Informe de MongoDB - Taller CRUD

Estudiante: Jhon Esteban Josa Quinchoa
Docente: Brayan Arcos
Fecha: 16/10/2024

Instituto tecnológico del putumayo
Almacenamiento masivo
Octavo semestre

Índice

1. Resumen Ejecutivo

Este informe detalla la realización de un taller práctico en MongoDB que se centra en las operaciones CRUD (Crear, Leer, Actualizar y Eliminar). En el taller, se trabajó con varios conjuntos de datos en formato JSON para practicar la manipulación de datos en una base de datos NoSQL. El objetivo principal fue consolidar conocimientos sobre cómo insertar, consultar, actualizar y eliminar documentos en MongoDB, utilizando funcionalidades que permiten flexibilidad y eficiencia en el manejo de datos.

2. Introducción

Contexto y Motivación

La creciente necesidad de gestionar grandes volúmenes de datos ha llevado al uso de bases de datos NoSQL, como MongoDB, que permiten un almacenamiento eficiente y una estructura de datos flexible. MongoDB se ha vuelto popular en el mundo del desarrollo debido a su capacidad de manejar datos semi-estructurados y su rendimiento en aplicaciones de alto volumen de datos. Este taller se enfoca en aplicar las principales operaciones CRUD en MongoDB para reforzar la comprensión práctica de esta tecnología.

Alcance del Informe

Este informe documenta las actividades realizadas en el taller de MongoDB, centrándose en las operaciones CRUD aplicadas a colecciones de datos JSON. Se cubren aspectos de creación, consulta, actualización y eliminación de documentos dentro de MongoDB.

Objetivos

El objetivo de este informe es demostrar la implementación práctica de operaciones CRUD en MongoDB a través de un conjunto de ejercicios. Con estas actividades, se busca entender cómo MongoDB permite la manipulación de datos no relacionales de forma rápida y eficiente.

3. Metodología

Herramientas Utilizadas
se utilizó:

- MongoDB como el sistema de gestión de bases de datos.
- MongoDB Compass para la visualización y administración de los datos.
- Studio 3D, como entorno grafico

Procedimientos
Creación de colecciones y carga de datos: Se crearon diversas colecciones en MongoDB y se cargaron archivos JSON que contenían datos de productos, episodios de series, calificaciones, etc.
Operaciones CRUD: En cada colección, se realizaron operaciones de:
Inserción de datos mediante insertOne() y insertMany().
Consulta de datos con find() y findOne().
Actualización de documentos usando updateOne() y updateMany().
Eliminación de documentos con deleteOne() y deleteMany().

4. Desarrollo del Informe

Descripción de la Base de Datos Escuela

La base de datos escuela incluye cuatro colecciones principales: estudiantes, profesores, cursos e inscripciones. Cada una de estas colecciones fue diseñada para reflejar relaciones específicas, aprovechando la flexibilidad de MongoDB para manejar datos embebidos y referencias.

Diseño de la Base de Datos

El diseño de la base de datos incluyó la normalización de los datos y la definición de relaciones adecuadas entre las colecciones. Por ejemplo, se diseñó una relación uno a uno entre estudiantes y tutores, una relación uno a muchos entre profesores y cursos, y una relación muchos a muchos para las inscripciones de estudiantes en cursos.

Métodos de Captura

Los métodos de captura consistieron en la utilización de archivos JSON para la creación de las colecciones. Para cada archivo, se realizaron operaciones de inserción de datos y luego se aplicaron funciones de actualización. En el caso de la base de datos escuela, se utilizaron las siguientes funciones:

- updateOne() con $set para asignar tutores a los estudiantes.
- updateOne() con upsert y setOnInsert para crear cursos si no existían.
- updateOne() con $addToSet y $each para añadir cursos a la inscripción de estudiantes sin duplicar datos.

Descripción de las Colecciones

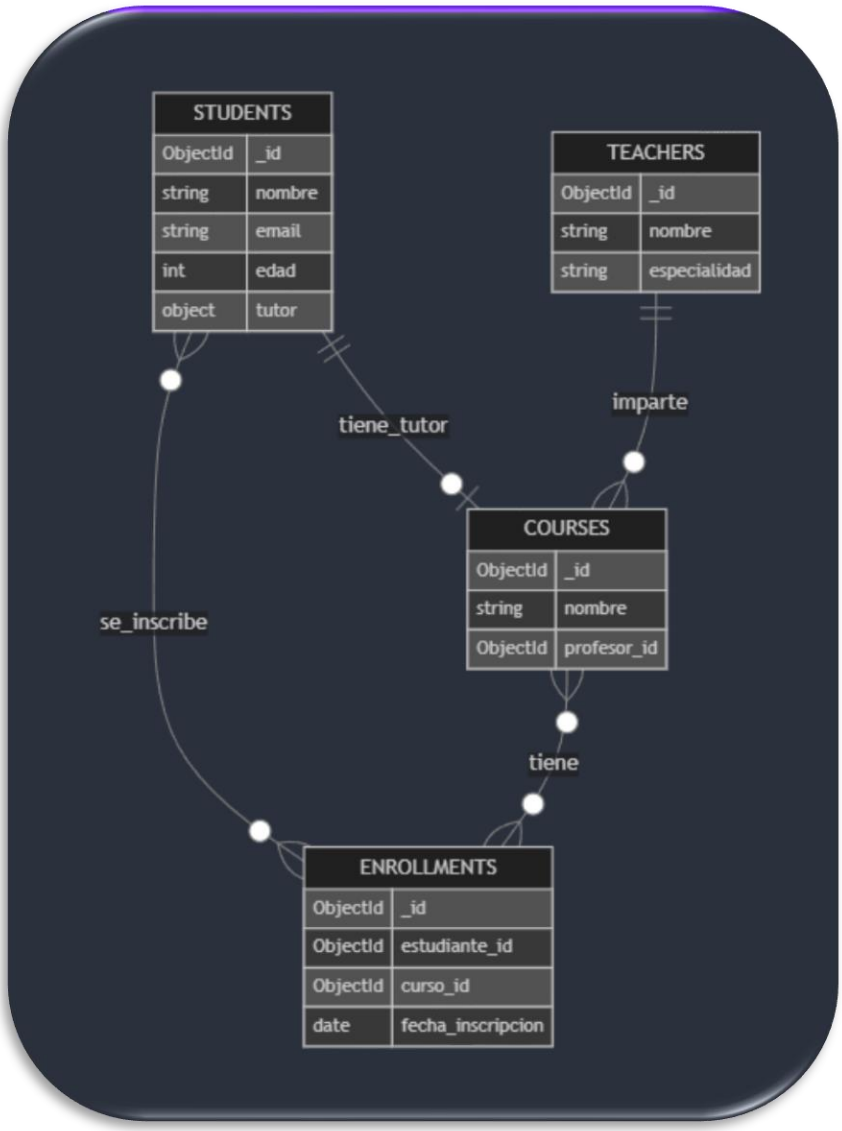
La base de datos escuela consta de las siguientes colecciones principales:

