УНИВЕРСИТЕТ ИТМО

Проектирование БД

Лекция 9. MongoDB

Про что мы сегодня? (в прошлый-то раз SQL)

MongoDB	ElasticSearch
Документо-ориентированная БД	Поисковая БД
Написана на C++	Написана на Java
Поддержка транзакций (через изоляцию снимков состояний)	Нет поддержки транзакций
Поддержка SQL-запросов на чтение	Собственный язык запросов, похожий на SQL

Сравнение понятий

PostgreSQL (any SQL)	MongoDB	ElasticSearch
Схема	Схема	Индекс
Таблица	Коллекция	Маппинг ~~(Тип)~~ (с ES 7)
Кортеж	BSON-объект	JSON-объект
Атрибут	Поле	Поле

MongoDB

- Базовыми понятиями в MongoDB являются коллекции и документы.
- Коллекция является эквивалентом таблицы в реляционных системах управления базами данных.
- Документ – запись в коллекции и базовая единица данных в MongoDB
- Документы в коллекции могут иметь разные поля

MongoDB

- Имеет мощные средства запросов, характерные для реляционных баз данных, и распределенную архитектуру, свойственную таким хранилищам, как Riak
- Хранит JSON-документы (в BSON-формате)
- Поддерживает произвольные запросы
- Удобен, когда массив данных большой, а бюджет ещё мал для приобретения дорогостоящего оборудования.

Представление данных в MongoDB

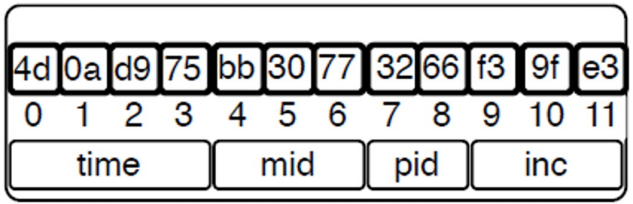
Единица хранения данных в Mongo – **документ** – документ в формате JSON. **Коллекция**– набор документов в формате JSON.

```
$ mongo book

> db.champions.insertOne (
{ name: "Max Verstappen",
  age: 26,
  last_title: ISODate("2023-10-07"),
  famous_for: [”sim racing", ”youngest debut" ],
  team: {
          name: ”Red Bull",
          engine: ”RBPT Honda" }
}
)
```

Идентификатор документа в MongoDB

MongoDB добавляет к каждому документу поле _id типа ObjectID (12 байт), играющего роль первичного ключа.




```
$ mongo book

> db.towns.insert (
{ "_id" : ObjectId("4d0ad975bb30773266f39fe3"),
  name: "Max Verstappen",
  age: 26,
  last_title: ISODate("2023-10-07"),
  famous_for: [”sim racing", ”youngest debut" ],
  team: {
          name: ”Red Bull",
          engine: ”RBPT Honda" }
}
)
```

Особенности документа в MongoDB

В документе хранятся однотипные строго структурированные записи: ключи и значения. Близкий аналог - строка в таблице реляционной базы данных. Но в отличие от строки документ более гибкий — для отдельных записей можно, например, опускать те или иные поля.

Максимальный размер документа — 16 Мб (без GridFS)

А если надо хранить больше 16 Мб?

Используем Grid File System (GridFS). По факту, это то же сегментирование, но в рамках документа. Массивные данные хранятся в двух коллекциях: files и chunks:

files — коллекция со сведениями о файлах (имена, метаданные об объеме и других параметрах);

chunks — коллекция с файлами, разбитыми на сегменты. Размер сегмента обычно 256 Кб, но может настраиваться.

Особенности связей документов в MongoDB

В реляционных БД есть JOIN, помогающий объединить между собой сведения из разных таблиц.

В MongoDB другая идеология: всё находится внутри одной коллекции, а операции JOIN не предусмотрено. Для связей и разделений используются **встроенные документы** — структуры встраиваются друг в друга.

Запросы к MongoDB

Составляются с использованием языка JavaScript.

```
> db.collection_name.command
```

- **db** – JS-объект, который содержит информацию о текущей БД
- **collection_name** – JS-объект, представляющий коллекцию с именем collection_name.
- command – JS-функции (как стандартные – insert (), find (), count (), так и созданные пользователем

Операции над данными в MongoDB

Операция	В языке SQL	В командах MongoDB
Создание	INSERT	insertOne, insertMany
Чтение	SELECT	find
Редактирование	UPDATE	updateOne, updateMany, replaceOne
Удаление	DELETE	deleteOne, deleteMany

Пример создания данных

В языке SQL	В командах mongosh для MongoDB

`insert into champions (name, age, status)`
`values (“Lewis Hamilton”, 39,`
`“active”)`
	`db.champions.insertOne(`
	`{ name : “Lewis Hamilton”, age : 39,`
	`status : “active” },`
	`{ writeConcern: { w : "majority",`
	`wtimeout : 100 } }`
	`)`

	`db.champions.insertMany(`
	`{ name : “Lewis Hamilton”, age : 39,`
	`status : “active” },`
	`{ name : “Nico Rosberg”, age : 38,`
	`status : “retired” }}`
	`)`

Пример чтения данных

В языке SQL	В командах mongosh для MongoDB

`select *`
`  from champions`
`  where status = “active”`
	`db.champions.find(`
	`  { status : “active”}`
	`)`

`select *`
`  from champions`
`  where status = “active”`
`  or age = 38`
	`db.champions.find(`
	`  { $or [ { status : “active” },`
	`  { age : 38} ]`
	`)`

`select count (*)`
`  from champions`
	`db.champions.count()`
	или
	`db.champions.find().count()`

Пример редактирования данных

В языке SQL	В командах mongosh для MongoDB

`update champions`
`  set status = “retired”`
`where age > 50`
`  and name != “Fernando Alonso”`
	`db.champions.updateMany(`
	`  { age : { $gt : 50} },`
	`  { name : { $ne : “Fernando Alonso”} }`
	`  { $set :  { status : “retired” }`
	`)`

`update champions`
`  set status = “jedi”`
`where name = “Fernando Alonso”`
	`db.champions.updateOne(`
	`  { name : “Fernando Alonso” }`
	`  { $set :  { status : “jedi” }`
	`)`

Пример удаления данных

В языке SQL	В командах mongosh для MongoDB

`delete from champions`
`  where status = ”retired”`
	`db.champions.deleteMany(`
	`  { status : “retired” }`
	`)`

`delete from champions`	`db.champions.deleteMany(`
	`  { }`
	`)`

`delete from champions`
`  where name = “Filipe Massa”`
	`db.champions.deleteOne(`
	`  { name : “Felipe Massa” }`
	`)`

Извлечение данных в MongoDB

Запрос конкретного документа:

```
> db.champions.find({"_id" : ObjectId("4d0ad975bb30773266f39fe3") })
```

Запрос конкретных полей конкретного документа:

```
> db.champions.find({"_id" : ObjectId("4d0ad975bb30773266f39fe3") },
                {name : 1})
```

Запрос к вложенным массивам (поиск конкретного значения, сравнение с подстрокой и т.п.):

```
> db.champions.find({famous_for : 'fashion'})
```

Извлечение данных по критериям в MongoDB

Формулировка запросов по значениям полей, диапазонам или с несколькими критериями:

```
field : { $op : value }
```

где $op – операция (= , > , < и другие)

Например, «Найти все города на букву P с населением меньше 10000 человек:

```
> db.champions.find({ name : /^P/, age : { $lt : 40 } }
```

Извлечение данных с сортировкой в MongoDB

К команде find() можно присоединять дополнительные методы

Например, sort - с указанием полей, по которым надо сортировать, и используя указатель 1 для сортировки по возрастанию и -1 – для сортировки по убыванию

```
db.champions.find().sort({age, -1})
```

Извлечение данных с сортировкой в MongoDB

Как и в реляционных СУБД, MongoDB может использовать индексы для сортировки. При этом **без** индекса MongoDB ограничивает размер сортируемых данных.

Если попытаться отсортировать большой объем данных, не используя индекс - **поймаем ошибку**.

Некоторые булевские операции

Операция	Описание
$or	OR
$lt	Меньше
$lte	Меньше или равно
$regex	Соответствие строки регулярному выражению, совместимому с синтаксисом PCRE
$exists	Проверяет существование поля
$all	Соответствие всем элементам массива
$in	Соответствие хотя бы одному элементу массива
$nin	Несоответствие ни одному элементу массива
$elemMatch	Соответствие всех полей вложенного документа

Виды агрегирующих запросов в MongoDB

count (<условие>) – возвращает количество удовлетворяющих критерию документов

```
db.champions.count({ ‘titles’ : {$gt : 4 } })
```

distinct (<условие>) – возвращает удовлетворяющие критерию значения (не полные документы), если хотя бы одно существует

```
db.champions.distinct(titles,
                      { ‘titles’ : {$gt : 4 } })
> [ 5, 7, 7 ]
```

Виды агрегирующих запросов в MongoDB

group(initial, reduce, cond, key) - агрегирует результаты примерно так же, как запросы GROUP BY в SQL.

