26/3/24

LAB-2

Perform the following DB operations using MongoDB

1. Create a database 'Student' with the following attributes Roll no, Age, Contact no, Email-Id

→ db.createCollection("Student");
{ok : 1}

2. Insert appropriate values

→ db.student.insert({RollNo: 1, Age: 21, cont: 9876, email: "abc.6@gmail.com"});

db.Student.insert({RollNo: 2, Age: 21, cont: 5576, email: ab@

db.Student.insert({RollNo: 10, Age: 10, cont: 2276, email: "pb@gmail"});

db.student.find()

3. Write query to update Email-id of student whose rollno 10

→ db.Student.update({RollNo: 10}, {$set: {email: nuku@gmail.com}})

4. Replace the student name from "ABC" to "FEM" of rollno 11

→ db.Student.insert({Rollno: 11, Age: 22, Name: "ABC", cont: 2276, email: "de@gmail.com"});

db.student.update({Rollno: 11, Name: "ABC"}, {$set: {Name: "FEM"}})

5. Perform the following DB operations using MongoDB

2.

1. Create a collection by name customers with the following attribute Cust_id, Acc_bal, Acc_type.

→ db.createCollection ("customers");

2. Insert atleast 5 values into the table

→ db.Customer.insert({cust_id:1, acc_bal:2300, acc_type: "A"});
db.customer.insert({cust_id:2, acc_bal:23000, acc_type: "B"});
db.customer.insert({cust_id:3, acc_bal:8300, acc_type: "C"});
db.customer.insert({cust_id:4, acc_bal:8300, acc_type: "D"});
db.customer.insert({cust_id:5, acc_bal:3300, acc_type: "2"});

3. Write a query to display those records whose total account balance is greater than 1200 of account type '2' for each element of customer_id

→
db.customer.find({acc_bal: {$gt: 1200}, acc_type: "2"})

output

{_id: ObjectId('660293663dddd6488'),
cust_id: 5,
acc_bal: 3300,
acc_type: '2'

4. Determine min and max balance for each customer_id.

→ db.customer.aggregate([{$group: {_id: "$cust_id", min_balance: {"$acc_bal"}, max_balance: {$max: "$acc_bal"}}}])

LAB-3

1) Operations on Student collection

5. Display Student Name and grade if old (grade is not present) where the _id column is 1.

→ db.Student.update({RollNo: 1}, {$set: {Grade: 'A'}});

db.Student.find()

Result

{_id: ObjectId('_ _ _'),
RollNo: 1,
Age: 21,
Cont: 9876,
email: 'abc@gmail.com',
Grade: 'A'
}

6. Update to add hobbies:

→ db.Student.update({RollNo: 1}, {$set: {Hobbies: "cricket"}});

db.Student.find()

Result

{
id: ObjectId(' _ _'),
RollNo: 1,
Age: 21,
Cont: 9876,
email: 'abc@gmail.com',
Grade: 'A',
Hobbies: 'cricket'
}

7. Find documents where hobbies is not matching to chess not to skating.

→ db.Student.find({hobbies: {$nin: ['chess', 'skating']}}).pretty();

Result:-

{
_id: ObjectId('__'),
RollNo: 1,
Age: 21,
cont: 9876,
email: 'abc@gmail.com',
Grade: 'A',
hobbies: 'cricket'
}

8. Find documents whose names begins with A

→ db.student.find({Name: /^A/}).pretty();

Result:

_id: ObjectId('__'),
RollNo: 3,
Age: 11,
cont: 3576,
email: 'abcd@gmail.com',
hobbies: 'baseball',
Name: 'Aritha'
}

either to

skating"]}})

9. Customer Collection

5. Sort the document

-> db.Customer.find().sort({"cust_id":1, "balance":-1});

6. db.Customer.find().skip(1).limit(2);

2/4/24

Week - 03

(11) Create a collection by the name blogposts and it has 3 fields id, title and comments. In the collection the comments field is an array which consists of user details. Each collection consists of two user details inside the comments array - user name and text.

