L7_L10

MongoDB

Agenda

- What is MongoDB?
- Why MongoDB?
  - Using JSON
  - Creating or Generating a Unique Key
  - Support for Dynamic Queries
  - Storing Binary Data
  - Replication
  - Sharding
  - Terms used in RDBMS and MongoDB
- Data Types in MongoDB
- CRUD (Insert(), Update(), Save(), Remove(), Find())
  - MapReduce Functions
  - Aggregation
  - Java Scripting
  - MongoImport
  - MongoExport

What is MongoDB?

MongoDB is:

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

Why MongoDB?

Why MongoDB?

- Open Source
- Distributed
- Fast In-Place Updates
- Replication
- Full Index Support
- Rich Query Language
- Easy Scalability
- Auto sharding

Database – Collection - Document

- Database:
  - Collection of collections
  - Created by a reference or on demand
  - MongoDB → several DBs → Set of files
- Collection:
  - Analogous to a table of RDBMS
  - Created on demand or attempt to save a document that references it
  - Hold several MongoDB documents
  - Does not enforce a schema: no. of fields / order of fields may vary
- Document:
  - Analogous to a tuple in an RDBMS table

JSON

JSON (Java Script Object Notation)

Sample JSON Document

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

{
FirstName: Andrews,
LastName: Symmonds,
ContactNo: +456 7890 1234
}

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		

Support for Dynamic Queries

MongoDB has extensive support for dynamic queries.

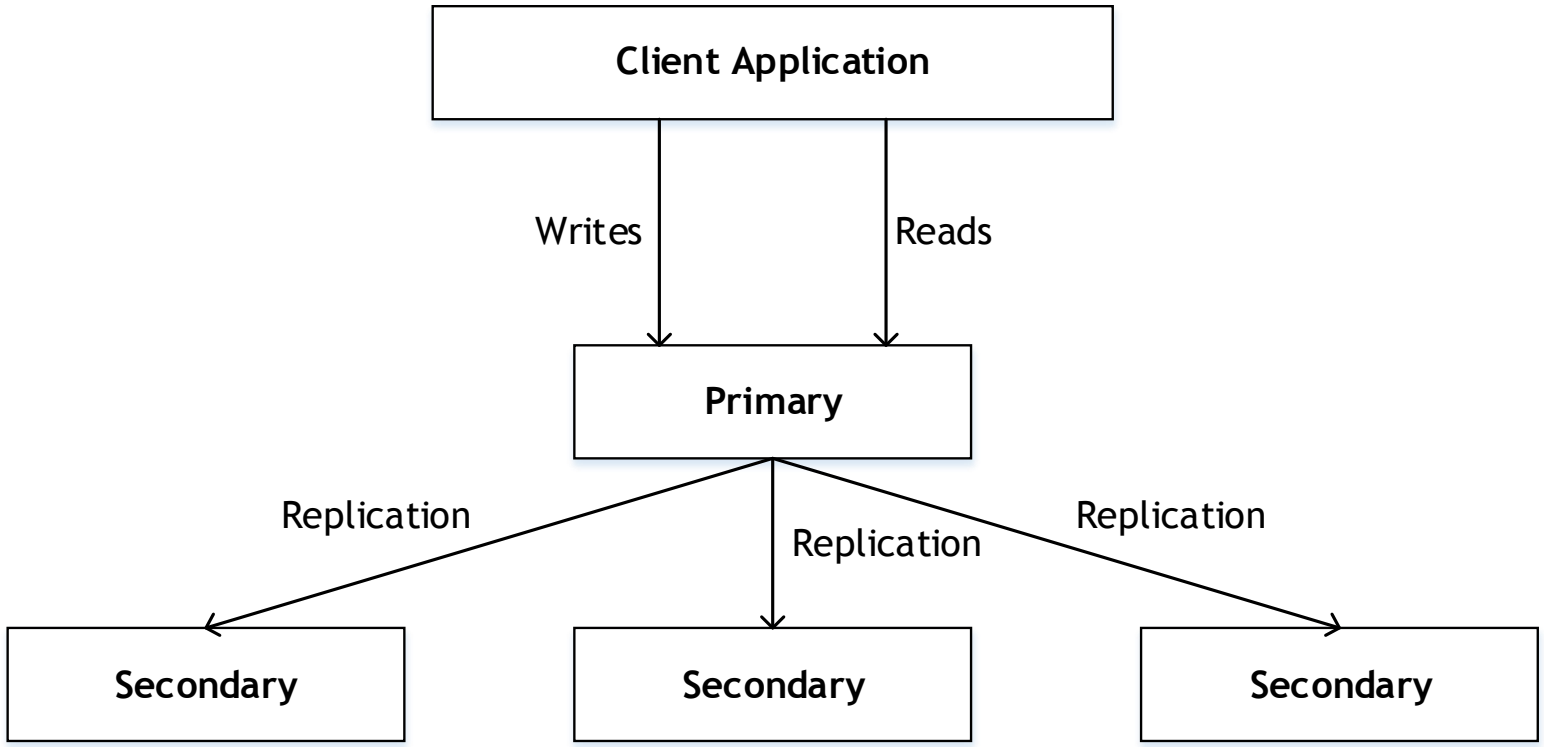
This is in keeping with traditional RDBMS wherein we have static data and dynamic queries.