- **students**: Almacena datos básicos de los estudiantes, como nombre, email y edad.
- **teachers**: Contiene los datos de los profesores y sus especialidades.
- **courses**: Define los cursos que ofrece la escuela y el profesor asignado a cada curso.
- **enrollments**: Registra las inscripciones de estudiantes en cursos, permitiendo una relación de muchos a muchos.

Relaciones Implementadas

- **Relación Uno a Uno**: Cada estudiante tiene asignado un tutor en el campo tutor, el cual está embebido dentro de la colección students.
- **Relación Uno a Muchos**: La colección courses contiene un campo profesor_id, que hace referencia al profesor que imparte el curso. Cada profesor puede impartir múltiples cursos.
- **Relación Muchos a Muchos**: La colección enrollments vincula estudiantes y cursos, registrando las inscripciones en cursos a través de estudiante_id y curso_id.

Diagrama no racional bd_escuela



Inserción de Datos

Los datos se insertaron en cada colección usando los métodos insertOne() y insertMany() para facilitar la carga inicial de los documentos. Por ejemplo, para insertar estudiantes en la colección students:

```
db.students.insertMany([
    { "_id": ObjectId(), "nombre": "Ana Pérez", "email": "ana.perez@escuela.com", "edad": 15 },
    { "_id": ObjectId(), "nombre": "Carlos López", "email": "carlos.lopez@escuela.com", "edad": 16 }
]);
```

Actualización de Datos

Se emplearon funciones avanzadas para actualizar documentos, incluyendo:

- **Asignación de tutor** a un estudiante mediante updateOne() con $set:

```
db.students.updateOne(
    { "nombre": "Ana Pérez" },
    { $set: { "tutor": { "nombre": "María López", "telefono": "555-1234" } } }
);
```

- **Creación de curso con upsert y setOnInsert** para añadir un curso si no existe:

```
db.courses.updateOne(
    { "_id": ObjectId("507f1f77bcf86cd799439015") },
    {
        $setOnInsert: {
            "nombre": "Química",
            "profesor_id": ObjectId("507f1f77bcf86cd799439012")
        }
    },
    { upsert: true }
);
```

- **Añadir cursos a la inscripción de un estudiante** con $addToSet y $each:

```
db.enrollments.updateOne(
    { "estudiante_id": ObjectId("507f1f77bcf86cd799439013") },
    { $addToSet: { "cursos": { $each: [ObjectId("507f1f77bcf86cd799439011"), ObjectId("507f1f77bcf86cd799439012")] } } }
);
```

Opiniones

El diseño y desarrollo de la base de datos escuela demuestra la versatilidad de MongoDB para manejar relaciones no tradicionales entre datos. Con las operaciones CRUD y herramientas de actualización avanzadas, MongoDB ofrece una estructura de datos flexible y escalable, ideal para aplicaciones educativas y otros sistemas que requieren un manejo dinámico de datos.

Descripción del Taller CRUD

El taller consistió en manipular datos a través de operaciones CRUD en varias colecciones, tales como:

- Productos: Inserción y actualización de precios de productos.
- Episodios de Series: Consulta de episodios con diferentes criterios, como la calificación promedio.
- Calificaciones: Gestión de calificaciones de estudiantes en diferentes materias.

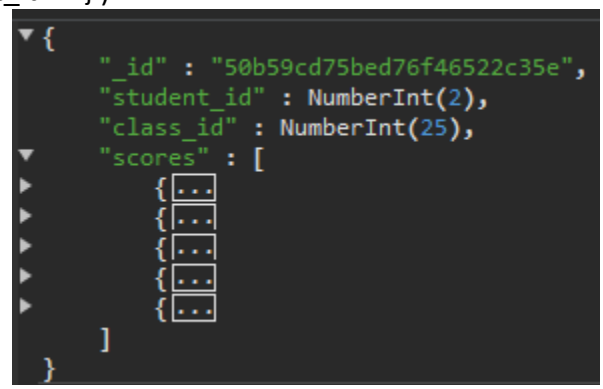
Consultas NoSQL

1. Operaciones de Lectura (READ)

1.1. Encontrar documentos por student_id

Encuentra todos los documentos en la colección grades donde el student_id es 2.

