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 №7
«NoSQL-Neo4j»

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

НИЖНИЙ НОВГОРОД

2021 г.

Оглавление

Введение

Noe4j

Neo4j — графовая система управления базами данных с открытым исходным кодом, реализованная на Java. По состоянию на 2015 год считается самой распространённой графовой СУБД. Разработчик — американская компания Neo Technology, разработка ведётся с 2003 года.

Данные хранит в собственном формате, специализированно приспособленном для представления графовой информации, такой подход в сравнении с моделированием графовой базы данных средствами реляционной СУБД позволяет применять дополнительную оптимизацию в случае данных с более сложной структурой. Также утверждается о наличии специальных оптимизаций для SSD-накопителей, при этом для обработки графа не требуется его помещение целиком в оперативную память вычислительного узла, таким образом, возможна обработка достаточно больших графов.

Основные транзакционные возможности — поддержка ACID и соответствие спецификациям JTA, JTS и XA. Интерфейс программирования приложений для СУБД реализован для многих языков программирования, включая Java, Python, Clojure, Ruby, PHP, также реализовано API в стиле REST. Расширить программный интерфейс можно как с помощью серверных плагинов, так и с помощью неуправляемых расширений (unmanaged extensions); плагины могут добавлять новые ресурсы к REST-интерфейсу для конечных пользователей, а расширения позволяют получить полный контроль над программным интерфейсом, и могут содержать произвольный код, поэтому их следует использовать с осторожностью.

В СУБД используется собственный язык запросов — Cypher, но запросы можно делать и другими способами, например, напрямую через Java API и на языке Gremlin[en], созданном в проекте с открытым исходным кодом TinkerPop. Cypher является не только языком запросов, но и языком манипулирования данными, так как предоставляет функции CRUD для графового хранилища.

Задание

- Создать базу данных Neo4j из 20-ти объектов, дополнительно создать несколько других объектов и создать связи между ними
- 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");
const neo4j = require('neo4j-driver');
```

Выполняется соединение с базой данных

```
const driver= neo4j.driver("bolt://localhost",neo4j.auth.basic("neo4j", "pass"));
const session = driver.session();
```

Делается обращение к базе данных и заполняется база данных

```
session.run("MATCH (n) RETURN n").then(res => {
    if (res.records.length == 0) {
        session.run(
            "CREATE (device1:device $device1) " +
            "CREATE (device2:device $device2) " +
            "CREATE (device3:device $device3) " +
            "CREATE (game1:gamename $game1) " +
            "CREATE (game2:gamename $game2) " +
            "CREATE (game3:gamename $game3) " +
            "CREATE (game4:gamename $game4) " +
            "CREATE (game5:gamename $game5) " +
            "CREATE (game6:gamename $game6) " +
            "CREATE (game7:gamename $game7) " +
            "CREATE (game8:gamename $game8) " +
            "CREATE (game9:gamename $game9) " +
            "CREATE (game10:gamename $game10) " +
            "CREATE (game11:gamename $game11) " +
            "CREATE (game12:gamename $game12) " +
            "CREATE (game13:gamename $game13) " +
            "CREATE (game14:gamename $game14) " +
            "CREATE (game15:gamename $game15) " +
            "CREATE (game16:gamename $game16) " +
            "CREATE (game17:gamename $game17) " +
            "CREATE (game18:gamename $game18) " +
            "CREATE (game19:gamename $game19) " +
            "CREATE (game20:gamename $game20) " +
            "CREATE" +
            "(game1)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game2)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game3)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game4)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game5)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game6)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game7)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game8)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
```

```
            "(game9)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game10)-[:MENTIONED_CHARACTERS {chars: []}]->(device2)," +
            "(game11)-[:MENTIONED_CHARACTERS {chars: []}]->(device3)," +
            "(game12)-[:MENTIONED_CHARACTERS {chars: []}]->(device3)," +
            "(game13)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game14)-[:MENTIONED_CHARACTERS {chars: []}]->(device2)," +
            "(game15)-[:MENTIONED_CHARACTERS {chars: []}]->(device2)," +
            "(game16)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game17)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game18)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game19)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game20)-[:MENTIONED_CHARACTERS {chars: []}]->(device1);",
            {
                device1: { title: "PC", company: "null" },
                device2: { title: "Xbox", company: "Microsoft" },
                device3: { title: "PlayStation", company: "Sony" },
                game1: {
                    gamename: "Fallout",
                    platform: "Bethesda Launcher", price: 35, release_date: 1994
                },
```

фрагмент кода заполнения

После этого выполняется публикация сообщения о готовности к созданию сервера

```
}).then(res => {
                process.emit('readyToServerCreate');
```

По этому сигналу происходит создание сервера и вывод сообщения в консоль

