

Bazy danych – NoSQL MongoDB – zadania

Imię i nazwisko: Dawid Białka.

Tydzień B, czwartek godz lab.: 14:20.

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*). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
db.Business.distinct("city").sort( { "city": 1 } )
```

- b. Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (włącznie).

```
db.Review.aggregate( [
    { $project: { year: { $substr: [ "$date", 0, 4 ] } } },
    { $match: { year: { $gte: '2011' } } },
    { $group: { _id: null, count: { $sum: 1 } } }
] )
```

- c. Zwróć dane wszystkich zamkniętych (*open*) firm (*business*) z pól: nazwa, adres, gwiazdki (*stars*).

```
db.Business.find({open: false}, {name:1, full_address:1, stars:1})
```

- d. Zwróć dane wszystkich użytkowników (*user*), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (*funny lub useful*), wynik posortuj alfabetycznie według imienia użytkownika.

```
db.User.aggregate( [
    { $match:
        { $and: [ {"votes.funny": { $eq: 0 } }, {"votes.useful": { $eq: 0 } } ] }
    },
    { $sort: { name: 1 } }
] )
```

- e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (*tip*) w 2012. Wynik posortuj alfabetycznie według liczby (*tip*).

```
db.Tip.aggregate( [
    { $match: { "date": /2012/ } },
    { $group: { _id: "$business_id", count: { $sum: 1 } } },
    { $lookup: { from: "Business",
        localField: "_id",
        foreignField: "business_id",
        as: "business_data" } },
    { $unwind: "$business_data" },
    { $project: { "name": "$business_data.name", "count": "$count" } },
    { $sort: { count: 1 } }
] )
```

- f. Wyznacz, jaką średnią ocen (*stars*) uzyskała każda firma (*business*) na podstawie wszystkich recenzji. Wynik ogranicz do recenzji, które uzyskały min 4.0 gwiazdki.

```
db.Review.aggregate ( [
    { $match: { stars: { $gte: 4 } } },
    { $group: { _id: "$business_id", avg_stars: { $avg: "$stars" } } },
    { $lookup: { from: "Business",
        localField: "_id",
        foreignField: "business_id",
        as: "business_data" } },
    { $unwind: "$business_data" },
    { $project: { "name": "$business_data.name", "avg_stars": "$avg_stars" } }
] )
```

- g. Usuń wszystkie firmy (*business*), które posiadają ocenę (*stars*) równą 2.0.

```
db.Business.deleteMany( { "stars": { $eq: 2 } } )
```

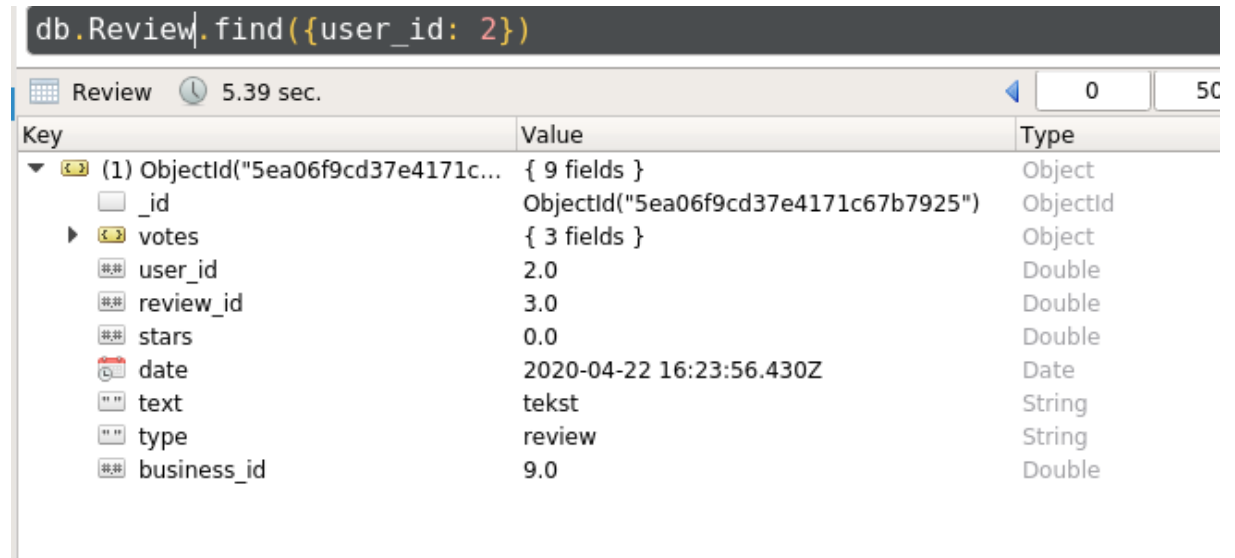
2. Zdefiniuj funkcję (*MongoDB*) umożliwiającą dodanie nowej recenzji (*review*). Wykonaj przykładowe wywołanie.