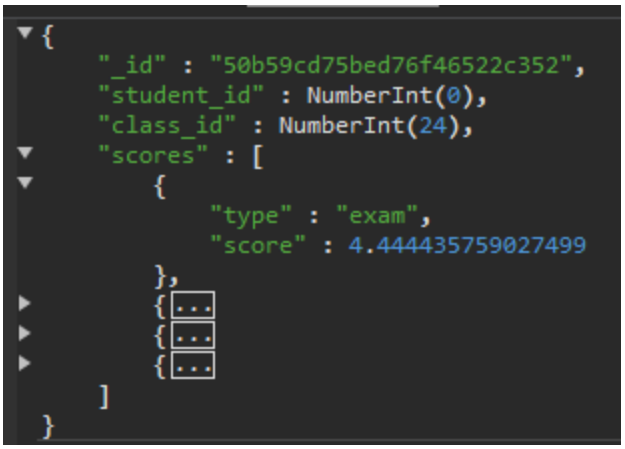
```
db.grades.find( { student_id: 2 } )
```



1.2. Encontrar documentos con puntajes específicos

Encuentra documentos donde el primer score (scores.0) es menor o igual a 10.

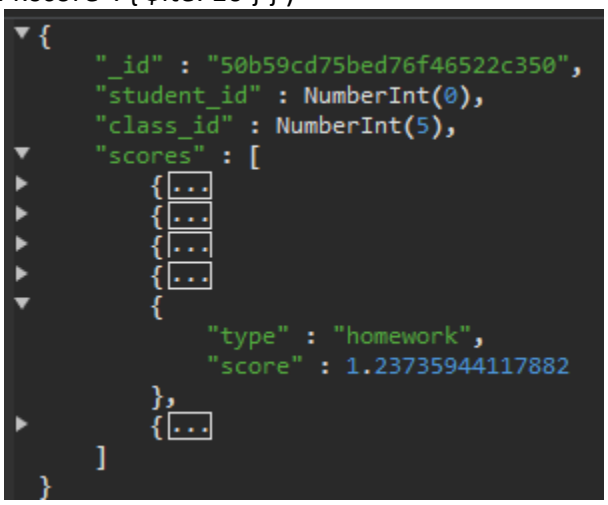
```
db.grades.find( { "scores.0.score": { $lte: 10 } } )
```



1.3. Encontrar documentos por el quinto score

Encuentra documentos donde el quinto score (scores.4) es menor o igual a 10.

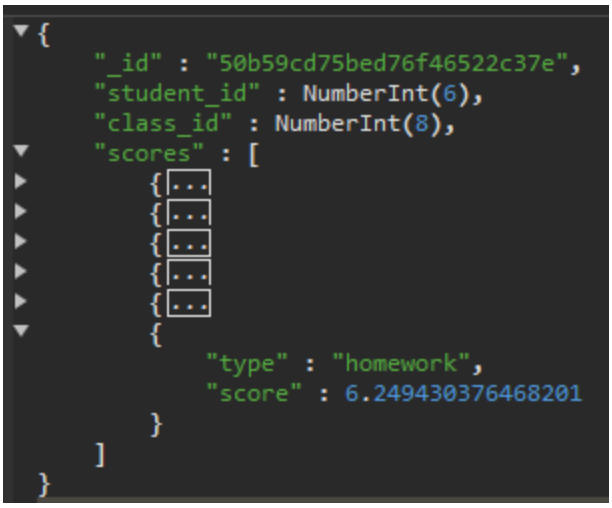
db.grades.find({ "scores.4.score": { $lte: 10 } })



1.4. Encontrar documentos por el sexto score

Encuentra documentos donde el sexto score (scores.5) es menor o igual a 10.

db.grades.find({ "scores.5.score": { $lte: 10 } })



1.5. Encontrar documentos por el séptimo score

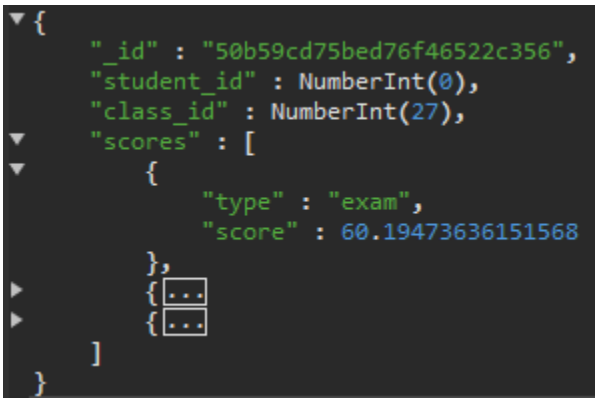
Encuentra documentos donde el séptimo score (scores.6) es menor o igual a 10.

db.grades.find({ "scores.6.score": { $lte: 10 } })

1.6. Rango de scores

Encuentra documentos donde el primer score (scores.0) está entre 60 y 61 (inclusive).

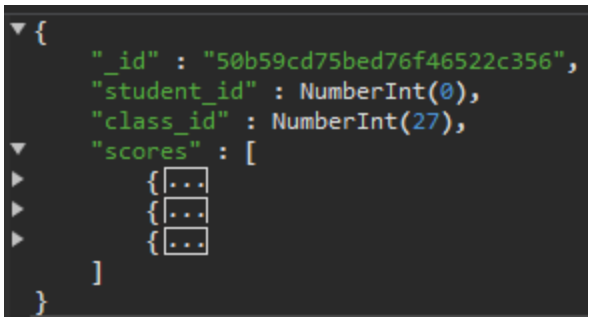
db.grades.find({ "scores.0.score": { $gte: 60, $lte: 61 } })



1.7. Rango de scores y ordenación

Encuentra documentos donde el primer score (scores.0) está entre 60 y 61 y ordena los resultados por student_id en orden ascendente.

