Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

ОТЧЕТ

по лабораторной работе №8

по предмету «Системы и технологии интеллектуальной обработки данных»

Запросы MongoDB с использованием MapReduce, Веб-сервис конвертации данных

|  |  |
| --- | --- |
| Выполнил: | Студент группы 772303  Новик В.С. |
| Проверил:    Минск 2020 | Ассистент кафедры ЭИ Кунцевич А.А. |

**Указания к заданию**

1. Разработайте консольную утилиту для преобразования лога веб-сервера в формате CSV (Comma Separated Values), в формат JSON. Лог должен содержать поля со следующими названиями: URL, IP, timeStamp, timeSpent.

2. Разработайте запросы для загрузки полученных данных в формате JSON в СУБД MongoDB.

**Разработка запросов в СУБД MongoDB**

1. Разработайте следующие запросы, используя встроенные в СУБД MongoDB средствами выборки:

1) Выдать упорядоченный список URL ресурсов.

2) Выдать упорядоченный список IP-адресов пользователей, посетивших ресурс с заданным URL.

3) Выдать упорядоченный список URL ресурсов, посещенных в заданный временной период.

4) Выдать упорядоченный список URL ресурсов, посещенных пользовате-лем с заданным IP-адресом.

2. Разработайте следующие запросы, используя встроенные в СУБД MongoDB средства программирования на основе парадигмы MapReduce:

1) Выдать список URL ресурсов с указанием суммарной длительности по-сещения каждого ресурса, упорядоченный по убыванию.

2) Выдать список URL ресурсов с указанием суммарного количества посе-щений каждого ресурса, упорядоченный по убыванию.

**Результат**

Для выполнения заданий использован сервер NodeJS.

• Исходные данные в формате CSV (Рисунок 1).

• Результат преобразования в формат JSON с помощью эндпоинта сервера NodeJS.

• Результат загрузки данных в MongoDB

• Выдать упорядоченный список URL ресурсов.

• Выдать упорядоченный список IP-адресов пользователей, посетивших ресурс с заданным URL.

• Выдать упорядоченный список URL ресурсов, посещенных в заданный временной период

• Выдать упорядоченный список URL ресурсов, посещенных пользовате-лем с заданным IP-адресом

• Выдать список URL ресурсов с указанием суммарной длительности посещения каждого ресурса, упорядоченный по убыванию

• Выдать список URL ресурсов с указанием суммарного количества посе-щений каждого ресурса, упорядоченный по убыванию

Для реализации базы данных было использовано облачная база данных MongoDB Atlas

**Код сервера:**

const express = require("express");

const mongoose = require("mongoose");

const path = require('path')

const bodyParser = require('body-parser')

const cookieParser = require('cookie-parser')

const config = require("./config/index.js")

const app = express();

let appWs = require('express-ws')(app)

app.use(bodyParser())

app.use(cookieParser());

app.use("/lab",require('./routes/lab'))

app.use("/", express.static(path.join(\_\_dirname,'client', 'build')))

app.get('\*',(req,res)=>{

res.sendFile(path.resolve(\_\_dirname,'client', 'build','index.html'))

})

async function start(){

*try*{

app.listen(config.PORT);

}

*catch* (e) {

console.log("Ошибка подключения к базе данных ", e.message)

process.exit(1);

}

}

start();

const {Router} = require("express")

const mongo = require('mongodb').MongoClient

const router = Router();

const config = require('../config/index')

const url = config.databaseUrl

const csv = require('csvtojson')

const jsonParser = express.json();

router.get("/reduceByCount", jsonParser, (request, response) => {

mongo.connect(url, (err, client) => {

client.db("lab").collection("reduceByCount").drop({});

client.close()

})

executeMongoRequest((error, collection) => {

collection.mapReduce(

mapReduceByTimeSpent,

reduceReduceByCount,

{out: "reduceByCount"}

);

})

mongo.connect(url, (err, client) => {

client.db("lab").collection("reduceByCount").find({}).sort({\_id: -1}).toArray((err, result) => {

*if*(err) {

response.send(err)

}

response.send(result)

})

client.close()

})

})

router.post('/convert', jsonParser, (request, response) => {

csv()

.fromString(request.body.data)

.then((jsonObj) => {

response.send(jsonObj)

})

})

router.post('/save', jsonParser, (request, response) => {

let data = request.body.data;

executeMongoRequest((err, collection) => {

collection.insertMany(data, (err, result) => {

*if* (err) {

response.send(err)

*return* console.log(err)

}

console.log(result.ops)

response.send(result.ops)

})

})

})

router.get("/sortedUrlList", jsonParser, (request, response) => {

executeMongoRequest((error, collection) => {

collection.find({}, {fields: {URL: 1}}).sort({URL: 1}).toArray((error, result) => {

response.send(result)

})

})

})

router.get("/sortedByIpListWithUrl", jsonParser, (request, response) => {

executeMongoRequest((error, collection) => {

let url = request.query.url

collection.find({URL: url}, {fields: {IP: 1}}).sort({IP: 1}).toArray((error, result) => {

response.send(result)

})

})

})

router.get("/sortedByUrlListWithIp", jsonParser, (request, response) => {

executeMongoRequest((error, collection) => {

let ip = request.query.ip

collection.find({IP: ip}, {fields: {URL: 1}}).sort({URL: 1}).toArray((error, result) => {

response.send(result)

})

})

})

router.get("/sortedUrlListWithinPeriod", jsonParser, (request, response) => {

executeMongoRequest((error, collection) => {

let start = request.query.start

let end = request.query.end

collection.find({timeStamp: {$gte: start, $lte: end}}, {fields: {URL: 1}}).sort({URL: 1})

.toArray((error, result) => {

response.send(result)

})

})

})

function mapReduceByTimeSpent() {

emit(this.URL, this.timeSpent);

}

function reduceReduceByTimeSpent(keyURL, valuesTimeSpents) {

*return* valuesTimeSpents.reduce((a, b) => Number.parseInt(a) + Number.parseInt(b))

}

router.get("/reduceByTimeSpent", jsonParser, (request, response) => {

mongo.connect(url, (err, client) => {

client.db("lab").collection("reduceByTimeSpent").drop({});

client.close()

})

executeMongoRequest((error, collection) => {

collection.mapReduce(

mapReduceByTimeSpent,

reduceReduceByTimeSpent,

{out: "reduceByTimeSpent"}

);

})

mongo.connect(url, (err, client) => {

client.db("lab").collection("reduceByTimeSpent").find({}).sort({\_id: -1}).toArray((err, result) => {

*if*(err) {

response.send(err)

}

response.send(result)

})

client.close()

})

})

function reduceReduceByCount(keyURL, valuesTimeSpents) {

*return* valuesTimeSpents.length;

}

function executeMongoRequest(request) {

mongo.connect(url, (err, client) => {

request(err, client.db("lab").collection("lab"))

client.close()

})

}

module.exports = router;