ЛАБОРАТОРНАЯ РАБОТА 7 Управление данными(Базы данных)

ВВЕДЕНИЕ В СУБД MONGODB. УСТАНОВКА MONGODB. НАЧАЛО РАБОТЫ С БД

Цель: овладеть практическими навыками установки СУБД MongoDB.

Программное обеспечение: СУБД MongoDB не ниже 5.0.8.

ВВЕДЕНИЕ

Как документ-ориентированная СУБД, Mongo — это обобщённое NoSQL решение. Её можно рассматривать, как альтернативу реляционным СУБД. Подобно реляционным СУБД, она также может выигрышно дополняться более специализированными NoSQL решениями. MongoDB реализует новый подход к построению баз данных, где нет таблиц, схем, запросов SQL, внешних ключей и многих других вещей, которые присущи объектно-реляционным базам данных. В отличие от реляционных баз данных MongoDB предлагает документо-ориентированную модель данных, благодаря чему MongoDB работает быстрее, обладает лучшей масштабируемостью, ее легче использовать.

Но, даже учитывая все недостатки традиционных баз данных и достоинства MongoDB, важно понимать, что задачи бывают разные и методы их решения бывают разные. В какой-то ситуации MongoDB действительно улучшит производительность приложения, например, если надо хранить сложные по структуре данные. В другой же ситуации лучше будет использовать традиционные реляционные базы данных. Кроме того, можно использовать смешанный подход: хранить один тип данных в MongoDB, а другой тип данных в традиционных БД.

Вся система MongoDB может представлять не только одну базу данных, находящуюся на одном физическом сервере. Функциональность MongoDB позволяет расположить несколько баз данных на нескольких физических серверах, и эти базы данных смогут легко обмениваться данными и сохранять целостность.

Формат данных в MongoDB

Одним из популярных стандартов обмена данными и их хранения является JSON (JavaScript Object Notation). JSON эффективно описывает сложные по структуре данные. Способ хранения данных в MongoDB в этом плане похож на JSON, хотя формально JSON не используется. Для хранения в MongoDB применяется формат, который называется BSON или сокращение от binary JSON.

BSON позволяет работать с данными быстрее: быстрее выполняется поиск и обработка. Хотя надо отметить, что BSON в отличие от хранения данных в формате JSON имеет небольшой недостаток: в целом данные в JSON-формате занимают меньше места, чем в формате BSON, с другой стороны, данный недостаток с лихвой окупается скоростью.

Кроссплатформенность

MongoDB написана на C++, поэтому ее легко портировать на самые разные платформы. MongoDB может быть развернута на платформах Windows, Linux, MacOS, Solaris. Можно также загрузить исходный код и самому скомпилировать MongoDB, но рекомендуется использовать библиотеки с офсайта.

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.

Ключ представляет простую метку, с которым ассоциировано определенный кусок данных.

Однако при всех различиях есть одна особенность, которая сближает MongoDB и реляционные базы данных. В реляционных СУБД встречается такое понятие как первичный ключ. Это понятие описывает некий столбец, который имеет уникальные значения. 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.

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 (в зависимости от настроек конкретной бд) 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.

Коллекции

Если в традиционном SQL есть таблицы, то в 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.

Репликация

Система хранения данных в MongoDB представляет набор реплик. В этом наборе есть основной узел, а также может быть набор вторичных узлов. Все вторичные узлы сохраняют целостность и автоматически обновляются вместе с обновлением главного узла. И если основной узел по каким-то причинам выходит из строя, то один из вторичных узлов становится главным.

Простота в использовании

Отсутствие жесткой схемы базы данных и в связи с этим потребности при малейшем изменении концепции хранения данных пересоздавать эту схему значительно облегчают работу с базами данных MongoDB и дальнейшим их масштабированием. Кроме того, экономится время разработчиков. Им больше не надо думать о пересоздании базы данных и тратить время на построение сложных запросов.

Графический интерфейс

MongoDB Compass - это мощный графический интерфейс для запросов, агрегирования и анализа ваших данных MongoDB в визуальной среде https://www.mongodb.com/docs/compass/current/

С помощью Compass, графического интерфейса для MongoDB, легко исследовать базу данных и управлять ею. Интуитивно понятный и гибкий Compass предоставляет подробную визуализацию схемы, показатели производительности в реальном времени, сложные возможности обработки запросов и многое другое.

