А. И. Костюк
Д. А. Беспалов

Администрирование баз данных и компьютерных сетей

учебное пособие

ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ЮФУ
РОСТОВ-НА-ДОНУ-ТАГАНРОГ

УДК 004.65(075.8)
ББК 32.973-018.2я73
К728

Печатается по решению кафедры вычислительной техники Института компьютерных технологий и информационной безопасности Южного федерального университета (протокол № 6 от 23 января 2020 г.)

Рецензенты:
кандидат физико-математических наук, доцент, директор Института радиотехнических систем и управления *А. С. Болдырев*

кандидат технических наук, доцент кафедры систем автоматического управления автоматизированного проектирования ИКТИБ ЮФУ *О. Б. Лебедев*

Костюк, А. И.

К728 Администрирование баз данных и компьютерных сетей : учебное пособие / А. И. Костюк, Д. А. Беспалов ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2020. – 127 с.

ISBN 978-5-9275-3577-4

В учебном пособии описываются основы администрирования баз данных и компьютерных сетей.

Предназначено для студентов направления 09.03.01 и 09.04.01 «Информатика и вычислительная техника», специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения» всех форм обучения и слушателей СФПК.

УДК 004.65(075.8)
ББК 32.973-018.2я73

ISBN 978-5-9275-3577-4

СОДЕРЖАНИЕ

1. NOSQL БАЗА ДАННЫХ MONGODB

1.1. Введение

MongoDB реализует совершенно новый подход к построению баз данных, где нет таблиц, схем, прямых запросов SQL, внешних ключей и многих других вещей, которые присущи привычным нам реляционным базам данных.

В отличие от последних, MongoDB предлагает документо-ориентированную модель данных, благодаря чему MongoDB работает быстрее для определенного типа операций, обладает лучшей масштабируемостью, ее легче использовать.

Однако, учитывая все недостатки традиционных баз данных и достоинства MongoDB, важно понимать, что задачи бывают разные и методы их решения тоже отличаются.

В какой-то ситуации MongoDB действительно улучшит производительность приложения, например, если надо хранить сложные по структуре данные или же схема данных не определена заранее или может меняться в процессе работы и даже от записи к записи.

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 Базах Данных (БД) [1]. Это абсолютно нормально, так как MongoDB невероятно быстрая при запросах документов по идентификатору, но по умолчанию не умеет сливать несколько документов в один (SQL JOIN).

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

Этот факт делает MongoDB удобным для «стартапов» и для небольших проектов, которые постоянно адаптируются к требованиям и к условиям эксплуатации.

Рассмотрим далее основные особенности MongoDB.

1.2. Особенности MongoDB

Для начала стоит сказать, что основным способом хранения данных в MongoDB является представление в формате, похожем на JSON (JavaScript Object Notation). Сам по себе JSON эффективно описывает сложные по структуре данные и является стандартом «де-факто» для современных языков программирования, например JavaScript, Python.

Способ хранения данных в MongoDB в этом плане очень похож на оригинальный JSON, хотя формально сам JSON в чистом виде не используется. Для хранения в MongoDB применяется свой совместимый формат, который называется BSON или сокращение от binary JSON.

BSON позволяет работать с данными быстрее: например, поиск и обработка информации занимают намного меньше времени, чем в других форматах. Хотя надо отметить, что BSON в отличие от 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.

Рассмотрим основные возможности данной СУБД:

- Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных).
- Достаточно гибкий язык для формирования запросов.
- Динамические запросы.
- Полная поддержка индексов.
- Профилирование запросов.
- Быстрые обновления «на месте».
- Эффективное хранение двоичных данных больших объёмов, например фото и видео.
- Журналирование операций, модифицирующих данные в БД.
- Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и шардинг (один из видов распределенного хранения).
- Может работать в соответствии с парадигмой MapReduce.
- Полнотекстовый поиск, в том числе на русском языке, с поддержкой морфологии.

СУБД MongoDB эффективно управляет наборами JSON-подобных документов, хранимых в двоичном виде, благодаря вызовам протокола GridFS.

