

ЗАНЯТИЕ 1.6

NoSQL&MongoDB



Алексей Кузьмин

Директор разработки; Data Scientist

ДомКлик.ру



aleksej.kyzmin@gmail.com

—

NoSQL

Что такое базы данных NoSQL?

- Специально созданы для определенных моделей данных
- Обладают гибкими схемами
- Получили широкое распространение в связи с простотой разработки, функциональностью и производительностью при любых масштабах
- Применяются различные модели данных

САР-теорема

- Consistency (Согласованность). Как только мы успешно записали данные в наше распределенное хранилище, любой клиент при запросе получит эти последние данные.
- Availability (Доступность). В любой момент клиент может получить данные из нашего хранилища, или получить ответ об их отсутствии, если их никто еще не сохранял.
- Partition Tolerance (Устойчивость к разделению системы). Потеря сообщений между компонентами системы (возможно даже потеря всех сообщений) не влияет на работоспособность системы. Здесь очень важный момент состоит в том, что если какие-то компоненты выходят из строя, то это тоже подпадает под этот случай, так как можно считать, что данные компоненты просто теряют связь со всей остальной системой.

Теорема САР (известная также как теорема Брюера) — эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств

Классы систем

В системе класса СА во всех узлах данные согласованы и обеспечена доступность, при этом она жертвует устойчивостью к распаду на секции.

Система класса СР в каждый момент обеспечивает целостный результат и способна функционировать в условиях распада, но достигает этого в ущерб доступности: может не выдавать отклик на запрос.

В системе класса АР не гарантируется целостность, но при этом выполнены условия доступности и устойчивости к распаду на секции. Хотя системы такого рода известны задолго до формулировки принципа САР (например, распределённые веб-кэши или DNS).



Для чего можно использовать базы данных NoSQL?

- Гибкость.
- Масштабируемость.
- Высокая производительность.
- Широкие функциональные возможности.

Типы баз данных NoSQL



- БД на основе пар «ключ-значение».



- Документ



- Графовые БД



- БД в памяти



- Поисковые БД.

—

MongoDB

Модель устройства базы данных в MongoDB



1. База данных состоит из коллекций
2. Центральным понятием является документ
3. Документ представляет набор пар ключ-значение



Реляционная БД - MongoDB

База данных - База данных

Таблица - Коллекция

Ряд - Документ

Значение в столбце - Поле

Первичный ключ (primary key) - По умолчанию MongoDB генерирует
Default key_id

--Типичный документ

```
{  
  "name": "Bill",  
  "surname": "Gates",  
  "age": "48",  
  "company": {  
    "name" : "microsoft",  
    "year" : "1974",  
    "price" : "3000000"  
  }  
}
```

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": 29}

-
- 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

Запросы обладают регистрозависимостью и строгой типизацией.

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

Не идентичны

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


Идентификатор документа

Для каждого документа в MongoDB определен уникальный идентификатор, который называется `_id`:

- идентификатор создается автоматически
- можно явным образом задать идентификатор

Выбор и создание базы данных

use название-базы-данных

Не важно, существует ли такая бд или нет

show dbs - вывести названия всех имеющихся бд на консоль (если есть такие права)

- Можно задать почти любое имя
- Не должно быть символов `/, \, ., ", *, <, >, :, |, ?, $`
- Ограничение длины 64 байта
- Зарезервированные имена: `local`, `admin`, `config`.

db - узнать текущую БД

db.stats() - получение статистики по текущей БД

db.название-коллекции.stats() - получение статистики по коллекции

Ограничение на имена коллекций:

- не должно начинаться с префикса `system`.
- не должно содержать символ `$`.

