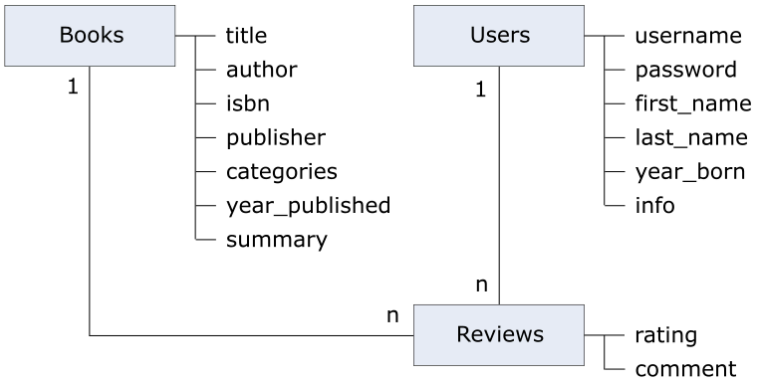
Εργαστηριακό μάθημα 7
-
MongoDB
Υλοποίηση ερωτημάτων

Εισαγωγή

Παρακάτω δίνεται το σχήμα της ΒΔ **Library**, το οποίο περιλαμβάνει τις συλλογές (collections) και τη δομή αυτών (fields). Επιπλέον δίνεται η πραγματική συσχέτιση μεταξύ των collections (χωρίς να μπορεί να εφαρμοστεί με χρήση ξένων κλειδιών - αναφορική ακεραιότητα).



Δημιουργία της ΒΔ

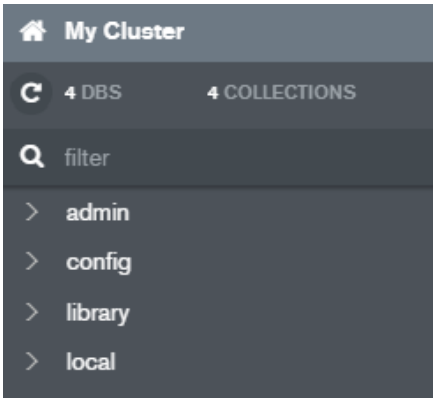
1. Ανοίξτε την εφαρμογή **MongoDB Compass** και συνδεθείτε στη mongo (όπως στο πρώτο εργαστήριο).
2. Αν η βάση library υπάρχει ήδη διαγράψτε τη. Κάνετε κλικ πάνω της, επιλέξτε “Drop database” και στο μενού που θα εμφανιστεί γράψτε το όνομα της βάσης και επιλέξτε “Drop database”.
3. Εισάγετε τα δεδομένα (users, reviews, books) όπως στο πρώτο εργαστήριο μέσα από το MongoDB Compass.

   Εναλλακτικά, ανοίξτε ένα terminal (και μεταφερθείτε στο bin directory της mongo αν δεν είναι στο path) και τρέξτε την παρακάτω εντολή:

   mongorestore --db library --archive=library.dump

(το dump έγινε με mongodump --db library --archive=library.dump)

4. Κάντε refresh από το MongoDB Compass για να βεβαιωθείτε ότι η βάση έχει φορτωθεί σωστά



Είστε έτοιμοι να υποβάλετε ερωτήματα στη βάση σας!

Εναλλακτικά αν θέλετε να εκτελέσετε τα ερωτήματά σας στη γραμμή εντολών, ανοίξτε ένα terminal (και μεταφερθείτε στο bin directory της mongo αν δεν είναι στο path) και τρέξτε την παρακάτω εντολή:

```
mongo
```

Στο shell που θα ανοίξει δώστε την εντολή show databases; και εφόσον βλέπετε τη βάση library δώστε την εντολή use library;

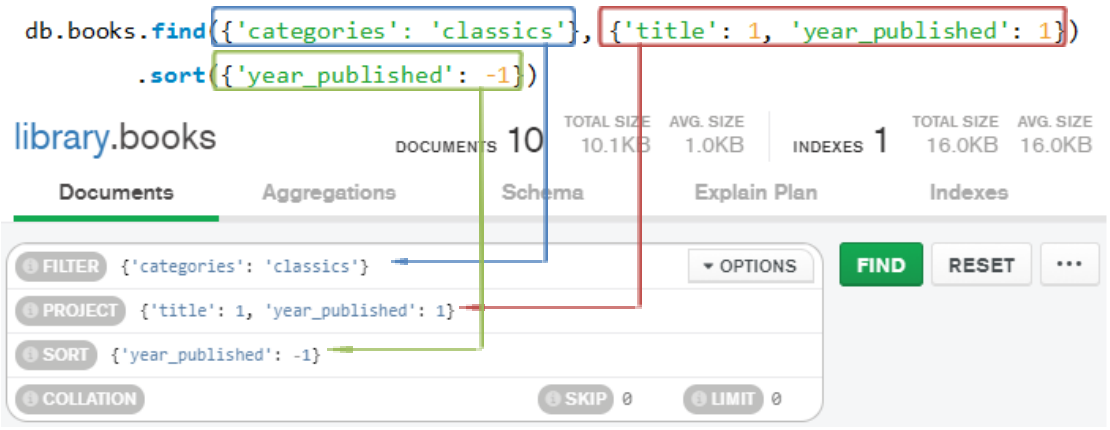
Συγγραφή Ερωτημάτων

Τα ερωτήματα στη εκτελούνται πάνω σε κάποιο collection. Η δομή τους είναι:

```
db.COLLECTION.find(FILTER, PROJECT).sort(SORT)
```