```
db.system.js.save( {  
  _id: "insert_review",  
  value : function insert_review(user_id, review_id, text, business_id)  
  {  
    db.Review.insert( {  
      votes:{funny: 0, useful: 0, cool: 0},  
      user_id: user_id,  
      review_id: review_id,  
      stars: 0,  
      date: new Date(),  
      text: text,  
      type: "review",  
      business_id: business_id  
    } )  
  }  
} )
```

Wywołanie:

```
db.loadServerScripts();
```

```
insert_review(2, 3, "tekst", 9)
```



Key	Value	Type
(1) ObjectId("5ea06f9cd37e4171c...")	{ 9 fields }	Object
_id	ObjectId("5ea06f9cd37e4171c67b7925")	ObjectId
votes	{ 3 fields }	Object
user_id	2.0	Double
review_id	3.0	Double
stars	0.0	Double
date	2020-04-22 16:23:56.430Z	Date
text	tekst	String
type	review	String
business_id	9.0	Double

3. Zdefiniuj funkcję (*MongoDB*), która zwróci wszystkie biznesy (*business*), w których w kategorii znajduje się podana przez użytkownika cechę. Wartość kategorii należy przekazać do funkcji jako parametr. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.

```
db.system.js.save( {  
  _id: "get_business",  
  value : function get_business(category)  
  {  
    return db.Business.find({categories: {$eq: category}})  
  }  
} )
```

Wywołanie:

```
db.loadServerScripts();
```

```
var business = get_business("Restaurants");
```

```
business.forEach(  
  function(doc)  
  { print(doc) } )
```

```
{
  "_id" : ObjectId("5e85cd99c2ae6df2373e0c12"),
  "business_id" : "JwUE5GmEO-sH1FuwJgKBlQ",
  "full_address" : "6162 US Highway 51\nDe Forest, WI 53532",
  "hours" : {},
  "open" : true,
  "categories" : [
    "Restaurants"
  ],
  "city" : "De Forest",
  "review_count" : 26,
  "name" : "Pine Cone Restaurant",
  "..." : "..."
}
```

4. Zdefiniuj funkcję (*MongoDB*), która umożliwi modyfikację nazwy użytkownika (*user*) na podstawie podanego id. Id oraz nazwa mają być przekazywane jako parametry.

