Лабораторная работа 5

Задачи лабораторной работы:

* Установить MongoDB
* Рассмотреть различия в моделировании данных для реляционных БД и документоориентированных.
* Попрактиковаться в составлении запросов к MongoDB: добавление, обновление, селекция, фильтрация, агрегация.
* Настроить репликацию данных, инициировать отключение мастер-узла в процессе интенсивного обновления данных, проанализировать действия сервера и проверить целостность измененных данных.

**Содержание**

* 1 Установка MongoDB
* 2 Импорт данных
* 3 Экспорт данных
* 4 Моделирование данных в MongoDB
  + 4.1 Коллекция Movie
  + 4.2 Индексы
  + 4.3 Подокументное преобразование данных
  + 4.4 Дополнительно
* 5 Язык запросов MongoDB
  + 5.1 Базовые операции
    - 5.1.1 Выборка
    - 5.1.2 Обновление
    - 5.1.3 Дополнительно
  + 5.2 Aggregation Framework
    - 5.2.1 Дополнительно
* 6 Репликации MongoDB (необязательное)
  + 6.1 Принцип работы репликаций в MongoDB
  + 6.2 Конфигурация набора реплик
  + 6.3 Сценарий отказа мастер-узла
  + 6.4 Стартовый скрипт инициализации сбоя во время работы
  + 6.5 Дополнительно
* 7 Задания на защиту
  + 7.1 Защита лабораторной работы

Установка MongoDB

Для лабораторных работ достаточно версии 2.4+

* Ubuntu: sudo apt-get install mongodb
* Mac OS: brew install mongodb [Подробнее](http://www.mongodbspain.com/en/2014/11/06/install-mongodb-on-mac-os-x-yosemite/)
* Windows: Скачать установщик: <https://www.mongodb.com/download-center#community>

Удобная IDE для работы с MongoDB:

* <https://robomongo.org/>

Импорт данных

Коллекции, содержащие фильмы и длинные текстовые факты о фильмах:

* https://disk.yandex.ru/d/Lcef8VQo4Z27PA

Для импорта коллекций в базу movies выполните:

bzip2 -dc movies.bson.bz2 | mongoimport -d movies -c Movie

bzip2 -dc moviesdocs.bson.bz2 | mongoimport -d movies -c MovieDoc

Импортированная коллекция займет примерно **6Гб** на диске.

Чтобы проверить корректность импорта, зайдите в консоль MongoDB:

mongo

Показать базы данных:

show dbs;

Переключиться на базу данных (если ее нет, то создастся):

use movies

Показать коллекции текущей базы данных:

show collections;

Экспорт данных

В случае, если понадобится экспортировать данные, то сделать это можно (для версий 2.\*) по коллекциям, запуская:

mongoexport --db movies -c MovieDoc -o - | bzip2 - > moviesdocs.bson.bz2

mongoexport --db movies -c Movie -o - | bzip2 - > movies.bson.bz2

Если вы создадите индексы, то они будут лежать в отдельной коллекции, ее также нужно экспортировать.

Моделирование данных в MongoDB

Так как MongoDB не стремится экономить место на диске, а наоборот активно резервирует место под обновления данных, итоговая база занимает более 30Гб. В данной лабораторной работе рассматривается только сущность фильмов и связанные с ней факты (информация аналогичная хранимой в title, movie\_info, keywords).

Схема модели данных в MongoDB:

В дампе базы есть коллекции:

* Movie - для заголовков фильмов, ключевых слов и некоторых фактов
* MovieDoc - для длинных текстовых фактов (описания, цитаты и тд)

**Коллекция Movie**

Это основная коллекция, ключом в которой является отформатированная строка, состоящая из: названия фильма, года выпуска, названия эпизода (для сериалов). Пример поиска сериала и эпизода по ключу:

db.Movie.find({"\_id" : "\"12 oz. Mouse\" (2005)"}) // выбрать сериал или фильм

db.Movie.find({"SeriesID" : "\"12 oz. Mouse\" (2005)"}) // выбрать сериал и все эпизоды

db.Movie.find({"\_id" : "\"12 oz. Mouse\" (2005) {Adventure Mouse (#1.7)}"}) // выбрать конкретный эпизод

Для удобства восприятия информации используйте .pretty():

> db.Movie.find({"\_id" : "\"12 oz. Mouse\" (2005)"}).pretty()

