Mongodb

➢ CLASS 1: INTRO TO MONGODB

MongoDB is an open-source document-oriented database. It is categorized under the NoSQL(Not only SQL) database because the storage and retrieval of data in MongoDB are not in the form of tables.

- Structured Data:

The information is typically organized in a specific format, often using tables with rows and columns. This makes it easier to search, filter, and analyze the data.

- Database Management System (DBMS):

This is the software that acts like the filing cabinet manager. It allows you to store, retrieve, update, and manage all the data within the database.

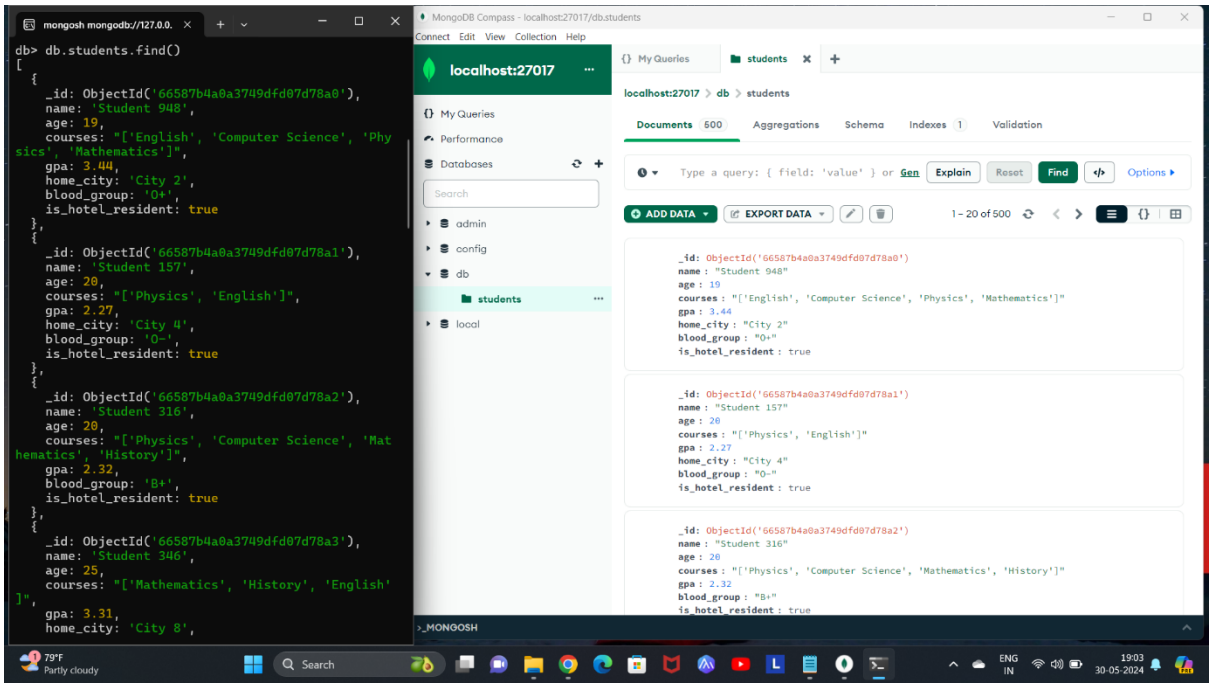
- Data Types:

Databases can hold various kinds of information, including text, numbers, images, videos, and more.

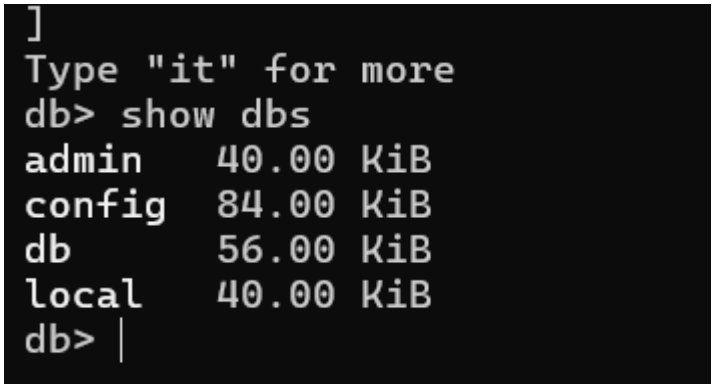
Mongodb

➢ CLASS 2: ADD , UPDATE AND DELETE

To find the data that is present in the collections ,we can use the command “db.collection_name.find()”



In the above example the collection name is “students”



By using “show dbs” command all database are shown

Mongodb

```
db> use db
already on db db
db>
```

The “use db” command is used to connect and use db

- Collections :

A collection is a group of documents.

