



mongoDB

INDEXATION

Plan

- Création d'index
- Utilisation de explain
- Types d'index :
 - Index unique
 - Index composé
 - Index partiel
 - Index textuel
 - Index géo-spatiale

Indexation

- Très similaire aux SGBDR, l'indexation dans MongoDB se fait sur un ou plusieurs champs.
- Permet d'améliorer les performances de recherche.
- Les indexes sont stockés au niveau des collections.
- Le fonctionnement interne est très proche de ce que l'on trouve dans les SGBD actuels.
- Un champ peu requêté n'a aucun intérêt à être indexé
- Utilisé dans une requête : select, update, sort
- Recherche plus rapide mais écriture plus lente car mise a jour des index

Création et suppression

- Création d'un index

```
db.students.createIndex({"student_id" : 1},{name:"ind1",unique: true})
```

- Affichage des index d'une collection

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

- Suppression d'un index

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

```
db.students.dropIndex("ind1")
```

- Suppression de tous les indexes

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

Index composé

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

Index unique

- Index unique: pas de duplication de valeurs.
- Ne peut être créée que pour des champs à valeur unique.

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

- Remarque :
 - 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

Sparse

```
{"userid" : "newbie" }
```

```
{"userid" : "abby", "score" : 82 }
```

```
{"userid" : "nina", "score" : 90 }
```

- sparse : indexer que les documents contenant des valeurs non nulles de student_id

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

```
db.scores.find().sort( { score: -1 } ) // n'utilise pas l'index
```

```
Forcer l'utilisation de l'index db.scores.find().sort( { score: -1 } ).hint( { score: 1 } )
```

Index partiel

- Une version plus général de sparse.

```
db.contacts.createIndex({ name: 1 },{ partialFilterExpression: { name: { $exists: true } } })
```

- Exemple :

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

- Utilisation index :

```
db. students.find( { score: 62 } ) // index utilisé
```

```
db. students.find( { score: 15 } ) // n'utilise pas l'index
```

Utilisation de explain

explain : retourne les informations sur l'utilisation des index

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

```
    "inputStage" : {
        "stage" : "IXSCAN",
        "nReturned" : 3.0,
        "executionTimeMillisEstimate" : 0.0,
        "works" : 4.0,
        "advanced" : 3.0,
        "needTime" : 0.0,
        ...
        "isUnique" : false,
        "isSparse" : false,
        "isPartial" : false,
        "indexVersion" : 2.0,
        "direction" : "forward",
        "indexBounds" : {
            "student_id" : [
                "[50.0, 50.0]"
            ],
            ...
        }
    },
    "executionSuccess" : true,
    "nReturned" : 3.0,
    "executionTimeMillis" : 6.0,
    "totalKeysExamined" : 3.0,
    "totalDocsExamined" : 3.0,
    "executionStages" : {
        "stage" : "FETCH",
        "nReturned" : 3.0,
        ...
    }
}
```

Hint

- Forcer l'utilisation de l'index student_id

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

Recherche textuelle

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

Index géo-spatiale

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

Exemple :

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

- Créer un index géo-spatiale. Type:1 : ordre ascendant

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

- Les indexes de type '2dsphere' supportent les données stockées en tant qu'objet de type **GeoJSON**

Exemple :

```
{"_id":111,"location":{"coordinates":[-73.961704,40.662942],"type":"Point"},"name":"Wendy'S"}
```

- Créer un index géo-spatiale sphérique

```
db.stores.createIndex({location:'2dsphere'})
```

Index géo-spatiale : Exemples

- index géo-spatiale

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

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

- index géo-spatiale sphérique

Pour chercher un point de type GeoJSON, l'opérateur \$near requiert un index de type 2dsphere.

Exemple : Afficher les stores qui sont proche de max 1km du point dont les coordonnées sont [-130, 39]

```
db.stores.find({ location:{ $near:{ $geometry: { type: "Point", coordinates: [-130, 39]}, $maxDistance:1000 } } })
```