Unit I – Introduction to MongoDB

Agenda

- – Introduction to NoSQL Database
- – Why to Use MongoDB
- – Difference between MongoDB & RDBMS
- – MongoDB Download & Installation
- – Common Terms in MongoDB
- – Implementation of Basic CRUD Operations using MongoDB

Introduction to NoSQL

Who coined NoSQL?

- The term **NoSQL** was used by **Carlo Strozzi** in **1998** to name his lightweight, **Strozzi NoSQL open-source relational database.**
- **Johan Oskarsson** of **Last.fm** reintroduced the term **NoSQL** in early **2009** when he organized an event to discuss **"open source distributed, non relational databases".**
- Not Only SQL
- Non-relational, distributed, open-source and horizontally scalable
- Started in 2009
- Characteristics:
  - Schema Free
  - Store huge amount of data
- **More than 156 NoSQL databases are available**

NoSQL databases (A basic classification based on data model)

Wide column stores / Column Family databases

- Hadoop/Hbase
- Cassandra
- Hybertable
- Accumulo
- Amazon SimpleDB
- Cloud Data
- HPCC
- Flink
- Splice

Document Store Database

- MongoDB
- Elastic Search
- Couchbase Server
- CouchDB
- RethinkDB
- RavenDB
- MarkLogic Server
- Clusterpoint server
- NeDB
- Terrastore
- Lotus Notes

Key Value / Tuple Store Database

- Amazon Dynamo DB
- Azure Table Storage
- Riak
- Redis
- Aerospike
- FoundationDB
- LevelDB
- BerkelyDB
- Oracle NoSQL Database
- GenieDB

Multi-model Database

- ArangoDB
- DatomicDB
- FatDB
- AlchemyDB
- CortexDB
- WonderDB

Graph Database

- Allegro
- Neo4J
- InfiniteGraph
- OrientDB
- Virtuoso
- Stardog

CAP Theorem / Brewer’s Theorem

It is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

1. **Consistency** (all nodes see the same data at the same time)
2. **Availability** (a guarantee that every request receives a response about whether it succeeded or failed)
3. **Partition tolerance** (the system continues to operate despite arbitrary partitioning due to network failures)

But, we can select only two of them.

Primary Goal of NoSQL Databases

Horizontal Scalability

Need strong network **partition tolerance.** This requires either giving up **consistency** or **availability**

BASE Approach

NoSQL databases follow BASE principles mentioned below:

1. Basically Available
2. Soft State
3. Eventual Consistency)

Basic Availability. NoSQL databases spread data across many storage systems with a high degree of replication. In the unlikely event that a failure disrupts access to a segment of data, this does not necessarily result in a complete database outage.

Soft State. One of the basic concepts behind BASE is that data consistency is the developer's problem and should not be handled by the database.

Eventual Consistency. The only requirement that NoSQL databases have regarding consistency is to require that at some point in the future, data will converge to a consistent state. No guarantees are made, however, about when this will occur.

Why not ACID properties?

- **ACID** comes from a paradigm of **one database with many users** and that transactions on datasets are made only one at the time have the ability to change a value.**BASE** comes from the **data, which is distributed and synchronized**.
- ACID is pessimistic and forces consistency at the end of every operation, BASE is optimistic and accepts that the database consistency will be in a state of flux.

Why to Use MongoDB

Education, Training and Assessment

Why MongoDB?

MongoDB is Consistent, Partition-Tolerant System

MongoDB : Background

- Mongo comes from "hu***mongo***us" database —with performance and easy data access as core design goals.
- Leading NoSQL database
- Written in C++, Javascript, C
- Released in 2009.
- It is maintained by a company called 10gen as open source and is freely available under the GNU AGPL v3.0 license

MongoDB is:

