Лабораторная работа по теме «Нереляционные БД»

Цель работы: Знакомство с кроссплатформенной свободно распространяемой документно-ориентированной СУБД MongoDB

Задачи:

1. [Установка MongoDB](#Установка)
2. [Знакомство с взаимодействием СУБД](#Знакомство)
3. [Выполнить описанные задачи в работе](#Начало)

Вступление

В этой работе мы познакомимся с документо-ориентированной **СУБД MongoDB** (https://www.mongodb.org). Эта СУБД кроссплатформенная и относится к категории свободного программного обеспечения (Open Source). Для управления данными используется язык JavaScript.

Установка MongoDB

* *Для дальнейшей работы необходимо произвести установку MongoDB по* [*этой*](https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-18-04-ru) *(* [*https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-18-04-ru*](https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-18-04-ru) *) согласно инструкции.*

Знакомство с взаимодействием СУБД

Данные хранятся не в виде таблиц, а в виде коллекций документов. Документ – это объект, имеющий свойства, то есть пары «имя-значение». Главное свойство документа – это его идентификатор (код), который всегда играет роль первичного ключа.

В одной коллекции могут быть совершенно разные документы с разным набором свойств, это и отличает документо-ориентированную БД от реляционной.

Информация об объекте записывается в фигурных скобках, например,:

{ name: "Вася", age: 16 }

Этот объект имеет два свойства (поля) – свойство name со значением «Вася» и свойство age со значением 16. Такой текстовый формат записи называется JSON (англ. JavaScript Object Notation = запись объектов с помощью JavaScript).

Свойства объекта могут быть списками значений (массивами), они перечисляются в квадратных скобках через запятую:

{ name: "Вася", age: 16, lang: ["C", "Pascal", "JavaScript"] }

Здесь свойство lang – массив, в котором записаны названия языков программирования.

Свойства могут сами быть объектами со своими свойствами, например:

{

name: "Вася",

family: { mother: "Вера", father: "Петя" }

}

Свойство family содержит два внутренних свойства (поля): mother и father.

Начало работы

1. Запустите программу mongo в консольном режиме. Определите рабочую базу данных, выполнив команду db (от англ. database = база данных)

**db**

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Мы построим простую базу данных для блога в Интернете. Переключитесь на базу данных blog с помощью команды

use blog

Если этой базы данных раньше не было, она будет создана. Проверьте, что она действительно стала активной.

3. Для добавления объекта в коллекцию используется оператор insert (вставить):

db.posts.insert ( документ )

Первая часть записи, «db», означает обращение к рабочей базе данных; вторая, «posts» - название коллекции (если такой коллекции нет, она будет создана), в скобках записываются свойства объекта-документа. В нашем примере в базе blog будет одна коллекция posts (сообщения, записи, «посты»). Для каждого поста нужно задать дату и текст. Добавьте одну запись следующим образом[[1]](#footnote-1):

db.posts.insert (

{date: new Date("04/23/2013"), text: "Привет!"} )

Дата (свойство date) строится с помощью функции Date; ей передается символьная запись даты в формате, принятом в США (месяц/день/год).

4. Проверьте, создана ли коллекция posts с помощью команды, которая показывает все коллекции текущей базы:

show collections

5. Проверьте, добавлена ли запись, с помощью команды

db.posts.find()

которая ищет и показывает все документы, входящие в коллекцию posts. Для вывода в красивом формате можно дополнительно вызвать функцию pretty (англ. приятный)

db.posts.find().pretty()

В этом случае, если список свойств документа не помещается в одну строку, он выводится в столбик:

{

"\_id": ObjectId("5176abbc06a6380da34966a2"),

"date": ISODate("2013-04-22T20:00:00Z"),

"text": "Привет!"

}

СУБД автоматически добавила поле «\_id» (идентификатор, код), которое представляет собой суррогатный первичный ключ и строится случайным образом с помощью функции ObjectId. Дата преобразована в формат Международной организации по стандартизации (ISO).

