SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Department of Information Technology

19CST202 – DATABASE MANAGEMENT SYSTEM

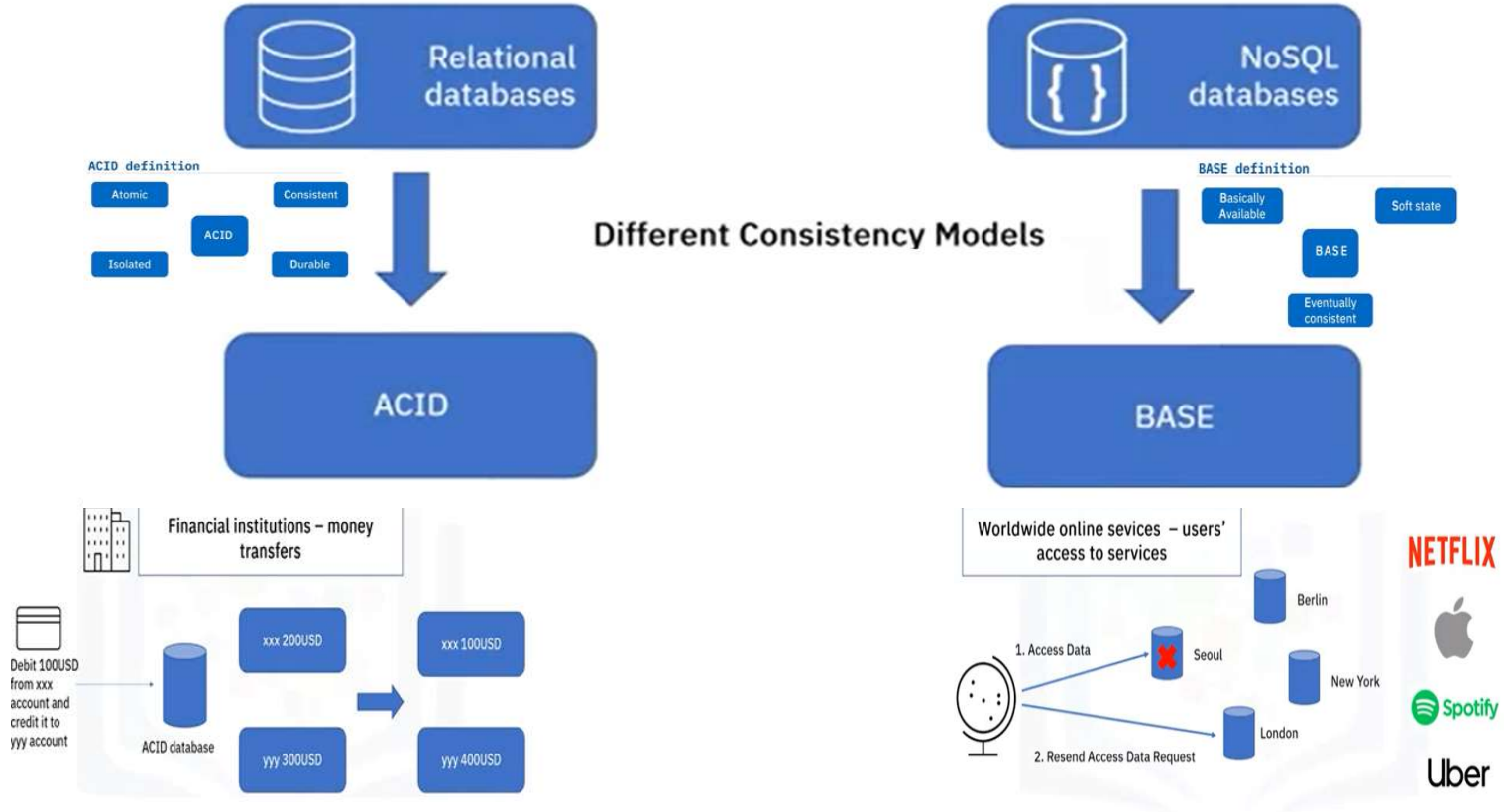
III B.Tech. IT/ VI SEMESTER

SQL vs NoSQL

- ✓ NoSQL (often interpreted as Not only SQL) database
- ✓ It provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

SQL	NoSQL
Relational Database Management System (RDBMS)	Non-relational or distributed database system.
These databases have fixed or static or predefined schema	They have dynamic schema
These databases are best suited for complex queries	These databases are not so good for complex queries
Vertically Scalable	Horizontally scalable
Follows ACID property	Follows BASE property

SQL vs NoSQL

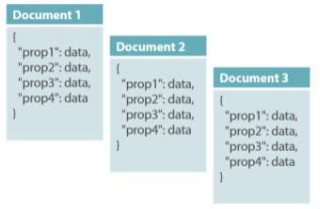

Relational databases
ACID definition
Atomic
Consistent
ACID
Isolated
Durable
ACID
Financial institutions – money transfers
Debit 100USD from xxx account and credit it to yyy account
ACID database
xxx 200USD
xxx 100USD
yyy 300USD
yyy 400USD
Different Consistency Models
NoSQL databases
BASE definition
Basically Available
Soft state
BASE
Eventually consistent
BASE
Worldwide online sevices – users' access to services
Berlin
1. Access Data
Seoul
New York
London
2. Resend Access Data Request
NETFLIX
Spotify
Uber

NoSQL Types

Graph database

Neo4j the graph database

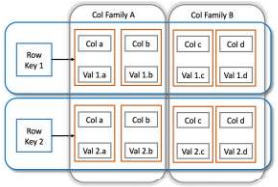
HyperGraphDB



Document-oriented

mongoDB

CouchDB relax



Column family

Cassandra

HBASE

mongoDB®

What is MongoDB?