Storing Binary Data

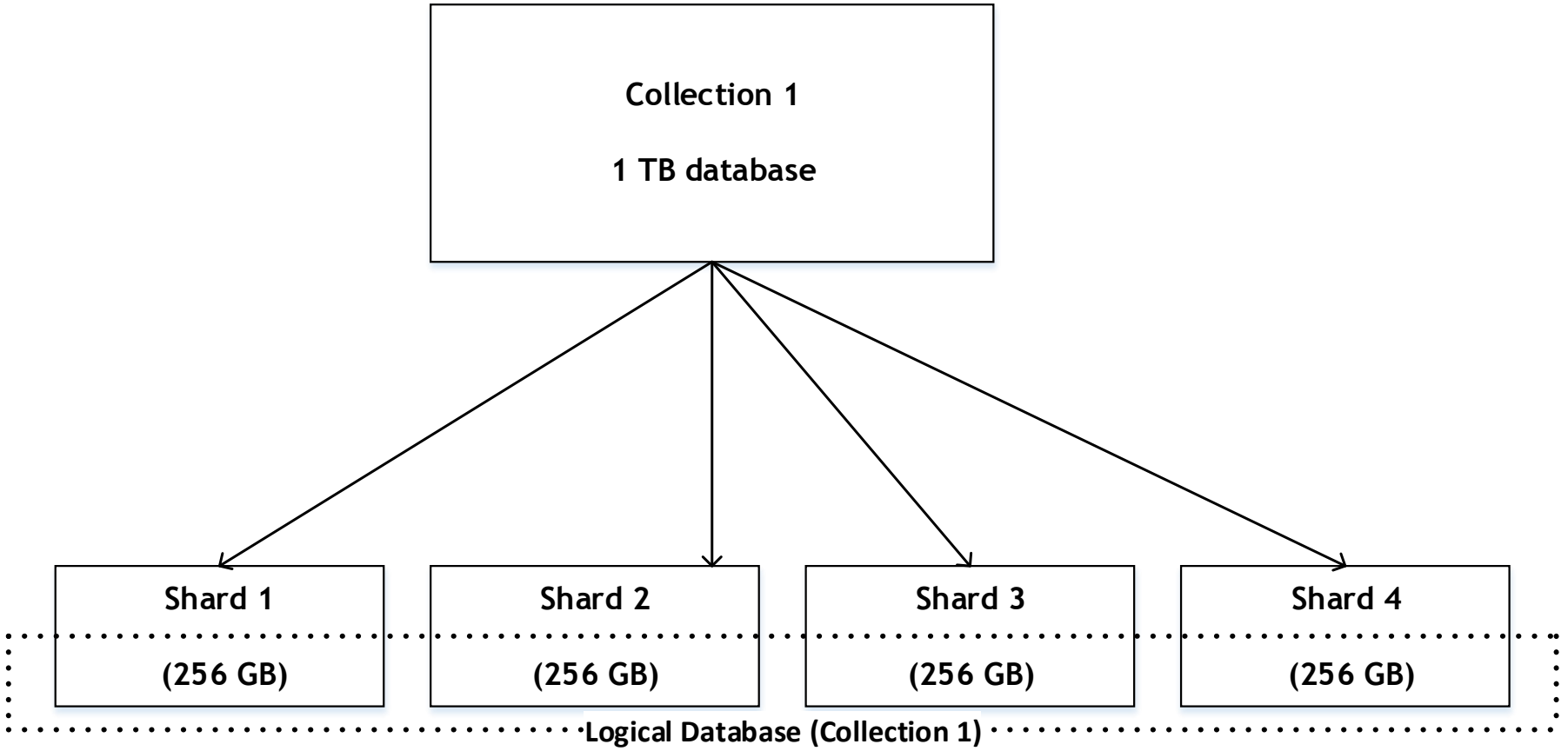
- MongoDB provides GridFS to support the storage of binary data.
- It can store up to 4 MB of data.