1. Cross-platform.
2. Open source.
3. Non-relational.
4. Distributed.
5. NoSQL.
6. Document-oriented data store.

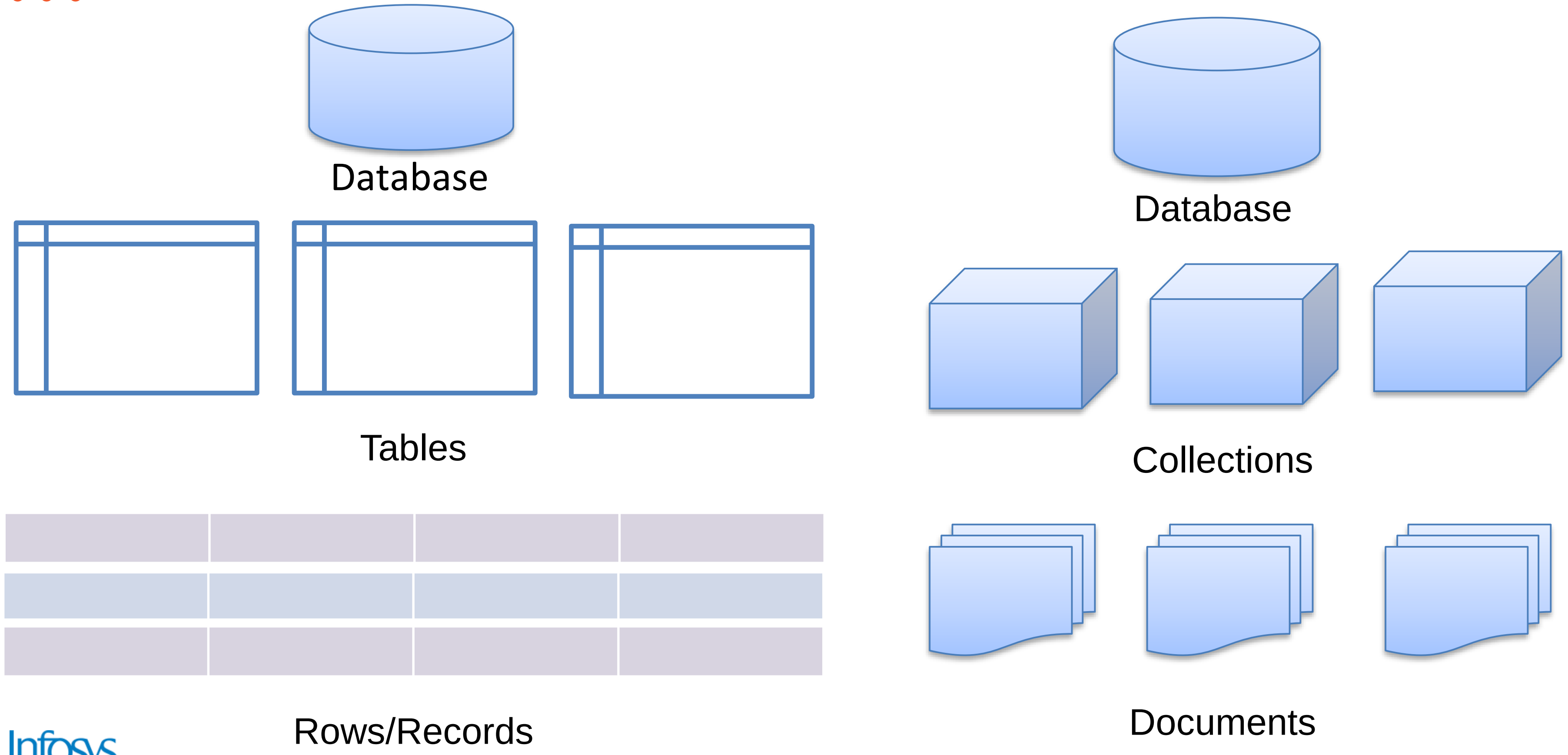
Elements of MongoDB ecosystem

- **High-availability and replication services** for scaling across local and wide-area networks.
- A **grid-based file system (GridFS)**, enabling the storage of large objects by dividing them among multiple documents.
- **MapReduce** to support analytics and aggregation of different collections/documents.
- A **sharding service** that distributes a single database across a cluster of servers in a single or in multiple data centers.
- A **querying service** that supports ad hoc queries, distributed queries, and full-text search.

Difference between MongoDB & RDBMS

Infosys® | Education, Training and Assessment

Traditional Vs MongoDB



MongoDB Download & Installation

Education, Training and Assessment

Installation and Setup

- You can visit the official site to download MongoDB latest version:

  https://www.mongodb.com/

- As it is works on client-server architecture, we need to run both server and client:

- Server - Open command prompt and start MongoDB server using ‘mongod’ command

  – **c:/program files/mongodb/server/3.2/bin> mongod - - dbpath “d:\data\db”**

- Client -- Open command prompt and start MongoDB client using ‘mongo’ command

  – **c:/program files/mongodb/server/3.2/bin> mongo**

MongoDB – Applications

-- bsondump
- Reads contents of BSON-formatted rollback files.

