## MongoDB бази данни – операции създаване, изтриване и обновяване

Заявките към една база данни могат да бъдат Create, Read, Update и Delete (CRUD). Целта на лекцията е да научим как програмно с Java код може да създаваме бази данни, колекции и документи, както и да обновяваме стойностите на полета и да изтриваме документи.

### *1. Създаване на бази данни*

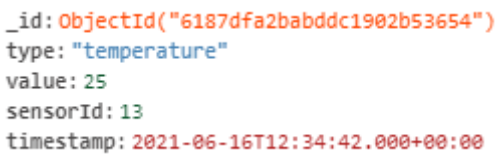
Създаването на една база данни програмно се реализира в следната последователност:

- Задаване на името на базата данни с която ще работим. За целта ще използваме метод **getDatabase**. Той изисква за атрибут името на базата данни. Методът връща обект от клас MongoDatabase. Ако базата с указаното име не съществува, методът я създава.
- Създаване на колекция. Ще използваме метод **getCollection**, който приема като аргумент името на колекцията. Ако колекцията не съществува тя ще бъде създадена. Методът getCollection връща обект, колекция от документи: MongoCollection<Document>.
- Вмъкване на необходимите документи в избрана колекция. Вмъкването може да се реализира един по един или множество наведнъж. При първият случай се използва метод **insertOne**, а при вторият – метод **insertMany**. Методите изискват като аргумент да се зададе обект-документ или колекция от обекти-документи. Документите може да бъде импортирани от файл или създадени програмно динамично.

Нека документите, които трябва да се запишат в колекцията, са в текстов файл под формата на JSON обект. Обектът има едно поле с име „data”. Стойността на полето е масив от документи, които съдържат информация от сензори за температура и влажност на въздуха:

```
{
      "data":
        [
            {
                "sensorId": 13,
                "timestamp": {
                    "$date": "2021-06-16T12:34:42Z"
                },
                "type": "temperature",
                "value": 25
            },
            {
                "sensorId": 18,
                "timestamp": {
                    "$date": "2021-06-10T05:55:57Z"
                },
                "type": "humidity",
                "value": 26
            }
        ]
}
```

Всеки документ има 4 полета. Поле “timestamp” приема за стойност дата и час в ISODate формат. За целта неговата стойност е JSON обект с поле със специфичното име “$date”. Това гарантира, че сървърът ще създаде поле от тип ISODate:




Програмният код, който прочита данните от файла, конвертира ги до документи, създава нова колекция и вмъква един по един всички документи, е следния:

```
try {
   // -------- Прочитане на съдържанието на файла с документите
   String content = new String(Files.readAllBytes(Paths.get(fpath)),
                                                  StandardCharsets.UTF_8);

   // ---- Конвертиране на прочетеното съдържание до JSON обект
   JSONObject obj = new JSONObject(content);

   // ------------------ Извличане на съдържанието на поле data
   JSONArray data = obj.getJSONArray("data");

   // ---------------------------------------- Connect to server
   MongoClient mongoClient = new MongoClient("localhost", 27017);

   //------------------------------------------ Create database
   MongoDatabase db = mongoClient.getDatabase("sensors");

   //------------------------------------------ Create collection
   MongoCollection<Document> collection = db.getCollection("data");

   // ------------------- Запис на документите в колекция data
   int n = data.length();
   for (int i = 0; i < n; i++) {
      String dataAsString = data.getJSONObject(i).toString();
      Document doc = Document.parse(dataAsString);
      collection.insertOne(doc);
   }
   // ---------------------------------------------------------------------
} catch (IOException | JSONException e) {
   System.out.println(e);
}
```

Ако е необходимо в колекцията да се вмъкнат всички документи едновременно трябва да използвате метод **insertMany**:

```
int n = data.length();
List<Document> docs = new ArrayList<>();
for (int i = 0; i < n; i++) {
    String dataAsString = data.getJSONObject(i).toString();
    Document doc = Document.parse(dataAsString);
    docs.add(doc);
}
collection.insertMany(docs);
```

### *2. Обновяване на полета*

Промяната на стойностите на полета от документ се реализира чрез метод **update**. Синтаксисът на метода е следния:

db.collection.**update**(query, update, options)

където:

- "query" задава критериите за избор на обновяване. Налични са същите селектори за заявка като при метода find;
- "update" - промените, които трябва да се приложат към документите, избрани чрез query.
- "options" – параметри чрез които се конкретизира обновяването. Например, при параметър "multi" със стойност "true" обновяването се отнася за всички документи, отговарящи на филтъра зададен чрез query. По подразбиране се обновява съдържанието на един документ.

Чрез метод **update** може да се модифицират определени полета на съществуващ документ или документи или да се замени изцяло съществуващ документ в зависимост от стройността на параметър "update". По подразбиране методът актуализира един документ. Чрез опция "multi": true, може да актуализирате всички документи, които отговарят на критериите на заявката.

