Bazy Danych - NoSQL

MongoDB - zadania

Dawid Majchrowski

Data Laboratorium : 20.11.2019

Data Wykonania: 8.12.2019

1. Wykorzystując bazę danych yelp dataset wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty:
   a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business).

```
db.business.distinct("city");
```

   b. Zwróć liczbę wszystkich recenzji, które pojawiły się w roku 2011 i 2012.

```
db.review.find({date: /201(1|2)/}).count();
```

   c. Zwróć dane wszystkich otwartych (open) firm (business) z pól: id, nazwa, adres.

```
db.business.find({open: true}, {name: 1, full_address: 1});
```

   d. Zwróć dane wszystkich użytkowników (user), którzy uzyskali przynajmniej jeden pozytywny głos z jednej z kategorii (funny, useful, cool), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

```
db.user.find({$or: [
    {"votes.funny": {$gte: 1}},
    {"votes.useful": {$gte: 1}},
    {"votes.cool": {$gte: 1}}]}).sort({name: 1});
```

   e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2013. Wynik posortuj alfabetycznie na podstawie nazwy firmy.

```
db.tip.aggregate([
  {$match:  {date: /2013/}},
  {$group:  { _id: '$business_id', count: { $sum: 1 } } },
  {$lookup: {
            from: "business",
            localField: "_id",
            foreignField: "business_id",
            as: "business"}},
  {$unwind:"$business"},
  {$project: {
            name: "$business.name",
            count : 1}},
  {$sort: {"name": 1}}
]);
```

f. Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji, wynik posortuj on najwyższego uzyskanego wyniku.

```
db.review.aggregate([
  {$group:  { _id: '$business_id', average: { $avg: "$stars" } } },
  {$sort: {average: -1}},
  {$lookup: {
            from: "business",
            localField: "_id",
            foreignField: "business_id",
            as: "business"}},
  {$unwind:"$business"},
  {$project: {
            name: "$business.name",
            average: 1}}
]);
```

g. Usuń wszystkie firmy (business), które posiadają ocenę (stars) poniżej 3

```
db.business.remove({stars: {$lt: 3}});
```

2. Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej wskazówki/napiwku (tip). Wykonaj przykładowe wywołanie.

```
function insertTip(user_id, text, business_id, likes){
        var date = (new Date()).toJSON().substr(0, 10);
        var non_neg_likes = likes > 0 ? likes : 0;
        db.tip.insert({
        user_id: user_id,
        text:text,
        business_id: business_id,
        likes: non_neg_likes,
        date: date,
        type: "tip"
        });
};

insertTip("Bdmk6RQUP0sbXA_V9WiI3Q", "Text in there", "uGykseHzyS5xAMWoN6YUqA", 1);
```

3. Zdefiniuj funkcję (MongoDB), która zwróci wszystkie wskazówki/napiwki (tip), w których w tekście znajdzie się fraza podana jako argument. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.

```
function findTipByRegex(regex){
    return db.tip.find({text: {$regex: regex}})
};

findTipByRegex("Text in there");
```

4. Zdefiniuj funkcję (MongoDB), która umożliwi modyfikację nazwy firmy (business) na podstawie id. Id oraz nazwa mają być przekazywane jako parametry.

```
function updateBusinessName(id, name){
    db.business.update(
    {business_id: id},
    {$set: {name: name}});
};

updateBusinessName("LRKJF43s9-3jG9Lgx4zODg", "New Name");
```

5. Zwróć średnia ilość wszystkich recenzji użytkowników, wykorzystaj map reduce.

```
db.user.mapReduce(
    function() { emit("all", this.review_count); },      // map everything into one key "all"
    function(key, values) { return Array.avg(values); }, // reduce avg for pair (key, values), but there is only one key="all"
    {out: "avg_price"}                                   // {_id: "all", value: (average)} stored in out
);
db.avg_price.find();
```

6. Odwzoruj wszystkie zadania z punktu 1 w języku programowania (np. JAVA) z pomocą API do MongoDB. Wykorzystaj dla każdego zadania odrębną metodę.

(!!! Używamy API do MongoDB w wersji 3 !!!)

