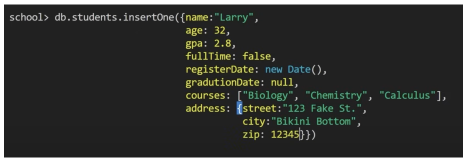
NoSQL

1 Simple command with NoSQL

To use the database, we will need to open the terminal and then open the server of NoSQL with the command: sudo mongod –dbpath=/Users/mymac/data/db

- show dbs: show all databases in the connection of the NoSQL
- use databaseName : use the database given in the NoSQL
- db.dropDatabase(): drop the current database from the connection
- db.createCollection(): create a collection in the database
- We can insert data into the database by using the db.collectName.insertOne(data) to insert the data into the collection. By using the insertOne function we only insert one data at a time to the collection. We can use db.collectName.insertMany(data) to insert multiple data at a time.
- You can see the database by using the db.collectName.find() to show all the data in one collection.
- There are different data types in the MongoDB: strings which are in the "", integer, double, boolean, date, null value, arrays which are in the [], nested datatype which are in the {}

Examples:



```
school> db.students.insertOne({name:"Larry",
                               age: 32,
                               gpa: 2.8,
                               fullTime: false,
                               registerDate: new Date(),
                               gradutionDate: null,
                               courses: ["Biology", "Chemistry", "Calculus"],
                               address: {street:"123 Fake St.",
                                         city:"Bikini Bottom",
                                         zip: 12345}})
```

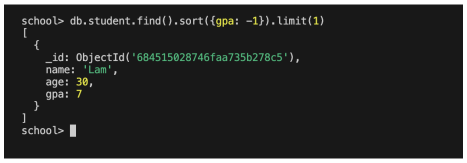
Figure 1: Example 1

+) Note that whenever you create a Date datatype, you should declare it as new Date("datetime") or it will use the current UTC datetime.

2 Sort documents in MongoDB

We can sort the documents in MongoDB by using:
- db.collectName.find().sort(the method of how you want to sort your data)

You can sort your data by alphabetical order of one variable by using variable-Name: 1 to sort by order or -1 to reverse the order
When sorting the number, -1 is from the bigger to smaller and 1 for smaller to bigger.
- db.collectName.find().limit(number): You limit the number of the documents you want to show

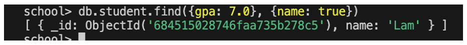


```
school> db.student.find().sort({gpa: -1}).limit(1)
[
  {
    _id: ObjectId('684515028746faa735b278c5'),
    name: 'Lam',
    age: 30,
    gpa: 7
  }
]
school> 
```

Figure 2: Example of include both sort and limit

3 Find documents in MongoDB

- db.collectName.find(method): find the specific data in the collection by using the method to search for the data you need. - db.collectName.find(methodprojection): return the only data that meets both the projection and the method.

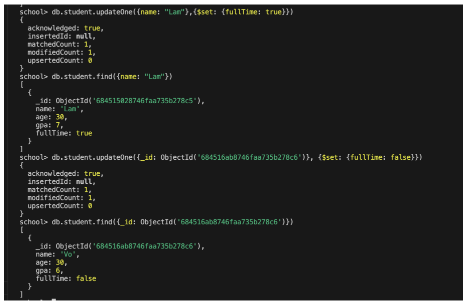


```
school> db.student.find({gpa: 7.0}, {name: true})
[ { _id: ObjectId('684515028746faa735b278c5'), name: 'Lam' } ]
school> 
```

Figure 3: Return the data meets both criteria

4 Updating data in MongoDB

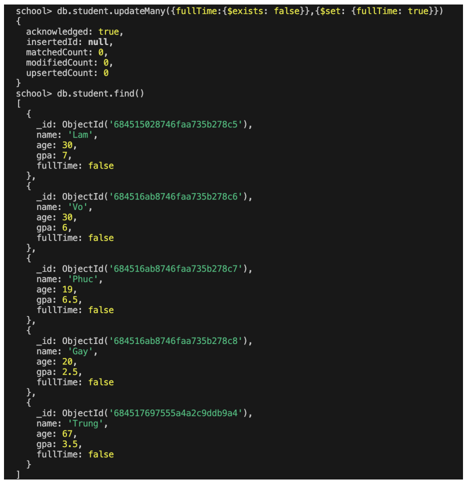
- We can update the data in the MongoDB by using the db.collectName.updateOne(filter, updated data) to update one key value in a collection. It is advisable that we should update the data in our collection by filtering the id. - We can also update



```
school> db.student.updateOne({name: "Lam"},{$set: {fullTime: true}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
school> db.student.find({name: "Lam"})
[
  {
    _id: ObjectId('684515028746faa735b278c5'),
    name: 'Lam',
    age: 30,
    gpa: 7,
    fullTime: true
  }
]
school> db.student.updateOne({_id: ObjectId('684516ab8746faa735b278c6')}, {$set: {fullTime: false}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
school> db.student.find({_id: ObjectId('684516ab8746faa735b278c6')})
[
  {
    _id: ObjectId('684516ab8746faa735b278c6'),
    name: 'Vo',
    age: 30,
    gpa: 6,
    fullTime: false
  }
]
```

