



Backend Workshop - Teamwork 3

Workshop

Subject: NOSQL & Expressjs

Learning Goals

- Migrating from RDBS to NOSQL databases.
- Ability to perform CRUD operations in NOSQL.
- Increase our general knowledge about backend and Expressjs

Introduction

Express.js and MongoDB are technologies frequently used together to develop web applications. Express.js handles server-side operations while MongoDB database is used to store and manage data. By combining these two technologies, you can create dynamic and data-driven web applications.

The cooperation between the two mainly occurs in the following ways.

- Database Access: Establishes a connection between Express.js and MongoDB for web applications to perform operations such as inserting, updating, querying, and deleting data into MongoDB.

- API Development: Express.js can be used to create APIs to serve MongoDB data. For example, a RESTful API...
- Database Operations: You can use MongoDB for Express.js to make it easier to manage users in your web application to create their accounts, log in, and update their information.

This study aims to reinforce the preliminary knowledge required to perform the above tasks.

Pre-requirements

We will use the VSCode and at the same time, we need to install MongoDB and Mongosh on our computer.

Lets start

1. How many types of SQL languages are there, which one is for executing which type of commands?
2. What does NOSQL stand for?
3. Why Express.js use mongoDB, what canbe reasons?
4. How can we represent the data in the following tables in MongoDb?

Table name "post"

post_id	title
1	amazing duo Express.js and MongoDB
2	MVC architecture

Table name "comment"

comment_id	post_id	author	content
1	1	Qadir	I agree with you
2	1	Victor	Their communities are very good
3	2	Rafe	Django use MVT
4	1	Cedric	Good post

5. Sometimes analogies are used to understand the subject. Let's do that too. Can you match the MVC structure with the story below?

Which one is the controller, which one is the model, which one is the view?

- Assume that you visit a restaurant. You won't cook food in the kitchen, even though you can do so at home. Instead, you visit the establishment and wait for the waiter to arrive.
- The waiter now approaches you, and you place your dinner order. The server has only written down the specifics of your food order; he is unaware of who you are or what you desire.
- The waiter then makes his way to the kitchen. The server does not prepare your dish in the kitchen.
- Your food is prepared by the cook. Your order and your table number are given to the waiter.
- Cook, then have food prepared for you. The dish is prepared by him using ingredients. Let's say you select the vegetable sandwich option. He then uses the refrigerator to get the bread, tomato, potato, capsicum, onion, bit, and cheese that he needs.
- The chef gives the waiter the last of the meal. Moving this meal outside the kitchen is now the waiter's responsibility.
- The server is now aware of the foods you've ordered and how to serve them.

Teamwork

Subject:

Learning Goals

- Being able to perform operation on MongoDB.

Introduction

We performed operations on relational databases with SQL. Now let's do a study to perform similar operations on MongoDB, a NOSQL database.

Lets start

Write mongoDB statements corresponding to the SQL statements given below.

1.

```
SELECT *  
FROM people
```

2.

```
SELECT id,  
       user_id,  
       status  
FROM people
```

3.

```
SELECT user_id, status  
FROM people
```

4.

```
SELECT *  
FROM people  
WHERE status = "A"
```

5.

```
SELECT user_id, status  
FROM people  
WHERE status = "A"
```

6.

```
SELECT *  
FROM people  
WHERE status != "A"
```

7.

```
SELECT *  
FROM people  
WHERE status = "A"  
AND age = 50
```

8.

```
SELECT *  
FROM people  
WHERE status = "A"  
OR age = 50
```

9.

```
SELECT *  
FROM people  
WHERE age > 25
```

10.

```
SELECT *  
FROM people  
WHERE age < 25
```

11.

```
SELECT *  
FROM people  
WHERE age > 25  
AND age <= 50
```

12.

```
SELECT *  
FROM people  
WHERE user_id like "%bc%"
```

13.

```
SELECT *  
FROM people  
WHERE status = "A"  
ORDER BY user_id ASC
```

14.

```
SELECT *  
FROM people  
WHERE status = "A"  
ORDER BY user_id DESC
```

15.

```
SELECT COUNT(*)  
FROM people
```

16.

```
SELECT COUNT(user_id)  
FROM people
```

17.

```
SELECT COUNT(*)  
FROM people  
WHERE age > 30
```

18.

```
SELECT DISTINCT(status)  
FROM people
```

19.

```
SELECT *  
FROM people  
LIMIT 1
```

20.

```
SELECT *  
FROM people  
LIMIT 5  
SKIP 10
```


21.

```
CREATE TABLE people (  
    id MEDIUMINT NOT NULL  
        AUTO_INCREMENT,  
    user_id Varchar(30),  
    age Number,  
    status char(1),  
    PRIMARY KEY (id)  
)
```

22.

```
ALTER TABLE people  
ADD join_date DATETIME
```

23.

```
DROP TABLE people
```

24.

```
INSERT INTO people(user_id,age,status)  
VALUES ("bcd001",45,"A")
```

25.

```
UPDATE people  
SET status = "C"  
WHERE age > 25
```

26.

```
UPDATE people  
SET age = age + 3  
WHERE status = "A"
```

27.

```
DELETE FROM people  
WHERE status = "D"
```

28.

```
DELETE FROM people
```

😊 Thanks for Attending 🙌