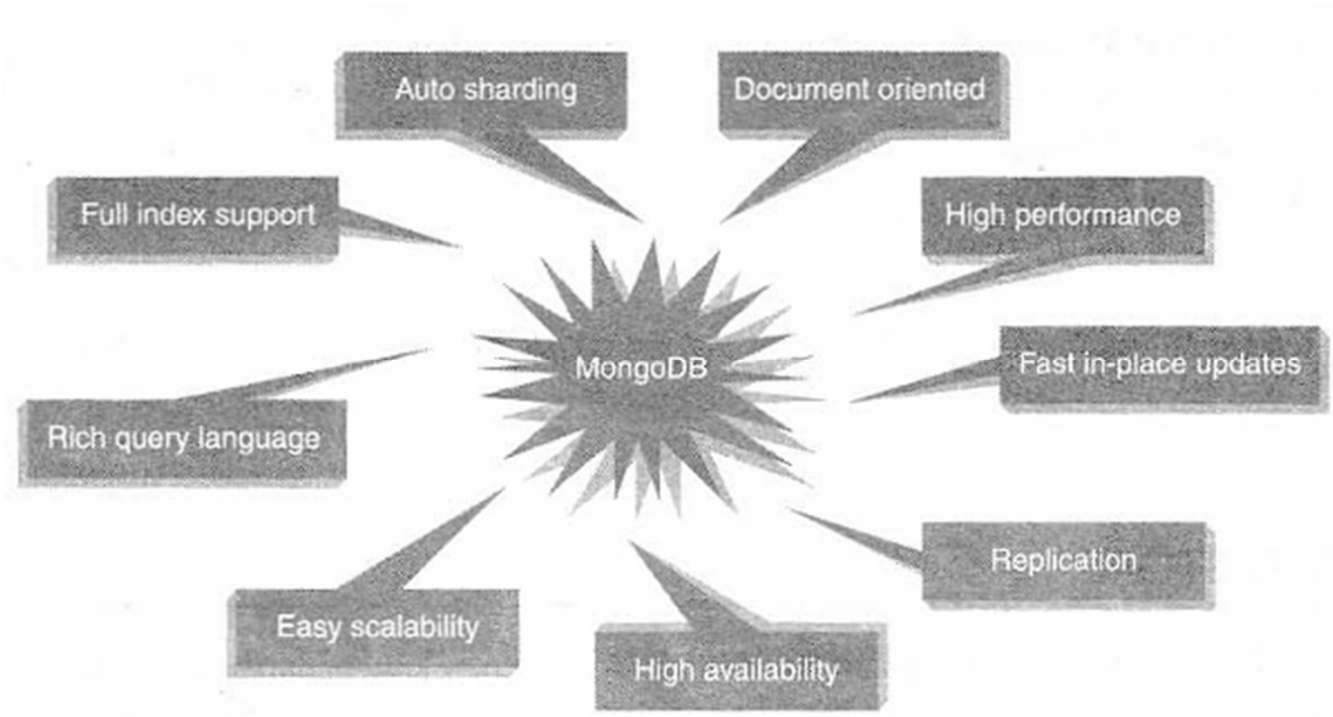
MongoDB is

- Cross Platform
- Open Source
- Non relational
- Distributed
- NoSQL
- Document Oriented Data Source

```
{
   "_id" : ObjectId("5114e0bd42…"),
   “FirstName" : "John",
   “LastName" : "Doe",
   “Age" : 39,
   “Interests" : [ "Reading",  "Mountain Biking ]
   “Favorites": {
                  "color": "Blue",
                  "sport": "Soccer“
         }
}
```

✓ Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas(schema-free, schema less).

Why MongoDB?


Auto sharding
Document oriented
Full index support
High performance
MongoDB
Fast in-place updates
Rich query language
Replication
Easy scalability
High availability

MongoDB is Easy to Use

mongoDB®

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

— no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB Document

```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
   { model: 'Bentley',
     year: 1973,
     value: 100000, … },
   { model: 'Rolls Royce',
     year: 1965,
     value: 330000, … }
  ]
}
```

Journey of XML to JSON

YAML

```yaml
apis:
  - name: login
    port: 8080
  - name: profile
    port: 8090
```

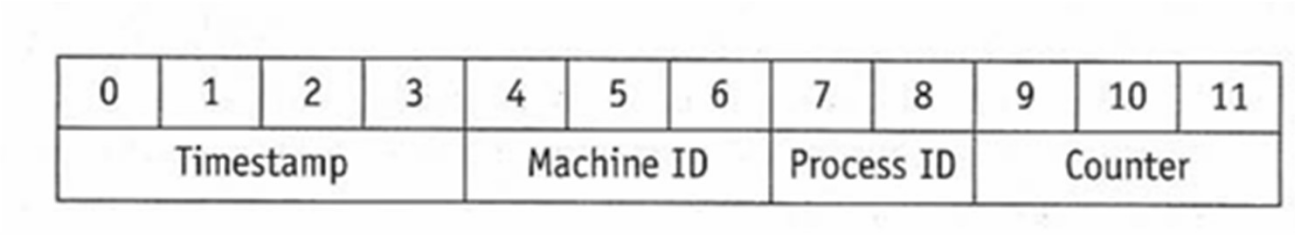
XML

```xml
<apis>
    <api>
        <name>login</name>
        <port>8080</port>
    </api>
    <api>
        <name>profile</name>
        <port>8090</port>
    </api>
</apis>
```

