ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий
Кафедра “Прикладная математика”

Отчет по лабораторной работе курса “Базы данных”

Лабораторная работа №6
«NoSQL-MongoDB»

Выполнил студент группы 18-ПМ:
Винокуров М.С
Проверил:
Моисеев А.Е.

НИЖНИЙ НОВГОРОД

2021 г.

Оглавление

Введение

MongoDB

MongoDB (от англ. humongous — огромный) —документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++.

Система поддерживает ad-hoc-запросы: они могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов.

Имеется поддержка индексов.

Система может работать с набором реплик, то есть, содержать две или более копии данных на различных узлах. Каждый экземпляр набора реплик может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой. Вспомогательные реплики поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, набор реплик проводит выбор, которая из реплик должна стать основной. Второстепенные реплики могут дополнительно являться источником для операций чтения.

Система масштабируется горизонтально, используя технику сегментирования (англ. sharding) объектов баз данных — распределение их частей по различным узлам кластера. Администратор выбирает ключ сегментирования, который определяет, по какому критерию данные будут разнесены по узлам (в зависимости от значений хэша ключа сегментирования). Благодаря тому, что каждый узел кластера может принимать запросы, обеспечивается балансировка нагрузки.

Система может быть использована в качестве файлового хранилища с балансировкой нагрузки и репликацией данных (функция Grid File System; поставляется вместе с драйверами MongoDB). Предоставляются программные средства для работы с файлами и их содержимым. GridFS используется в плагинах для Nginx и lighttpd. GridFS разделяет файл на части и хранит каждую часть как отдельный документ.

Может работать в соответствии с парадигмой MapReduce. Во фреймворке для агрегации есть аналог SQL-инструкции GROUP BY. Операторы агрегации могут быть связаны в конвейер подобно UNIX-конвейрам. Фреймворк так же имеет оператор $lookup для связки документов при выгрузке и статистические операции такие как среднеквадратическое отклонение.

Поддерживается JavaScript в запросах, функциях агрегации (например, в MapReduce).

MongoDB поддерживает коллекции с фиксированным размером. Такие коллекции сохраняют порядок вставки и по достижении заданного размера ведут себя как кольцевой буфер.

В июне 2018 года (в версии 4.0) добавлена поддержка транзакций, удовлетворяющих требованиям ACID.

У MongoDB есть официальные драйверы для основных языков программирования (Си, C++, C#, Erlang, Go, Haskell, J#, Java, JavaScript, Lisp, Perl, PHP, Python, Ruby, Delphi, Scala). Существует также большое количество неофициальных или поддерживаемых сообществом драйверов для других языков программирования и фреймворков.

Основным интерфейсом к базе данных была командная оболочка «mongo». С версии MongoDB 3.2 в качестве графической оболочки поставляется «MongoDB Compass». Существуют продукты и сторонние проекты, которые предлагают инструменты с GUI для администрирования и просмотра данных.

Задание

- Создать базу данных MongoDB из 20-ти объектов
- Вывести на веб-страницу содержимое базы данных:
    - На странице кроме таблицы с содержимом базы данных находятся поля для ввода фильтров
    - Содержимое полей отправляется на сервер
    - Сервер на основе содержимого фильтров формирует запрос к базе данных
    - Общение между клиентом и серверов в формате JSON
    - Таблица формируется на стороне клиента

Выполнение работы

Был создан проект Node.JS.
В качестве объектов хранящихся в базе данных выступают «игры».

В начале программы подключаются необходимые модули

```
var fs = require("fs")
var http = require("http");
var mongodb = require("mongodb");
var escape = require("mongo-escape")
```

Выполняется соединение с базой данных

```
var client = mongodb.MongoClient("mongodb://localhost:27017", { useUnifiedTopolog
y: true });
```

Делается обращение к базе данных, для проверки наличия коллекции

```
client.connect(function (err, client) {
    if (err) {
        console.error(err);
        return;
    }
    var db = client.db('GameDB');
```

В случае её отсутствия выполняется создание и заполнение коллекции

```
if (cols.length == 0) {
            var collection = db.collection("GameCollection");
            var games = [
                {
                    game: "Fallout",
                    platform: "Steam", price: 24, release_date: 1994
                }
```

фрагмент кода заполнения

После этого выполняется создание сервера и вывод сообщения в консоль

```
http.createServer(server_callback).listen(3000);
console.log("Listen at http://localhost:3000/");
```

В функции обработки запроса к серверу

```
var server_callback = function (request, response) {
    console.log("request to: " + request.url + " method: " + request.method)
    if (request.method == "GET") {
        handle_GET(request, response);
    } else {
        handle_POST(request, response);
    }
}
```

