



NoSQL

The NoSQL DBMS

NoSQL:

- A fast, portable, open-source RDBMS
- A derivative of the RDB database system
- Based on the “operator/stream paradigm”

- One common interpretation of NoSQL is “not only SQL” or like “non-relational”

NoSQL

- NoSQL is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users).
- These type of data storing may not require fixed schema, avoid join operations and typically scale horizontally.

RDBMS

- Structured and organized data
- Structured query language (SQL)
- Data and its relationships are stored in separate tables.
- Data Manipulation Language, Data Definition Language
- Tight Consistency

NoSQL

- Stands for Not Only SQL
- No declarative query language
- No predefined schema
- Key-Value pair storage, Column Store, Document Store, Graph databases
- Eventual consistency rather ACID property
- Unstructured and unpredictable data
- Prioritizes high performance, high availability and scalability

NoSQL Examples

| | | | | | | | |
|---------------------------|------------------|--------------------|--------------------|------------------------|------------------|----------------------|----------------------------------|
| Hbase | Cassandra | Hypertable | Accumulo | Amazon SimpleDB | SciDB | Stratosphere | flare |
| Cloudata | BigTable | QD Technology | | SmartFocus KDI | Alterian | Cloudera | C-Store |
| Vertica | Qbase-MetaCarta | | OpenNeptune | HPCC | Mongo DB | CouchDB | Clusterpoint ServerTerrastore |
| Jackrabbit | OrientDB | Perservere | CoudKit | Djondb | SchemaFreeDB | SDB | JasDB |
| RaptorDB | ThruDB | RavenDB | DynamoDB | Azure Table Storage | Couchbase Server | | Riak |
| LevelDB | Chordless | GenieDB | Scalaris | Tokyo | Kyoto Cabinet | Tyrant | Scalien |
| Berkeley DB | Voldemort | Dynomite | KAI | MemcacheDB | Faircom C-Tree | | HamsterDB STSdb |
| Tarantool/Box | Maxtable | Pincaster | RaptorDB | TIBCO Active Spaces | allegro-C | nessDB | HyperDex |
| Mnesia | LightCloud | Hibari | BangDB | OpenLDAP/MDB/Lightning | Scality | Redis | |
| KaTree | TomP2P | Kumofs | TreapDB | NMDB | luxio | actord | Keyspace |
| schema-free | RAMCloud | SubRecord | Mo8onDb | Dovetaildb | JDBM | Neo4 | InfiniteGraph |
| Sones | InfoGrid | HyperGraphDB | | DEX | GraphBase | Trinity | AllegroGraph BrightstarDB |
| Bigdata | Meronymy | OpenLink Virtuoso | | VertexDB | FlockDB | Execom IOG | Java Univ Netwrk/Graph Framework |
| OpenRDF/Sesame | | Filament | OWLim | NetworkX | iGraph | Jena | SPARQL OrientDb |
| ArangoDB | AlchemyDB | Soft NoSQL Systems | | Db4o | Versant | Objectivity | Starcounter |
| ZODB | Magma | NEO | PicoList | siaqodb | Sterling | Morantex | EyeDB |
| HSS DatabaseFramerD | | Ninja Database Pro | | StupidDB | KiokuDB | Perl solution | Durus |
| GigaSpaces | Infinispan | Queplix | Hazelcast | GridGain | Galaxy | SpaceBase | JoafipCoherence |
| eXtremeScale | MarkLogic Server | | EMC Documentum xDB | eXist | Sedna | BaseX | Qizx |
| Berkeley DB XML | | Xindice | Tamino | Globals | Intersystems | Cache | GT.M EGTM |
| U2 | OpenInsight | Reality | OpenQM | ESENT | jBASE | MultiValue | Lotus/Domino |
| eXtremeDB | RDM Embedded | | ISIS Family | Prevayler | Yserial | Vmware vFabric | GemFire Btrieve |
| KirbyBase | Tokutek | Recutils | FileDB | Armadillo | illuminate | Correlation Database | FluidDB |
| Fleet DB | Twisted Storage | | Rindo | Sherpa | tin | Dryad | SkyNet Disco |
| MUMPS | Adabas | XAP In-Memory Grid | | eXtreme Scale | | MckoiDDB | Mckoi SQL Database |
| Oracle Big Data Appliance | Innostore | FleetDB | | No-List | KDI | Perst | IODB |

