ЛР1. MongoDB. Знакомство с MongoDB

MongoDB – это кроссплатформенная документоориентированная база данных NoSQL с открытым исходным кодом. Она не требует описания схемы таблиц, как в реляционных БД. Данные хранятся в виде коллекций и документов, документы хранятся в бинарном JSON-подобном формате. Полная документация доступна по ссылке: https://www.mongodb.com/docs/v6.0/.

1. Организация данных

Рассмотрим основные понятия, используемые в MongoDB. *Документ* – это упорядоченный набор ключей со связанными значениям. Документ представляет собой основную единицу данных в MongoDB и приблизительно эквивалентен строке в реляционной системе управления базами данных (СУБД). Документы имеют динамическую схему, т.е. документы в одной коллекции не обязательно должны иметь одинаковый набор полей или структуру, а общие поля в документах коллекции могут содержать данные разных типов. У каждого документа есть специальный ключ "`_id`", который является уникальным в рамках коллекции. *Коллекция* представляет собой группу документов MongoDB. В терминологии реляционных СУБД коллекция соответствует таблице. *База данных* – это физический контейнер для коллекций.

2. Вставка данных

Вставка – основной метод добавления данных в MongoDB. Чтобы вставить один документ, используется метод коллекции `insertOne`:

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

Метод добавит в документ ключ "`_id`" (если его не указали явно) и сохранит документ в MongoDB. Если коллекция не существует, операции вставки создадут коллекцию. WriteCOncern – опциональный параметр, позволяющий узнать, успешно ли завершилась операция вставки, в т.ч. подтверждать запись в несколько реплик.

Если нужно вставить несколько документов в коллекцию, можно использовать метод `insertMany`:

```
db.collection.insertMany(
   [ <document 1> , <document 2>, ... ],
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)
```

Этот метод позволяет передавать массив документов в базу данных, что эффективнее добавление документов по одному. Опциональный параметр `ordered` позволяет указывать, использовать упорядоченное или неупорядоченное добавление. Если указано значение `true` (значение по умолчанию), то документы будут добавляться в указанном порядке. Также, если добавление массива документов завершится с ошибкой вставки на каком-то документе, ни один документ за пределами этой точки в массиве не будет вставлен. В случае с неупорядоченными вставками MongoDB попытается вставить все документы независимо от того, приводят ли некоторые вставки к ошибкам.

Пример

Создание нескольких документов с помощью `insertMany()`*:*

```
db.products.insertMany( [
    { item: "card", qty: 15 },
    { item: "envelope", qty: 20 },
    { item: "stamps" , qty: 30 }
]);
```

2. Выборка данных

Основной метод, используемый для выборки данных из коллекции – это метод `find`:

```
db.collection.find(query, projection)
```

Метод принимает два опциональных параметра: `query`, который определяет условия фильтрации документов с использованием операторов запроса, и `projection`, определяющий поля, которые должны возвращаться в документах, соответствующих фильтру запроса. Если параметр `projection` не указан, то метод вернет все поля документов, советующих запросу. Метод возвращает курсор (или указатель) на выбранные документы. Для фильтрации документом используются

условия следующего вида: либо поле явно сравнивается со значением, используя синтаксическую конструкцию `<field1>: <value1>`, либо дополнительно используются операторы сравнения через конструкцию `<field2>: { <operator>: <value> }`. Один документ, описывающий запрос фильтрации, может содержать несколько условий. Ниже представлены основные операторы.

Оператор	**Описание**
$eq	Равно (=)
$ne	Не равно (!=)
$gt	Больше (>)
$gte	Больше или равно (>=)
$lt	Меньше (<)
$lte	Меньше или равно (<=)
$in	Соответствие хотя бы одному элементу массива
$nin	Несоответствие ни одному элементу массива
$and	Логическое И
$or	Логическое ИЛИ
$not	Логическое отрицание
$nor	Логическое НЕ-ИЛИ. Возвращает все документы, которые не соответствуют обоим предложениям
$exists	Возвращает документы, которые содержат указанное поле
$type	Возвращает документы, если поле имеет указанный тип.
$regex	Возвращает документы, значения полей которых соответствуют указанному регулярному выражению
$all	Возвращает документы, массивы которых содержат все элементы, указанные в запросе
$elemMatch	Возвращает документы, если элемент поля-массива соответствует всем указанным условиям
$size	Возвращает документы, если поле-массив имеет указанный размер

Метод `find` возвращает курсор указатель на набор документов, которые удовлетворяют условию поиска. Для результирующего курсора можно вызвать методы для сортировки, ограничения числа возвращаемых записей и т.д. Часть основных методов представлена ниже.

