MongoDB

שיעור ראשון מבוא

מה זה MongoDB
?

- MongoDB הוא מסד נתונים מונחה מסמכים בקוד פתוח, העובד בכל הפלטפורמות, מספק ביצועים גבוהים,
- זמינות גבוהה ויכולת הרחבה קלה.
- MongoDB עובד על הרעיון של אוספים ומסמכים במקום טבלאות, שורות ועמודות כמו RDB(Relational Databases) מסורתיים כגון:

MySQL, SQL Server .

כמה הגדרות

- **מאגר מידע** - מסד הנתונים הוא מעין מיכל לאוספים (collections) וכל מסד נתונים מקבל סט קבצים משלו במערכת הקבצים. שרת MongoDB אחד יכול, ובדרך כלל גם מכיל מסדי נתונים מרובים.
- **אוסף (Collection)**- אוסף הוא מה שאנו מכנים קבוצה של מסמכי (Document) MongoDBוהוא המקביל לטבלת RDB.

למרות שאוספים אינם אוכפים סכימה מסוימת, בדרך כלל לכל המסמכים באוסף יש מטרה דומה או קשורה אחת לשניה.

- **מסמך – Document**-מסמך הוא קבוצה של צמדי מפתח : ערך,

(key : value כמו ב- JSON (ויש להם סכימה דינמית, מה שאומר

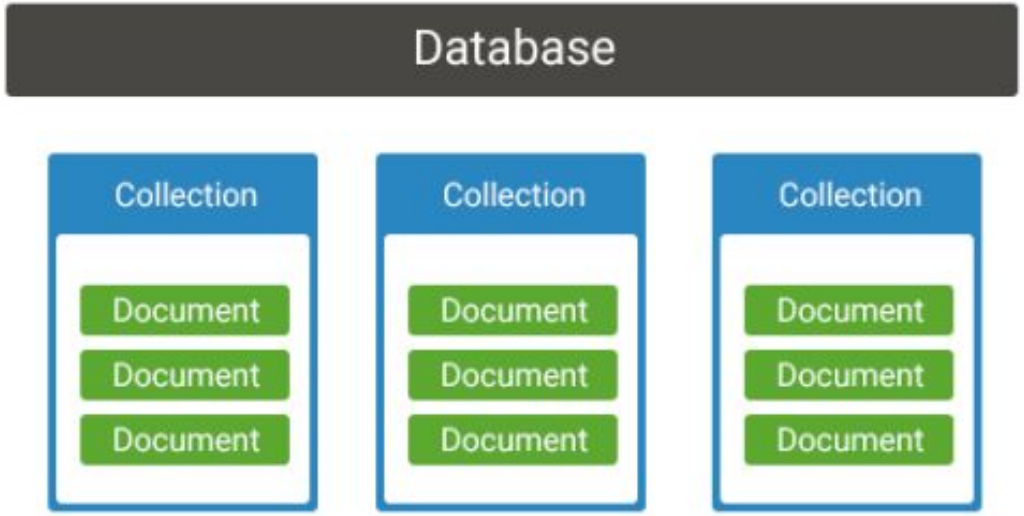
שמסמכים באותו אוסף לא בהכרח צריכים להיות בעלי אותה קבוצה של שדות או מבנה.

Structure

Collection (אוסף)

Collection הוא פשוט קבוצה של documents (מסמכים) ב-MongoDB.

ב- RDB, collection הוא שווה ערך ל- table.



השוואה בין MongoDB ל SQL

- פורמט נתונים: JSON (BSON) לעומת טבלאות -
- גמישות סכימה: סכימה גמישה לעומת קבועה -
- סקלאביליות: אופקית (MongoDB) מול אנכית (SQL) -
- שימושים עיקריים: Big Data, אפליקציות Web -

פורמט נתונים: JSON (BSON)

- What is BSON?
- BSON is a binary encoded Javascript Object Notation (JSON)—a textual object notation widely used to transmit and store data across web based applications. JSON is easier to understand as it is human-readable, but compared to BSON, it supports fewer data types. BSON encodes type and length information, too, making it easier for machines to parse.
- What Does BSON Stand For?
- aBSON stands for Binary Javascript Object Notation. It is a binary-encoded serialization of JSON documents. BSON has been extended to add some optional non-JSON-native data types, like dates and binary data.

