Requêtes supplémentaires

Romain Tavenard

1 Avant-propos

Les bases de données utilisées ici sont les mêmes que celles utilisées lors du TD2. Si besoin, pensez à les recharger sur le serveur.

2 La base `test_singlecollec`

Pour chacune des deux manipulations suivantes, vous proposerez deux versions : l'une utilisant la clé `"type"`, l'autre non.

1. Affichez les documents de l'unique collection de la base `test_singlecollec` correspondant à des posts (et non à des auteurs).

```
> db.blog.find({"type": "post"})
> db.blog.find({"title": {$exists: true}})
```

2. Affichez les documents de l'unique collection de la base `test_singlecollec` correspondant à des auteurs (et non à des posts).

```
> db.blog.find({"type": "author"})
> db.blog.find({"firstname": {$exists: true}})
```

3 La base `test`

3. Affichez les titres (mais pas les autres clés) de tous les posts de la base.

```
> db.blog.posts.find({}, {"title": true, "_id": false})
```

4. Affichez la liste des posts de l'auteur Romain Tavenard. Pour cela, il faudra récupérer l'identifiant de cet auteur dans une variable, à l'aide d'un appel à `findOne()`, puis utiliser cette variable dans la requête finale.

```
> author_id = db.blog.authors.findOne({"firstname": "Romain",
                                       "lastname": "Tavenard"})["_id"]
> db.blog.posts.find({"author_id": author_id})
```

4 La base `etudiants`

5. Affichez la liste des étudiantes sans note.

```
> db.notes.find({$or: [{"sexe": "F", "notes": {$size: 0}},
                       {"sexe": "F", "notes": {$exists: false}}]})
```

6. Affichez la liste des étudiants nés en 1995.

```
> date_avant = new Date("1995")
> date_apres = new Date("1996")
> db.notes.find({"ddn": {$gte: date_avant, $lt: date_apres}})
```

7. Affichez la liste des étudiants ayant au moins une note supérieure à 13.

```
> db.notes.find({"notes": {$gt: 13}})
```

8. Affichez la liste des étudiants ayant au moins une note comprise entre 10 et 15. Pour cela, jetez un oeil à l'aide en ligne pour le mot-clé `$elemMatch` : http://docs.mongodb.org/manual/reference/operator/query/elemMatch/

```
> db.notes.find({"notes": {$elemMatch: {$gte: 10, $lte: 15}}})
```

9. Affichez la liste des étudiants ayant toutes leurs notes supérieures ou égales à 10. Ici, il pourra être utile de reformuler la requête sous sa forme négative : on veut tous les étudiants pour lesquels il n'existe pas de note inférieure à 10. Vous aurez donc probablement besoin de l'opérateur `$not` pour lequel vous trouverez de l'aide en ligne : http://docs.mongodb.org/manual/reference/operator/query/not/

```
> db.notes.find({"notes": {$not: {$lt: 10}}})
```

10. Même question en rejetant les étudiants n'ayant pas eu de note.

```
> db.notes.find({$and: [{"notes": {$exists: true, $not: {$size: 0}}},
                        {"notes": {$not: {$lt: 10}}}]})
```

5 La base `food`

11. Combien y a-t-il de restaurants pour lesquels le zipcode est `"10462"` ? Pour mettre en place des conditions sur des clés de sous-document (ici, le sous-document `address`), on peut utiliser la syntaxe suivante :

```
> db.collec.find({"sousdoc.cle": valeur})
```

```
> db.NYfood.find({"address.zipcode": "10462"}).count()
```

12. Affichez la liste des notes attribuées à des restaurants du quartier "Manhattan" (attribut `grades.grade`).

```
> db.NYfood.distinct("grades.grade", {"borough": "Manhattan"})
```

13. Affichez la liste des restaurants ayant au moins une note "C".

```
> db.NYfood.find({"grades.grade": "C"})
```

14. Affichez la liste des restaurants n'ayant aucune note "C".

```
> db.NYfood.find({"grades.grade": {$not: {$eq: "C"}}})
```

15. Affichez la liste des restaurants n'ayant que des notes "A".

```
> db.NYfood.find({"grades.grade": {$not: {$gte: "B"}}})
```