Среди других отличий от традиционных реляционных СУБД [2]:

• Отсутствует оператор «join». Обычно данные могут быть организованы более денормализованным способом, но на разработчиков ложится дополнительная нагрузка по обеспечению непротиворечивости данных.

• Нет такого понятия, как «транзакция». Атомарность гарантируется только на уровне целого документа, т.е. частичного обновления документа произойти не может.

• Отсутствует понятие «изоляции». Любые данные, которые считываются одним клиентом, могут параллельно изменяться другим клиентом.

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 официальных источников.

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 которой ассоциирован определенный кусок данных.

Однако при всех различиях есть одна особенность, которая несколько сближает MongoDB и реляционные базы данных. В реляционных СУБД встречается такое понятие как **первичный ключ (идентификатор, ObjectId)**.

Это понятие описывает некий столбец, который имеет уникальные значения. В MongoDB для каждого документа имеется уникальный идентификатор, который называется _id. Если явным образом не указать его значение, то MongoDB автоматически сгенерирует для него значение, например: ObjectId("507f1f77bcf86cd799439011").

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

значения, ему, в зависимости от настроек конкретной БД, можно присвоить значение NULL. В MongoDB это не работает. 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.

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.

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 [3].

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 [1].

Рассмотрим далее обобщенное устройство базы данных MongoDB. Устройство базы данных MongoDB.

Всю модель устройства базы данных в MongoDB можно представить следующим образом (рис. 1).

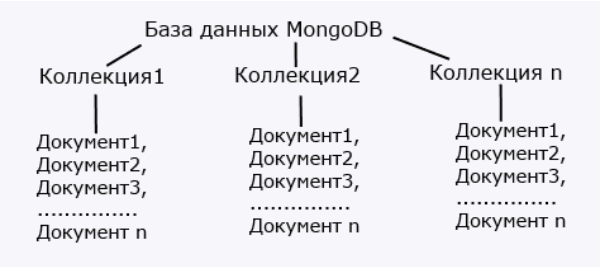

Рис. 1. Структура базы данных

Документ в базе данных, как уже говорилось выше, можно представить как объект, хранящий некоторую информацию. В некотором смысле он подобен строкам в реляционных СУБД, где строки хранят информацию об отдельном элементе.

Например, типичный документ MongoDB:

```
{
    "name": "Bill",
    "surname": "Gates",
    "age": "48",
    "company": {
        "name": "microsoft",
        "year": "1974",
        "price": "300000"
        }
}
```

Этот документ и представляет набор пар ключ-значение. Например, в выражении "name": "Bill" name представляет ключ, а Bill – значение [3].

Ключи представляют собой обычные строки.

Значения же могут различаться по типу данных. В данном случае у нас почти все значения также представляют строковый тип, и лишь один ключ (company) ссылается на отдельный объект. 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, например, {"age": 38}.

• **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.

• **Symbol** – тип данных, идентичный строковому. Используется преимущественно для тех языков, в которых есть специальные символы.

• **Timestamp** – применяется для хранения времени.

В отличие от строк, документы могут содержать разнородную информацию. Так, рядом с документом, описанным выше, в одной коллекции может находиться другой объект, например:

```
{
    "name": "Tom Loun",
    "birthday": "1985.06.28",
    "place": "Yangtown",
    "languages": [
        "english",
        "german",
        "spanish"
    ]
}
```

Казалось бы, разные объекты за исключением отдельных свойств, но все они могут находиться в одной коллекции.

Еще пара важных замечаний: в MongoDB запросы обладают регистрозависимостью и строгой типизацией. То есть следующие два документа не будут идентичны:

```
{"age": "28"}
{"age": 28}
```

Если в первом случае для ключа age определена в качестве значения строка, то во втором случае значением является число.

Для каждого документа в MongoDB определен уникальный идентификатор, который при добавлении документа в коллекцию создается автоматически. Однако разработчик может сам явным образом задать идентификатор, а не полагаться на автоматически генерируемые, указав соответствующий ключ и его значение в документе, хоть это и не приветствуется на практике.

