MongoDB

Андрей Ерошкин

21 ноября 2021 г.

Overview

Устройство базы данных. Документы

В отличие от реляционных баз данных MongoDB предлагает документо-ориентированную модель данных, благодаря чему MongoDB работает быстрее, обладает лучшей масштабируемостью, ее легче использовать.
Для хранения в MongoDB применяется формат, который называется BSON или сокращение от binary JSON. Способ хранения данных в MongoDB в похож на JSON.

Устройство базы данных. Документы

Если реляционные базы данных хранят строки, то MongoDB хранит документы. В отличие от строк документы могут хранить сложную по структуре информацию. Документ можно представить как хранилище ключей и значений.
В MongoDB для каждого документа имеется уникальный идентификатор, который называется _id. И если явным образом не указать его значение, то MongoDB автоматически сгенерирует для него значение.

Устройство базы данных. Документы

Каждому ключу сопоставляется определенное значение. Но здесь также надо учитывать одну особенность: если в реляционных базах есть четко очерченная структура, где есть поля, и если какое-то поле не имеет значение, ему можно присвоить значение NULL. В MongoDB все иначе. Если какому-то ключу не сопоставлено значение, то этот ключ просто опускается в документе и не употребляется.
Если в традиционном мире SQL есть таблицы, то в мире MongoDB есть коллекции. И если в реляционных БД таблицы хранят однотипные жестко структурированные объекты, то в коллекции могут содержать самые разные объекты, имеющие различную структуру и различный набор свойств.

Устройство базы данных. Документы

Всего имеется следующие типы значений:

- String: строковый тип данных, как в приведенном выше примере (для строк используется кодировка UTF-8)
- Array (массив): тип данных для хранения массивов элементов
- Binary data (двоичные данные): тип для хранения данных в бинарном формате
- Boolean: булевый тип данных, хранящий логические значения TRUE или FALSE, например, "married": FALSE
- Date: хранит дату в формате времени Unix
- Double: числовой тип данных для хранения чисел с плавающей точкой
- Integer: используется для хранения целочисленных значений размером 32 бита, например, "age": 29

Устройство базы данных. Документы

- Long: используется для хранения целочисленных значений размером 64 бита
- JavaScript: тип данных для хранения кода javascript
- Min key/Max key: используются для сравнения значений с наименьшим/наибольшим элементов BSON
- Null: тип данных для хранения значения Null
- Object: строковый тип данных, как в приведенном выше примере
- ObjectId: тип данных для хранения id документа
- Regular expression: применяется для хранения регулярных выражений
- Decimal128: тип данных для хранения десятичных дробных чисел размером 128 бит, которые позволяют решить проблемы с проблемой точности вычислений при использовании дробных чисел, которые представляют тип Double.

Добавление данных

Для добавления в коллекцию могут использоваться три ее метода:
insertOne(): добавляет один документ
insertMany(): добавляет несколько документов
insert(): может добавлять как один, так и несколько документов

Выборка из БД

Чтобы извлечь все документы из коллекции:

```
db.users.find()
```

Функция findOne() работает похожим образом, только возвращает один документ:

```
db.users.findOne()
```

Фильтрация данных

Выведем все документы, в которых name=Tom:

```
db.users.find({name: "Tom"})
```

Усложним запрос и получим те документы, у которых в массиве languages одновременно два языка: "english"и "german":

```
db.users.find({languages: ["english", "german"]})
```

И чтобы найти все документы, у которых в ключе company вложенное свойство name=microsoft, нам надо использовать оператор точку:

```
db.users.find({"company.name": "Microsoft"})
```

MongoDB предоставляет замечательную возможность, создавать запросы, используя язык JavaScript. Например, создадим запрос, возвращающий те документы, в которых name=Tom. Для этого сначала объявляется функция:

```
fn = function() { return this.name=="Tom"; }
db.users.find(fn)
```

Выборка из БД

Функция limit. Она задает максимально допустимое количество получаемых документов.

```
db.users.find().limit(3)
```

Пропустим первые три записи:

```
db.users.find().skip(3)
```

MongoDB предоствляет возможности отсортировать полученный из бд набор данных с помощью функции sort. Передавая в эту функцию значения 1 или -1, мы можем указать в каком порядке сортировать: по возрастанию (1) или по убыванию (-1).

```
db.users.find().sort({name: 1})
```

Команды группировки

С помощью функции count() можно получить число элементов в коллекции

```
db.users.find({name: "Tom"}).count()
```

Только уникальные различающиеся значения для одного из полей документа:

```
db.users.distinct("name")
["Tom", "Bill", "Bob"]
```

Операторы выборки

Условные операторы задают условие, которому должно соответствовать значение поля документа:

- $eq (равно)
- $ne (не равно)
- $gt (больше чем)
- $lt (меньше чем)
- $gte (больше или равно)
- $lte (меньше или равно)
- $in определяет массив значений, одно из которых должно иметь поле документа
- $nin определяет массив значений, которые не должно иметь поле документа

```
db.users.find ({age: {$lt : 30}})
```

Операторы выборки

Логические операторы выполняются над условиями выборки:

- $or: соединяет два условия, и документ должен соответствовать одному из этих условий
- $and: соединяет два условия, и документ должен соответствовать обоим условиям
- $not: документ должен НЕ соответствовать условию
- $nor: соединяет два условия, и документ должен НЕ соответстовать обоим условиям

```
db.users.find (
    {$or : [{name: "Tom"}, {age: 22}]}
)
```

Обновление данных

Если нам надо полностью заменить один документ другим, также может использоваться функция replaceOne:

```
db.users.replaceOne(
    {name: "Bob"}, {name: "Bob", age: 25}
)
```

Часто не требуется обновлять весь документ, а только значение одного или нескольких его свойств. Для этого применяются функции updateOne() - она обновляет только один документ и updateMany() - позволяет обновить множество документов.

```
db.users.updateOne(
    {name : "Tom", age: 25},
    {$set: {age : 28}}
)

db.users.updateOne(
    {name : "Tom"}, {$unset: {salary: 1}}
)

db.users.updateOne(
    {name : "Tom"}, {$push: {languages: "russian"}}
)
```

Обновление данных

Для удаления документов в MongoDB предусмотрены функции deleteOne() - удаляет один документ и deleteMany() - позволяет удалить несколько документов.

```
db.users.deleteOne({name : "Tom"})
```

Для удаления всех документов, которые соответствуют фильтруб применяется функция deleteMany():

```
db.users.deleteMany({name : "Tom"})
```

Агрегация

Агрегация – это группировка значений многих документов. Операции агрегирования позволяют манипулировать такими сгруппированными данными

```
db.developers.aggregate(
    [
        {$group :
        {
            _id : "Developers",
            total_salary: { $sum : "$salary" }
        }}
    ]
)
{ "_id" : "Developers", "total_salary" : 10500 }
```

Работа с данными в MongoDB Compass

Для работы с MongoDB также можно использовать официальный графический клиент MongoDB Compass.

ODM

Mongoose представляет специальную ODM-библиотеку (Object Data Modelling) для работы с MongoDB, которая позволяет сопоставлять объекты классов и документы коллекций из базы данных.