**Bazy danych – NoSQL**

**MongoDB – zadania**

Autor rozwiązań: Tomasz Zawadzki

Data laboratorium: 27.11.2019 r.

Data wykonania: 15.12.2019 r.

1. Wykorzystując bazę danych **yelp dataset** wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty:

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

db.business.distinct('city').sort()

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

db.review.find({ 'date': { $gte: '2011-' } }).count()

* 1. 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 }

)

* 1. 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.find({

$or: [

{'votes.funny': 0},

{'votes.useful': 0}

]

}).sort({ name: 1 })

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

db.tip.aggregate([

{

$match: {

'date': {

$regex: '^2012-'

}

}

},

{

$group: {

\_id: '$business\_id',

count: {

$sum: 1

}

},

},

{

$lookup: {

from: 'business',

localField: '\_id',

foreignField: 'business\_id',

as: 'business\_info'

}

},

{

$unwind: '$business\_info'

},

{

$project: {

name: '$business\_info.name',

count: '$count'

}

},

{

$sort: {

count: 1

}

}

])

* 1. 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([

{

$match: {

stars: {

$gte: 4.0

}

}

},

{

$group: {

\_id: '$business\_id',

avg\_stars: {

$avg: '$stars'

}

}

},

{

$lookup: {

from: 'business',

localField: '\_id',

foreignField: 'business\_id',

as: 'business\_info'

}

},

{

$unwind: '$business\_info'

},

{

$project: {

name: '$business\_info.name',

avg\_stars: '$avg\_stars'

}

},

])

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

db.business.remove({ stars: 2.0 })

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

function addReview(user\_id, review\_id, stars, text, business\_id) {

db.review.insert({

votes: {

funny: 0,

useful: 0,

cool: 0

},

user\_id: user\_id,

review\_id: review\_id,

stars: stars,

date: ISODate(),

text: text,

type: 'review',

business\_id: business\_id

});

}

addReview(

'qtrmBGNqCvupHMHL\_bKFgQ',

'9uHzyOu5CTCPl1q6cfvO7u',

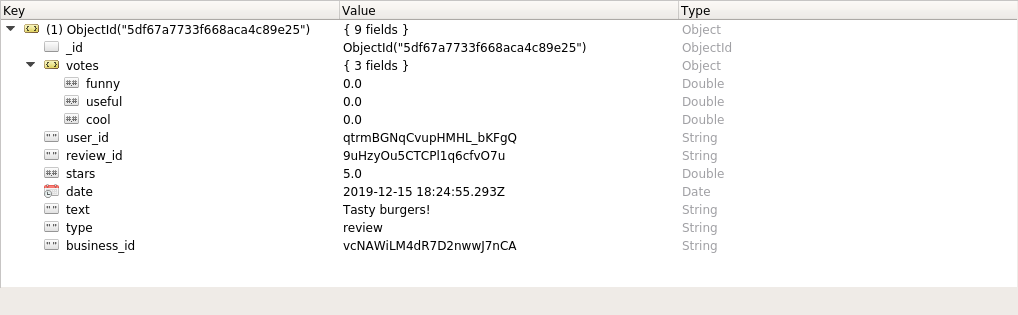
5.0,

'Tasty burgers!',

'vcNAWiLM4dR7D2nwwJ7nCA'

)

db.review.find({ review\_id: '9uHzyOu5CTCPl1q6cfvO7u' })



1. 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 findBusinessByCategory(feature) {

return db.business.find({

categories: {

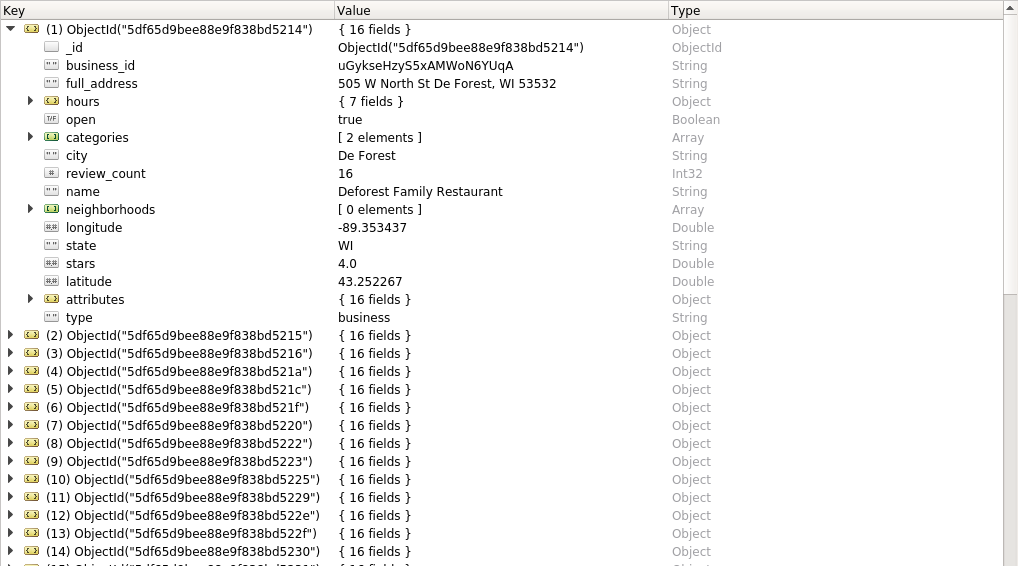
$in: [feature]

}

})

}

findBusinessByCategory('Restaurants')



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

function changeUserName(\_id, new\_name) {

db.user.update(

{ \_id: \_id },

{ $set: { name: new\_name }}

)

