Ministry of Higher Education and Scientific Research
University of Abdelhafid Boussouf - Mila
Institute of Mathematics and Computer Science
Department of Computer Science
Master 2 I2A – Big Data
2025/2026

Directed Works TD 7 – NoSQL Data Modeling & MongoDB

Objective: Understand Document-Oriented concepts, design schemas using Embedding vs. Referencing, and master the MongoDB Query Language (MQL).

Exercise 1: Deciphering MQL (Code Interpretation)

Before writing your own queries, analyze the following MongoDB commands. Translate each command into a plain English sentence describing exactly what it does.

Command A:

```
db.users.insertOne({
   "username": "amine_23",
   "role": "student",
   "is_active": true
})
```

Command B:

```
db.products.find(
   { "category": "Laptops", "price": { "$lt": 50000 } },
   { "name": 1, "price": 1, "_id": 0 }
)
```

Hint: Pay attention to the second set of braces {...}. What does 0 and 1 mean here?

Command C:

```
db.products.updateMany(
   { "category": "Smartphones" },
   { "$inc": { "stock": 5 } }
)
```

Hint: Distinguish between $set (replace value) and $inc (mathematical addition).

Command D:

```
db.logs.deleteMany({})
```

Question: Why is this command considered dangerous?

Exercise 2: Thinking in JSON (Syntax & Structure)

The following BSON/JSON documents contain syntax errors or structural issues. Identify and correct them.

Document A (Product):

```
{
   name: "Gaming Mouse",
   price: 50,00,  // Issue here
   tags: ["gaming", "accessory",], // Issue here
   "is_available": True  // Issue here
}
```

Document B (User Profile):

```
{
   "_id": "user_123",
   "email": "test@gmail.com",
```

```
  "address": [
     "street": "12 Rue de la Liberté",
     "city": "Algiers"
  ] // Issue with data structure type
}
```

Exercise 3: Schema Design (The TechStore Scenario)

We are designing the database for **TechStore**, an electronics e-commerce site. We have three entities: **User**, **Product**, and **Order**.

Context:

- **Access Pattern A:** When we load a User's profile, we *always* display their shipping address immediately.
- **Access Pattern B:** A User can have thousands of Orders over their lifetime. We rarely show the full order history; usually, we only show the "Last 5 Orders."
- **Access Pattern C:** A Product has a list of technical specifications (RAM, CPU, Screen). This list is small (max 20 items) and rarely changes.

Questions:

1. **User ↔ Address:** Should we Embed or Reference the Address? Justify your choice based on Access Pattern A.
2. **User ↔ Orders:** Should we Embed the Orders inside the User document or Reference them? Justify based on Access Pattern B and the MongoDB document size limit (16MB).
3. **Product ↔ Specs:** Propose a JSON structure for the Product entity that handles Access Pattern C efficiently.

Exercise 4: Writing Queries (MQL)

Assume we have a collection "**products**" with the following structure:

```
{
  "_id": 1,
  "name": "Laptop Pro",
  "category": "Computers",
  "price": 1200,
  "stock": 50,
  "tags": ["office", "work"],
  "specs": { "ram": "16GB", "ssd": "512GB" }
}
```

Write the MongoDB shell command for the following requirements:

1. **Select:** Find all products in the "Computers" category that cost less than 1500.
2. **Logical:** Find products that have the tag "gaming" **OR** have a price greater than 3000.
3. **Embedded:** Find all products where the `specs` have exactly "16GB" of RAM.
4. **Update:** The "Laptop Pro" (id: 1) has been restocked. Increase its `stock` by 10 units (Atomic operation).
5. **Array Update:** Add a new tag "bestseller" to the "Laptop Pro", but only if it doesn't already exist.

Exercise 5: Modeling Relationships (Many-to-Many)

We have a Suppliers collection. A Product can be sold by multiple Suppliers, and a Supplier sells multiple Products.

1. Propose a schema that allows us to find **"All products sold by Supplier X"** without scanning the entire Products collection.
2. Explain the concept of "Denormalization" in this context.