- initial – инициализация объекта-результата
- key – поле, по которому производится группировка;
- cond – условие, определяющее интересующие значения
- reduce – функция, которая решает, как выводить результаты

Обновление данных в MongoDB

updateMany/updateOne(criteria, operation)

- criteria – условие отбора (как, например для функции find())
- operation – либо объект, поля которого заменяют поля отобранных документов, либо модификатор

```
db.champions.updateOne(
     { "_id" : ObjectId("4d0ad975bb30773266f39fe3") },
     { $set :  { status : “retired” } }
     )
```

Обновление данных в MongoDB

!! С обновлением надо быть аккуратнее.

Если не использовать модификатор $set, будет выполнена **замена** поля, по которому мы делали поиск нужного документа.

Это может быть удобно, когда требуются некоторые динамические обновления, но может приводить к ошибкам.

Обновление со вставкой в MongoDB

Обновление/вставка обновляет документ, если он найден, или создаёт новый - если не найден. Чтобы разрешить вставку при обновлении, нужно установить параметр upsert в true.

Например, нам нужно увеличить количество титулов, а значит надо посмотреть, есть ли уже запись о таком чемпионе вообще, и - в зависимости от результата - выполнить update либо insert

Обновление данных в MongoDB

Модификатор	Описание
$set	Записывает указанное значение в указанное поле
$unset	Удаляет поле
$inc	Прибавляет указанное число к указанному полю
$push	Помещает новый элемент в массив
…	…

Что вместо JOIN в MongoDB?

Операции соединения отсутствуют. Из-за распределенной природы системы соединение оказалось бы крайне неэффективной операцией.

Допускается создание ссылок между документами:

```
{ $ref : “collection_name”, $id : “reference_id” }
```

А ещё что вместо JOIN в MongoDB?

Денормализация

Можно при проектировании заложить (заранее понимая бизнес-логику работы с данными по востребованности данных) повтор информации в документе из другого документа

Удаление документов в MongoDB

deleteOne() / deleteMany() – функции удаления документа/документов из коллекции. Документ удаляется целиком.

```
db.champions.deleteOne({
      "_id" : ObjectId("4d0ada1fbb30773266f39fe4")
})
```

Что интересного для нас в MongoDB

Масштабирование на несколько серверов через

- репликацию (копирование данных на другие серверы)
- сегментирование (разбиение коллекции на части)

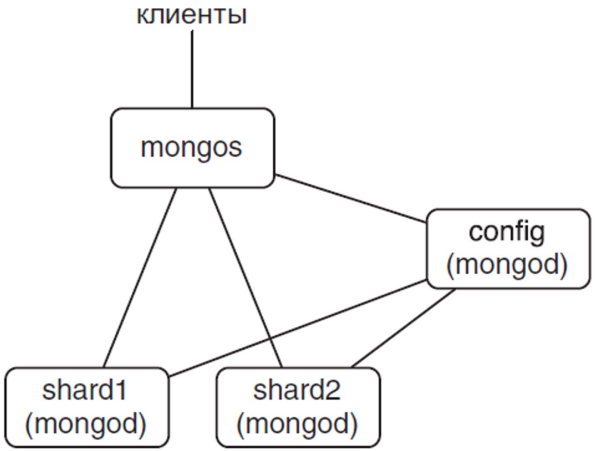
Есть параллельное выполнение запросов.

Репликация производится в режиме Master-Slave

Как сделано сегментирование в MongoDB

При сегментации указывается коллекция и поле сегментирования. Дальнейшее распределение берет на себя механизм автосегментирования

Конфигурационный сервер – управляет информацией о сегментировании



Правила перехода к документно-ориентированной модели базы данных

- Если степень связи 1:1 и класс принадлежности обоих сущностей является обязательным, то создается один документ, одним из полей которого является вложенный документ, содержащий данные о второй сущности
- Если степень связи 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то создается один документ, одним из полей которого может быть вложенный документ, содержащий данные о второй сущности

Правила перехода к документно-ориентированной модели базы данных

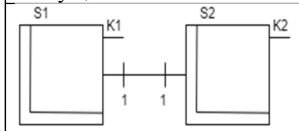
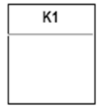
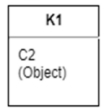
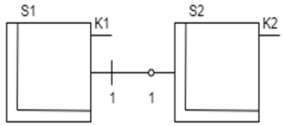
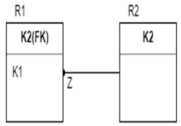
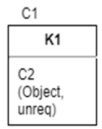
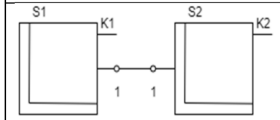
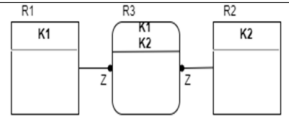
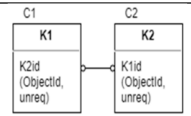
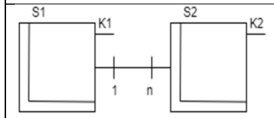
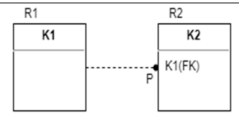
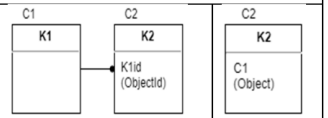
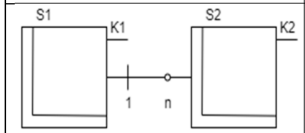
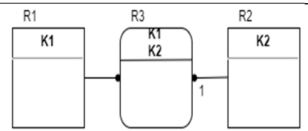
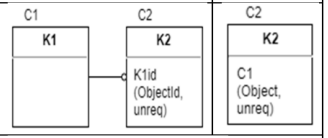
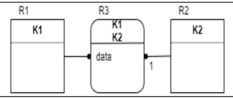
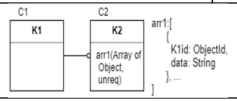
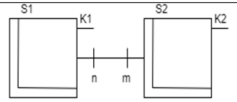
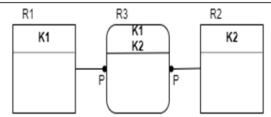
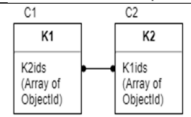
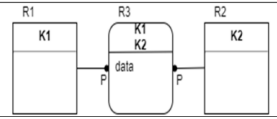
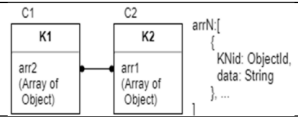
- Если степень связи 1:1 и класс принадлежности ни одной сущности не является обязательным, то создается два документа, при этом в каждом из них может содержаться поле, хранящее ссылку(идентификатор) на другой документ.
- Если степень связи 1:n и класс принадлежности n-связной сущности является обязательным, возможны 2 варианта: -
  - один документ, одним из полей которого является вложенный документ, содержащий данные о второй сущности
  - два документа, при этом второй документ содержит поле, хранящее ссылку(идентификатор) на первый документ.

Правила перехода к документно-ориентированной модели базы данных

- Если степень связи 1:n, класс принадлежности n-связной сущности является необязательным и нет дополнительных полей, то возможны 2 варианта:
  - один документ, одним из полей которого может быть вложенный документ, содержащий данные о второй сущности
  - два документа, при этом второй документ может содержать поле, хранящее ссылку(идентификатор) на первый документ.

Правила перехода к документно-ориентированной модели базы данных

- Если степень связи m:n и нет дополнительных полей, создается 2 документа, каждый из которых содержит массив ссылок на другой документ.
- Если степень связи m:n и есть дополнительные поля, создается 2 документа, каждый из которых содержит массив объектов следующего типа: arrN:[{ KNid: ObjectId, data: String }, ...], где KNid – ссылка на другой документ data – дополнительное поле

Концептуальное проектирование S - сущность	Реляционное проектирование (методология IDEF1X) R - отношение	Документно-ориентированное проектирование C - коллекция
		
		
		
		
		
		
		
		

УНИВЕРСИТЕТ ИТМО

ITMO more than a UNIVERSITY

Преимущества MongoDB

- Скорость работы
- Гибкость
- Лёгкая масштабируемость
- Отсутствие JOIN
- Возможность распределённой работы («из коробки»)
- Поддержка в языках программирования (н-р, Mongoose)

Недостатки MongoDB

- Отсутствие хранимых процедур и функций
- Неполное соответствие ACID
- Сложности с транзакциями
- Отсутствие JOIN

Сложности с транзакциями в MongoDB

Что вместо транзакций в MongoDB?

- Множество атомарных операций ($inc, $set, $findAndModify и т.п.)
- Двухфазный commit (независимое от хранилища решение выполняемое в коде. Состояние транзакции хранится внутри обновляющегося документа, а все необходимые шаги – init-pending-commit/rollback – выполняются вручную)

Что за findAndModify?

```
db.champions.findAndModify({
     query : { name : “Max Verstappen”, age : { $gt : 25 } },
     sort : { age : 1 },
     update : { $inc : { titles : 1 } }
})
```

Так же для команды предусмотрена опция upsert (обновление/вставка), позволяющая при значении true создавать новый документ, если ничего не было найдено

Но с версии 4.0 (2018) транзакции есть!

Помните 12-й слайд? (ну конечно же, помните)

```
db.champions.insertOne(
      { name : “Lewis Hamilton”, age : 39,
        status : “active” },
      { writeConcern: { w : "majority",
        wtimeout : 100 } }
)
```

Пример ручного управления конкретной командой вставки с гарантией записи по большинству

Спасибо за внимание!

www.ifmo.ru

ITMO more than a UNIVERSITY