TP05: NoSQL Databases – MongoDB

kiram mohammed ali

Introduction

This report presents the solution to TP05: "Bases de données NoSQL – MongoDB". The objective is to install MongoDB, create a database, insert documents, and perform CRUD operations using the Mongo Shell. All screenshots of the execution are stored in the folder `photo/`.

1. Installing MongoDB

MongoDB 7.0 was installed on Pop!OS/Linux using the official GPG key and repository.

Commands Used

```
sudo apt update && sudo apt upgrade -y
curl -fsSL https://pgp.mongodb.com/server-7.0.asc |
  sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg --dearmor

echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/
mongodb-server-7.0.gpg ] https://repo.mongodb.org/apt/ubuntu
jammy/mongodb-org/7.0 multiverse" |
sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list

sudo apt update
sudo apt install -y mongodb-org
sudo systemctl start mongod
mongosh
```

Overview

MongoDB Community Server is now running, and the Mongo Shell (`mongosh`) is ready for database operations.

2. Creating the Database

Execution

```
use info
```

Overview

The command creates (lazily) and switches to a database named `info`.

3. Creating the Collection and Inserting Documents

First Document: Macbook Pro

```
db.produits.insertOne({
    nom: "Macbook Pro",
    fabriquant: "Apple",
    prix: 11435.99,
    options: [
        "Intel Core i5",
        "Retina Display",
        "Long life battery"
    ]
})
```

Second Document: Macbook Air

```
db.produits.insertOne({
    nom: "Macbook Air",
    fabriquant: "Apple",
    prix: 125794.73,
```

```
    ultrabook: true,
    options: [
        "Intel Core i7",
        "SSD",
        "Long life battery"
    ]
})
```

Third Document: Thinkpad X230

```
db.produits.insertOne({
    nom: "Thinkpad X230",
    fabriquant: "Lenovo",
    prix: 114358.74,
    ultrabook: true,
    options: [
        "Intel Core i5",
        "SSD",
        "Long life battery"
    ]
})
```

Screenshot

Figure 1: Inserting the three documents into the `produits` collection.

4. CRUD Queries

4.1 Retrieve all products

```
db.produits.find()
```

4.2 Retrieve the first product

```
db.produits.findOne()
```

4.3 Retrieve Thinkpad X230 by name

```
db.produits.findOne({ nom: "Thinkpad X230" })
```

4.4 Retrieve Thinkpad X230 by ObjectId

```
db.produits.findOne({ _id: ObjectId("692431225a0e6418199dc29f") })
```

4.5 Products with price > 13723

```
db.produits.find({ prix: { $gt: 13723 } })
```

4.6 First ultrabook

```
db.produits.findOne({ ultrabook: true })
```

4.7 First product whose name contains “Macbook”

```
db.produits.findOne({ nom: /Macbook/ })
```

4.8 Products whose name starts with “Macbook”

```
db.produits.find({ nom: /^Macbook/ })
```

Screenshot

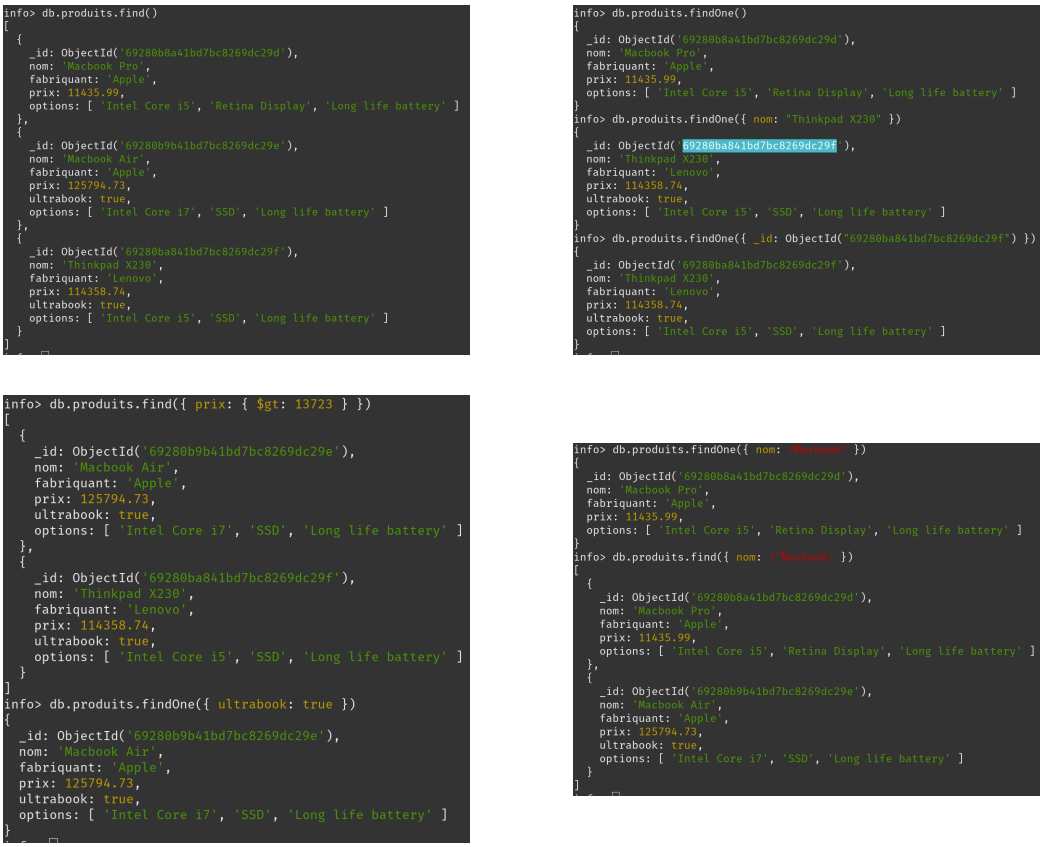

Figure 2: Four images (2x2 grid)

5. Delete Operations

5.1 Delete all Apple products

```
db.produits.deleteMany({ fabriquant: "Apple" })
```

5.2 Delete Thinkpad X230 by ObjectId

```
db.produits.deleteOne({ _id: ObjectId("692431225a0e6418199dc29f") })
```

Screenshot

```
info> db.produits.deleteMany({ fabriquant: "Apple" })
{ acknowledged: true, deletedCount: 2 }
info> db.produits.find()
[
  {
    _id: ObjectId('69280ba841bd7bc8269dc29f'),
    nom: 'Thinkpad X230',
    fabriquant: 'Lenovo',
    prix: 114358.74,
    ultrabook: true,
    options: [ 'Intel Core i5', 'SSD', 'Long life battery' ]
  }
]
info> db.produits.deleteOne({ _id: ObjectId("69280ba841bd7bc8269dc29f") })
{ acknowledged: true, deletedCount: 1 }
info> db.produits.find()

info>
```

Figure 3: Deletion of Apple products and the Lenovo device.

Overview

After deletion, the `produits` collection becomes empty, confirming that all delete operations were executed successfully.

Conclusion

This TP demonstrates the essential CRUD operations in MongoDB using the Mongo Shell. We successfully created a database, inserted multiple documents, executed advanced queries, and performed deletions by filtering and by ObjectId. MongoDB's flexible document model makes it highly suitable for semi-structured data and rapid prototyping.