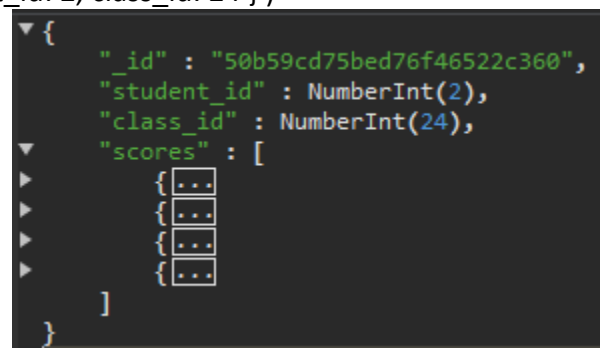
db.grades.find({ "scores.0.score": { $gte: 60, $lte: 61 } }).sort({ student_id: 1 })



1.8. Buscar documentos con múltiples criterios

Encuentra documentos donde el student_id es 2 y el class_id es 24.

db.grades.find({ student_id: 2, class_id: 24 })



1.9. Buscar por rango de scores en una clase

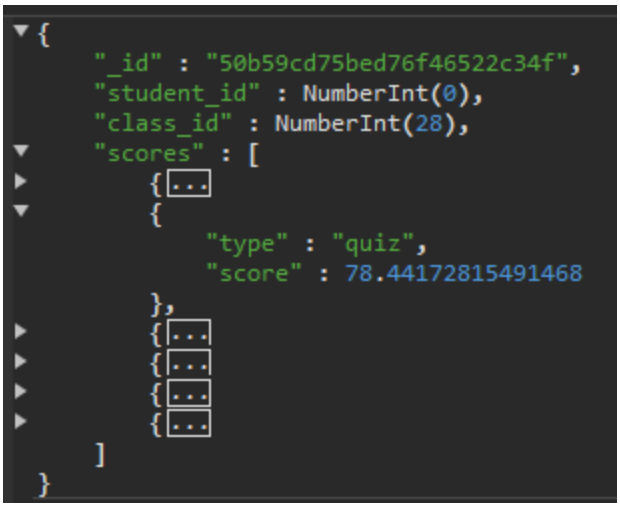
Encuentra documentos con class_id 20 donde el primer score (scores.0) está entre 15 y 30 (inclusive).

db.grades.find({ class_id: 20, $and: [{"scores.0.score": { $gte: 15 } }, { "scores.0.score": { $lte: 30 } }] })

1.10. Buscar por tipo de score 'quiz'

Encuentra documentos que tienen al menos un score tipo 'quiz' con un score mayor o igual a 50.

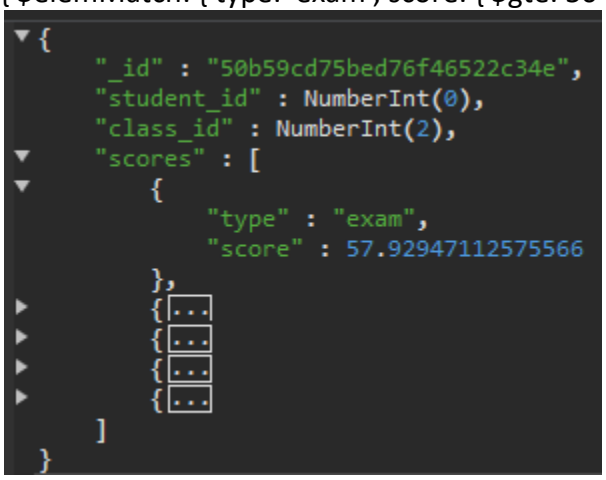
db.grades.find({ scores: { $elemMatch: { type: 'quiz', score: { $gte: 50 } } } })



1.11. Buscar por tipo de score 'exam'

Encuentra documentos que tienen al menos un score tipo 'exam' con un score mayor o igual a 50.

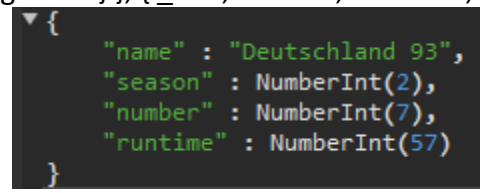
db.grades.find({ scores: { $elemMatch: { type: 'exam', score: { $gte: 50 } } } })



1.12. Filtrar por runtime en la colección "Narcos"

Encuentra documentos en la colección Narcos donde el runtime es mayor o igual a 55 y solo devuelve name, season, number y runtime (excluye _id).

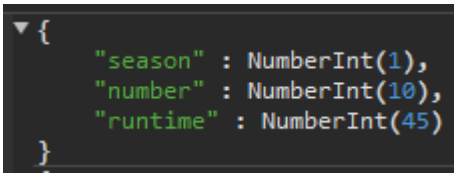
db.Narcos.find({ runtime: { $gte: 55 } }, { _id:0, name:1, season:1, number:1, runtime:1 })



1.13. Ordenar por season y number

Encuentra documentos donde el runtime es mayor o igual a 15, devuelve season, number y runtime, y ordena por season ascendente y number descendente.

db.Narcos.find({ runtime: { $gte: 15 } }, { _id:0, season:1, number:1 }).sort({ season:1, number:-1 })