Przygotowanie:

1) Importy

```
import com.mongodb.MongoClient;
import com.mongodb.client.*;
import com.mongodb.client.model.*;;
import org.bson.conversions.Bson;

import java.util.Arrays;
import java.util.LinkedList;
import java.util.regex.Pattern;
```

2) Statyczne kolekcje z bazy

```
public class Main { // Mongo Driver 3 style

    private static MongoClient mongoClient;
    private static MongoDatabase database;
    private static MongoCollection business;
    private static MongoCollection review;
    private static MongoCollection user;
    private static MongoCollection tip;

    static {
        mongoClient = new MongoClient(); // Default localhost connection
        database = mongoClient.getDatabase( databaseName: "DawidMajchrowski2");
        business = database.getCollection( s: "business");
        review = database.getCollection( s: "review");
        user = database.getCollection( s: "user");
        tip = database.getCollection( s: "tip");
    }
```

3) Wywołanie wszystkich metod a-g (oczywiście po ich zdefiniowaniu)

```
88      public static void main(String[] args) {
89          System.out.println(a().into(new LinkedList())); // a)
90          System.out.println(b());                        // b)
91          System.out.println(c().into(new LinkedList())); // c)
92          System.out.println(d().into(new LinkedList())); // d)
93          System.out.println(e().into(new LinkedList())); // e)
94          System.out.println(f().into(new LinkedList())); // f)
95          g();                                            // g)
96      }
97  }
98
```

a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business).

```
static DistinctIterable a() {
    // db.business.distinct("city");
    return business.distinct( s: "city", String.class);
}
```

b. Zwróć liczbę wszystkich recenzji, które pojawiły się w roku 2011 i 2012.

```
static long b() {
    // db.review.find({date: /201(1|2)/}).count();
    return review.countDocuments(Filters.regex( fieldName: "date", Pattern.compile("201[12]")));
}
```

c. Zwróć dane wszystkich otwartych (open) firm (business) z pól: id, nazwa, adres.

```
static FindIterable c() {
    // db.business.find({open: true}, {name: 1, full_address: 1});
    return business.find(Filters.eq( fieldName: "open",  value: true)).projection(Projections.include( ...fieldNames: "name", "full_address"));
}
```

d. Zwróć dane wszystkich użytkowników (user), którzy uzyskali przynajmniej jeden pozytywny głos z jednej z kategorii (funny, useful, cool), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

```
static FindIterable d() {
    // db.user.find({$or: [{"votes.funny": {$gte: 1}}, {"votes.useful": {$gte: 1}}, {"votes.cool": {$gte: 1}}]}).sort({name: 1});
    Bson or = Filters.or(
            Filters.gte( fieldName: "votes.useful",  value: 1),
            Filters.gte( fieldName: "votes.funny",  value: 1),
            Filters.gte( fieldName: "votes.cool",  value: 1)
    );
    return user.find(or).sort(Sorts.ascending( ...fieldNames: "name"));
}
```

e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2013. Wynik posortuj alfabetycznie na podstawie nazwy firmy.

```
static AggregateIterable e() {
    // db.tip.aggregate([ {$match:  {date: /2013/}}, {$group:  { _id: '$business_id', count: { $sum: 1 } } },
    // {$lookup: { from: "business", localField: "_id", foreignField: "business_id", as: "business"}},
    // {$unwind:"$business"}, {$project: {name: "$business.name", count : 1}},
    // {$sort: {"name": 1}}])
    return tip.aggregate(Arrays.asList(
            Aggregates.match(Filters.regex( fieldName: "date", Pattern.compile("2013"))),
            Aggregates.group( id: "$business_id", Accumulators.sum( fieldName: "count",  expression: 1)),
            Aggregates.limit(100), // just for faster join
            Aggregates.lookup( from: "business",  localField: "_id",  foreignField: "business_id",  as: "business"),
            Aggregates.unwind( fieldName: "$business"),
            Aggregates.project(Projections.include( ...fieldNames: "count", "business.name")),
            Aggregates.sort(Sorts.ascending( ...fieldNames: "business.name"))));
}
```

f. Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji, wynik posortuj on najwyższego uzyskanego wyniku.

```
static AggregateIterable f() {
    // db.review.aggregate([{$group:  { _id: '$business_id', average: { $avg: "$stars" } } }, {$sort: {average: -1}},
    // {$lookup: { from: "business", localField: "_id", foreignField: "business_id", as: "business"}}, {$unwind:"$business"},
    //  {$project: {name: "$business.name", average: 1}}])
    return review.aggregate(Arrays.asList(
            Aggregates.group( id: "$business_id", Accumulators.avg( fieldName: "average",  expression: "$stars")),
            Aggregates.sort(Sorts.descending( ...fieldNames: "average")),
            Aggregates.limit(100), // just for faster join
            Aggregates.lookup( from: "business",  localField: "_id",  foreignField: "business_id",  as: "business"),
            Aggregates.unwind( fieldName: "$business"),
            Aggregates.project(Projections.include( ...fieldNames: "average", "business.name"))));
}
```

g. Usuń wszystkie firmy (business), które posiadają ocenę (stars) poniżej 3

```
static void g() {
// db.business.remove({stars: {$lt: 3}});
    business.deleteMany(Filters.lt( fieldName: "stars",  value: 3));
}
```

7. Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: studentów, przedmiotów oraz sal zajęciowych. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych.

Struktura: () - Atrybut złożony, [] - Tablica
Kursy: Nazwa, Ects, Godziny(wykład/ćwiczenia), prowadzący[]
Sale: Budynek, Numer Sali, Piętro, Adres(miasto, ulica, kod pocztowy)
Studenci: Imię, Nazwisko, Adres(miasto, ulica, kod pocztowy), telefon, email, kursy(id_kursu, id_sali, wykład czy ćwiczenia, prowadzący, data(dzień, godzina))[].

Struktura również pokazana w poniższych screenach.

```
db.course.insert({
    name: "ASD",
    ECTS: "6",
    hours: {
        lecture: "30",
        recitation: "30"
    },
    leaders: ["PF", "L1", "L2"]
});
```

```
db.classroom.insert({
    building: "D17",
    number: "1.16",
    floor: 1,
    address: {
        city: "Krakow",
        street: "Kawiory 21",
        zip_code: "30-055"
    }
});
```

```
db.student.insert({
    first_name: "Tom", second_name: "Paul",
    address: {
        city: "Krakow",
        street: "Somewhere",
        zip_code: "30-055"
    },
    phone: "999-999-999", email: "example@gmail.com",
    courses: [{
        course_id: "5dec3d9f511675e0951fb8d8",
        lecture: true,
        classroom: "5dec6411511675e0951fb8db",
        date: {
            day: "Monday",
            hour: "14:40"
            }, lead: "PF" },
    {
        course_id: "5dec63fb511675e0951fb8da",
        lecture: false,
        classroom: "5dec3da3511675e0951fb8d9",
        date: {
            day: "Monday",
            hour: "14:40"
            },
        lead: "L1"
    }
    ]
});
```

Oraz na poniższych screenach struktura bazy danych dla dodanych przykładowych rekordów.

```
/* 1 */
{
    "_id" : ObjectId("5dec6411511675e0951fb8db"),
    "building" : "D17",
    "number" : "1.16",
    "floor" : 1.0,
    "address" : {
        "city" : "Krakow",
        "street" : "Kawiory 21",
        "zip_code" : "30-055"
    }
}
```

```
/* 1 */
{
    "_id" : ObjectId("5dec63fb511675e0951fb8da"),
    "name" : "ASD",
    "ECTS" : "6",
    "hours" : {
        "lecture" : "30",
        "recitation" : "30"
    },
    "leaders" : [
        "PF",
        "L1",
        "L2"
    ]
}
```

```
/* 1 */
{
    "_id" : ObjectId("5dec650d511675e0951fb8dc"),
    "first_name" : "Tom",
    "second_name" : "Paul",
    "address" : {
        "city" : "Krakow",
        "street" : "Somewhere",
        "zip_code" : "30-055"
    },
    "phone" : "999-999-999",
    "email" : "example@gmail.com",
    "courses" : [
        {
            "course_id" : "5dec3d9f511675e0951fb8d8",
            "lecture" : true,
            "classroom" : "5dec6411511675e0951fb8db",
            "date" : {
                "day" : "Monday",
                "hour" : "14:40"
            },
            "lead" : "PF"
        },
        {
            "course_id" : "5dec63fb511675e0951fb8da",
            "lecture" : false,
            "classroom" : "5dec3da3511675e0951fb8d9",
            "date" : {
                "day" : "Monday",
                "hour" : "14:40"
            },
            "lead" : "L1"
        }
    ]
}
```