Bazy danych – NoSQL
MongoDB – zadania

Autor zadań: Piotr Wróbel
Data laboratorium: 20.11.2019 r.
Data wykonania: 8.12.2019 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).

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

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

   ```
   db.review.find({$or: [{date: /2011/}, {date: /2012/}]});
   ```

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

   ```
   db.business.find({open: true}, {business_id:1, 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.

   ```
   // założony indeks w celu przyspieszenia zapytania
   db.user.ensureIndex({"name": 1});

   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.

   ```
   // indeks założony w celu j.w.
   db.business.ensureIndex({"name": 1});

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

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

```
var cursor = db.review.aggregate([
    {"$group":
        {_id: "$business_id",
         avg_stars: {$avg: "$stars"}
    }},
    {"$match": {avg_stars: {$lt: 3.0}}}
]);

cursor.forEach(function(bus) {
        db.business.remove({business_id: bus._id});
});
```

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

```
function insertTip(
    user_id, text, business_id, date) {
        db.tip.insert({
            user_id: user_id,
            text: text,
            business_id: business_id,
            likes: 0,
            date: ISODate(date),
            type: "tip"
        });
}

insertTip("5Xh4Qc3rxhAQ_NcNtxLssQ", "bardzo fajne, podoba mi sie","JwUE5GmEO-sH1FuwJgKBlQ",
    "2018-01-02");
```

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

```
function findTipByPhrase(phrase) {
    db.tip.find({text: {$regex: ".*" + phrase + ".*"}}).forEach(function(res) { print(res);
        });
}

findTipByPhrase("bardzo fajne");
```

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

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

modifyBusinessName("JwUE5GmEO-sH1FuwJgKBlQ", "Zmiana nazwy");
```

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

```
var mapF = function() {
    emit(this.user_id, 1);
};

var reduceF = function(user_id, reviewNum) {
    return Array.sum(reviewNum);
};

db.review.mapReduce(mapF, reduceF, {out: "reviews_per_user"});

db.reviews_per_user.aggregate([
    {"$group":
        {_id: null,
         avg_rev: {$avg: "$value"}}}
]);
```

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).

```
private void distinctBusinessCities() {
  DBCollection businessCollection = db.getCollection("business");
  for(Object o: businessCollection.distinct("city")) {
    System.out.println(o);
}
}
```

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

```
private void reviewsBetweenYears(int yearFrom, int yearTo) {
  DBCollection reviewCollection = db.getCollection("review");

  Pattern yearFromPattern = Pattern.compile(String.valueOf(yearFrom), Pattern.
      CASE_INSENSITIVE);
  Pattern yearToPattern = Pattern.compile(String.valueOf(yearTo), Pattern.
      CASE_INSENSITIVE);

  ArrayList<BasicDBObject> orArgs = new ArrayList<>();
  orArgs.add(new BasicDBObject("date", yearFromPattern));
  orArgs.add(new BasicDBObject("date", yearToPattern));
  BasicDBObject query = new BasicDBObject("$or", orArgs);

  DBCursor result = reviewCollection.find(query);

  try {
```

```
    while(result.hasNext()) {
      System.out.println(result.next());
    }
  } finally {
    result.close();
  }
}
```

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

```
private void openedBusiness() {
  DBCollection businessConnection = db.getCollection("business");

  BasicDBObject query = new BasicDBObject("open", true);

  DBObject projectionFields = new BasicDBObject("business_id", 1);
  projectionFields.put("name", 1);
  projectionFields.put("full_address", 1);

  DBCursor result = businessConnection.find(query, projectionFields);

  try {
    while(result.hasNext()) {
      System.out.println(result.next());
    }
  } finally {
    result.close();
  }
}
```

(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 usersWithPositiveVotes() {
  DBCollection userCollection = db.getCollection("user");

  userCollection.createIndex(new BasicDBObject("name", 1));

  ArrayList<BasicDBObject> orArgs = new ArrayList<>();
  orArgs.add(new BasicDBObject("votes.funny", new BasicDBObject("$gte", 1)));
  orArgs.add(new BasicDBObject("votes.useful", new BasicDBObject("$gte", 1)));
  orArgs.add(new BasicDBObject("votes.cool", new BasicDBObject("$gte", 1)));

  BasicDBObject query = new BasicDBObject("$or", orArgs);
  BasicDBObject sortingOrder = new BasicDBObject("name", 1);

  DBCursor result = userCollection.find(query).sort(sortingOrder);

  try {
    while(result.hasNext()) {
      System.out.println(result.next());
    }
  } finally {
    result.close();
  }
}
```

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

