

JEGYZŐKÖNYV

Modern adatbázis rendszerek MSc

2022. tavasz féléves feladat

MongoDB

Készítette: **Toronya Bertalan**

Neptunkód: **F8GVBF**

A feladat leírása:

1. Alap műveletek MongoDB-vel

- a,** Hozzunk létre egy mdbgyak adatbázist.
- b,** Készítsünk egy auto kollekciót mely rendelkezik egy típus, szín, ár és gyártás_év adattagokkal.
- c,** Vegyünk fel az auto táblába egy, majd egyszerre több rekordot is.
- d,** Írassuk ki azoknak az autóknak a típusát, amelyeknek az ára több mint 1.000.000 és számoljuk meg őket.
- e,** Az auto kollekció rekordjaihoz fűzzünk hozzá egy új mezőt „állapot” néven.
- f,** A 2004 előtt gyártott autók állapotát állítsuk sérültre.
- g,** Töröljük a sérült állapotú autó rekordokat az auto kollekcióból.

2. Bonyolultabb műveletek

a, készítsünk egy tulajdonos kollekciót mely név és kor mezőkkel rendelkezik.

b, Készítsünk egy tárolt függvényt „save_tulaj” néven, amely új tulajdonosokat tud felvenni tulajdonos kollekcióba, úgy, hogy az ID-jét is mi adjuk meg.

c, Készítsünk tárolt függvényt „save_auto” néven, amellyel új rekordot tudunk felvenni az auto kollekcióba és adjunk hozzá egy tulaj mezőt is mely a tulajdonos tábla elemeinek az ID-jét tartalmazza.

d, Készítsünk tárolt függvényt „getTulajByName” néven, amely a paraméterként kapott tulajdonos neve alapján kilistázza az összes olyan nevű tulajdonost.

e, Írassuk ki az összes autót, amely „Pál” -hoz kapcsolódik.

f, Csökkentsük a sérült állapotú autók árát 300.000 -el.

g, A **\$where** használatával, számoljuk meg azokat az autókat, amelyeknek az ára kisebb mint 1.000.000. és a gyártás éve 2010 előtti.

h, Vegyünk fel még néhány auto rekordot, és írassuk ki típus szerint csoportosítva az átlag árat.

i, Írassuk ki csökkenő sorrendben, hogy típusonként mennyi autó rendelkezik sérült státusszal.

3. JAVA feladatok

a, Először is hozzunk létre egy MAVEN projektet és egy Teszt osztály az mdbgyak csomagban.

A pom.xml-be illesszük be a

```
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongo-java-driver</artifactId>
  <version>3.12.10</version>
</dependency>
```

Függőséget.

b, Kapcsolódjunk az mdbgyak adatbázishoz és jelöljük ki az auto kollekciót.

c, Listázzuk ki az auto kollekció rekordjait.

d, Vegyünk fel egy új elemet az auto kollekcióba. Típusa legyen JavaInserted (insertOne())

e, Vegyünk fel egyszerre több elemet az auto kollekcióba. (insertMany())

f, A JavaInserted típusú autók állapotát állítsuk sérültre.

g, Töröljük a JavaInserted típusú rekordot.

A feladat elkészítésének lépései:

A feladatok elkezdéséhez szükség van a MongoDB telepítésére. A gyakorlaton a MongoDB community editionnal dolgoztunk.

Érdemes a MongoDB Compass-t is feltelepíteni mert azon keresztül igazán könnyen adatbázist hozhatunk létre.

A Robo3T szoftver egy GUI és parancs felületet is biztosít mely megkönnyíti a MongoDB-s parancsok kipróbálását.

A feladat 3 részre van bontva. Az 1.részben a MongoDB-vel elvégezhető alap parancsokat és alpműveleteket lehet gyakorolni.

A 2.részben bonyolultabb MongoDB-s műveleteket lehet gyakorolni.

A 3.részben pedig egy MongoDB JAVA drivert használva java programozási nyelven alpműveleteket mutattam be.

MongoDB java driver: **mongo-java-driver**

Maven java projectben a pom.xml-ben dependencyként szerepel a 3.feladatban.