```
process.on('readyToServerCreate', () => {
    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("./db7.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 db7.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

```
    request.on('end', function () {
        var filters = JSON.parse(data);
        var db_data = {};
        console.log(filters)
        session.run("MATCH (game:gamename)-[:MENTIONED_CHARACTERS]-
>(device) WHERE "+
        "game.gamename CONTAINS $title AND game.price>=$from AND game.price<=$to
"+
        "RETURN game, device;",
        {
            title: filters.gamename,
            from: parseInt(filters.price_from),
            to: parseInt(filters.price_to)
        }).then(res => {
            var table = [];
            res.records.forEach(rec => {
                var row = {};
                row.gamename = rec.get('game').properties.gamename;
                row.platform = rec.get('game').properties.platform;
                row.price = rec.get('game').properties.price;
                row.device_title = rec.get('device').properties.title;
                row.release_date = rec.get('game').properties.release_date;
                table.push(row);
            });
            db_data.table = table;
            console.log(db_data)
            response.end(JSON.stringify(db_data));
        }).catch(err => {
            console.error(err);
        });
    });
```

Выполняется запрос к Neo4j 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 from Neo4j.
    </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.gamename = 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].gamename + "</td><td align=\
"center\">" + rows[i].platform + "</td><td align=\"center\">" + rows[i].price + "
$</td><td align=\"center\">" + rows[i].device_title + "</td><td align=\"center\">
" + rows[i].release_date +  "</td></tr>\n";
                        console.log(rows[i])
                    }
                    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 Neo4j.

Game Search Game Price from: 0 to: 120 $.

Fallout	*Bethesda Launcher*	*35$*	*PC*	*1994*
Fallout 2	*Bethesda Launcher*	*30$*	*PC*	*1996*
Fallout Nevada	*non*	*0$*	*PC*	*2003*
Fallout New Vegas	*Steam*	*40$*	*PC*	*2009*
Halo Combat Evolved	*Xbox Live*	*60$*	*PC*	*2003*
Halo 2	*Xbox Live*	*60$*	*PC*	*2005*
Dota 2	*Steam*	*0$*	*PC*	*2011*
League of legends	*RIOT Launcher*	*0$*	*PC*	*2009*
Counter Strike - Condition Zero	*Steam*	*25$*	*PC*	*2004*
Call of duty 3	*Xbox Live*	*45$*	*Xbox*	*2005*
Horizon Zero Dawn	*PS Network*	*60$*	*PlayStation*	*2017*
The last of Us	*PS Network*	*60$*	*PlayStation*	*2013*
Fallout 76	*Bethesda Launcher*	*50$*	*PC*	*2018*
Halo 5	*Xbox Live*	*60$*	*Xbox*	*2015*
Stubbs the Zombie	*Xbox Live*	*45$*	*Xbox*	*2004*
Battlefield 1942	*Retail*	*55$*	*PC*	*2002*
Team Fortress 2	*Steam*	*5$*	*PC*	*2008*
Half-Life	*Retail*	*45$*	*PC*	*1999*
Half-Life 2	*Steam*	*60$*	*PC*	*2004*
Half-life 3	*CyberPunkedSteam*	*120$*	*PC*	*2077*