-- mongo
- The database shell.

-- mongod
- The core database server.

-- mongodump
- Database backup utility.

-- mongoexport
- Export utility (JSON, CSV, TSV), not reliable for backup.

-- mongofiles
- Manipulates files in GridFS objects.

-- mongoimport
- Import utility (JSON, CSV, TSV), not reliable for recoveries.

-- mongooplog
- Pulls oplog entries from another mongod instance.

-- mongoperf
- Check disk I/O performance.

--mongorestore
- Database backup restore/import utility.

--mongos
- Mongodb sharding routerprocess.

--mongosniff
- Sniff/traces database activity in real time, Unix-like systems only.

--mongostat
- Returns counters of database operation.

--mongotop
- Tracks/reports MongoDB read/write activities.

Common Terms in MongoDB

Infosys® | Education, Training and Assessment

Basic Commands within the MongoDB Shell

S.No.	Command	Function
1.	show dbs	Shows the names of the available databases
2.	show collections	Shows the collections in the current database
3.	show users	Shows the users in the current database.
4.	use <db name>	Sets the current database to <db name>

JSON (Java Script Object Notation)

Light weight data interchange format

Easy to write and read

It is a text format and language independent

It is built on two structures:

- A collection of name / value pairs
- An ordered list of values

Sample JSON Document

```
{
FirstName: ‘John’,
LastName: ‘Mathews’,
ContactNo: [+123 4567 8900, +123 4444 5555]
}
```

Structure of MongoDB

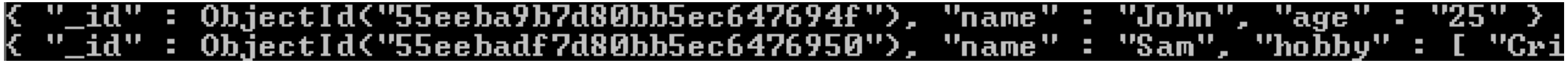
- MongoDB is schema-less
- In MongoDB, collections can have documents with different shapes/sizes

```
{name : “John”,  age:”25”, hobby:[“cricket”, “music”]}
{name: “Sam”, hobby:[“football”, “photography”, “reading”]}
{name: “Tom”}
```

Unique Identifier

- Each JSON document should have a unique identifier. It is the _id key.

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



```
{ "_id" : ObjectId("55eeba9b7d80bb5ec647694f"), "name" : "John", "age" : "25" }
{ "_id" : ObjectId("55eebadf7d80bb5ec6476950"), "name" : "Sam", "hobby" : [ "Cri
```

BSON

- Binary encoded JSON like documents
- It is lightweight, traversable and efficient data format
- MongoDB uses BSON (Binary JSON) for data storage and data transfer
- In MongoDB BSON documents are used for three things:

Data Storage (user documents)

- INSERT command is used to send documents to database
- A user document element name should not begin with $ should not hava a . in the name.
- The _id (element name) is used as a primary key id. You can store anything that is unique in that field

Query "selector" Documents

- These documents are used in QUERY, DELETE and UPDATE operations.
- These documents should use special markers such as $where to query the document in the database

Modifier Documents

- Documents with 'modifier actions' modifies the user documents in the case of an update

BSON in Python

```
>>> from bson import BSON
>>> bson_string = BSON.encode({"hello": "world"})
>>> bson_string
'\x16\x00\x00\x00\x02hello\x00\x06\x00\x00\x00world\x00\x00'
>>> bson_string.decode()
{'hello': 'world'}
```