Управление коллекцией

`db.createCollection(имяколлекции, свойстваколлекции)` - создание коллекции

Имя коллекции (String) - имя создаваемой коллекции

Свойства коллекции (Document) - определяет такие свойства, как индексация и память:

- `capped` (Boolean) - создаёт ограниченную по размеру коллекцию. Если количество элементов в коллекции достигло максимума, то каждая следующая запись будет перезаписывать более старые данные.
- `autoIndexID` (Boolean) - автоматически создаёт индекс для поля `_id`.
- `size` (number) - определяет максимальный размер (в байтах) ограниченной (`capped`) коллекции. Если коллекция ограничена, то необходимо обязательно указать максимальный размер.
- `max` (number) - определяет максимальное количество документов в ограниченной коллекции.

`db.имя_коллекции.renameCollection("новое_название")` - переименование коллекции

Добавление данных

Данные хранятся в БД в формате BSON, который близок к JSON

Три метода добавления:

- `insertOne()`: добавляет один документ
- `insertMany()`: добавляет несколько документов
- `insert()`: может добавлять как один, так и несколько документов
- `load()` - загрузка данных из файла

Выборка из БД

db.ИМЯ_КОЛЛЕКЦИИ.find() - аналог SELECT в SQL

db.ИМЯ_КОЛЛЕКЦИИ.find().pretty() - структурированный вывод данных коллекции

[]

db.users.find({name: "Tom"})

SELECT * FROM Table WHERE Name='Tom'

Проекция

включить в выборку только нужные поля

db.ИМЯ_КОЛЛЕКЦИИ.find({}, {АТТРИБУТ: 1})

Условные операторы

\$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 должны иметь поле документа

Логические операторы

`$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 соответствовать обоим условиям

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


Команды группировки

`db.ИМЯ_КОЛЛЕКЦИИ.[find({}, {АТТРИБУТ: 1}).]count()` - число элементов в коллекции, удовлетворяющих условию

`db.ИМЯ_КОЛЛЕКЦИИ.distinct(АТТРИБУТ)` - найти только уникальные различающиеся значения

`db.ИМЯ_КОЛЛЕКЦИИ.group([параметры])` - аналогично GROUP BY SQL

Возврат набора данных

Параметры:

- `key`: указывает на ключ, по которому надо проводить группировку
- `initial`: инициализирует поля документа, который будет представлять группу документов
- `reduce`: представляет функцию, возвращающую количество элементов. Эта функция принимает в качестве аргументов два параметра: текущий элемент и агрегатный результирующий документ для текущей группы
- `keyf`: необязательный параметр. Используется вместо параметра `key` и представляет функцию, которая возвращает объект `key`
- `cond`: необязательный параметр. Представляет собой условие, которое должно возвращать `true`, иначе документ не примет участия в группировке. Если данный параметр не указан, то в группировке участвуют все документы
- `finalize`: необязательный параметр. Представляет функцию, которая срабатывает перед тем, как будут возвращены результаты группировки.

```
db.users.group ({key: {name : true}, initial: {total : 0},  
reduce : function (curr, res){res.total += 1}})
```

Поиск по массивам

\$all: определяет набор значений, которые должны иметься в массиве

\$size: определяет количество элементов, которые должны быть в массиве

\$elemMatch: определяет условие, которым должны соответствовать элементы в массиве

Операторы выборки

\$exists - извлечь только те документы, в которых определенный ключ присутствует или отсутствует

\$type извлекает только те документы, в которых определенный ключ имеет значение определенного типа

\$regex задает регулярное выражение, которому должно соответствовать значение поля

Обновление данных

`db.ИМЯКОЛЛЕКЦИИ.update(КРИТЕРИЙВЫБОРА, ИЗМЕНЁННЫЕ_ДАННЫЕ [,ОПЦИИ])` - изменяет значения уже существующего документа

КРИТЕРИЙ_ВЫБОРА: принимает запрос на выборку документа, который надо обновить

ИЗМЕНЁННЫЕ_ДАННЫЕ: представляет документ с новой информацией, который заместит старый при обновлении

ОПЦИИ: определяет дополнительные параметры при обновлении документов. Может принимать два аргумента: `upsert` и `multi`.

