MongoDB. Часть 2.

Команды группировки

Отдельно стоит рассмотреть команды группировки: `count, group, distinct`.

Число элементов в коллекции

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

```
1> db.users.count()
```

Можно группировать параметры поиска и функцию count, чтобы подсчитать, сколько определенных документов, например, у которых name=Tom:

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

Более того мы можем создавать цепочки функций, чтобы конкретизировать условия подсчета:

```
1> db.users.find({name: "Tom"}).skip(2).count(true)
```

Здесь надо отметить, что по умолчанию функция count не используется с функциями limit и skip. Чтобы их использовать, как в примере выше, в функцию count надо передать булевое значение `true`

Функция distinct

В коллекции могут быть документы, которые содержат одинаковые значения для одного или нескольких полей. Например, в нескольких документах определено `name: "Tom"`. И нам надо найти только уникальные различающиеся значения для одного из полей документа. Для этого мы можем воспользоваться функцией distinct:

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

Здесь я запрашиваю только уникальные значения для поля `name`. И на следующей строке консоль выводит в виде массива найденные уникальные значения.

Группировка

Агрегация – это группировка значений многих документов. Операции агрегирования позволяют манипулировать такими сгруппированными данными (например, подсчёт – COUNT(*)). В MySQL аналогом агрегации является команда group by.

В MongoDB для агрегации используется метод aggregate(). Данный метод имеет следующий синтаксис:

db.ИМЯ_КОЛЛЕКЦИИ.aggregate(ОПЕРАЦИЯ_АГРЕГАЦИИ)

Метод aggregate() может принимать следующие параметры:

$project

Используется для выбора некоторых специальных полей из коллекции.

$match

Операция фильтрации, которая может уменьшить количество документов, которые передаются для ввода в следующий этап.

$group

Непосредственно, сама агрегация

$sort

Сортирует документы

$skip

С помощью данной команды мы можем проигнорировать список документов в имеющемся множестве.

$limit

Ограничивает количество документов для вывода на количество, переданное методу, начиная, с текущей позиции.

$unwind

Используется для обработки документов, который используют массивы.

Список операций для агрегации документов:

Выраженеие	Описание
$sum	Суммирует указанные значения всех документов в коллекции.
$avg	Рассчитывает среднее значение указанного поля поля для всех документов коллекции.
$min	Получает минимальное значение указанного поля документа в коллекции

$max	Получает максимальное значение указанного поля документа в коллекции
$push	Вставляет значение в массив в результирующем документе.
$addToSet	Вставляет значение в массив в результирующем документе, но не создаёт дубликаты.
$first	Получает первый документ из сгруппированных. Обычно используется вместе с сортировкой.
$last	Получает крайний документ из сгруппированных. Обычно используется вместе с сортировкой.

Чтобы вывести количество различных имен пользователей в нашей коллекции необходимо выполнить:

```
db.users.aggregate([{$group:{_id:"$name",summa:{$sum:1}}}])
```

Вывести средний возраст пользователей со знанием английского языка:

```
db.users.aggregate([{$match:{languages:"english"}},
{$group:{_id:"users",avg:{$avg:"$age"}}}])
```

Операторы выборки

Условные операторы

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

Например, найдем все документы, у которых значение ключа age меньше 30:

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

Аналогично будет использование других операторов сравнения. Например, тот же ключ, только больше 30:

```
1> db.users.find ({age: {$gt : 30}})
```

Обратите внимание, что сравнение здесь проводится над целочисленными типами, а не строками. Если ключ age представляет строковые значения, то соответственно надо проводить сравнение над строками: `db.users.find ({age: {$gt : "30"}})`, однако результат будет тем же.

Но представим ситуацию, когда нам надо найти все объкты со значением поля age больше 30, но меньше 50. В этом случае мы можем комбинировать два оператора:

```
1> db.users.find ({age: {$gt : 30, $lt: 50}})
```

Найдем пользователей, возраст которых равен 22:

```
1> db.users.find ({age: {$eq : 22}})
```

По сути это аналогия следующего запроса:

```
1> db.users.find ({age: 22})
```

Обратная операция - найдем пользователей, возраст которых НЕ равен 22:

```
1> db.users.find ({age: {$ne : 22}})
```

Оператор $in определяет массив возможных выражений и ищет те ключи, значение которых имеется в массиве:

```
1> db.users.find ({age: {$in : [22, 32]}})
```

Противоположным образом действует оператор $nin - он определяет массив возможных выражений и ищет те ключи, значение которых отсутствует в этом массиве:

```
1> db.users.find ({age: {$nin : [22, 32]}})
```

Логические операторы

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

Оператор $or

Оператор $or представляет логическую операцию ИЛИ и определяет набор пар ключ-значение, которые должны иметься в документе. И если документ имеет хоть одну такую пару ключ-значение, то он соответствует данному запросу и извлекается из бд:

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

Это выражение вернет нам все документы, в которых либо name=Tom, либо age=22.

Другой пример вернет нам все документы, в которых name=Tom, а age равно либо 22, либо среди значений languages есть "german":

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

В подвыраженях or можно применять условные операторы:

```
1> db.users.find ({$or : [{name: "Tom"}, {age: {$gte:30}}]})
```

В данном случае мы выбираем все документы, где name="Tom" или поле age имеет значение от 30 и выше.

Оператор $and

Оператор $and представляет логическую операцию И (логическое умножение) и определяет набор критериев, которым обязателньо должен соответствовать документ. В отличие от оператора $or документ должен соответствовать всем указанным критериям. Например:

```
1> db.users.find ({$and : [{name: "Tom"}, {age: 32}]})
```

Здесь выбираемые документы обязательно должны имееть имя Tom и возраст 32 - оба этих признака.

Поиск по массивам

Ряд операторов предназначены для работы с массивами:

- $all: определяет набор значений, которые должны иметься в массиве
- $size: определяет количество элементов, которые должны быть в массиве
- $elemMatch: определяет условие, которым должны соответствовать элемены в массиве

$all

Оператор $all определяет массив возможных выражений и требует, чтобы документы имели весь определяемый набор выражений. Соответственно он применяется для поиску по массиву. Например, в документах есть массив languages, хранящий иностранные языки, на которых говорит пользователь. И чтобы найти всех людей, говорящих одновременно и по-английски, и по-французски, мы можем использовать следующее выражение:

```
1> db.users.find ({languages: {$all : ["english", "french"]}})
```

Оператор $elemMatch

Оператор $elemMatch позволяет выбрать документы, в которых массивы содержат элементы, попадающие под определенные условия. Например, пусть в базе данных будет коллекция, которая содержит оценки пользователей по определенным курсам. Добавим несколько документов:

```
 > db.grades.insertMany([{student: "Tom", courses:[{name: "Java",
1grade: 5}, {name: "MongoDB", grade: 4}]},
2{student: "Alice", courses:[{name: "C++", grade: 3}, {name:
 "MongoDB", grade: 5}]}])
```

Каждый документ имеет массив courses, который в свою очередь состоит из вложенных документов.

Теперь найдем студентов, которые для курса MongoDB имеют оценку выше 3:

```
1> db.grades.find({courses: {$elemMatch: {name: "MongoDB", grade: {$gt: 3}}}})
```

Оператор $size

Оператор $size используется для нахождения документов, в которых массивы имеют число элементов, равным значению $size. Например, извлечем все документы, в которых в массиве laguages два элемента:

```
1> db.users.find ({languages: {$size:2}})
```

Такой запрос будет соответствовать, например, следующему документу:

```
1{"name": "Tom", "age": 32, languages: ["english", "german"]}
```

Оператор $exists

Оператор $exists позволяет извлечь только те документы, в которых определенный ключ присутствует или отсутствует. Например, вернем все документы, в который есть ключ `company`:

```
1> db.users.find ({company: {$exists:true}})
```

Если мы укажем у оператора $exists в качестве параметра `false`, то запрос вернет нам только те документы, в которых не определен ключ company.

Оператор $type

Оператор $type извлекает только те документы, в которых определенный ключ имеет значение определенного типа, например, строку или число:

```
1> db.users.find ({age: {$type:"string"}})
2> db.users.find ({age: {$type:"number"}})
```

Оператор $regex

Оператор $regex задает регулярное выражение, которому должно соответствовать значение поля. Например, пусть поле name обязательно имеет букву "b":

```
1> db.users.find ({name: {$regex:"b"}})
```

Важно понимать, что $regex принимает не просто строки, а именно регулярные выражения, например: `name: {$regex:"om$"}` - значение name должно оканчиваться на "om".

Задание для самоподготовки

1. Вывести общее количество фотоаппаратов в магазине
2. Вывести количество зеркальных фотоаппаратов
3. Вывести наименования всех фирм производителей
4. Вывести наименование фирмы производителя и общее количество фотоаппаратов, представленных и интернет-магазине, для каждой фирмы

5. Вывести первые 10 наименований фотоаппаратов, отсортированных по возрастанию цены, стоимость которых не превосходит 100 000 рублей
6. Вывести все фотоаппараты с полнокадровой матрицей в ценовом диапазоне от 100 до 200 тыс. руб.
7. Вывести все фотоаппараты с полнокадровой матрицей компании Nikon, либо другой фирмы, но цена которых - до 150 000 р.
8. Вывести все фотоаппараты компании Nikon, либо фотоаппараты с объективом, фокусное расстояние которого равно «24-85».
9. Вывести все фотоаппараты компаний Nikon и Canon, которые продаются без объектива (body)
10. Вывести список фотоаппаратов, которые продаются с объективом «Nikkor» и светосилой «1.4» и «1.2»
11. Вывести список фотоаппаратов, для которых в описании имеется графа «Страна производитель»
12. Вывести список фотоаппаратов от компании Canon, название которых начинается на слово «MARK»
13. Вывести список фотоаппаратов от компании Nikon, название которых заканчивается на «00»