Support for Dynamic Queries

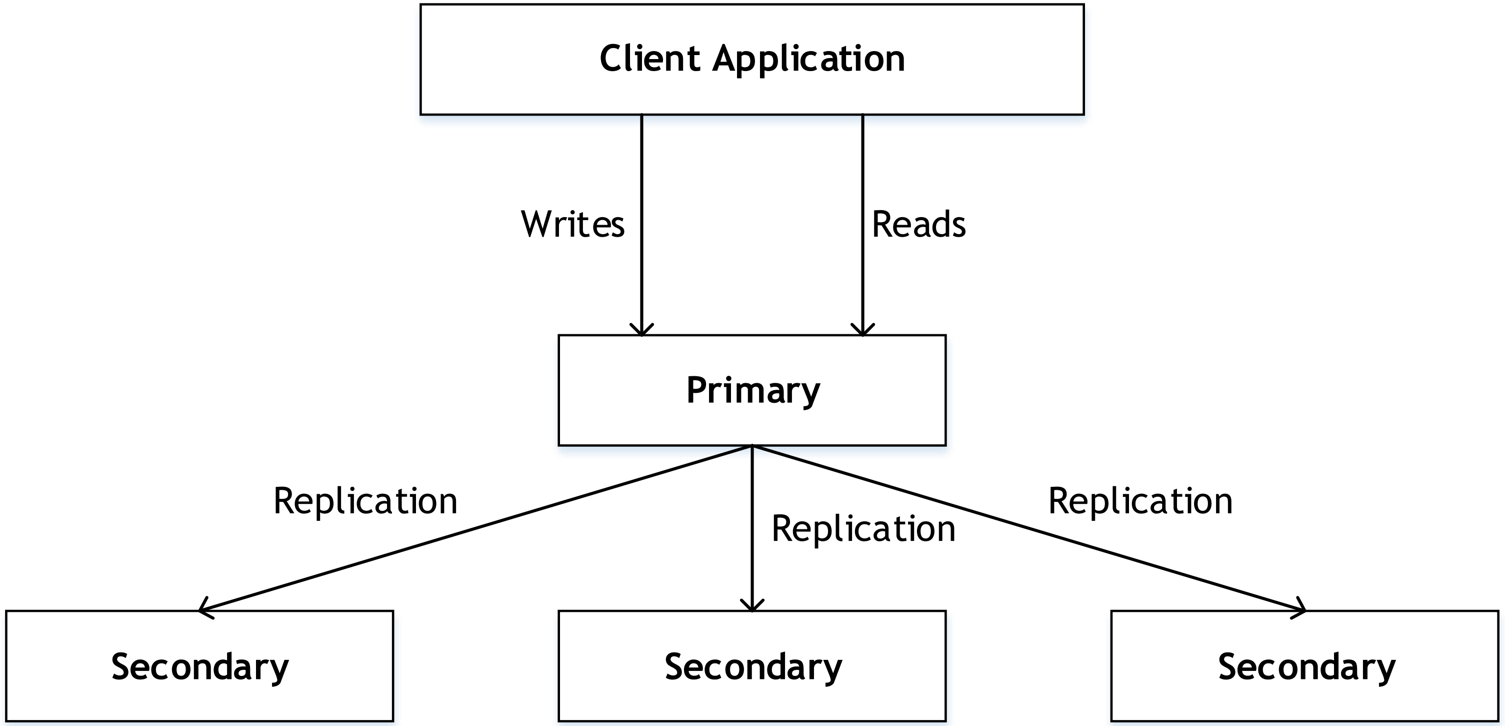
- MongoDB has extensive support for dynamic queries.

- This is in keeping with traditional RDBMS wherein we have static data and dynamic queries.