MongoDB

- MongoDB is a document database designed for ease of development and scaling.
- MongoDB offers both a *Community* and an *Enterprise* version of the database.
- A record in MongoDB is a document, which is a data structure composed of field and value pairs.
- MongoDB documents are similar to JSON objects.
- The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using document database are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

Installing MongoDB

The screenshot shows the MongoDB website. The top navigation bar includes links for Cloud, Software, Pricing, Learn, Solutions, and Docs. A banner for the MongoDB live series is visible. The main content area features a grid of product categories: Atlas, Database, Search, Enterprise Advanced, Enterprise Server, Ons Manager, Community Edition (highlighted with a red circle), Community Server, and Cloud. The 'Community Edition' link is the target for the installation process.

Go to:
www.mongodb.com

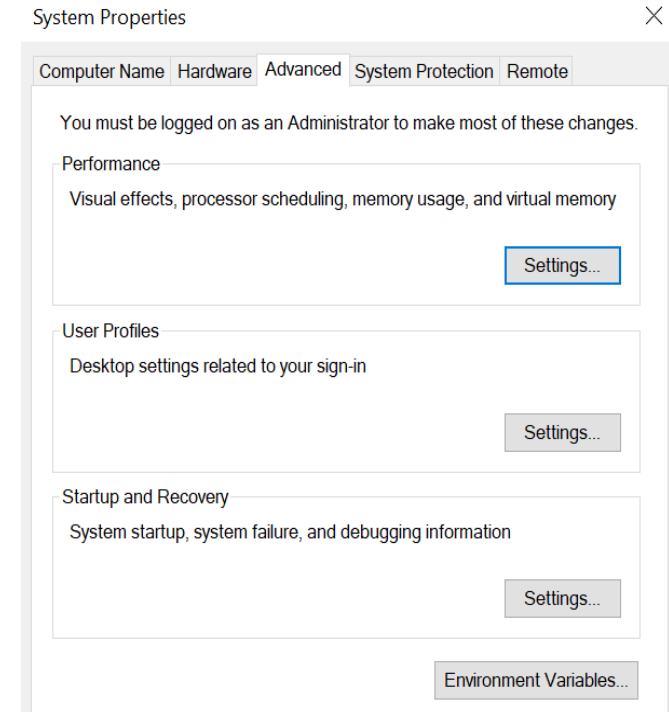
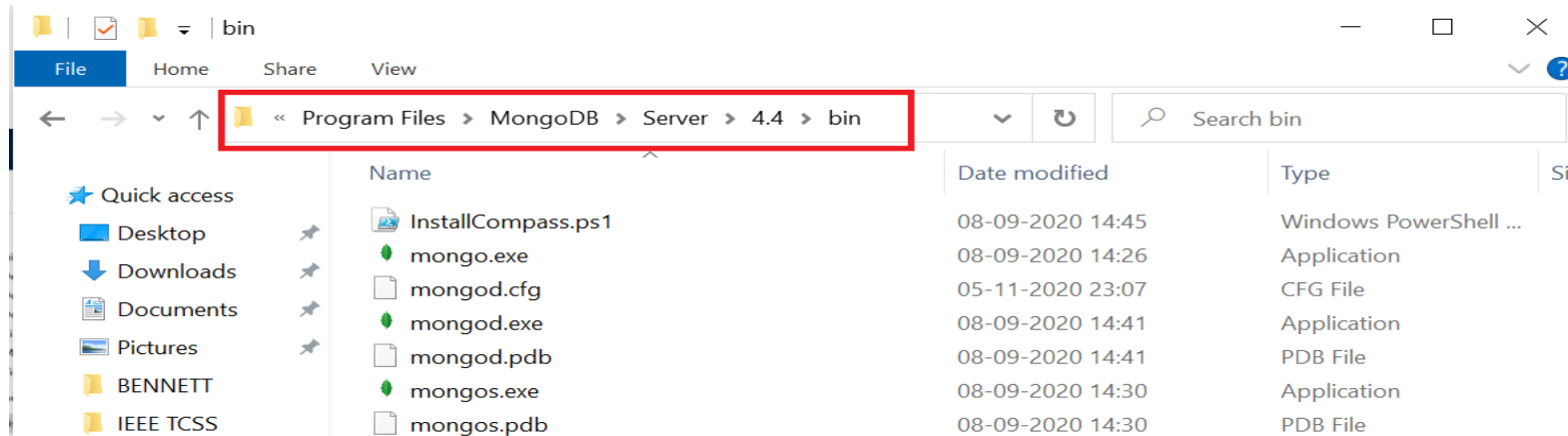
Then go to
Community
Edition ->
Community
Server