όπου *COLLECTION* είναι η συλλογή πάνω στην οποία εκτελείται το ερώτημα, *FILTER* είναι το ερώτημα σε μορφή JSON, *PROJECT* είναι η προβολή στο αποτέλεσμα του ερωτήματος (σε μορφή JSON) και *SORT* είναι ο τρόπος ταξινόμησης των αποτελεσμάτων του ερωτήματος (σε μορφή JSON).

Στην παρακάτω εικόνα μπορείτε να δείτε την αντιστοίχιση ενός ερωτήματος MongoDB με τον τρόπο εισαγωγής του στο MongoDB Compass.



Περισσότερες πληροφορίες σχετικά τη σύνταξη ερωτημάτων στη MongoDB θα βρείτε στον παρακάτω σύνδεσμο:
https://docs.mongodb.com/manual/tutorial/query-documents/

Επιπλέον, για πληροφορίες και παραδείγματα σχετικά με τους τελεστές ερωτημάτων μεταβείτε στον σύνδεσμο:
https://docs.mongodb.com/manual/reference/operator/query/

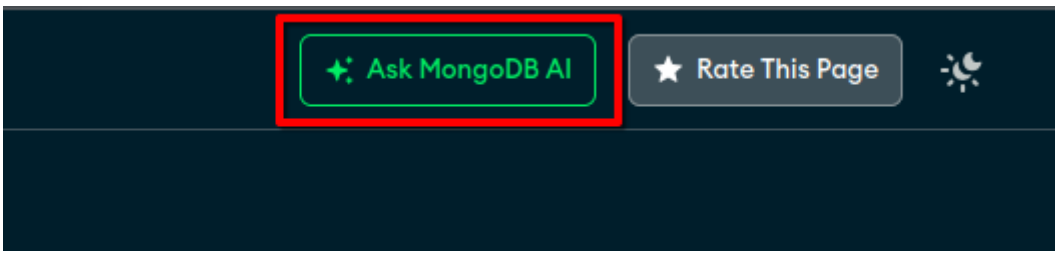
Επίσης, η mongodb παρέχει τη δυνατότητα εκτέλεσης ερωτημάτων τύπου συνάθροισης, ακολουθώντας την παρακάτω δομή,

```
db.COLLECTION.aggregate({“$group”: GROUPING, OPERATION})
```

και αντίστοιχους τελεστές (count, max, sum, κ.α.) για τους οποίους μπορείτε να βρείτε πληροφορίες στους παρακάτω συνδέσμους:

- https://www.mongodb.com/docs/manual/aggregation/
- https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/

Σημείωση: Μπορείτε να χρησιμοποιήσετε το AI που είναι ενσωματωμένο στο documentation της mongodb (πάνω δεξιά στη σελίδα).
https://www.mongodb.com/docs/v7.0/reference/operator/query/



Εργαστηριακές Ασκήσεις

1. ***Να βρεθούν το id, το όνομα (first_name, last_name) και το έτος γέννησης όλων των χρηστών που είναι καταχωρημένοι στην ΒΔ.***

```
db.users.find({}, {first_name: 1, last_name: 1, year_born: 1})
```

2. ***Να βρεθούν οι τίτλοι και τα έτη δημοσίευσης των βιβλίων που δημοσιεύτηκαν μετά το 1900 και να ταξινομηθούν με βάση το έτος σε φθίνουσα σειρά.***

```
db.books.find({year_published: {$gt: 1900}},
              {title: 1, year_published: 1}).sort({year_published:
-1})
```

3. ***Να βρεθούν οι τίτλοι των βιβλίων που έχουν δημοσιευτεί από το 1900 μέχρι το 1960 (συμπεριλαμβανομένων των 1900 και 1960).***

```
db.books.find({$and: [{year_published: {$gte: 1900}},
                      {year_published: {$lte: 1960}}]},
              {title: 1})
```

Εναλλακτικά μπορούμε να γράψουμε το ερώτημα ως:

```
db.books.find({year_published: {$gte: 1900, $lte: 1960}},
              {title: 1})
```

4. ***Να βρεθούν τα βιβλία που έχουν γραφτεί από τον "Charles Dickens".***

```
db.books.find({'author.first_name': 'Charles',
                    'author.last_name': 'Dickens'})
```