Пожалуйста, обратите внимание, что MongoDB Compass поставляется в трех версиях: полной версии со всеми функциями, версии только для чтения без возможностей записи или удаления и изолированной версии, единственное сетевое подключение которой - к экземпляру MongoDB..

GridFS

Одной из проблем при работе с любыми системами баз данных является сохранение данных большого размера. Можно сохранять данные в файлах, используя различные языки программирования. Некоторые СУБД предлагают специальные типы данных для хранения бинарных данных в БД (например, BLOB в MySQL).

В отличие от реляционных СУБД MongoDB позволяет сохранять различные документы с различным набором данных, однако при этом размер документа ограничивается 16 мб. Но MongoDB предлагает решение - специальную технологию GridFS, которая позволяет хранить данные по размеру больше, чем 16 мб.

Система GridFS состоит из двух коллекций. В первой коллекции, которая называется files, хранятся имена файлов, а также их метаданные, например, размер. А в другой коллекции, которая называется chunks, в виде небольших сегментов хранятся данные файлов, обычно сегментами по 256 кб.

Для тестирования GridFS можно использовать специальную утилиту mongofiles, которая идет в пакете mongodb.

Вопросы для самоконтроля

1. *Какую модель данных поддерживает СУБД MongoDB?*
2. *Какой формат данных поддерживается в MongoDB?*
3. *Из чего состоит база данных в MongoDB?*
4. *Имеет ли БД в MongoDB жесткую структуру?*

Установка СУБД MongoDB

Официальный сайт предоставляет пакеты дистрибутивов для различных платформ: Windows, Linux, MacOS, Solaris.

При использовании MongoDB следует учитывать, что MongoDB использует отображаемые в памяти файлы, поэтому на 32-х разрядных системах ее действие ограничено 2 Гб памяти. При развертывании на 64-х разрядных системах подобных ограничений нет.

Для установки и запуска СУБД выполните следующие шаги:

1. Зайдите на официальную страницу скачивания и скачайте бесплатную версию MongoDB Community Server.
2. По умолчанию все файлы устанавливаются по пути `C:\Program Files\MongoDB`. Можно изменить каталог установки. Для этого создайте на диске `C` новую папку `mongodb` и затем укажите ее в качестве места установки.
3. Распакуйте архив и перейдите в папку `bin`. `mongod`— это сервер,.
4. Запустите сервер mongod.
5. Чтобы подключиться к запущенному серверу, нужно запустить клиент `mongosh` . Оболочка MongoDB (mongosh) не установлена с сервером MongoDB. Вам необходимо следовать инструкциям по установке mongosh, чтобы загрузить и установить mongosh отдельно.
6. С помощью **Compass** (отдельно надо устанавливать https://www.mongodb.com/try/download/shell), **графического интерфейса** для MongoDB, легко исследовать базу данных и управлять ею, здесь присутствует панель командной оболочки **mongosh** .

Папку `bin` можно для удобства добавить в переменную окружения `PATH`. Для пользователей MacOSX и Linux инструкции практически те же самые. Всё, что нужно сделать – изменить пути.

Если есть ошибки, нужно читать сообщения в консоли – сервер подробно и ясно выводит диагностические сообщения.

Работа с консолью

Для проверки работоспособности следует запустить клиент mongosh.

Для работы с mongosh используйте документацию https://www.mongodb.com/docs/mongodb-shell/reference/access-mdb-shell-help/

В случае успешной связи с сервером команда выведет версию MongoDB и сообщит, что текущей является база данных test:

```
Current Mongosh Log ID: 65609f46f1a4a30bfe1079b9
Connecting to:
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&
appName=mongosh+2.0.2
Using MongoDB:          7.0.3
Using Mongosh:          2.0.2
……
test>
```

Есть несколько глобальных команд, например `help` или `exit`. Команды (или методы), которые запускаются применительно к текущей базе данных, исполняются у объекта `db`, например, `db.help()` или `db.stats()`. Команды, которые запускаются применительно к конкретной коллекции, исполняются у объекта `db.ИМЯ_КОЛЛЕКЦИИ`, например, `db.unicorns.help()` или `db.unicorns.count()`.

Чтобы проверить версию сервера, необходимо использовать команду `db.version()`. Список команд для объекта `db` можно получить командой `db.help()`.