Разделяются POST и GET запросы к серверу
В обработчике GET запросов

```
var handle_GET = function (request, response) {
    switch (request.url) {
        case "/":
            fs.readFile("./db6.html", function (err, content) {
                if (!err) {
                    response.writeHead(200, { "Content-
Type": "text/html; charset=utf-8" });
                    response.end(content, "utf-8")
                } else {
                    response.writeHead(500, { "Content-
Type": "text/plain; charset=utf-8" });
                    response.end(err.message, "utf-8");
                    console.log(err);
                }
            });
            break;
        default:
            response.writeHead(404, { "Content-Type": "text/html; charset=utf-
8" });
            response.end("<!DOCTYPE html>\n" +
                "<html>\n" +
                "    <head>\n" +
                "        <meta charset='utf-8'>\n" +
                "    </head>\n" +
                "    <body>\n" +
                "404, NOT FOUND: " + request.url +
                " \n</body>\n" +
                "</html>"
            );
    }
}
```

Клиенту отправляется либо содержимое файла db6.html, либо сообщение об ошибке.

В обработчике POST запросов

```
var handle_POST = function (request, response) {
    if (request.url != "/get_table") {
        response.writeHead(500, { "Content-Type": "text/plain; charset=utf-8" });
        response.end();
    }

    var data = '';
    request.on('data', function (chunk) {
        data += chunk;
    });
    request.on('end', function () {
        var filters = JSON.parse(data);
```

формируется полученный от клиента объект в формате JSON и котором хранятся значения фильтров

```
 var db_data = {};
        var collection = client.db('GameDB').collection("GameCollection");
        collection.find({
            game: { $regex: escape.escape(filters.game), $options: 'i' },

            price: { $gte: parseInt(filters.price_from), $lte: parseInt(filters.p
rice_to) }
        }).toArray(function (err, res) {
            if (err) {
                console.error(err);
                return;
            }
            db_data.table = res;
            response.end(JSON.stringify(db_data));
        })
```

Выполняется запрос к MongoDB (используется сторонний модуль NodeJS для защиты от NoSQL-инъекций, так как в драйвере MongoDB такой возможности не предусмотрено) и в случае успеха результат оправляется клиенту в формате JSON, в случае ошибки отправляет сообщение об ошибке.

Так же была создана веб-страница, в теле которой

```
<body>
    <p>
        GameDB_4 from MongoDB.
    </p>
    <p>
        Game Search<input id="game" type="text"> Game Price from: <input id="from
"
            type="number" value=0> to: <input id="to" type="number" value=100>$.
    </p>
    <p>
        <span id="res" style="font-style: italic"></span>
    </p>
</body>
```

Находятся 4 поля для ввода – фильтры и поле, в котором будет находится таблица с результатом запроса к серверу
При загрузке браузером страницы

```
window.onload = function () {
            var game_input = document.getElementById("game");
            game_input.oninput = function () {
                game_filter = game_input.value;
                update();
            };
            var fr_input = document.getElementById("from");
            fr_input.oninput = function () {
                price_filter_from = fr_input.value;
                update();
            };
            var to_input = document.getElementById("to");
            to_input.oninput = function () {
                price_filter_to = to_input.value;
                update();
            };
            update();
```

Устанавливаются обработчики изменения значений в полях ввода – при их изменении вызывается функция отправки серверу новых значений и изменения содержимого таблицы

```
function update() {
            var filters = {};
            filters.game = game_filter;
            filters.price_from = price_filter_from;
            filters.price_to = price_filter_to;
            readServer("/get_table", JSON.stringify(filters), function (err, resp
onse) {
                if (err) document.getElementById("res").innerHTML = err;
                else {
                    var temp = "";
                    temp = "<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n";
                    var rows = JSON.parse(response).table;
                    for (var i = 0; i < rows.length; i++) {
                        temp += "<tr><td>" + rows[i].game + "</td><td align=\"cen
ter\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date +
"</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n";
                    }
                    temp += "</table>";
                    document.getElementById("res").innerHTML = temp;
                }
            });
        }
```

В ней используется функция, отправляющая серверу фоновый запрос по технологии AJAX

```
function readServer(url, data, callback) {
            var req = new XMLHttpRequest();
            req.onreadystatechange = function () {
                if (req.readyState === 4) {//"loaded"
                    if (req.status === 200) {//"OK"
                        callback(undefined, req.responseText);
                    } else {
                        callback(new Error(req.status));
                    }
                }
            };

            req.open("POST", url, true);
            req.setRequestHeader('Content-Type', 'application/json');
            req.send(data);
        }
```

Эта функция принимает на вход URL сервера(можно относительный), данные для отправки в теле запроса и функцию – обработчик результата.