6. Если нужно, значение ключа «\_id» можно задать самостоятельно, какое мы хотим. Удалим все документы из коллекции командой remove:

db.posts.remove()

и добавим четыре новых поста в базу, указав явно идентификаторы[[2]](#footnote-2):

db.posts.insert ( {\_id: 1, date: new Date("04/23/2013"), text: "Привет!" } )

db.posts.insert ( {\_id: 2, date: new Date("04/24/2013"), text: "Это второй пост." } )

db.posts.insert ( {\_id: 3, date: new Date("04/25/2013"), text: "Это третий пост." } )

db.posts.insert ( {\_id: 4, date: new Date("04/26/2013"), text: "Это четвёртый пост." } )

Убедитесь, что документы действительно добавлены в коллекцию.

Поиск

7. Для поиска нужных документов используется уже знакомая нам функция find. В скобках можно задать критерий поиска – это объект (записанный в фигурных скобках), который содержит название поля и значение этого поля, которое мы ищем. Например, найдите документ с кодом (идентификатором, \_id), равным 2:

db.posts.find( {\_id: 2} )

8. В условиях можно использовать не только строгие равенства, как в предыдущее примере, но неравенства. Неравенство – это тоже объект, у которого специальное название свойства, начинающееся знаком $:

$ne – не равно,

$lt – меньше, $lte – меньше или равно,

$gt – больше, $gte – больше или равно

Например, найдем записи с идентификатором больше 2:

db.posts.find( {\_id: {$gt: 2} } )

Сколько документов найдено?

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. Найдем посты, написанные 24.04.2013 или позднее:

db.posts.find({date: {$gte: new Date("04/24/2013")} })

10. Для поиска можно использовать сразу несколько условий. Если все условия нужно выполнить одновременно, их записывают как один объект с несколькими свойствами.

Например, следующий запрос находит все документы, у которых свойство «\_id» больше 2, а дата создания – не раньше 26.04.2013:

db.posts.find( { \_id: {$gt: 2},

date: {$gtе: new Date("04/26/2013")} } )

11. Самое мощное средство поиска в MongoDB – это оператор $where, которому можно передать строку в кавычках, задающую условие поиска на JavaScript, например,

db.posts.find( { $where: "this.\_id > 2" } )

Здесь this – это объект (документ), который требуется проверить; через точку записывается название нужного поля. Оператору $where можно передать любую функцию на JavaScript, которая возвращает логическое значение (истинное, если условие отбора выполняется). Например, запрос, показанный в п. 10, можно было записать так:

db.posts.find( { $where: function()

{ return this.\_id > 2 &&

this.date >= new Date("04/26/2013")

}

} )

Пара символов && в языке JavaScript обозначает логическую операцию «И», а символы || – логическую операцию «ИЛИ». Проверка на равенство записывается как «==», а условие «не равно» – как «!=».

12. Составьте запрос для поиска всех документов, у которых свойство \_id равно 1 или дата создания равна 26.04.2013:

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сортировка

13. Для сортировки используется функция sort. Она сортирует те документы, которые предварительно найдены с помощью find. При вызове функции sort в скобках указывается порядок сортировки – объект (в фигурных скобках), содержащий название поля для сортировки; значение этого поля может быть 1 (сортировка по возрастанию) или «-1» (по убыванию).

Отсортируйте документы по возрастанию даты

db.posts.find().sort( {date: 1} )

а теперь по убыванию:

db.posts.find().sort( {date: -1} )

Изменение

14. Для изменения документов используют команду update. Допустим, мы хотим добавить к документу с идентификатором 1 новое логическое свойство visible (англ. видимый) и присвоить ему значение false (ложь), которое означает, что это пост пока скрыт и выводить его на веб-страницу не нужно.

Функции update передаются два объекта в фигурных скобках: условие, позволяющее найти нужный документ, и свойства, которые у него нужно изменить. Попробуйте выполнить команду

db.posts.update( {\_id: 1}, {visible: false} )

и посмотрите, что получилось в результате.

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

15. Восстановите исходный пост с кодом 1.

16. Для того, чтобы не заменять документ полностью, а изменить (или добавить) значение какого-то поля, нужно использовать специальный объект со свойством $set (установить):

db.posts.update( {\_id: 1}, {$set: {visible: false}} )

Примените эту команду и проверьте результат её выполнения.

Обратите внимание, что сейчас в коллекции posts находятся документы с разной структурой: один из них имеет свойство visible, а остальные – нет. При этом никаких изменений в структуру базы данных вносить не пришлось.

17. Найдите все документы, у которых свойство visible равно false. Какой запрос нужно для этого выполнить?