```
private void businessTipsNumber(int year) {
  DBCollection businessCollection = db.getCollection("business");
  DBCollection tipCollection = db.getCollection("tip");

  businessCollection.createIndex(new BasicDBObject("name", 1));

  Pattern yearPattern = Pattern.compile(String.valueOf(year), Pattern.CASE_INSENSITIVE);

  DBObject match = new BasicDBObject("$match", new BasicDBObject("date", yearPattern));

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

  DBObject lookupFields = new BasicDBObject("from", "business");
  lookupFields.put("localField", "_id");
  lookupFields.put("foreignField", "business_id");
  lookupFields.put("as", "B");
  DBObject lookup = new BasicDBObject("$lookup", lookupFields);

  DBObject projectFields = new BasicDBObject("_id", 1);
  projectFields.put("count", 1);
  projectFields.put("B.name", 1);
  DBObject project = new BasicDBObject("$project", projectFields);

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

  List<DBObject> pipeline = Arrays.asList(match, group, lookup, project, sort);
  AggregationOutput output = tipCollection.aggregate(pipeline);

  for(DBObject result: output.results()) {
    System.out.println(result);
  }
}
```

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

```
private void businessStars() {
  DBCollection reviewCollection = db.getCollection("review");

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

  DBObject sort = new BasicDBObject("$sort", new BasicDBObject("avg_stars", -1));

  DBObject lookupFields = new BasicDBObject("from", "business");
  lookupFields.put("localField", "_id");
  lookupFields.put("foreignField", "business_id");
  lookupFields.put("as", "B");
  DBObject lookup = new BasicDBObject("$lookup", lookupFields);
```

```
  List<DBObject> pipeline = Arrays.asList(group, sort, lookup);
  AggregationOutput output = reviewCollection.aggregate(pipeline);

  for(DBObject result: output.results()) {
    System.out.println(result);
  }
}
```

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

```
private void deleteBusinessUnderThreshold(double threshold) {
  DBCollection reviewCollection = db.getCollection("review");

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

  DBObject match = new BasicDBObject("$match", new BasicDBObject("avg_stars", new
      BasicDBObject("$lt", threshold)));

  List<DBObject> pipeline = Arrays.asList(group, match);
  AggregationOutput output = reviewCollection.aggregate(pipeline);
  DBCollection businessCollection = db.getCollection("business");

  for(DBObject result: output.results()) {
    businessCollection.remove(new BasicDBObject("business_id", result.get("_id")));
  }
}
```

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.

(a) studenci – każdy student poza podstawowymi danymi osobowymi (imię, nazwisko, adres, pesel) ma także pola z danymi uczelnianymi (numer legitymacji, wydział, kierunek i rodzaj studiów, bieżący rok oraz listę kursów na które jest zapisany, przykładowe dane:

```
student1 = {
    _id: 1,
    first_name: "Michał",
    last_name: "Kowalski",
    pesel: "94091801234",
    enrolled_courses: [
        {course_id: 11, type: "obligatory"},
        {course_id: 12, type: "facultative"}
    ],
    student_card_ID: 1100119,
    address: {
  street: "Mickiewicza 29/1",
  postal_code: "30-856",
  city: "Kraków",
  country: "Poland"
    },
    faculty: "WIET",
    field_of_study: "informatyka",
    study_type: {
```

```
  mode: "dzienne",
  step: "inżynierskie"
    }
    year: "III"
}
```

(b) przedmioty – każdy przedmiot ma swoją nazwę, liczbę punktów ECTS, oznaczenie czy kończy się egzaminem, listę terminów i miejsc zajęć, wraz z oznaczeniem co ile tygodni się odbywają, liczbę godzin poszczególnych składowych kursu oraz listę kursów, które należy mieć zaliczone przed przystąpieniem do danego, przykładowe dane:

```
course11 = {
    _id: 11,
    name: "Bazy danych",
    ects: 3,
    has_exam: false,
    classes_terms: [
        {room_id: 21,
         from: "16:15",
         to: "17:45",
         day: "Wednesday",
   type: "laboratory",
         week_interval: 2
        },
        {room_id: 22,
         from: "12:50",
         to: "14:20",
         day: "Thursday",
   type: "lecture"
         week_interval: 2
        }
    ],
    number_of_hours: {
        lecture: 15,
        laboratory: 15
    }
    prerequisites: [
        {course_id: 10},
        {course_id: 15}
    ]
}
```

(c) sale zajęciowe – każda sala ma swój numer, oznaczenie budynku, w którym się znajduje, liczbę dostępnych miejsc oraz swój typ, przykładowe dane;

```
room21 = {
    _id: 21,
    name: "4.38",
    building: "D17",
    number_of_places: 15,
    room_type: "computer laboratory"
}
```