f45369246"), "First_Name" : "Mary", "Last_Name" : "Kate", "Gender" : "Female", "age" : 23 }
{ "_id" : ObjectId("5e5a7c1a94d2a62f45369247"), "First_Name" : "Sam", "Last_Name" : "Jackson", "email" : [ "sam@cmi.ac.in", "sam@gmail.com" ] }
{ "_id" : ObjectId("5e5a7c1a94d2a62f45369248"), "First_Name" : "Chris", "Last_Name" : "Jack", "Gender" : "Male" }
db.students.find().pretty();
{
  "_id" : ObjectId("5e5a7b6494d2a62f45369245"),
  "First_Name" : "John",
  "Last_Name" : "Dicken"
}
{
  "_id" : ObjectId("5e5a7b7594d2a62f45369246"),
  "First_Name" : "Mary",
  "Last_Name" : "Kate",
  "Gender" : "Female",
  "age" : 23
}
{
  "_id" : ObjectId("5e5a7c1a94d2a62f45369247"),
  "First_Name" : "Sam",
  "Last_Name" : "Jackson",
  "email" : [
    "sam@cmi.ac.in",
    "sam@gmail.com"
  ]
}
{
  "_id" : ObjectId("5e5a7c1a94d2a62f45369248"),
  "First_Name" : "Chris",
  "Last_Name" : "Jack",
  "Gender" : "Male"
}
```

```
mongo@shell>:9:5
> db.students.insert([
...   {First_Name:"Jim",
...     Last_Name:"Carry",
...     email:["Jim@cmi.ac.in","Jim@gmail.com"],
...     Address:{
...       houseno:"123/1",
...       street:"1st cross st",
...       locality:"chennai",
...       pincode:"600073"}
...   }
... ])
BulkWriteResult<<
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 1,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
}>
```

```
> db.students.find( { 'Address.pincode': "600073" });
{ "_id" : ObjectId("5e5a80fd94d2a62f45369249"), "First_Name" : "Jim", "Last_Name" : "Carry", "email" : [ "Jim@cmi.ac.in", "Jim@gmail.com" ], "Address" : { "houseno" : "123/1", "street" : "1st cross st", "locality" : "chennai", "pincode" : "600073" } }
```

MONGO DB COMMANDS

- update record

```
db.students.update({First_Name:"Sam"},{First_Name:"Samuel",Last_Name:"Jackson"})
```

```
db.students.update({First_Name:"Samuel"},{$set:{Gender:"Male"}})
```

```
> db.students.update({First_Name:"Sam"},{First_Name:"Samuel",Last_Name:"Jackson"});
WriteResult<< "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 >>
> db.students.update({First_Name:"Samuel"},{$set:{Gender:"Male"}});
WriteResult<< "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 >>
```

```
db.students.update({First_Name:"Juli"},{$inc:{age:2}})
```

```
> db.students.update({First_Name:"Juli"},{$inc:{age:2}});
WriteResult<< "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 >>
```

```
db.students.update({First_Name:"Mary"},{$unset:"Last_Name"})
```

```
> db.students.update({First_Name:"Mary"},{$unset:{Last_Name:""}});
WriteResult<< "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 >>
```


MONGO DB COMMANDS

- Remove record

```
db.students.remove({First_Name:"Chris"});
```

```
> db.students.remove({First_Name:"Chris"});
WriteResult({ "nRemoved" : 1 })
>
```

- Find record

```
db.students.find({First_Name:"Mary"})
```

```
db.students.find({$or:[{First_Name:"Samuel"},{First_Name:"Mary"}]})
```

```
db.students.find({age:{$gt:12}})
```

```
> db.students.find({First_Name:"Mary"});
{ "_id" : ObjectId("5e5a7b7594d2a62f45369246"), "First_Name" : "Mary", "Gender" : "Female", "age" : 25 }
> db.students.find({$or:[{First_Name:"Samuel"},{First_Name:"Mary"}]});
{ "_id" : ObjectId("5e5a7b7594d2a62f45369246"), "First_Name" : "Mary", "Gender" : "Female", "age" : 25 }
{ "_id" : ObjectId("5e5a7c1a94d2a62f45369247"), "First_Name" : "Samuel", "Last_Name" : "Jackson", "Gender" : "Male" }
> db.students.find({age:{$gt:12}});
{ "_id" : ObjectId("5e5a7b7594d2a62f45369246"), "First_Name" : "Mary", "Gender" : "Female", "age" : 25 }
>
```

MONGO DB COMMANDS

- Sort

```
db.students.find().sort({Last_name:-1}).pretty()
```

```
db.students.find().sort({Last_name:1}).pretty()
```

1 is used for ascending order while -1 is used for descending order.

```
> db.students.find().sort({Last_Name:-1}).pretty();
{
  "_id" : ObjectId("5e5a7c1a94d2a62f45369247"),
  "First_Name" : "Samuel",
  "Last_Name" : "Jackson",
  "Gender" : "Male"
}
{
  "_id" : ObjectId("5e5a7b6494d2a62f45369245"),
  "First_Name" : "John",
  "Last_Name" : "Dicken"
}
{
  "_id" : ObjectId("5e5a80fd94d2a62f45369249"),
  "First_Name" : "Jim",
  "Last_Name" : "Carry",
  "email" : [
    "Jim@cmi.ac.in",
    "Jim@gmail.com"
  ],
  "Address" : {
    "housetno" : "123/1",
    "street" : "1st cross st",
    "locality" : "chennai",
    "pincode" : "600073"
  }
}
{
  "_id" : ObjectId("5e5a7b7594d2a62f45369246"),
  "First_Name" : "Mary",
  "Gender" : "Female",
  "age" : 25
}
```

```
> db.students.find().sort({Last_Name:1}).pretty();
{
  "_id" : ObjectId("5e5a7b7594d2a62f45369246"),
  "First_Name" : "Mary",
  "Gender" : "Female",
  "age" : 25
}
{
  "_id" : ObjectId("5e5a80fd94d2a62f45369249"),
  "First_Name" : "Jim",
  "Last_Name" : "Carry",
  "email" : [
    "Jim@cmi.ac.in",
    "Jim@gmail.com"
  ],
  "Address" : {
    "housetno" : "123/1",
    "street" : "1st cross st",
    "locality" : "chennai",
    "pincode" : "600073"
  }
}
{
  "_id" : ObjectId("5e5a7b6494d2a62f45369245"),
  "First_Name" : "John",
  "Last_Name" : "Dicken"
}
{
  "_id" : ObjectId("5e5a7c1a94d2a62f45369247"),
  "First_Name" : "Samuel",
  "Last_Name" : "Jackson",
  "Gender" : "Male"
}
```

MONGO DB COMMANDS

- **Aggregation**

`db.students.find().count()`

```
> db.students.find().count();
4
>
```

`db.students.find().forEach(function(doc){print("First Name is "+ doc.First_Name)})`

```
> db.students.find().forEach(function(doc){print( "First Name is "+ doc.First_Name )});
First Name is John
First Name is Mary
First Name is Samuel
First Name is Jim
>
```

Reference: <https://docs.mongodb.com/manual/crud/>

Quiz 6

- Choose the correct answer from the options given below:
- A) In Mongo DB, documents in same collection need not have the same set of fields.
- B) `db.students.find({age:{$gt:20, $lt:30}})`; returns records having age between 20 & 30
- C) Mongo DB supports nested JSON document
- D) All the above.

THANK YOU