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

18. Теперь выполним множественное обновление: установим свойство visible равным true (истина) для всех документов, для которых это свойство не установлено (или, что то же самое, равно специальному нулевому значению null):

db.posts.update( {visible: null},

{$set: {visible: true}}, {multi: true} )

Третий параметр – объект со свойством multi (англ. множественный), равным true (истина) разрешает изменение нескольких документов сразу (если его не указать, будет изменён только один документ – тот, который найден первым).

19. Составьте запрос для поиска всех документов, у которых свойство visible равно true, а дата создания – не позднее 25.04.2013:

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверьте результат его работы.

20. Теперь добавим к комментарии к одному из постов (с идентификатором 2). Заметьте, что благодаря документо-ориентированной СУБД заранее планировать наличие комментариев не нужно – мы можем добавлять новые свойства к любому документу «на ходу».

Комментариев может быть много, поэтому новое свойство comments будет массивом. Для добавления нового элемента в массив используется специальный объект со свойство $push (англ. втолкнуть):

db.posts.update( {\_id: 2},

{$push: {comments: "Комментарий 1"} })

db.posts.update( {\_id: 2},

{$push: {comments: "Комментарий 2"} })

db.posts.update( {\_id: 2},

{$push: {comments: "Комментарий 3"} })

Когда выполняется первая из этих команд, у документа с идентификатором 2 еще нет свойства comments – оно будет создано автоматически.

21. Все документы с комментариями можно найти с помощью объекта со свойством $exists (существует):

var c = db.posts.find( {comments: {$exists: true}} )

Результат этого запроса не выводится на экран, а записывается в переменную c. Затем можно определить его длину (число найденных документов) с помощью функции length (англ. длина) и вывести на экран в цикле в формате JSON с помощью функции printjson:

for(i = 0; i < c.length(); i++) printjson( c[i] )

Можно работать и с отдельными свойствами, например, вывести на экран даты всех найденных постов с помощью функции print:

for(i = 0; i < c.length(); i++) print( c[i].date )

Удаление

22. Как вы уже знаете, для удаления всех документов из коллекции используется команда remove. Запишите команду, которая удаляет все документы из коллекции posts, но не выполняйте её.

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

23. С помощью команды remove можно удалять отдельные записи – условие для поиска нужных записей задается как параметр функции. Например:

db.posts.remove( {\_id: 4} )

Выполните эту команду и проверьте её выполнение.

24. Запишите команду для удаления всех документов, у которых свойство visible установлено в false выполните её.

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

25. С помощью команды drop вся коллекция удаляется из базы. Например,

db.posts.drop()

26. Объясните, в чём отличие между командами remove() и drop().

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

27. Проверьте, что коллекция posts действительно была удалена. Какую команду нужно для этого использовать?

Ответ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дополнительная информация:

1. https://www.mongodb.com/ – официальный сайт разработки MongoDB.

2. https://www.mongodb.com/docs/manual/installation/ – инструкция по установке MongoDB для разных операционных систем.

3. https://huncode.com/js/mongo-book/ru – русский перевод книги K. Seguin «The Little MongoDB Book».

**Установка MongoDB**

**Ubuntu**

**Установка пакета**

1. В Терминале введите следующую команду для импорта публичного GPG-ключа (http://docs.mongodb.org/10gen-gpg-key.asc):

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10

2. Создайте файл /etc/apt/sources.list.d/10gen.list и добавьте в него строку для обращения к репозиторию компании 10gen:

deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen

3. Перегрузите список репозиториев командой

sudo apt-get update

4. Установите последнюю версию пакета командой

sudo apt-get install mongodb-10gen

**Запуск программы**

5. Запустите серверную часть mongod как службу командой

sudo service mongodb start

6. Запустите командную оболочку (консоль) командой

mongo

1. Если вы работаете в операционной системе Windows, не используйте русские буквы в значениях свойств. Это связано с тем, что в MongoDB для хранения символьных строк используется кодировка UTF-8, которая не очень хорошо поддерживается в консоли Windows. [↑](#footnote-ref-1)
2. Можно создать в рабочем (текущем) каталоге текстовый файл с расширением .js (JavaScript), включающий эти команды, и загрузить его с помощью команды load, например:  
   load('data.js') [↑](#footnote-ref-2)