Select
version and
platform

Then Click
Download

Installing Steps

- Run downloaded .msi file
- Install complete version
- Install MongoDB as service (by default)
- Keep data directory and log directory as it is
- Install MongoDB compass
- Wait for installation to finish.



1. Copy the path 2. Click environment variables 3. System variable 4. Add path 5. Add the address (upto bin) 6. save this
2. A) Now go to C folder B) Create 'data' folder C) within data create 'db' folder

Run Command prompt

Command Prompt

```
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\smuhu>cd C:\Program Files\MongoDB\Server\4.4\bin

C:\Program Files\MongoDB\Server\4.4\bin>
```

Go to the bin folder

Command Prompt - mongod

```
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\smuhu>cd C:\Program Files\MongoDB\Server\4.4\bin

C:\Program Files\MongoDB\Server\4.4\bin>mongod
{"t":{"$date":"2020-11-05T23:28:30.447+05:30"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatically
disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2020-11-05T23:28:30.449+05:30"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No TransportL
ayer configured during NetworkInterface startup"}
{"t":{"$date":"2020-11-05T23:28:30.450+05:30"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"main","msg":"Implicit TCP
FastOpen in use."}
{"t":{"$date":"2020-11-05T23:28:30.451+05:30"},"s":"I", "c":"STORAGE", "id":4615611, "ctx":"initandlisten","msg":"Mong
oDB starting","attr":{"pid":6424,"port":27017,"dbPath":"C:/data/db/","architecture":"64-bit","host":"LAPTOP-Q7UL5R16"}}
{"t":{"$date":"2020-11-05T23:28:30.452+05:30"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg":"Targ
et operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2020-11-05T23:28:30.452+05:30"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Buil
d Info","attr":{"buildInfo":{"version":"4.4.1","gitVersion":"ad91a93a5a31e175f5cbf8c69561e788bbc55ce1","modules":[],"all
ocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2020-11-05T23:28:30.452+05:30"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg":"Open
```

Open command prompt. Command is mongod. It will start the mongo demon

Command Prompt - mongo

```
C:\Users\smuhu>cd C:\Program Files\MongoDB\Server\4.4\bin

C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("9ff8a1c3-6b58-4641-ab8f-89cb35f40b77") }

MongoDB server version: 4.4.1
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
  https://community.mongodb.com
---
```

Open another command prompt. Command is "mongo". It will start the mongo shell. Now you can write programs of NoSQL.

Shell script will show ">" sign

Commands

Show Databases

> Show dbs

```
> show dbs
admin      0.000GB
bennett    0.000GB
config     0.000GB
local      0.000GB
mapdb      0.000GB
```

Create new database

> Use database_name

> Use thapar

```
> use thapar
switched to db thapar
>
```

Create collection

Here collection represents table

>db.createCollection("table_name")

>db.createCollection("testtable")

```
> db.createCollection("testtable")
{ "ok" : 1 }
>
```

```
> use another
switched to db another
> show dbs
admin      0.000GB
bennett    0.000GB
config     0.000GB
local      0.000GB
mapdb      0.000GB
thapar     0.000GB
```

Lets create another database named "another"

Show databases

It will not show "another" as no table or collection is created

```
> db.createCollection("Extra")
{ "ok" : 1 }
> show dbs
admin      0.000GB
another    0.000GB
bennett    0.000GB
config     0.000GB
local      0.000GB
mapdb      0.000GB
thapar     0.000GB
> db.dropDatabase()
{ "dropped" : "another", "ok" : 1 }
> show dbs
admin      0.000GB
bennett    0.000GB
config     0.000GB
local      0.000GB
mapdb      0.000GB
thapar     0.000GB
>
```

Insert a collection "Extra"

Now show databases will show "another"

For delete database use
>db.dropDatabase()