Данное поле должно иметь уникальное значение в рамках коллекции. И если мы попробуем добавить в коллекцию два документа с одинаковым идентификатором, то добавится только один из них, а при добавлении второго мы получим ошибку БД.

Если идентификатор не задан явно, то MongoDB создает специальное бинарное значение размером 12 байт. Это значение состоит из нескольких сегментов: значение типа timestamp размером 4 байта, идентификатор машины из 3-х байт, идентификатор процесса из 2-х байт и счетчик из 3-х байт.

Таким образом, первые 9 байт гарантируют уникальность среди других машин, на которых могут быть реплики базы данных. А следующие 3 байта гарантируют уникальность в течение одной секунды для одного процесса.

Такая модель построения идентификатора гарантирует с высокой долей вероятности, что он будет иметь уникальное значение, ведь она позволяет создавать до 16 777 216 уникальных объектов ObjectId в секунду для одного процесса [3].

При разработке MongoDB авторы исходили из необходимости специализации баз данных, благодаря чему им удалось отойти от принципа «один размер подо всё». За счёт минимизации семантики для работы с транзакциями появляется возможность решения целого ряда проблем,

связанных с недостатком производительности, причём горизонтальное масштабирование становится проще. Используемая модель документов хранения данных (JSON/BSON) проще кодируется, проще управляется (в том числе за счёт применения так называемого «бессхемного стиля» (англ. schemaless style)), а внутренняя группировка релевантных данных обеспечивает дополнительный выигрыш в быстродействии. Нереляционный подход весьма удобен для создания баз данных, у которых горизонтальное масштабирование подразумевает разворачивание на множестве машин. Возможность обеспечивать наилучшую производительность должна существовать параллельно с поддержкой более обширной функциональности, чем это позволяет использование пар «ключ-значение» (в чистом виде). Технология баз данных должна работать везде, начиная с серверов пользователя и виртуальных машин и заканчивая облачными технологиями [2].

MongoDB, по мнению разработчиков, должна заполнить разрыв между простыми хранилищами данных типа «ключ-значение» (быстрыми и легко масштабируемыми) и большими РСУБД (со структурными схемами и мощными запросами).

1.3. Установка MongoDB

Для установки MongoDB в ОС Linux на примере Ubuntu необходимо выполнить следующие шаги [4]:

1. Импортировать публичный ключ к репозиторию на машину разработчика:

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
```

2. Если импорт не удался, следует установить пакет gnupg с зависимостями:

```
sudo apt-get install gnupg
```

3. Повторить импорт ключа
4. Добавить новый репозиторий в список на машине разработчика:

Для Ubuntu 20.04

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
```

Для Ubuntu 18.04

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
```

5. Перезагрузить базу пакетов в системе:

```
sudo apt-get update
```

6. Установить последнюю версию MongoDB:

```
sudo apt-get install -y mongodb-org
```

7. Или установить специфическую версию MongoDB:

```
sudo apt-get install -y mongodb-org=4.4.0 mongodb-org-server=4.4.0 mongodb-org-shell=4.4.0 mongodb-org-mongos=4.4.0 mongodb-org-tools=4.4.0
```

Далее можно запустить MongoDB вручную или добавить ее как сервис в систему:

```
sudo systemctl start mongod
```

Проверить статус:

```
sudo systemctl status mongod
```

Сделать запускаемой на старте ОС:

```
sudo systemctl enable mongod
```

Так же можно получить доступ к консоли MongoDB путем выполнения команды:

```
mongo
```

Далее рассмотрим способ установки MongoDB на ОС Wiindows.

Установка в ОС Windows заключается в загрузке дистрибутива из специального центра в сети Интернет:

https://www.mongodb.com/try/download/community?tck=docs_server.

Там необходимо выбрать версию ОС и нажать кнопку Download.

После установки надо создать на жестком диске каталог, в котором будут находиться базы данных MongoDB.