

Bazy danych – NoSQL
MongoDB – zadania

Mateusz Nabywaniec
data laboratorium 27.11.19 r.
data wykonania 15.12.19 r.

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.getCollection('business').distinct("city").sort()
```

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

```
db.getCollection('review').count({
  'date': {"$gte" : ("2011-01-01")}
})
```

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

```
db.getCollection('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 na podstawie imienia użytkownika.

```
db.getCollection('user').find({ $and: [
  {'votes.funny' : {$lte : 0}},
  {'votes.useful' : {$lte : 0}},
]}
).sort({'name' : 1})
```

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

Na początku tworzę kolekcję tips_amount zawierającą sumę tipów dla każdej firmy. Następnie odwołuję się do tej tabeli i dopasowuję nazwę biznesu do jego id. Na końcu sortuję wynik najpierw wg liczby napiwków, a potem alfabetycznie.

```
db.tip.aggregate([
  {
    $match: {
      'date': new RegExp('2012')
    }
  },
  {
    $group: {
      _id: "$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: {
      "tips": -1,
      "name": 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([
  $group: {
    _id: "$business_id",
    avg_stars: {
      $avg: "$stars"
    }
  },
  {
    $out: "avg_stars"
  }
]
)

db.business.aggregate([
  $lookup: {
    from: "avg_stars",
    localField: 'business_id',
    foreignField: '_id',
    as: "average"
  },
  {
    $unwind: {
      path: "$average",
      preserveNullAndEmptyArrays: true
    }
  }
]
)

```

```

    }
  },
  {
    $project: {
      _id : '$_id',
      business_id : "$business_id",
      name : '$name',
      avg_stars : {$fiter :{
        input: "$average",
        as : "average"
        cond : {
          "$average.avg_stars" : {$gte : 4}
        }
      }
    }
  }
},
{
  $sort: {
    "avg_stars" : -1
  }
}
]
])

```

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.

```

function insertReview(votes, user_id, review_id, stars, date, text,
business_id){
  db.review.insert({
    votes:votes,
    user_id:user_id,
    review_id:review_id,
    stars:stars,
    date:date,
    text:text,
    business_id:business_id,
    type:'review'
  });
}

```

Przykładowe wywołanie:

```

insertReview([0,10,10], "MWhR9Lv0dRbqtu1I_DRFBg",
"afjaksdajkhriuweabdzkj", Number(5),
"2019-10-10", "Fantastycznie", "vcNAWiLM4dR7D2nwwJ7nCA")

db.review.find({
  date:"2019-10-10"
})

```

Key	Value	Type
▼ (1) ObjectId("5de69f980f93c94e... { 9 fields }		Object
_id	ObjectId("5de69f980f93c94ef53aeb02")	ObjectId
▶ votes	[3 elements]	Array
user_id	MWhR9LvOdRbqtu1I_DRFBg	String
review_id	afjaksdajkhriuweabdzkj	String
stars	5.0	Double
date	2019-10-10	String
text	Fantastycznie	String
business_id	vcNAWiLM4dR7D2nwwj7nCA	String
type	review	String

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

```
function findBusinessByCategory(category){
  return db.business.find({
    categories:category
  });
}

findBusinessByCategory("Food")
```

business 0.002 sec.		
Key	Value	Type
▶ (1) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (2) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (3) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (4) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (5) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (6) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (7) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (8) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (9) ObjectId("5de676285f1b18... { 16 fields }		Object
▶ (10) ObjectId("5de676285f1b1... { 16 fields }		Object
▼ (11) ObjectId("5de676285f1b1... { 16 fields }		Object
_id	ObjectId("5de676285f1b18370acc2d8...)	ObjectId
business_id	bMnr4RK5HEFvpuxB5xRoA	String
full_address	6637 University Ave Middleton, WI 53...	String
hours	{ 6 fields }	Object
open	true	Boolean
▶ categories	[2 elements]	Array
[0]	Bakeries	String
[1]	Food	String
city	Middleton	String
review_count	12	Int32
name	Scott's Pastry Shoppe	String
▶ neighborhoods	[0 elements]	Array
longitude	-89.4957363	Double
state	WI	String
stars	4.0	Double
latitude	43.0961424	Double
▶ attributes	{ 4 fields }	Object
type	business	String
▶ (12) ObjectId("5de676285f1b1... { 16 fields }		Object

