Bazy danych – NoSQL
MongoDB – zadania

Mikołaj Tomalik
27.11.2019
11.12.2019

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*). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

   ```
   db.buisiness.createIndex({"city": 1})
   db.buisiness.distinct("city”).sort();
   ```

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

   ```
   db.review.find({"date" : {"$gte" : "2011-01-01"}});
   ```

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

   ```
   db.buisiness.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.user.createIndex({"name": 1})
   db.user.find({$or:[{'votes.funny': 0},{'votes.useful': 0},]})
   .sort({'name': 1})
   ```

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

   ```
   db.tip.aggregate
   ([
   {$group: {_id: "$business_id", count: {$sum: 1}}},
   {$lookup:{from: "buisiness", localField: "_id", foreignField: "business_id", as:
   "business_tip"}},
   {$unwind: "$business_tip"},
   {$project:{"name": "$business_tip.name","count": "$count"}},
   {$match: {"$business_tip.date" : { $gte: 2012-01-01, $lte: 2012-12-31 } } },
   {$sort: {tip: 1}}
   ])
   ```

   f. Wyznacz, jaką średnia 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", avgStars: {$avg: "$stars"}}},
   {$lookup:{from: "buisiness",localField: "_id",foreignField: "business_id",as:
   "business_review"}},
   {$unwind: "$business_review"},
   {$project:{"name": "$business_review.name","avgStars": "$avgStars"}},
   { $match : { avgStars: { $gte: 4 } } }
   ])
   ```

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

```
db.buisiness.deleteMany({"stars": 2.0})
```

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

```
function insertReview(user_id, review_id, business_id, stars, text)
{
    db.review.insert(
    {
        user_id: user_id,
        review_id: review_id,
        business_id: business_id,
        stars: stars,
        text: text,
        votes:{funny: 0, useful: 0, cool: 0},
        date: new Date(),
        type: "review"
    })
}

insertReview("uid", "rid", "bid", 4, "Review text")
```

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.

```
function get_business_of_category(category)
{
    return db.buisiness.find({categories: {"$in" : [category]}})
}
var tips = get_business_of_category("Doctors")
tips.forEach(function(doc){print(doc)})
```

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

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

change_user_name("5Xh4Qc3rxhAQ_NcNtxLssQ", "Mikolaj")
```

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

```
var mapFunction = function()
{
    var key = this.business_id;
        var value ={sum: this.review_count,count: 1}
    emit(key, value);
};

var reduceFunction = function(key, values)
{
    var result ={sum: 0,count: 0};
    values.forEach(function(value)
        {result.sum += value.sum;result.count += value.count;})
    return result;
}

var finalizeFunction = function(key, reducedValue)
{
    var avgReviews = reducedValue.sum / reducedValue.count;
    return avgReviews;
}

db.buisiness.mapReduce(
    mapFunction,
    reduceFunction,
    {
        out: "reviews_per_business_avg",
        finalize: finalizeFunction,
    }
)
```

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> getSortedCities()
{
    List<String> result =
            db.getCollection("buisiness")
            .distinct("city", String.class)
            .into(new ArrayList<>());
    Collections.sort(result);
    return result;
}
```

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

```
private long reviewsAfter2010()
{
    Bson filter = (Filters.gte("date", "2011-01-01"));
    return db.getCollection("review").countDocuments(filter);
}
```

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

```
private List<Document> closedBusinesses()
    {
       return db.getCollection("buisiness")
                .find(eq("open", false))
                .projection(fields(include("name", "full_address", "stars"), excludeId()))
                .into(new ArrayList<>());
    }
```

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 FindIterable<Document> usersWithVotes()
{
    db.getCollection("user").createIndex(Indexes.ascending("name"));
    Bson filter = or(eq("votes.funny", 0), eq("votes.useful", 0));
    return db.getCollection("user")
             .find(filter)
             .sort(ascending("name"));
}
```

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

```
private AggregateIterable<Document> tipsFrom2012PerBusiness()
{
    Bson match = match(Filters.regex("date", "2012"));
    Bson group = group("$business_id", Accumulators.sum("count", 1));
    Bson lookupOperation = lookup("buisiness", "_id", "business_id", "business_tip");
    Bson unwind = unwind("$business_data");
    Bson project = project(fields(include("business_tip.name", "count"), excludeId()));
    Bson sort = Aggregates.sort(ascending("count"));
    return db.getCollection("tip")
             .aggregate(Arrays.asList(match, group, lookupOperation, unwind, project, sort));
}
```

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 AggregateIterable<Document> avgStarsPerBusiness()
{
    Bson match = match(Filters.gte("avgStars", 4));
    Bson group = group("$business_id", Accumulators.avg("avgStars", "$stars"));
    Bson lookupOperation = lookup("buisiness", "_id", "business_id", "business_review");
    Bson unwind = unwind("$business_review");
    Bson project = project(fields(include("business_review.name", "avgStars"), excludeId()));
        return db.getCollection("review")
                 .aggregate(Arrays.asList(group, lookupOperation, unwind, project, match));
}
```

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

```
private void deleteBusinessesWith2Stars()
{
    db.getCollection("buisiness").deleteMany(eq("stars", 2));
}
```

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.

Order:


```
{
    "_id" : ObjectId("5ded067997f0cbd201d64a55"),
    "username" : "Mishra",
    "product_id" : "1",
    "date" : "Sun Dec 08 2019 15:19:37 GMT+0100"
}
```


Product:


```
{
    "_id" : "1",
    "name" : "Teddy",
    "units_in_stock" : 15.0,
    "description" : "Awesome",
    "orders" : [],
    "orderes" : [
        {
            "$ref" : "orderes",
            "$id" : "Mishra"
        }
    ]
}
```


User:


```
{
    "_id" : "Mishra",
    "fullname" : {
        "firstname" : "Mikolaj",
        "lastname" : "Tomalik"
    },
    "address" : {
        "city" : "Krakow",
        "street" : "Czarnowiejska",
        "number" : 37.0
    },
    "password" : "1234",
    "email" : "Mishra@gmail.com",
    "ordered_products" : [
        {
            "$ref" : "ordered_products",
            "$id" : "1"
        }
    ]
}
```