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:

   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()
```

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

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

   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 na podstawie imienia użytkownika.

```
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([
    {
        $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
        }
    }
])
```

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([
    {
        $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'
        }
    },
])
```

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

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

2. 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'
)
```

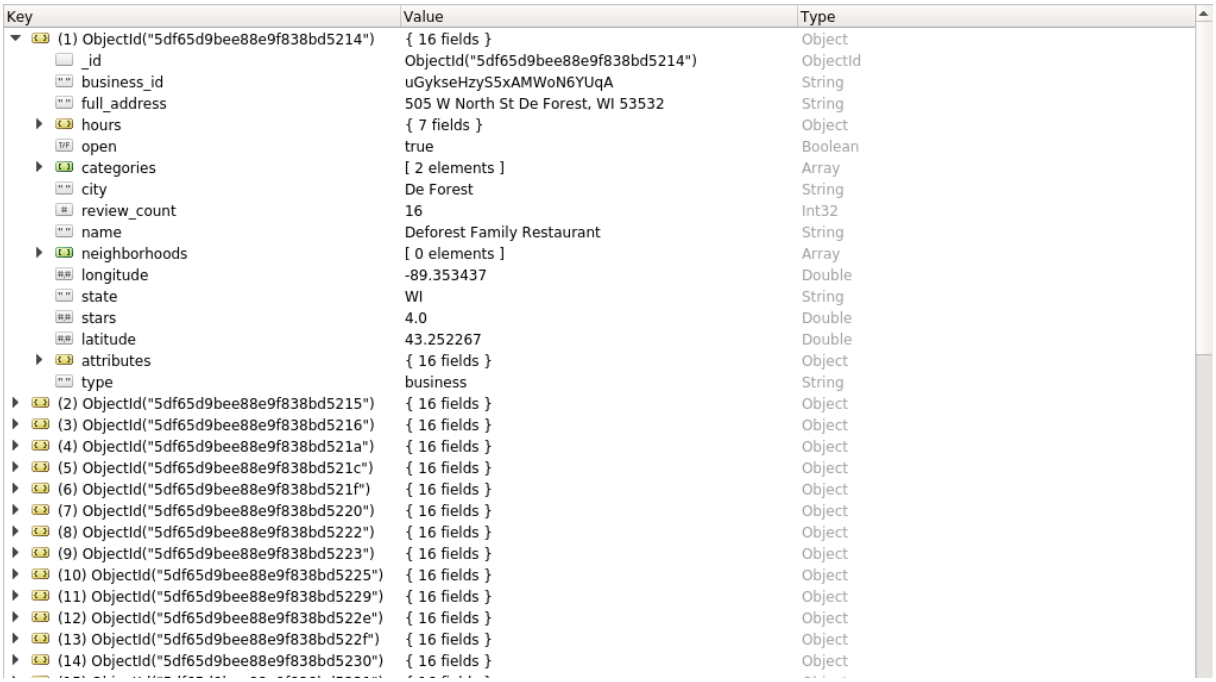
```
db.review.find({ review_id: '9uHzyOu5CTCPl1q6cfvO7u' })
```

Key	Value	Type
(1) ObjectId("5df67a7733f668aca4c89e25")	{ 9 fields }	Object
_id	ObjectId("5df67a7733f668aca4c89e25")	ObjectId
votes	{ 3 fields }	Object
funny	0.0	Double
useful	0.0	Double
cool	0.0	Double
user_id	qtrmBGNqCvupHMHL_bKFgQ	String
review_id	9uHzyOu5CTCPl1q6cfvO7u	String
stars	5.0	Double
date	2019-12-15 18:24:55.293Z	Date
text	Tasty burgers!	String
type	review	String
business_id	vcNAWiLM4dR7D2nwwJ7nCA	String

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 findBusinessByCategory(feature) {
    return db.business.find({
        categories: {
            $in: [feature]
        }
    })
}
```

```
findBusinessByCategory('Restaurants')
```