{

"AltTitles" : [

"\"12 Ouns Mouce\" (2005)",

"\"Oz. Mo\" (2005)"

],

"Countries" : [

"USA"

],

"Genres" : [

"Animation",

"Comedy"

],

"Keywords" : [

"abbreviation-in-title",

"absurdism",

"adult-animation",

"animal-in-title",

"avant-garde",

"beer",

"character-name-in-title",

"digit-in-title",

"drunkenness",

"late-night",

"mouse",

"non-sequitur",

"number-in-title",

"period-in-title",

"surrealism"

],

"Languages" : [

"English"

],

"Locations" : [

"Atlanta, Georgia, USA"

],

"MovieID" : "\"12 oz. Mouse\" (2005)",

"Parental" : {

"Certificates" : [

"Australia:MA15+",

"Canada:14+\t(TV rating)",

"New Zealand:M",

"USA:TV-14",

"USA:TV-MA\t(one episode)"

]

},

"Rating" : {

"RatingDist" : "1000001103",

"Rating" : "6.9",

"RatingVotes" : "1365"

},

"ReleaseYear" : "2005",

"RunningTime" : "15",

"SeriesEndYear" : "",

"SeriesID" : "\"12 oz. Mouse\" (2005)",

"SeriesType" : "S",

"Technical" : {

"Colors" : [

"Color"

]

},

"\_id" : "\"12 oz. Mouse\" (2005)"

}

Расшифровка SeriesType:

* F - фильм
* S - сериал
* E - эпизод

**Индексы**

В предлагаемом дампе нет индексов (только первичный ключ \_id), в ходе выполнения заданий они могут понадобиться.

Просмотр существующих индексов в коллекции:

db.collection\_name.getIndexes()

Функция просмотра всех индексов для всех коллекций:

db.getCollectionNames().forEach(function(collection) {

indexes = db[collection].getIndexes();

print("Indexes for " + collection + ":");

printjson(indexes);

});

Создание индексов:

db.collection\_name.createIndex( { field\_name: 1 } )

db.collection\_name.createIndex( { field\_name: "hashed" } ) // хэш-индексы

**Подокументное преобразование данных**

Недостатком предлагаемой базы является то, что все атрибуты фильмов по-прежнему строковые, хотя некоторые, например RunningTime или Rating.RatingVotes содержат только числа. Это часто встречаемая ситуация, поэтому далее рассматривается пример функции, которая помогает выполнять подобные преобразования.

Конвертировать продолжительность фильма (задана для всех):

db.Movie.find().forEach(function(data) {

db.Movie.update({

"\_id": data.\_id

}, {

"$set": {

"RunningTime": parseInt(data.RunningTime)

}

});

});

Конвертировать рейтинг там, где он задан:

db.Movie.find({"Rating.Rating": {$exists: true}, "Rating.RatingVotes": {$exists: true}}).forEach(function(data) {

db.Movie.update({

"\_id": data.\_id

}, {

"$set": {

"Rating.Rating": parseFloat(data.Rating.Rating),

"Rating.RatingVotes": parseInt(data.Rating.RatingVotes)

}

});

})

**Задание:** Попробуйте добавить фильмам атрибут "длина названия", его нельзя вычислить с помощью обычной команды find, но можно сохранить эту информацию заранее. И также попробуйте преобразовать годы выхода сериалов для тех записей, где они указаны.

**Дополнительно**

* Подробное описание скрипта и модели данных: <https://www.packtpub.com/books/content/mongo-goes-to-the-movies>

Язык запросов MongoDB

**Базовые операции**

**Выборка**

Поиск по текстовым полям поддерживает синтаксис регулярных выражений, для нечеткого поиска нужно заключить выражение в /title/ (аналог "%title%"). Пример:

db.Movie.find({"MovieID": /Matrix.\*1999/})

Чтобы вывести только нужные поля, используйте второй параметр функции find:

> db.Movie.find({"\_id" : "The Matrix (1999)"}, {Genres: 1, Business: 1}).pretty()

Поиск по значениям в списках не отличается от обычного:

db.Movie.find({"Genres": "Action"}).limit(3)

**Обновление**

Добавить новый документ:

db.Movie.insert({\_id: "\"How I am getting 10 for db exam\" (2016)",

"MovieID" : "\"How I am getting 10 for db exam\" (2016)",

"Countries" : [ "Russia" ], "Genres" : [ "Drama", "Comedy" ],

"Keywords": [ "study", "working-hard", "success" ], "ReleaseYear" : 2016})

db.Movie.insert({\_id: "\"How I am getting 10 for project\" (2016)", "MovieID" :