Если параметр `upsert` имеет значение `true`, что `mongodb` будет обновлять документ, если он найден, и создавать новый, если такого документа нет. Если же он имеет значение `false`, то `mongodb` не будет создавать новый документ, если запрос на выборку не найдет ни одного документа.

Параметр `multi` указывает, должен ли обновляться первый элемент в выборке (используется по умолчанию, если данный параметр не указан) или же должны обновляться все документы в выборке.

db.ИМЯ_КОЛЛЕКЦИИ.save({id:ObjectId(), НОВЫЕДАННЫЕ}) - используется для перезаписи уже существующего документа новым

Если метод находит документ с таким значением _id, то документ обновляется.

Если же с подобным _id нет документов, то документ вставляется. Если параметр _id не указан, то документ вставляется, а параметр _id генерируется автоматически как при обычном добавлении через функцию insert.

оператор \$set - обновление значения одного из ключей документа. Если документ не содержит обновляемое поле, то оно создается.

оператор \$unset - удаление отдельного ключа. Если подобного ключа в документе не существует, то оператор не оказывает никакого влияния. Можно удалять несколько полей.

updateOne() - обновить один документ

updateMany() - обновить все документы

Обновление массивов

Оператор `$push` - добавить еще один элемент в существующий массив

оператор `$position` - задает позицию в массиве для вставки элементов,

а оператор

оператор `$slice` - указывает, сколько элементов оставить в массиве

после вставки

оператор `$addToSet` - добавляет данные в массив, если их еще нет

`$pop` - удаляет элемент из массива

Удаление данных

`db.ИМЯКОЛЛЕКЦИИ.remove(КРИТЕРИЙУДАЛЕНИЯ [, justOne])` - удаление документов из коллекций

Параметры:

- критерий удаления - Критерий, по которому документ будет удалён
- justOne - Если значение параметра 'true' – удаляет только один документ

`db.ИМЯ_КОЛЛЕКЦИИ.drop()` - удаление коллекции

`db.dropDatabase()` - удаление БД

Установка ссылок в БД

В реляционных базах данных можно устанавливать внешние ключи, когда поля из одной таблицы ссылаются на поля в другой таблице. В MongoDB также можно устанавливать ссылки.

Пользовательская установка ссылок

Установка ссылок сводится к присвоению значения поля `_id` одного документа полю другого документа

```
db.companies.insert({"_id" : "microsoft", year: 1974});  
db.users.insert({name: "Tom", age: 28, company: "microsoft"});  
user = db.users.findOne();  
db.companies.findOne({_id: user.company})
```

```
C:\mongodb\bin\mongo.exe

MongoDB shell version: 3.0.3
connecting to: test
> db.companies.insert(<{"_id" : "microsoft", year: 1974}>)
WriteResult(< "nInserted" : 1 >)
> db.users.insert(<{name: "Tom", age: 28, company: "microsoft"}>)
WriteResult(< "nInserted" : 1 >)
> user = db.users.findOne()
{
  "_id" : ObjectId("556abb9aa6fc9ffd908901ff"),
  "name" : "Tom",
  "age" : 28,
  "company" : "microsoft"
}
> db.companies.findOne(<{"_id" : user.company}>)
{ "_id" : "microsoft", "year" : 1974 }
>
```

Автоматическое связывание

Используя функциональность DBRef

Для связывания с документов используется выражение:

```
{ "$ref" : название_коллекции, "  
id":значение\[, "  
db": название_бд ] }
```

\$ref указывает на коллекцию, где хранится связанный документ.

"\$id" указывает на значение, которое и будет представлять "внешний ключ".

"\$db" указывает на базу данных.