1.14. Filtrar por tipo de dato

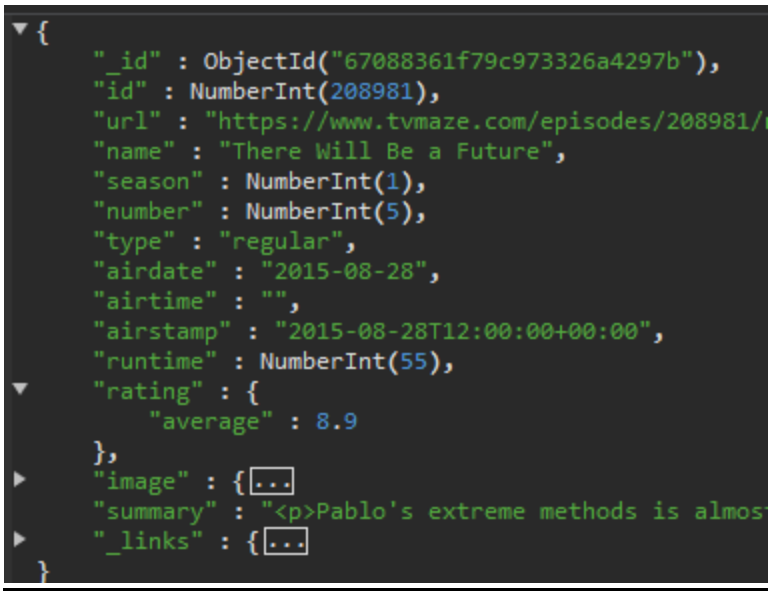
Encuentra documentos donde el campo season es de tipo número.

db.Narcos.find({ season: { $type: 'number' } })

1.15. Existencia de un campo

Encuentra documentos que tienen el campo rating.

db.Narcos.find({ rating: { $exists: 1 } })



1.16. Filtrar por tipo de dato de rating

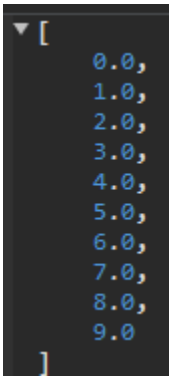
Encuentra documentos que tienen el campo rating y que es de tipo string.

db.Narcos.find({ rating: { $exists: 1 }, rating: { $type: "string" } })

1.17. Valores únicos de student_id

Encuentra valores únicos del campo student_id en la colección grades.

db.grades.distinct("student_id")



1.18. Contar documentos en grades

Cuenta el número total de documentos en la colección grades.

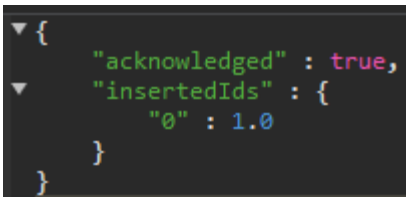
db.grades.countDocuments()

2. Operaciones de Creación (CREATE)

2.1. Inserción de un documento usando insert

Inserta un documento usando el método insert (método obsoleto).

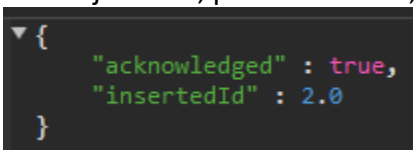
db.LabItp01.insert({ _id: 1, name: "pepe", phone: 123456, class: [20, 22, 25] })



2.2. Inserción de un documento usando insertOne

Inserta un documento usando insertOne.

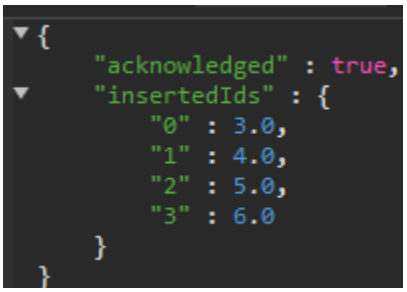
db.LabItp01.insertOne({ _id: 2, name: "juanito", phone: 654789, class: [10, 12, 15] })



2.3. Inserción de múltiples documentos

Inserta múltiples documentos usando insertMany.

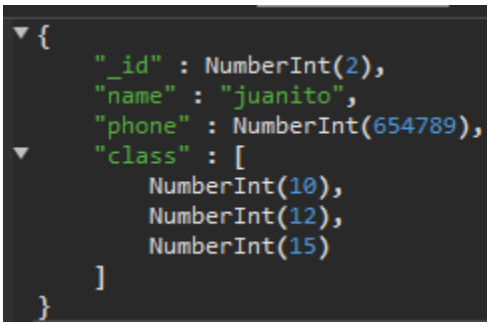
```
db.LabItp01.insertMany([
        { _id: 3, name: "carlito", phone: 639852, class: [11, 10] },
        { _id: 4, name: "camilito", phone: 741258, class: [15] },
        { _id: 5, name: "anita", phone: 852741, class: [10] },
        { _id: 6, name: "joselito", phone: 1254896, class: [55, 458, 236, 20, 22, 10, 15] }
])
```



2.4. Consultar documentos con un valor en un array

Consulta documentos donde el array class contiene el valor 10.

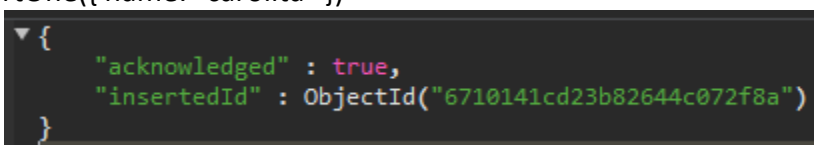
db.LabItp01.find({ class: 10 })



2.5. Inserción de un documento simple

Inserta un documento simple con solo el campo name.