Key	Value	Type
(1) ObjectId("5df65d9bee88e9f838bd5214")	{ 16 fields }	Object
_id	ObjectId("5df65d9bee88e9f838bd5214")	ObjectId
business_id	uGykseHzyS5xAMWoN6YUqA	String
full_address	505 W North St De Forest, WI 53532	String
hours	{ 7 fields }	Object
open	true	Boolean
categories	[2 elements]	Array
city	De Forest	String
review_count	16	Int32
name	Deforest Family Restaurant	String
neighborhoods	[0 elements]	Array
longitude	-89.353437	Double
state	WI	String
stars	4.0	Double
latitude	43.252267	Double
attributes	{ 16 fields }	Object
type	business	String
(2) ObjectId("5df65d9bee88e9f838bd5215")	{ 16 fields }	Object
(3) ObjectId("5df65d9bee88e9f838bd5216")	{ 16 fields }	Object
(4) ObjectId("5df65d9bee88e9f838bd521a")	{ 16 fields }	Object
(5) ObjectId("5df65d9bee88e9f838bd521c")	{ 16 fields }	Object
(6) ObjectId("5df65d9bee88e9f838bd521f")	{ 16 fields }	Object
(7) ObjectId("5df65d9bee88e9f838bd5220")	{ 16 fields }	Object
(8) ObjectId("5df65d9bee88e9f838bd5222")	{ 16 fields }	Object
(9) ObjectId("5df65d9bee88e9f838bd5223")	{ 16 fields }	Object
(10) ObjectId("5df65d9bee88e9f838bd5225")	{ 16 fields }	Object
(11) ObjectId("5df65d9bee88e9f838bd5229")	{ 16 fields }	Object
(12) ObjectId("5df65d9bee88e9f838bd522e")	{ 16 fields }	Object
(13) ObjectId("5df65d9bee88e9f838bd522f")	{ 16 fields }	Object
(14) ObjectId("5df65d9bee88e9f838bd5230")	{ 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 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 })
```

Key	Value	Type
(1) ObjectId("5df65e24ee88e9f838...	{ 12 fields }	Object
_id	ObjectId("5df65e24ee88e9f838d5cbf2")	ObjectId
yelping_since	2012-02	String
votes	{ 3 fields }	Object
funny	1	Int32
useful	5	Int32
cool	0	Int32
review_count	6	Int32
name	Tomek	String
user_id	qtrmBGNqCvupHMHL_bKFgQ	String
friends	[0 elements]	Array
fans	0	Int32
average_stars	3.83	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.

```
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' }
        }
    }
])
```

6. 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()
```

a. 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))
```

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

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

c. 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)
```

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.

```
def d():
    cursor = db.user.find({
        '$or': [
            {'votes.funny': 0},
            {'votes.useful': 0}
        ]
    }).sort('name', 1)
    for user in cursor:
        print(user)
```

e. 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']}")
```

f. 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']}")
```

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

```
def g():
    db.business.remove({ 'stars': 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.

```
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()
```

New Connection localhost:27017 TomaszZawadzki

```
db.getCollection('customer').find({})
```

customer 0.002 sec. 0 50

Key	Value	Type
(1) ObjectId("5df6ab6333f668aca4...	{ 6 fields }	Object
_id	ObjectId("5df6ab6333f668aca4c89ea1")	ObjectId
customer_id	ObjectId("5df6ab6333f668aca4c89e9d")	ObjectId
name	Tomek Zawadzki	String
address	{ 3 fields }	Object
street	{ 2 fields }	Object
name	Kawiory	String
building_no	21	String
zipCode	31-055	String
city	Kraków	String
verified	true	Boolean
points	100.0	Double

New Connection localhost:27017 TomaszZawadzki

```
db.getCollection('product').find({})
```

product 0.001 sec. 0 50

Key	Value	Type
(1) ObjectId("5df6ab6333f668aca4...	{ 6 fields }	Object
_id	ObjectId("5df6ab6333f668aca4c89ea2")	ObjectId
product_id	ObjectId("5df6ab6333f668aca4c89e9e")	ObjectId
name	Logitech G29 Racing Wheel	String
price	849.0	Double
available	true	Boolean
units_on_stock	7.0	Double
(2) ObjectId("5df6ab6333f668aca4...	{ 5 fields }	Object
_id	ObjectId("5df6ab6333f668aca4c89ea3")	ObjectId
product_id	ObjectId("5df6ab6333f668aca4c89e9f")	ObjectId
name	699.0	Double
available	true	Boolean
units_on_stock	3.0	Double

New Connection localhost:27017 TomaszZawadzki

```
db.getCollection('order').find({})
```

order 0.002 sec. 0 50

Key	Value	Type
(1) ObjectId("5df6ab6333f668aca4...	{ 6 fields }	Object
_id	ObjectId("5df6ab6333f668aca4c89ea4")	ObjectId
order_id	ObjectId("5df6ab6333f668aca4c89ea0")	ObjectId
customer_id	ObjectId("5df6ab6333f668aca4c89e9d")	ObjectId
items	[2 elements]	Array
[0]	{ 2 fields }	Object
product_id	ObjectId("5df6ab6333f668aca4c89e9e")	ObjectId
units	1.0	Double
[1]	{ 2 fields }	Object
product_id	ObjectId("5df6ab6333f668aca4c89e9f")	ObjectId
units	2.0	Double
submitted_on	2019-12-15 21:53:39.307Z	Date
remarks	null	Null