If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table.

- Database:

MongoDB groups collections into databases.

A single instance of MongoDB can host several databases, each grouping together zero or more collections.

Mongodb

➢ CLASS 3: Where AND ,OR & CRUD

- WHERE:

Given a Collection you want to FILTER a subset based on a condition. That is the place WHERE is used.

```
db> db.students.find({ gpa:{$gt:3}})
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a0'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Phy
sics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a3'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English'
]",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

In this example we use the condition gpa greater than 3 . So the result is shown above is based on this condition

Here '$gt' means greater than

Mongodb

- AND:

Given a Collection you want to FILTER a subset based on multiple conditions ,then you use AND.

```
Type "it" for more
db> db.students.find({
... $and:[
... {home_city:"City 1"},
... {blood_group:"O-"}
... ]
... })
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78c0'),
    name: 'Student 384',
    age: 18,
    courses: "['Mathematics', 'Computer Science']"
,
    gpa: 3.9,
    home_city: 'City 1',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d7950'),
    name: 'Student 702',
    age: 22,
    courses: "['History', 'Mathematics', 'English'
]",
    gpa: 3.74,
    home_city: 'City 1',
    blood_group: 'O-',
    is_hotel_resident: false
  },
```

Above example is filtered based on 2 condition i.e

‘home_city : City1’ and ‘blood_group : O-‘

- OR:

Given a collection you want to FILTER a subset based on multiple conditions but any one is sufficient.

```
db> db.students.find({
... $or:[
... {blood_group:"O+"},
... {gpa:{$lt:3.5}}
... ]
... })
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a0'),
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Phy
sics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a1'),
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

Above example ,the students database is filtered based on either ‘blood_group : O+’ or ‘gpa less than 3.5’

NOTE: $lt is less than.

Mongodb

- CRUD:
  C – Create / Insert
  R – Remove
  U – Update
  D – Delete

This is applicable for a collection (table) or a document (row)

- Insert:

We can insert data to the collections .

```
db> const studentData = {
... "name":"Sam",
... "age":22,
... "courses":["Computer Science" , "Mathematics"]
,
... "gpa":3.4,
... "home_city":"City 3",
... "blood_group":"B+",
... "is_hotel_resident":false
... }

db> db.students.insertOne(studentData)
{
  acknowledged: true,
  insertedId: ObjectId('6658a0c70cce0c5ec1cdcdf6')
}
db>
```

Here we are inserting the student details name ‘Sam’ and other information to the collection ‘students’.the insertion is done one time.

Mongodb

- Update:

We can update any data that is present in the collections.

```
db> db.students.updateOne( { name:"Sam"} , {$set:{
gpa:3} } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
db>
```

To update we use ‘$set’ command .

- Delete:

It is used to delete the data present in the collection.

```
db> db.students.deleteOne({ name:"Sam" })
{ acknowledged: true, deletedCount: 1 }
db>
```

Here the details of ‘Sam’ is deleted.

- Projection:

This is used when we don’t need all columns / attributes.

```
db> db.students.deleteOne({ name:"Sam" })
{ acknowledged: true, deletedCount: 1 }
db> db.students.find({} , {name:1 , gpa:1 })
[
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a0'),
    name: 'Student 948',
    gpa: 3.44
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a1'),
    name: 'Student 157',
    gpa: 2.27
  },
  {
    _id: ObjectId('66587b4a0a3749dfd07d78a2'),
    name: 'Student 316',
    gpa: 2.32
```

Here it only shows the name and gpa . Because the command is give as ‘name:1’ and ‘gpa:1’

- Benefits of Projection:
  - ✓ Reduced data transferred between the database and your application.
  - ✓ Improves query performance by retrieving only necessary data.
  - ✓ Simplifies your code by focusing on the specific information you need.

➢ CLASS 4: Limit and Selectors

- Limit:

The limit operator is used with the find method.

It's chained after the filter criteria or any sorting operations.

Syntax:

db.collection.find({filter}, {projection}).limit(number)

```
Type "it" for more
db> db.students.find({} , {_id:0}).limit(5)
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
```

To get only first 5 document we use limit(5).

- Selectors:
  - ✓ Comparison gt and lt
  - ✓ AND operator
  - ✓ OR operator