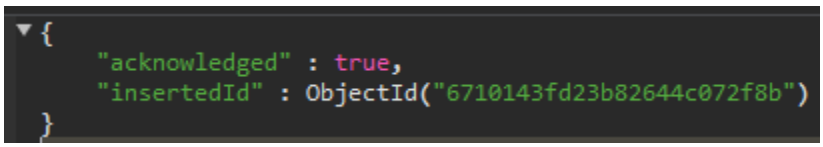
db.LabItp02.insertOne({ name: "carolita" })



2.6. Inserción de documento con subdocumento

Inserta otro documento con más campos, incluyendo un subdocumento en information.

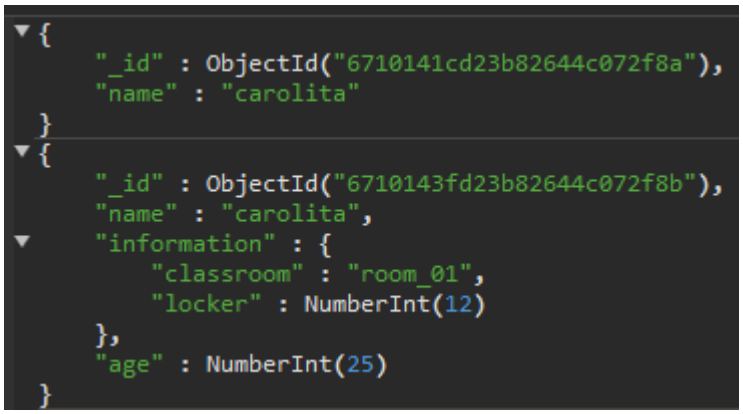
```
db.LabItp02.insertOne({
        name: "carolita",
        information: {
        classroom: "room_01",
        locker: 12
},
age: 25
})
```



2.7. Mostrar todos los documentos en LabItp02

Muestra todos los documentos en la colección LabItp02.

db.LabItp02.find()

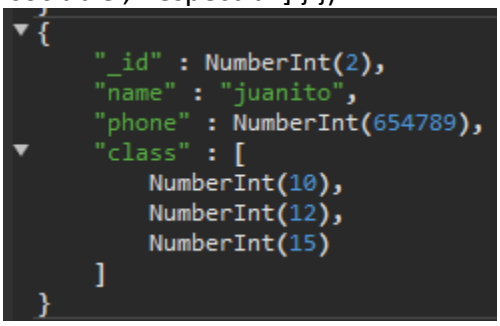


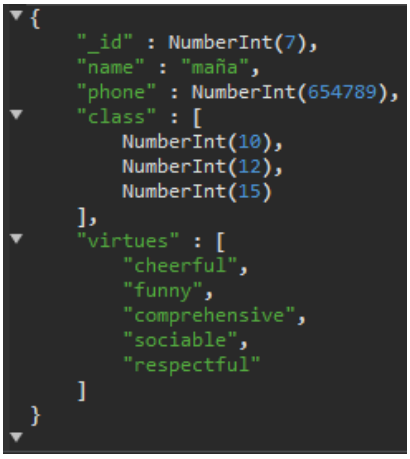
3. Operaciones de Actualización (UPDATE)

3.1. Actualizar un documento

Actualiza el campo virtues del documento con _id: 7.

```
db.LabItp01.updateOne( { _id: 7 }, { $set: { virtues: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'] } })
```

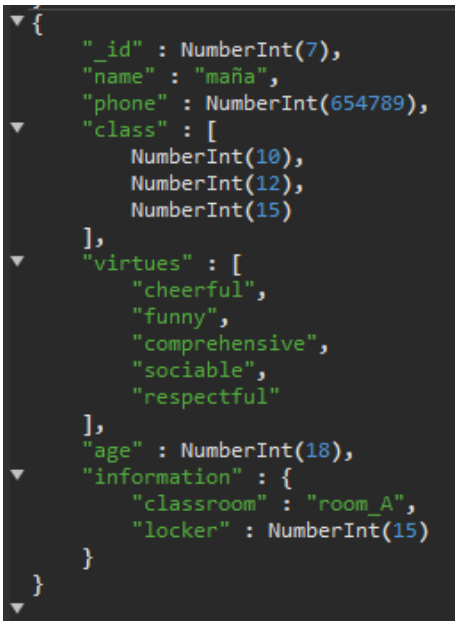




3.2. Actualizar múltiples campos

Actualiza los campos information y age del documento con _id: 7.

```
db.LabItp01.updateOne( { _id: 7 }, { $set: { information: { classroom: "room_A", locker: 15 }, age: 18 } })
```



3.3. Actualizar con fecha de modificación

Actualiza virtues y añade/modifica la fecha lastModified a la fecha actual para el documento con _id: 7.

db.LabItp01.updateOne({ _id: 7 }, { $set: { virtues: ['cheerful', 'funny', 'comprehensive', 'sociable', 'respectful'] }, $currentDate: { lastModified: true } })


```
{
    "_id" : NumberInt(7),
    "name" : "maña",
    "phone" : NumberInt(654789),
    "class" : [
        NumberInt(10),
        NumberInt(12),
        NumberInt(15)
    ],
    "virtues" : [
        "cheerful",
        "funny",
        "comprehensive",
        "sociable",
        "respectful"
    ],
    "age" : NumberInt(18),
    "information" : {
        "classroom" : "room_A",
        "locker" : NumberInt(15)
    },
    "lastModified" : ISODate("2024-10-16T19:40:44.517+0000")
}
```


3.4. Actualización con upsert

Si no existe un documento con _id: 10, lo inserta; de lo contrario, lo actualiza con los nuevos datos.