Show the databases again

"another" database is deleted

Commands

Go to the database
See all the collection
Presents using
>show collections
For delete
>db.collection_name.drop()

```
> use thapar
switched to db thapar
> show collections
testtable
> db.testtable.drop()
true
> show collections
>
```

```
> db.testtable.insertOne({"id":1, "name":"ABC","dept":"CSE"})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5fa4514a8bda3bca99bae594")
}
>
```

Insert 1 data into the collection/table
using "insertOne" operation

```
@(shell):1:1
> db.testtable.insertMany([{"id":1, "name":"ABC","dept":"CSE"}, {"id":2, "name":"PQR", "dept":"CSE"}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5fa452e68bda3bca99bae595"),
    ObjectId("5fa452e68bda3bca99bae596")
  ]
}
>
```

Insert many data into the collection
using "insertMany" operation

Db.tablename.insertMany([{ }, { }, { }])

Commands

Retrieve data

>db.table_name.find();

```
> db.testtable.find()
{ "_id" : ObjectId("5fa452e68bda3bca99bae595"), "id" : 1, "name" : "ABC", "dept" : "CSE" }
{ "_id" : ObjectId("5fa452e68bda3bca99bae596"), "id" : 2, "name" : "PQR", "dept" : "CSE" }
>
```

Select some particular value

>db.table_name.find({attribute: {\$option: value}})

```
> db.testtable.find({id:{$eq:2}})
{ "_id" : ObjectId("5fa452e68bda3bca99bae596"), "id" : 2, "name" : "PQR", "dept" : "CSE" }
>
```

You can find details in the following website:

<https://docs.mongodb.com/manual/tutorial/query-documents/>

<https://docs.mongodb.com/manual/reference/operator/query/>

List of options available

| Name | Description |
|-----------------------|---|
| \$eq | Matches values that are equal to a specified value. |
| \$gt | Matches values that are greater than a specified value. |
| \$gte | Matches values that are greater than or equal to a specified value. |
| \$in | Matches any of the values specified in an array. |
| \$lt | Matches values that are less than a specified value. |
| \$lte | Matches values that are less than or equal to a specified value. |
| \$ne | Matches all values that are not equal to a specified value. |
| \$nin | Matches none of the values specified in an array. |

Commands

Delete row

>db.table_name.deleteMany({attribute:value})

```
> db.testtable.deleteMany({ name : "ABC" })  
{ "acknowledged" : true, "deletedCount" : 1 }
```

Again show the records

>db.table_name.find()

```
> db.testtable.find()  
{ "_id" : ObjectId("5fa452e68bda3bca99bae596"), "id" : 2, "name" : "PQR", "dept" : "CSE" }  
> _
```

the record with the name "ABC" has been deleted.

`db.collection.updateOne()`

Updates at most a single document that match a specified filter even though multiple documents may match the specified filter.

New in version 3.2.

`db.collection.updateMany()`

Update all documents that match a specified filter.