- Metadata: file
- Data → chunks → chunks collection → scalability

Replication in MongoDB



Sharding in MongoDB



Terms Used in RDBMS and MongoDB

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)

Data Types 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.

CRUD in MongoDB

Database

- To create a DB
  - use myDB;
- To confirm the existence of DB
  - db;
- To get a list of all DBs
  - show dbs
- To Drop a DB
  - db.dropDatabase();
- To switch to a new DB
  - use myDB1;
- To display the list of collection
  - show collections
- TO display statistics
  - db.stats();

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”);
```

Collections

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"});
```

```
show collections
```

```
db.Students.find();
db.Students.find().pretty();
```

Insert additional records

db.Students.insert({_id:2, StudName:"Mabel Mathews", Grade: “VII", Hobbies: “Baseball"});

Show collections

db.Students.find();

db.Students.find().pretty();

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:false});

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

Update the Hobbies of Aryand David to “Chess”

Find Method

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

db.Students.find();

db.Students.find({_id:3});

db.Students.find({Grade: “VII”});

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

Find Method

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});

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

db.Students.find({Grade:"VII"}).pretty().limit(2);

Save Method

db.Students.save({_id:4,StudName:”Vamsi Bapat", Grade: “VI"});

db.Students.find().pretty();

db.Students.save({_id:4,StudName:”Vamsi Bapat", Grade: “VI“, Hobbies:”Cricket”});

db.Students.find().pretty();

Adding a new field

db.Students.update({_id:4},{$set:{Location: “Bangalore"}})

db.Students.find().pretty();

db.Students.update({Grade: “VII”},{$set:{Location: “Mumbai“}})

db.Students.find().pretty();

db.Students.update({Grade: “VII”}, "},{$set:{Location: “Mumbai"}}, {multi:true})

db.Students.find().pretty();

Removing a document / field

```
db.Students.remove({_id:4})
db.Students.find().pretty();
```

```
db.Students.insert({_id:4,StudName:”VamsiBapat",Grade:“VI“,
Hobbies:”Cricket”});
```

```
db.Students.update({_id:4}, "},{$unset:{Location: “Bangalore"}})
db.Students.find({_id:4}).pretty();
```

Find Method

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

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

Other relational operators:
ne, gt, lt, gte, lte

Find Method

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 ();

db.Students.find ({Hobbies :{ $nin: [‘Cricket']}}).pretty ();

Find Method

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

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

Find Method

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

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

Find Method

To find documents from the Students collection where the StudName ends in "a".

db.Students.find({StudName:/a$/}).pretty();

Find Method

To find documents from the Students collection where id is 3 or 4.

db.Students.find({$or:[{_id:3},{_id:4}]}).pretty();