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

```
function updateUser(user_id, new_name){
  db.user.update(
    {user_id:user_id},
    {$set: {name: new_name}}
  )
}
```

```
updateUserName("5Xh4Qc3rxhAQ_NcNtxLssQ", "Mateusz Nab")
db.user.find({name:"Mateusz Nab"})
```

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

```
var mapFunc = function(){
    emit(this.business_id, 1)
}

var reduceFunc = function (business_id, tips){
    return Array.sum(tips)
}

db.tip.mapReduce(
    mapFunc,
    reduceFunc,
    {out : "tips_per_business"}
)

db.tips_per_business.aggregate([
    {$group: { _id : null, avg: {$avg : "$value"}}}
])

}
```

tips_per_business 0.021 sec.		
Key	Value	Type
<div>(1) null</div> <div> <div>id</div> <div>avg</div> </div>	<div>{ 2 fields }</div> <div>null</div> <div>13.4812263867063</div>	<div>Object</div> <div>Null</div> <div>Double</div>

user 0.381 sec.		
Key	Value	Type
<div>(1) ObjectId("5de676bc5f1b183...")</div> <div> <div>id</div> <div>yelping_since</div> <div>votes</div> <div>review_count</div> <div>name</div> <div>user_id</div> <div>friends</div> <div>fans</div> <div>average_stars</div> <div>type</div> <div>compliments</div> <div>elite</div> </div>	<div>{ 12 fields }</div> <div>ObjectId("5de676bc5f1b18370ae4a72a")</div> <div>2012-01</div> <div>{ 3 fields }</div> <div>1</div> <div>Mateusz Nab</div> <div>5Xh4Qc3rxhAQ_NcNtxLssQ</div> <div>[0 elements]</div> <div>0</div> <div>1.0</div> <div>user</div> <div>{ 0 fields }</div> <div>[0 elements]</div>	<div>Object</div> <div>ObjectId</div> <div>String</div> <div>Object</div> <div>Int32</div> <div>String</div> <div>String</div> <div>Array</div> <div>Int32</div> <div>Double</div> <div>String</div> <div>Object</div> <div>Array</div>

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

a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

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

public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
}
```

```

List<String> businesses = mongoLab.exerciseA();
Collections.sort(businesses);
for(String business : businesses) {
    System.out.println(business);
}
//mongoLab.showCollections();
}

```

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

```

private long exerciseB(){
   DBObject query = new BasicDBObject("date", new BasicDBObject("$gte", "2011-01-01"));
    return db.getCollection("review").count(query);
}

```

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

```

private void exerciseC(){
    DBCursor cursor = db.getCollection("business")
        .find( new BasicDBObject("open", false),
              new BasicDBObject().append("name",1)
                                .append("full_address",1)
                                .append("stars", 1));

    while (cursor.hasNext()) {
        System.out.println(cursor.next());
    }
}

```

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

```

private void exerciseD() {
    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 2012. Wynik posortuj alfabetycznie na podstawie liczby (tip).

```

private AggregatIterable<Document> exerciseE(){
    AggregatIterable<Document> tipsAmount =
    database.getCollection("tips").aggregate(Arrays.asList(
        new Document("$match", new Document("date",
        Pattern.compile("2012"))),
        new Document("$group", new Document("_id", "$business_id")
        .append("tips_amount", new Document("$sum", 1))),

```

```

        new Document("$out", "tips_amount")
    ));

    Aggregateterable<Document> result =
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("tips", -1))
));
    return result;
}