"\"How I am getting 10 for db exam\" (2016)",

"Countries" : [ "Russia" ], "Genres" : [ "Drama", "Comedy" ],

"Keywords": [ "study", "working-hard", "success" ], "ReleaseYear" : 2016})

Обновить атрибут для всех документов по результатам поиска:

db.Movie.update({MovieID: /How I am getting 10 for/}, {$set: {"Locations" : [ "Moscow, Russia" ]}}, multi=true)

* Первый аргумент - условия выборки обновляемых объектов
* Второй аргумент - что именно делать
* multi=true (по умолчанию false), обновляет все найденные объекты или только первый

Добавить элемент в список документа:

db.Movie.update({MovieID: /How I am getting 10 for/}, {$push: {"Keywords": "true-story"}}, multi=true)

**Дополнительно**

* <https://docs.mongodb.com/manual/crud/>

**Aggregation Framework**

Для выполнения агрегирующих запросов в MongoDB используется команда aggregate.

Сама агрегация выглядит как pipeline для данных, проходя через шаги, данные преобразуются (группируются, фильтруются и тд) в зависимости от команды этого шага. Пример:

Также можно разгруппировать списки в документах ($unwind).

Чтобы отделять переменные внутри шагов от строк используется префикс $.

Найти документы и разгруппировать ($unwind) их по ключевым словам:

db.Movie.aggregate({$match: {MovieID: /How I am getting 10 for/}},

{$unwind: "$Keywords"})

Переформировать список выбираемых атрибутов ($project) аналогично второму параметру в find:

db.Movie.aggregate({$match: {MovieID: /How I am getting 10 for/}},

{$project: {"MovieID": 1, "Keywords": 1, "\_id": 0} },

{$unwind: "$Keywords"})

Сгруппировать по атрибуту и применить агрегирующую функцию:

db.Movie.aggregate({$match: {MovieID: /How I am getting 10 for/}},

{$project: {"MovieID": 1, "Keywords": 1, "\_id": 0} },

{$unwind: "$Keywords"},

{$group: {\_id: "$Keywords", "MoviesList": {$push: "$MovieID"} }})

**Дополнительно**

* <https://docs.mongodb.com/manual/aggregation/>

Репликации MongoDB (необязательное)

**Принцип работы репликаций в MongoDB**

**Конфигурация набора реплик**

**Сценарий отказа мастер-узла**

**Стартовый скрипт инициализации сбоя во время работы**

**Дополнительно**

* <https://docs.mongodb.com/manual/core/replication-introduction/>

Задания на защиту

1. Составить запросы:

* Для каждого года из первой декады XXI века посчитать количество снятых фильмов.
* Выбрать топ-10 наиболее популярных ключевых слов для фильмов заданной страны.

2. Составить свою формулу рейтинга (общего для всех фильмов или по какому-либо фильтру), используя параметры словарей Movie.Rating (Rating - среднее значение оценкой, RatingVotes - количество оценок) или любые другие.

Для примера вот формула [Top 250 IMDb Movies](http://www.imdb.com/chart/top)

ранг фильма = (v/(v+k))\*X + (k/(v+k))\*C

где:

X = рейтинг фильма

v = количество голосов

k = минимальное количество голосов, чтобы попасть в список (для данного списка 25000)

C = средний рейтинг для всех фильмов в этом списке (для данного списка 6.90)

MongoDB поддерживает многие арифметические операции: <https://docs.mongodb.com/manual/reference/operator/aggregation-arithmetic/>

Данное задание можно выполнить не одним запросом через агрегацию, а используя функцию на JavaScript, которая выполняется в консоле MongoDB.

3. (**необязательное**) Написать скрипт на bash или любом другом скриптовом языке, который запускает сервер MongoDB в режиме реплики, начинает интенсивно изменять данные, определяет мастер-узел, убивает процесс мастер-узла, после этого заканчивает изменения данных и проверяет целостность данных.

**Защита лабораторной работы**

* Показать и выполнить запросы с агрегацией, объяснить структуру запросов
* Продемонстрировать работу скрипта инициализации сбоя во время работы и объяснить, что происходит с репликами MongoDB