Bazy danych – NoSQL
MongoDB – zadania

Krzysztof Bieniasz
20.11.2019
1.12.2019

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.getCollection('Business').distinct("city")
```

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

Komentarz: W sytuacji, kiedy warunki dotyczące daty były bardziej wymagające najpewniej należałoby wykorzystać formuły dedykowane dla dat. W przypadku jednak, kiedy daty w dokumencie przechowywane są jako String i mamy warunek tylko na określony rok najprostszym i pewnym rozwiązaniem jest utworzenie wyrażenia regularnego tak jak poniżej.

```
db.getCollection('Review').count({
    $or: [{
        'date': new RegExp("2011")
    }, {
        'date': new RegExp("2012")
    }]
})
```

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

```
db.getCollection('Business')
    .find({
        "open": true
    }, {
        "_id": 1,
        "business_id": 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.

Komentarz: Dla szybszego wykonania sortowania można dodać indeks w następujący sposób:
db.User.createIndex({ 'name': 1 })

```
db.getCollection('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.

Komentarz: W pierwszym kroku tworzę tabelę tips_amount, którą następnie jest argumentem funkcji lookup dla agregacji w tabeli Businnes

```
db.Tip.aggregate([{
        $match: {
            'date': new RegExp('2013')
```

```
            }
        },
        {
            $group: {
                _id: "$business_id",
                tips_amount: {
                    $sum: 1
                }
            }
        },
        {
            $out: "tips_amount"
        }
    ])


    db.Business.aggregate([{
            $lookup: {
                from: "tips_amount",
                localField: 'business_id',
                foreignField: '_id',
                as: "tips"
            }
        },
        {
            $unwind: {
                path: "$tips",
                preserveNullAndEmptyArrays: true
            }
        },

        {
            $project: {
                _id: '$_id',
                business_id: "$business_id",
                name: '$name',
                tips: "$tips.tips_amount"
            }
        },
        {
            $sort: {
                "name": 1
            }
        }

    ])
```

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

Komentarz: W pierwszym kroku tworzę tabelę reviews_avareges, którą następnie jest argumentem funkcji lookup dla agregacji w tabeli Businnes

```
    db.Review.aggregate([{
            $group: {
                _id: "$business_id",
                average_stars: {
                    $avg: "$stars"
                }
            }
        },
        {
            $out: "reviews_avareges"
        }
    ])


    db.Business.aggregate([{
```

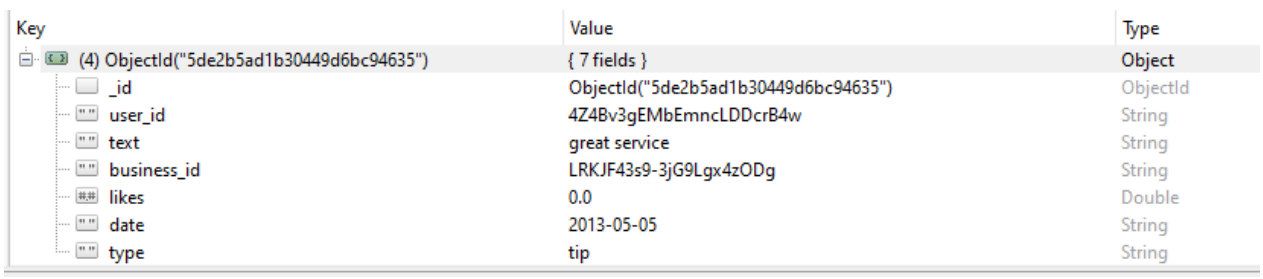
```
                $lookup: {
                    from: "reviews_avareges",
                    localField: 'business_id',
                    foreignField: '_id',
                    as: "avarages"
                }
            },
            {
                $unwind: {
                    path: "$avarages",
                    preserveNullAndEmptyArrays: true
                }
            },

            {
                $project: {
                    _id: '$_id',
                    name: '$name',
                    average_stars: "$avarages.average_stars"
                }
            },
            {
                $sort: {
                    "average_stars": -1
                }
            }

        ])
```

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

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

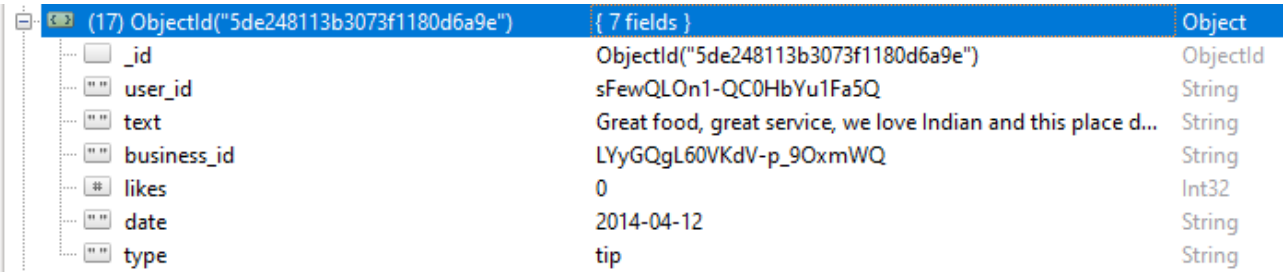
2. Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej wskazówki/napiwku (tip). Wykonaj przykładowe wywołanie.

```
function insertTip(user_id, text, business_id, likes, date, type){
    db.Tip.insert({
        user_id:user_id,
        text:text,
        business_id:business_id,
        likes:likes,
        date:date,
        type:type
    });
}


insertTip("4Z4Bv3gEMbEmncLDDcrB4w","great service","LRKJF43s9-
3jG9Lgx4zODg",0,"2013-05-05","tip")

db.getCollection('Tip').find({text:"great service"})
```



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 searchPhraseTip(phrase){
    return db.getCollection('Tip').find({"text": new RegExp(phrase)})
}

searchPhraseTip("great service")
```



(17) ObjectId("5de248113b3073f1180d6a9e")	{ 7 fields }	Object
_id	ObjectId("5de248113b3073f1180d6a9e")	ObjectId
user_id	sFewQLOn1-QC0HbYu1Fa5Q	String
text	Great food, great service, we love Indian and this place d...	String
business_id	LYyGQgL60VKdV-p_9OxmWQ	String
likes	0	Int32
date	2014-04-12	String
type	tip	String

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

```
function updateCompanyName(business_id, new_name){
    db.Business.update(
        {business_id:business_id},
        {$set: {name: new_name}}
    )
}

updateCompanyName("LRKJF43s9-3jG9Lgx4zODg", "New")
```

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

```
db.User.mapReduce(
    function() {
        emit(1, this.review_count);
    },
    function(keys, values) {
        var users_counter = 0;
        var sum_of_review_counter = 0;
        values.forEach(function(v) {
            sum_of_review_counter += v;
            users_counter++;
        });
        return sum_of_review_counter / users_counter;
    }, {
        out: 'average_reviews'
    }
)
```

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

Inicjalizacja klasy:

```
private MongoClient mongoClient;
private DB db;
private MongoDatabase database;
public MongoLab() throws UnknownHostException {
```

```
        mongoClient = new MongoClient("localhost", 27017);
        db = mongoClient.getDB("DataSet");
        database = mongoClient.getDatabase("DataSet");
}
```

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

```
        private List<String> exerciseA(){
                return db.getCollection("Business").distinct("city");
        }
```

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

```
        private long exerciseB(){
                Pattern regex2011 = Pattern.compile("2011");
                DBObject clause1 = new BasicDBObject("date", regex2011);
                Pattern regex2012 = Pattern.compile("2012");
                DBObject clause2 = new BasicDBObject("date", regex2012);
                BasicDBList or = new BasicDBList();
                or.add(clause1);
                or.add(clause2);
                DBObject query = new BasicDBObject("$or", or);
                return db.getCollection("Review").count(query);
        }
```

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

```
private void exerciseC()
{
        DBCursor cursor = db.getCollection("Business")
                        .find( new BasicDBObject("open", true),
                                        new BasicDBObject().append("businnes_id",1)
                                                .append("name",1)
                                                .append("full_address",1));
        while (cursor.hasNext()) {
                System.out.println(cursor.next());
        }
}
```

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.

```
        private void exerciseD()
        {
                DBObject clause1 = new BasicDBObject("votes.funny",
                        new BasicDBObject("$gte", 1));
                DBObject clause2 = new BasicDBObject("votes.useful",
                        new BasicDBObject("$gte", 1));
                DBObject clause3 = new BasicDBObject("votes.cool",
                        new BasicDBObject("$gte", 1));
                BasicDBList or = new BasicDBList();
                or.add(clause1);
                or.add(clause2);
                or.add(clause3);
                DBObject query = new BasicDBObject("$or", or);
                DBCollection collection = db.getCollection("User");
                collection.createIndex(new BasicDBObject("name",1));
                DBCursor cursor =  db.getCollection("User").
                                find(query).sort(new BasicDBObject("name",1));
                while (cursor.hasNext()) {
                        System.out.println(cursor.next());
                }
        }
```

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

```
	private AggregateIterable<Document> exerciseE()
	{


		AggregateIterable<Document> documents1 =
				database.getCollection("Tip").aggregate(Arrays.asList(
			new Document("$match",new
Document("date",Pattern.compile("2013"))),
			new Document("$group", new Document("_id","$business_id")
				.append("tips_amount", new Document("$sum",1))),
			new Document("$out", "tips_amount")
		    ));

		AggregateIterable<Document> documents2 =

database.getCollection("Business").aggregate(Arrays.asList(
			new Document("$lookup", new Document("from", "tips_amount")
				.append("localField","business_id")
				.append("foreignField","_id")
				.append("as","tips")),
			new Document("$unwind",
				new Document("path","$tips")
				   .append("preserveNullAndEmptyArrays",true)),
			new Document("$project",new Document("_id","$_id")
				.append("business_id","$business_id")
				.append("name","$name")
				.append("tips", "$tips.tips_amount")),
			new Document("$sort", new Document("name",1))
		));
		return documents2;

	}
```

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

```
private AggregateIterable<Document> exerciseF() {


	AggregateIterable<Document> documents1 =
			database.getCollection("Review").aggregate(Arrays.asList(
			new Document("$group", new Document("_id", "$business_id")
				.append("average_stars", new Document("$avg",
"$stars"))),
			new Document("$out", "reviews_avareges")
	));

	AggregateIterable<Document> documents2 =
			database.getCollection("Business").aggregate(Arrays.asList(
			new Document("$lookup", new Document("from",
"reviews_avareges")
				.append("localField","business_id")
				.append("foreignField","_id")
				.append("as","avarages")),
			new Document("$unwind",
				new Document("path","$avarages")
					.append("preserveNullAndEmptyArrays",true)),
			new Document("$project",new Document("_id","$_id")
					.append("business_id","$business_id")
					.append("name","$name")
					.append("average_stars",
"$avarages.average_stars")),
			new Document("$sort", new Document("average_stars",-1))
```

```
        ));

        return documents2;

    }
```

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

```
        private void exerciseG()
        {
            database.getCollection("Business").
                deleteMany(new Document("stars", new Document("$lt",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.

```
db.Students.insert({
    name: "Robert",
    surname: "Lewandowski",
    birthdate: ISODate("1988-08-21"),
    address: {
        street: "Bulgarska 7",
        city: "Poznan"
    },
    class: "1e",
    grades: [{
            subject: "Sport",
            teacher: "Brzeczek",
            grade: 5.0
        },
        {
            subject: "Sport",
            teacher: "Brzeczek",
            grade: 4.0
        }
    ]
})


db.Subjects.insert({
    name: "Sport",
    teachers: [{
            teacher: "Brzeczek",
            classes: ["1e", "2b", "3a"]
        },
        {
            teacher: "Boniek",
            classes: ["2e", "1b", "3b"]
        }
    ],
    classrooms: ["big gym", "small gym"]
})


db.Classrooms.insert({
    name: "12a",
    max_capacity: 30,
    features: ["TV", "boombox"]
})
```