```

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

```

private Aggregateterable<Document> exerciseF(){
    Aggregateterable<Document> doc1 = database.getCollection("review").aggregate(Arrays.asList(
        new Document("group", new Document("_id", "$business_id")
            .append("avg_stars", new Document("$avg", "$stars))),
        new Document("$out", "avg_stars")));

    Aggregateterable<Document> result = database.getCollection("business").aggregate(Arrays.asList(
        new Document("$lookup", new Document("from", "avg_stars")
            .append("localField", "business_id")
            .append("foreignField", "_id")
            .append("as", "averages")),
        new Document("$unwind", new Document("path", "$averages")
            .append("preserveNullAndEmptyArrays", true)),
        new Document("$project", new Document("_id", "$_id")
            .append("business_id", "$business_id")
            .append("name", "$name")
            .append("avg_stars",
                new Document("$filter", new Document("input",
"$averages")
                    .append("as", "average")
                    .append("cond", new
Document("avg_stars", new Document("$gte", (new Document("averages.avg_stars", 4))))))),
            new Document("$sort", new Document("name", 1))
        ));
    return result;
}

```

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

```

private void exerciseG()
{
    database.getCollection("business").
        deleteMany(new Document("stars", new Document("$eq", 2.0)));
}

```

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.

```
db.customer.insert({
  first_name: "Mateusz",
  last_name: "Kowalski",
  birthdate: ISODate("1998-01-30"),
  address: {
    street: "Mickiewicza 10",
    city: "Krakow"
  },
  phone: ["512124125", "623231352"]
  mail: "mkowalski@mail.eu"
})

db.product.insert({
  name : "Maka Lubelska",
  price : 4.5,
  category : "Food",
  units_in_stock : 1000,
  expiry_date : ISODate("2020-02-11")
})

db.product.insert({
  product_id : "makaron11",
  name : "Makaron",
  price : 6,
  category : "Food",
  units_in_stock : 500,
  expiry_date : ISODate("2020-03-01")
})

db.purchase.insert({
  purchase_id : "mkowalpuch1",
  client_id : "mkowalski1mick10krak1",
  products_list : [{
    product_id : "makalub1",
    units : 10
  },
  {
    product_id : "makaron11",
    units : 5
  }],
  price : 75,
  purchase_date : ISODate("2019-12-11")
})

db.product.insert({
  product_id : "makaron11",
  name : "Makaron",
  price : 6,
  category : "Food",
  units_in_stock : 500,
```



```
    expiry_data : ISODate("2020-03-01")
  })
```

Struktura dokumentów w kolekcjach:

customer 0.003 sec.			0	50				
Key	Value	Type						
▼ (1) ObjectId("5df513fc66e434e4...")	{ 8 fields }	Object						
_id	ObjectId("5df513fc66e434e4e15d7add")	ObjectId						
customer_id	mkowalski1mick10krak1	String						
first_name	Mateusz	String						
last_name	Kowalski	String						
birthdate	1998-01-30 00:00:00.000Z	Date						
▼ (2) address	{ 2 fields }	Object						
street	Mickiewicza 10	String						
city	Krakow	String						
▼ (2) phone	[2 elements]	Array						
[0]	512124125	String						
[1]	623231352	String						
mail	mkowalski@mail.eu	String						

product 0.009 sec.			0	50				
Key	Value	Type						
▶ (1) ObjectId("5df5270966e434e4...")	{ 7 fields }	Object						
▼ (2) ObjectId("5df5287d66e434e4...")	{ 7 fields }	Object						
_id	ObjectId("5df5287d66e434e4e15d7ae0")	ObjectId						
product_id	makaron11	String						
name	Makaron	String						
price	6.0	Double						
category	Food	String						
units_in_stock	500.0	Double						
expiry_data	2020-03-01 00:00:00.000Z	Date						

purchase 0.002 sec.			0	50				
Key	Value	Type						
▼ (1) ObjectId("5df5303066e434e4...")	{ 6 fields }	Object						
_id	ObjectId("5df5303066e434e4e15d7ae2")	ObjectId						
purchase_id	mkowalpuch1	String						
client_id	mkowalski1mick10krak1	String						
▼ (2) products_list	[2 elements]	Array						
▼ (2) [0]	{ 2 fields }	Object						
product_id	makalub1	String						
units	10.0	Double						
▼ (2) [1]	{ 2 fields }	Object						
product_id	makaron11	String						
units	5.0	Double						
price	75.0	Double						
purchase_date	2019-12-11 00:00:00.000Z	Date						