5. ***Να βρεθούν οι τίτλοι και οι εκδοτικοί οίκοι των βιβλίων που έχουν εκδοθεί από τον οίκο Penguin ή από τον οίκο Oxford University Press.***

```
db.books.find({$or: [{publisher: 'Penguin'},
                     {publisher: 'Oxford University Press'}]},
              {title: 1, publisher: 1})
```

Εναλλακτικά μπορούμε να γράψουμε το ερώτημα ως:

```
db.books.find({publisher: {$in: ['Penguin', 'Oxford University
Press']}},
              {title: 1, publisher: 1})
```

6. ***Να βρεθούν τα βιβλία που ανήκουν στην κατηγορία science fiction.***

```
db.books.find({categories: 'science fiction'})
```

7. ***Να βρεθούν τα βιβλία για τα οποία η πιο σημαντική κατηγορία (αυτή που βρίσκεται στην πρώτη θέση) είναι η κατηγορία science fiction.***

```
db.books.find({'categories.0': 'science fiction'})
```

8. ***Να βρεθούν τα βιβλία που ανήκουν σε ακριβώς τρεις κατηγορίες.***

```
db.books.find({categories: {$size: 3}})
```

9. ***Να βρεθούν τα βιβλία που ανήκουν τουλάχιστον στις κατηγορίες classics και fiction.***

```
db.books.find({categories: {$all: ['classics', 'fiction']}})
```

Σημειώστε με το ερώτημα χωρίς το keyword **$all**, *δηλαδή* {categories: ['classics', 'fiction']}, *η βάση θα επέστρεφε μόνο τις εγγραφές που έχουν ακριβώς τις κατηγορίες classics και fiction και μάλιστα με αυτή τη σειρά.*

π.χ. ένα έγγραφο με κατηγορίες ['classics', 'fiction'] *αλλά όχι έγγραφα με κατηγορίες* ['classics', 'fiction', 'science fiction'] *ή με κατηγορίες* ['fiction', 'classics'].

10. ***Να βρεθούν οι κριτικές που έχουν γραφτεί από το χρήστη με id 1 και έχουν συνοδευτικό σχόλιο.***

```
db.reviews.find({$and: [{user_id: 1}, {comment: {$exists: true}}]})
```

11. ***Να βρεθούν όλες οι κριτικές που δεν έχουν γραφτεί από το χρήστη με id=1.***

```
db.reviews.find({user_id: {$not: {$eq: 1}}})
```

Εναλλακτικά μπορούμε να γράψουμε το ερώτημα ως:

```
db.reviews.find({user_id: {$ne: 1}})
```

12. ***Να βρεθούν όλες οι κριτικές που δεν έχουν βαθμολογία 3 ή 4.***

```
db.reviews.find({$and: [{rating: {$ne: 3}}, {rating: {$ne: 4}}]})
```

Εναλλακτικά μπορούμε να γράψουμε το ερώτημα ως:

```
db.reviews.find({rating: {$nin: [3, 4]}})
```

13. ***Να βρεθούν οι τίτλοι των βιβλίων που αρχίζουν με τη λέξη The.***

```
db.books.find({title: {$regex: 'The .*'}}, {title: 1})
```

14. ***Να βρεθούν οι τίτλοι και οι εκδοτικοί οίκοι των βιβλίων που έχουν εκδοθεί από οίκους που περιέχουν το Penguin στο όνομά τους.***

```
db.books.find({publisher: {$regex: '.*Penguin.*'}},
              {title: 1, publisher: 1})
```

15. ***Να βρεθούν οι τίτλοι των βιβλίων για τα οποία έχει κάνει κριτική ο χρήστης που έχει id 1.***

Σημειώστε πως η απάντηση δεν είναι δυνατό να αποτελείται από ένα ερώτημα. Χρειάζεται αρχικά το παρακάτω ερώτημα για τις κριτικές του χρήστη με id 1:

```
db.reviews.find({user_id: 1}, {book_id: 1})
```

Και εφόσον έχουμε τα ids των βιβλίων εκτελούμε το παρακάτω ερώτημα για κάθε id (π.χ. για id ίσο με 4):

```
db.books.find({book_id: 4}, {title: 1})
```

16. ***Να βρεθούν ο μέσος όρος των rating όλων των κριτικών, συνολικά (για όλους τους χρήστες) και ανά χρήστη.***

Σημειώστε πως τα δύο ερωτήματα που ζητούνται είναι τύπου συνάθροισης (aggregation), οπότε απαιτείται η δομή aggregate. Το πρώτο ερώτημα είναι το παρακάτω:

```
db.reviews.aggregate({$group: {_id: null}, {total: {$avg:
'$rating'}}})
```

ενώ το δεύτερο ερώτημα είναι το παρακάτω:

```
db.reviews.aggregate({$group: {_id: '$user_id'}, {total: {$avg:
'$rating'}}})
```

Στο MongoDB Compass τα ερωτήματα δίνονται στο tab Aggregations όπως παρακάτω: