Rapport mongo – BAILLY – GIBAND

1 – créer et insérer un étudiant (étudiant est une variable JSON)

2 – db.etudiants.findOne() 🡪 les 3 boutons affichés : Affiche l’arborescence complète, affiche sous forme de tableau, JSON

3 – db.etudiants.find() retourne tous les étudiants.

Requetes :

1 – db.etudiants.find(

{‘adresse.ville’ : {$eq : ‘Dijon}}

)

2 – db.etudiants.find(

{‘adresse.ville’ : {$eq : ‘Dijon’}},

{\_id :0, nom :1}

)

3 – db.etudiants.find(

db.etudiants.find( { $or : [

{

‘adresse.ville’ : {$eq : ‘Dijon’},

‘adresse.ville’ : {$eq : ‘Annecy’}

}

]

{\_id :0, nom :1}

)

4 - db.etudiants.find( {$or: [

{'adresse.ville' : {$eq : 'Dijon'},

'adresse.ville' : {$eq : 'Annecy'} }]},

{\_id:0, nom:1}

).sort({nom:-1});

5 - db.etudiants.find(

{

$or:

[

{'adresse.ville' : {$eq : 'Dijon'},

'adresse.ville' : {$eq : 'Annecy'} }

]

},

{\_id:0, nom:1}

).sort({nom:-1});

6 - db.etudiants.find(

{ nom:

{ $in:

['DURAND', 'EVENOT']

}

},

{\_id:0, nom:1}

).sort({nom:-1});

7 - db.etudiants.find(

{

'adresse.ville': {

$ne: 'Dijon'

}

}, {\_id:0, nom:1}

).sort({nom:-1});

8 - db.etudiants.find(

{

$or: [{

'adresse.ville': {$ne: 'Dijon'},

'adresse.ville': {$ne: 'Annecy'}

}]},

{\_id:0, nom:1}

).sort({nom:-1});

9 - db.etudiants.find(

{

{ 'prenoms': {$nin: ['Vincent', 'Marc']}

}

}, {\_id:0, nom:1}).sort({nom:-1});

10 - db.etudiants.find(

{ $and: [

{'adresse.ville': {$eq: 'Annecy'}},

{'adresse.rue': {$ne: 'Arc en Ciel'}}

] }, {\_id:0, nom:1} ).sort({nom:-1});

11 - db.etudiants.find(

{'annee experience': { $gt: 3 }} ).sort({nom:-1}

);

12 - db.etudiants.find(

{'annee experience': {$exists : true}}

, {\_id:0, nom:1}

).sort({nom:-1});

13 - db.etudiants.find(

{prenoms: {$elemMatch: {$regex : '^Al'}}}

)

14 - db.etudiants.aggregate([

{$skip : 1}

]);

15 - db.etudiants.aggregate( [

{$sort : {nom: -1}}

] );

16 - db.etudiants.aggregate( [

{ $group:

{ \_id: "$item",

avgAE: { $avg: "$annee experience" }

}

}

] );

17 - db.etudiants.aggregate( [ {

$group: { \_

id: "$adresse.ville",

avgAE: { $avg: "$annee experience" }

}

} ] );

18 - db.etudiants.aggregate( [

{$match: { 'adresse.ville': { $eq: 'Annecy' } } },

{ $count: "cptDijon" }

] );

19 - db.etudiants.aggregate( [

{ $match: { 'annee experience': { $exists: true } } },

{ $count: "cptexpPro" }

] );

20 - db.etudiants.distinct('nom').length;

21 - db.etudiants.aggregate( [ {

$unwind: "$prenoms"},

{"$group": { "\_id": { "$toLower": "$prenoms" }, "count": { "$sum": 1 } } },

{ $sort : { \_id : -1}

} ] )

Création de la collection formations :

var formation1={ nomformation: "dutinfo", intitule: "DUT informatique", annee: "2005", composante:"IUT Annecy" }

var formation2={ nomformation: "lpbdd", intitule: "License professionnelle Bases de Données", annee: "2011" }

var formation3={ nomformation: "masterstic", intitule: "Master Informatique et Systèmes coopératifs", annee: "2005", composante: "SCEM" }

var formation4={ nomformation: "irc", intitule: "Informatique et réseaux de communication", annee: "2007" }

db.formations.insertMany([formation1, formation2, formation3, formation4]);

liaison des collections :

db.etudiants.update( {'\_id':1}, { $set:{"formation" : "5cf65bdc3e2243c96f942f0c"} } ) db.etudiants.update( {'\_id':2}, { $set:{"formation" : "5cf65bdc3e2243c96f942f0d"} } ) db.etudiants.update( {'\_id':3}, { $set:{"formation" : "5cf65bdc3e2243c96f942f0e"} } )

Lookup

db.etudiants.aggregate([{

$lookup: {from: "formations",

localField: "formations.\_id",

foreignField: "formation",

as: "formation»}}])

Indexs :

Un index Composé est une structure contenant des références à plusieurs champs de documents d’une collection.

db.collection.createIndex({ <field1>: <type>, <field2>: <type2>, ... })

Les index simples sont définis par défaut par MongoDB, sur l’\_id. L’index simple parcourt cet index dans n'importe quel sens.

db.records.createIndex( { score: 1 } )

Les indexes Unique sont des indexes uniques, qui assurent à l’utilisateur qu’aucun duplicata de valeur ne soit stocké. Les indexes Primaires sont la même chose, mais créés de base sur les champs primaires.

db.collection.createIndex( <key and index type specification>, { unique: **true** } )

Un index textuel offre un support pour les recherches de texte. Il peut contenir n’importe quelle chaine de caractères ou tableau de chaines de caractères. Il peut aussi porter sur plusieurs champs.

db.reviews.createIndex( { comments: "text" } )

Les indexes hachés utilisent une fonction de hachage afin de calculer la clé de la valeur du champ de l’index. Attention, la fonction de hachage peut gérer plusieurs champs, mais pas les tableaux.

db.collection.createIndex( { \_id: "hashed" } )

Les index MultiKey permettent à indexer le contenu stocké dans des tableaux. Le 1 ou le -1 spécifiés à la création de l’index correspond à l’ordre d’indexation.   
NB : MongoDB gère la création du multikey Index seul, et il n’est donc pas différent d’un index normal à la création.

db.coll.createIndex( { nom: < 1 or -1 > } )

Les indexes épars sont appelés ainsi car ils ne contiennent pas tous les documents de la BD, mais uniquement ceux qui possèdent le champ indexé (même si sa valeur est nulle).

db.addresses.createIndex( { "\_id": 1 }, { sparse: **true** } )

L’index 2dSphere permet de faire des requêtes qui calcule la géométrie sur une sphère terrestre, et l'index 2d est utilisé pour les données stockées sous forme de point sur un plan en 2 dimensions.

db.collection.createIndex( { coordinates : "2dsphere" } )

Les index TTL sont des index spéciaux que MongoDB peut utiliser pour supprimer automatiquement les documents d’une collection après un certain temps.

db.eventlog.createIndex(

{ "nom": 1 },

{ expireAfterSeconds: 3600 }

)

L'index partiel indexe seulement les documents d'une collection qui correspondent à un filtre spécifié lors de la création.

db.restaurants.createIndex(

{ annee experience: 1, name: 1 },

{ partialFilterExpression: { rating: { $gt: 2 } } }

)

L'index d'insensibilité à la casse permet de faire des requêtes qui comparent des chaînes de caractères sans faire attention à la casse.

db.collection.createIndex(

{ "nom" : 1 },

{ collation: { locale : ‘en’, strength : 1|2 } }

)