פורמט נתונים: JSON (BSON)

```
{
   _id: ObjectId(7af41fd8902g)
   firstName: 'Aviad',
   lastName: 'Derli',
   date-of-birth: new Date("1991-04-13"),
   department: 'Legal',
   children: [
      {
         firstName:'Orel',
         lastName: 'Derli',
         dateCreated: new Date("2016-12-20")
      },
      {
         firstName:'Nuriel David',
         lastName: 'Derli',
         dateCreated: new Date("2021-10-01")
      }
   ]
}
```

CRUD -MongoDB

- יצירה:db.collection.insertOne()
- קריאה: db.collection.find()
- עדכון:db.collection.updateOne()
- מחיקה: db.collection.deleteOne()

יצירה:db.collection.insertOne()

מה זה? הסבר

MongoDB insertOne() Method

The MongoDB insertOne() method is used to add a single document to a .collection

It adds the document to the specified collection and assigns it a unique _id if one is .not provided

We can insert documents with or without the _id field. If we insert a document in the collection without the _id field, MongoDB will automatically add an _id field .and assign it with a unique ObjectId

if we insert a document with the _id field, then the value of the _id field must be .unique to avoid the duplicate key error

.This method can also throw either writeError or writeConcernError exception

יצירה:db.collection.insertOne()

טמפלייט

Syntax:

```
db.Collection_name.insertOne(
<document>,
{
    writeConcern: <document>
})
```

יצירה:db.collection.insertOne()

דוגמה

```
db.student.insertOne({Name: "Akshay", Marks: 500})
```

Output:

```
> use gfg
switched to db gfg
> db.student.insertOne({Name: "Akshay", Marks: 500})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("600e8c710cf217478ba93565")
}
> db.student.find().pretty()
{
        "_id" : ObjectId("600e8c710cf217478ba93565"),
        "Name" : "Akshay",
        "Marks" : 500
}
>
```

קריאה: db.collection.find()

מה זה? הסבר

find() method in MongoDB is a tool for retrieving documents from a collection. It supports various query operators and enabling complex queries. It also allows selecting specific fields to optimize data transfer and benefits from .automatic indexing for better performance

()MongoDB **Find**

.It is a primary method for retrieving documents from a collection in MongoDB

It Supports a wide range of query operators, including comparison, logical and .element operators also allowing for complex queries

It allows us to specify which fields to include or exclude in the result set and .reduce the amount of data transferred

It automatically uses indexes to optimize query performance also improving .speed and efficiency

קריאה: db.collection.find()

טמפלייט

Syntax:

```
db.Collection_name.find(selection_criteria, projection,options)
```

Explanation:

- **`db.Collection_name.find`:** This specifies the database (`db`) and the collecttion (`Collection_name`) we are querying from and initiates a find operation.
- **`(selection_criteria, projection, options)`:** These are optional arguments that refine the results:
  - **`selection_criteria` (document):** Defines which documents to find based on specific conditions. An empty document (`{}`) retrieves all documents.
  - **`projection` (document):** Specifies which fields to include or exclude from the returned documents.
  - **`options` (document):** Allows for additional options like sorting or limiting results.

קריאה: db.collection.find()

דוגמה 1

Example 1

Let's Find all student records from the "**student**" collection where the student's age is exactly **18.**

```
db.student.find()
```

Output:

```
> db.student.find().pretty()
{
        "_id" : ObjectId("60227eaff19652db63812e8d"),
        "name" : "Akshay",
        "age" : 18
}
{
        "_id" : ObjectId("60227eaff19652db63812e8e"),
        "name" : "Bablue",
        "age" : 17,
        "score" : {
                "math" : 230,
                "science" : 234
        }
}
{
        "_id" : ObjectId("60227eaff19652db63812e8f"),
        "name" : "Chandhan",
        "age" : 18
}
>
```

קריאה: db.collection.find()

דוגמה 2

Example 2

Find all the documents present in the collection:

```
db.student.find({age:18})
```

Output:

```
> db.student.find({age:18})
{ "_id" : ObjectId("60227eaff19652db63812e8d"), "name" : "Akshay", "age" : 18 }
{ "_id" : ObjectId("60227eaff19652db63812e8f"), "name" : "Chandhan", "age" : 18
}
>
```

קריאה: db.collection.find()

