Week 09 & 10

1) Illustrate ACID transaction in MongoDB

ACID stands for Atomicity, consistency, Isolation, and Durability.

These Properties ensure reliablity of the database transaction

- Atomicity

ALL operations in a transaction are executed as a single unit - either all succeed or none. It means if any operation is performed on the data, either it should be executed completely or should not be executed at all.

Example : If transferring ₹500 from Account "A" to Account "B":

* Debit ₹500 from A
* Credit ₹500 ~~from~~ to B

If the second operation fails the first one is rolled back.

- Consistency

A transaction takes the database from one valid state to another valid state, enforcing the rules. Consistency guarantees that changes made within a transaction are consistent with database constraints.

Example : In a banking system, a rule is defined that an account balance cannot go below ₹0.

If an account has ₹1,000 and the customer tries to withdraw ₹2,000, the balance would become -₹1,000. This violates the rule, so the transaction must be rejected to maintain consistency.

- **Isolation**

Isolation means keeping transactions seperate so they don't interfere with each other. The changes are not visible to other transaction until commited.

Example: If two people try to withdraw from the same account at the same time, one transaction waits until the other finishes.

- **Durability**

Once a transaction is commited, the changes are permanent, even in case of power loss or crash. In case of a crash, the system can revert to the last commited state, to preserving data integrity

2) Compare MySQL and MongoDB

MySQL	MongoDB
• It is relational database	• It is non-relational database
• suitable for the structured data with clear schema	• Suitable for unstructured data
• Represents data as tables & rows	• Represents data as JSON DOCS

• Must specify tabels and columns	• No need to declare the schema
• Allows Join operations	• Doesn't allow Join operations
• utilizes structured query language	• Query language is JavaScript
• More suitable when you are just starting the data base which doesn't scale much	• More suitable when you need high availability of data
• There is no auto failure recovery	• It supports auto failure recovery
• It is slower	• It is faster than MySQL
• works better for smaller data sets	• works better for larger datasets.

3) Demonstrate CRUD operation in : MongoDB.

~~the basic methods of~~

The basic methods of interacting with a MongoDB server are called CRUD operations.

CRUD stands for

C -> Create

R -> Read

U -> Update

D -> Delete

• Create operation

It is used to insert new documents into the collection. If a collection doesnot exist, then it will create a new collection in the database.

insertOne(): It is used to insert a single document in the collection.

Syntax:

```
db.collection_name.insertOne({key1:"value1", key2:
                                "value2"});
```

Ex:

```
db.students.insertOne({ name: "Geethesh",
                        age : 20,
                        branch : "CS",
                        grade : "A",
                        Subjects : ["Maths", "FOC", "English"],
                        feespaid : 30,000
                        attendence : 81
                      });
```

insertMany(): It is used to insert multiple documents in the collection.

Syntax:

```
db.collection_name.insertMany([{document 1},
                               {document 2 ...},
                               {ordered : true/false}])
```

Ex:

```
db.students.insertMany([{ name : "Deekshith L",
                          age : 20,
                          grade : "A",
                          branch : "CSE",
                          Subjects : ["Fullstack", "math"],
                          feespaid : 16,000
                        },
```

```
{ name : "Ajith Birwa",
  age : 18,
  branch : "CSE",
  feespaid : 10000
}])
```

- **Read operation**

The read operation are used to retrive documents from the collection.

findOne(): It is used to retrive a singel document that matches the query criteria.

Syntax:

```
db.collection_name.findOne({key : value});
```

Ex:

```
db.students.findOne({name : "sujnan"});
```

find(): It is used to retrive multiple documents from the collection.

Syntax:

```
db.collection_name.find({key : value});
```

Ex:

```
db.students.find({age : {$gte : 20}})
```

- **Update Operation**

It is used to modify the existing documents in the collection.

updateOne(): It is used to update a single document in the collection that satisfy given criteria

Syntax:

db.collection_name.updateOne({key : value}, {$set : {key to update : "newvalue"}});

Ex:

db.students.updateOne({name : "Suresh"}, {$set : {grade : "A+"}})

updateMany(): It is used to update multiple documents in the collection that satisfy criteria.

Syntax:

db.collection_name.updateMany({key : value}, {$set : {key to update : "newvalue"}});

Ex:

db.students.updateMany({age : {$age : 20}}, {$set : {grade : "B+"}});

replaceOne(): it is used to replace the single document in the collection that satisfy the criteria.

Syntax:

db.collection_name.replaceOne({key : value} {key1 : "new value1", key2 : "newvalue2"});

Ex:

```
db.students.replaceOne({ name: "Pavan Rav"},
                        { name: "Pavan Rav",
                          age: 19,
                          branch: "CSE",
                          grade: "A+"
                          subjects: ["It skill", "Math"],
                          feesPaid: 1000
                        });
```

- **Delete operations:**

It is used to remove the document from a collection.