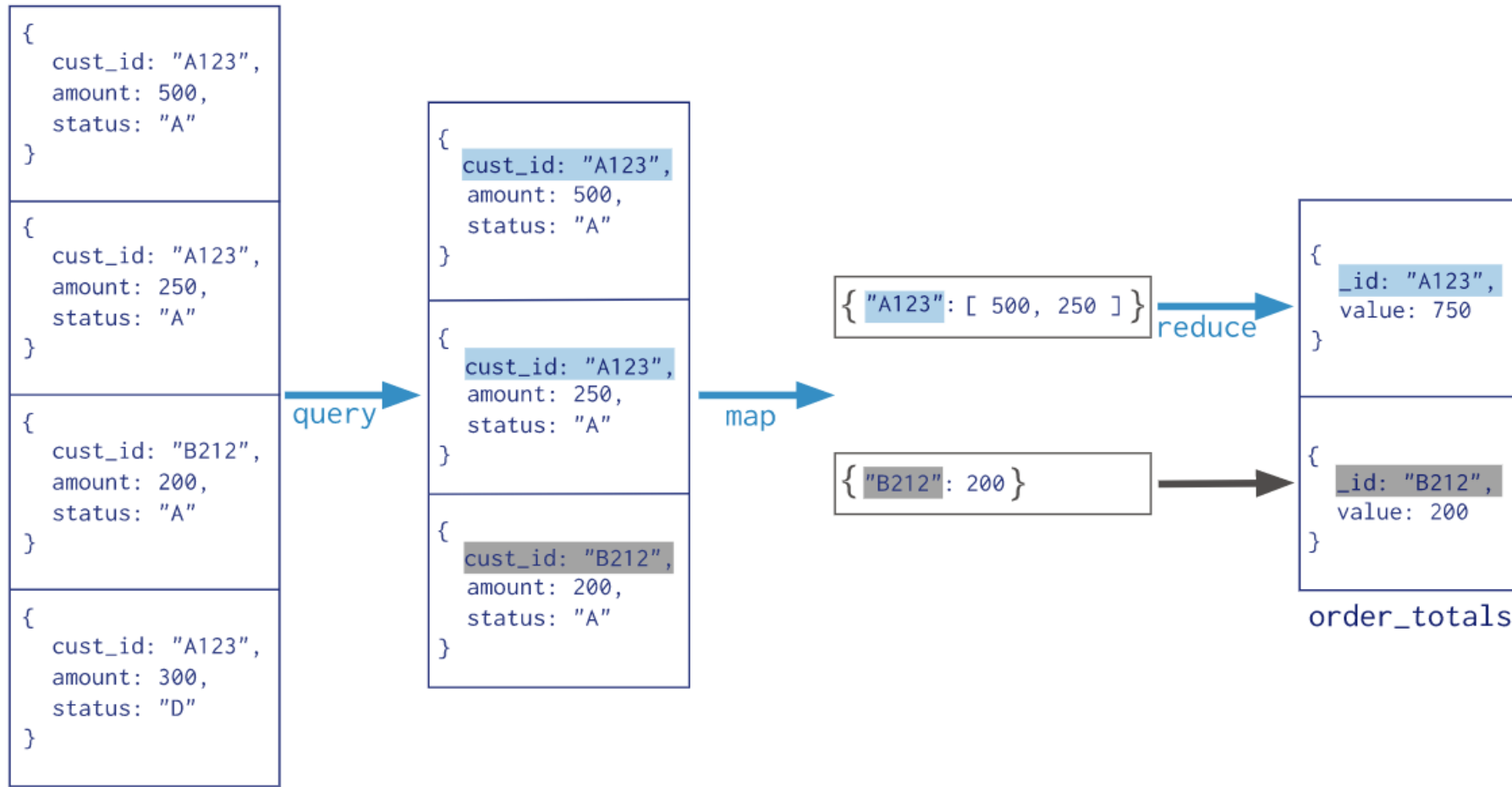
New in version 3.2.

`db.collection.replaceOne()`

Replaces at most a single document that match a specified filter even though multiple documents may match the specified filter.

Different types of
update options available

MapReduce Example



Source: <https://docs.mongodb.com/manual/aggregation/>

MapReduce Example

Create a database
Create collection "cust"

| |
|---|
| { cust_id: "A123", amount: 500, status: "A" } |
| { cust_id: "A123", amount: 250, status: "A" } |
| { cust_id: "B212", amount: 200, status: "A" } |
| { cust_id: "A123", amount: 300, status: "D" } |

```
> use mapdb
switched to db mapdb
> db.createCollection("cust")
{ "ok" : 1 }
> db.cust.insertMany([{"id":"A123","amount":500,"status":"A"}, {"id":"A123","amount":250,"status":"A"}, {"id":"B212",
"amount":200,"status":"A"}, {"id":"A123","amount":300,"status":"D"}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5fa501d9bfc8ebf4554384ce"),
    ObjectId("5fa501d9bfc8ebf4554384cf"),
    ObjectId("5fa501d9bfc8ebf4554384d0"),
    ObjectId("5fa501d9bfc8ebf4554384d1")
  ]
}
```

query →

| |
|---|
| { cust_id: "A123", amount: 500, status: "A" } |
| { cust_id: "A123", amount: 250, status: "A" } |
| { cust_id: "B212", amount: 200, status: "A" } |

Use aggregate function. Take the data of "A" and sum the amount based on "id"

```
> db.cust.aggregate([{$match:{status:"A"}}, {$group:{_id:"$id", total:{$sum:"$amount"}}}])
{ "_id" : "A123", "total" : 750 }
{ "_id" : "B212", "total" : 200 }
>
```

Reference

http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page

Manual: <https://docs.mongodb.com/manual>