



ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

MongoDB & Basic Queries

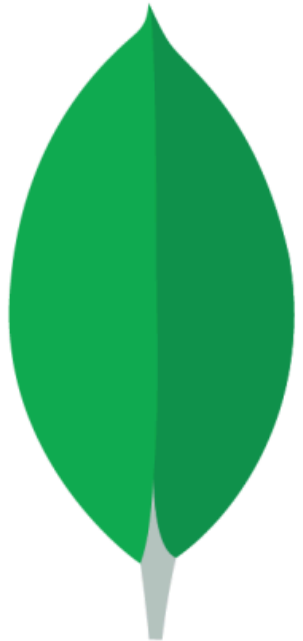
Μάρκος Καραμπάτσης



MongoDB

MongoDB είναι μια document schema-free, open-source βάση, γραμμένη σε C++.

Χρησιμοποιεί JSON documents για να αποθηκεύσει δεδομένα, ή πιο σωστά μια δυαδική μορφή τους, που λέγεται BSON







mongoDB®



5η στην κατάταξη του db-engines.com

410 systems in ranking, February 2023

Rank			DBMS	Database Model	Score		
Feb 2023	Jan 2023	Feb 2022			Feb 2023	Jan 2023	Feb 2022
1.	1.	1.	Oracle +	Relational, Multi-model i	1247.52	+2.35	-9.31
2.	2.	2.	MySQL +	Relational, Multi-model i	1195.45	-16.51	-19.23
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model i	929.09	+9.70	-19.96
4.	4.	4.	PostgreSQL +	Relational, Multi-model i	616.50	+1.65	+7.12
5.	5.	5.	MongoDB +	Document, Multi-model i	452.77	-2.42	-35.88
6.	6.	6.	Redis +	Key-value, Multi-model i	173.83	-3.72	-1.96
7.	7.	7.	IBM Db2	Relational, Multi-model i	142.97	-0.60	-19.91
8.	8.	8.	Elasticsearch	Search engine, Multi-model i	138.60	-2.56	-23.70
9. 	10.	10. 	SQLite +	Relational	132.67	+1.17	+4.30
10. 	9.	9. 	Microsoft Access	Relational	131.03	-2.33	-0.23



Γιατί να χρησιμοποιήσουμε MongoDB;

- Η SQL δημιουργήθηκε στα 70's για την αποθήκευση **δεδομένων (data)**
- Το 2007 η εταιρεία 10Gen δημιούργησε τη MongoDB για την αποθήκευση **αντικειμένων ή αλλιώς τεκμηρίωσης (objects aka documents)**
- Σήμερα όλοι χρησιμοποιούμε στον προγραμματισμό αντικείμενα (Python, Ruby, Java, κτλ)
- Χρειαζόμαστε μια βάση δεδομένων για την αποθήκευση των **αντικειμένων**

Γιατί να μην αποθηκεύουμε τα αντικείμενα κατευθείαν στη βάση δεδομένων;

- Η ενσωματωμένη τεκμηρίωση (embedded documents) και οι πίνακες εξαλείφουν την ανάγκη για joins και χρήση πολλαπλών πινάκων.



Χαρακτηριστικά της MongoDB

- Επεκτάσιμη και υψηλής απόδοσης, ανοικτού κώδικα, βάση δεδομένων
- προσανατολισμένη στις τεκμηριώσεις (documents)
- Ταχύτητα στην απόδοση (built for speed)
- Ενσωματώνει πλούσια και ευανάγνωστη διάλεκτο στις αναζητήσεις της τεκμηρίωσης (document queries)

Μια εγγραφή στο MongoDB είναι ένα **document**, το οποίο είναι μια δομή δεδομένων που αποτελείται από **ζεύγη πεδίων και τιμών**. Τα MongoDB documents είναι παρόμοια με τα αντικείμενα JSON. Οι τιμές των πεδίων μπορεί να περιλαμβάνουν άλλα documents, πίνακες και πίνακες από documents



Χαρακτηριστικά της MongoDB

Database: είναι ένα φυσικό container από collections. Ένας MongoDB server περιέχει πολλαπλές databases.

Collection: είναι κάτι αντίστοιχο με τους πίνακες στις σχεσιακές βάσεις δεδομένων. Μία collection ανήκει σε μία database. Οι collections περιέχουν MongoDB documents, κάτι αντίστοιχο με τις εγγραφές στις σχεσιακές βάσεις δεδομένων. Τα MongoDB documents σε μια collection μπορούν να έχουν και διαφορετικά πεδία.

Document: είναι ένα σύνολο ζευγών κλειδιών-τιμών. Τα έγγραφα (documents) έχουν δυναμικό σχήμα. Δυναμικό σχήμα σημαίνει ότι τα έγγραφα στην ίδια συλλογή (collection) δεν χρειάζεται να έχουν το ίδιο σύνολο από πεδία ή δομή. Τα κοινά πεδία στα documents μιας συλλογής ενδέχεται να περιέχουν διαφορετικούς τύπους δεδομένων.



Χαρακτηριστικά της MongoDB

Τα πλεονεκτήματα της χρήσης εγγράφων είναι:

- Τα documents είναι τύποι δεδομένων που χρησιμοποιούνται σε πολλές γλώσσες προγραμματισμού.
- Τα ενσωματωμένα documents και οι πίνακες από documents μειώνουν την ανάγκη για joins.
- Το δυναμικό σχήμα υποστηρίζει ρευστό πολυμορφισμό.

Παράδειγμα document στη MongoDB

```
{
  "author": {
    "user_id" : "121213",
    "first_name" : "John",
    "last_name": "Lennon"
  },
  "title": "Εισαγωγή στην MongoDB",
  "body": "Η MongoDB είναι μια βάση δεδομένων ανοικτού κώδικα ...",
  "timestamp": ISODate("2021-02-25T10:03:46.000Z"),
  "tags": ["MongoDB", "NoSQL"],
  "comments": [
    {
      "author": {
        "user_id" : "1319",
        "first_name" : "Satisfied",
        "last_name": "Student"
      },
      "date": ISODate("2020-05-18T14:10:30.000Z"),
      "text": "Καταπληκτική εισαγωγή, συγχαρητήρια!"
    },
    {
      "author": {
        "user_id" : "6666",
        "first_name" : "Unhappy",
        "last_name": "Student"
      },
      "date": ISODate("2021-01-15T06:31:15.000Z"),
      "text": "Δεν κατάλαβα τίποτα :-(
```



Αντιστοιχία εννοιών MongoDB & RDBMS

- **MongoDB** (no SQL) & RDBMS:
 - Αποτελείται από **συλλογές (collections)** αντί πίνακες (tables)
 - **Δημιουργείται αυτόματα στην πρώτη αλληλεπίδραση**
- **Collections** & Tables: (maximum size collection **unlimited**)
 - **Δεν έχουν σχήμα** για τα δεδομένα που περιέχουν (schema-less)
 - Δεικτοδοτούνται από ένα ή περισσότερα κλειδιά
 - **Δημιουργούνται αυτόματα στην πρώτη αλληλεπίδραση**
 - Capped collections: με συγκεκριμένη χωρητικότητα τεκμηριώσεων (documents), οι παλιές τεκμηριώσεις αντικαθίστανται από τις νέες
- **Documents** & Records/Rows (maximum size του document **16MB**)
 - Αποθηκεύονται στις συλλογές και έχουν αυτόματο πρωτεύων κλειδί το **_id**
 - Αποθηκεύονται σε μορφή BSON (Binary JSON)



Αντιστοιχία όρων

Αντιστοιχία όρων Σχεσιακών Βάσεων με MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)



JSON and BSON

JSON

JavaScript Object Notation. Ένα έγγραφο JSON είναι μια συλλογή πεδίων και τιμών σε μια δομημένη μορφή

BSON

Το BSON σημαίνει Binary JSON, το οποίο είναι ένα binary-coded serialization έγγραφο που μοιάζει με JSON

```
{  
  "first_name": "John",  
  "last_name": "Doe",  
  "age": 22,  
  "skills": ["Programming", "Databases", "API"]  
}
```

```
{  
  _id: ObjectId("5f339953491024badf1138ec"),  
  title: "MongoDB Tutorial",  
  isbn: "978-4-7766-7944-8",  
  published_date: new Date('June 01, 2020'),  
  author: {  
    first_name: "John",  
    last_name: "Doe"  
  }  
}
```



MongoDB - Collections

- Σε αντίθεση με έναν πίνακα που έχει σταθερό σχήμα, μια συλλογή έχει δυναμικό σχήμα.
- Μια συλλογή μπορεί να περιέχει έγγραφα που έχουν οποιοδήποτε αριθμό διαφορετικών «σχημάτων»

Name space

Εάν το όνομα της συλλογής είναι **books** και το όνομα της βάσης δεδομένων είναι **bookdb** τότε το namespace της συλλογής books θα είναι **bookdb.books**.

```
{
  title: "MongoDB Tutorial",
  published_date: new Date('June 01, 2020')
}

{
  title: "MongoDB Basics",
  published_date: new Date('Jan 01, 2021'),
  isbn": "978-4-7766-7944-8"
}
```




MongoDB - Χρήση

1. Τοπική εγκατάσταση (<https://www.mongodb.com/try/download/community>)
2. Χρήση μέσω cloud, MongoDB Atlas (<https://www.mongodb.com/atlas/database>)

Community edition

Version	6.0.8 (current)	▼
Platform	Windows x64	▼
Package	msi	▼

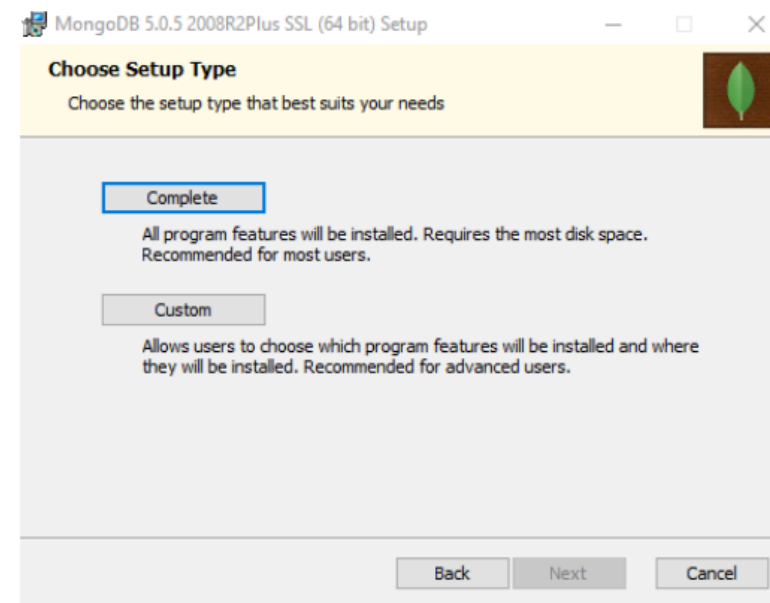
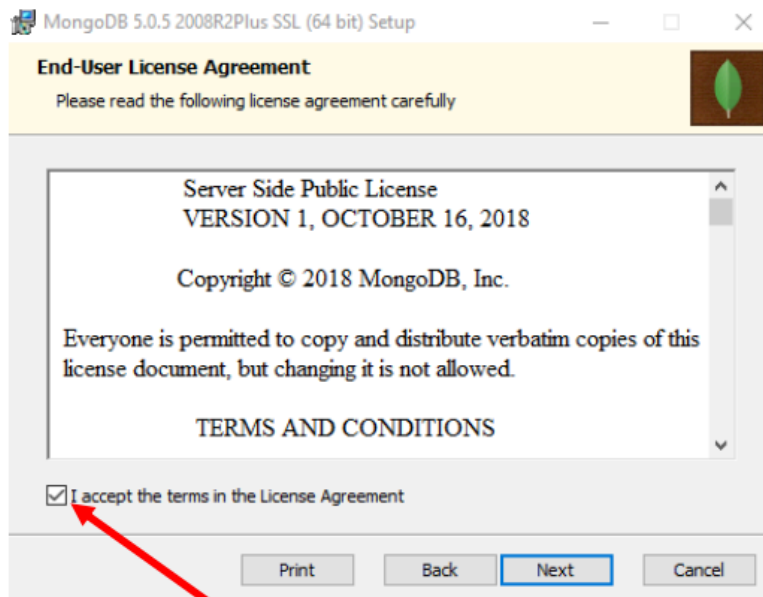
Download ⬇

 Copy link

More Options ...



MongoDB - Εγκατάσταση





MongoDB - Εγκατάσταση

MongoDB 5.0.5 2008R2Plus SSL (64 bit) Service Customization

Service Configuration
Specify optional settings to configure MongoDB as a service.

☒ Install MongoDB as a Service

☒ Run service as Network Service user

☐ Run service as a local or domain user:

Account Domain:

Account Name:

Account Password:

Service Name:

Data Directory:

Log Directory:

< Back Next > Cancel

MongoDB Compass

Install MongoDB Compass
MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: <https://www.mongodb.com/products/compass>

☐ Install MongoDB Compass

Back Next Cancel

MongoDB 5.0.5 2008R2Plus SSL (64 bit) Setup

Ready to install MongoDB 5.0.5 2008R2Plus SSL (64 bit)

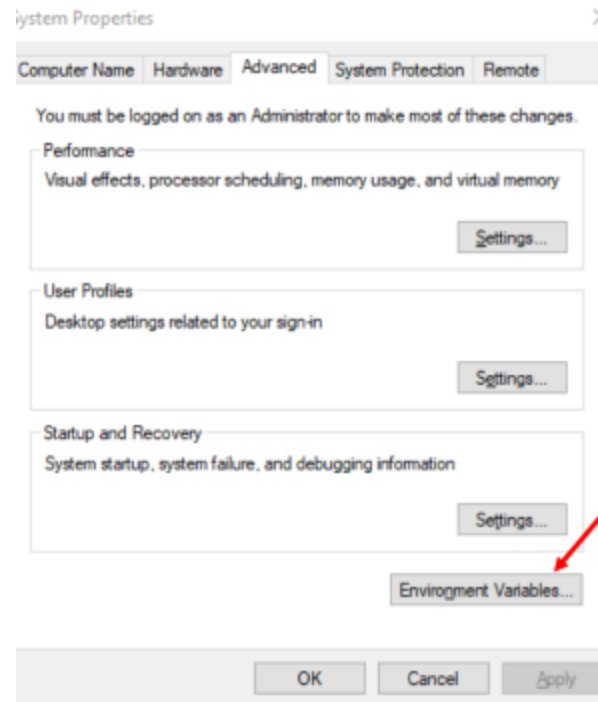
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Back Install Cancel



MongoDB - Εγκατάσταση

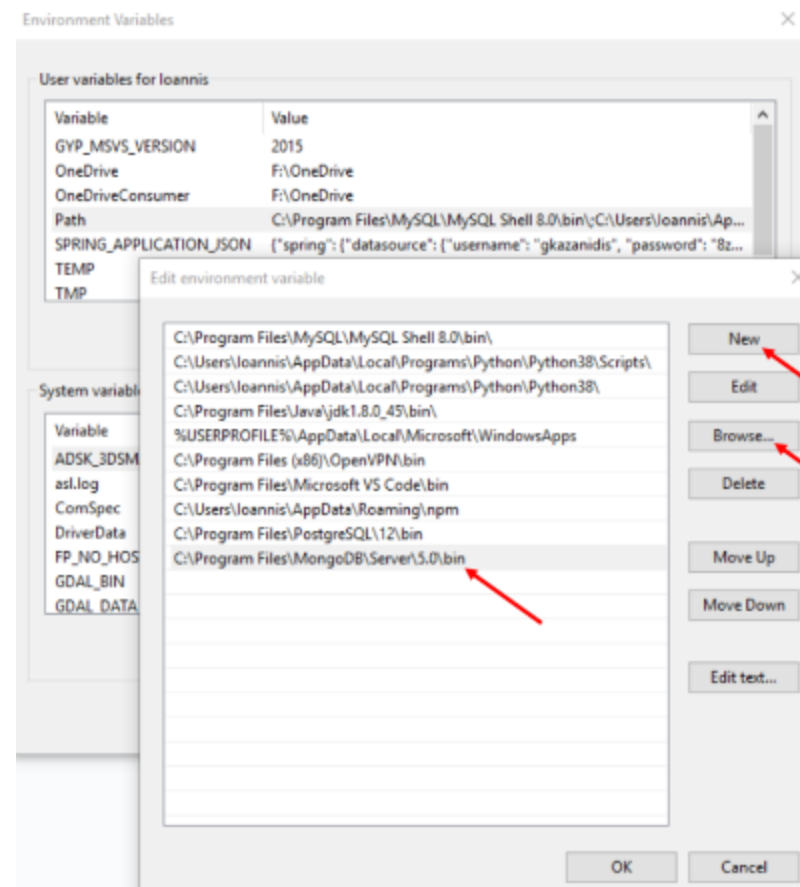
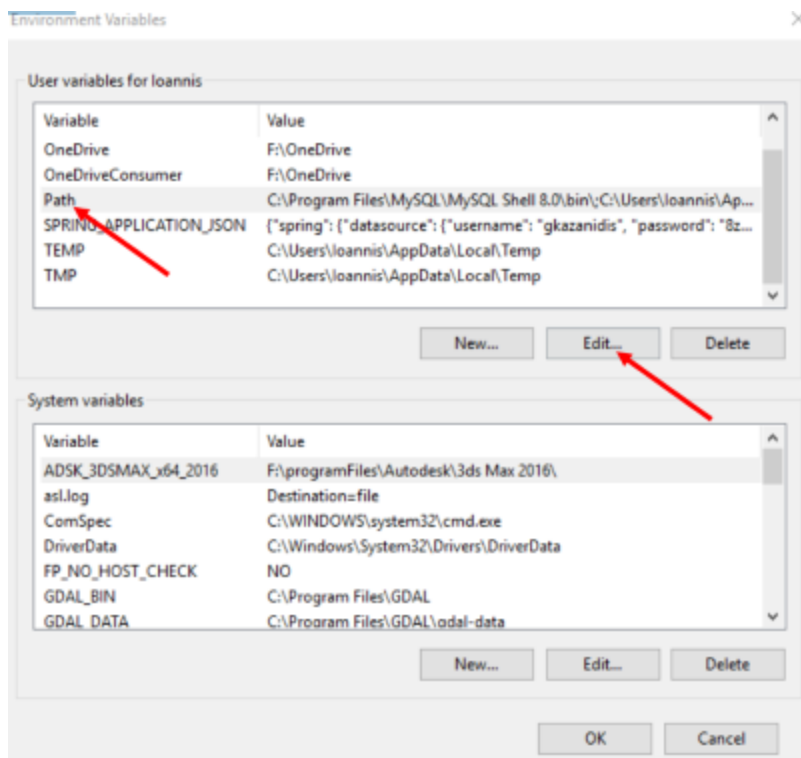
Προσθήκη του **mongo shell installation directory** στο PATH environment variables.





MongoDB - Εγκατάσταση

Προσθήκη του στο PATH environment variables.





MongoDB - Βασικές εντολές

Εντολή	Περιγραφή
mongo	Έναρξη mongoDB
db	Προβολή τρέχουσας Βάσης Δεδομένων
show dbs	Προβολή Βάσεων Δεδομένων
use databasename	Χρήση ΚΑΙ Δημιουργία ΒΔ
show collections	Προβολή των collections της ΒΔ
db.collectionName	Δημιουργία και χρήση ενός collection
db.collectionName.insertOne(...)	Δημιουργία ενός document
db.collectionName.count()	Πλήθος των documents

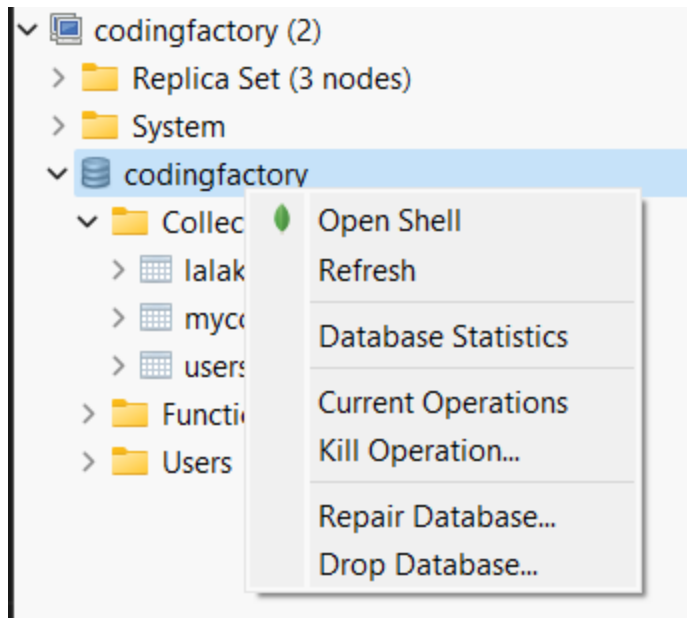
Coding Factory



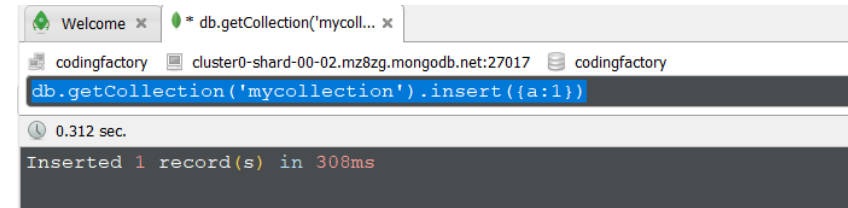
Εισαγωγή δεδομένων σε collection

Με χρήση GUI όπως Robot 3T ή Studio 3T

1ος Τρόπος



- Δεξί κλίκ στην βάση μας και κλίκ στην επιλογή **Open Shell**



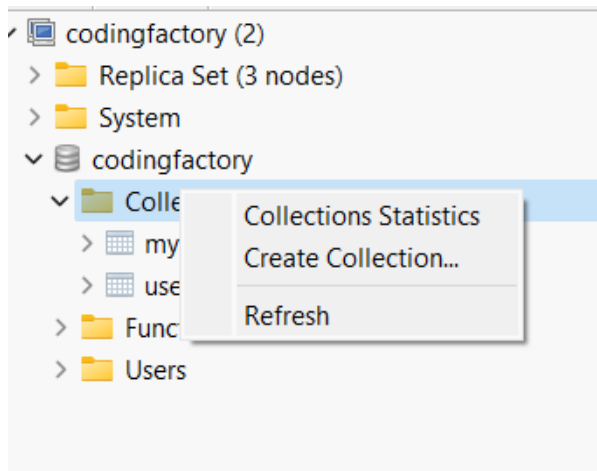
- Στο πεδίο εισαγωγής κώδικα πληκτρολογούμε:

```
db.getCollection('mycollection').insertOne({a:1})
```
- **getCollection('mycollection')**: επιλέγει την collection `mycollection` αν δεν υπάρχει τη δημιουργεί
- **insertOne**: εισάγει ένα document

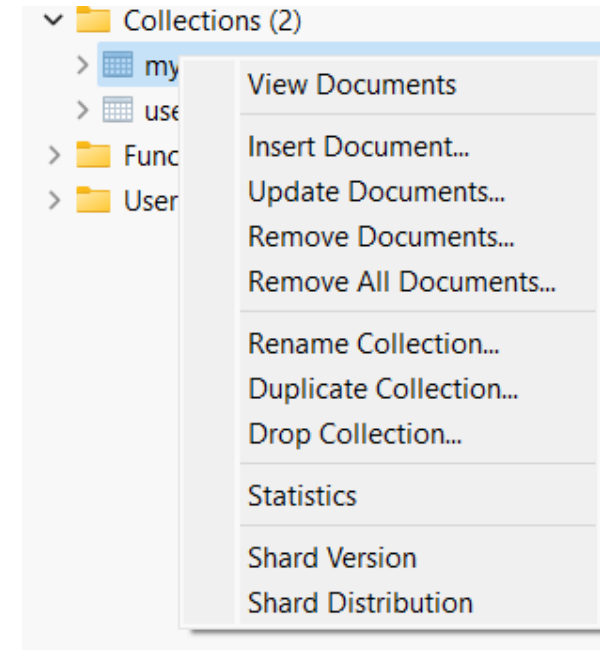


Εισαγωγή δεδομένων σε collection

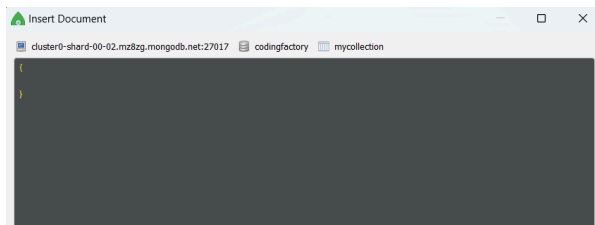
2ος Τρόπος



- Δεξί κλικ στην επιλογή Collections, κλικ στην επιλογή **Create Collection**



- Δεξί κλικ στην Collection μας και κλικ στην επιλογή **Insert Document**
- Στο πεδίο εισαγωγής πληκτρολογούμε το document





Εισαγωγή δεδομένων σε collection

- Χρησιμοποιώντας την πρώτη μέθοδο εισάγουμε το document

```
db.getCollection('users').insertOne({  
  username: "user1",  
  name: "John"  
});
```

- Παρατηρήστε πως στο νέο document της collection αποδόθηκε αυτόματα το **πρωτεύων κλειδί _id** με τύπο δεδομένων **ObjectId**
- Το objectId περιέχει μέσα και την ημερομηνία δημιουργίας του document

Η insertOne εισάγει ένα document

```
codingfactory cluster0-shard-00-02.mz8zg.mongodb.net:27017 codingfactory  
db.getCollection('users').insertOne({  
  username: "user1",  
  name: "John"  
});
```

0.212 sec.

Key	Value	Type
> (1)	{ 2 fields }	Object

```
{  
  "_id" : ObjectId("63dbf1d0616f8130d0d0d7dd"),  
  "username" : "user1",  
  "name" : "John"  
}
```



Ορίστε εσείς ένα _id

```
db.getCollection('users').insertOne({
  _id: 1111,
  username: "user1",
  name: "John"
});
```

codingfactory cluster0-shard-00-02.m28zg.mongodb.net:27017 codingfactory

```
db.getCollection('users').insertOne({
  _id: 1111,
  username: "user1",
  name: "John"
});
```

0.548 sec.

Key	Value	Type
✓ (1)	{ 2 fields }	Object
✓ acknowledged	true	Boolean
✓ insertedId	1111.0	Double

- Το πεδίο `_id` είναι το πρωτεύων κλειδί της collection

- Αν προσπαθήσουμε να εισάγουμε δεύτερο document με το ίδιο `_id` λαμβάνουμε μήνυμα λάθους

```
Error
Failed to execute script.

Error:
WriteError({
  "index": 0,
  "code": 11000,
  "errmsg": "E11000 duplicate key error collection:
codingfactory.users index: _id_ dup key: { _id: 1111.0}",
  "op": {
    "_id": 1111,
    "username": "user1",
    "name": "John"
  }
}):
WriteError({
  "index": 0,
  "code": 11000,
  "errmsg": "E11000 duplicate key error collection:
codingfactory.users index: _id_ dup key: { _id: 1111.0}",
  "op": {
    "_id": 1111,
    "username": "user1",
    "name": "John"
  }
})
WriteError@src/mongo/shell/bulk_api.js:458:48
mergeBatchResults@src/mongo/shell/bulk_api.js:855:49
executeBatch@src/mongo/shell/bulk_api.js:919:13
Bulk/this.execute@src/mongo/shell/bulk_api.js:1163:21
DBCollection.prototype.insertOne@src/mongo/shell/
crud_api.js:264:9
@(shell):1:1

OK
```



Εισαγωγή δεδομένων σε collection

- Μετακινηθείτε στο προηγούμενο παράθυρο και πατήστε ξανά το πράσινο βέλος
- Θα γίνει νέα εισαγωγή με τα ίδια δεδομένα (στο Raw Shell Output θα δούμε νέα ένδειξη `WriteResult({"nInserted" : 1})`)
- Διπλό κλίκ στο `users` αναζητά ξανά όλα τα documents της collection
- Παρατηρήστε πως υπάρχουν πλέον 2 documents με ίδια JSON δεδομένα **εκτός από το πεδίο `_id`**
- Η MongoDB φρόντισε αυτόματα να δημιουργήσει **νέο μοναδικό αναγνωριστικό** για το νέο document και το αποθήκευσε αυτόματα στο πεδίο `_id`

Key	Value	Type
▼ (1) ObjectId("63dbf1d0616f8...") { 3 fields }	{ 3 fields }	Object
_id	ObjectId("63dbf1d0616f8130d0d7...")	ObjectId
username	user1	String
name	John	String
▼ (2) ObjectId("63dbf388616f81...") { 3 fields }	{ 3 fields }	Object
_id	ObjectId("63dbf388616f8130d0d7...")	ObjectId
username	user1	String
name	John	String



Εισαγωγή δεδομένων στη collection Users

1. Κάντε δεξί κλικ στον φάκελο **Collections**, επιλέξτε **Create Collection**
2. Δημιουργήστε την collection **users**
3. Κάντε δεξί κλικ στην Collection **users**, επιλέξτε **Insert Document**
4. Εισάγεται τα documents του αρχείου users.json
5. Κάντε δεξί κλικ στον φάκελο **Collections**, επιλέξτε **Create Collection**
6. Δημιουργήστε την collection **products**
7. Κάντε δεξί κλικ στην Collection **products**, επιλέξτε **Insert Document**
8. Εισάγεται τα documents του αρχείου products.json



Αναζήτηση document

Σύνταξη μεθόδου find ή findOne

- `db.collection.find()`: επιστρέφει πολλά documents που ικανοποιούν το φίλτρο
- `db.collection.findOne()`: επιστρέφει το πρώτο document που ικανοποιεί το φίλτρο

```
db.getCollection("users").find({password: "12345"});  
db.getCollection("users").findOne({name: "Bob", surname: "Dylan"})
```

Options στις αναζητήσεις

```
db.getCollection("users").find({ name: "Bob" },  
    { _id: 0, surname: 1 });  
db.getCollection("users").find({},  
    { _id: 0, name: 1, surname: 1 })  
db.getCollection("users").find({},  
    { _id: 0, name: 1, surname: 1 }).sort({name: 1})
```

{ _id: 0, surname: 1 }: δεν επιστρέφει το `_id` και επιστρέφει μόνο το `surname`

{_id: 0, name: 1, surname: 1 }).sort({name: 1}): δεν επιστρέφει το `_id`, επιστρέφει μόνο `name`, `surname`. Sort με το `name`



Αναζήτηση σε array από documents

Σύνταξη μεθόδου find ή findOne

- `db.collection.find(<filter>)`: επιστρέφει πολλά documents που ικανοποιούν το φίλτρο
- `db.collection.findOne(<filter>)`: επιστρέφει το πρώτο document που ικανοποιεί το φίλτρο

```
db.getCollection('users').find({ "phone.number": "2102222222" })
db.getCollection('users').find({
  $or: [
    {"phone.number": "2102222222"},
    {"phone.number": "2103333333"}
  ])
db.getCollection('users').find({
  "phone.number": { $in: ["2102222222", "2103333333"] }
})
```

Επιστρέφει όλα τα documents που έχουν στο πεδίο phone του array phone το τηλέφωνο

2102222222 ή 2103333333



Τροποποίηση δεδομένων σε collection \$SET

Σύνταξη μεθόδου Update (find by id)

```
db.collection.updateOne(<filter>, <update>, <options>)
```

```
db.getCollection('users').updateOne(
  {_id: ObjectId("63dbf920e2c84d81c136f6a6")},
  {
    $set: {
      name: "George",
      surname: "Harrison"
    }
  });
```

codingfactory cluster0-shard-00-02.mz8zg.mongodb.net:27017 codingfactory

```
db.getCollection('users').updateOne(
  {_id: ObjectId("63dbf920e2c84d81c136f6a6")},
  {
    $set: {
      name: 'Lakis',
      surname: 'Lalakis'
    }
  });
```

0.207 sec.

Key	Value	Type
✓ (1)	{ 3 fields }	Object
<input type="checkbox"/> acknowledged	true	Boolean
<input type="checkbox"/> matchedCount	1.0	Double
<input type="checkbox"/> modifiedCount	1.0	Double



Τροποποίηση δεδομένων σε collection \$SET

Find by field

```
db.getCollection('users').updateOne(  
  {name: "George"},  
  {  
    $set: {  
      name: "Bob",  
      surname: "Dylan"  
    }  
  });
```

Update many, insert new fields

```
db.getCollection('users').updateMany(  
  {},  
  {  
    $set: {  
      newField1: "myfield 1",  
      newField2: [1, 2, 3]  
    }  
  },  
  );
```

- Προσθέτει σε όλα τα documents τα πεδία newField1, newField2



Εισαγωγή σε array of subdocuments (1/2)

\$push method

```
db.collection('users').updateOne(
  { username: "user2"},
  {
    $push: {
      phone: {
        _id: new ObjectId(),
        type: "work",
        number: 2102222221
      },
    }
  }
);
```

```
{
  "_id" : ObjectId("63dbf921e2c84d81c136f6de"),
  "username" : "user2",
  "password" : "12345",
  "name" : "Johnny",
  "surname" : "Rivers",
  "email" : "user2@aueb.gr",
  "address" : {
    "area" : "area1",
    "road" : "road3"
  },
  "phone" : [
    {
      "type" : "home",
      "number" : "2103333333"
    },
    {
      "type" : "mobile",
      "number" : "6933333333"
    }
  ],
  "_id" : ObjectId("63dc3467616f8130d0d0d7e1"),
  "type" : "home",
  "number" : "2102222221"
},
"newField1" : "myfield 1",
"newField2" : [ 1.0, 2.0, 3.0 ]
}
```



Εισαγωγή σε array of subdocuments (2/2)

```
db.getCollection('users').updateOne(
  { username: "user2"},
  {
    $push: {
      newField2: 4,
    }
  }
);
```

```
codingfactory cluster0-shard-00-02.mz8zg.mongodb.net:27017
db.getCollection('users').updateOne(
  { username: "user2"},
  {
    $push: {
      newField2: 4,
    }
  }
);
```

0.195 sec.

Key	Value
(1)	{ 3 fields }
acknowledged	true
matchedCount	1.0
modifiedCount	1.0

```
{
  "_id" : ObjectId("63dbf921e2c84d81c136f6de"),
  "username" : "user2",
  "password" : "12345",
  "name" : "Johnny",
  "surname" : "Rivers",
  "email" : "user2@aueb.gr",
  "address" : {
    "area" : "area1",
    "road" : "road3"
  },
  "phone" : [
    {
      "type" : "home",
      "number" : "2103333333"
    },
    {
      "type" : "mobile",
      "number" : "6933333333"
    },
    {
      "_id" : ObjectId("63dc3467616f8130d0d0d7e1"),
      "type" : "home",
      "number" : "2102222221"
    }
  ],
  "newField1" : "myfield 1",
  "newField2" : [ 1.0, 2.0, 3.0, 4.0 ]
}
```



Τροποποίηση σε array of subdocuments \$SET

```
db.getCollection('users').updateMany(  
  { "phone.number": "2102222221" },  
  { $set: { "phone.$.number": 210 } }  
);
```

codingfactory cluster0-shard-00-02.mz8zg.mongodb.net:27017

```
db.getCollection('users').update(  
  { "phone.home": 2102222221 },  
  { $set: { "phone.$.home": 210 } }  
);
```

0.385 sec.

1 Updated 0 record(s) in 381ms

2

```
{  
  "_id" : ObjectId("63dbf921e2c84d81c136f6de"),  
  "username" : "user2",  
  "password" : "12345",  
  "name" : "Johnny",  
  "surname" : "Rivers",  
  "email" : "user2@aueb.gr",  
  "address" : {  
    "area" : "area1",  
    "road" : "road3"  
  },  
  "phone" : [  
    {  
      "type" : "home",  
      "number" : "2103333333"  
    },  
    {  
      "type" : "mobile",  
      "number" : "6933333333"  
    }  
  ],  
  "_id" : ObjectId("63dc3467616f8130d0d0d7e1"),  
  "type" : "home",  
  "number" : "210"  
},  
  "newField1" : "myfield 1",  
  "newField2" : [ 1.0, 2.0, 3.0, 4.0 ]  
}
```



Διαγραφή subdocument από array ή πεδίου

Διαγραφή subdocument από array: **\$pull** method

```
db.getCollection('users').updateOne(
  { username: "user2" },
  {
    $pull: {
      phone: { type:"home" } }
  }
);
```

```
codingfactory cluster0-shard-00-02.mz8zg.mongodb.net:27017 codingfactory
db.getCollection('users').updateOne(
  { username: "user2" },
  {
    $pull: {
      phone: { type:"home" } }
  }
);
```

0.161 sec.

Key	Value	Type
✓ (1)	{ 3 fields }	Object
acknowledged	true	Boolean
matchedCount	1.0	Double
modifiedCount	1.0	Double

Διαγραφή πεδίου από document: **\$unset** method

```
db.getCollection('users').updateMany(
  {},
  {
    $unset: {
      newField1: "",
      newField2: ""
    }
  },
);
```



Διαγραφή document απο collection

Σύνταξη μεθόδου Delete

```
db.collection.deleteOne(<filter>)
```

```
db.getCollection('users').deleteOne(  
  { username: "user5" }  
);
```

Σύνταξη μεθόδου deleteMany

```
db.getCollection('users').deleteMany( { } )
```

deletes all documents



MongoDB Aggregations

Οι λειτουργίες του **Aggregation** επεξεργάζονται πολλαπλά **documents** και επιστρέφουν ένα αποτέλεσμα.

Οι λειτουργίες του Aggregation χρησιμοποιούνται όταν θέλουμε:

- να ομαδοποιήσουμε τιμές από πολλαπλά documents.
- να πραγματοποιήσουμε λειτουργίες στα ομαδοποιημένα δεδομένα για να επιστρέψει ένα μόνο αποτέλεσμα.
- να αναλύσουμε τις αλλαγές των δεδομένων

Για να πραγματοποιήσουμε aggregation λειτουργίες χρησιμοποιούμε **aggragation pipelines**, όπως: `$match`, `$project`, `$group`, `$unwind`, `$lookup`



Aggregation Pipelines

\$match : φιλτράρει τα documents για να περάσει μόνο τα έγγραφα που ταιριάζουν με τις καθορισμένες συνθήκες στο επόμενο στάδιο του pipeline.

Μορφή της \$match: { \$match: { <query> } }

\$project: επιλέγει απο τα documents συγκεκριμένα πεδία τα οποία και περνά στο επόμενο pipeline. Τα πεδία μπορεί να είναι υπάρχοντα πεδία από τα documents εισόδου ή πεδία που έχουν υπολογιστεί.

Μορφή της \$project: { \$project: { <specification(s)> } }



Aggregation Pipelines

\$group : διαχωρίζει τα documents σε ομάδες σύμφωνα με ένα "κλειδί ομάδας". Το αποτέλεσμα είναι ένα document για κάθε μοναδικό κλειδί της ομάδας.

Ένα κλειδί ομάδας είναι συχνά ένα πεδίο ή ομάδα πεδίων. Χρησιμοποιούμε το πεδίο `_id` για να ορίσουμε το κλειδί ομάδας.

Μορφή της \$group

```
$group:
{
  _id: <expression>, // Group key
  <field1>: { <accumulator1> : <expression1> },
  ...
}
```



Aggregation Pipelines

\$unwind: Αποδομεί ένα πεδίο πίνακα από τα έγγραφα εισόδου για να εξάγει ένα έγγραφο για κάθε στοιχείο. Κάθε έγγραφο εξόδου είναι το έγγραφο εισόδου με την τιμή του πεδίου πίνακα να αντικαθίσταται από το στοιχείο.

Deconstructs an array field from the input documents to output a document for each element. Each output document is the input document with the value of the array field replaced by the element.

```
Μορφή της $unwind  
{ $unwind: <field path> }
```



Aggregation Pipelines

\$lookup: Εκτελεί ένα left join σε με collection που βρίσκεται στην ίδια βάση. Η \$lookup προσθέτει ένα νέο πεδίο πίνακα σε κάθε έγγραφο εισόδου. Το νέο πεδίο πίνακα περιέχει τα έγγραφα που ταιριάζουν από τη "joined" collection.

Μορφή της \$lookup

```
{
  $lookup:
  {
    from: <collection to join>,
    localField: <field from the input documents>,
    foreignField: <field from the documents of the "from" collection>,
    as: <output array field>
  }
}
```



Aggregation Pipelines

\$sort: Ταξινομεί όλα τα έγγραφα εισόδου και τα επιστρέφει ταξινομημένα.

Μορφή της \$sort

```
{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }
```

\$sum: Υπολογίζει και επιστρέφει το άθροισμα αριθμητικών τιμών

\$count: Επιστρέφει τον αριθμό των documents σε μια ομάδα



Παράδειγμα 1

Aggregation σε μία collection

```
db.getCollection("users").aggregate([
  {
    $match:
      {
        "address.area": "area2"
      }
  },
  {
    $project:
      {
        count: { $size:"$phone" }
      }
  }
]);
```

Αρχικά με την `$match` φιλτράρει τα documents και επιστρέφει όλα τα documents που έχουν στο πεδίο `area` την τιμή `area2`.

Στην συνέχεια διοχετεύει το αποτέλεσμα στην `$project` και επιστρέφει πόσες εγγραφές υπάρχουν στο πεδίο `phone`



Παράδειγμα 2

Aggregation σε μία collection

```
db.getCollection("users").aggregate([
  {
    $project: {
      "_id" : 0,
      "phone" : 1
    }
  },
  { $unwind: "$phone" },
  {
    $project: {
      "phone" : "$phone.type"
    }
  },
]);
```

Η \$project θα επιστρέψει στο pipeline μόνο το πεδίο phone απο τα documents

Η \$unwind θα δημιουργήσει διαφορετικό document για κάθε στοιχείο του πίνακα phone.

Η \$project θα δημιουργήσει ένα νέο πεδίο phone που έχει ως τιμή το phone.type



Παράδειγμα 3

Aggregation σε δύο collection

```
db.getCollection("users").aggregate([
  {
    $project: {
      _id: 0,
      username: 1,
      products: 1
    }
  },
  { $unwind: "$products" },
  {
    $lookup:
    {
      from: "products",
      localField: "products.product",
      foreignField: "product",
      as: "results"
    }
  }
])
```

Η \$project επιστρέφει στο pipeline τα πεδία username και products για κάθε document

Η \$unwind θα δημιουργήσει διαφορετικό document για κάθε στοιχείο του πίνακα products.

Η \$lookup κάνει left join στην collection products και επιστρέφει το αποτέλεσμα στο πεδίο results.



Παράδειγμα 4

```
db.getCollection('users').aggregate([
  {
    $unwind: "$products"
  },
  {
    $project: {
      id: 1,
      username: 1,
      products: 1
    }
  },
  {
    $group: {
      _id: { username: "$username" },
      totalAmount: {
        $sum: {
          $multiply: [ "$products.cost", "$products.quantity" ]
        }
      },
      count: { $sum: 1 }
    }
  }
])
```

Για καθε user βρίσκει το συνολικό ποσό αγοράς και πόσα προϊόντα έχει αγοράσει



Παράδειγμα 5

```
db.getCollection('users').aggregate([
  {
    $unwind: "$products"
  },
  {
    $project: {
      id: 1,
      username: 1,
      products: 1
    }
  },
  {
    $group: {
      _id: {
        username: "$username",
        product: "$products.product" },
      totalAmount: {
        $sum: {
          $multiply: [ "$products.cost", "$products.quantity" ]
        }
      },
      count: { $sum: 1 }
    }
  },
  {
    $sort: { username : 1, product : 1 }
  },
])
```

Για καθε user βρίσκει ανα προϊόν το συνολικό ποσό αγοράς και πόσες φορές το έχει αγοράσει.

Στο τέλος κάνει sort ανα username και προϊόν



Παράδειγμα 6

```
db.getCollection('users').aggregate([
  {
    $match: {
      $or: [
        { username: "user1" },
        { username: "user2" } ]
    }
  },
  { $unwind: "$products" },
  {
    $project: {
      id: 1,
      username: 1,
      products: 1
    }
  },
  {
    $group: {
      _id: {
        username: "$username",
        product: "$products.product"
      },
      totalAmount: {
        $sum: {
          $multiply: [ "$products.cost", "$products.quantity" ] } },
      count: { $sum: 1 }
    }
  },
  { $sort: { username : 1, product : 1 } }
])
```

Αρχικά κάνει match και επιστρέφει μόνο τα documents των user με username user1 και user2

Για κάθε user βρίσκει ανα προϊόν το συνολικό ποσό αγοράς και πόσες φορές το έχει αγοράσει.

Στο τέλος κάνει sort ανα username και προϊόν



Παράδειγμα 7

```
db.getCollection('users').aggregate([
  { $unwind: "$products" },
  {
    $project: {
      id: 1,
      username: 1,
      products: 1
    }
  },
  {
    $match : {
      "products.product": "product 1"
    }
  },
  {
    $group:
    {
      _id: { username: "$username" },
      totalAmount: {
        $sum: {
          $multiply: [ "$products.cost", "$products.quantity" ] } },
      count: { $sum: 1 }
    }
  },
  { $sort:{ username : 1, product : 1 } }
])
```

Για καθε user βρίσκει για το προϊόν "product 1" το συνολικό ποσό αγοράς και πόσες φορές το έχει αγοράσει.

Στο τέλος κάνει sort ανα username και προϊόν



Παράδειγμα 8

```
db.getCollection('users').aggregate([
  { $unwind: "$products" },
  {
    $project: {
      id: 1,
      products:1
    }
  },
  {
    $group:
    {
      _id: { products: "$products.product" },
      totalAmount: {
        $sum: {
          $multiply: [ "$products.cost", "$products.quantity" ] } },
      count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      product: "$_id.products",
      totalAmount: "$totalAmount",
      count: "$count"
    }
  },
  { $sort:{ product : 1 } }
])
```

Εμφανίζει ανα προϊόν συνολικό ποσό από αγορές και πόσες φορές αγοράστηκαν



MongoDB Schema Design καλές πρακτικές

- Στο Schema Design στη MongoDB, το μόνο που έχει σημασία είναι να σχεδιάσουμε ένα σχήμα που θα λειτουργεί καλά για την εφαρμογή μας.
- Δύο διαφορετικές εφαρμογές που χρησιμοποιούν τα ίδια ακριβώς δεδομένα ενδέχεται να έχουν πολύ διαφορετικά σχήματα, ειδικά εάν οι εφαρμογές χρησιμοποιούν με διαφορετικό τρόπο τα δεδομένα.
- Όταν σχεδιάζουμε ένα σχήμα, το βασικό που θα πρέπει να λάβουμε υπόψη είναι τα query ερωτήματα μας να είναι όσο γίνεται πιο αποτελεσματικά, αποδοτικά και γρήγορα.
- Το Schema Design στη MongoDB έχει δύο βασικές επιλογές για κάθε κομμάτι δεδομένων.
 - i. Είτε να ενσωματώσουμε αυτά τα δεδομένα απευθείας (Embedding).
 - ii. Είτε να αναφέρουμε ένα άλλο τμήμα δεδομένων χρησιμοποιώντας τον τελεστή \$lookup, παρόμοιο με ένα JOIN (Referencing).



MongoDB - Embedding

Πλεονεκτήματα

- Μπορούμε να ανακτήσουμε όλες τις σχετικές πληροφορίες σε ένα μόνο ερώτημα.
- Αποφεύγουμε την εφαρμογή join στον κώδικα εφαρμογής ή τη χρήση \$lookup.
- Ενημερώνουμε ένα document με μία μεμονωμένη λειτουργία.

Μειονεκτήματα

- Μέγάλα document έχουν ως συνέπεια μεγαλύτερο overhead. Ενδεχόμενος πολλά από τα πεδία του document να μην χρειάζονται στο έρωτημα μας. Αυτό μπορούμε να το αποφύγουμε κόβοντας πεδία από το document μέσω του query μας
- Υπάρχει ένα όριο στο μέγεθος των document, 16-MB. Εάν σε ένα document ενσωματώνουμε μεγάλο όγκο πληροφορίας υπάρχει πιθανότητα αυτό το μέγεθος να το ξεπεράσουμε.



MongoDB - Referencing

Πλεονεκτήματα

- Διαχωρίζοντας δεδομένα, θα έχουμε μικρότερα έγγραφα
- Λιγότερο πιθανό να φτάσει ένα document το όριο των 16 MB.
- Σπάνια πρόσβαση σε πληροφορίες (πεδία) που δεν χρειάζονται για κάθε ερώτημα.
- Μειώνεται ο όγκος των διπλών δεδομένων. Ωστόσο, είναι σημαντικό να σημειωθεί ότι η εμφάνιση διπλών δεδομένων δεν πρέπει να αποφεύγεται εάν οδηγεί σε καλύτερο σχήμα.

Μειονεκτήματα

Για να ανακτηθούν όλα τα δεδομένα στα αναφερόμενα έγγραφα, απαιτούνται τουλάχιστον δύο ερωτήματα ή \$lookup για την ανάκτηση όλων των πληροφοριών.



MongoDB - Τύποι σχέσεων

One-to-One

- Στο παράδειγμα μας ένας χρήστης μπορεί να έχει μόνο ένα όνομα.
- Αυτό αποτελεί ένα παράδειγμα μιας σχέσης **one-to-one**.
- Μπορούμε να μοντελοποιήσουμε όλα τα δεδομένα ένα προς ένα ως ζεύγη κλειδιού-τιμής στη βάση δεδομένων μας

```
{  
  "_id": ObjectId('AAA'),  
  "name": "Joe Karlsson",  
  "company": "MongoDB",  
  "twitter": "@JoeKarlsson1",  
  "twitch": "joe_karlsson",  
  "tiktok": "joekarlsson",  
  "website": "joekarlsson.com"  
}
```



MongoDB - Τύποι σχέσεων

One-to-Few

- Στο παράδειγμα μας σε ένα χρήστη μπορεί να χρειαστεί να αποθηκεύσουμε πολλές διευθύνσεις που σχετίζονται με αυτόν.
- Αυτό αποτελεί ένα παράδειγμα μιας σχέσης **one-to-few**.

```
{
  "_id": ObjectId('AAA'),
  "name": "Joe Karlsson",
  "company": "MongoDB",
  "twitter": "@JoeKarlsson1",
  "twitch": "joe_karlsson",
  "tiktok": "joekarlsson",
  "website": "joekarlsson.com",
  "addresses": [
    { "street": "123 Sesame St", "city": "Anytown", "cc": "USA" },
    { "street": "123 Avenue Q", "city": "New York", "cc": "USA" }
  ]
}
```

Κανόνας 1: Κάνουμε χρήση του embedding εκτός και εάν υπάρχει επιτακτικός λόγος να μην το κάνουμε.



MongoDB - Τύποι σχέσεων

One-to-Many

- Στο παράδειγμα μας ένα προϊόν αποτελείται από πολλά επιμέρους εξαρτήματα. Κάθε εξάρτημα έχει document με σχετικές πληροφορίες για αυτό. Σε αυτή στη collection Products δημιουργούμε ένα array πεδίο parts που κάνει reference στο id του εξαρτήματος από τη collection Parts
- Αυτό αποτελεί ένα παράδειγμα μιας σχέσης **one-to-many**.



MongoDB - Τύποι σχέσεων

One-to-Many

Κανόνας 2: Αν δε χρειάζεται να έχουμε όλα τα δεδομένα για ένα array από subdocuments μετά από ένα request. Αλλά είναι σημαντικό αυτή η σχέση να διατηρείται στο σχήμα μας. Τότε αυτό αποτελεί λόγο για να μην τα ενσωματώσουμε αλλά να τα κάνουμε reference.

Κανόνας 3: Συνήθως αποφεύγουμε τα joins/lookups αν όμως μπορούν να παρέχουν καλύτερο σχεδιασμό σχήματος τα χρησιμοποιούμε.

```
Products:
{
  "name": "left-handed smoke shifter",
  "manufacturer": "Acme Corp",
  "catalog_number": "1234",
  "parts": [ObjectID('AAAA'), ObjectID('BBBB'), ObjectID('CCCC')]
}

Parts
{
  "_id" : ObjectID('AAAA'),
  "partno" : "123-aff-456",
  "name" : "#4 grommet",
  "qty": "94",
  "cost": "0.94",
  "price": " 3.99"
}
```



MongoDB - Τύποι σχέσεων

One-to-Squillions

- Στη περίπτωση που έχουμε ένα σχήμα όπου θα μπορούσαν να υπάρχουν δυνητικά εκατομμύρια δευτερεύοντα έγγραφα ή περισσότερα. Για παράδειγμα μας ένας logging server όπου έχουμε τους hosts και κάθε host έχει πολλά logs. Σε αυτή τη περίπτωση στην collection log σε κάθε document αυτής έχουμε μια αναφορά στο host.
- Αυτό αποτελεί ένα παράδειγμα μιας σχέσης **one-to-squillions**.



MongoDB - Τύποι σχέσεων

One-to-Squillions

Κανόνας 4: Οι πίνακες δεν πρέπει να μεγαλώνουν χωρίς περιορισμό.

- Εάν υπάρχουν περισσότερα από μερικές εκατοντάδες έγγραφα στην πλευρά των "πολλών", δεν τα ενσωματώνουμε.
- Εάν υπάρχουν περισσότερα από μερικές χιλιάδες έγγραφα στην πλευρά των "πολλών", δεν χρησιμοποιούμε αναφορές σε ObjectId.

Host:

```
{
  "_id": ObjectId('AAAB'),
  "name": "goofy.example.com",
  "ipaddr": "127.66.66.66"
}
```

Log Message

```
{
  "time": ISODate("2014-03-28T09:42:41.382Z"),
  "message": "cpu is on fire!",
  "host": ObjectId("AAAB")
}
```



MongoDB - Τύποι σχέσεων

Many-to-Many

Ας υποθέσουμε ότι έχουμε μια εφαρμογή όπου κάθε χρήστης μπορεί να έχει πολλά tasks και κάθε task μπορεί να έχει πολλούς χρήστες. Για να διατηρηθούν αυτές οι σχέσεις μεταξύ χρηστών και tasks, θα πρέπει να υπάρχουν αναφορές από έναν χρήστη στα πολλά tasks και αναφορές από το ένα task στους πολλούς χρήστες.

Δομούμαι τα δεδομένα έτσι κατά τέτοιο τρόπο ώστε αυτά να ταιριάζουν με τους τρόπους με τους οποίους η εφαρμογή υποβάλλει ερωτήματα και τα ενημερώνει.

```
User:
{
  "_id": ObjectId("AAF1"),
  "tasks": [ObjectId("ADF9"), ObjectId("AE02"), ObjectId("AE73")]
}

Tasks
{
  "_id": ObjectId("ADF9"),
  "owners": [ObjectId("AAF1"), ObjectId("BB3G")]
}
```

Κανόνας 5: Στη MongoDB, ο τρόπος με τον οποίο μοντελοποιούμε τα δεδομένα μας εξαρτάται – εξ ολοκλήρου – από τα μοτίβα πρόσβασης στα δεδομένα της εφαρμογής.



Χρήσιμα Site

<https://www.mongodb.com/docs/>

Welcome to the MongoDB Documentation

Find the guides, samples, and references you need to use the database, visualize data, and build applications on the MongoDB data platform.

<https://www.mongodb.com/developer/>

MongoDB Developer Center

The latest MongoDB tutorials, videos and code examples with your languages and tools. A global community of more than 7 million developers. Build something {big} with MongoDB.



Πρακτική εξάσκηση

Για τα παρακάτω ερωτήματα δώστε τα απαραίτητα query

1. Διαγράψτε όλα τα documents από τις συλλογές users & product.
2. Εισάγεται τα documents του αρχείου users.json στη συλλογή users.
3. Εισάγεται τα documents του αρχείου products.json στη συλλογή products.
4. Αναζητήστε τα document που έχουν "area": "area1" και ταξινομήστε τα βάση του πεδίου surname.
5. Αναζητήστε τα document στα οποία το πεδίο quantity του products είναι ≥ 10 , ταξινομήστε τα βάση του πεδίου surname και επιστρέψτε μόνο το πεδίο username.
6. Τροποποιήστε όλους τους χρήστες που έχουν "area": "area1" και αλλάξτε το area1 σε Patisson.



Πρακτική εξάσκηση

7. Τροποποιήστε όλους τους χρήστες που το πεδίο `quantity` του `products` είναι ≥ 10 που και προσθέστε ένα νέο πεδίο `special` με τιμή `true`.
8. Τροποποιήστε όλους τους χρήστες και εισάγεται στο `subdocuments` του πεδίου `phone` το πεδίο `country` με τιμή `greece`.
9. Διαγράψτε από όλους τους χρήστες που το πεδίο `quantity` του `products` είναι ≥ 10 το πεδίο `country` του `phone`.
10. Βρείτε το συνολικό ποσό αγορών που έχουν γίνει καθώς και το σύνολο των προϊόντων που έχουν αγοραστεί
11. Βρείτε ανά προϊόν το συνολικό ποσό αγορών που έχουν γίνει καθώς και το σύνολο των προϊόντων που έχουν αγοραστεί