

**University of El Oued**

Faculty of Exact Sciences

Department of Computer Science

# **Lab Report 5: NoSQL Databases**

MongoDB Setup & CRUD Operations

**Student:**

Mohammed Seddik Lifa

**Specialization:**

Master II: AI & Data Science

**Date:**

11-10-2025

# 1 Objective

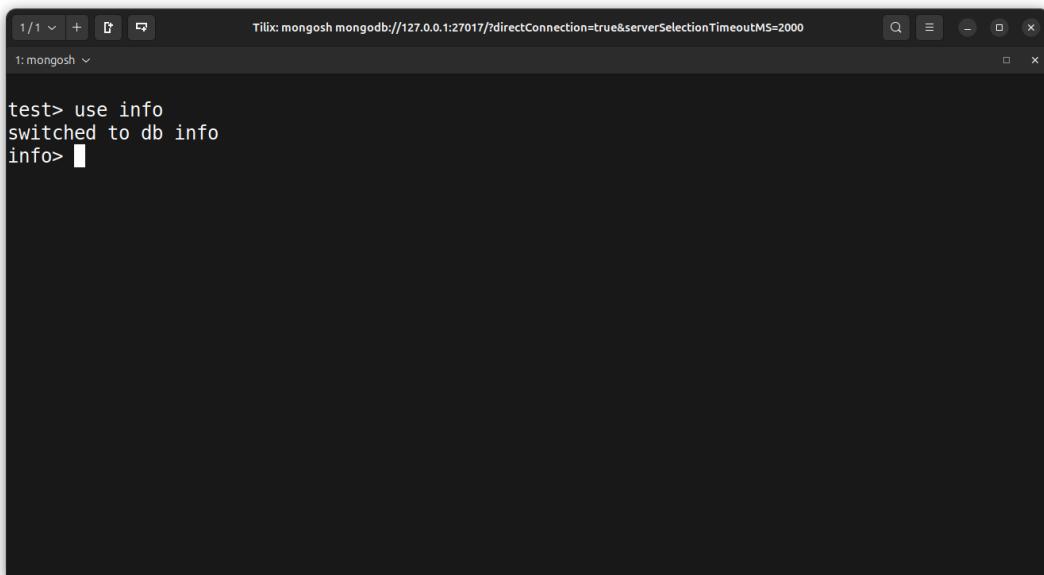
The primary objective of this lab is to install MongoDB, configure the background service, and perform fundamental Create, Read, Update, and Delete (CRUD) operations using the modern mongosh client. This report demonstrates the successful execution of these tasks.

## 2 Implementation & Proof of Execution

### 2.1 A. Create a Database

**Instruction:** Create and switch to a new database named `info` using the command `use info`.

**Execution Proof:** The shell successfully switched context to the new database.

A screenshot of a terminal window titled "Tilix: mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000". The window shows the command "test> use info" being run, followed by "switched to db info" and the prompt "info>".

```
test> use info
switched to db info
info>
```

Figure 1: Command execution: Switching to database 'info'

### 2.2 B. Create a Collection (Insert Documents)

**Instruction:** Insert a dataset into the `produits` collection. The dataset includes details (manufacturer, price, and options) for a Macbook Pro, Macbook Air, and Thinkpad X230.

**Code Pattern Executed:**

```
1 db.produits.insertMany([
2   { nom: "Macbook\u201cPro", fabriquant: "Apple", prix: 11435.99 ... },
3   { nom: "Macbook\u201cAir", fabriquant: "Apple", ultrabook: true ... },
4   { nom: "Thinkpad\u201cX230", fabriquant: "Lenovo", ultrabook: true ... }
```

5 || ] )

**Listing 1:** Insertion Query Structure

**Execution Proof:** The output shows acknowledged: true and three generated ObjectIDs.

Figure 2: Successful insertion of 3 documents with ObjectIDs

## 2.3 C. Read (Query) Documents

**Instruction:** Perform various queries to retrieve data. Specifically:

1. Retrieve a single document.
2. Filter documents where ultrabook is true.
3. Use Regex to find products starting with "Macbook".