```
db.system.js.save( {
  _id: "update_user",
  value : function update_user(user_id, name)
  {
    db.User.update(
      {user_id: user_id},
      {$set: {name: name} } )
  }
} )
```

Wywołanie:

```
db.loadServerScripts();
```

```
update_user("MWhR9LvOdRbqtu1I_DRFBg", "Zygmunt")
```

db.User.find({})		
User 0.001 sec. 0 50		
Key	Value	Type
(1) ObjectId("5e85cfd9aa0f27f2c1...")	{ 12 fields }	Object
_id	ObjectId("5e85cfd9aa0f27f2c1df8dfa")	ObjectId
yelping_since	2011-12	String
votes	{ 3 fields }	Object
review_count	1	Int32
name	Zygmunt	String
user_id	MWhR9LvOdRbqtu1I_DRFBg	String
friends	[13 elements]	Array
fans	0	Int32
average_stars	5.0	Double
type	user	String
compliments	{ 0 fields }	Object
elite	[0 elements]	Array

5. Zwróć średnią ilość wszystkich wskazówek/napiwków dla każdego z biznesów, wykorzystaj map reduce.

```
var map_function = function()
{
    var key = this.business_id
    var value = {sum: this.tip_count, count: 1 }
    Emit(key, value);
};

var reduce_function = function(key, values)
{
    var result = {sum: 0, count: 0 };

    values.forEach(
        function(value)
        {
            result.sum = result.sum + value.sum;
            result.count = result.count + value.count;
        }
    )

    return result;
}

var finalize_function = function(key, reduced_value)
{
    var average_tips = reduced_value.sum / reduced_value.count;

    return average_tips;
}

db.business.mapReduce(
    map_function,
    reduce_function,
    {
        out: "tips_business_average",
        finalize: finalize_function
    }
)
```

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

```
package pl.edu.agh.bd.mongo;

import java.net.UnknownHostException;

import java.util.Arrays;

import com.mongodb.DB;
import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;

public class MongoLab {

    private MongoClient mongoClient;

    private DB db;

    public MongoLab() throws UnknownHostException {

        mongoClient = new MongoClient();

        db = mongoClient.getDB("admin");

    }

    //1a

    public List<String> getCities() {

        return db.getCollection("business")

            .distinct("city", String.class)

            .sort(ascending("city"))

            .into(new ArrayList<>());

    }

}
```

```

//1b

public void getreviews() {

    DBCollection coll = db.getCollection("Bussines");

    BasicDBObject project = new BasicDBObject("$project",

        new BasicDBObject("year",

            new BasicDBObject("$substr", "[ "$date", 0, 4 ]" ) ) );

    BasicDBObject match = new BasicDBObject("$match",

        new BasicDBObject("year",

            new BasicDBObject("$gte", "2011")));

    BasicDBObject group = new BasicDBObject("$group",

        new BasicDBObject("_id", null) )

        .put("count", new BasicDBObject("$sum", 1));

    List<DBObject> pipeline = Arrays.asList(project, match group);

    AggregationOutput output = col.aggregate(pipeline);

    for(DBObject result : output.results()) {

        System.out.println(resul);

    }

}

```



```
//1c

public void closedBusiness() {

    DBCollection coll = db.getCollection("Bussines");

    BasicDBObject field = new BasicDBObject("name, 1")
        .put("full_address", 1)
        .put("starts", 1);

    BasicDBObject query = new BasicDBObject("open", false)
        .put(field)

    cursor = coll.find(query);

    try {

        while(cursor.hasNext()) {

            System.out.println(cursor.next());

        }

    } finally {

        cursor.close();

    }

}
```

```

//1d

public void getUsersWithoutVote() {

    DBCollection coll = db.getCollection("User");

    BasicDBObject field = new BasicDBObject("votes.funny",
        new BasicDBObject("$eq", 0) )
        .put("votes.useful",
            new BasicDBObject("$eq", 0) );

    BasicDBObject match = new BasicDBObject("$match",
        new BasicDBObject("$and", field) )

    BasicDBObject sort = new BasicDBObject("$sort",
        new BasicDBObject("name", 1));

    List<DBObject> pipeline = Arrays.asList(match, sort);
    AggregationOutput output = col.aggregate(pipeline);

    for(DBObject result : output.results()) {

        System.out.println(resul);
    }
}

```

```

//1e

public void tipsPerBussines(){

    DBCollection coll = db.getCollection("Tip");