1. Alap műveletek

a, Hozzunk létre egy mdbgyak adatbázist.

MognoDB Compassban 1 gombnyomás.

b, Készítsünk egy auto kollekciót mely rendelkezik egy típus, szín, ár és gyártás_év adattagokkal.

```
db.createCollection("auto") use auto
```

c, Vegyünk fel az auto táblába egy, majd egyszerre több rekordot is.

1., Egy rekord felvétele:

```
db.auto.insert({'típus' : 'Opel', 'szin' : 'piros' , 'ár' : 500011, 'gyév':2001});
db.auto.insert({'típus' : 'Suzuki', 'szin' : 'sárga' , 'ár' : 600000, 'gyév':2005, "állapot" : "jó"});
db.auto.find();
```

2 , Több rekord felvétele:

```
db.auto.insertMany([{'típus' : 'Suzuki', 'szin' : 'sárga' , 'ár' : 600000, 'gyév':2005} ,
                    {'típus' : 'Toyota', 'szin' : 'fehér' , 'ár' : 1600000, 'gyév':2010},
                    {'típus' : 'Wolkswagen', 'szin' : 'zöld' , 'ár' : 904000, 'gyév':2007},
                    {'típus' : 'Audi', 'szin' : 'kék' , 'ár' : 3220000, 'gyév':2014},
                    {'típus' : 'Opel', 'szin' : 'piros' , 'ár' : 500011, 'gyév':2001} ] );
```

d, Írassuk ki azoknak az autóknak a típusát, amelyeknek az ára több mint 1.000.000 és számoljuk meg őket

```
db.auto.find({'ár' : {'$gt': 1000000}},{'típus' : 1 , _id : 0}) // kiírat
db.auto.find({'ár' : {'$gt': 1000000}},{'típus' : 1 , _id : 0}).count(); //megszámol
```

e, A már létező rekordokhoz adjunk hozzá egy új mezőt „állapot” néven és állítsuk „jó”-ra.

```
db.auto.updateMany ({},{ "$set":{"állapot": "jó"}})
```

f, A 2004 előtt gyártott autók állapotát állítsuk sérültre.

```
db.auto.updateMany ({'gyév' : {'$lt' : 2004}}, {"$set":{"állapot": "sérült"}})
```

g, A sérült állapotú autó rekordokat töröljük ki a táblából.

```
db.auto.deleteMany({'állapot' : "sérült"});
```

//Több autó felvétele

```
db.auto.insertMany([{'tipus' : 'Suzuki', 'szin' : 'sárga', 'ár' : 600000, 'gyév':2005},  
    {'tipus' : 'Toyota', 'szin' : 'fehér', 'ár' : 1600000, 'gyév':2010},  
    {'tipus' : 'Wolkswagen', 'szin' : 'zöld', 'ár' : 904000, 'gyév':2007},  
    {'tipus' : 'Audi', 'szin' : 'kék', 'ár' : 3220000, 'gyév':2014},  
    {'tipus' : 'Opel', 'szin' : 'piros', 'ár' : 500011, 'gyév':2001}] )
```

```
db.auto.insertMany([{'tipus' : 'Opel', 'szin' : 'sárga', 'ár' : 600000, 'gyév':2005},{'tipus' : 'Opel',  
'szin' : 'fehér', 'ár' : 1600000, 'gyév':2010},  
    {'tipus' : 'Wolkswagen', 'szin' : 'zöld', 'ár' : 904000, 'gyév':2007},  
    {'tipus' : 'Toyota', 'szin' : 'kék', 'ár' : 3220000, 'gyév':2014},  
    {'tipus' : 'Toyota', 'szin' : 'piros', 'ár' : 500011, 'gyév':2001}])
```

```
db.auto.find()
```