דוגמה 3

Example 3

Let's Find student records from the "**student**" collection where the student's **math** score is **230** and **science** score is **234**.

```
db.student.find({score:{math: 230, science: 234}})
```

Output:

```
> db.student.find({score:{math: 230, science: 234}})
{ "_id" : ObjectId("60227eaff19652db63812e8e"), "name" : "Bablue", "age" : 17, "
score" : { "math" : 230, "science" : 234 } }
>
```

SQL מול MongoDB:

יצירת שדה/מידע+פלט מידע-דוגמה

SQL:

CREATE TABLE Students (ID INT, Name VARCHAR(50));

INSERT INTO Students VALUES (1, 'Alice');

SELECT * FROM Students;

MongoDB:

db.Students.insertOne({ "_id": 1, "Name": "Alice" });

db.Students.find();

עדכון:db.collection.updateOne()

מה זה? הסבר

MongoDB’s updateOne() method provides a powerful way to update a single document in a collection based on specified criteria. This method is particularly useful when Accuracy is needed in modifying specific .documents without affecting others

MongoDB updateOne()

MongoDB updateOne() is a method that is used to update a single .document that matches a specified filter

We can target and update exactly one document that matches our filter .criteria

It Compared to methods such as updateMany() which updates multiple documents at once updateOne() is more efficient when we only need to .update a single document

עדכון: db.collection.updateOne()

טמפלייט

Syntax

```
db.collection.updateOne(<filter>, <update>, {
   upsert: <boolean>,
   writeConcern: <document>,
   collation: <document>,
   arrayFilters: [<filterdocument1>, ...],
   hint: <document|string> // Available starting in MongoDB 4.2.1
})
```

Parameters:

- **<filter>:** Specifies the selection criteria for the update.
- **<update>:** A document or pipeline containing modifications to apply.

Optional Parameters:

- **upsert:** The default value of this parameter is false. When it is true it will make a new document in the collection when no document matches the given condition in the update method.
- **writeConcern:** It is only used when you do not want to use the default write concern. The type of this parameter is a document.
- **collation:** It specifies the use of the collation for operations. It allows users to specify the language-specific rules for string comparison like rules for lettercase and accent marks. The type of this parameter is a document.
- **arrayFilters:** It is an array of filter documents that indicates which array elements to modify for an update operation on an array field. The type of this parameter is an array.
- **hint:** It is a document or field that specifies the index to use to support the filter. It can take an index specification document or the index name string and if you specify an index that does not exist, then it will give an error.

עדכון:db.collection.updateOne()

דוגמה 1

Query:

```
db.student.updateOne({name: "Annu"}, {$set:{age:25}})
```

Output:

```
> db.student.updateOne({name: "Annu"}, {$set:{age:25}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("600e9afd0cf217478ba93566"),
        "name" : "Annu",
        "age" : 25
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93567"),
        "name" : "Bhannu",
        "age" : 24
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93568"),
        "name" : "Bhannu",
        "age" : 24
}
>
```

עדכון:db.collection.updateOne()

דוגמה 2

Example 2: Update a String Value in the Document

Update the name of the first matched document whose name is Bhannu to Babita

Query:

```
db.student.updateOne({name:"Bhannu"},{$set:{name:"Babita"}})
```

Output:

```
> db.student.updateOne({name: "Bhannu"}, {$set:{name: "Babita"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("600e9afd0cf217478ba93566"),
        "name" : "Annu",
        "age" : 25
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93567"),
        "name" : "Babita",
        "age" : 24
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93568"),
        "name" : "Bhannu",
        "age" : 24
}
```

עדכון:db.collection.updateOne()

דוגמה 3

Example 3: Insert a new field in the document

Query:

```
db.student.updateOne({name: "Bhannu"}, {$set:{class: 3}})
```

Output

```
> db.student.updateOne({name: "Bhannu"}, {$set:{class: 3}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("600e9afd0cf217478ba93566"),
        "name" : "Annu",
        "age" : 25
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93567"),
        "name" : "Babita",
        "age" : 24
}
{
        "_id" : ObjectId("600e9afd0cf217478ba93568"),
        "name" : "Bhannu",
        "age" : 24,
        "class" : 3
}
```

מחיקה: db.collection.deleteOne()

מה זה? הסבר

The deleteOne() method in MongoDB is used to remove a single .document from a collection that matches a specified filter

MongoDB deleteone()

The deleteOne() method in MongoDB is used to remove a single .document from a collection that matches a specified filter

This method is particularly useful when we want to delete one .specific document based on certain criteria

Using deleteOne() effectively allows us to manage our MongoDB collections by removing specific documents based on our criteria, .maintaining the integrity and relevance of our data

מחיקה: db.collection.deleteOne()

טמפלייט

Syntax

```
db.Collection_name.deleteOne(
selection_criteria:<document>,
{
    writeConcern: <document>,
    collation: <document>,
    hint: <document|string>
})
```

Parameter

- **Selection criteria:** Specifies criteria to delete documents. The type of this parameter is a document.

