**Day 13: 01-11-2025:**

Projection or retrieve specific document fields from collection

db.CollectionName.find({condition},{projection});

retrieve name and \_id fields

db.Employees.find({},{name:1});

retrieve name, \_id,salary fields

db.Employees.find({},{name:1,salary:1});

retrieve name and salary

db.Employees.find({},{name:1,name:1,\_id:0});

**update query**

updateOne() : condition with \_id then use updateOne()

updateMany() : condition with apart from \_id use updateMany();

db.Employees.updateOne({\_id:1},{$set:{age:25}});

db.Employees.updateMany({city:"Bangalore"},{$set:{city:"Bengaluru"}});

**delete query**

db.Employees.deleteOne({\_id:6});

db.Employees.deleteMany({city:”Bengaluru”});

Student

SID(PK), SName, Age,SkillSet :

1, Raj,21, C/C++/Java

**Store array value in collection**

In mongo DB value can be array with primitive types.

db.Students.insertMany([

{\_id:1,sname:"Reeta",age:21,skillSet:["Java","Python"]},

{\_id:2,sname:"Veeta",age:22,skillSet:["HTML","CSS","JS","ReactJS"]},

{\_id:3,sname:"Keeta",age:23,skillSet:["AI"]}

]);

Want to add new value in array types.

$push

$pop

$in

Adding new value of array type

db.Students.updateOne({\_id:1},{$push:{skillSet:"Spring boot"}});

remove from array elements

db.Students.updateOne({\_id:2},{$pop:{skillSet:1}});

db.Students.find({skillSet:"HTML"},{sname:1,\_id:0});

remove specific value from array

db.Students.updateOne({\_id:1},{$pull:{skillSet:{$in:["Python"]}}});

**Mongo DB relationship**

In mongo db we can achieve relationship using 2 ways

1. Embedded style : store all information in one collection
2. Linking style : store information more than one collection

Embedded style : value can be object type as well as array of object types.

In mongo db we can achieve relationship on document level rather than collection level.

db.EmployeeInfo.insertMany(

[

{\_id:100,name:"Ravi",age:21,salary:45000,address:{city:"Bangalore",state:"Kar"}},

{\_id:101,name:"Raju",age:24,salary:46000,address:[{city:"Bangalore",state:"Kar"},{city:"Mumbai",state:"Mh"}]},

{\_id:102,name:"Ram",age:25,salary:48000,address:[{city:"Bangalore",state:"Kar"},{city:"Mumbai",state:"Mh"}],

projects:[{pid:1111,tech:"Java"}]},

{\_id:103,name:"Rajesh",age:27,salary:52000,address:[{city:"Bangalore",state:"Kar"}],

projects:[{pid:1111,tech:"Java"},{pid:2222,tech:"Python"}]}

]

);

Appy condition with complex type values

db.EmployeeInfo.find({"address.city":"Mumbai"});

db.EmployeeInfo.find({"address.city":"Mumbai"},{name:1});

**Trainer and Student relationship using linking style**

Trainer and Student relationship using linking style

Student collection hold trainer details

db.Trainer1.insertMany(

[

{\_id:1,tname:"Raj",tech:"Java"},

{\_id:2,tname:"Ravi",tech:"Python"}

]);

db.Student1.insertMany([

{\_id:100,sname:"Reeta",age:21,trainer:db.Trainer1.findOne({\_id:1})},

{\_id:101,sname:"Veeta",age:22,trainer:db.Trainer1.findOne({\_id:1})},

{\_id:102,sname:"Keeta",age:23,trainer:db.Trainer1.findOne({\_id:2})},

{\_id:103,sname:"Meeta",age:24,trainer:[db.Trainer1.findOne({\_id:1}),db.Trainer1.findOne({\_id:2})]},

]);

Trainer collection hold student details

db.Student2.insertMany([

{\_id:100,sname:"Reeta",age:21},

{\_id:101,sname:"Veeta",age:22},

{\_id:102,sname:"Keeta",age:23},

{\_id:103,sname:"Meeta",age:24}

]);

db.Trainer2.insertMany(

[

{\_id:1,tname:"Raj",tech:"Java",students:[db.Student2.findOne({\_id:100}),db.Student2.findOne({\_id:101}),db.Student2.findOne({\_id:103})]},