Find Method

To find documents from the Students collection where location is null.

db.Students.find({Location:null}).pretty();

Find Method

To find the number of documents in the Students collection.

db.Students.count();

db.Students.count({Grade:”VII”});

Find Method

To sort the documents from the Students collection

db.Students.find().sort({StudName:1}).pretty(); // ascending order

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

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

Find Method

To display the records from the Students collection

db.Students.find().skip(2).pretty(); // skip first 2 documents from the output

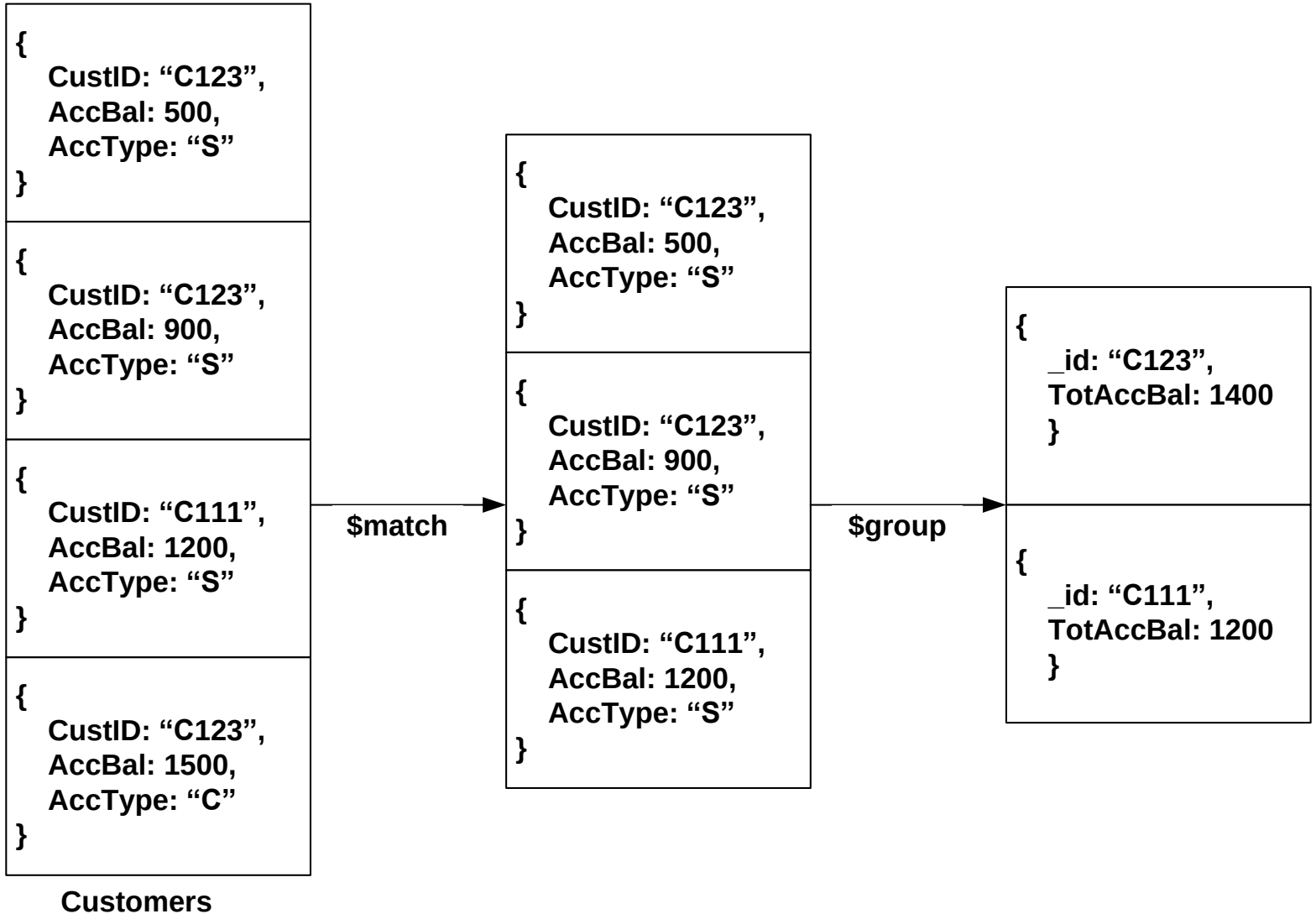
db.Students.find().skip(1).sort({StudName:1}).pretty();

db.Students.find().skip(db.Students.count()-2).pretty();

db.Students.find().skip(2).limit(2).pretty();

