Aim: Design and develop mongo DB, Queries using CRUD operations (use CRUD operations SAVE method, logical operators.

Problem Statement: Create an Artide conertion and perform basic operations (CRUD) like Insert, Display, Update and delete documents.

Objective:

To understand basic CRUD operations, use SAVE method, logical Operations (and, not, or)

Software Requirement : OS - linux Ubuntu 18 (64 bit)
Database MongoDB

Theory:

i) Mongo DB is document - oriented database program widely classified as NOSQL database program.

ii) In Mongo DB CRUD operations refers for creating, reading, updating and deleting of documents

1) CREATE

Create a new documents in which we can create or insert operations to collection IF the collection doesnot currently exist insert operation will create the collection.

- db.collection.insertone()
- db.collection.insertmany()

SAVE method:

The db.collection.save() method is used to update an existing document or insert a new documents depending on its document parameter.

Parameters

Name	Description	Required / Optional
document:	A document to save to the connection.	Required
write concern	A document expressing the write concern	Optional.

FIND

- Read Operations retrieve documents from a collection query collection query collection for documents.

db.collection-name.find().

you can specify query filters or criteria that identify the documents to return.

- Comparison Operators.

$eq match the equal value.
$gt match the greater than value.
$gte match the greater than or equal value
$lte match the less than or equal value
$ne match the not equal value.
$in match any of the value specified
$nin Match the name of the value

3) UPDATE

update operations modify existing documents on collection

Syntax : db. collection Name. update
(<query / condition>
< update with $ set or $ unset >,
{
upset : <boolean>,
multi : <boolean>
}
)

4) DELETE

Delete operations remove documents from collection mongoDB provide the following method to delete documents of Collection.

db. collection . detete one( )
db. collection. delete many( )

Conclusion

Understanding the CRUD operations in Mongo DB.

Aim : Implement aggregation and indexing with suitable example using mongoDB

Objective : To study the indexing in mongoDB

Outcome : student will able to implement aggregation and indexing with suitable example using mongo DB

Software Requirement :- Os linux Ubuntu 18 (64 bit)
Database - MongoDB

Theory :-

Index supports the efficient execution of queries in Mongo DB without Indexes mongoDB must scan every document in collection to select those documents that match the larger volume of data than an index for each operation

Fundamentals indexes in mongo DB uses similar to indexes in other database system. Mongo DB defines Indexes at collection level and support indexes on any field of sub field of document in Mongo DB collection.

In mongo shell, you can create an Index by calling the createIndex() method.

Arguments to ensureIndex() resemble the following.

```
{ "field" : 1 }
{ "product quantity" : 1 }
{ "product" : 1 , "quantity" : 1 }
```

Unique Indexes.

- Unique Indexes causes mongo DB to reject all documents that contain duplicate value for Indexed field to create a unique on user_id fields of member convertion use the following operation in mongo

```
db.address.ensureIndex ( { "user_id" : 1 }
                         { unique : true } )
```

- Sparse Indexes

Sparse Indexes only contains entries for documents that have the indexed field. Any document that is missing the field is not indexed.

The Index is sparse because of missing documents in ~~cor~~ convection and state

null values for documents that do not contain indexed field.

Aggregation

Once data is stored in MongoDB you may want to do more than just retrieve it.

a) The aggretian pipeline
b) The map reduce function.
c) Single purpose aggregation.

Several pipeline expressions

There are several expression available.

$ group.
Grouping allows you to group documents based on certain fields and combine their values.

$ unwind
turns each field into seperate document.

$ sort
you can sort by any field "normal". it isn't efficient for langes skips.

$ limit.

$ limit take a number n and returns the first n resulting documenting.

Conclusion:

Understand the operations of aggregation and indexing in mongo DB

Aim : Mongo DB - Map reduces operations

Problem statement : create an order collection with fields customer-id, order date, status price and items, quantity

Objective : To study map reduce operation is mongodb to perform complex Aggregation tasks

Outcomes : 1) understand Map reduce operation in mongo DB
2) perform complex aggregation tasks with map reduce

Software Requirement : OS - linux ubuntu 18 (64 bits)
Database - Mongo DB

Theory :

Map reduce is powerful and flexible tool for aggregating data. It can solve some problems that are ~~not~~ too complex to express using the aggregation from query language.

This element is returned to shuffle step until each was containing single

value the results

```
>map = function () {
    for (var key in this)
{ int (key, { count : 1) } };
}}
```

Now we have a ton of little

{ count : 13
documents floating around, each each associated with key from collection on array of one or more of these { count : 13 documents will be passed to reduce function.

```
> reduce = function. (key, emits)
{
.... total = 0;
    for (var in emits)
  {
      total + = emits [i] . count;
      return ( "count {" total };
  }
```

Reduce trust be able to be called repeatedly on result from either the map phase or previous reduce phases.

```
> r1 = reduce ("x" , ({ count : 1, id : 1 },
    { count : 1, id :}
    { count : 2 }
> r2 = reduce ("x" , [{ count : 1 , id : 33}
    { count : 1 }
> reduce ("x" , [r1 , r2])
    { count : 3 }
```

You commet depend on second argument always helding one of initial documents

~~Algo~~ Altogether this map reduce function would look like

```
> m = db.run command ({ "map reduce" :
        " map" map , "reduce" : reduce})
  {
    " result : " tmp.mapreduce - 1266781-1"}
      " time millis : 12,
    " counts " : {
      " input " : 6
      " emit " : 14
```

```
    "output" : 5
}
    "ok" : true
}
```

Conclusion:

1) Understand the operations of Map reduce in mongoDB
2) understand the working and performance of complex Aggression tasks in map reduce.