{\_id:2,tname:"Ravi",tech:"Python",students:[db.Student2.findOne({\_id:102}),db.Student2.findOne({\_id:103})]}

]);

Student collection hold trainer id only

db.Trainer3.insertMany(

[

{\_id:1,tname:"Raj",tech:"Java"},

{\_id:2,tname:"Ravi",tech:"Python"}

]);

db.Student3.insertMany([

{\_id:100,sname:"Reeta",age:21,trainer:db.Trainer3.findOne({\_id:1}).\_id},

{\_id:101,sname:"Veeta",age:22,trainer:db.Trainer3.findOne({\_id:1}).\_id},

{\_id:102,sname:"Keeta",age:23,trainer:db.Trainer3.findOne({\_id:2}).\_id},

{\_id:103,sname:"Meeta",age:24,trainer:[db.Trainer3.findOne({\_id:1}).\_id,db.Trainer3.findOne({\_id:2}).\_id]}

]);

Trainer collection hold student id only

**Mongo Db Aggregate function or methods**

It is use to perform advanced data processing and analysis on document within a collection

Using this we can transform, filter, group and computer data like similar to SQL group by, join, and where clause but with much more flexibility.

Aggregate works on aggregate pipeline, which is sequence of stages.

Each stage transform the document passed the result to next stage like lambda with intermediate operator or methods.

Syntax

Db.collectionName.aggregate([

{stage1},

{stage2},

{stage3},

{stagen},

])

$count

$match

$group

$project

$sort

$lookup

Etc

find all employee count

db.Employee.aggregate([

{$count:"totalEmployees"}

]);

find all employees count with match as city

db.Employee.aggregate([

{$match:{city:"Bengaluru"}},

{$count:"totalEmployees"}

]);

find all employees count with match as deptId

db.Employee.aggregate([

{$match:{deptId:100}},

{$count:"totalEmployees"}

]);

$group operator which help to make the group ie deptId, city

db.Employee.aggregate([

{$group:{\_id:"$city"}}

]);

db.Employee.aggregate([

{$group:{\_id:"$deptId"}}

]);

$group operator which help to make the group we can do maths operation like sum, max,min,avg, count

db.Employee.aggregate([

{$group:{\_id:"$city",totalSalary:{$sum:"$salary"}}}

]);

db.Employee.aggregate([

{$group:{\_id:"$city",maxSalary:{$max:"$salary"}}}

]);

db.Employee.aggregate([

{$group:{\_id:"$city",minSalary:{$min:"$salary"}}}

]);

db.Employee.aggregate([

{$group:{\_id:"$city",avgSalary:{$avg:"$salary"}}}

]);

db.Employee.aggregate([

{$group:{\_id:"$city",numberOfEmp:{$sum:1}}}

]);

db.Employee.aggregate([

{$match:{city:"Bengaluru"}},

{$group:{\_id:"$city",numberOfEmp:{$sum:1}}}

]);

db.Employee.insertMany(

[

{ \_id: 1, name: 'Ravi', age: 25, salary: 45000, city: 'Bengaluru',deptId:100},

{ \_id: 2, name: 'Ramesh', age: 24, salary: 42000, city: 'Mumbai',deptId:101 },

{ \_id: 3, name: 'Rajesh', age: 28, salary: 49000, city: 'Bengaluru' ,deptId:102},

{ \_id: 4, name: 'Lokesh', age: 29, salary: 41000, city: 'Pune' ,deptId:100},

{ \_id: 5, name: 'Mahesh', age: 30, salary: 46000, city: 'Bengaluru',deptId:101},

{ \_id: 6, name: 'Reeta', age: 25, salary: 45000, city: 'Bengaluru',deptId:102},

{ \_id: 7, name: 'Meeta', age: 24, salary: 42000, city: 'Mumbai',deptId:101 },

{ \_id: 8, name: 'keeta', age: 28, salary: 49000, city: 'Bengaluru' ,deptId:102},

{ \_id: 9, name: 'Veeta', age: 29, salary: 41000, city: 'Pune' ,deptId:102},

{ \_id: 10, name: 'Mahesh', age: 30, salary: 46000, city: 'Bengaluru',deptId:100 }

]);