**Queries Run:**

```
1 db.produits.findOne()
2 db.produits.findOne({ ultrabook: true })
3 db.produits.find({ nom: { $regex: "Macbook" } })
```

**Execution Proof:**

The screenshot shows the mongo shell interface with the command line 'mongosh' at the top. Below it, the mongo shell prompt 'info>' is followed by three commands. The first command 'db.produits.findOne()' returns a single document for a Macbook Air. The second command 'db.produits.findOne({ ultrabook: true })' also returns the same Macbook Air document. The third command 'db.produits.find({ nom: { \$regex: "Macbook" } })' returns two documents: one for Macbook Pro and one for Macbook Air.

```
info> db.produits.findOne()
{
  _id: ObjectId('691248078c0c70bb99ce5f47'),
  nom: 'Macbook Air',
  fabriquant: 'Apple',
  prix: 125794.73,
  ultrabook: true,
  options: [ 'Intel Core i7', 'SSD', 'Long life battery' ]
}
info> db.produits.findOne({ ultrabook: true })
{
  _id: ObjectId('691248078c0c70bb99ce5f48'),
  nom: 'Macbook Air',
  fabriquant: 'Apple',
  prix: 125794.73,
  ultrabook: true,
  options: [ 'Intel Core i7', 'SSD', 'Long life battery' ]
}
info> db.produits.find({ nom: { $regex: "Macbook" } })
[
  {
    _id: ObjectId('691248078c0c70bb99ce5f47'),
    nom: 'Macbook Pro',
    fabriquant: 'Apple',
    prix: 11435.99,
    options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
  },
  {
    _id: ObjectId('691248078c0c70bb99ce5f48'),
    nom: 'Macbook Air',
    fabriquant: 'Apple',
    prix: 125794.73,
    ultrabook: true,
    options: [ 'Intel Core i7', 'SSD', 'Long life battery' ]
  }
]
```

Figure 3: Query Results: findOne, Boolean filtering, and Regex matching

## 2.4 D. Delete Documents

**Instruction:** Remove specific data from the collection:

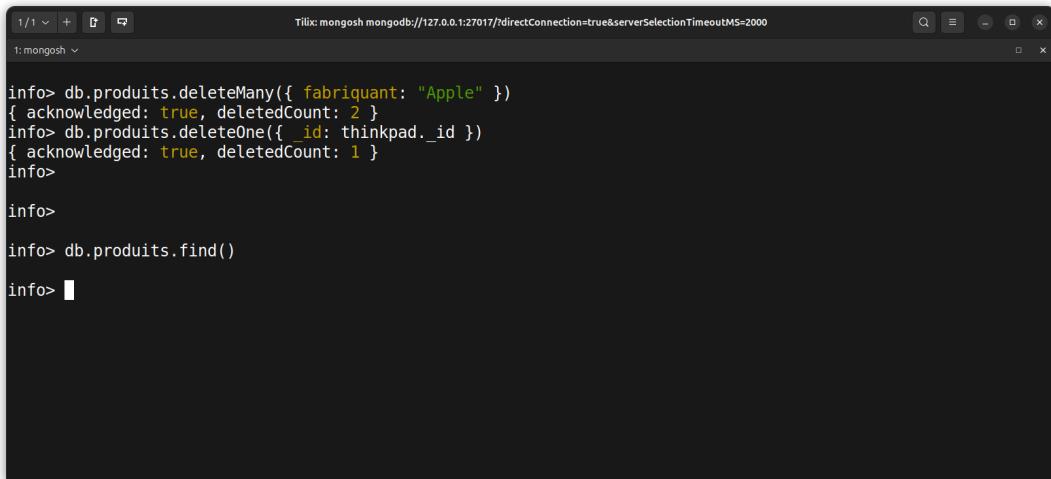
- Delete all documents where manufacturer is "Apple".
- Delete a specific document by its unique `_id`.

**Queries Run:**

```
1 // Delete all Apple products
2 db.produits.deleteMany({ fabriquant: "Apple" })
3
```

```
4 // Delete specific ID (Thinkpad)
5 db.produits.deleteOne({ _id: thinkpad_id })
6
7 // Verify Empty Collection
8 db.produits.find()
```

**Execution Proof:** The output confirms deletedCount: 2 for Apple products and deletedCount: 1 for the Thinkpad. The final find command returns nothing, confirming the collection is empty.



The screenshot shows a terminal window titled 'Tilix: mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000'. The session starts with '1: mongosh <'. It then executes several commands:  
- 'info> db.produits.deleteMany({ fabriquant: "Apple" })': Returns '{ acknowledged: true, deletedCount: 2 }'  
- 'info> db.produits.deleteOne({ \_id: thinkpad.\_id })': Returns '{ acknowledged: true, deletedCount: 1 }'  
- 'info>'  
- 'info>'  
- 'info> db.produits.find()': Returns 'info> [ ]'  
The terminal window has a dark background with light-colored text. The command input is in white, and the output is in yellow.

Figure 4: Deletion operations and final verification