esprit
Se former autrement

Chapitre 5 : MongoDB
Indexation

cdio
CONCEIVE DESIGN IMPLEMENT OPERATE

Enropean Accreditation of Engineering Programmes EUR-ACE®

Délivré par Cti Commission des Titres d'Ingénieur

CONFÉRENCE DES GRANDES ÉCOLES

Plan

- Introduction
- Utilisation de *explain*
- Création et affichage d’un index
- Suppression d’un index
- Types d’index

Introduction

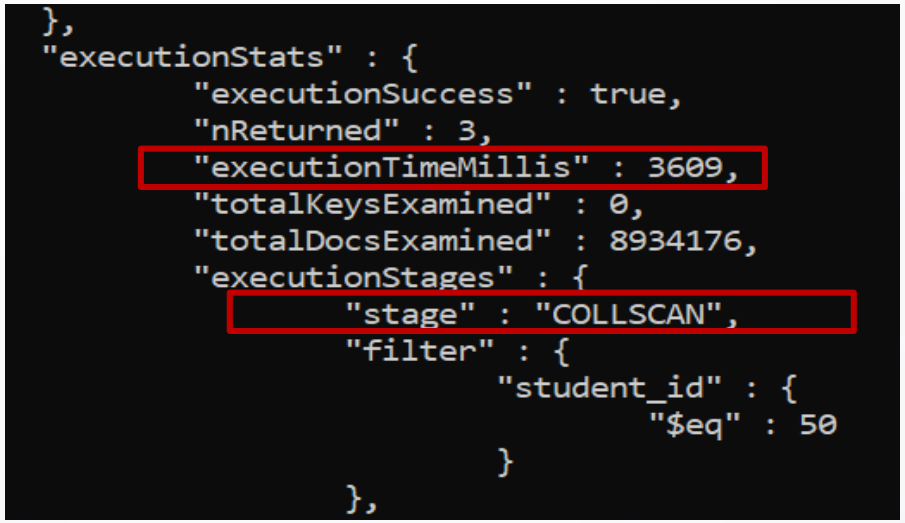
- Les index améliorent les performances en lecture, mais peuvent avoir un impact négatif sur les performances en écriture, car l'insertion d'un document nécessite la mise à jour de tous les index.
- Dans Mongodb les index sont très similaire aux SGBD relationnels, l'indexation dans MongoDB se fait sur un ou plusieurs champs.

Utilisation de *explain*

- explain permet de retourner les informations sur l'utilisation des index

- Exemple :

```
db.students.find({student_id : 50}).explain("executionStats")
```



Création et affichage d'un index

- Création d'un index

```
db.students.createIndex({"student_id":1})
```

- Affichage des index d'une collection

```
db.students.getIndexes()
```

- Information sur l'index créé

```
db.students.find({student_id : 50}).explain("executionStats")
```

Suppression d’un index

- Suppression d’un index

```
db.students.dropIndex("student_id":1)
```

- Suppression de tous les indexes

```
db.students.dropIndexes()
```

Types d'index - Index composé

- L'indexation dans MongoDB se fait sur un ou plusieurs champs. Cela crée un index sur plusieurs champs.

```
db.students.createIndex({"student_id" : 1,"type":1, "score":1},{name:"ind2"})
```

- Dans le cas d'un index composé, MongoDB utilise l'index dans les requêtes dont les critères de recherche sont :
  - ✔ student_id
  - ✔ student_id,type
  - ✔ student_id, type, score

```
db.students.find({student_id : 50}).explain("executionStats")
```

- MongoDB ne peut pas utiliser cet index dans les requêtes dont les critères de recherche sont :
  - ✔ type
  - ✔ score
  - ✔ type, score

```
db.students.find({type:"Essay"}).explain("executionStats")
```

→ Il est impératif de respecter l'ordre de création des index lors de requêtage de la collection.

Types d’index - Index unique

- Appliquer l'unicité à l'index défini (unique ou composé)
- Ne peut être créer que pour des champs à valeur unique

```
db.students.createIndex({"student_id":1},{unique:true})
```

- Remarques :
  - ✔ Si un document n’a pas de valeur pour le champ indexé, l’index aura la valeur nulle.
  - ✔ MongoDB permet l’insertion d’une seule valeur nulle pour un index unique.

Types d'index – *sparse*

- Insertion de documents

```
db.scores.insert({"userid" : "newbie" })
db.scores.insert({"userid" : "abby", "score" : 82 })
db.scores.insert({"userid" : "nina", "score" : 90 })
```

- *sparse* permet d'indexer uniquement les documents contenant des valeurs non nulles d'un champ.

```
db.scores.createIndex( { score: 1 } , { sparse: true } )
```

- Utilisation de l'index créé

```
db.scores.find().sort({ score: 1 }).explain("executionStats")
db.scores.find().sort({ score: -1 }).explain("executionStats")
```

Types d’index - Index partiel : *partialFilterExpression*

- Les index partiels représentent un sur-ensemble des fonctionnalités offertes par les index.
- Ils déterminent les entrées d'index en fonction du filtre spécifié.
- Exemple :

```
db.students.createIndex({ score: 1 },{ partialFilterExpression: { score: { $gt: 50 } } })
```

- Utilisation de l’index :

```
db. students.find( { score: 62} ). explain("executionStats")
```

→ Utilise l’index créé

```
db. students.find( { score: 15 } ). explain("executionStats")
```

→ N’utilise pas l’index créé

Type d’index - *hint*

- *hint* force l’utilisation d’un index spécifique pour répondre à la requête.
- Exemple :

  Forcer l’utilisation de l'index student_id :

```
db.students.find().hint({"student_id" : 1}).explain("executionStats")
```

Types d’index - Index textuel

- MongoDB fournit des index de texte pour prendre en charge les requêtes de recherche de texte sur le contenu de chaîne. Les index de texte peuvent inclure n'importe quel champ dont la valeur est une chaîne ou un tableau d'éléments de chaîne.
- Exemples :

```
db.livres.createIndex({description:"text"})
```

```
db.livres.find({$text:{$search:"serveur"}})
```

```
db.livres.find({$text:{$search:"serveur",$caseSensitive:true}})
```

```
db.livres.find({$text:{$search:"\"Javascript cote\""}},{description:1})
```

```
db.livres.find({$text:{$search:"Javascriptcote"}},{score:{$meta:"textScore"}}).sort({score:{$meta:"textScore"}})
```

Types d'index - Index géo-spatial (1/2)

- On utilise l'index de type '2d' pour les données stockées en tant que points avec deux coordonnées.
- Exemple 1 :

{"city": "GOODWATER", "loc": [-86.078149, 33.074642], "pop": 3813, "state": "AL", "_id": "35072"}

- Créer un index géo-spatial. (type:1 : ordre ascendant)

```
db.cities.createIndex({loc:'2d',type:1})
```

- Exemple 2 :

Afficher les trois stores qui existent à proximité de la position[50,50] :

```
db.cities.find ({loc: { $near : [50,50] }}).limit(3)
```

Types d’index - Index géo-spatial (2/2)

- Création d’un document

```
db.places.insert ( { name: "Central Park", location: [ -73.97, 40.77], category: " Parks" } )
```

- Création d’un index géo-spatial

```
db.places.createIndex( { location: '2d' } )
```

- La requête suivante utilise l'opérateur $near pour renvoyer des documents situés à au moins 100 mètres et au plus 500 mètres du point GeoJSON spécifié, triés dans l'ordre, du plus proche au plus éloigné.

```
db.places.find( { location: { $near: [ -173.9667, 140.78 ] , $minDistance: 100, $maxDistance: 500}})
```