JSON

```json
{
  "apis": [
    {
      "name": "login",
      "port": 8080
    },
    {
      "name": "profile",
      "port": 8090
    }
  ]
}
```

Creating or Generating a Key



0	1	2	3	4	5	6	7	8	9	10	11
Timestamp				Machine ID			Process ID		Counter		

Index is automatically built on the unique identifier

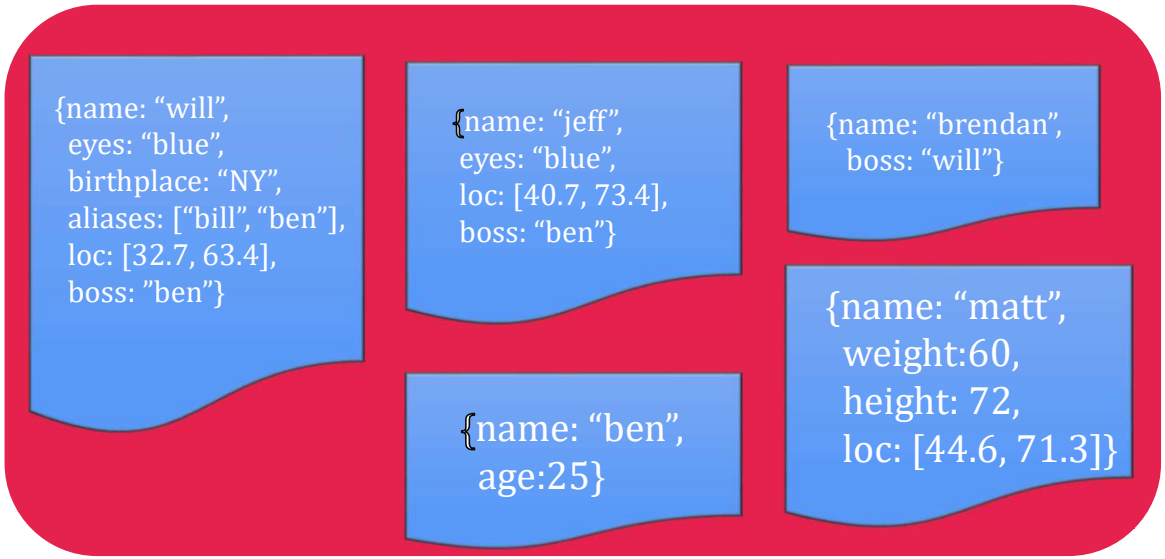
It is based on user choice

- Either you have to allocate the unique key
- Mongo Shell will do the same

Scheme Free

MongoDB does not need any pre-defined data schema

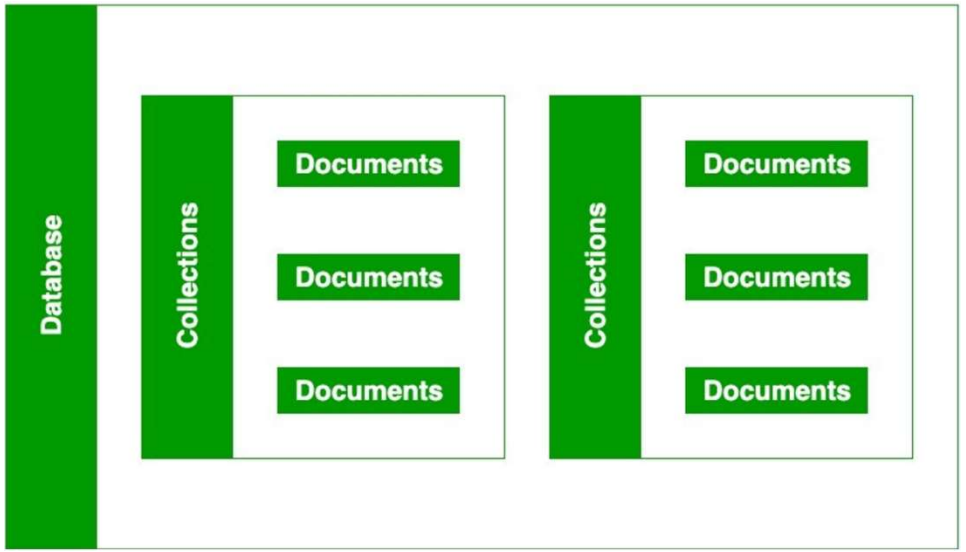
Every document could have different data!