Поскольку консоль интерпретирует JavaScript, если попытаться выполнить метод без скобок, то в ответ выведется тело метода, но он не выполнится: `function (...){…}`. Например, если ввести `db.help` (без скобок), выведется внутреннее представление метода `help`.

Работа с БД

Создание и удаление базы данных

Не существует явного способа создания базы данных, база данных создается автоматически при записи в любую из её коллекций документа. Однако, для удаления базы данных существует специальный метод `dropDatabase()`:

```
use database
```

Напрмер, для создания БД «semenov» команда: `db> use semenov`

```
db.dropDatabase()
```

Здесь `db` — это псевдоним, обозначающий текущую базу данных, которая выбирается при помощи команды `use`.

Получить список доступных баз данных можно при помощи команды:

```
show dbs
```

В качестве результат возвращается список доступных баз данных и занимаемый ими объем в гигабайтах.

Статистика базы данных

Выяснить текущее состояние базы данных и занимаемый ей размер можно при помощи метода stats().

```
db.stats()
{
        "db" : "test",
        "collections" : 5,
        "objects" : 100020,
```

```
        "avgObjSize" : 40.00359928014397,
        "dataSize" : 4001160,
        "storageSize" : 11280384,
        "numExtents" : 11,
        "indexes" : 4,
        "indexSize" : 5796784,
        "fileSize" : 201326592,
        "nsSizeMB" : 16,
        "dataFileVersion" : {
                "major" : 4,
                "minor" : 5
        },
        ok" : 1"
}
```

Поле **db** сообщает текущее название базы данных, **collections** — количество коллекций, **indexes** — количество индексов, **objects** — количество объектов, содержащихся в базе данных, **avgObjSize** — средний размер объекта. Поле **fileSize** сообщает о размере дискового пространства зарезервированного под базу данных, фактически это сумма файлов коллекции в каталоге данных. Поле **dataSize** сообщает объем всех BSON-объектов базы данных. Поле **storageSize** сообщает объем, занимаемый BSON-объектами с учетом места зарезервированного для роста коллекций и областями, оставшимися после удаленных объектов. Поле **nsSizeMB** сообщает о размере пространства имен (ориентируется на размер файла с расширением ns в каталоге данных). **dataFileVersion** — сообщает версию BSON-формата.

В каждой базе данных существует системная коллекция **system.namespaces**, которая содержит пространства имен базы данных (физически данные этой коллекции хранятся по ns-файлам)

```
db.system.namespaces.find();
{ "name" : "test.test" }
{ "name" : "test.system.indexes" }
{ "name" : "test.test.$_id_" }
{ "name" : "test.mybase.$_id_" }
{ "name" : "test.mybase" }
{ "name" : "test.articles.$_id_" }
{ "name" : "test.articles" }
```

Помимо system.namespaces, каждая база данных имеет системную коллекцию **system.indexes**.

```
db.system.indexes.find();
{ "v" : 1, "key" : { "_id" : 1 }, "ns" : "test.test", "name" : "_id_" }
{ "v" : 1, "key" : { "_id" : 1 }, "ns" : "test.mybase", "name" : "_id_" }
{ "v" : 1, "key" : { "_id" : 1 }, "ns" : "test.articles", "name" : "_id_" }
```

Коллекции

Коллекции предназначены для хранения документов и сами в свою очередь хранятся в базах данных. Имя коллекции может содержать символы английского алфавита, цифры и точку. MongoDB поддерживает расширенные имена, включающее имя базы данных и имя коллекции, разделенные точкой. Например, коллекция **articles** из базы данных **test** имеет полное имя **test.articles**. Обычно вместо точного названия базы данных используется псевдоним **db**, ссылающийся на текущую базу данных, выбранную при помощи оператора **use**. Полное имя не должно превышать 128 символов.

Коллекции подразделяются на следующие виды:

- стандартные — создаются по умолчанию;
- ограниченные — коллекции постоянного размера, при достижении которого старые записи удаляются автоматически, освобождая место для новых;

- системные — коллекции предназначенные для внутреннего использования, а также выполняющие статистическую роль информационной схемы в реляционных СУБД. Примерами таких коллекций служат system.namespaces и system.indexes.

Создание коллекции

Явного создания коллекции не требуется — она создается неявно при вставке в неё первого документа. Однако, при необходимости можно воспользоваться методом `createCollection()`.