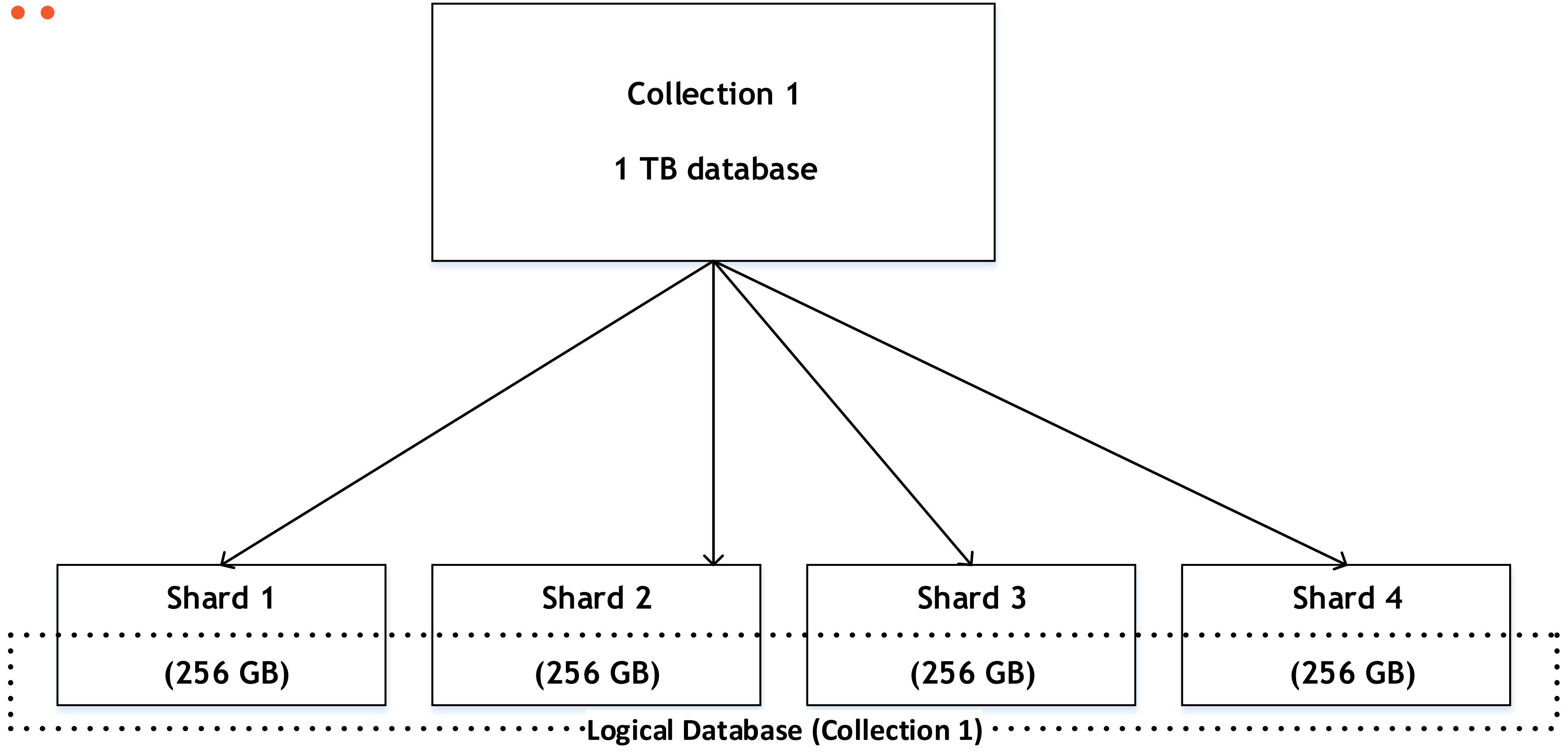
Storing Binary Data

- MongoDB provides GridFS to support the storage of binary data.
- It can store up to 4 MB of data.

Replication in MongoDB



Sharding in MongoDB



Data Types in MongoDB

String	Must be UTF-8 valid. Most commonly used data type.
Integer	Can be 32-bit or 64-bit (depends on the server).
Boolean	To store a true/false value.
Double	To store floating point (real values).
Min/Max keys	To compare a value against the lowest or highest BSON elements.
Arrays	To store arrays or list or multiple values into one key.
Timestamp	To record when a document has been modified or added.
Null	To store a NULL value. A NULL is a missing or unknown value.
Date	To store the current date or time in Unix time format. One can create object of date and pass day, month and year to it.
Object ID	To store the document's id.
Binary data	To store binary data (images, binaries, etc.).
Code	To store javascript code into the document.
Regular expression	To store regular expression.