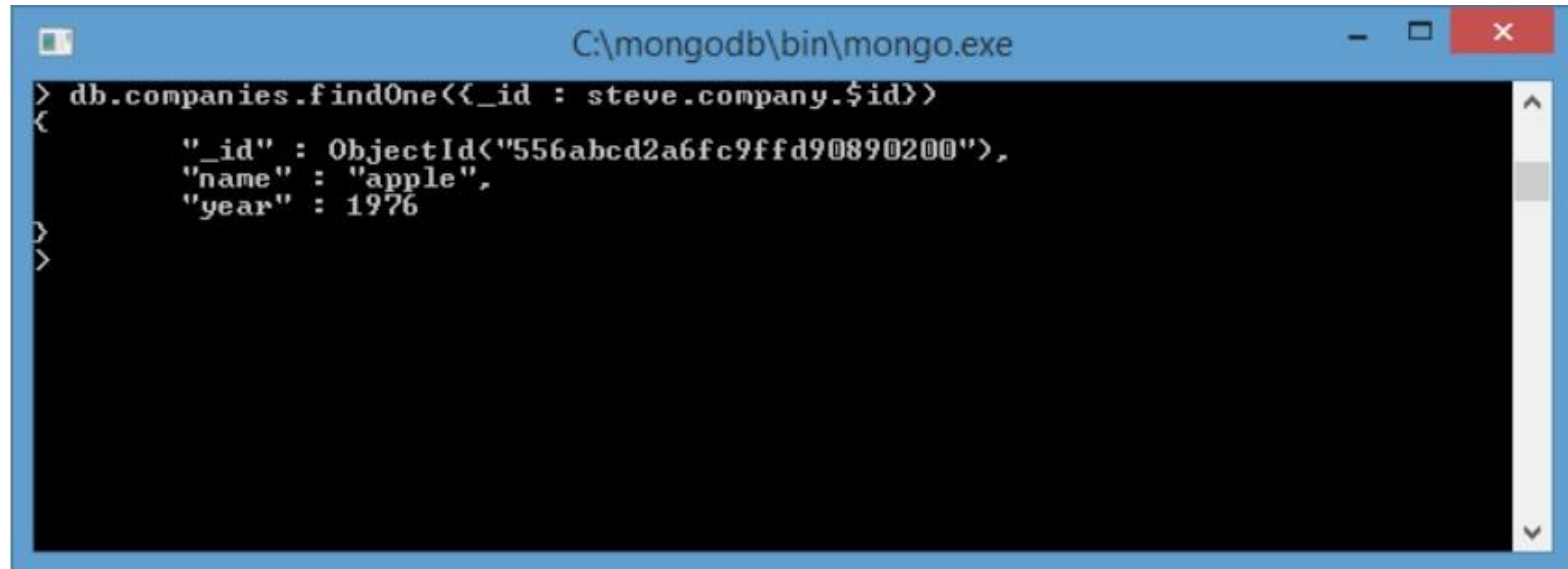
```
apple=({"name" : "apple", "year": 1976});
```

```
db.companies.save(apple);
```

```
steve = ({"name": "Steve", "age": 25, company: new DBRef('companies', apple._id)});
```

```
db.users.save(steve);
```

```
db.companies.findOne({_id: steve.company.$id});
```



A screenshot of a MongoDB Shell window titled "C:\mongodb\bin\mongo.exe". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main area is a black terminal with white text. The command entered is `db.companies.findOne(<_id : steve.company.$id>)`. The output is a JSON document: `{ "_id" : ObjectId<"556abcd2a6fc9ffd90890200">, "name" : "apple", "year" : 1976 }`. The cursor is on a new line below the output. A vertical scrollbar is on the right side of the terminal area.

```
> db.companies.findOne(<_id : steve.company.$id>)
{
  "_id" : ObjectId<"556abcd2a6fc9ffd90890200">,
  "name" : "apple",
  "year" : 1976
}
>
```




НЕТОЛОГИЯ
групп

Спасибо за
внимание!

Алексей Кузьмин



aleksej.kyzmin@gmail.com