После загрузки страницы:

GameDB from MongoDB.

Game Search Game Price from: 0 to: 100 $.

Fallout	*24$*	*1994*	*Steam*
Fallout 2	*29$*	*1996*	*Steam*
Fallout Nevada	*0$*	*2003*	*non*
Fallout New Vegas	*40$*	*2009*	*Steam*
Halo Combat Evolved	*60$*	*2003*	*Xbox Live*
Halo 2	*60$*	*2005*	*Xbox Live*
Dota 2	*0$*	*2011*	*Steam*
League of legends	*0$*	*2009*	*RIOT Launcher*
Counter Strike - Condition Zero	*25$*	*2005*	*Steam*
Call of duty 3	*45$*	*2006*	*Xbox Live*
Horizon Zero Dawn	*60$*	*2017*	*PS Network*
The last of Us	*60$*	*2013*	*PS Network*
Fallout 76	*50$*	*2018*	*Bethesda Launcher*
Halo 5	*60$*	*2015*	*Xbox Live*
Stubbs the Zombie	*45$*	*2004*	*Xbox Live*
Battlefield 1942	*55$*	*2003*	*Retail*
Team Fortress 2	*5$*	*2008*	*Steam*
Half-Life	*45$*	*1999*	*Retail*
Half-Life 2	*60$*	*2004*	*Steam*

После ввода значений в поля фильтров:

GameDB from MongoDB.

Game Search 2 Game Price from: 2 to: 60 $.

Fallout 2	*29$*	*1996*	*Steam*
Halo 2	*60$*	*2005*	*Xbox Live*
Battlefield 1942	*55$*	*2003*	*Retail*
Team Fortress 2	*5$*	*2008*	*Steam*
Half-Life 2	*60$*	*2004*	*Steam*

Вывод

В ходе выполнения работы был создан веб-сервер и веб-страница, которые взаимодействуют с помощью AJAX. Веб-сервер взаимодействует с базой данных и отправляет данные клиенту, на стороне клиента формируется таблица с содержимым базы данных и отправляются содержащиеся в полях ввода – фильтрах значения на сервер.

Список источников

1. Статья о MongoDB в Википедии - https://ru.wikipedia.org/wiki/MongoDB
2. Документация MongoDB - https://docs.mongodb.com

Приложение. Листинг.