Database

Creation of Database

>use *roy*

switched to db roy

Drop Database

>db.dropDatabase()

{“dropped” : “roy”, “ok” : 1 }

Implementation of Basic CRUD Operations using MongoDB

Infosys® | Education, Training and Assessment

Working with Collections

- To create a collection by the name “person”. Let us take a look at the collection list prior to the creation of the new collection “Person”.

```
db.createCollection(“person”);
```

- To drop a collection by the name “person”.

```
db.person.drop();
```

Insert Method

- Create a collection by the name “Students” and store the following data in it.

```
db.Students.insert({_id:1, StudName:"Michelle Jacintha", Grade: "VII", Hobbies: "Internet Surfing"});
```

Bulk Insert

Syntax:

```
Bulk.insert(<document>)
```

Example:

```
var bulk = db.items.initializeUnorderedBulkOp();
bulk.insert( { item: "abc123", defaultQty: 100, status: "A", points: 100 } );
bulk.insert( { item: "ijk123", defaultQty: 200, status: "A", points: 200 } );
bulk.insert( { item: "mop123", defaultQty: 0, status: "P", points: 0 } );
bulk.execute();
```

```
bulk.find( { status: "A" } ).removeOne();
bulk.find( { status: "A" } ).update( { $set: { points: 0 } } );
```

Update Method

- Insert the document for “Aryan David” into the Students collection only if it does not already exist in the collection.
- However, if it is already present in the collection, then update the document with new values. (Update his Hobbies from “Skating” to “Chess”.)
- Use “Update else insert” (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

```
db.Students.update({_id:3, StudName:"Aryan David", Grade: "VII"},{$set:{Hobbies: "Skating"}},{upsert:true});
```

Relational Operators

S.No.	MongoDB Operator	Meaning
1	$eq	Equal to
2	$ne	Not equal to
3	$gte	Greater than or equal to
4	$lte	Less than or equal to
5	$gt	Greater than
6	$lt	Less than
7	$in	Looking for an item in the list
8	$nin	Looking for an item not in the list

Find Method

To search for documents from the “Students” collection based on certain search criteria.

```
db.Students.find({StudName:"Aryan David"});
```

To display only the StudName and Grade from all the documents of the Students collection. The identifier _id should be suppressed and NOT displayed.

```
db.Students.find({},{StudName:1,Grade:1,_id:0});
```

To find those documents where the Grade is set to ‘VII’

```
db.Students.find({Grade:{$eq:'VII'}}).pretty();
```

To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

```
db.Students.find ({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();
```

Find Method (Contd…)

To find documents from the Students collection where the StudName begins with “M”.

```
db.Students.find({StudName:/^M/}).pretty();
```

To find documents from the Students collection where the StudName has an “e” in any position.

```
db.Students.find({StudName:/e/}).pretty();
```

To find the number of documents in the Students collection.

```
db.Students.count();
```

To sort the documents from the Students collection in the descending order of StudName.

```
db.Students.find().sort({StudName:-1}).pretty();
```

RDBMS Vs MongoDB

Operation	RDBMS	MongoDB
Insert	Insert into Students (rollno,name,grade,hobby,doj) values(‘101’, ‘Ram’, ‘X’, ‘Singing’, ‘10-06-2013’)	db.Students.insert({_id:1,rollno:’101’,name:’Ram’, grade:’X’,hobby:’Singing’,doj:’10-06-2013’})
Update	Update Students set hobby=‘Cricket’ where rollno=‘101’	db.Students.update({rollno:’101’},{$set:{hobby:’Cricket’}})
	Update Students set hobby=‘Cricket’	db.Students.update({},{$set:{hobby:’Cricket’}},{multi:true})
Delete	Delete from Students where rollno=‘101’	db.Students.remove({rollno:’101’})
	Delete from Students	db.Students.remove({})
Select	Select * from Students	db.Students.find()
	Select * from Students where rollno=‘101’	db.Students.find({rollno:’101’})
	Select rollno, name, hobby from Students	db.Students.find({},{rollno:1,name:1,hobby:1,_id:0})
	Select rollno,name,hobby from students where rollno=‘101’	db.Students.find({rollno:’101’,{rollno:1,name:1,hobby:1,_id:0})

RDBMS Vs MongoDB (Contd…)

