

Les bases de données NoSQL : MongoDB



Chapitre 2 : Opérateurs logiques et de comparaison

Pr BENDAHDANE

Types d'opérateurs

Comme pour les bases de données relationnelles

- opérateurs de comparaison
- opérateurs logiques
- ...

Commençons par créer la collection suivante (etudiant) :

```
{"_id" : 1, "nom" : "wick", "notes": [10, 15, 12], "age" : 19 }  
{"_id" : 2, "nom" : "bob", "notes": [18, 8, 12], "age" : 35 }  
{"_id" : 3, "nom" : "wolf", "notes": [7, 6, 13], "age" : 25 }  
{"_id" : 4, "nom" : "green", "notes": [18, 16, 9], "age" : 22 }
```

exemple

Comment sélectionner les étudiants âgés de plus de 30 ans

```
db.etudiant.find({"age": {$gt:20}})
```

```
{"_id" : 2, "nom" : "bob", "notes": [18, 8, 12], "age" : 35 }  
{"_id" : 3, "nom" : "wolf", "notes": [7, 6, 13], "age" : 25 }  
{"_id" : 4, "nom" : "green", "notes": [18, 16, 9], "age" : 22 }
```

Opérateur de comparaison

Les opérateurs de comparaison

- `$gt` : greater than (supérieur à)
- `$gte` : greater than or equal (supérieur ou égal)
- `$lt` : less than (inférieur à)
- `$lte` : less than or equal (inférieur ou égal)
- `$eq` : equal (égal à)
- `$ne` : not equal (différent de)
- `$in` : dans (un tableau...)
- `$nin` : not in (pas dans)

Opérateur de comparaison

Comment sélectionner les étudiants dont l'âge est entre 30 et 40 ans

```
db.etudiant.find(  
  {$and:  
    [  
      { age:{$gte:20}},  
      { age:{$lte:30}}  
    ]  
  }  
)
```

```
{"_id" : 1, "nom" : "wick", "notes": [10, 15, 12], "age" : 19 }
```

```
{"_id" : 3, "nom" : "wolf", "notes": [7, 6, 13], "age" : 25 }
```

```
{"_id" : 4, "nom" : "green", "notes": [18, 16, 9], "age" : 22 }
```

Opérateurs logiques

Les opérateurs logiques

- `$and` : et
- `$or` : ou
- `$not` : le non logique
- `$nor` : ou exclusif

Opérateurs logiques

Afficher les personnes dont le champ `name` existe

```
db.personne.find(  
  {name:  
    {$exists:true}  
  }  
)
```

Afficher les personnes dont l'age est divisible par 5

```
db.etudiant.find(  
  { age:  
    { $mod: [ 5, 0 ] }  
  }  
)
```

Autres opérateurs

élément

Nom	La description
<code>\$exists</code>	Correspond aux documents qui ont le champ spécifié.
<code>\$type</code>	Sélectionne les documents si un champ est du type spécifié.

Évaluation

Nom	La description
<code>\$expr</code>	Permet l'utilisation d'expressions d'agrégation dans le langage de requête.
<code>\$jsonSchema</code>	Validez les documents par rapport au schéma JSON donné.
<code>\$mod</code>	Effectue une opération modulo sur la valeur d'un champ et sélectionne les documents avec un résultat spécifié.
<code>\$regex</code>	Sélectionne les documents dont les valeurs correspondent à une expression régulière spécifiée.
<code>\$text</code>	Effectue une recherche de texte.
<code>\$where</code>	Correspond aux documents qui satisfont à une expression JavaScript.

Autres opérateurs

Des champs

Nom	La description
<code>\$currentDate</code>	Définit la valeur d'un champ à la date du jour, sous forme de date ou d'horodatage.
<code>\$inc</code>	Incrémente la valeur du champ du montant spécifié.
<code>\$min</code>	Ne met à jour le champ que si la valeur spécifiée est inférieure à la valeur du champ existant.
<code>\$max</code>	Ne met à jour le champ que si la valeur spécifiée est supérieure à la valeur du champ existant.
<code>\$mul</code>	Multiplie la valeur du champ par le montant spécifié.
<code>\$rename</code>	Renomme un champ.
<code>\$set</code>	Définit la valeur d'un champ dans un document.
<code>\$setOnInsert</code>	Définit la valeur d'un champ si une mise à jour entraîne l'insertion d'un document. N'a aucun effet sur les opérations de mise à jour qui modifient les documents existants.