Aggregate Function

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:

```
// Insert customers documents
db.Customers.insert([{CustID:"C123",AccBal:500,AccType:"S"},
{CustID:"C123",AccBal:900,AccType:"S"},
{CustID:"C11",AccBal:1200,AccType:"S"},
{CustID:"C123",AccBal:1500,AccType:"C"}]);
```

Aggregate Function

First filter on “AccType:S” and then group it on “CustID” and then compute the sum of “AccBal”

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

Compute average(avg)/maximum(max)/minimum(min) balance

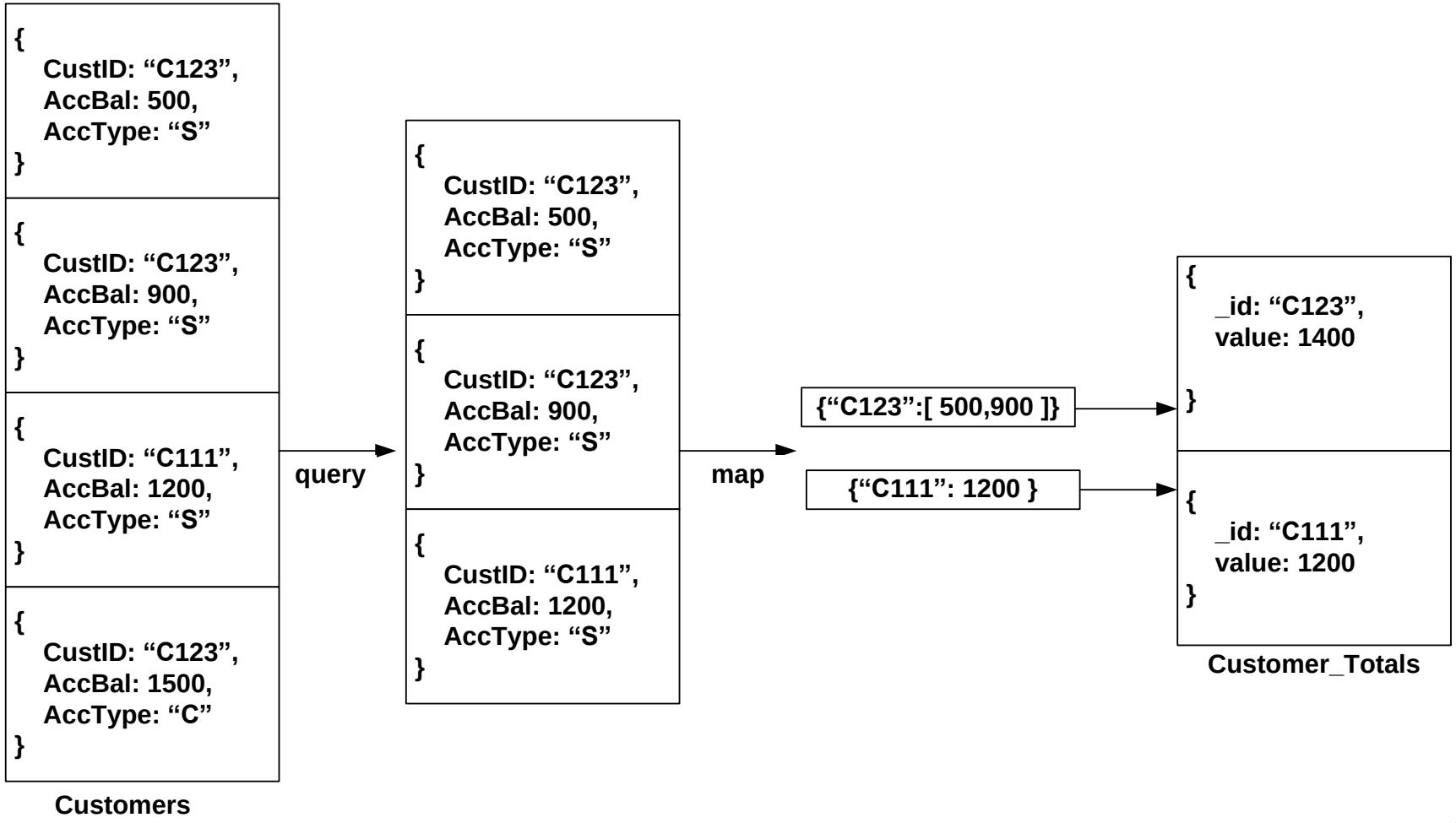
Aggregate Function

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 } }});
```

MapReduce

MapReduce Framework



MapReduce

1. Map function:

```
var map = function() {
emit(this.CustID, this.AccBal); }
```

2. Reduce Function:

```
var reduce = function(key, values) {
return Array.sum(values) ; }
```

MapReduce

1. Map function:

```
var map = function() {
emit(this.CustID, this.AccBal); }
```

2. Reduce Function:

```
var reduce = function(key, values) {
return Array.sum(values) ; }
```

3. **Map-Reduce query:**

```
db.Customers.mapReduce(
map, reduce, {out: “Customer_Totals”, query: {AccType: “S”}}  );
```

4. **Output:**

```
db.Customer_Totals.find();
```

Arrays

Arrays

Create a collection by the name “food” and then insert documents into the foold collection

```
db.food.insert({_id:1, fruits:['banana', 'apple', 'cherry']});
db.food.insert({_id:2, fruits:['orange', 'butterfruit', 'mango']});
db.food.insert({_id:3, fruits:['peneapple', 'strawberry', 'grapes']});
```

Arrays

- Find those collections from the “food” collection which the fruits array constituted of ‘banana’, ‘apple’ and ‘cherry’.

- db.food.find({fruits:['banana','apple','cherry']});

Arrays

- Find those collections from the “food” collection which the fruits array having ‘banana’ as an element.

- db.food.find({fruits: ‘banana’});

Arrays

- Find those collections from the “food” collection which the fruits array having ‘banana’ in the first index position

- db.food.find({‘fruits.0’: ‘banana’});

Arrays

- Find those collections from the “food” collection where size of the array is two.

- db.food.find({fruits: {$size: 2}});

Arrays

- Find collections from the “food” collection and display first two elements

- db.food.find({},{fruits: {$slice: 2}});

Arrays

- Find collections from the “food” collection and display two elements starting with the element at 1st index position

- db.food.find({},{fruits: {$slice: [1,2]}});

Arrays

- Update the element at 0th index position of document with _id:3 by ‘apple’

- db.food.update({_id:3},{$set: {‘fruits.0’: ‘apple’}});

Arrays

- Update the document with _id:1 and push new key value pairs in the fruits array

- db.food.update({_id:1},{$push: {price:{banane: 50, apple:150, cherry:100}}});

Arrays

- Update document with _id:3 by adding an element an ‘Orange’

- db.food.update({_id:3},{$addToSet:{fruits:'Orange'}});

Arrays

- Update document with _id:3 by popping an element

- db.food.update({_id:3},{$pop:{fruits:1}});

Arrays

- Update document with _id:3 by popping an element from the beginning of the array

- db.food.update({_id:3},{$pop:{fruits:-1}});

Arrays

- Update document with _id:2 by popping two elements from the list : ‘orange’ and ‘mango’.

- db.food.update({_id:2},{$pullAll:{fuits:['orange','mango']}});

Arrays

- Update document with _id:2 by popping two elements from the list : ‘orange’ and ‘mango’.

- db.food.update({_id:2},{$pullAll:{fuits:['orange','mango']}});

Arrays

- To pull out an array element based on index position’.

- db.food.update({_id:1},{$unset:{"fuits.1":null}});

- db.food.update({_id:1},{$pull:{"fuits":null}});

Cursors

Cursors

- To create a collection of “a”, “b”,….”z”.by the name “alphabets”

- db.alphabets.insert({_id:1, alphabet: “a”});
- …….
- db.alphabets.insert({_id:26, alphabet: “z”});

- var myCur = db.alphabets.find();
- myCur

Cursors: hasNext(), next()

Diplay all alphabets using cursor:

- var myCur = db.alphabets.find();
- while (myCur.hasNext()) {
- var myRec= myCur.next();
- print("The alphabet is: " + myRec.alphabet); }

Indexes

Indexes

Create an index on AccType for Customers Collection

- db.Customers.ensureIndex(“AccType”:1});
- db.Customers.find({"AccType":"S"}).hint({"AccType":1});

Java Script Programming

Java Script Programming

To compute the factorial of a given positive number. The user is required to create a function by the name “factorial” and insert it into the “system.js” collection.