GameDB-6.js

```
var fs = require("fs")
var http = require("http");
var mongodb = require("mongodb");
var escape = require("mongo-escape")

var client = mongodb.MongoClient("mongodb://localhost:27017", { useUnifiedTopolog
y: true });

var handle_GET = function (request, response) {
    switch (request.url) {
        case "/":
            fs.readFile("./db6.html", function (err, content) {
                if (!err) {
                    response.writeHead(200, { "Content-
Type": "text/html; charset=utf-8" });
                    response.end(content, "utf-8")
                } else {
                    response.writeHead(500, { "Content-
Type": "text/plain; charset=utf-8" });
                    response.end(err.message, "utf-8");
                    console.log(err);
                }
            });
            break;
        default:
            response.writeHead(404, { "Content-Type": "text/html; charset=utf-
8" });
            response.end("<!DOCTYPE html>\n" +
                "<html>\n" +
                "   <head>\n" +
                "       <meta charset='utf-8'>\n" +
                "   </head>\n" +
                "   <body>\n" +
                "404, NOT FOUND: " + request.url +
                " \n</body>\n" +
                "</html>"
            );
    }
}

var handle_POST = function (request, response) {
    if (request.url != "/get_table") {
        response.writeHead(500, { "Content-Type": "text/plain; charset=utf-8" });
        response.end();
    }
```

```
    var data = '';
    request.on('data', function (chunk) {
        data += chunk;
    });
    request.on('end', function () {
        var filters = JSON.parse(data);
        var db_data = {};
        var collection = client.db('GameDB').collection("GameCollection");
        collection.find({
            game: { $regex: escape.escape(filters.game), $options: 'i' },

            price: { $gte: parseInt(filters.price_from), $lte: parseInt(filters.p
rice_to) }
        }).toArray(function (err, res) {
            if (err) {
                console.error(err);
                return;
            }
            db_data.table = res;
            response.end(JSON.stringify(db_data));
        })
    });
}

var server_callback = function (request, response) {
    console.log("request to: " + request.url + " method: " + request.method)
    if (request.method == "GET") {
        handle_GET(request, response);
    } else {
        handle_POST(request, response);
    }
}

client.connect(function (err, client) {
    if (err) {
        console.error(err);
        return;
    }
    var db = client.db('GameDB');
    db.collections(function (err, cols) {
        if (err) {
            console.error(err);
            return;
        }
        if (cols.length == 0) {
            var collection = db.collection("GameCollection");
            var games = [
                {
                    game: "Fallout",
                    platform: "Steam", price: 24, release_date: 1994
                },
```

```
{
    game: "Fallout 2",
    platform: "Steam", price: 29, release_date: 1996
},
{
    game: "Fallout Nevada",
    platform: "non", price: 0, release_date: 2003
},
{
    game: "Fallout New Vegas",
    platform: "Steam", price: 40, release_date: 2009
},
{
    game: "Halo Combat Evolved",
    platform: "Xbox Live", price: 60, release_date: 2003
},
{
    game: "Halo 2",
    platform: "Xbox Live", price: 60, release_date: 2005
},
{
    game: "Dota 2",
    platform: "Steam", price: 0, release_date: 2011
},
{
    game: "League of legends",
    platform: "RIOT Launcher", price: 0, release_date: 2009
},
{
    game: "Counter Strike - Condition Zero",
    platform: "Steam", price: 25, release_date: 2005
},
{
    game: "Call of duty 3",
    platform: "Xbox Live", price: 45, release_date: 2006
},
{
    game: "Horizon Zero Dawn",
    platform: "PS Network", price: 60, release_date: 2017
},
{
    game: "The last of Us",
    platform: "PS Network", price: 60, release_date: 2013
},
{
    game: "Fallout 76",
    platform: "Bethesda Launcher", price: 50, release_date: 2018
},
{
    game: "Halo 5",
    platform: "Xbox Live", price: 60, release_date: 2015
```

```
                },
                {
                    game: "Stubbs the Zombie",
                    platform: "Xbox Live", price: 45, release_date: 2004
                },
                {
                    game: "Battlefield 1942",
                    platform: "Retail", price: 55, release_date: 2003
                },
                {
                    game: "Team Fortress 2",
                    platform: "Steam", price: 5, release_date: 2008
                },
                {
                    game: "Half-Life",
                    platform: "Retail", price: 45, release_date: 1999
                },
                {
                    game: "Half-Life 2",
                    platform: "Steam", price: 60, release_date: 2004
                },
                {
                    game: "Half-life 3",
                    platform: "Steam", price: 120, release_date: 2077
                }
            ]
            collection.insertMany(games, function (err, res) {
                if (err) {
                    console.error(err);
                    return;
                }
            })
        }
        http.createServer(server_callback).listen(3000);
        console.log("Listen at http://localhost:3000/");
    });
});
```

db6.html

```
<!DOCTYPE html>
<html>

<head>
    <title>NodeJS+AJAX</title>
    <meta charset="utf-8" />
    <script type="text/javascript">
        var game_filter = "";
        var price_filter_from = 0;
        var price_filter_to = 100;
```

```
        function readServer(url, data, callback) {
            var req = new XMLHttpRequest();
            req.onreadystatechange = function () {
                if (req.readyState === 4) {//"loaded"
                    if (req.status === 200) {//"OK"
                        callback(undefined, req.responseText);
                    } else {
                        callback(new Error(req.status));
                    }
                }
            };

            req.open("POST", url, true);
            req.setRequestHeader('Content-Type', 'application/json');
            req.send(data);
        }
        function update() {
            var filters = {};
            filters.game = game_filter;
            filters.price_from = price_filter_from;
            filters.price_to = price_filter_to;
            readServer("/get_table", JSON.stringify(filters), function (err, resp
onse) {
                if (err) document.getElementById("res").innerHTML = err;
                else {
                    var temp = "";
                    temp = "<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n";
                    var rows = JSON.parse(response).table;
                    for (var i = 0; i < rows.length; i++) {
                        temp += "<tr><td>" + rows[i].game + "</td><td align=\"cen
ter\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date +
"</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n";
                    }
                    temp += "</table>";
                    document.getElementById("res").innerHTML = temp;
                }
            });
        }

        window.onload = function () {
            var game_input = document.getElementById("game");
            game_input.oninput = function () {
                game_filter = game_input.value;
                update();
            };
            var fr_input = document.getElementById("from");
            fr_input.oninput = function () {
                price_filter_from = fr_input.value;
                update();
```

```
            };
            var to_input = document.getElementById("to");
            to_input.oninput = function () {
                price_filter_to = to_input.value;
                update();
            };
            update();
        }
    </script>
</head>

<body>
    <p>
        GameDB from MongoDB.
    </p>
    <p>
        Game Search<input id="game" type="text"> Game Price from: <input id="from
"
            type="number" value=0> to: <input id="to" type="number" value=100>$.
    </p>
    <p>
        <span id="res" style="font-style: italic"></span>
    </p>
</body>

</html>
```