2. Bonyolultabb műveletek

a, készítsünk egy tulajdonos kollekciót mely név és kor mezőkkel rendelkezik.

```
db.createCollection(„tulajdonos”);
```

b, Készítsünk egy tárolt függvényt, amely új tulajdonosokat tud felvenni a tulajdonos kollekcióba, úgy, hogy az ID-jét is mi adjuk meg.

```
db.system.js.save(  
  {  
    _id : "save_tulaj",  
    value: function(id,nev, kor)  
    {  
      db.tulajdonos.insert({"_id": id, "név": nev, "kor": kor})  
    }  
  });
```

```
db.loadServerScripts();
```

```
save_tulaj("T5","Pál", 20);
```

```
db.tulajdonos.find()
```

c, Készítsünk tárolt függvényt „save_auto” néven, amellyel új autó rekordot tudunk felvenni az auto kollekcióba és adjunk hozzá egy tulaj mezőt is mely a tulajdonos tábla elemeinek az ID-jét tartalmazza.

```
db.system.js.save(  
  {  
    _id : "save_auto",  
    value: function(tipus, szin, ar, gyart_ev, allapot , tulaj )  
    {  
      db.auto.insertOne({"tipus": tipus, "szin": szin, "ár": ar, "gyév": gyart_ev, "állapot": allapot,  
"tulaj":  
tulaj})  
    }  
  });  
db.loadServerScripts();
```

```
save_auto("auto2","fehér",3241,2010, "jó", "T5");
```


d, Készítsünk tárolt függvényt „getTulajByName” néven, amely a paraméterként kapott tulajdonos neve alapján kilistázza az összes olyan nevű tulajdonost.

```
db.system.js.save(  
  {  
    _id : "getTulajByName",  
    value: function(nev)  
    {  
      var n = db.tulajdonos.find({"név": nev})    while(n.hasNext()){  
        print(n.next());  
      }  
    }  
  }  
);
```

e, Írassuk ki az összes autót, amely „Pál” -hoz kapcsolódik.

```
var oi = db.tulajdonos.findOne({"név":"Pál"})  
  
db.auto.find({"tulaj":oi._id})
```

f, Csökkentsük a sérült állapotú autók árát 300.000 -el.

```
db.auto.find().forEach( function(obj){  
  if(obj.áallapot == "sérült"){  
    db.auto.update({_id: obj._id}, {$inc : {'ár': -300000}});  
  }  
});  
db.auto.find();
```

g, A \$where használatával, számoljuk meg azokat az autókat, amelyeknek az ára kisebb mint 1.000.000. és a gyártás éve 2010 előtti.

```
db.auto.find({"$where" : function(){  
  if(this.ár < 1000000 && this.gyév < 2010 )  
    return true;  
  else  
    return false;  
}}).count()
```

h, Vegyünk fel még néhány auto rekordot, és írassuk ki típus szerint csoportosítva az átlag árat.

```
db.auto.insertMany([{"tipus": "BMW" , "szin" : "zöld", "ár" : 12000000, "gyév" : 2008, "állapot" : "sérült"},
    {"tipus": "Audi" , "szin" : "fekete", "ár" : 4001100, "gyév" : 2007, "állapot" : "jó"},
    {"tipus": "Opel" , "szin" : "zöld", "ár" : 12000000, "gyév" : 2008, "állapot" : "sérült"},
    {"tipus": "Toyota" , "szin" : "Fehér", "ár" : 4221100, "gyév" : 2000, "állapot" : "sérült"},
    {"tipus": "BMW" , "szin" : "zöld", "ár" : 6001100, "gyév" : 2008, "állapot" : "jó"}]);

db.auto.aggregate([
    {$group : {"_id": "$tipus", "avgp": {"$avg" : "$ár"} }},
    {$sort : {"avgp": -1}}

]);
```

i, Írassuk ki csökkenő sorrendben, hogy típusonként mennyi autó rendelkezik sérült státusszal.

```
db.auto.aggregate([
    {$match : {"állapot" : "sérült"}},
    {$group : {"_id": "$tipus" , "sérültek száma" : {"$sum" : 1 }}, {"$sort : {"száma" : -1}}

]);
```

3. JAVA rész következik.

a, Először is hozzunk létre egy MAVEN projektet és egy Teszt osztály az mdbgyak csomagban.

A pom.xml-be illesszük be a

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongo-java-driver</artifactId>
    <version>3.12.10</version>
</dependency>
```

Függőséget.

b, Kapcsolódjunk az mdbgyak adatbázishoz és jelöljük ki az auto kollekciót.