В Табл. 1 са описани основните оператори, които могат да бъдат използвани при обновяване на полета:

Табл. 1 Оператори при обновяване

| Оператор | Описание |
|---|---|
| $currentDate | Задава стойността на полето като текуща дата, като дата или времеви печат. |
| $inc | Увеличава текущата стойността на полето с указана стойност, която може да е различна от 1 (както положителна, така и отрицателна). |
| $min | Актуализира полето само ако зададената стойност е по-малка от съществуващата стойност на полето. |
| $max | Актуализира полето само ако зададената стойност е по-голяма от съществуващата стойност на полето. |
| $mul | Умножава стойността на полето по посочената стойност. |
| $rename | Преименува поле. |
| $set | Задава стойността на поле в документ. |
| $setOnInsert | Задава стойността на поле, ако актуализацията води до вмъкване на документ. Няма ефект върху операции за актуализация, които променят съществуващи документи. |
| $unset | Премахва посоченото поле от документ. |

**Внимание**: Ако аргумент "update" съдържа само двойка от вида "fieldname:value", то съдържанието на документа се изтрива и в него се вмъква новото поле и неговата стойност. Ако искате само да обновите стойността на поле, използвайте оператор $set.

Когато се налага обновяване на полета от масив могат да бъдат използвани операторите, описани в Табл. 2:

Табл. 2 Оператори при обновяване на полета от масив

| Оператор | Описание |
|---|---|
| $ | Действа като заместител за актуализиране на първия елемент, който отговаря на условието на заявката. |
| $[] | Действа като заместител за актуализиране на всички елементи в масив за документите, които отговарят на условието на заявката. |
| $[<идентификатор>] | Действа като заместител, за да актуализира всички елементи, които отговарят на условието зададено чрез опция “arrayFilters”, за документите, които отговарят на условието в заявката. |
| $addToSet | Добавя елементи към масив само ако те вече не съществуват в множеството. |
| $pop | Премахва първия или последния елемент от масив. |
| $pull | Премахва всички елементи на масива, които отговарят на зададена заявка. |
| $pullAll | Премахва всички съвпадащи стойности от масив. |
| $push | Добавя елемент в масив. |

Към някои от описаните оператори могат да бъдат приложени модификатори които променят тяхното действие. Тези модификатори са описани в Табл.3.

Табл. 3 Модификатори към оператори за работа с масиви

| Модификатор | Описание |
|---|---|
| $each | Модифицира действието на оператори $push и $addToSet с цел добавяне на множество елементи при актуализиране на масива. |
| $position | Модифицира оператор $push - указва позицията в масива за добавяне на елементи. |
| $slice | Модифицира оператор $push - ограничава размера на актуализираните масиви. |
| $sort | Модифицира оператор $push - сортира документите, съхранявани в масив. |

Като трети аргумент към метод update могат да бъдат зададени множество опции. Те са описани в Табл. 4.

Табл.4 Опции към метод update

| Опция | Описание |
|---|---|
| upsert (true/false) | Създава нов документ, ако няма документи, които да отговарят на заявката или актуализира един документ, който отговаря на заявката. Ако и двете стойности upsert и multi са true и няма документи, които да отговарят на заявката, операцията update вмъква само един документ. По подразбиране стойността на upsert е false - не се вмъква нов документ, когато не е намерено съвпадение. |
| multi (true/false) | Ако е зададена стойност true, се актуализират няколко документа, които отговарят на критериите на заявката. Ако е зададена стойност false, се актуализира само един документ. Стойността по подразбиране е false. |

| | |
|---|---|
| writeConcern (document) | От MongoDB 4.4 наборите от реплики и клъстерите с шардове поддържат задаване на глобална грижа за запис по подразбиране. Операциите, които не задават изрично загриженост при запис, наследяват глобалните настройки за значение за запис по подразбиране.<br>Не задавайте изрично загриженост за запис за операции, които се изпълнява в транзакция. |
| collation (document) | Collation позволява на потребителите да задават специфични за езика правила за сравняване на низове. |
| arrayFilters (масив) | Масив от документи за филтриране, които определят кои елементи на масива да се модифицират при операция за актуализиране на поле от масив. |
| hint (document или string) | Документ или низ, който определя индекса, който да се използва за поддържане на предиката на заявката.<br>Опцията може да приеме документ за спецификация на индекса или низ от името на индекса. Ако укажете индекс, който не съществува, операцията ще бъде грешна |

**Задача**: Напишете Java конзолно приложение, което променя оценката по зададена дисциплина на студент, зададен чрез неговия факултетен номер. Работете с база данни “students”, колекция “data1”.

Нека да работим с документа, който описва студента Георги Котев:


```
{
    "_id": {
        "$oid": "60dddd8fd7aeaed049730d3e"
    },
    "id": "21705001",
    "name": {
        "firstName": "Георги",
        "lastName": "Котев"
    },
    "grade": [{
        "subject": "НБД",
        "value": 4
    }, {
        "subject": "КМС",
        "value": 6
    }, {
        "subject": "ММС",
        "value": 5
    }, {
        "subject": "ДСП",
        "value": 2
    }]
}
```


Ще променим оценката по дисциплината НБД, от Добър(4) на Мн. добър(5). Следователно, стойността на атрибут “query” трябва да е

{“id”:”21705001”,”grade.subject.НБД”},

Като стойност на атрибут “update” трябва да се укаже, че се модифицира само едно поле, а не целия документ. Следователно трябва да се използва оператор $set:

{“$set”: {grade.$.value: 5}}

Тъй като “grade” е масив, трябва да се използва оператора за позициониране $.