Exercice 1

- ▶ Lancer la commande mongod (le serveur)
- ▶ Décompresser le dossier employes.rar
- ▶ Dans la console, exécuter la commande
mongorestore --db gescom cheminabsoludufichier/employes.bson --port numeroPort
- ▶ Se connecter à la base gescom avec la commande mongo
- ▶ Afficher le contenu de la base de données gescom
- ▶ Afficher le nombre de documents dans la collection employes

Exercice 2

Écrire les requêtes MongoDB qui permettent de :

1. afficher toutes les collections de la base
2. afficher tous les documents de la base
3. compter le nombre de documents de la collection **employes**
4. insérer de deux manières différentes deux employés avec les champs **nom**, **prénom** et soit **prime** soit **ancienneté**
5. afficher la liste des employés dont le **prénom** est David
6. afficher la liste des employés dont le **prénom** commence ou se termine par D
7. afficher la liste des personnes dont le **prénom** commence par D et contient exactement 5 lettres
8. afficher la liste des personnes dont le **prénom** commence et se termine par une voyelle
9. afficher la liste des personnes dont le **prénom** commence et se termine par une même lettre

Exercice 3

10. afficher les **nom** et **prénom** de chaque employé ayant une **ancienneté** > 10
11. afficher les **nom** et **adresse** complète des employés ayant un attribut **rue** dans l'objet **adresse**
12. incrémenter de 200 la **prime** des employés ayant déjà le champ **prime**
13. afficher les trois premières personnes ayant la plus grande valeur d'ancienneté
14. regrouper les personnes dont la ville de résidence est **Toulouse** (afficher **nom**, **prénom** et **ancienneté**)
15. afficher les personnes dont le prénom commence par M et la ville de résidence est soit Foix soit Bordeaux
16. mettre à jour l'adresse de Dominique Mani : nouvelle adresse (`{ numero : 20, ville : 'Marseille', codepostal : '13015' }`). **Attention, il n'y aura plus d'attribut **rue** dans **adresse****

Opérateurs sur les tableaux d'un document

Opérateurs

Listes des opérations

- `$push` : pour ajouter un élément au tableau
- `$pop` : pour supprimer le premier ou le dernier élément d'un tableau
- `$pull` : pour supprimer une ou plusieurs valeurs d'un tableau
- `$pullAll` : pour supprimer tous les éléments d'un tableau
- `$position` : à utiliser avec `push` pour indiquer la position d'insertion dans un tableau
- `$slice` : à utiliser avec `push` pour préciser les éléments à garder dans un tableau
- `$sort` : à utiliser avec `push` pour ordonner les éléments d'un tableau
- ...

Considérons le document suivant :

- `db.personne.insert({ _id : 5, nom : 'wick', sport: ['foot', 'hand', 'tennis'] })`

Ajouter un nouveau sport au tableau

- `db.personne.update({ _id: 5 }, { $push: { "sport": "basket" } })`

Comment ajouter plusieurs sports avec une seule requête ? Ainsi :

- `db.personne.update({ _id: 5 }, { $push: { sport: ['hockey', 'sky'] } })`

Non, ça rajoute un tableau dans notre tableau

Ou comme-ça ?

```
db.personne.update( { _id: 5 }, { $push: { sport: [ 'hockey', 'sky' ] } } )
```

Non, ça génère une erreur

Solution

- `db.personne.update({ _id: 5 }, { $push: { "sport": { $each: ['basket', 'sky'] } } })`

Remarque

- `$push` : ajoute naturellement l'élément après le dernier élément du tableau

Ajouter un élément à une position

Et si on veut ajouter un élément à une position précise

- `db.personne.update({ _id: 5 }, { $push: { "sport": { $each: ['volley'], $position: 2 } } })`

Explication

- Ceci rajoute l'élément `volley` à la position 2 du tableau `sport` (la première position est d'indice 0)
- Les autres éléments seront décalés

Comment supprimer le premier élément d'un tableau ?

- `db.personne.update({ _id: 5 }, { $pop: { sport: -1 } })`

Comment supprimer le dernier élément d'un tableau ?

- `db.personne.update({ _id: 5 }, { $pop: { sport: 1 } })`

Comment supprimer un élément quelconque d'un tableau ?

- `db.personne.update({ _id: 5 }, { $pull: { "sport": "foot" } })` : supprime l'élément `foot` du tableau `sport`

Supprimer plusieurs éléments

Comment supprimer plusieurs éléments avec une seule requête ?