```
MongoClient mongo = new MongoClient("localhost", 27017);
MongoDatabase database = mongo.getDatabase("mdbgyak");
//Kijelöli az auto kollekciót.
MongoCollection<Document> collection = database.getCollection("auto");
```

c, Listázzuk ki az auto kollekció rekordjait.

```
FindIterable<Document> iterDoc = collection.find();
    int i = 1;
    // Getting the iterator
    Iterator it = iterDoc.iterator();
    while (it.hasNext()) {
        System.out.println(it.next());
        i++;
    }
```

d, Vegyünk fel egy új elemet az auto kollekcióba. Típusa legyen JavaInserted (insertOne())

```
Document document = new Document("tipus", "javaInserted")
    .append("szin", "kék")
    .append("ár", 2000110)
    .append("gyév", 2009)
    .append("állapot", "jó");

collection.insertOne(document);
```

e, Vegyünk fel egyszerre több elemet az auto kollekcióba. (insertMany())

```
Document document1 = new Document("tipus", "javaInserted")
    .append("szin", "kék")
    .append("ár", 2000110)
    .append("gyév", 2009)
    .append("állapot", "jó");

Document document2 = new Document("tipus", "javaInserted2")
    .append("szin", "zöld")
    .append("ár", 1223)
    .append("gyév", 2009)
    .append("állapot", "sérült");
```

g, A JavaInserted típusú autók állapotát állítsuk sérülte.

```
collection.updateMany(Filters.eq("tipus", " javaInserted "),
    Updates.set("állapot", "sérült"));
```

h, Töröljük a JavaInserted típusú rekordokat.

```
collection.deleteMany(Filters.eq("tipus", "javaInserted"));
```

Teljes kód:

```
package mdbgyak;

import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Updates;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.bson.Document;

public class Teszt {

    public static void main(String[] args) {

        MongoClient mongo = new MongoClient("localhost", 27017);
        MongoDatabase database = mongo.getDatabase("mdbgyak");
        System.out.println("Connected to the database successfully");

        //Kijelöli az auto kollekciót.
        MongoCollection<Document> collection =
            database.getCollection("auto");
        System.out.println("Collection auto selected successfully");

        //Kilistáz mindent
        FindIterable<Document> iterDoc = collection.find();
        int i = 1;

        Iterator it = iterDoc.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
            i++;
        }

        //Beilleszt egy elemet

        Document document = new Document("tipus", "javaInserted")
            .append("szin", "kék")
            .append("ár", 2000110)
            .append("gyév", 2009)
            .append("állapot", "jó");

        //Inserting document into the collection
        collection.insertOne(document);
        System.out.println("Document inserted successfully");
    }
}
```

//több dokumentum beillesztése kollekcióba.

```
Document document1 = new Document("tipus", "javaInserted")
    .append("szin", "kék")
    .append("ár", 2000110)
    .append("gyév", 2009)
    .append("állapot", "jó");
```

```
Document document2 = new Document("tipus", "javaInserted2")
    .append("szin", "zöld")
    .append("ár", 1223)
    .append("gyév", 2009)
    .append("állapot", "sérült");
```

```
List<Document> list = new ArrayList<Document>();
list.add(document1);
list.add(document2);
collection.insertMany(list);
```

//update

```
collection.updateMany(Filters.eq("tipus", "javaInserted"),
Updates.set("állapot", "sérült"));
System.out.println("Document updated successfully...");
```

//Törlés DB-ből

```
collection.deleteMany(Filters.eq("tipus", "javaInserted"));
System.out.println("Document deleted successfully...");
```

```
}
```

```
}
```

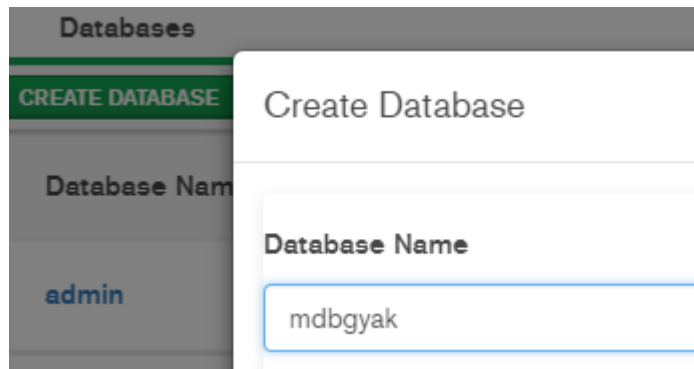
```
}
```

A futtatás eredménye:

1.feladatrész

1.a

Első lépésben létre kell hozni egy adatbázist. Érdemes a MongoDB Compassban létrehozni.



1.b

```
db.createCollection("auto")
```

0.363 sec.

```
/* 1 */
{
  "ok" : 1.0
}
```

1.c

Egy rekord felvétele:

```
db.auto.insert({'tipus' : 'Opel', 'szin' : 'piros', 'ár' : 500011, 'gyév':2001});
```

0.124 sec.

Inserted 1 record(s) in 124ms

Több rekord felvétele:

```
db.auto.insertMany([{'tipus' : 'Suzuki', 'szin' : 'sárga', 'ár' : 600000, 'gyév':2005},
{'tipus' : 'Toyota', 'szin' : 'fehér', 'ár' : 1600000, 'gyév':2010},
{'tipus' : 'Wolkswagen', 'szin' : 'zöld', 'ár' : 904000, 'gyév':2007},
{'tipus' : 'Audi', 'szin' : 'kék', 'ár' : 3220000, 'gyév':2014},
{'tipus' : 'Opel', 'szin' : 'piros', 'ár' : 500011, 'gyév':2001}]);
```

0.003 sec.

```
/* 1 */
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("627a4a4ad3e86d0be134b73f"),
    ObjectId("627a4a4ad3e86d0be134b740"),
    ObjectId("627a4a4ad3e86d0be134b741"),
    ObjectId("627a4a4ad3e86d0be134b742"),
    ObjectId("627a4a4ad3e86d0be134b743")
  ]
}
```

1.d

Feltételt teljesítő típusok:

```
db.auto.find({'ár' : {'$gt': 1000000}},{'tipus' : 1 , _id : 0})
```

auto 0.001 sec.

```
/* 1 */
{
  "tipus" : "Toyota"
}

/* 2 */
{
  "tipus" : "Audi"
}
```

A feltételt teljesítő rekordok darabszáma:

```
db.auto.find({'ár' : {'$gt': 1000000}},{'tipus' : 1 , _id : 0}).count();
```

0.001 sec.

2

1.e

Az auto kollekció rekordjaihoz állapot mező hozzáfűzése:

```
db.auto.updateMany ({} ,{'$set':{'allapot': "jó"}})
db.auto.find()
```

0.002 sec.

```
/* 1 */
{
  "acknowledged" : true,
  "matchedCount" : 1.0,
  "modifiedCount" : 0.0
}

{
  "_id" : ObjectId("627a49b0d3e86d0be134b73e"),
  "tipus" : "Opel",
  "szin" : "piros",
  "ár" : 500011.0,
  "gyév" : 2001.0,
  "állapot" : "jó"
}

/* 2 */
{
  "_id" : ObjectId("627a4a4ad3e86d0be134b73f"),
  "tipus" : "Suzuki",
  "szin" : "sárga",
  "ár" : 600000.0,
  "gyév" : 2005.0,
  "állapot" : "jó"
}
```

1.f

A 2004 előtt gyártott autó rekordok állapotának sérülte állítása.

```
db.auto.updateMany ({"gyév" : {"$lt" : 2004}}, {"$set":{"állapot": "sérült"}})
db.auto.find()
```

0.002 sec.

```
/* 1 */
{
  "acknowledged" : true,
  "matchedCount" : 2.0,
  "modifiedCount" : 2.0
}
```

<pre>"_id" : ObjectId("627a49b0d3e86d0be134b73e"), "tipus" : "Opel", "szin" : "piros", "ár" : 500011.0, "gyév" : 2001.0, "állapot" : "sérült"</pre>	<pre>"_id" : ObjectId("627a4a4ad3e86d0be134b743"), "tipus" : "Opel", "szin" : "piros", "ár" : 500011.0, "gyév" : 2001.0, "állapot" : "sérült"</pre>
---	---

1.g

A sérült állapotú autó rekordok törlése:

```
db.auto.deleteMany({"állapot" : "sérült});
```

0.071 sec.

```
/* 1 */
{
  "acknowledged" : true,
  "deletedCount" : 2.0
}
```

2. feladatrész

2.a

```
db.createCollection("tulajdonos");
```

0.187 sec.

```
/* 1 */
{
  "ok" : 1.0
}
```

2.b

System
auto
tulajdonos
Functions (3)
 fx getTulajByName
 fx save_auto
 fx save_tulaj
Users

```
db.system.js.save(
  {
    _id : "save_tulaj",
    value: function(id,nev, kor)
    {
      db.tulajdonos.insert({"_id": id, "név": nev, "kor": kor})
    }
  });
```

0.075 sec.
Updated 1 new record(s) in 2ms

```
db.loadServerScripts();

save_tulaj("T5","Pál", 20);

db.tulajdonos.find() |
```

tulajdonos 0.001 sec.

```
/* 1 */
{
  "_id" : "T5",
  "név" : "Pál",
  "kor" : 20.0
}
```

2.c

System
auto
tulajdonos
Functions (3)
 fx getTulajByName
 fx save_auto
 fx save_tulaj
Users

```
db.system.js.save(
  {
    _id : "save_auto",
    value: function(típus, szin, ar, gyart_ev, állapot , tulaj )
    {
      db.auto.insertOne({"típus": típus, "szin": szin, "ár": ar, "gyév": gyart_ev, "állapot": állapot, "tulaj": tulaj})
    }
  });
```

0.069 sec.
Updated 1 new record(s) in 69ms

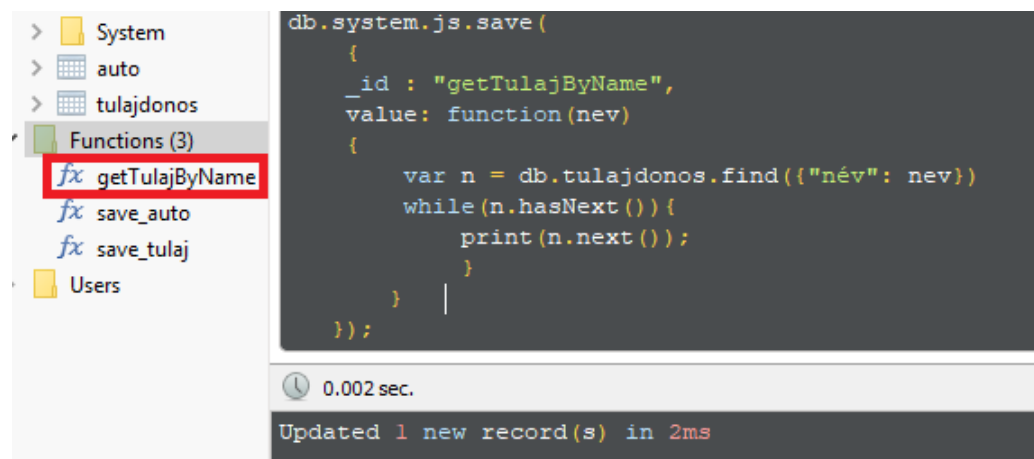
```
db.loadServerScripts();
save_auto("auto2","fehér",3241,2010, "jó", "T5");
db.auto.find()
```

auto 0.001 sec.

```
"állapot" : "jó"
}

/* 5 */
{
  "_id" : ObjectId("627a529ed3e86d0be134b745"),
  "típus" : "auto2",
  "szin" : "fehér",
  "ár" : 3241.0,
  "gyév" : 2010.0,
  "állapot" : "jó",
  "tulaj" : "T5"
}
```

2.d



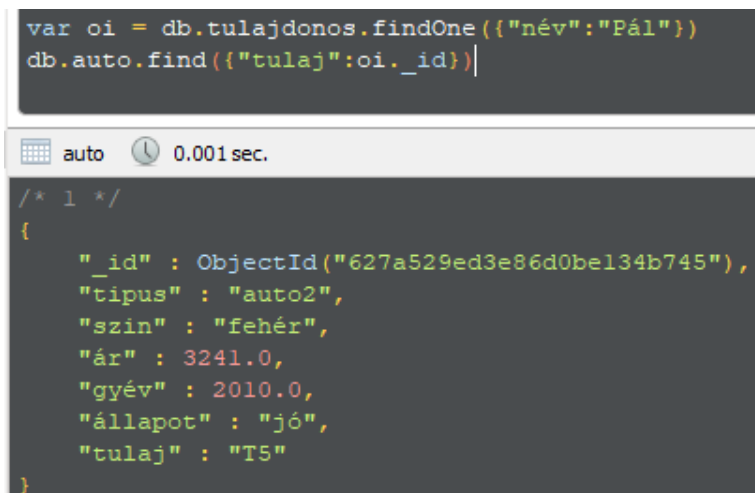
```
db.system.js.save({
  _id: "getTulajByName",
  value: function(nev)
  {
    var n = db.tulajdonos.find({"név": nev})
    while(n.hasNext()){
      print(n.next());
    }
  }
});
```

0.002 sec.

Updated 1 new record(s) in 2ms

2.e

Pálhoz tartozó autók: Jelenleg 1.



```
var oi = db.tulajdonos.findOne({"név": "Pál"})
db.auto.find({"tulaj": oi._id})
```

auto 0.001sec.

```
/* 1 */
{
  "_id" : ObjectId("627a529ed3e86d0be134b745"),
  "tipus" : "auto2",
  "szin" : "fehér",
  "ár" : 3241.0,
  "gyév" : 2010.0,
  "állapot" : "jó",
  "tulaj" : "T5"
}
```

2.f

Mivel a sérült állapotú autókat az 1.g feladatban az auto kollekcióból kitöröltük ezért, új auto rekordok felvételére van szükség melyek állapota = sérült.

```
db.auto.insert({'típus' : 'Suzuki', 'szin' : 'sárga', 'ár' : 600000, 'gyév':2005, "állapot" : "sérült"});
db.auto.insert({'típus' : 'Toyota', 'szin' : 'fehér', 'ár' : 1200000, 'gyév':2010, "állapot" : "sérült"});
```

```
db.auto.find().forEach( function(obj){
    if(obj.állapot == "sérült"){
        db.auto.update({'_id': obj._id}, {$inc : {'ár': -300000}});
    }
});
db.auto.find()
```

db.auto.fi.. ✕

auto 0 sec.

```
{
  "_id" : ObjectId("627a5bdad3e86d0be134b74a"),
  "típus" : "Suzuki",
  "szin" : "sárga",
  "ár" : 300000.0,
  "gyév" : 2005.0,
  "állapot" : "sérült"
}

/* 8 */
{
  "_id" : ObjectId("627a5bdad3e86d0be134b74b"),
  "típus" : "Toyota",
  "szin" : "fehér",
  "ár" : 900000.0,
}
```

2.g

```
db.auto.find({"$where" : function(){
    if(this.ár < 1000000 && this.gyév < 2010 )
        return true;
    else
        return false;
}}).count()
```

0.542 sec.

3

2.h

```
db.auto.aggregate([
    {$group : {"_id":"$típus","avgp":{"$avg" : "$ár"} }},
    {$sort : {"avgp": -1}}
]);
```

auto 0.001 sec.

```
/* 1 */
{
  "_id" : "Opel",
  "avgp" : 12000000.0
}

/* 2 */
{
  "_id" : "BMW",
  "avgp" : 9000550.0
}
```

2.i

```
db.auto.aggregate([
  {$match : {"állapot" : "sérült"}},
  {$group : {"_id": "$tipus" , "sérültek száma" : {"$sum" : 1 }}},{ $sort : {"száma" : -1}}
]);|
```

auto 0.08 sec.

```
/* 1 */
{
  "_id" : "Suzuki",
  "sérültek száma" : 2.0
}

/* 2 */
{
  "_id" : "BMW",
  "sérültek száma" : 1.0
}

/* 3 */
{
  "_id" : "Opel",
  "sérültek száma" : 1.0
}

/* 4 */
{
  "_id" : "Toyota",
  "sérültek száma" : 3.0
}
```

3. feladatrész

Java feladatok eredménye:

3.b

Kapcsolódás az mdbgyak adatbázishoz és az auto kollekció kijelölése.

```
máj. 10, 2022 3:12:42 DU. com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, req
Connected to the database successfully
Collection auto selected successfully
```

3.c

auto kollekció rekordjai:

```
INFO: opened connection [connectionid[localvalue:2], servervalue:74]] to localhost:27017
Document{{_id=627a4a4ad3e86d0be134b740, tipus=Toyota, szin=fehér, ár=1600000.0, gyév=2010.0, állapot=jó}}
Document{{_id=627a4a4ad3e86d0be134b741, tipus=Wolkswagen, szin=zöld, ár=904000.0, gyév=2007.0, állapot=jó}}
Document{{_id=627a4a4ad3e86d0be134b742, tipus=Audi, szin=kék, ár=3220000.0, gyév=2014.0, állapot=jó}}
Document{{_id=627a529ed3e86d0be134b745, tipus=auto2, szin=fehér, ár=3241.0, gyév=2010.0, állapot=jó, tulaj=T5}}
Document{{_id=627a5845d3e86d0be134b747, tipus=Toyota, szin=fehér, ár=600000.0, gyév=2010.0, állapot=sérült}}
Document{{_id=627a58eed3e86d0be134b749, tipus=Suzuki, szin=sárga, ár=0.0, gyév=2005.0, állapot=sérült}}
Document{{_id=627a5bdad3e86d0be134b74a, tipus=Suzuki, szin=sárga, ár=300000.0, gyév=2005.0, állapot=sérült}}
Document{{_id=627a5bdad3e86d0be134b74b, tipus=Toyota, szin=fehér, ár=900000.0, gyév=2010.0, állapot=sérült}}
Document{{_id=627a5d62d3e86d0be134b74c, tipus=BMW, szin=zöld, ár=1.2E7, gyév=2008.0, állapot=sérült}}
Document{{_id=627a5d62d3e86d0be134b74d, tipus=Audi, szin=fekete, ár=4001100.0, gyév=2007.0, állapot=jó}}
```

3.d

Egy rekord beillesztése az auto kollekcióba melynek típusa: javaInserted

```
Document{{_id=627a66181c85ab3f9d1f3358, tipus=javaInserted, szin=kék, ár=2000110, gyév=2009, állapot=jó}}
Document inserted successfully
```

3.e

Több rekord felvétele

```
Document{{_id=627a66181c85ab3f9d1f3358, tipus=javaInserted, szin=kék, ár=2000110, gyév=2009, állapot=jó}}
Document{{_id=627a66b7ab829b69b3daa589, tipus=javaInserted, szin=kék, ár=2000110, gyév=2009, állapot=jó}}
Document{{_id=627a66b7ab829b69b3daa58a, tipus=javaInserted2, szin=zöld, ár=1223, gyév=2009, állapot=sérült}}
```

3.f

A javaInserted típusú rekordok állapotának módosítása sérülte.

```
Document{{_id=627a66181c85ab3f9d1f3358, tipus=javaInserted, szin=kék, ár=2000110, gyév=2009, állapot=sérült}}
Document{{_id=627a66b7ab829b69b3daa589, tipus=javaInserted, szin=kék, ár=2000110, gyév=2009, állapot=sérült}}
```

3.g

javaInserted típusú auto rekordok törlése a kollekcióból:

```
Document{{_id=627a5d62d3e86d0be134b74f, tipus=Toyota, szin=Fehér, ár=4221100.0, gyév=2000.0, állapot=sérült}}
Document{{_id=627a5d62d3e86d0be134b750, tipus=BMW, szin=zöld, ár=6001100.0, gyév=2008.0, állapot=jó}}
Document{{_id=627a66b7ab829b69b3daa58a, tipus=javaInserted2, szin=zöld, ár=1223, gyév=2009, állapot=sérült}}
```