basics

Sunday, July 20, 2025 3:06 AM

MONGODB : Flow Chart

DATABASES ->COLLECTION ->DOCUMENTS->JSON/OBJECTS(FIELD VALUE PAIRS)

Databases Types

- Structured data is organized in rows/columns like in SQL (Structured Query Language)
  -type of data where documents or fields are predefined or fixed
  -ex are MySQL ,

Name	roll no
Tulsi	23
Mansi	34

- Unstructured data is not organized and flexible like in NoSQL
  -type of data where fields are not predefined and all documents are independent of each other
  -like in student collection tom can have fields like pincode ,hobby whereas jerry may not have this field
  -ex are MongoDB ,

  ```
  Student={

  Name : tulsi
  Rollno:56
  Hobby:dance
  }
  ```

Collection holds multiple related documents .

Document is a data which is stored in JSON-like format.

CRUD

Create:
db.collection.insertOne({ name: "tulsi })
Read:
db.collection.find({ name: "bob" })
Update:
db.collection.updateOne({ name: "tom" }, { $set: { age: 25 } })
Delete:
db.collection.deleteOne({ name: "yoy" })
Find:
db.collection.find({filter} ,{project}) project is when we want a specific key to access

Operators

- $set – updates or adds a field: { $set: { age: 20 } }
- $project – gives a specific field : { $project: { name: 1 } }
- $unwind – splits array fields into separate documents. It divides document into multiple objects so it becomes easy to access using dotoperator

  For ex

  a document name **orders** ({productId: " " },{ProductId:" " },{ProductId:" " },{ProductId:" " })
  and I want to **access** each **productId** from this document sooooo ....I directly cannot do it when I use **unwind**
  { ProductId:} ,{ProductId:} ......like this it becomes separate objects and now I can access them as **$orders.ProductId**

- $group – groups data or groups the related documents which are given through **_id(the mandatory field tells by what it should group by)** and performs operations like $sum, $avg etc
- $lookup – joins the two collections (localfield where both collections contains same key -values)

```
db.orders.aggregate([
 {
  $lookup: {
   from: "customers" ,localField: "customerId",
   foreignField: "_id",
   as: "customer"
  }
 },
 { $unwind: "$customer" },
 {
  $lookup: {
   from: "cities",
   localField: "customer.cityId",
   foreignField: "_id",
   as: "customer.city"
  }
```

```
 },
 { $unwind: "$customer.city" },
 {
   $project: {
     productId: 1,
     "customer.name": 1,
     "customer.city.name": 1,
     "customer.city.country": 1
   }
 }
]);
```

- $gt : greater than , $gte : greater than or equal to   similarly $lt and $lte
- $ne: not equal to
- $unset: removes the field
- $exists: checks if the field exists or not
- $and: logical And similarly $or ,$not

student → subjects → marks   (this is how 2 level collections are related)
Student is a parent class -> subject is a child of student -> marks is a child of subject
We can relate the data using the unique Object_Id which is generated while inserting a particular object

indexing means it creates a sorted data structure in documents
indexing it is used for searching the documents
it makes the process faster and easier
when we search for any document I searches line by line or one by on,we can aviod it by using indexing

JSON
(JavaScript Object Notation) also known as object in javascript