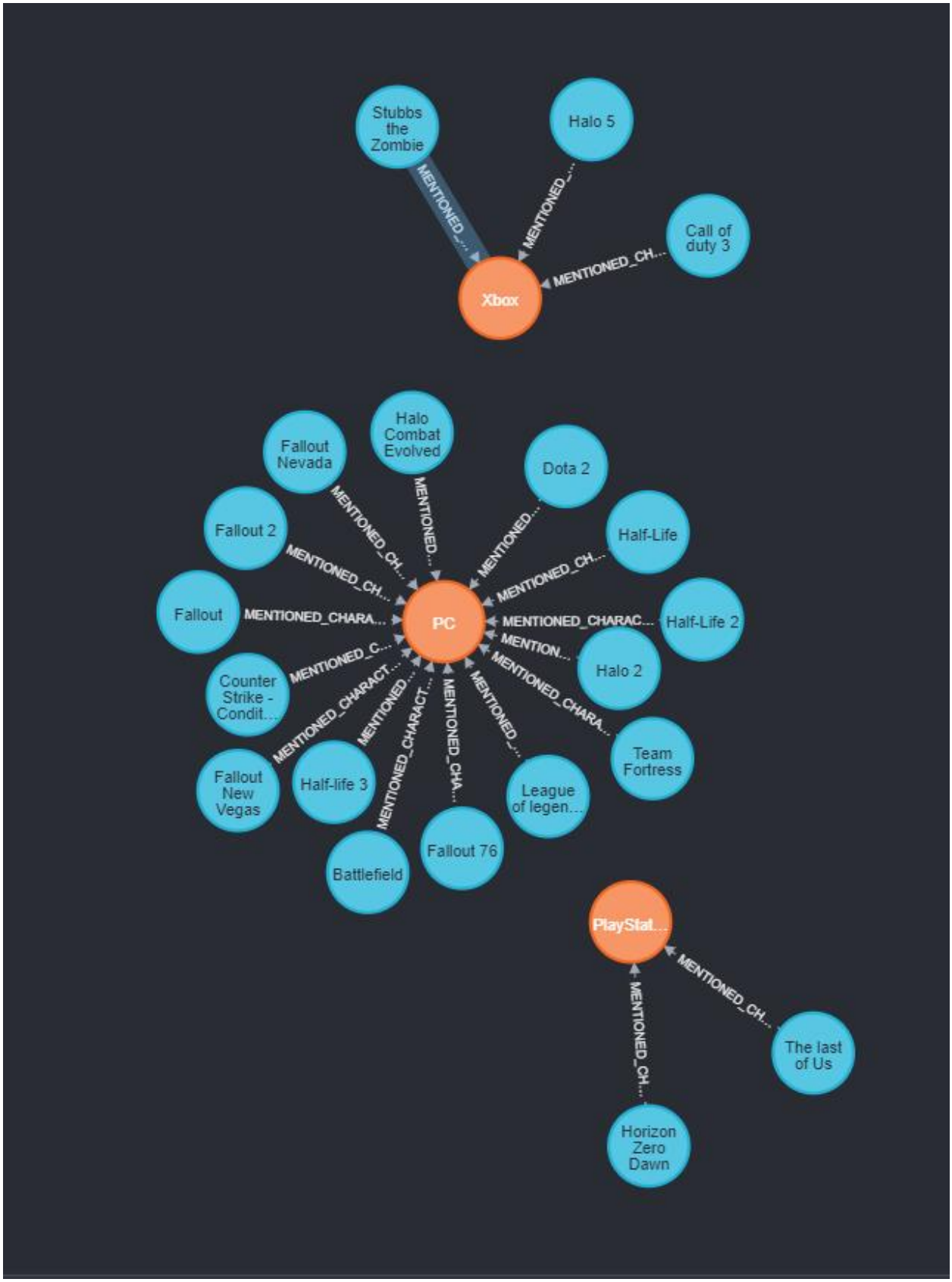
После ввода значений в поля фильтров:

GameDB from Neo4j.

Game Search Halo Game Price from: 0 to: 120 $.

Halo Combat Evolved	*Xbox Live*	*60$*	*PC*	*2003*
Halo 2	*Xbox Live*	*60$*	*PC*	*2005*
Halo 5	*Xbox Live*	*60$*	*Xbox*	*2015*

Граф хранящийся в Neo4j:



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 Neo4j в Википедии - https://ru.wikipedia.org/wiki/Neo4j
2. Документация Noe4j - https://neo4j.com/docs/

Приложение. Листинг.