~~Syntax~~:

deleteOne(): It is used to delete a single document in the collection that satisfy the given criteria.

Syntax:

```
db.collection_name.deleteOne({key:"value"});
```

Ex:

```
db.students.deleteOne({ name: "Nishanth"});
```

deleteMany(): It is used to delete the multiple documents from the collection that satisfy the given criteria.

Syntax:

```
db.collection_name.deleteMany({key:value});
```

Ex:

```
db.students.deleteMany({age:{$lte:20}});
```

4) CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database. Explain how the create operation is used to insert new documents in the MongoDB database.

= In MongoDB the create operation allows you to insert new documents into a collection.

- InsertOne()
- InsertMany()

InsertOne()

The insertOne() methods adds a single document to a specific collection. If the collection doesn't exists, MongoDB creates it automatically. If _id is not given, MongoDB generate a unique object Id.

Syntax:

db.collection_name.insertOne(document)

Ex:

db.Products.insertOne({item:"Pen", price:10, quantity:50});

A documents with fields item, Price, and quantity is added to the Products Collection

insertMany()

The insertMany() methods allows the insertion of multiple documents in the single operation

Syntax:

db.collectionName.insertMany([{document 1},
{document 2},

Syntax

```
db.collectionName.insertMany([{document 1},
                              {document 2},
                              ...], {ordered: true/false})
```

Example

```
db.student.insertMany([{name: "Anand", age:21},
                       {name: "Bhavya", age:20},
                       {name: "Kavitha", age:19}
                       ]);
```

This command inserts 3 documents into the student collection in one operation. Each document recieves a unique _id if not explicitly provided.

{ordered: false} allows remaining documents to be inserted even if one fails.

5) Perform the following operations using MongoDB

i) Create an employee database

ii) Insert a records into the employee collection with suitable fields

iii) Sort the employee by name

iv) Search the employee by Id

v) Remove the employee Id

i) In MongoDB, we can create a Database using the use command. using this command, we can also switch from on database to another

```
use EmployeeDB
```

ii) The two documents to employee collection can be added using <u>insertMany()</u> method. It is used to insert multiple documents in the collection.

```
db.employee.insertMany([{ id:1, name:"Anand",
                          emp-dep:"HR"},
                        { id:2, name:"Bhavya",
                          emp-dep:"Finance"}])
```

iii) The documents of employee collection are sort by using the <u>find().sort()</u> methods. The Ascending or descending criteria is mentioned using 1 and -1 respectivly.

```
db.employee.find().sort({name:1})
```

iv) The document of employee collection is retrived using the <u>find()</u> method. The attribute passed to <u>find()</u> form the basis for searching the document.

```
db.employee.find({id:1})
```

v) A single document of employee collection is removed by using the <u>deleteOne()</u> method that satisfy the given criteria.

```
db.employee.deleteOne({id:1})
```

6) Perform the following operations using MongoDB

i) Create a student database

In MongoDB, we can create a database using the use command. using this command we can also switch from one database to another

```
use studentDB
```

ii) Insert 2 records into the student collection with suitable fields

The two documents to student collection can be added using insertMany() method. it is used to insert multiple documents in the collection.

```
db.student.insertMany([{_id:1,
                        name: "Anand",
                        age : 20,
                        course : "Computer science"
                        },
                        {_id:2,
                        name: "Bhavya",
                        age : 21,
                        course: "Electronics"
                        }])
```

iii) Sort the student by the name

The documents of student collection is sorted by using the find().sort() method. The attribute was to sort() method form the basis for sorting the document. The ascending and the descending criteria is mentioned using 1 and -1 respectively.

db.student.find().sort({name : 1})

iv) search the student by Id

The document of student collection is retrived using the find method. The attributes passed to find() method form the basis for searching the document.

db.student.find({_id : 1})

v) Remove the student by Id

A single document of student collection is removed by using the deleteOne() method that satisfy the given criteria

db.student.deleteOne({id : 1})

7) An organization needs to maintain its employee details in database. illustrate how MongoDB is very effective.

i) creating & dropping database

ii) Creating & dropping collections

iii) Creating user & Roll

iv) Removing an employee details based on employee id.

v) creating five documents in employee collection

i) • Creating the database

use database_name is used to create the database it will create a new database if it doesnot exist, otherwise it will return the existing database.

```
use organizationDB
```

• Dropping the database

db.dropDatabase() method is used to drop the database it removes the current database, deleting all associated files.

```
db.dropDatabase()
```

ii) • Creating the collection

db.createCollection(name, options) is used to create collection in mongoDB

```
db.createCollection("users")
```