`db.createCollection("articles")` Например: `semenov> db.createCollection("art")`

Метод `createCollection()` необходим, когда под коллекцию требуется зарезервировать определенное количество байт на диске, например, чтобы обеспечить неразрывное хранение документов.

```
db.createCollection("articles", {size: 64000})
```

Для того, чтобы создать ограниченную коллекцию, помимо размера, необходимо установить атрибут `capped` в значение `true`.

```
db.createCollection("articles", {capped: true, size: 64000})
```

По достижению размера, заданного атрибутом size старые записи будут удаляться автоматически, размер коллекции при этом будет оставаться постоянным.

Список коллекций

Просмотреть список текущих коллекций можно при помощи команды

```
show collections
```

или метода

`db.getCollectionNames()`.Например:`semenov> db.getCollectionNames()`, тогда метод выдаст `[ 'статья', 'art' ]`

Переименование коллекции

Для переименования коллекции предназначен метод `renameCollection()`.

```
db.articles.renameCollection("pages")
```

Удаление коллекции

Для удаления коллекции, вместе со всеми индексами и документами можно воспользоваться методом `drop()`.

```
db.articles.drop()
```

Статистика коллекций

Получить детальную информацию о конкретной коллекции можно при помощи метода `stats()`. Ниже приводится пример использования метода `stats()` запрашивающего детальную информацию о коллекции articles текущей базы данных.

```
db.articles.stats()
{
        "ns" : "test.articles",
        "count" : 3,
        "size" : 368,
        "avgObjSize" : 122.66666666666667,
        "storageSize" : 12288,
        "numExtents" : 1,
        "nindexes" : 1,
        "lastExtentSize" : 12288,
```

```
        "paddingFactor" : 1,
        "systemFlags" : 1,
        "userFlags" : 0,
        "totalIndexSize" : 8176,
        "indexSizes" : {
                "_id_" : 8176
        },
        "ok" : 1
}
```

Вставка нового документа

Внутри текущей базы данных создаются коллекции, вставка нового документа в коллекцию при помощи метода insert приводит к её автоматическому созданию. В качестве аргумента, метод insert принимает JSON-объект, который служит телом документа. Ниже в базу данных вставляются документы с единственным ключом title, в качестве значения которого выступают названия различных СУБД.

```
db.mybase.insert({title: "MySQL"})
db.mybase.insert({title: "PostreSQL"})
db.mybase.insert({title: "MongoDB"})
db.mybase.insert([{title: "MS SQL"}, {title: "Oracle"}])
```

Формат JSON допускает создание как единичных объектов, так и массивов. Записи "MySQL", "PostreSQL" и "MongoDB" вставляются отдельными вставками, в то время как "MS SQL" и "Oracle" при помощи массива, элементы которого в JSON заключаются в квадратные скобки. Для каждого документа автоматически создается идентификатор"_id", который можно задавать вручную, с единственным условием, чтобы он оставался уникальным в рамках текущей базы данных. Идентификатор, который формируется по умолчанию, формируется по следующему алгоритму: в старших четырех байтах находится время создания записи в формате UNIXSTAMP, следующие три байта — идентификатор компьютера, следующие два — идентификатор процесса, последние три — локальный счетчик процесса.

Загрузка данных из файла

Данные для базы данных mongodb можно определять в обычном текстовом файле, что довольно удобно, поскольку мы можем переносить или пересылать этот файл независимо от базы данных mongodb.