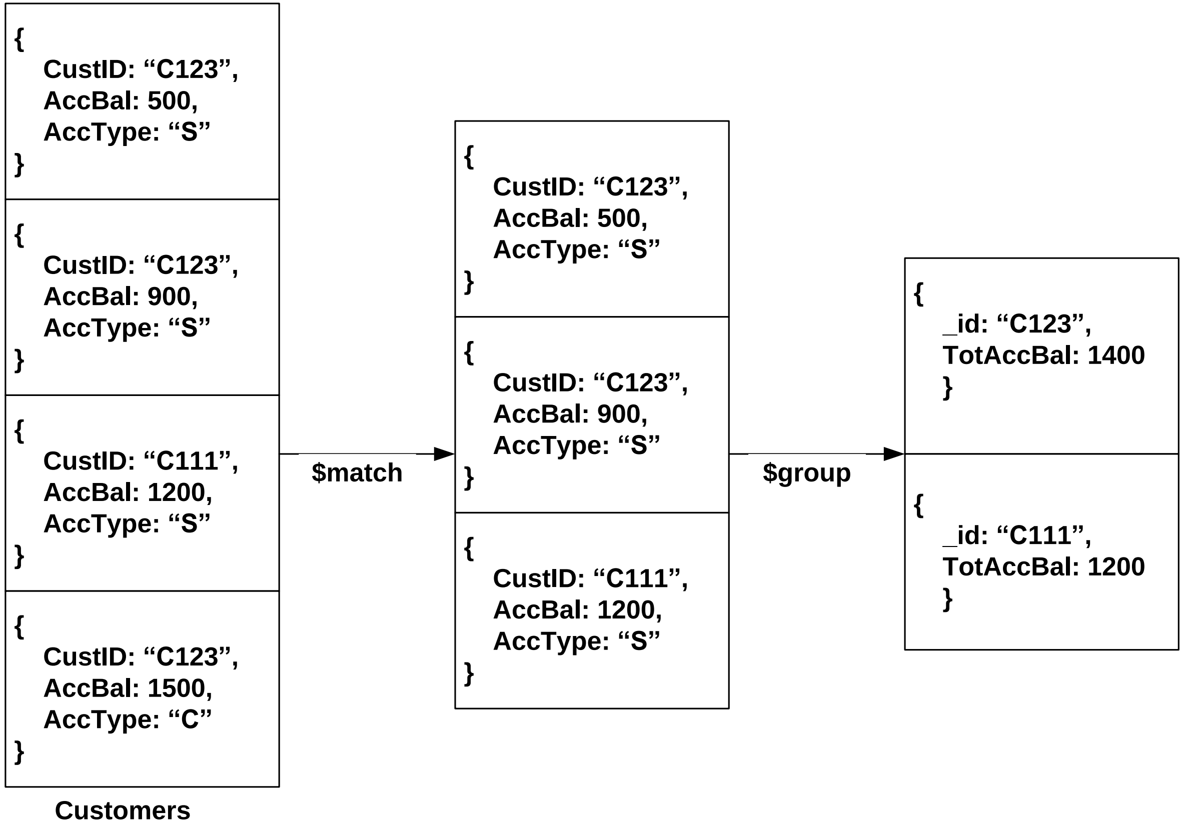
Infosys® | Education, Training and Assessment

Operation	RDBMS	MongoDB
Select	Select rollno,name,hobby from Students where grade='X' and hobby='Cricket'	db.Students.find({grade:'X',hobby:'Cricket'}, {rollno:1,name:1,hobby:1,_id:0})
	Select rollno,name,hobby from Students where grade='X' or hobby='Cricket'	db.Students.find({$or:[{grade:'X',hobby:' Cricket'}], rollno:1,name:1,hobby:1,_id:0})
	Select * from Students where name like 'R%'	db.Students.find({name:/^S/}.pretty()
	Select * from Students where hobby is Null;	db.Students.find({Location:{$eq:null}})
Sort	Select * from Students order by name asc;	db.Students.find().sort({name:1}).pretty( )
	Select * from Students order by name desc;	db.Studetns.find().sort({name:-1}).pretty()

Infosys be more

PUBLIC

Aggregate Function



Aggregate Function

First filter on “AccType:S” and then group it on “CustID” and then compute the sum of “AccBal” and then filter those documents wherein the “TotAccBal” is greater than 1200, use the below syntax:

```
db.Customers.aggregate( { $match : {AccType : "S" } },
{ $group : { _id : "$CustID",TotAccBal : { $sum : "$AccBal" } } },
{ $match : {TotAccBal : { $gt : 1200 } }});
```

$group

Syntax:

{ $group: { _id: <expression>, <field1>: { <accumulator1> : <expression1> }, ... }