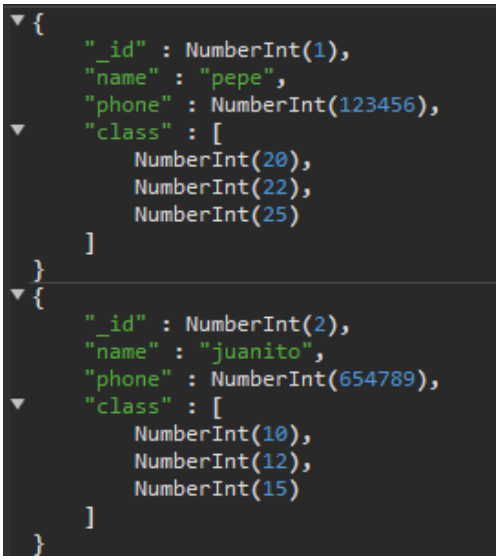
db.LabItp01.updateOne({ _id: 10 }, { $set: { name: "Joan", age: 19, virtues: [], information: {} }, $currentDate: { lastModified: true } }, { upsert: true })

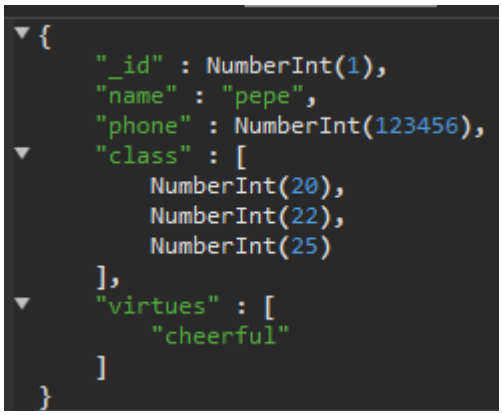

```
{
    "_id" : NumberInt(10),
    "age" : NumberInt(19),
    "information" : {

    },
    "lastModified" : ISODate("2024-10-16T19:41:56.812+0000"),
    "name" : "Joan",
    "virtues" : [

    ]
}
```


3.5. Actualización de múltiples documentos

Actualiza los documentos con _id de 1 a 6, agregando el campo virtues con un único valor.

db.LabItp01.updateMany({ _id: { $in: [1, 2, 3, 4, 5, 6] } }, { $set: { virtues: ['cheerful'] } })

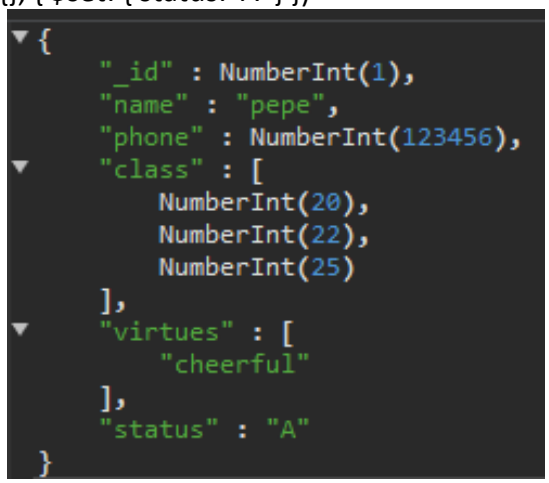




3.6. Actualizar todos los documentos

Actualiza todos los documentos de la colección añadiendo el campo status con valor 'A'.

db.LabItp01.updateMany({}, { $set: { status: 'A' } })



3.7. Actualización por nombre

Actualiza los documentos donde el nombre sea 'pepe' o 'camilito', añadiendo el campo role con valor 'student'.

db.LabItp01.updateMany({ name: { $in: ['pepe', 'camilito'] } }, { $set: { role: 'student' } })

```
{
    "_id" : NumberInt(1),
    "name" : "pepe",
    "phone" : NumberInt(123456),
    "class" : [
        NumberInt(20),
        NumberInt(22),
        NumberInt(25)
    ],
    "virtues" : [
        "cheerful"
    ],
    "status" : "A",
    "role" : "student"
}
```


4. Operaciones de Eliminación (DELETE)

4.1. Eliminar un documento

Elimina un documento en la colección LabItp01 donde el campo name es igual a "carlito".

db.LabItp01.deleteOne({ name: "carlito" })


```
{
    "acknowledged" : true,
    "deletedCount" : 1.0
}
```


4.2. Eliminar un documento en grades

Elimina el primer documento en la colección grades donde el campo student_id es igual a 0.

db.grades.deleteOne({ student_id: 0 })


```
{
    "acknowledged" : true,
    "deletedCount" : 1.0
}
```


4.3. Eliminar múltiples documentos

Elimina todos los documentos en la colección grades donde el campo student_id es igual a 0.

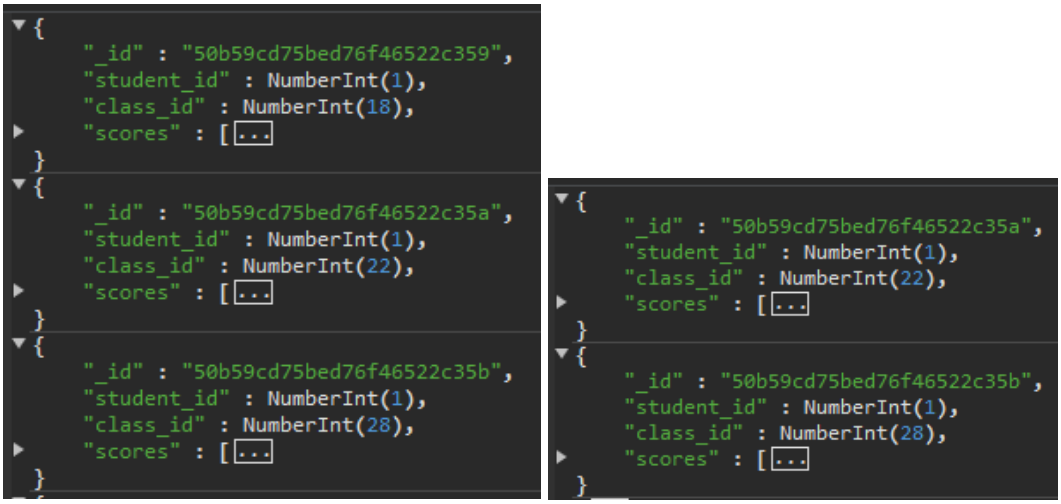
db.grades.deleteMany({ student_id: 0 })


```
{
    "acknowledged" : true,
    "deletedCount" : 1.0
}
```