- `db.personne.update({ _id: 5 }, { $pull: { sport: { $in: ['hockey','basket'] } } })`

Considérons le document créé de la façon suivante :

```
db.personne.insert({
  _id : 6,
  nom : 'wick',
  sport : [ 'foot', 'hand', 'tennis' ]
})
```

Si on exécute

```
db.personne.update(
  { _id: 6 },
  { $push: {
    sport: {
      $each: [ 'hockey', 'sky', 'volley' ],
      $slice: -5
    }
  }
})
```

Ensuite

```
db.personne.find({_id:6}).pretty();
```

Le résultat sera :

```
{
  "_id" : 6,
  "nom" : "wick",
  "sport" : [
    "hand",
    "tennis",
    "hockey",
    "sky",
    "volley"
  ]
}
```

Considérons le document créé de la façon suivante :

```
db.personne.insert({
  _id : 7,
  nom : 'wick',
  sport : [ 'foot', 'hand', 'tennis' ]
})
```

Si on exécute

```
db.personne.update(
  { _id: 7 },
  { $push: {
    sport: {
      $each: [ 'hockey', 'sky', 'volley' ],
      $slice: 5
    }
  }
})
```

Ensuite

```
db.personne.find({_id:7}).pretty();
```

Le résultat sera :

```
{
  "_id" : 7,
  "nom" : "wick",
  "sport" : [
    "foot",
    "hand",
    "tennis",
    "hockey",
    "sky"
  ]
}
```

Considérons le document créé de la façon suivante :

```
db.personne.insert({
  _id : 8,
  nom : 'wick',
  sport : ['hand', 'foot', 'tennis']
})
```

Si on exécute

```
db.personne.update(
  { _id: 8 },
  { $push: {
    sport: {
      $each: ['sky', 'volley', 'hockey'],
      $sort: 1
    }
  }
})
```

Ensuite

```
db.personne.find({_id:8}).pretty();
```

Le résultat sera :

```
{
  "_id" : 8,
  "nom" : "wick",
  "sport" : [
    "foot",
    "hand",
    "hockey",
    "tennis",
    "sky",
    "volley"
  ]
}
```

Pour trier dans l'ordre décroissant

- \$sort: -1

Et quand il s'agit d'un tableau d'objet ?

Considérons le document suivant :

```
db.personne.insert({
  _id : 10,
  nom : 'wick',
  notes: [
    {'programmation': 17, 'coefficient': 4},
    {'OS': 10, 'coefficient': 2}
  ]
})
```

Comment ajouter un nouvel élément au tableau ?

```
db.personne.update(
  { _id: 10 },
  { $push:
    { notes :
      { 'compilation': 15, '
        coefficient': 1}
    }
  }
)
```

Et pour supprimer ?

```
db.personne.update(
  { _id: 10 },
  { $pull:
    { notes :
      { 'compilation': 15, 'coefficient': 1}
    }
  }
)
```

Considérons le document suivant :

```
db.personne.insert({
  _id : 11,
  nom : 'wick',
  notes: [
    { 'programmation': 17, 'coefficient': 4,
      optionnel: false},
    { 'OS': 10, 'coefficient': 2}
  ]
})
```


Que fait la requête de suppression suivante ?

```
db.personne.update(
  { _id: 11 },
  { $pull:
    { notes :
      {'programmation': 17, 'coefficient': 4}
    }
  }
)
```

Elle supprime quand-même l'objet même s'il n'y pas de correspondance complète

Comment faire pour éviter cette problématique ?

```
db.personne.update(
  { _id: 11 },
  { $pull: {
    notes : {
      $elemMatch:
        {'programmation': 17, 'coefficient':
          4}
    }
  }
} )
```

Cette fois-ci, ça ne supprime pas l'objet car l'attribut `optional: true` n'a pas été précisé

Pour chercher un document selon une valeur dans son tableau d'objet

```
db.personne.find(  
  {"notes.programmation":  
    {$ne:17}  
  }  
)
```

Cela permet de chercher toutes les personnes dont la note en programmation est différente de 17.

Exercice 4

17. attribuer une prime de 1 500 à tous les employés n'ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris. ²
18. remplacer le champ `tel`, pour les documents ayant un champ `tel`, par un tableau nommé `téléphone` contenant la valeur du champ `tel` (le champ `tel` est à supprimer)
19. créer un champ `prime` pour les documents qui n'en disposent pas et de l'affecter à $100 * \text{nombre de caractère du nom de la ville}$
20. créer un champ `mail` dont la valeur est égale soit à `nom.prénom@formation.fr` pour les employés ne disposant pas d'un champ `téléphone`, soit à `prénom.nom@formation.fr` (`nom` et `prénom` sont à remplacer par les vraies valeurs de chaque employé)