```
C:\windows\system32\cmd.exe - mongo
> db.system.js.insert({_id:"factorial",
... value:function(n)
... {
...  if (n==1)
... return 1;
...  else
...  return n * factorial(n-1);
...  }
... }
...  );
writeResult({ "nInserted" : 1 })
>
```

```
C:\windows\system32\cmd.exe - mongo
> db.eval("factorial(3)");
6
>
```

MongoImport

Import data from a CSV file

Given a CSV file “sample.txt” in the D: drive, import the file into the MongoDB collection, “SampleJSON”. The collection is in the database “test”.

Mongoimport --db test --collection SampleJSON --type csv --headerline --file d:\sample.txt

MongoExport

Export data to a CSV file

This command used at the command prompt exports MongoDB JSON documents from “Customers” collection in the “test” database into a CSV file “Output.txt” in the D: drive.

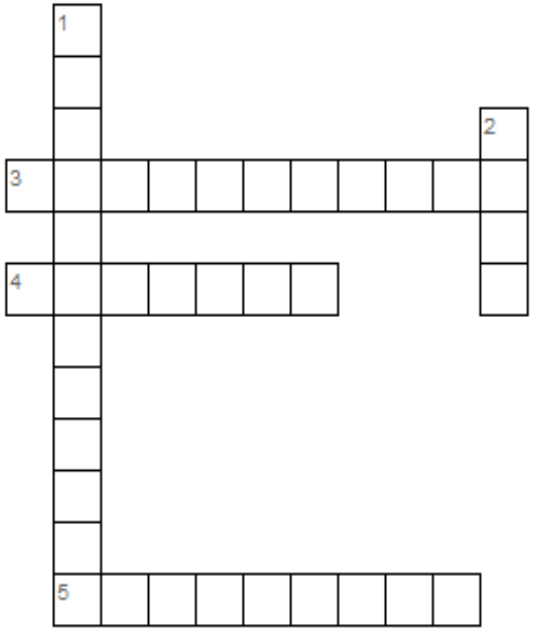
Mongoexport --db test --collection Customers --csv --fieldFile d:\fields.txt --out d:\output.txt

Answer a few quick questions …

Crossword

MongoDB

Basic puzzle on MongoDB



Across

3 MongoDB database stores its data in .....

4 MongoDB uses ---------- schemas.

5 A collection holds one or more -- ----------.

Down

1 MongoDB uses ---------- files.

2 MongoDB uses ----------, a binary object format similar to, but more expensive than JSON.

Answer Me

- What is MongoDB?
- Comment on Auto-sharding in MongoDB.
- What are collections and documents?
- What is JSON?
- Explain your understanding of Update In-Place.

References …

Further Readings

http://www.mongodb.org/

https://university.mongodb.com/

http://www.tutorialspoint.com/mongodb/

Thank you