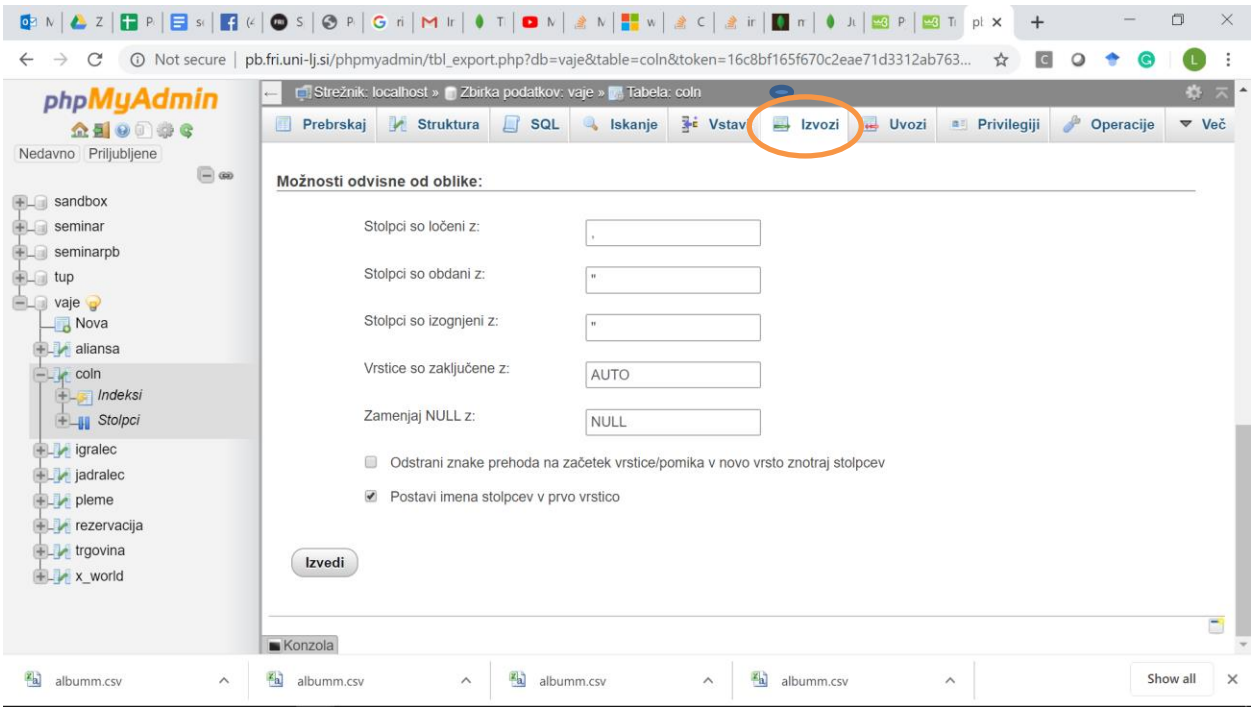