RDBMS vs MongoDB

RDBMS		MongoDB
Database	➡	Database
Table	➡	Collection
Row	➡	Document (JSON, BSON)
Column	➡	Field
Index	➡	Index
Join	➡	Embedded Document
Partition	➡	Shard

Database, Collection and Document



Database, Collection and Document

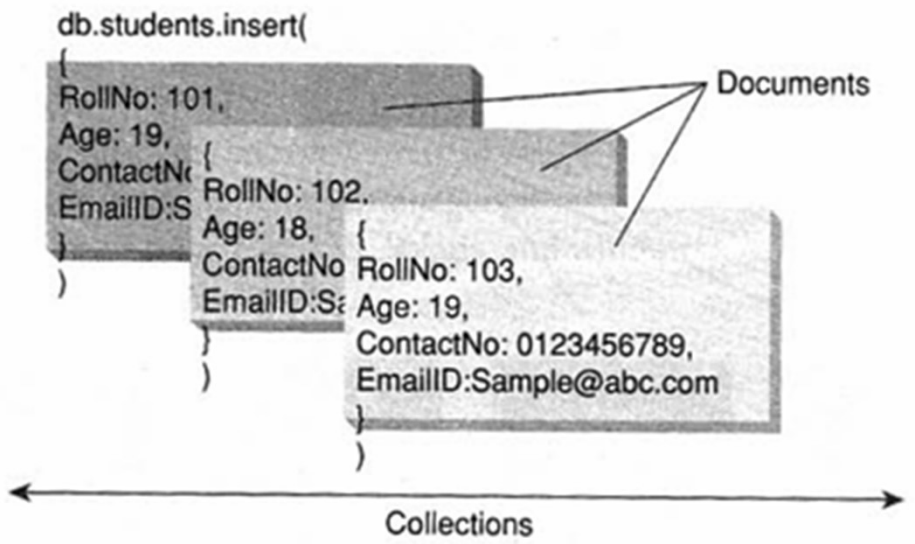


Figure 6.2 A collection "students" containing 3 documents.

Static Queries and Dynamic Data

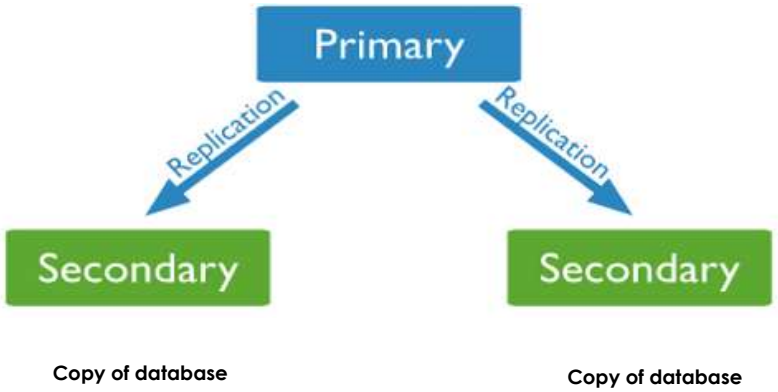
Storing Binary data

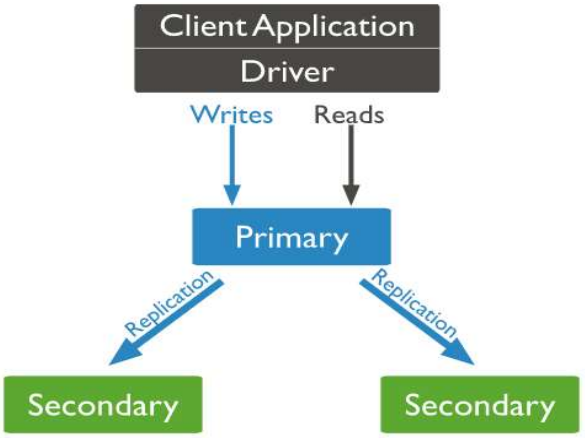
- It supports GridFS for Storing Binary data
- Store upto 4 MB of data – Profile Picture, Small Audio File.
- Wish to store Movie Clip, MongoDB provide
- Metadata – Data about data along with context information in the collection - Files