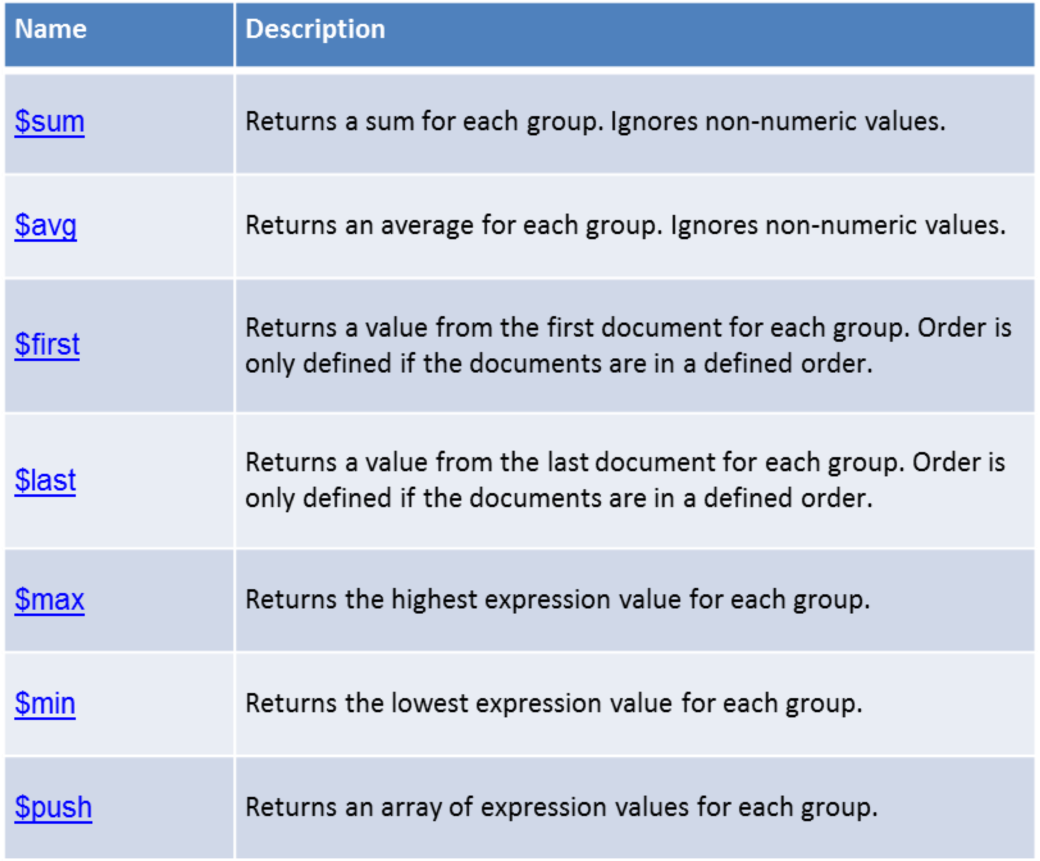
Note: The _id field is mandatory; however, you can specify an _id value of null to calculate accumulated values for all the input documents as a whole

Example:

{ $group : { _id : "$CustID",TotAccBal : { $sum : "$AccBal" } } }

```
db.orders.aggregate([
                  { $match: { status: "A" } },
                  { $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
                  { $sort: { total: -1 } }
                ])
```

Accumulator Operator



Name	Description
$sum	Returns a sum for each group. Ignores non-numeric values.
$avg	Returns an average for each group. Ignores non-numeric values.
$first	Returns a value from the first document for each group. Order is only defined if the documents are in a defined order.
$last	Returns a value from the last document for each group. Order is only defined if the documents are in a defined order.
$max	Returns the highest expression value for each group.
$min	Returns the lowest expression value for each group.
$push	Returns an array of expression values for each group.

Summary

- You are now Knowledgeable on:
  - Scope of NoSQL
  - Need of MongoDB
  - Implementation of CURD operations using MongoDB

Reference

- **Links:**
  - https://www.mongodb.com/
  - https://www.tutorialspoint.com/mongodb/
  - https://docs.mongodb.com/manual/tutorial
  - https://www.javatpoint.com/mongodb-tutorial

- **Videos**:
  - MongoDB Tutorial for Beginners - 1 - Installing Mongo - YouTube

Thank You

Infosys
be more