}

object\_id = ObjectId('5df65e24ee88e9f838d5cbf2')

changeUserName(object\_id, 'Tomek')

db.user.find({ \_id: object\_id })



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

db.tip.mapReduce(

function() {

emit(this.business\_id, 1);

},

function(business\_id, tips) {

return tips.length;

},

{ out: 'business\_tips\_count' }

)

db.business\_tips\_count.aggregate([

{

$group: {

\_id: null,

average: { $avg: '$value' }

}

}

])

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

import pymongo

client = pymongo.MongoClient()

db = client.TomaszZawadzki

if \_\_name\_\_ == '\_\_main\_\_':

a()

b()

c()

d()

e()

f()

g()

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

def a():

cities = sorted(db.business.distinct('city'))

print('\n'.join(cities))

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

def b():

print(db.review.find({ 'date': { '$gte': '2011-' } }).count())

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

def c():

cursor = db.business.find({

'open': False

}, {

'name': 1,

'full\_address': 1,

'stars': 1

})

for business in cursor:

print(business)

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

def d():

cursor = db.user.find({

'$or': [

{'votes.funny': 0},

{'votes.useful': 0}

]

}).sort('name', 1)

for user in cursor:

print(user)

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

def e():

cursor = db.tip.aggregate([

{

'$match': {

'date': { '$regex': '^2012-' }

}

},

{

'$group': {

'\_id': '$business\_id',

'count': { '$sum': 1 }

},

},

{

'$lookup': {

'from': 'business',

'localField': '\_id',

'foreignField': 'business\_id',

'as': 'business\_info'

}

},

{

'$unwind': '$business\_info'

},

{

'$project': {

'name': '$business\_info.name',

'count': '$count'

}

},

{

'$sort': { 'count': 1 }

}

])

for business in cursor:

print(f"{business['name']}: {business['count']}")

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

def f():

cursor = db.review.aggregate([

{

'$match': {

'stars': {

'$gte': 4.0

}

}

},

{

'$group': {

'\_id': '$business\_id',

'avg\_stars': {

'$avg': '$stars'

}

}

},

{

'$lookup': {

'from': 'business',

'localField': '\_id',

'foreignField': 'business\_id',

'as': 'business\_info'

}

},

{

'$unwind': '$business\_info'

},

{

'$project': {

'name': '$business\_info.name',

'avg\_stars': '$avg\_stars'

}

},

])

for business in cursor:

print(f"{business['name']}: {business['avg\_stars']}")

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

def g():

db.business.remove({ 'stars': 2.0 })

1. 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.

customer\_id = ObjectId()

product1\_id = ObjectId()

product2\_id = ObjectId()

order\_id = ObjectId()

db.customer.insert({

customer\_id: customer\_id,

name: 'Tomek Zawadzki',

address: {

street: {

name: 'Kawiory',

building\_no: '21'

},

zipCode: '31-055',

city: 'Kraków'

},

verified: true,

points: 100

})

db.product.insert({

product\_id: product1\_id,

name: 'Logitech G29 Racing Wheel',

price: 849.00,

available: true,

units\_on\_stock: 7

})

db.product.insert({

product\_id: product2\_id,

name: 699.00,

available: true,

units\_on\_stock: 3

})

db.order.insert({

order\_id: order\_id,

customer\_id: customer\_id,

items: [

{product\_id: product1\_id, units: 1},

{product\_id: product2\_id, units: 2},

],

submitted\_on: ISODate(),

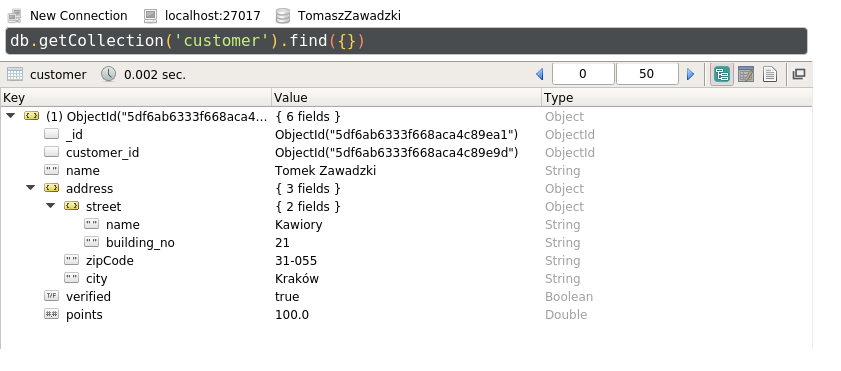
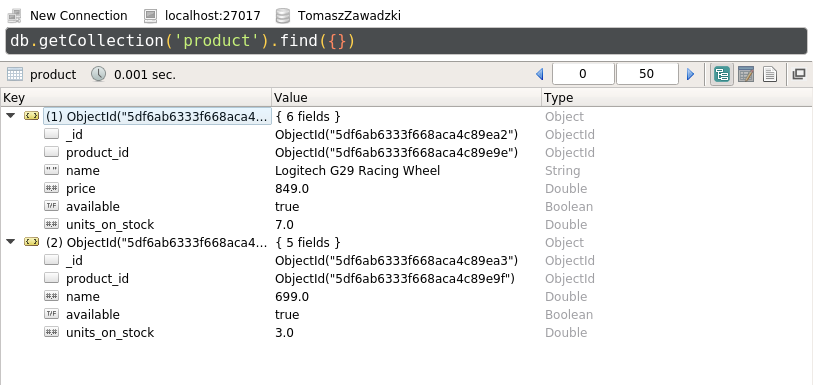
remarks: null

})

db.customer.find()

db.product.find()

db.order.find()

2. 
3. 
4. 