Optional Parameters

- **writeConcern:** It is only used when you do not want to use the default write concern. The type of this parameter is a document.
- **Collation:** It specifies the use of the collation for operations. It allows users to specify the language-specific rules for string comparison like rules for lettercase and accent marks. The type of this parameter is a document.
- **hint:** It is a document or field that specifies the index to use to support the filter. It can take an index specification document or the index name string and if you specify an index that does not exist, then it will give an error.

מחיקה: db.collection.deleteOne()

דוגמה 1

Delete the first matched document whose age is 17:

```
db.student.deleteOne({age:17})
```

Here, we delete the first document that matches the filter query(i.e., age:17) from the student collection using the deleteOne() method.

```
> db.student.deleteOne({age:17})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
```

After deletion:

```
> db.student.find().pretty()
{
        "_id" : ObjectId("601ef9405f8d812f4c95ec39"),
        "name" : "Nikhil",
        "age" : 21
}
{
        "_id" : ObjectId("601ef9405f8d812f4c95ec3b"),
        "name" : "Sagar",
        "age" : 17
}
{
        "_id" : ObjectId("601ef9405f8d812f4c95ec3c"),
        "name" : "Sahil",
        "age" : 19
}
```

מחיקה: db.collection.deleteOne()

דוגמה 2

Example 2: Deleting a Document from the 'student' Collection Based on Age

Delete the first matched document whose age is less than 18:

```
db.student.deleteOne({age:{$lt:18}})
```

Here, we delete the first document that matches the filter query(i.e., {age:{$lt:18}}) from the student collection using the deleteOne() method.

Note: $lt operator means less than.

```
> db.student.deleteOne({age:{$lt:18}})
{ "acknowledged" : true, "deletedCount" : 1 }
```

After deletion:

```
> db.student.deleteOne({age:{$lt:18}})
{ "acknowledged" : true, "deletedCount" : 1 }
> db.student.find().pretty()
{
        "_id" : ObjectId("601ef9405f8d812f4c95ec39"),
        "name" : "Nikhil",
        "age" : 21
}
{
        "_id" : ObjectId("601ef9405f8d812f4c95ec3c"),
        "name" : "Sahil",
        "age" : 19
}
>
```

SQL מול MongoDB:

עדכון שדה/מידע+מחיקה מידע-דוגמה

:SQL

UPDATE table_name SET column1 = value1, column2 = value2,...
WHERE condition

;DELETE FROM table_name WHERE condition

:MongoDB

db.student.updateOne({name: "Annu"}, {$set:{age:25}})

db.collection.deleteOne()

טיפוסים

Data Type	Description
Array	This data type can store multiple values in a single key.
Binary	This data type can store binary data.
Boolean	This data type can store boolean values like true (1) or false (0).
Code	This data type can store Javascript code into the document.
Date	This data type is the date and/or time in Unix time format.
Double	This data type can store floating point numerical values like 3.14.
Integer	This data type can store whole numeric values. The value can 32-bit or 64-bit depending on the architecture of your server.
Min/Max Keys	This data type can compare a value against the lowest or highest elements.
Null	This data type can store null values. A null value is the absense of a value.
Object	This data type is used for embedded documents.
Regular expression	This data type is used to store **regular expressions** .
String	This data type can store valid UTF 8 characters, words, sentences, numbers etc. which must be wrapped in quotes. It's the most used data type in MongoDB.
Symbol	This data type is typically used for languages that use a specific type.

תחילת עבודה עם MongoDB

- 1. הורדה והתקנת MongoDB
- 2. יצירת מסד נתונים באמצעות הפקודה `use`.
- 3. הוספת מסמכים לקולקציה באמצעות המתודות `insertOne` ו-`insertMany`.

תחילת עבודה עם MongoDB

יצירתCollection

כדי ליצור אוסף, אנו משתמשים בשיטת **createCollection.db()** .

פונקציה זו מקבלת 2 ארגומנטים.

הארגומנט הראשון הוא שם האוסף שאנו רוצים ליצור,

הארגומנט השני מאפשר לנו לציין אפשרויות כלשהן לגבי גודל זיכרון ואינדקס .

db.createCollection(name, Options)

הארגומנט **Options** הוא **document** והוא אופציונלי לחלוטין. אם אנחנו רוצים להשתמש באפשרויות ברירת

המחדל, אנחנו יכולים פשוט להשמיט את הארגומנט השני.

יש לזכור שמחרוזות, כמו שמות, חייבים להיות עטופים במירכאות.

db.createCollection("employee")

נקבל חזרה:

{{ ok" : 1"

אם בכל זאת נרצה לממש את **option**, נוכל ל עיין בתיעוד הרשמי של MongoDB עבור שיטה זו.

פקודות בסיסיות ב MongoDB

- מעבר/יצירת בסיס נתונים `use <DatabaseName>` -
- מציג את בסיס הנתונים הפעיל. `db`
- מציג את כל בסיסי הנתונים. `show dbs`
- מציג את כל הקולקציות בבסיס הנתונים. `show collections`
- מוסיף מסמך אחד. `db.<CollectionName>.insertOne()`
- מציג את כל המסמכים בקולקציה `db.<CollectionName>.find()`

פקודות בסיסיות ב MongoDB

db.help(): help on db methods
db.mycoll.help(): help on collection methods
sh.help() :sharding helpers
...
show dbs: show database names
show collections: show collections in current database
show users: show users in current database

זמן תרגול

- https://www.w3schools.com/mongodb/exercise.php?filename=exercise_aggregations_intro1
- לבצע עד (CRUD).delete

תרגיל 1

יצירת בסיס נתונים

- - צור בסיס נתונים בשם University.
- - הוסף קולקשיין בשם Students.
- - הוסף מסמכים: { "id": 1, "Name": "Alice", "Age_"
- : 22 }

תרגיל 2

עדכון מידע

- עדכן את הגיל של Bob ל -26.

הסבר על פקודות בסיסיות

- `use <DatabaseName>` -:
- אם בסיס הנתונים לא קיים, הוא ייווצר.
- `show dbs` -:
- מציג רשימה של כל בסיסי הנתונים הקיימים.
- `db.<CollectionName>.find()` -:
- מאפשר לראות את כל המסמכים או לבצע שאילתות סינון.
- `db.<CollectionName>.insertOne()` -:
- משמש להוספת מסמך אחד לקולקציה.

סיכום

- MongoDB - היא מערכת NoSQL מבוססת מסמכים(Collections).
- מתאימה לאפליקציות מודרניות הדורשות גמישות.
- קלות שימוש ולמידה עם פונקציות CRUD פשוטות.

יצירה:db.collection.insertOne()

קריאה: db.collection.find()

עדכון:db.collection.updateOne()

מחיקה: db.collection.deleteOne()

סיכום

MongoDB מול SQL:

CRUD	MONGODB	**MYSQL**
Create	()db.createCollection	CREATE TABLE <<TABLE
Insert	db.collection.insertMan ()y	INSERT INTO <<TABLE
Select	()db.collection.find	SELECT * FROM <<TABLE
Update	db.collection.updateOn ()e	<UPDATE <TABLE
Delete	db.collection.deleteOne ()	DELETE FROM <<TABLE

תרגילי בית

- 1. ** יצירת מסד נתונים חדש:** צור מסד נתונים בשם `Library`, הוסף קולקשיין `Books` עם לפחות 5 ספרים הכוללים שדות כמו `Year`, `Author`, `Title` ו-`Genre`.
- 2. ** שאילתת סינון:** מצא את כל הספרים מהז'אנר `Fiction` שפורסמו לאחר שנת 2010.
- 3. ** עדכון מסמכים:** שנה את הז'אנר של כל הספרים מאת מחבר מסוים ל-`Classic`.
- 4. ** מחיקת מסמכים:** מחק את כל הספרים שפורסמו לפני שנת 2000.
- 5. ** ספירת מסמכים:** השתמש בפקודה `count` כדי לספור את כמות הספרים מכל ז'אנר בקולקציה.