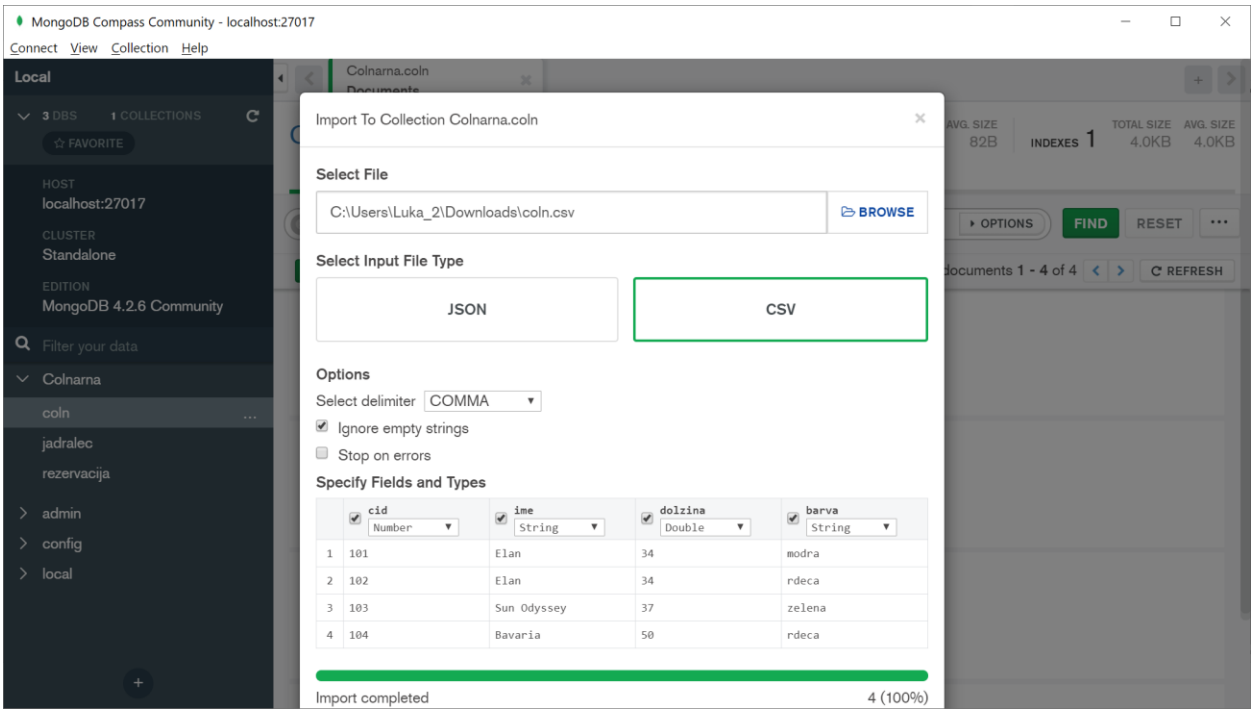
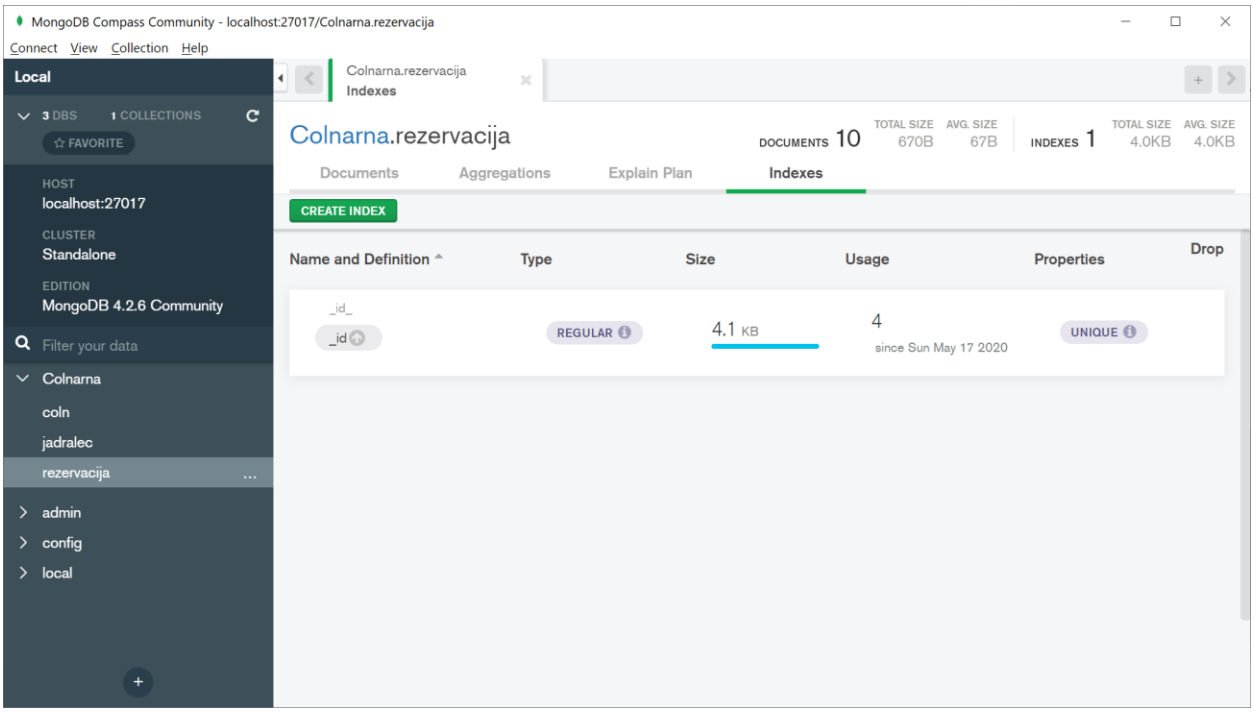
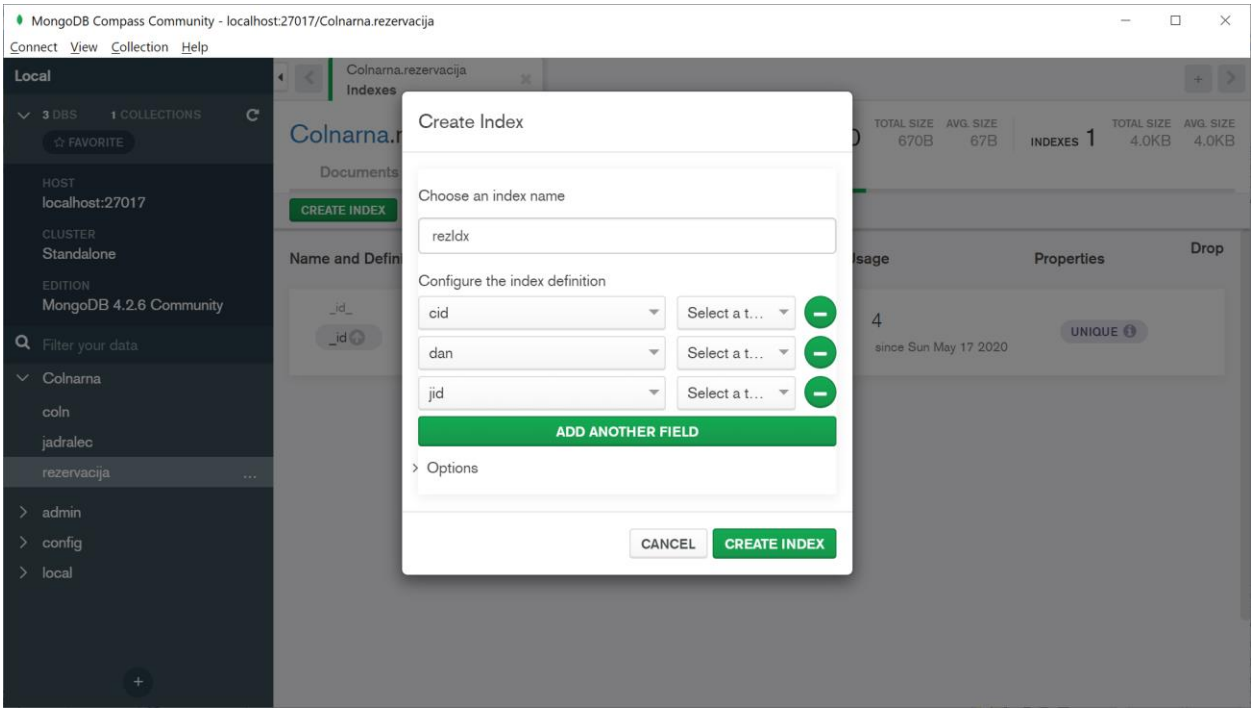


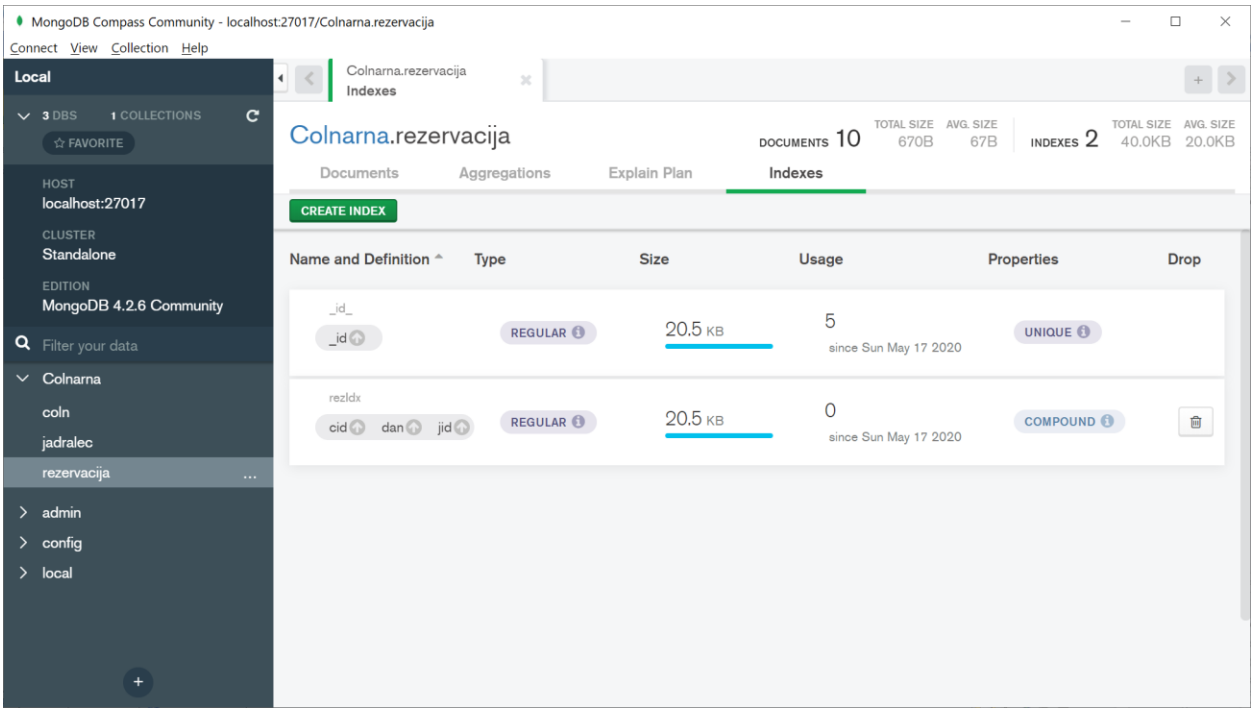


Lahko naredimo tudi indekse (_id je vedno indeksiran in avtomatsko dobi svoj GUID):

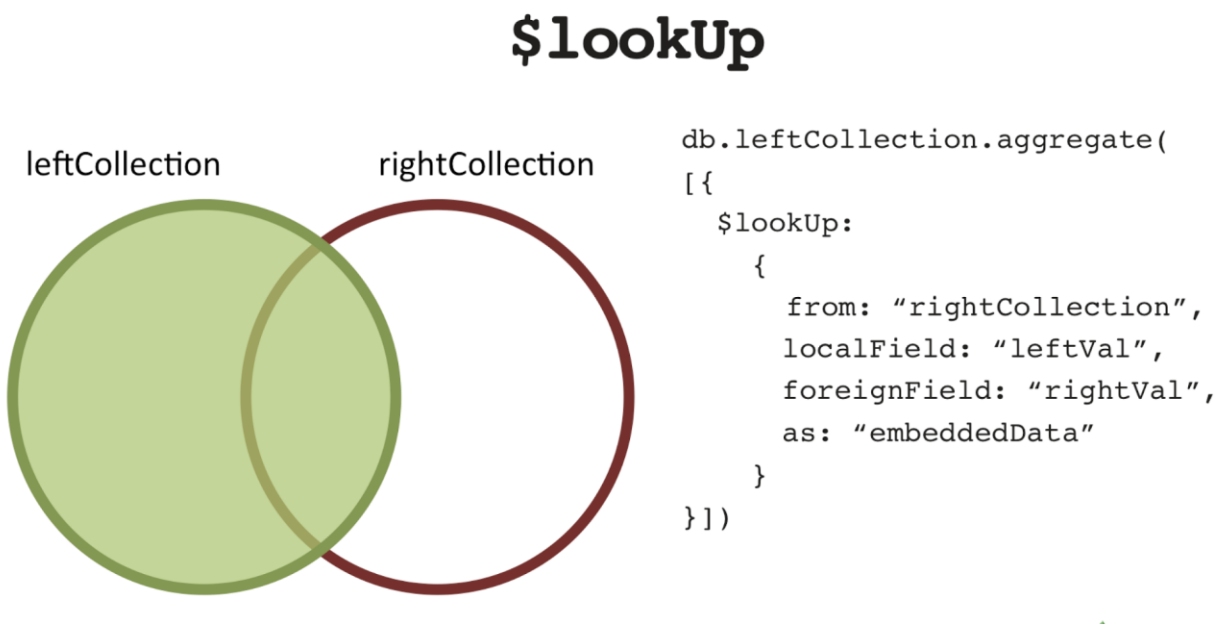


Lahko določimo indeks na skupek atributov:





Možni so tudi joini s funkcijo $lookUp, je pa dosti bolj zamudno (https://www.mongodb.com/blog/post/joins-and-other-aggregation-enhancements-coming-in-mongodb-3-2-part-1-of-3-introduction):



Recimo:

```
db.coln.aggregate(
        [{
```

```
                $lookup:
                        {
                                from: "rezervacija",
                                localField: "cid",
                                foreignField: "cid",
                                as: "stik"
                        }
        }]
)
```

V datoteki C:\Program Files\MongoDB\Server\4.2\log\mongod.log imamo vso zgodovino dela z bazo:

…

2020-05-17T17:08:41.023+0200 I NETWORK [conn21] received client metadata from 127.0.0.1:51187 conn21: { driver: { name: "nodejs", version: "3.5.6" }, os: { type: "Windows_NT", name: "win32", architecture: "x64", version: "10.0.18362" }, platform: "'Node.js v12.4.0, LE (unified)", application: { name: "MongoDB Compass Community" } }

2020-05-17T17:08:41.073+0200 I INDEX   [conn20] index build: done building index _id_ on ns Colnarna.Coln

2020-05-17T17:08:41.084+0200 I NETWORK [listener] connection accepted from 127.0.0.1:51188 #22 (5 connections now open)

2020-05-17T17:08:41.084+0200 I NETWORK [listener] connection accepted from 127.0.0.1:51189 #23 (6 connections now open)

2020-05-17T17:08:41.085+0200 I NETWORK [listener] connection accepted from 127.0.0.1:51190 #24 (7 connections now open)

2020-05-17T17:08:41.090+0200 I NETWORK [conn22] received client metadata from 127.0.0.1:51188 conn22: { driver: { name: "nodejs", version: "3.5.6" }, os: { type: "Windows_NT", name: "win32", architecture: "x64", version: "10.0.18362" }, platform: "'Node.js v12.4.0, LE (unified)", application: { name: "MongoDB Compass Community" } }

2020-05-17T17:08:41.093+0200 I NETWORK [conn23] received client metadata from 127.0.0.1:51189 conn23: { driver: { name: "nodejs", version: "3.5.6" }, os: { type: "Windows_NT", name: "win32", architecture: "x64", version: "10.0.18362" }, platform: "'Node.js v12.4.0, LE (unified)", application: { name: "MongoDB Compass Community" } }

2020-05-17T17:08:41.094+0200 I NETWORK [conn24] received client metadata from 127.0.0.1:51190 conn24: { driver: { name: "nodejs", version: "3.5.6" }, os: { type: "Windows_NT", name: "win32", architecture: "x64", version: "10.0.18362" }, platform: "'Node.js v12.4.0, LE (unified)", application: { name: "MongoDB Compass Community" } }

2020-05-17T17:08:44.679+0200 I SHARDING [conn21] Marking collection Colnarna.Coln as collection version: <unsharded>

2020-05-17T17:09:21.978+0200 I STORAGE [conn23] createCollection: Colnarna.jadralec with generated UUID: fa88cac4-0450-4e93-bc8c-740b2d17cd5d and options: {}

2020-05-17T17:09:22.011+0200 I INDEX   [conn23] index build: done building index _id_ on ns Colnarna.jadralec

2020-05-17T17:09:22.091+0200 I SHARDING [conn24] Marking collection Colnarna.jadralec as collection version: <unsharded>

2020-05-17T17:09:40.425+0200 I STORAGE [conn22] createCollection: Colnarna.rezervacija with generated UUID: f2fc170a-6890-467d-a937-842db42c4d3a and options: {}

…

V konzoli odpremo C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe in lahko delamo z bazo v ukazni vrstici. Npr: show dbs pokaže vse zbirke. Ko smo v neki Novega uporabnika naredimo tako:

```
db.createUser(
   {
     user: "accountUser",
     pwd: passwordPrompt(),  // Or  "<cleartext password>"
     roles: [ "readWrite", "dbAdmin" ]
   }
)
```

Ukaz

use Colnarna
db.coln.find();

izpiše dokumente v zbirki čoln.