    BasicDBObject match = new BasicDBObject("$match",
        new BasicDBObject("date", "/2012/") );

    BasicDBObject group = new BasicDBObject("$group",
        new BasicDBObject("_id", "$business_id") )
        .put("count", new BasicDBObject("$sum", 1));

    BasicDBObject lookup = new BasicDBObject("from", "Business")
        .put("localField", "_id")
        .put("foreignField", "business_id")
        .put("as", "business_data");

    BasicDBObject unwind = new BasicDBObject("$unwind", "$business_data");

    BasicDBObject project = new BasicDBObject("name",
        "$business_data.name")
        .put("count", "$count");

    BasicDBObject sort = new BasicDBObject("$sort",
        new BasicDBObject("name", 1));

    List<DBObject> pipeline = Arrays.asList(match, group, lookup, unwind,
        project, sort);

    AggregationOutput output = col.aggregate(pipeline);

```

```

        for(DBObject result : output.results()) {

            System.out.println(resul);

        }

    }

    //1f

    public void averageStarsBusinessFromReview() {

        DBCollection coll = db.getCollection("Review");

        BasicDBObject match = new BasicDBObject("$match",
            new BasicDBObject("stars",
                new BasicDBObject("$gte", 4) ) );

        BasicDBObject group = new BasicDBObject("$group",
            new BasicDBObject("_id", "$business_id") )
            .put("avg_stars", new BasicDBObject("$avg", "$stars"));

        BasicDBObject lookup = new BasicDBObject("from", "Business")
            .put("localField", "_id")
            .put("foreignField", "business_id")
            .put("as", "business_data");

        BasicDBObject unwind = new BasicDBObject("$unwind", "$business_data");

        BasicDBObject project = new BasicDBObject("name",
            "$business_data.name")
            .put("avg_stars", "$avg_stars");
    }

```

```

        List<DBObject> pipeline = Arrays.asList(match, group, lookup, unwind,
            project);

        AggregationOutput output = col.aggregate(pipeline);

        for(DBObject result : output.results()) {

            System.out.println(resul);

        }

    }

    //1g

    public void delete_business() {

        DBCollection coll = db.getCollection("Business");

        BasicDBObject field = new BasicDBObject("$stars",
            new BasicDBObject("$eq", 2) );

        coll.deleteMany(field);

    }

```

7. Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: klientów, zakupu oraz przedmiotu zakupu. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych. Uzasadnij swoją propozycję

Klient:

```

{
    "_id ": "123456789 ",
    "firstName": "Zygmunt",
    "surname": "Waza",
    "contactInfo":
    {
        "email": zygmuntwaza@agh.edu.pl,
        "phone": "987654321",
    },
}

```

```
"purchasesMade":  
  
[  
  
  {  
  
    "$ref": "purchases,  
  
    "$id": "3643624652"  
  
  },  
  
  {  
  
    "$ref": "purchases,  
  
    "$id": "6547453457"  
  
  },  
  
]  
  
}
```

Zakupy:

```
{  
  
  "_id ": "123453789",  
  
  "amountPaid": "997",  
  
  "currency": "$",  
  
  "dateOfPurchase": 2020-04-20,  
  
  "items":  
  
  [  
  
    {  
  
      "$ref": "items",  
  
      "$id": "656425462",  
  
      "refund": false  
  
    },  
  
    {  
  
      "$ref": "items",  
  
      "$id": "3428473287",  
  
      "refund": true  
  
    }  
  
  ]  
  
}
```

```
},
```

Przedmioty:

```
{  
  "_id ": "143553789",  
  "category": "vehicle",  
  "name": "Fiat Multipla",  
  "inStock": true,  
  "quantity": 420,  
  "price": 55455  
  "dateOfPurchase": 2020-04-20,  
  "items":  
}
```

Klient może zrobić wiele zakupów, każdy zakup może składać się z wielu przedmiotów.