Оператор	Описание
cursor.count()	Изменяет курсор, чтобы он возвращал количество документов в результирующем наборе, а не сами документы
cursor.limit()	Ограничивает размер результирующего набора
cursor.map()	Применяет функцию к каждому документу в курсоре и возвращает полученные значения в виде массива
cursor.forEach()	Применяет JavaScript-функцию к каждому документу в курсоре
cursor.skip()	Возвращает курсор, который начинает возвращать результаты только после пропуска набора документов
cursor.sort()	Возвращает документы, упорядоченные в соответствии со спецификацией сортировки

Пример

Выборка документов по составному условию:

```
db.inventory.find( {
    status: "A",
    $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
}
```

Аналог запроса в SQL:

```
SELECT item,qty
FROM inventory
WHERE status = 'A' AND (qty < 30 OR item LIKE 'p%')
```

3. Обновление данных

Для обновления документов чаще всего используются методы `updateOne` для обновления значений полей одного документа, `updateMany` для обновления нескольких документов и `replaceOne` для замены всего документа:

`db.collection.updateOne(`
`  <filter>, <update>, <options>`
`)`
	Обновление одного документа

`db.collection.updateMany(`
`  <filter>, <update>, <options>`
`)`
	Обновление нескольких документов

`db.collection.replaceOne(`
`  <filter>, <update>, <options>`
`)`
	Замена документа

`Filter` – условие, по которому будет выбран документ (документы) для обновления. Могут применяться те же условия, что и в методе `find`. Для метода `updateOne`, если по указанному условию будет найдено несколько документов – обновится первый. `Update` – документ с описанием модификаций. В базовом случае описывает операторы обновления и параметры. Также можно указать набор опций. Например, если указано `upsert = true`, то функция `updateOne` создаст новый документ, если нет документов, соответствующих фильтру. По умолчанию этот параметр равен `false`.

Основные операторы обновления представлены ниже:

Оператор	**Описание**
$currentDate	Устанавливает значение поля на текущую дату
$inc	Увеличивает значение поля на указанную величину
$min	Обновляет поле только в том случае, если указанное значение меньше существующего значения поля
$max	Обновляет поле только в том случае, если указанное значение больше, чем существующее значение поля
$mul	Умножает значение поля на указанную величину
$rename	Переименовывает поле
$set	Устанавливает значение поля в документе
$setOnInsert	Задает значение поля, если обновление приводит к вставке документа
$unset	Удаляет указанное поле из документа
$	Модификатор, указывающий, что необходимо обновить первый элемент, соответствующий условию запроса
$[]	Модификатор, указывающий, что необходимо обновить все элементы, соответствующие условию запроса
$pop	Удаляет первый или последний элемент массива
$pull	Удаляет все элементы массива, соответствующие указанному запросу
$push	Добавляет элемент в массив
$pullAll	Удаляет все элементы массива, соответствующие указанным значениям

Пример

Обновление одного документа указанного статуса:

```
db.inventory.updateOne(
    { status: "D" },
    {
        $inc: { qty : -10 },
        $set: { "size.uom": "cm", item: "paper_updated" },
        $currentDate: { lastModified: true }
    }
)
```

4. Удаление данных

Для удаления документов из коллекции используются методы `deleteOne` и `deleteMany` для удаления одного или нескольких документов соответственно:

`db.collection.deleteOne(`
`  <filter>, <options>`
`)`
	Удаление одного документа

`db.collection.deleteMany(`
`  <filter>, <options>`
`)`
	Удаление нескольких документов

Пример

Удаление всех документов указанного статуса:

```
db.inventory.deleteMany (
    { status: "A" },
)
```

5. Агрегация данных

Операции агрегации данных обрабатывают набор документов и возвращают результаты обработки. Агрегация данных может быть использована для:

- группировки значений из нескольких документов;
- выполнения операций над сгруппированными данными для получения единственного результата;
- анализа изменения данных в течение определенного промежутка времени.

Для агрегации данных используются:

- *конвейеры агрегации*, которые являются предпочтительным методом агрегирования данных;

- *методы агрегации*, которые просты, но лишены возможностей конвейера агрегации.

Ниже представлены основные методы агрегации:

Метод	**Описание**
`db.collection .estimatedDocumentCount()`	Возвращает приблизительное количество документов в коллекции или представлении
`db.collection.count()`	Возвращает количество документов в коллекции или представлении
`db.collection.distinct()`	Возвращает массив документов, которые имеют различные значения для указанного поля

Конвейер агрегации состоит из одного или нескольких этапов обработки документов:

- каждый этап выполняет операцию над входными документами. Например, на этапе может происходить фильтрация документов, группировка документов и вычисление некоторых значений;
- документы, полученные как результат работы этапа, передаются на следующий этап;
- конвейер агрегации может возвращать результаты для групп документов, например, вернуть среднее, максимальное и минимальное значения;
- начиная с MongoDB 4.2, конвейер агрегации может использоваться для обновления документов.

