SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

19CAE730 – Fundamentals of NOSQL database System

II YEAR III SEM

UNIT III –INDEX IN MONGODB

SUM()-Example 1

```
db.student.aggregate  ([{$group :
{_id : "$subject",
marks : {$sum : "$marks"}}}]);
```

SQL Equivalent Query

```
Select subject, sum(marks) from
student  group by subject
```

MIN()

```
db.student.aggregate  ([{$group :
{_id : "$subject",
marks : {$min : "$marks"}}}]);
```

SQL Equivalent Query

```
Select subject, min(marks) from
student  group by subject
```

aggregate() method

Expression	Description
$sum	Sums up the defined value from all documents in the collection.
$avg	Calculates the average of all given values from all documents in the collection.
$min	Gets the minimum of the corresponding values from all documents in the collection.
$max	Gets the maximum of the corresponding values from all documents in the collection.
$first	Gets the first document from the source documents according to the grouping.
$last	Gets the last document from the source documents according to the grouping.

MAX()

```
db.student.aggregate  ([{$group :
{_id : "$subject",
marks : {$max : "$marks"}}}]);
```

SQL Equivalent Query

Select subject, max(marks) from student group by subject

$match

```
db.student.aggregate
([{ $match: {subject:"OSD"}}])
```

```
db.student.aggregate
 ([{ $match: {subject:"OSD"}},
 {$group:{_id:null,count:{$sum:1}}}]);
```

Indexing Types

Single Field Indexes	• A single field index only includes data from a single field of the documents in a collection.
Compound Indexes	• A compound index includes more than one field of the documents in a collection.
Multikey Indexes	• A multikey index is an index on an array field, adding an index key for each value in the array.
Geospatial Indexes	• Geospatial indexes support location-based searches.
Text Indexes	• Text indexes support search of string content in documents.
Hashed Index	• Hashed indexes maintain entries with hashes of the values of the indexed field and are used with sharded clusters to support hashed shard keys.

Index Creation

Using CreateIndex

- Single: db.stud.createIndex({ zipcode: 1})
- Compound: db.stud.createIndex({ dob: 1, zipcode: -1 })
- Unique: db.stud.createIndex({ rollno: 1 }, { unique: true })
- Sparse: db.stud.createIndex({ age: 1 }, { sparse: true })

Using ensureIndex

- Single: db.stud.ensureIndex({“name":1})
- Compound: db.stud.ensureIndex ({“address":1,“name":-1})

LAST()

```
db.student.aggregate
([{$group : {_id :"$subject",
  marks : {$last :"$marks"}}}]);
```

Unwind()

If following document is their in collection(Array)

- db.student.insert({rollno:9,name:"Anavi",marks:[80,30,50]});

Using Unwind the above document will be unwinded into 3 different document

- db.student.aggregate([{$unwind:"$marks"}])

SUM()- Example 3

```
db.student.aggregate
([{ $match: {subject:"OSD"}},
{$group:{_id:null,count:{$sum:1}}}
]);
```

SQL Equivalent Query

```
Select subject, count(*) from
student  group by subject
having subject="OSD"
```

Outline

Indexing

Aggregation

Outline

- Indexing
- Aggregation

Sort()

```
db.student.aggregate
([{ $match:{subject:"OSD"}},
{$sort:{marks:-1}}]);
```

```
db.student.aggregate
([{ $match:{subject:"OSD"}},
{$sort:{marks:1}}]);
```

Limit() & Skip()

```
db.student.aggregate
([{ $match:{subject:"OSD"}},
{$limit:1}]);
```

```
db.student.aggregate
([{ $match:{subject:"OSD"}},
{$skip:1}]);
```

Indexing and Querying

db.stud.find({}, {name:1,age:1}), using index {name:1,age:1}

db.c.find().sort({name:1,age:1}), using index {name:1,age:1}

db.stud.update({age:20}, {age:19}) using index {age:1}

db.stud.remove({name: “Jiya”}) using index {name:1}

Indexing and Querying

db.stud.find({}, {name:1,age:1}), using index {name:1,age:1}

db.c.find().sort({name:1,age:1}), using index {name:1,age:1}

db.stud.update({age:20}, {age:19}) using index {age:1}

db.stud.remove({name: “Jiya”}) using index {name:1}

Indexing and Querying

db.stud.find({}, {name:1,age:1}), using index {name:1,age:1}

db.c.find().sort({name:1,age:1}), using index {name:1,age:1}

db.stud.update({age:20}, {age:19}) using index {age:1}

db.stud.remove({name: “Jiya”}) using index {name:1}

AVG()

```
db.student.aggregate  ([{$group :
 {_id : "$subject",
marks : {$avg : "$marks"}}}]);
```

SQL Equivalent Query

```
Select subject, avg(marks) from
 student  group by subject
```

SUM(): Example 2

```
db.student.aggregate  ([{$group :
{_id : "$subject",
Count: {$sum : 1}}}]);
```

SQL Equivalent Query

```
Select subject, count(*) from
student  group by subject
```

Indexing

Indexes support the efficient execution of queries in MongoDB.