Figure 4: Update one column in the document

multiple keys value in the collection by using updateMany().
- While $set is used to set update and insert the key values in the collection, $unset is used to remove a field from the collection. - You can also update by checking whether a field exists or not by using $exists to check the existance of an object.



```
school> db.student.updateMany({fullTime:{$exists: false}},{$set: {fullTime: true}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
school> db.student.find()
[
  {
    _id: ObjectId('684515028746faa735b278c5'),
    name: 'Lam',
    age: 30,
    gpa: 7,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c6'),
    name: 'Vo',
    age: 30,
    gpa: 6,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c7'),
    name: 'Phuc',
    age: 19,
    gpa: 6.5,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c8'),
    name: 'Gay',
    age: 20,
    gpa: 2.5,
    fullTime: false
  },
  {
    _id: ObjectId('684517697555a4a2c9ddb9a4'),
    name: 'Trung',
    age: 67,
    gpa: 3.5,
    fullTime: false
  }
]
```

Figure 5: Check the existance and then update

5 Delete data in MongoDB

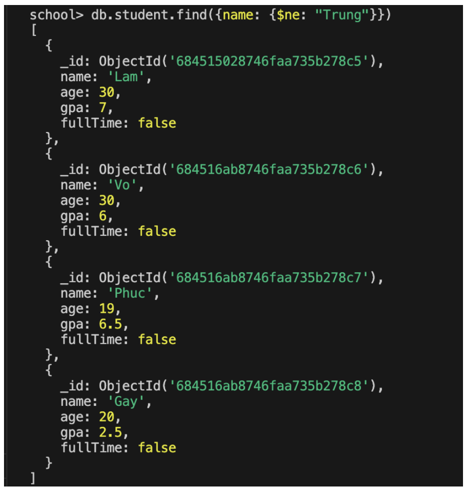
- We can delete one data by one criteria in the collection by using db.collectName.deleteOne(criteria). We can also use the db.collectName.deleteMany(criteria) to delete all the collections that match the criteria.

6 Comparisions in MongoDB

We can show all the data that not equals to sth by using $ne.

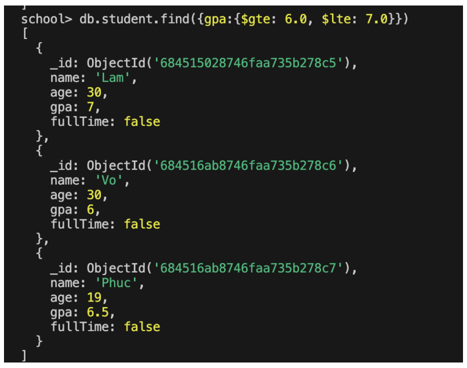
- $ne: not equal
- $lt: less than
- $lte: less than or equal to
- $gt: greater than
- $gte: greater than or equal to

We can also search for data in an array by using $in or not in $nin



```
school> db.student.find({name: {$ne: "Trung"}})
[
  {
    _id: ObjectId('684515028746faa735b278c5'),
    name: 'Lam',
    age: 30,
    gpa: 7,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c6'),
    name: 'Vo',
    age: 30,
    gpa: 6,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c7'),
    name: 'Phuc',
    age: 19,
    gpa: 6.5,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c8'),
    name: 'Gay',
    age: 20,
    gpa: 2.5,
    fullTime: false
  }
]
```

Figure 6: Show all the data that has the name not equal to something



```
school> db.student.find({gpa:{$gte: 6.0, $lte: 7.0}})
[
  {
    _id: ObjectId('684515028746faa735b278c5'),
    name: 'Lam',
    age: 30,
    gpa: 7,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c6'),
    name: 'Vo',
    age: 30,
    gpa: 6,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c7'),
    name: 'Phuc',
    age: 19,
    gpa: 6.5,
    fullTime: false
  }
]
```

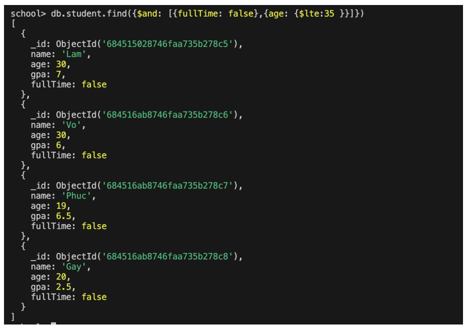
Figure 7: Show all the data that is between values and values

7 Logical operators

We can find the data by using the logical operations: $\$and$ and $\$or$ We can also use $\$nor$, which means that both conditions need to be false. We also have the logical operator: $\$not$ to search for all the data not in our logical comparison

8 Indexes

- We can use db.collectName.find(criteria).explain(”executionStats”) to show the stats in the collection. By using:
db.collectName.createIndex(criteria).
Note the criteria here is not search for the value equals to something. Instead



```
school> db.student.find({$and: [{fullTime: false},{age: {$lte:35 }}]})
[
  {
    _id: ObjectId('684515028746faa735b278c5'),
    name: 'Lam',
    age: 30,
    gpa: 7,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c6'),
    name: 'Vo',
    age: 30,
    gpa: 6,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c7'),
    name: 'Phuc',
    age: 19,
    gpa: 6.5,
    fullTime: false
  },
  {
    _id: ObjectId('684516ab8746faa735b278c8'),
    name: 'Gay',
    age: 20,
    gpa: 2.5,
    fullTime: false
  }
]
```

Figure 8: Using logical operations to find the values

it is the sorting method like 1 for smaller to bigger or -1 for bigger to smaller. We can drop index by using db.student.dropIndex("indexname") to drop the index. Apply the index can allow for quicker searching but it might take more memory and harder for updating or inserting new data.

9 Collections

- Collections is where we store all the objects that have data in common.
- We can see all the collections by show collections.
- We can create new collection by db.createCollection("collectionName",criteria)
- We can drop the collection by typing db.collectName.drop()