Для агрегации данных из коллекции collection используется метод `aggregate`:

`db.collection.aggregate( pipeline, options)`
 	
`db.collection.aggregate( [ { <stage> }, ... ] )`
	Агрегация данных из коллекции collection
	• pipeline – последовательность операций или этапов агрегации данных;
	• options – дополнительные параметры.

Ниже представлены основные операторы агрегации данных.

Оператор	Описание
$count	Возвращает количество документов на данном этапе конвейера агрегации.
$group	Группирует входные документы по указанному выражению-идентификатору и применяет выражения-аккумуляторы, если они указаны, к каждой группе. Выводит по одному документу для каждой отдельной группы. Выходные документы содержат только поле-идентификатор и, если указано, поля-аккумуляторы.
$limit	Передает первые *n* документов без изменений в конвейер, где n – указанное ограничение. Для каждого входного документа выводит либо один документ (для первых n документов), либо ноль документов (после первых n документов).
$lookup	Выполняет левое внешнее соединение с другой коллекцией в той же базе данных, чтобы фильтровать документы из «объединенной» коллекции для обработки.
$match	Фильтрует поток документов, позволяя только документам, удовлетворяющим условию, передаваться без изменений на следующий этап конвейера. Использует стандартные запросы MongoDB.
$merge	Записывает результирующие документы конвейера агрегации в коллекцию. Должен быть последним этапом конвейера.
$project	Изменяет структуру каждого документа в потоке, например, добавляя новые поля или удаляя существующие поля. Для каждого входного документа выводит один документ.
$set	Добавляет новые поля в документы. Как и $project, $set изменяет структуру каждого документа; в частности, путем добавления новых полей в выходные документы, которые содержат как существующие поля из входных документов, так и вновь добавленные поля.
$skip	Пропускает первые *n* документов, где *n* – указанный номер пропуска, и передает остальные документы без изменений в конвейер.
$sort	Переупорядочивает поток документов по указанному ключу сортировки. Меняется только порядок, документы остаются без изменений.
$unset	Удаляет/исключает поля из документов.
$unwind	Преобразует поле массива из входных документов для вывода документа для каждого элемента. Каждый выходной документ заменяет массив значением элемента. Для каждого входного документа выводит *n* документов, где *n* – количество элементов массива, которое может быть равно нулю для пустого массива.

Аккумуляторы – еще один тип выражений, использующихся для вычисления значения из значений полей, находящихся в нескольких документах. Аккумуляторы, которые предоставляет фреймворк агрегации, позволяют выполнять такие операции, как суммирование всех значений в определенном поле, вычисление среднего значения и т. д. Основные аккумуляторы данных представлены ниже.

Оператор	Описание
$accumulator	Возвращает результат пользовательской функции-аккумулятора.
$avg	Возвращает среднее числовых значений. Игнорирует нечисловые значения.
$first	Возвращает значение из первого документа для каждой группы. Порядок определяется только в том случае, если документы отсортированы.
$last	Возвращает значение из последнего документа для каждой группы. Порядок определяется только в том случае, если документы отсортированы.
$max	Возвращает наибольшее значение выражения для каждой группы.
$min	Возвращает наименьшее значение выражения для каждой группы.
$sum	Возвращает сумму числовых значений. Игнорирует нечисловые значения.

Пример

Выборка штатов с населением больше 10 миллионов:

```
db.zips.aggregate( [

    { $group: { _id: "$state", totalPop: { $sum: "$pop" } } },

    { $match: { totalPop: { $gte: 10 * 1000 * 1000 } } }

] )
```

ЛР1. Вопросы для контроля

1. NoSQL базы данных: недостатки реляционных решений, типы NoSQL баз данных.
2. Теорема CAP. BASE-архитектура.
3. MongoDB: основные понятия, возможности, недостатки.
4. MongoDB: агрегация данных.

ЛР1. Задание

Дистрибутив сервера и клиента MongoDB доступен по следующей ссылке: https://www.mongodb.com/try/download/community . Для выполнения лабораторной работы необходимо скачать и импортировать коллекцию `films` из архива https://github.com/itsecd/dbms-security/blob/main/labs/films.json .

1. Вывести всю информацию о фильмах с рейтингом `'NC-17'`.
2. Вывести количество фильмов категории `'Sports'`.
3. Используя метод добавления документов, изменить коллекцию так, чтобы результат запроса №2 изменился.
4. Вывести только названия фильмов категорий `'Games'` и `'Music'`.
5. Для каждого рейтинга фильмов подсчитать количество фильмов с данным рейтингом. Вывести данную информацию только для рейтингов, для которых количество фильмов больше 200. Отсортировать по возрастанию количества фильмов.
6. Вывести максимальное количество задействованных актеров в фильме и количество фильмов, имеющих такое число актеров.
7. Найти среднее число актеров, задействованных в фильмах категории `'Games'`.
8. Используя метод обновления документа(ов), изменить коллекцию так, чтобы результат запроса №7 изменился.