Lahko dodamo kar novo zbirko npr.:

```
> db.test.insert([{ime:"luka"},{priimek:"Sajn"}]);
BulkWriteResult({
	"writeErrors" : [ ],
	"writeConcernErrors" : [ ],
	"nInserted" : 2,
	"nUpserted" : 0,
	"nMatched" : 0,
	"nModified" : 0,
```

```
        "nRemoved" : 0,
        "upserted" : [ ]
})
```
Tako dobimo novo zbirko test, z dvema dokumentoma:
```
> db.test.find();
{ "_id" : ObjectId("5ec24169073b16ffddd81041"), "ime" : "luka" }
{ "_id" : ObjectId("5ec24169073b16ffddd81042"), "priimek" : "Sajn" }
```

Sedaj smo dodali dva dokumenta, v resnici smo hoteli pa enega, za to moramo paziti ali delamo object ali seznam.
```
> db.test.insert([{ime:"luka",priimek:"Sajn"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 1,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.test.find();
{ "_id" : ObjectId("5ec24169073b16ffddd81041"), "ime" : "luka" }
{ "_id" : ObjectId("5ec24169073b16ffddd81042"), "priimek" : "Sajn" }
{ "_id" : ObjectId("5ec24403073b16ffddd81043"), "ime" : "luka", "priimek" : "Sajn" }
```

Tukaj vidimo tudi, da so ustvarjeni id-ji zaporedni in ne naključni.

Od relacijskih baz smo navajeni, da imajo relacije strogo določeno strukturo preko sheme, tukaj pa to ne velja in lahko dodamo nekaj popolnoma vsebinsko drugačnega:

```
> db.test.insert({
... ime:"Miha",
... priimek:"Mraz",
... seznamPredmetov: ["Modeliranje računalniških omrežij","Zanesljivost in zmogljivost računalniških sistemov","Nekonvencionalne platforme in metode procesiranja","Veterinarska informatika (VF)","Izbrana poglavja iz računalniških sistemov 1"],
... ObjektNaslov: {
... kabinet: "R3.59",
... lab: "LRSS",
... },
... diplomanti: [{ime:"jan",leto:1995},
... {ime:"anja",leto:1999},
... {ime:"tone",leto:2020}
... ]
... })
WriteResult({ "nInserted" : 1 })
>
```

In dobimo:
> db.test.find().pretty(); **//pretty lepše izpiše**

```
{ "_id" : ObjectId("5ec24169073b16ffddd81041"), "ime" : "luka" }
{ "_id" : ObjectId("5ec24169073b16ffddd81042"), "priimek" : "Sajn" }
{
	"_id" : ObjectId("5ec24403073b16ffddd81043"),
	"ime" : "luka",
	"priimek" : "Sajn"
}
{
	"_id" : ObjectId("5ec24695073b16ffddd81044"),
	"ime" : "Miha",
	"priimek" : "Mraz",
	"seznamPredmetov" : [
		"Modeliranje računalniških omrežij",
		"Zanesljivost in zmogljivost računalniških sistemov",
		"Nekonvencionalne platforme in metode procesiranja",
		"Veterinarska informatika (VF)",
		"Izbrana poglavja iz računalniških sistemov 1"
	],
	"ObjektNaslov" : {
		"kabinet" : "R3.59",
		"lab" : "LRSS"
	},
	"diplomanti" : [
		{
			"ime" : "jan",
			"leto" : 1995
		},
		{
			"ime" : "anja",
			"leto" : 1999
		},
		{
			"ime" : "tone",
			"leto" : 2020
		}
	]
}
```

Update atributov lahko naredimo takole:

```
db.test.update({ime: "luka "},{$set:{spol: "M "}});
```

tako dobimo:

```
> db.test.update({ime: "luka"},{$set:{spol: "M"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.test.find().pretty();
{
	"_id" : ObjectId("5ec24169073b16ffddd81041"),
	"ime" : "luka",
	"spol" : "M"
}
```

Postarajmo Henrika za eno leto:

```
> db.jadralec.update({ime:"Henrik"},{$inc:{starost:1}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.jadralec.find({ime:"Henrik"});
{ "_id" : ObjectId("5ec1580ffcfa690e587540b3"), "jid" : 64, "ime" : "Henrik", "rating" : 7, "starost" : 36 }
{ "_id" : ObjectId("5ec1580ffcfa690e587540b5"), "jid" : 74, "ime" : "Henrik", "rating" : 9, "starost" : 35 }
```

Za uporabo v konzoli več na: https://www.youtube.com/watch?v=pWbMrx5rVBE