~~1. Adding an element into a~~

1. Adding an element into array

```
-> db.CreateCollection("blog posts");
db.blogPosts.insertOne({
  _id: 1,
  title: "MongoDB",
  comments: [
    {username: "User1", text: "Great article!"},
    {username: "User2", text: "Great!"}]})
db.blogPosts.updateOne(
  {_id: 1},
  {$push: {comments: {username: "user3",
    text: "Nice"}}})
```

2. Display second element

```
->db.blogPosts.find({_id: 1}, {"comments.1": 1, _id: 0})
```

3. Display size of the array

-> db.blogposts.aggregate([

{ $match: {_id: 1}},

{ $project: {commentsCount: {$size: "$comments"}}}

])

4. Display 2 elements of the array.

-> db.blogposts.find({_id: 1}, {'comments': $slice: 2},

_id: 0})

5. Update the document with id 4 & replace the element present in 1st index position of the array. with another array.

-> db.blogposts.updateOne(

{_id: 4},

{$set: {"comments.1": {username: "u4", text: "updated"}}})

pl

16/4/24 Week - 4

DB operations using Cassandra

1) Create a keyspace by name Employee

→
Create Keyspace Employee with Replication = {
'class': 'SimpleStrategy', 'replication_factor': 1};

2) Create a column family by name Employee-info

→
Create table Employee-info(Emp_id int primarykey,
Emp-name text, Designation text, Date of joining timestamp,
Salary int, Dept-name text);

3) Insert values into table in batch
Begin batch
→
Insert into Employee-info (Emp-id, Emp-name, Designation,
Dateofjoining, Salary, Dept-name) values (111, 'A', Analyst,
'2024-04-06', 10000, 'Data Science')
Apply Batch;

4) Update emp-name & Department of empid 111

→
Update Employee_info set emp-name = 'A', '
Dept-name = 'Analyst' where Emp-id = 111;

Week-5

5] Sort the details of employee records based on salary

→ paging off;
select * from employee_info where emp_id in (121, 122, 123, 124) order by salary DESC

6] Alter the schema of the table employee info to add a column projects which stores a set of project done by the corresponding employee.

→ alter table employee_info add projects set<text>;
update employee_info set projects = projects + {'emailspamdetection', 'ABC', 'IoT'} where emp_id = 121 & salary = 100000;

7] Create a TTL of 15 seconds to display the values of employees

→ insert into employee_info (emp_id, emp_name, designation, date_of_joining, salary, department) values (125, 'Rito', 'portwipk', '2010-06-05', 195900, 'ECE') using TTL 15;

2] Perform the following DB

1] Create keyspace by name name Library

→ create keyspace library with replication = {
'class' : 'SimpleStrategy', 'replication_factor' : 1};
use library;

2] Create column family name library-info with attributes stud-id, counter-value, stud-name, book-name, book-id, Date of issue.

→ create table 'library-info(stud-id, int, counter-value counter, stud-name text, book-name text, book-id text, Date of issue timestamp, primary key (stud-id, stud-name, book-name, book-id, Date of issue));

3] Insert values into the table

→ update library-info set counter-value = counter-value + 1, where stud-id = 112 and stud-name = 'Riya' and book-name = 'BDA' and 'Book-id' = 'BDA202' and dateofissue = '2024-06-09';

4] Display the details of the table and increment the counter

→ Select * from library-info;
update library-info set counter = counter + 1 where stud-id = 111 & stud-name = 'Riya' and book-name = 'BDA' and dateofissue = '2024-06-09';

select * from library_info;

5] Write a query to show that student with id 112 has taken books 2 times.

→ select * from library_info where stud_id = 112;

6] Export created csv file

→ copy library_info (stud_id, book_id, stud_name, book_name, dateofissue, count(*_value) to '/home/bmsce/library_info.csv'

7] Import the csv file

→ copy library_info with columns (stud_id, book_id, stud_name, book_name, dateofissue, count(* value) from '/home/bmsce/library_info.csv'