Breaks the data into small piece – Chunks
Store it in Chunks Collections

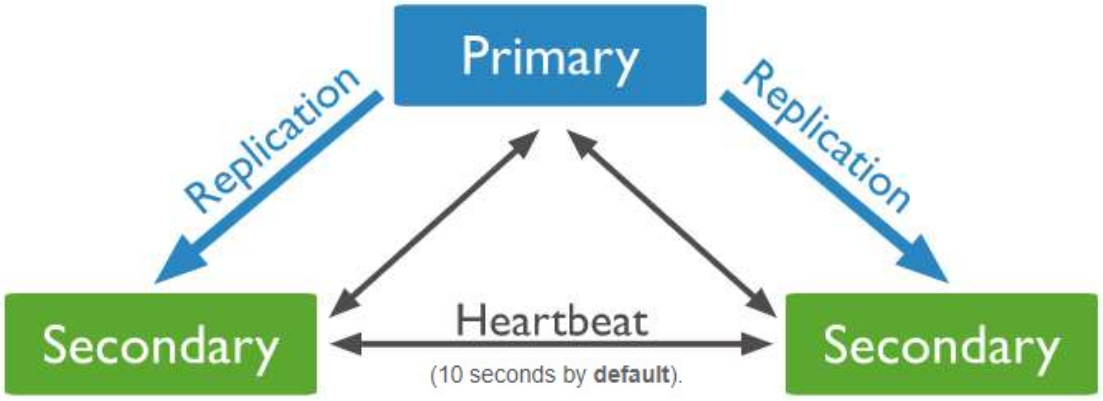
Replication

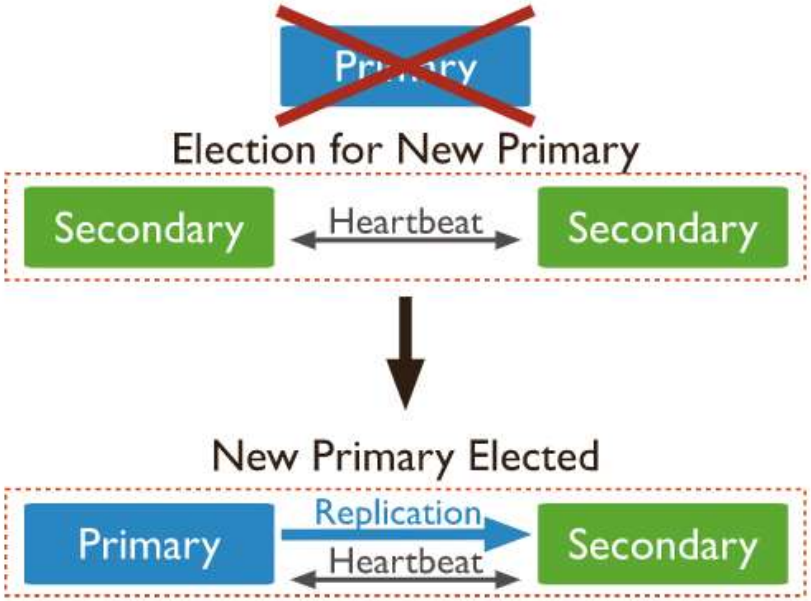
- Replication provides redundancy and increases data availability.
- With multiple copies of data on different database servers, replication provides a level of fault tolerance against the loss of a single database server.





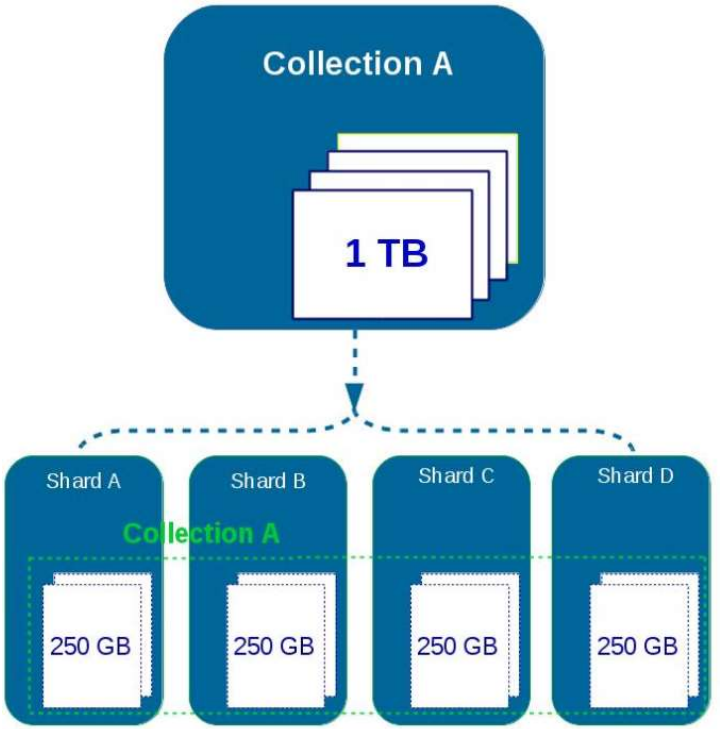
Replication





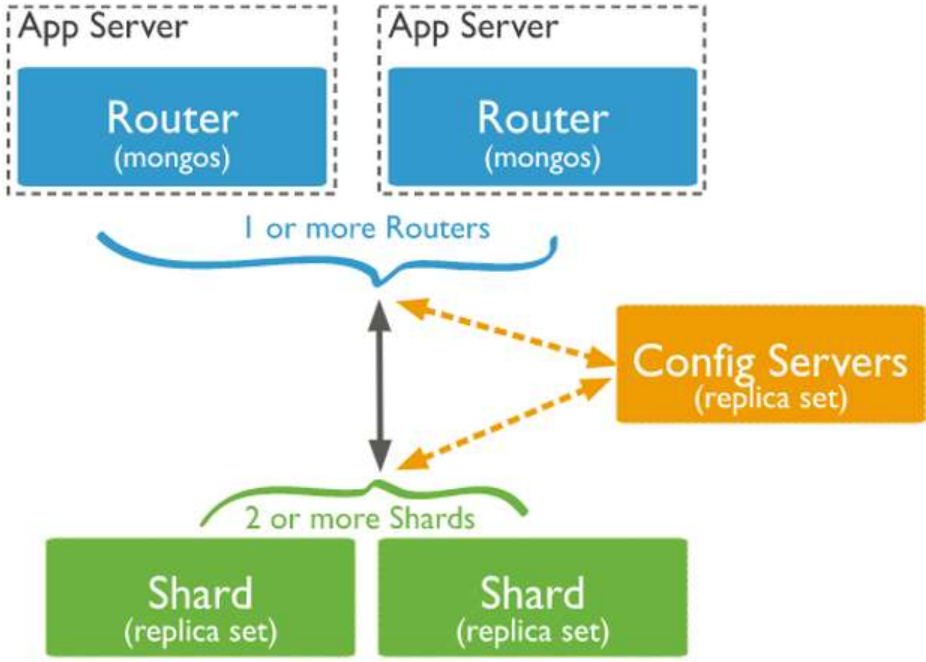
Sharding

- Sharding is a method for distributing data across multiple machines.
- MongoDB uses sharding to support deployments with very large data sets and **high throughput operations.**



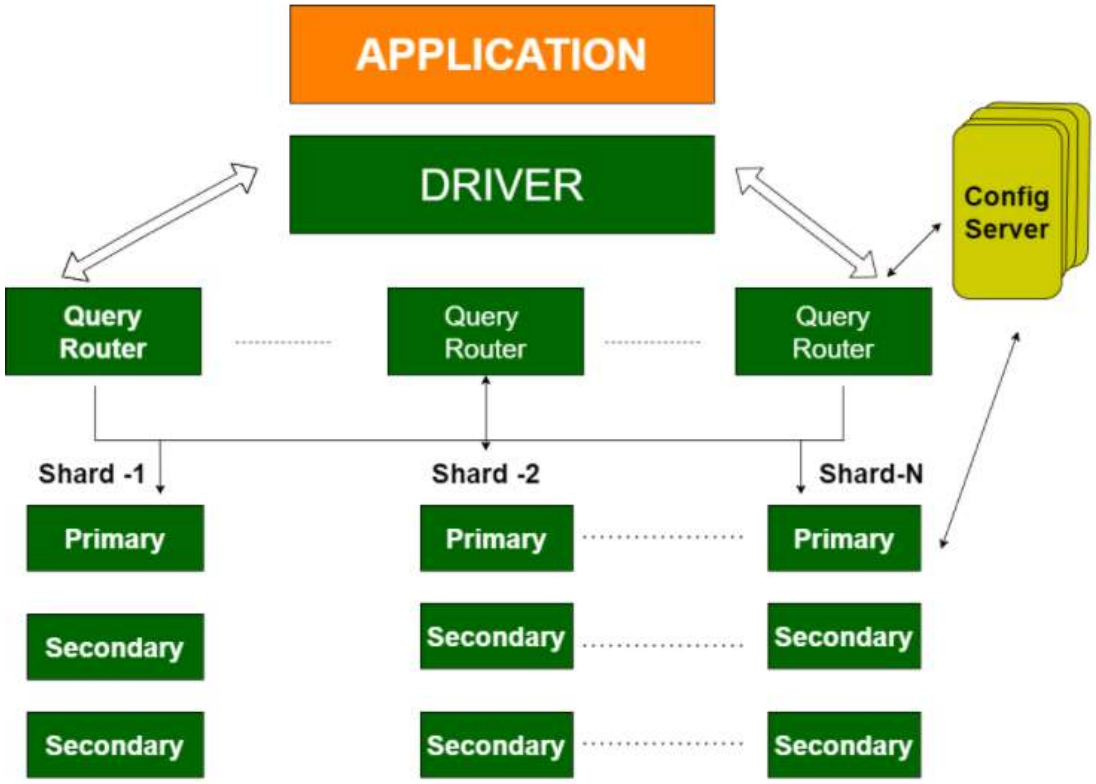
Sharding Architecture

- **Shard** is a Mongo instance to handle a subset of original data.
- **Mongos** is a query router to shards.
- **Config Server** is a Mongo instance which stores metadata information and configuration details of cluster.



Sharding/Replication

- Replication Split data sets across multiple data nodes for high availability.
- Sharding scale up/down horizontally when it is required for high throughput



Terms used in RDBMS and MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Record	Document
Columns	Fields / Key Value pairs
Index	Index
Joins	Embedded documents
Primary Key	Primary key (_id is a identifier)

	MySQL	Oracle	MongoDB
Database Server	MySqld	Oracle	Mongod
Database Client	MySql	SQL Plus	mongo

Features

- Document-Oriented storege
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Aggregation
- MongoDB Atlas
- Various APIs
  - JavaScript, Python, Ruby, Perl, Java, Java, Scala, C#, C++, Haskell, Erlang
- Community

MongoDB CURD Operations

C ⟶ Create

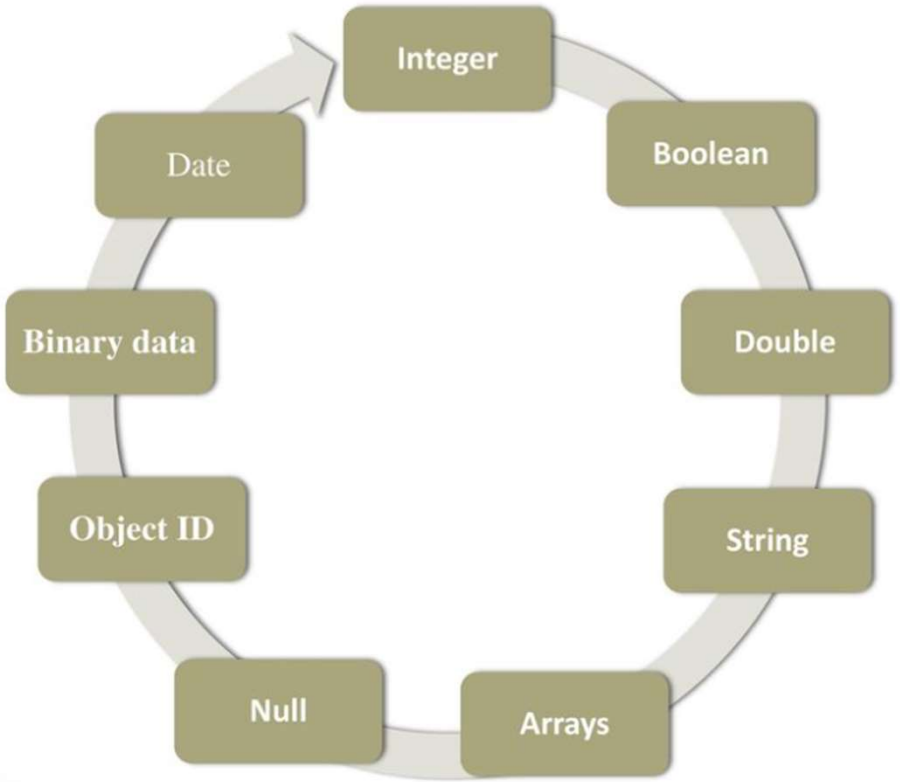
R ⟶ Read

U ⟶ Update

D ⟶ Delete

```
> use db
switched to db db
> db;
db
> Show dbs;
uncaught exception: SyntaxError: unexpected token: identifier :
@(shell):1:5
> show dbs
admin     0.000GB
config    0.000GB
database  0.000GB
dbms1     0.000GB
local     0.000GB
> db.dropDatabase;
function(writeConcern) {
    return this._dbCommand(
        {dropDatabase: 1, writeConcern: writeConcern ? writeConcern : _defaultWriteConcern});
}
>
```

Datatypes in MongoDB



To report the name of the current database:

```
> db
test
>
```

To display the list of databases:

```
> show dbs
admin   (empty)
local   0.078GB
myDB1   0.078GB
>
```

To switch to a new database, for example, myDB1:

```
> use myDB1
switched to db myDB1
>
```

To display the list of collections (tables) in the current database:

```
> show collections
system.indexes
system.js
>
```

To display the current version of the MongoDB server:

```
> db.version()
2.6.1
>
```

Create Operation

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.
db.createCollection()	It is used to create an empty collection.

db.tablename.insert({data}) or db.tablename.save({data}),

Create Operation

```
> db.student.insertOne({
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5e540cdc92e6dfa3fc48ddae")
}
>
```

```
> db.student.insertMany([
... {
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... },
...
... {
... name : "Rohit",
... age : 21,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... }
...
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5e540d3192e6dfa3fc48ddaf"),
                ObjectId("5e540d3192e6dfa3fc48ddb0")
        ]
}
>
```

Read Operations


```
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
>
```

Method	Description
db.collection.find()	It is used to retrieve documents from the collection.

```
> db.Students.find({_id:4}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
```

Update Operation

Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfy the given criteria.
db.collection.updateMany()	It is used to update multiple documents in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collection that satisfy the given criteria.

Update Operation

```
> db.student.updateOne({name: "Sumit"},{$set:{age: 24 }})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }
```

```
> db.student.updateMany({}, {$set: {year: 2020}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
```

Delete Operation

Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.

Delete Operation

```
> db.student.deleteOne({name: "Sumit"})
{ "acknowledged" : true, "deletedCount" : 1 }
```

```
> db.student.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 2 }
```

```
> db.stats();
{
        "db" : "db",
        "collections" : 0,
        "views" : 0,
        "objects" : 0,
        "avgObjSize" : 0,
        "dataSize" : 0,
        "storageSize" : 0,
        "totalSize" : 0,
        "indexes" : 0,
        "indexSize" : 0,
        "scaleFactor" : 1,
        "fileSize" : 0,
        "fsUsedSize" : 0,
        "fsTotalSize" : 0,
        "ok" : 1
}
```

```
> help
        db.help()                    help on db methods
        db.mycoll.help()             help on collection methods
        sh.help()                    sharding helpers
        rs.help()                    replica set helpers
        help admin                   administrative help
        help connect                 connecting to a db help
        help keys                    key shortcuts
        help misc                    misc things to know
        help mr                      mapreduce

        show dbs                     show database names
        show collections             show collections in current database
        show users                   show users in current database
        show profile                 show most recent system.profile entries with time >= 1ms
        show logs                    show the accessible logger names
        show log [name]              prints out the last segment of log in memory, 'global' is default
        use <db_name>                set current database
        db.mycoll.find()             list objects in collection mycoll
        db.mycoll.find( { a : 1 } )  list objects in mycoll where a == 1
        it                           result of the last line evaluated; use to further iterate
        DBQuery.shellBatchSize = x   set default number of items to display on shell
        exit                         quit the mongo shell
>
```