4.4. Remover un solo documento

Elimina un solo documento en la colección grades donde el campo student_id es igual a 1 (similar a deleteOne).

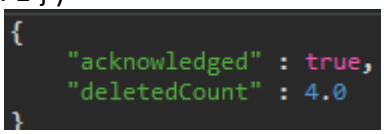
db.grades.remove({ student_id: 1 }, {justOne: true})



4.5. Remover todos los documentos

Elimina todos los documentos en la colección grades donde el campo student_id es igual a 1 (similar a deleteMany).

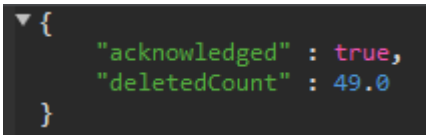
db.grades.remove({ student_id: 1 })



4.6. Vaciar la colección

Elimina todos los documentos en la colección grades sin condición, es decir, vacía la colección.

db.grades.remove({ })



4.7. Eliminar una colección

Elimina completamente la colección grades, incluyendo sus índices y metadatos.

db.grades.drop()

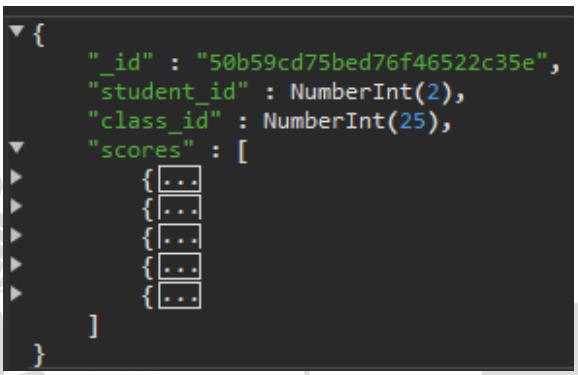
Consultas NoSQL

1. Operaciones de Lectura (READ)

1.1. Encontrar documentos por student_id

Encuentra todos los documentos en la colección grades donde el student_id es 2.

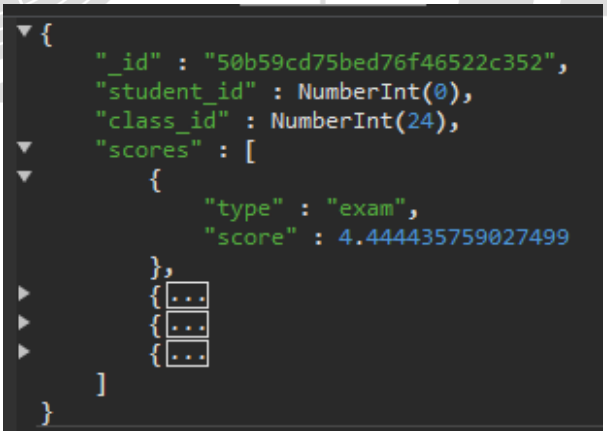
db.grades.find({ student_id: 2 })



1.2. Encontrar documentos con puntajes específicos

Encuentra documentos donde el primer score (scores.0) es menor o igual a 10.

db.grades.find({ "scores.0.score": { $lte: 10 } })



1.3. Encontrar documentos por el quinto score

Encuentra documentos donde el quinto score (scores.4) es menor o igual a 10.

db.grades.find({ "scores.4.score": { $lte: 10 } })

5. Análisis y Discusión

El análisis de los resultados muestra que MongoDB permite una gestión eficaz de datos semi-estructurados. Las operaciones CRUD proporcionan la flexibilidad necesaria para manipular y administrar datos en tiempo real, adaptándose a necesidades que varían rápidamente en volumen y estructura. Este taller demostró cómo MongoDB simplifica la creación y el mantenimiento de colecciones, y su capacidad para manejar relaciones entre datos, aunque no de manera tradicional como en una base de datos relacional.

INSTITUTO TECNOLOGICO DEL PUTUMAYO
El Saber como Arma de Vida

6. Conclusion

En conclusión, el taller demostró que MongoDB es una herramienta poderosa para el manejo de datos no estructurados y semi-estructurados. Las operaciones CRUD en MongoDB se ejecutan de manera eficiente, ofreciendo un modelo de datos flexible que facilita el desarrollo de aplicaciones dinámicas y con necesidades de datos complejas.

INSTITUTO TECNOLOGICO DEL PUTUMAYO
El Saber como Arma de Vida

7. Referencias

Apache Software Foundation. (2021). MongoDB Documentation. Retrieved from https://www.mongodb.com/docs/

Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.

Link repositorio bd_escuela: https://github.com/esteban2oo1/BD_escuela.git

INSTITUTO TECNOLOGICO DEL PUTUMAYO
El Saber como Arma de Vida

INSTITUTO TECNOLOGICO DEL PUTUMAYO
El Saber como Arma de Vida