GameDB-7.js

```
var fs = require("fs")
var http = require("http");
const neo4j = require('neo4j-driver');

const driver = neo4j.driver("bolt://localhost", neo4j.auth.basic("neo4j", "pass")
);
const session = driver.session();

var handle_GET = function (request, response) {
    switch (request.url) {
        case "/":
            fs.readFile("./db7.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 = {};
        console.log(filters)
        session.run("MATCH (game:gamename)-[:MENTIONED_CHARACTERS]-
>(device) WHERE "+
        "game.gamename CONTAINS $title AND game.price>=$from AND game.price<=$to
"+
        "RETURN game, device;",
        {
            title: filters.gamename,
            from: parseInt(filters.price_from),
            to: parseInt(filters.price_to)
        }).then(res => {
            var table = [];
            res.records.forEach(rec => {
                var row = {};
                row.gamename = rec.get('game').properties.gamename;
                row.platform = rec.get('game').properties.platform;
                row.price = rec.get('game').properties.price;
                row.device_title = rec.get('device').properties.title;
                row.release_date = rec.get('game').properties.release_date;
                table.push(row);
            });
            db_data.table = table;
            console.log(db_data)
            response.end(JSON.stringify(db_data));
        }).catch(err => {
            console.error(err);
        });
    });
}

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);
    }
}

session.run("MATCH (n) RETURN n").then(res => {
    if (res.records.length == 0) {
        session.run(
            "CREATE (device1:device $device1) " +
            "CREATE (device2:device $device2) " +
```

```
            "CREATE (device3:device $device3) " +
            "CREATE (game1:gamename $game1) " +
            "CREATE (game2:gamename $game2) " +
            "CREATE (game3:gamename $game3) " +
            "CREATE (game4:gamename $game4) " +
            "CREATE (game5:gamename $game5) " +
            "CREATE (game6:gamename $game6) " +
            "CREATE (game7:gamename $game7) " +
            "CREATE (game8:gamename $game8) " +
            "CREATE (game9:gamename $game9) " +
            "CREATE (game10:gamename $game10) " +
            "CREATE (game11:gamename $game11) " +
            "CREATE (game12:gamename $game12) " +
            "CREATE (game13:gamename $game13) " +
            "CREATE (game14:gamename $game14) " +
            "CREATE (game15:gamename $game15) " +
            "CREATE (game16:gamename $game16) " +
            "CREATE (game17:gamename $game17) " +
            "CREATE (game18:gamename $game18) " +
            "CREATE (game19:gamename $game19) " +
            "CREATE (game20:gamename $game20) " +
            "CREATE" +
            "(game1)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game2)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game3)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game4)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game5)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game6)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game7)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game8)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game9)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game10)-[:MENTIONED_CHARACTERS {chars: []}]->(device2)," +
            "(game11)-[:MENTIONED_CHARACTERS {chars: []}]->(device3)," +
            "(game12)-[:MENTIONED_CHARACTERS {chars: []}]->(device3)," +
            "(game13)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game14)-[:MENTIONED_CHARACTERS {chars: []}]->(device2)," +
            "(game15)-[:MENTIONED_CHARACTERS {chars: []}]->(device2)," +
            "(game16)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game17)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game18)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game19)-[:MENTIONED_CHARACTERS {chars: []}]->(device1)," +
            "(game20)-[:MENTIONED_CHARACTERS {chars: []}]->(device1);",
            {
                device1: { title: "PC", company: "null" },
                device2: { title: "Xbox", company: "Microsoft" },
                device3: { title: "PlayStation", company: "Sony" },
                game1: {
                    gamename: "Fallout",
                    platform: "Bethesda Launcher", price: 35, release_date: 1994
                },
                game2: {
```

```
        gamename: "Fallout 2",
        platform: "Bethesda Launcher", price: 30, release_date: 1996
    },
    game3: {
        gamename: "Fallout Nevada",
        platform: "non", price: 0, release_date: 2003
    },
    game4: {
        gamename: "Fallout New Vegas",
        platform: "Steam", price: 40, release_date: 2009
    },
    game5: {
        gamename: "Halo Combat Evolved",
        platform: "Xbox Live", price: 60, release_date: 2003
    },
    game6: {
        gamename: "Halo 2",
        platform: "Xbox Live", price: 60, release_date: 2005
    },
    game7: {
        gamename: "Dota 2",
        platform: "Steam", price: 0, release_date: 2011
    },
    game8: {
        gamename: "League of legends",
        platform: "RIOT Launcher", price: 0, release_date: 2009
    },
    game9: {
        gamename: "Counter Strike - Condition Zero",
        platform: "Steam", price: 25, release_date: 2004
    },
    game10: {
        gamename: "Call of duty 3",
        platform: "Xbox Live", price: 45, release_date: 2005
    },
    game11: {
        gamename: "Horizon Zero Dawn",
        platform: "PS Network", price: 60, release_date: 2017
    },
    game12: {
        gamename: "The last of Us",
        platform: "PS Network", price: 60, release_date: 2013
    },
    game13: {
        gamename: "Fallout 76",
        platform: "Bethesda Launcher", price: 50, release_date: 2018
    },
    game14: {
        gamename: "Halo 5",
        platform: "Xbox Live", price: 60, release_date: 2015
    },
```

```
                game15: {
                    gamename: "Stubbs the Zombie",
                    platform: "Xbox Live", price: 45, release_date: 2004
                },
                game16: {
                    gamename: "Battlefield 1942",
                    platform: "Retail", price: 55, release_date: 2002
                },
                game17: {
                    gamename: "Team Fortress 2",
                    platform: "Steam", price: 5, release_date: 2008
                },
                game18: {
                    gamename: "Half-Life",
                    platform: "Retail", price: 45, release_date: 1999
                },
                game19: {
                    gamename: "Half-Life 2",
                    platform: "Steam", price: 60, release_date: 2004
                },
                game20: {
                    gamename: "Half-life 3",
                    platform: "CyberPunkedSteam", price: 120, release_date: 2077
                }
            }).then(res => {
                process.emit('readyToServerCreate');
            }).catch(err => {
                console.error(err);
            });
    } else {
        process.emit('readyToServerCreate');
    }
}).catch(err => {
    console.error(err);
});

process.on('readyToServerCreate', () => {
    http.createServer(server_callback).listen(3000);
    console.log("Listen at http://localhost:3000/");
});
```

db7.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 = 1000;

        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.gamename = 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].gamename + "</td><td align=\
"center\">" + rows[i].platform + "</td><td align=\"center\">" + rows[i].price + "
$</td><td align=\"center\">" + rows[i].device_title + "</td><td align=\"center\">
" + rows[i].release_date +  "</td></tr>\n";
                        console.log(rows[i])
                    }
                    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 Neo4j.
    </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>
```