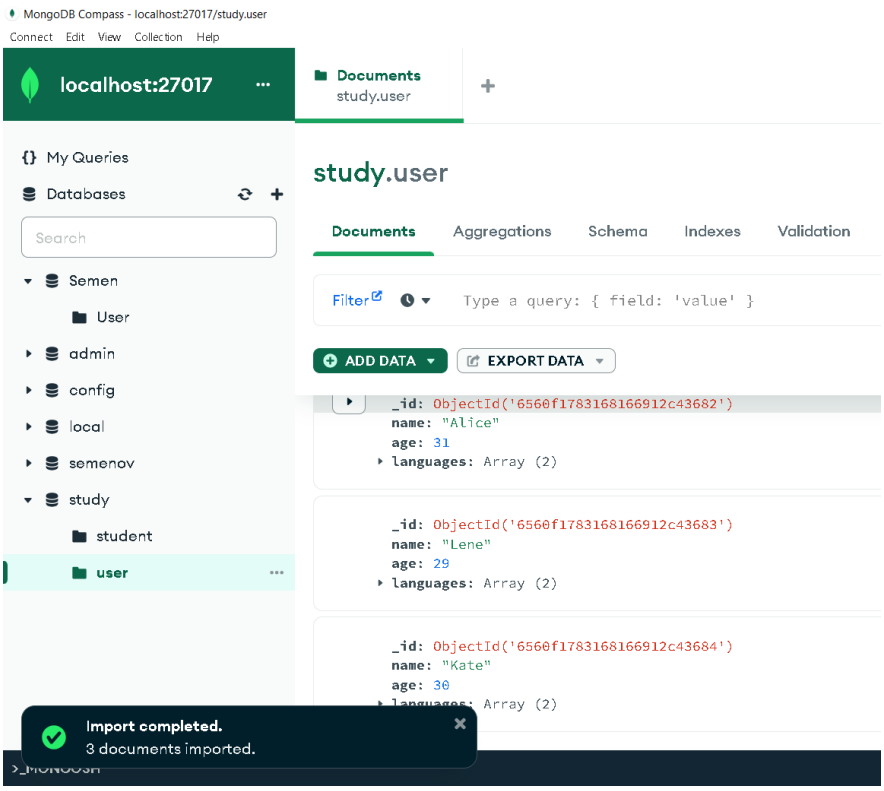
Например, определим где-нибудь на жестком диске файл **user.json** со следующим содержимым:

```
[
{"name": "Alice", "age": 31, "languages": ["english", "french"]},
{"name": "Lene", "age": 29, "languages": ["english", "spanish"]},
{"name": "Kate", "age": 30, "languages": ["german", "russian"]}
]
```

создадим в текущей БД коллекцию **user**

```
db.createCollection("user")
```

Для загрузки файла в текущую базу данных выбирается команда **Import JSON** из выпадающего списка **ADD DATA** приложения **MongoDB Compass**, где после открытия окна выбирается на жестком диске файл **user.json,** после подтверждения добавляются три документа в коллекцию.



Практическое задание:

1. Установите MongoDB в зависимости от типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента **mongo** или **mongosh**.
3. Выполните методы:
   a) db.help()
   b) db.help
   c) db.stats()
4. Получите список доступных БД.

5. Создайте БД в соответствии с вариантом задания Лабораторной работы №1, имя БД вводится латиницей в формате «Фамилия_группа», например,

```
db> use Ivanov_ks30
```

6. Создайте коллекцию, вставив в нее документы в соответствии со строками таблицы вашего варианта задания Лабораторной работы №1

Например:

Результаты ЕГЭ:

Дата экзамена	Школа, №	Балл	Ученик	Возраст	Дисциплина	Экзаменатор
3.06.06	114	85	Иванов	18	Математика	Кузнецов
15.06.06	295	79	Петрова	17	Русский Язык	Филатова
2.06.06	1197	91	Сидорова	19	Математика	Кузнецов
2.06.06	114	62	Иванов	18	Русский Язык	Филатова
14.06.06	1197	77	Сидорова	19	Физика	Гришин
4.06.06	295	81	Петрова	17	Математика	Кузнецов

В СУБД MongoDB создайте базу данных

7. Выведете список текущих коллекций вашей БД.

8. Выведете содержимое созданной коллекции

9. В приложении **MongoDB Compass** сделайте экспорт коллекции вашей БД в файл *json.

10. Файл отчета в формате WORD и файл коллекции в формате *.json загрузите на учебный портал в курса «Управление данными» в раздел «Сдать Лаб.7»

Структура отчета:

1. Наименование работы.
2. Цель работы.
3. Практическое задание.
4. Выполнение.

Указание: привести результаты выполнения практических заданий (номер задания, формулировка, команда, скриншот результата, вывод (при необходимости)).

5. Выводы.

Список источников:

1. MongoDB. Документация [Электронный ресурс] // Официальный сайт MongoDB. URL: https://docs.mongodb.com/manual/tutorial/manage-mongodb-processes/ (дата обращения: 02.11.2023).

2. Установка и начало работы с MongoDB на Windows [Электронный ресурс] https://metanit.com/nosql/mongodb/1.2.php (дата обращения: 02.11.2023)

3. MongoDB Tutorial [Электронный ресурс] https://www.w3schools.com/mongodb/index.php (дата обращения: 02.11.2023)