• Dropping the collection

db.collection.drop() is used to remove a collection from database. it also removes any indexies associated with drop collection.

```
db.users.drop()
```

iii) In MongoDB, we can also create a user for single Database using create user method. ~~db.create~~

```
db.createUser({ user : "Rakesh",
                Pwd  : "Pass123",
                role : [{ role : "userAdmin",
                          db : "organizationDB"}]})
```

iv) A single document of employee collection is removed by using the <u>deleteOne()</u> method that satisfy the given criteria

```
db.employee.deleteOne({employee_Id : 123})
```

v) <u>insertMany()</u> method is used to insert multiple documents in the collection

```
db.employee.insertMany([{_id : 1,
                         name : "Avith",
                         age : 20,
                         Place :
```

```
db.employee.insertMany([{ emp_id : 1,
                          name : "Avith",
                          dep : "HR",
                          Salary : 50,000},
                        { emp_id : 2,
                          name : "Chirag",
                          dep : "Finance",
                          Salary : 20,000}
                        { emp_id : 3,
                          name : "Geetresh",
                          dep : "IT",
                          Salary : 12,000}
```

{ emp-id : 4,
name : "Sunan",
dep : "Marketing",
salary : 19,000 }
{ emp-id : 5,
name : "Alvin",
dep : "operations",
salary : 9,000 }])

8) The write operations on the collections are very high. Explain the technic applicable to manage the given scenario.

when the write operations on the collections are very high, it is a good practice to use db.collection.bulkwrite() method to handle the repeated write operations on the collections.

The db.collection.bulkWrite() method provides the ability to perform bulk insert, update, and delete operations.

bulkwrite() supports the following write operations:

- insertOne
- updateOne
- updateMany
- replaceOne
- deleteOne
- deleteMany

Example:

db.iceCreams.insertMany([
{_id:0, flavor: "vanilla", size: "small", Price: 4},
{_id: 1, flavor: "Chocolate", size: "medium", Price: 7},
{_id: 2, flavor: "Strawberry", size: "large", Price: 8}.
])

The following bulkWrite() example runs these operations on the icecream collections:

- Adds two document using insertOne
- Update a document using updateOne
- Delete a document using deleteOne

```
db.iceCream.bulkWrite([
{insertOne: {document: {id:3, flavor: "mango",
                size: "medium", Price:6}}},
{insertOne: {document: {id:4, flavor: "Almond",
                size: "Large", Price:10}}},

{updateOne: {
   filter: {flavor: "Chocolate"},
   update: {$set: {Price: 8}}
}},
{deleteOne: {filter: {flavor: "vanilla"}}}])
```

9) List and explain aggregate functions in MongoDB

: Aggregate functions group value from multiple documents and perform a variety of operations on the grouped data to return a single results.

Example:

```
db.sales.insertMany([
{_id:1, item: "apple", quantity:5, Price:10, date:"2023-01-05"},
{_id:2, item: "banana", quantity:8, Price:6, date:"2023-01-05"},
{_id:3, item: "apple", quantity:3, Price:10, date:"2023-01-10"},
{_id:4, item: "orange", quantity:6, Price:7, date:"2023-01-08"},
{_id:5, item: "apple", quantity:7, Price:12, date:2023-01-01},
])
```

1) $sum:

It define values from all documents in the Coulection

Ex:

db.sales.aggregate([{$group: {_id: "$item", totalQuantity: {$sum: "$quantity"}}}])

2) $avg:

The average of all given values from all document in the collection.

Ex:

db.sales.aggregate([{$group: {_id: "$item", avgPrice: {$avg: "$Price"}}}])

3) $min:

gets the minimum of caresponding values from all documents in the collection.

Ex:

db.sales.aggregate([{$group: {_id: "$item", minQuantity: {$min: "$quantity"}}}])

4) $max:

Gets the maximum of cariesponding value from all documents in the Collection

Ex:

db.sales.aggregate([{$group: {_id: "$item", maxPrice: {$max: "$Price"}}}])

5) $Push

Adds all values to an array including the duplicates values.

Ex:

db.sales.aggregate([{$group: {_id: "$item", Prices: {$Push: "$Price"}}}])

6) $addToset:

Adds only unique values to an array.

Ex:

db.sales.aggregate([{$group: {_id: "$item", uniquePrice: {$addToset: "$Price"}}}])

7) $first

Returns the first document according to the grouping.

Ex:

db.sales.aggregate([{$sort: {date: 1}}, {$group: {_id: "$item", firstSaleDate: {$first: "$date"}}}])

8) $last

Returns the last document ~~agg~~ according to the grouping.

Ex:

db.sales.aggregate([{$sort: {date: 1}}, {$group: {_id: "$item", lastSaleDate: {$last: "$date"}}}])