Name: Atharva Mahamuni
Enrollment no: 2002276
class: Msc CS-1
Subject: Advance Database Concepts

Theory Assignment -3

Q1. Explain the features of MongoDB

→

① Supports ad hoc queries
In MongoDB, you can search by field, range query and it also supports regular expressions.

② Indexing
you can index any field in a document.

③ Replication
MongoDB supports master slave replication A master can perform Reads and writes and a slave copies data from the master and can only be used for reads or back up (not writes)

④ Duplication of data
MongoDB can run over multiple servers. The data is duplicated to keep system up and also keep it in running condition in case of failure

⑤ Load Balancing
It has an automatic load balancing configuration because, data is placed in shards.

⑥ supports MapReduce and Aggregation tools

⑦ Uses Javascript instead of procedures

⑧ It is a schema-less database written in C++

⑨ Provides high performance.

⑩ Easy to administer in case of failure.

PAGE No.
DATE / /

Q. 2. Explain CRUD in MongoDB with suitable syntax and example.

Ans. → * MongoDB CRUD operations.

CRUD operations create, read, update and delete documents.

*Create operations.

→ Create or insert operations add new documents to a collection. If the collections does not currently exist, insert operations will create the collection.

→ MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne()
- db.collection.insertMany().

→ example:

```
db.users.insertOne(                          ← collection
  {
    name : "sue",                            ← field : value
    age : 26,                                ← field : value   } document
    status : "pending"                       ← field : value
  }
)
```

*Read operations.

→ Read operations retreive documents from a collection i.e. query a collection for documents. MongoDB provide the following methods to read documents from collection.

- db.collection.find()

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

```
db.users.find(                    ← collection
  { age: { $gt: 18 } },           ← query criteria
  { name: 1, address: 1 }         ← projection
).limit(5)                        ← cursor modifier.
```

* Update Operations.

→ Update operations modify existing documents in a collection. MongoDB provides following methods to update documents of a collection:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

→ In MongoDB, updates operations target a single collection. All write operations in MongoDB are atomic on the level of a single collection.

```
db.users.updateMany(              ← collection
  { age: { $lt: 18 } },           ← update filter.
  { $set: { status: "reject" } }  ← update action.
)
```

* Delete Operations.

→ Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection.

- db.collection.deleteOne()
- db.collection.deleteMany()

→ In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of single document.

```
db.users.deleteMany(              ← collection
  { status: "reject" }            ← delete filter.
)
```

Q. 3. Explain sorting in mongoDB with suitable examples.

Ans. →

* sort() Method of mongoDB

→ For sorting your mongoDB documents, you need to make use of the sort() method.

→ This method will accept a document that has a list of fields and the order for sorting. For indicating the sorting order, you have to set the value 1 or 1 with the specific entity based on which ordering will be set and displayed.

→ One indicates organizing data in ascending order while -1 indicates organizing in descending order.

* Syntax:

→ The basic syntax for the sort() method is:

```
db.collection_name.find().sort({FieldName1: sort order 1 or -1,
    FieldName2 : sort order})
```

* Example:

→ Consider a collection that is having following data:

```
{"_id": objectId(5983548781331adf45ec5), "topic" : "CS" }
{"_id": objectId(556548781331ae745ec6), "topic" : "Digi Privacy"}
{"_id".objectId(5983549391331adf45ec7), "topic": "App Security"}
```

→ Suppose I want get data only from the topic field from all the documents in ascending order, then it will be executed like this:

→ Example:

```
db.techSubjects.find({}, {"topic":1, _id:0}).sort({"topic":1})
```

Q. 4. Explain skip() and limit() in mongoDB with suitable example.

Ans → * The limit() method in mongoDB :

→ This method limits the number of documents returned in response to particular query.

* Syntax :

```
db.collection_name.find().limit(number_of_documents)
```

* Using limit() method to limit documents in result :

→ Do not want all the documents matching the criteria want only selected number of documents then I can use limit() method to limit number of documents.

→ example :

```
db.studentdata.find({student_id:{$gt:2002}}).limit(1).pretty
{
    "_id" : ObjectId("59bf63500be1d7770c3982b0"),
    "student_name" : "carol",
    "student_id" : 2003,
    "student_age" : 22
}
```

* mongoDB skip() method :

→ The skip() method is used for skipping the given number of documents in query result.

→ To understand the use of skip() method, lets take the same example that have seen above. In above example we can see that by using limit(1) we managed to get only one document, which is first document that matched given criteria.

PAGE No.
DATE / /

* Using skip:

```
db.studentdata.find({student_id :{$gt :2002}}).limit(1).
        skip(1).pretty()
{
   "_id" : ObjectId("59bf63650be1d777c0c3982b1"),
  "student_name" : "Tim";
  "student-id" : 2004,
  "student_age" : 23,
}
```

Q. 5. Explain Aggregation in mongoDB with syntax & example.

Ans. →

→ Aggregation operations process data records and returns computed results.

→ Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return single result.

→ mongoDB provides three ways to perform aggrega° the aggregation pipeline, the map-reduce function, and single purpose aggregation methods.

* Aggregation Pipeline:

→ MongoDB's aggregation framework is modeled on the concept of data processing pipelines.

→ Documents enter a multi-stage pipeline that transform the documents into an aggregated result.

* Example:

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

→ First stage : The $match stage filters documents by the status field and passes to the next stage those documents that have status equal to "A".
→ Second stage: The $group stage groups documents by the cust_id field to calculate the sum of the amount for each unique cust_id.

* Single Purpose Aggregation Operations:
→ MongoDB also provides db.collection.estimatedDocumentCount(), db.collection.count() and db.collection.distinct().
→ All of these operations aggregate documents from a single collection.

Q. 6. How regex can be used in mongoDB?
Ans. → * Definition:
$regex:
→ Provides regular expression capabilities for pattern matching strings in queries. MongoDB uses Perl compatible regular expressions (i.e. "PCRE") version 8.42 with UTF-8 support.
→ To use $regex, use one of the following syntaxes:
{<field> : { $regex : /pattern/, $options : '<options>' }}
{ <field>: { $regex : 'pattern', $options : '<options>' } }
{ <field> : { $regex : /pattern/ <options> }}

* In MongoDB, you can also use regular expression objects (i.e. /pattern/) to specify regular expressions:

{<field> : /pattern/ <options> }

PAGE No.
DATE / /

* $options
→ The following <options> are available for use with reg.ex.
- i - case sensitivity to match
- m - match at beginning or end of each line for string
- x - extended capibility.
- s - Allows dot character to match all characters.

* Behavior:
$regex vs. /pattern/ syntax.
→ $ in Expressions.
To include a regular expression in an $in query exp, you can only use JavaScript regular expressions objects (i.e. /pattern/). For example,
{name : { $in : [/^acme/i, /^ack/]}}

→ You cannot use $regex operator expressions inside an $in.

* Implicit AND conditions for the field.
→ To include a regular expression in a comma-seperated list of query conditions for the field, use the $regex operator. For example,
{name : { $regex : /acme.*corp/i, $nin : ['acmeblahcorp']}}
{name : { $regex : /acme.*corp/, $options : 'i', $nin : ['acmeblahcorp']}}
{name : { $regex : 'acme.*corp', $options : 'i', $nin : ['acmeblahcorp']}}

* x and s options:
→ To use either than x option or s options, you must use the $regex operator expression with the $options operator.
{name : { $regex : /acme.*corp/, $options : "si" }}
{name : { $regex : 'acme.*corp', $options : "si" }}

Q.7. Explain the usage of indexing in MongoDB.

Ans. → *Indexing in MongoDB

→ MongoDB uses indexing in order to make the query processing more efficient.

→ If there is no indexing, then MongoDB must scan every document in the collection and retrieve only those documents that much query.

→ The indexes are order by the value of field specified in index.

*Creating an Index:

→ MongoDB provides a method called createIndex() that allows user to create an index.

•Syntax:

```
db.COLLECTION_NAME.createIndex({KEY:1})
```

→ The key determines the field on basic of which you want to create an index and 1 (or -1) determines the order in which these indexes will be arranged (ascending or descending).

•Example

```
db.mycol.createIndex({"age":1})
{
"createdCollectionAutomatically" : false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
"ok" : 1
}
```

* The createIndex() method also has a number of option parameters. These include:

•background (Boolean)

- unique (Boolean)
- name (string)
- sparse (Boolean)
- Drop an Index.

Q. 8. How cursor can be used in mongoDB?

Ans. → * Cursor

→ When the db.collection.find() function is used to search for documents in the collection, the result returns a pointer to the collection of documents returned which is called a cursor.

→ By default, the cursor will be iterated automatically when the result of query is returned.

→ But one can also explicitly go through the items returned in cursor one by one.

→ The following example shows how this can be done:

```
var myEmployees = db.Employee.find({Employeeid:{$gt:2}});
  while (myEmployee.hasNext())
    {
      print(tojson(myEmployee.next()));
    }
```

Q. 9. What is map reduce in mongoDB?

Ans. →

→ As per the mongoDB documentation, map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results.

→ mongoDB uses mapReduce command for map-reduce operations.

→ map-Reduce is generally used for processing large data sets.

* map Reduce command.

→ Following is the syntax of the basic mapReduce command

```
db.collection.mapReduce(
    function() {emit(key, value); },   //map function
    function(key, values) {return reducefunction},
                                        //reduce function.
    out : collection,
    query : document,
    sort : document,
    limit : number,
  }
)
```

→ The map-reduce function first queries the collection, then maps the result document to emit key-value pairs, which is then reduced based on the keys that have multiple values.

* In above syntax,

→ map is a javascript function that maps a value with a key and emits a key-value pair.

→ reduce is a javascript function that reduces or groups all documents having same key.

→ out specifies the location of map-reduce query result

→ query specifies optional selection criteria for selecting documents.

→ sort specifies the optional sort criteria.

→ limit specifies the optional maximum number of documents to be returned.

PAGE No.
DATE / /

6. Distributed Database concepts.

Q. 1. What is Distributed Database? Write characteristics of Distributed Database.

Ans. → * Definition:

→ A distributed database (DDB) is an integrated collection of database that is physically distributed across sites in a computer network.

→ To form a distributed database system (DDBS), the files must be structured, logically interrelated, and physically distributed across multiple sites.

* A DDBMS has following characteristics:

→ a collection of logically related shared data.

→ the data is split into a number of fragments

→ fragments may be replicated.

→ fragments/replicas are allocated to sites.

→ the sites are linked by a communication network

→ the data at each site is under the control of DBMS.

→ the DBMS at each site can handle local applications autonomously.

→ each DBMS participates in at least one global application.

Q 3. Write the advantages and disadvantages of Distributed DBMS.

Ans. → * Advantages of DDBMS:

→ The database is easier to expand as it is already spread across multiple systems and it is not too complicated to add a system.

→ The database can be stored according to department information in an organisation. In that case, it is easier for a organisational hierarchial access.

→ It is cheaper to create a network of systems containing a part of the databases. This database can also be easily increased or decreased.

→ Even if the some of the data nodes go offline, the rest of the database can continue its normal functions.

* Disadvantages of DDBMS:
→ The distributed database is quite complex and it is difficult to make sure that a user gets a uniform view of the database because it is spread across multiple locations.
→ This database is more expensive as it is complex and hence, difficult to maintain.
→ The distributed database is complicated and it is difficult to find people with necessary experience who can manage and maintain it.

Q. 4. What are the types of DDBMS?

Ans. →

* Types of Distributed Database:
→ Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with fuvthur sub-divisions, ~~as show~~.

* Homogeneous Distributed Databases:
→ In a homogeneous distributed database, all the sites use identical DBMS and operating systems.

* Types of Homogeneous distributed databases
→ Autonomous
→ Non-autonomous.

* Heterogeneous Distributed Databases:
→ In a heterogeneous distributed database, different sites having different operating systems, DBMS products and data models.

* Types of Heterogeneous Distributed Databases.
→ Federated
→ Un-federated.

Q. 5. Explain the architecture of Distributed Databases.

Ans. →

* Distributed DBMS Architectures:
→ DDBMS architectures are generally developed depending on three parameters-
- Distribution - It states physical distribution of data across the different sites.
- Autonomy - It indicates distribution of control of database system and the degree to which each constituent DBMS can operate independently
- Heterogenity - It refers to uniformity of data models, system components and databases.

* Architectural Models
→ Some of the common architectural models are-
- Client - Server Architecture for DDBMS.
- Peer - to - Peer Architecture for DDBMS
- Multi - DBMS Architecture.

Q. 6. Which are the design strategies of DDBS?

Ans. →

* Design strategies in Distributed Database management system (DDBMS):
→ These are two Approaches or Design Strategies in Distributed Database management system for

developing any database, the top-down method and bottom-up method.

→ While these approaches appear radically different, they share the common goal of utilizing system by describing all of the interaction between the processes.

1. TOP-down Design method.

→ The top-down design method starts from the general and moves to the specific.

→ In other words, you start with a general idea of what is needed for system and then work your way down to the more specific details of how the system will interact.

→ This process involves the identification of different entity types and the definition of each entity's attributes.

2. Bottom-up Design Method.

→ The bottom-up approach begins with specific details and moves up to the general.

→ This is done by first identifying the data elements (items) and then grouping them together in data sets.

→ In other words, this method first identifies the attributes, and then groups them to form entities.

Q. 7. What is data allocation? Explain various allocation strategies.

Ans → * Data Allocation

→ Data Allocation Problem (DAP) in distributed database system is a NP-hard optimization problems with great importance in distributed environments.

→ In DAP problem the main objective is to assign set of fragments to set of sites in order to minimize the total cost of transactions.

→ The main cost in the system comes from data transactions across the distributed system.

→ The optimization problems are significant challenge in majority of engineering applications in which researches have been extensively solving DAP as challenging problem by applying soft computing methods especially evolutionary algorithm due to their capability in extracting good solutions in an acceptable computational time.

Q. 8. What is data replication? Explain various replication schemes.

Ans. → * Data Replication in DBMS:

→ Data Replication is the process of storing data in more than one site or ~~more~~ node.

→ It is useful in improving the availability of data. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency.

→ The result is a distributed database in which users can access data relavant to their tasks without inter-ferring with work of others.

* Replication Schemes :

1. Full Replication: The most extreme case is replicat of the whole database at every site in distributed system.

2. No Replication: The other case of replication involves having No replication - that is, each fragment is stored at only one site.

3. Partial Replication: In this type, of replication some fragments of the database may be replicated whereas others may not. The description of replication of fragments is sometimes called replication schema.

Q. 9. What is Fragmentation? What are the types of fragmentation?

Ans. → * Fragmentation:

→ Fragmentation is the task of dividing a table into a set of smaller tables.

→ The subsets of the table are called Fragments.

→ Fragmentation can be of three types : horizontal, vertical and hybrid (combination horizontal & vertical).

* Vertical Fragmentation:

→ In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstriveness, each fragment should contain the primary key field(s) of table.

→ Vertical fragmentation can be used to enforce privacy of data.

* Horizontal Fragmentation:
→ Horizontal fragmentation groups of tuples of table in accordance to value of one or more fields.
→ Horizontal fragmentation should also confirm to rule of reconstructiveness. Each horizontal fragment must have all columns of original base table.

* Hybrid Fragmentation:
→ In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used.
→ This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information.
→ However, reconstruction of the original table is often an expensive task.

10. Explain Horizontal fragmentation with suitable example.

S. →

* Horizontal Fragmentation:
→ Horizontal fragmentation groups the tuples of table in accordance to value of one or more fields.
→ Horizontal fragmentation should also confirm to rule of reconstructiveness. Each horizontal fragment must have all columns of original base table.
→ For example, in the student schema, if details of all students of Computer Science course needs to be maintained at school of Computer Science,

then the designer will horizontally fragment the database as follows

```
CREATE COMP_STD AS
    SELECT * FROM STUDENT
    WHERE COURSE = "COMPUTER SCIENCE".
```

Q. 11. Explain Vertical Fragmentation with suitable Example.

Ans. →

* Vertical Fragmentation.

→ In vertical fragmentation the fields or columns of a table are grouped into fragments. In order to maintain reconstriveness, each fragment should contain the primary key field(s) of table.

→ vertical fragmentation can used to enforce the privacy of data.

→ For example, let us consider that a University database keeps records of all registered students in student table having following schema.

→ STUDENT

Regd-NO	Name	course	Address	Semester	Fees	marks

Now the fees details are maintained in accounts section. In this case, the designer will fragment database as follows :-

```
CREATE TABLE STD-FEES AS
    SELECT Regd-No, Fees
    FROM STUDENT;
```

PAGE No.
DATE / /

Q. 12. Explain Hybrid Fragmentation with suitable example.

Ans → * Hybrid Fragmentation:

→ In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used.

→ This is most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of original table is often an expensive task.

→ Hybrid fragmentation can be done into two alternative ways:

- At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.
- At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.