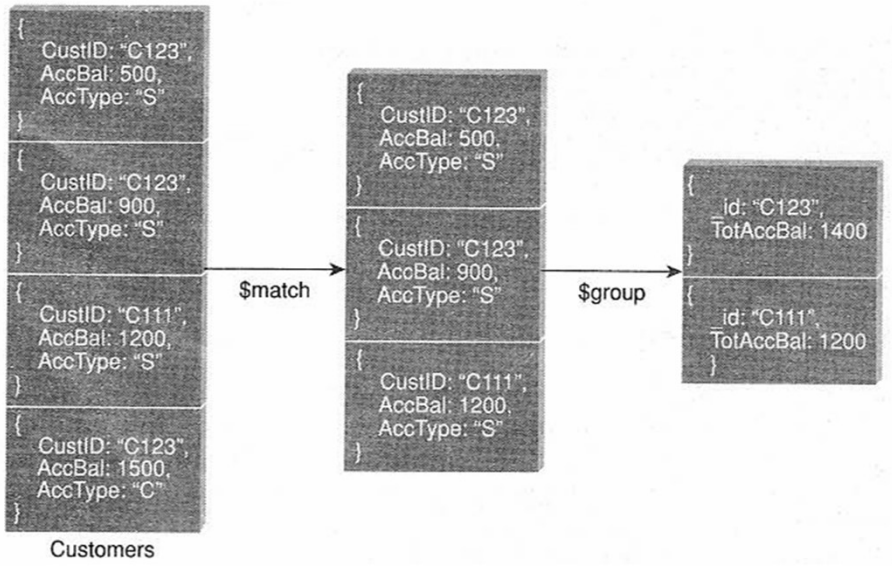
```
mongodb@ubuntu:~/mongodb-linux-x86_64-2.6.0$ bin/mongo
MongoDB shell version: 2.6.0
connecting to: test
> db.help()
DB methods:
    ...
    db.auth(username, password)
    db.cloneDatabase(fromhost)
    db.commandHelp(name) returns the help for the command
    db.copyDatabase(fromdb, todb, fromhost)
    db.createCollection(name, { size : ..., capped : ..., max : ... } )
    db.createUser(userDocument)
    db.currentOp() displays currently executing operations in the db
    db.dropDatabase()
    db.eval(func, args) run code server-side
    db.fsyncLock() flush data to disk and lock server for backups
    db.fsyncUnlock() unlocks server following a db.fsyncLock()
    db.getCollection(cname) same as db['cname'] or db.cname
    db.getCollectionNames()
    db.getLastError() - just returns the err msg string
    db.getLastErrorObj() - return full status object
    db.getMongo() get the server connection object
    ...
```

Array

```
> db.food.insert({id:3,fruits:['Apple','Orange','grapes','cherry']});
WriteResult({ "nInserted" : 1 })
> db.food.find();
{ "_id" : ObjectId("65e86b0ad327dd5cc1ef8bab"), "id" : 3, "fruits" : [ "Apple", "Orange", "grapes", "cherry" ] }
> db.food.find().pretty();
{
        "_id" : ObjectId("65e86b0ad327dd5cc1ef8bab"),
        "id" : 3,
        "fruits" : [
                "Apple",
                "Orange",
                "grapes",
                "cherry"
        ]
}
>
```

Aggregate Function

Objective: Consider the collection "Customers" as given below. It has four documents. We would like to filter out those documents where the "AccType" has a value other than "S". After the filter, we should be left with three documents where the "Acctype": "S". It is then required to group the documents on the basis of CustID and sum up the "AccBal" for each unique "CustID". This is similar to the output received with group by clause in RDBMS. Once the groups have been formed [as per the example below, there will be only two groups: (a) "CustID" : "C123" and (b) "CustID" : "C111"], filter and display that group where the "TotAccBal" column has a value greater than 1200.



```
> db.customers.insert([{CusID:"C123",Accbal:500,AccType:"S"},{CusID:"C123",Accbal:900,AccType:"S"},{CusID:"C111",Accbal:
1200,AccType:"S"},{CustID:"C123",Accbal:1500,AccType:"C"}]);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

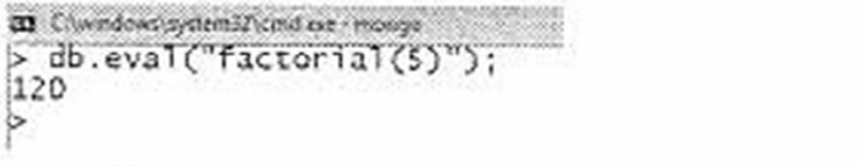
```
> db.customers.find().pretty();
{
        "_id" : ObjectId("65e86ecbd327dd5cc1ef8baf"),
        "CusID" : "C123",
        "Accbal" : 500,
        "AccType" : "S"
}
{
        "_id" : ObjectId("65e86ecbd327dd5cc1ef8bb0"),
        "CusID" : "C123",
        "Accbal" : 900,
        "AccType" : "S"
}
{
        "_id" : ObjectId("65e86ecbd327dd5cc1ef8bb1"),
        "CusID" : "C111",
        "Accbal" : 1200,
        "AccType" : "S"
}
{
        "_id" : ObjectId("65e86ecbd327dd5cc1ef8bb2"),
        "CustID" : "C123",
        "Accbal" : 1500,
        "AccType" : "C"
}
```

```
> db.customers.aggregate({$match:{AccType:"S"}},{$group:{_id:"$CusID",TotalAccBal:{$sum:"$Accbal"}}});
{ "_id" : "C123", "TotalAccBal" : 1400 }
{ "_id" : "C111", "TotalAccBal" : 1200 }
```

Java Script Programming

```
> db.system.js.insert({_id:"factorial",
... value:function(n)
... {
... if(n==1)
... return 1;
... else
... return n*factorial(n-1);
... }
... }
... );
WriteResult({ "nInserted" : 1 })
```

db.eval("factorial(5)");



TEXT BOOKS

Seema Acharya, Subhashini Chellappan, “Big Data and Analytics”, Wiley Publications, First Edition,2015

REFERENCES

1. Judith Huruwitz, Alan Nugent, Fern Halper, Marcia Kaufman, “Big data for dummies”, John Wiley & Sons, Inc. (2013)
2. Tom White, “Hadoop The Definitive Guide”, O’Reilly Publications, Fourth Edition, 2015
3. Dirk Deroos, Paul C.Zikopoulos, Roman B.Melnky, Bruce Brown, Rafael Coss, “Hadoop For Dummies”, Wiley Publications, 2014
4. Robert D.Schneider, “Hadoop For Dummies”, John Wiley & Sons, Inc. (2012)
5. Paul Zikopoulos, “Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw Hill, 2012