Следва програмния код, който реализира задачата:

```
final String DATABASE_NAME = "students";
final String COLLECTION_NAME = "data1";
final String STUDENT_ID = "21705001";
final String SUBJECT_NAME = "НБД";
final int GRADE = 5;

try {
   // ------------------------------------------------ Connect to server
   MongoClient mongoClient = new MongoClient("localhost", 27017);

   //-------------------------------------------------- Create database
   MongoDatabase db = mongoClient.getDatabase(DATABASE_NAME);

   //-------------------------------------------------- Create collection
   MongoCollection<Document> collection
                    = db.getCollection(COLLECTION_NAME);

   // ---------------------------------------------------------- Update field
   Bson query = and(eq("id", STUDENT_ID),eq("grade.subject", SUBJECT_NAME));
   Bson update = set("grade.$.value", GRADE);
   collection.updateOne(query, update);

   System.out.println("Document update successfully...");
   // ------------------------------------------------------------------------

} catch (JSONException e) {
   System.out.println(e);
}
```

**Задача**: Напишете Java конзолно приложение, което променя оценката по зададена дисциплина на всички студенти. Работете с база данни “students”, колекция “data1”.

Трябва да променим стойността на поле “query” и да използваме метод **updateMany**:

```
Bson query = eq("grade.subject", SUBJECT_NAME);
Bson update = set("grade.$.value", GRADE);
collection.updateMany(query, update);
```

### *3. Изтриване на документ(и)*

Може да изтривате един или няколко документа от избрана колекция. За целта използвайте метод **deleteOne** или **deleteMany**. Операциите за изтриване не премахват зададените от вас индекси, дори ако се изтрият всички документи от дадена колекция. Когато се налага да изтриете само един документ, който отговаря на дадено условие (независимо, че може да има и други документи, които отговарят на да отговарят на зададения филтър) може да използвате метод **deleteOne**. Синтаксисът на методите за изтриване на документи е следния:

db.collection.delete(query, options);

Чрез атрибут “query” се задава филтъра (критерии) на базата на който ще бъдат избрани кои документи да бъдат изтрити. Може да зададете празен документ { }, за да изтриете първия документ, върнат в колекцията. Вторият аргумент задава опции “writeConcern”, “collation” и “hint”. Те имат аналогично приложение, както едноименните опции при метод

update. Следва пример, който изтрива първия срещнат документ, за който стойността на поле “status” не е “ok”:

collection.**deleteOne**(ne("status", "ok"))

Ако трябва да изтриете всички документи, които отговарят на дадено условие, използвайте метод **deleteMany**. Можете да зададете критерии или филтри, които определят документите, които да бъдат изтрити. Филтрите използват същия синтаксис като при четене (метод find). Следващият пример премахва всички документи от колекция за които поле “toDelete” има стойност "yes":

collection.**deleteMany**(eq("toDelete", "yes"))

За да изтриете всички документи от дадена колекция, подайте празен обект като филтър на метода deleteMany:

collection.**deleteMany**(new Document());

Методът deleteMany връща обект от тип DeleteResult, който съдържа информация за статуса на операцията.

**Задача**: Напишете Java конзолно приложение, изтрива от база данни “my-sensors” всички документи, който са генерирани след зададена дата.

Използвайте метод **deleteMany**. Сравняването на дати изисква описание на датата като ISOData обект. Класът Instant в Java 8+ (java.time.Instant) за дата и време позволява създаване на обект, който описва конкретен момент във времето. Метод **parse** от този клас се използва за конвертиране на низ, съдържащ дата и час до обект от тип Instant. За да формираме заявка към MongoDB сървъра е необходимо този обект от тип Instant да преобразуваме до обект от тип Date. Това се реализира чрез статичния метод **from** от клас Date. Програмният код, който реализира условието на задачата е следния:

```
final String DATABASE_NAME = "my-sensors";
final String COLLECTION_NAME = "data";

try {
    // ---------------------------------------- Connect to server
    MongoClient mongoClient = new MongoClient("localhost", 27017);
    //------------------------------------------- Create database
    MongoDatabase db = mongoClient.getDatabase(DATABASE_NAME);
    //------------------------------------------- Create collection
    MongoCollection<Document> collection
                    = db.getCollection(COLLECTION_NAME);
    // ------------------------------------------------ Delete docs
    Instant instant = Instant.parse("2021-06-20T00:00:00.000Z");
    Date timestamp = Date.from(instant);
    Bson query = gt("timestamp", timestamp);
    collection.deleteMany(query);
    // ---------------------------------------------------------------

} catch (JSONException e) {
    System.out.println(e);
}
```