Possible stages in aggregation

- **$project** – Used to select some specific fields from a collection.
- **$match** – This is a filtering operation and thus this can reduce the amount of documents that are given as input to the next stage.
- **$group** – This does the actual aggregation as discussed above.
- **$sort** – Sorts the documents.
- **$skip** – With this, it is possible to skip forward in the list of documents for a given amount of documents.
- **$limit** – This limits the amount of documents to look at, by the given number starting from the current positions.
- **$unwind** – This is used to unwind document that are using arrays. When using an array, the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thus with this stage we will increase the amount of documents for the next stage.

FIRST()

```
db.student.aggregate
([{$group : {_id :"$subject",
marks : {$first :"$marks"}}}]);
```

Pipeline Concept

There is a set of possible stages and each of those is taken as a set of documents as an input and produces a resulting set of documents

MongoDB
Aggregation and Indexing

Index Creation

Using CreateIndex

- db.CollectionName.createIndex({ KeyName: 1 or -1})

Using ensureIndex

- db.CollectionName.ensureIndex({KeyName: 1 or -1})

1 for Ascending Sorting

-1 for Descending Sorting

Index Properties

Index Properties
The properties you can specify when building indexes.

TTL Indexes The TTL index is used for TTL collections, which expire data after a period of time.

Unique Indexes A unique index causes MongoDB to reject all documents that contain a duplicate value for the indexed field.

Sparse Indexes A sparse index does not index documents that do not have the indexed field.

Index Drop

Syntax

- db.collection.dropIndex()
- db.collection.dropIndex(*index*)

Example

- db.stud.dropIndex()
- db.stud.dropIndex({ “name" : 1 })

Indexing with Unique

db.collectionname.ensureIndex
({x:1}, {unique:true})

- Don't allow {_id:10,x:2} and {_id:11,x:2}
- Don't allow {_id:12} and {_id:13} (both match {x:null}

What if duplicates exist before index is created?

- Normally index creation fails and the index is removed
- db.ensureIndex({x:1}, {unique:true,dropDups:true})

Index Display

db.collection.getIndexes()

- Returns an array that holds a list of documents that identify and describe the existing indexes on the collection.

db.collection.getIndexStats()

- Displays a human-readable summary of aggregated statistics about an index's B-tree data structure.
- db.<collection>.getIndexStats({ index : "<index name>" })

Aggregation

- MongoDB's aggregation framework is modeled on the concept of data processing pipelines.
- Documents enter a multi-stage pipeline that transforms the documents into an aggregated result.
- Other pipeline operations provide tools for grouping and sorting documents by specific field or fields.
- In addition, pipeline stages can use operators for tasks such as calculating the average or concatenating a string.

Aggregation

- MongoDB's aggregation framework is modeled on the concept of data processing pipelines.
- Documents enter a multi-stage pipeline that transforms the documents into an aggregated result.
- Other pipeline operations provide tools for grouping and sorting documents by specific field or fields.
- In addition, pipeline stages can use operators for tasks such as calculating the average or concatenating a string.

Collection creation to run practical

- db.student.insert({Rollno:1,name:'Navin ',subject:'DMSA',marks:78});
  db.student.insert({Rollno:2,name:'anusha',subject:'OSD',marks:75});
  db.student.insert({Rollno:3,name:'ravi',subject:'TOC',marks:69});
  db.student.insert({Rollno:4,name:'veena',subject:'TOC',marks:70});

db.student.insert({Rollno:5,name: 'Pravini',subject: 'OSD',marks:80}); db.student.insert({Rollno:6,name: 'Reena',subject: 'DMSA',marks:50});

db.student.insert({Rollno:7,name: 'Geeta',subject: 'CN',marks:90});
db.student.insert({Rollno:8,name: 'Akash',subject: 'CN',marks:85});

Refernces

- https://bhavanakhivsara.wordpress.com
- https://www.tutorialspoint.com/mongodb/mongodb_aggrega_tion.htm
- http://pradipshewale.blogspot.in/2015/09/aggregation-and-_indexing-with-suitable.html
- https://www.infoq.com/articles/implementing-aggregation-_functions-in-mongodb
- http://docs.mongodb.org/manual/reference/operator/aggreg_ation/