Requêtage MongoDB

* Procédure d’installation de MongoDB et Robo3T en annexe de la présentation
* Récupération des données depuis : <https://github.com/mdeverdelhan/nosql-resources/tree/master/mongo>
* Documentation du *dataset* : <https://docs.atlas.mongodb.com/sample-data/sample-training/#std-label-sample-training>

# Posts de blog (Collection "posts")

1. Lister tous les posts

db.posts.find()

1. Lister tous les posts en ne conservant que les champs \_id, title et author

db.posts.find({},{"title" : 1,"author":1})

1. Lister tous les posts en ne conservant que les champs title et author

db.posts.find({},{"title" : 1,"author":1,"\_id":0})

# Tweets (Collection "tweets")

1. Lister tous les tweets en ne conservant que le texte et le nom de l'auteur

db.tweets.find({},{"text" : 1,"user.name":1,"\_id":0})

1. Lister tous les tweets écrits par l'auteur Andre Valadao

db.tweets.find({"user.name":"Andre Valadao"})

# Posts de blog (Collections "posts")

1. Lister tous les posts dont le titre est "US Constitution"

db.posts.find({"title":"US Constitution"})

1. Lister les 5 premiers posts dont le titre est "US Constitution"

db.posts.find({"title":"US Constitution"}).limit(5)

1. Lister les 5 posts suivants dont le titre est "US Constitution" (en sautant les 5 premiers)

db.posts.find({"title":"US Constitution"}).limit(5).skip(5)

1. Insérer un post comportant un titre, un auteur et un contenu

db.posts.insert({"title": "Un titre", "author":"Un auteur", "body":"Un corps de texte"})

1. Insérer un post comportant un titre, un auteur et un contenu et l'ID abcdef0123456

db.posts.insert({"\_id": "abcdef0123456", "title": "Un 2e titre", "author":"Un 2e auteur", "body":"Un 2e corps de texte"})

1. Insérer un post comportant le même \_id que le post inséré par la requête précédente

E11000 duplicate key error collection: sample-training.posts index: \_id\_ dup key: { \_id: "abcdef0123456" }

1. Rechercher les deux documents précédemment créés

db.posts.find({"author":"Un auteur"})

db.posts.find({"\_id":"abcdef0123456"})

1. Insérer un document ne comportant que des champs n'existant pas dans les documents de la collection

db.posts.insert({"\_id": "azerty", "champ1": "champ1", "champ2":"champ2", "champ3":"champ3"})

1. Mettre à jour le document précédemment créé avec un nouveau champ4 et en modifiant le champ1

db.posts.update({"\_id": "azerty"}, {"champ1":"maj1", "champ4":"44444"})

1. Rechercher le document mis à jour par la requête précédente. Quelle conclusion tirer du comportement de l’update ?

L’update remplace le document existant. Il ne l’enrichit pas.

# Tweets (Collection "tweets")

1. Lister tous les tweets des utilisateurs ayant plus de 100000 abonnés

db.tweets.find({"user.followers\_count":{$gt : 100000}})

1. Lister les noms des auteurs ayant plus de 100000 abonnés

db.tweets.find({"user.followers\_count":{$gt : 100000}}, {"user.name":1, "\_id":0})

1. S'il y a des doublons dans les résultats de la requête précédente : les enlever

db.tweets.distinct("user.name", {"user.followers\_count":{$gt : 100000}})

1. Lister les tweets des utilisateurs ayant entre 41 et 43 abonnés

db.tweets.find({"user.followers\_count":{$gt : 41, $lt : 43}})

1. Lister les tweets des utilisateurs ayant entre 41 et 43 abonnés en incluant les bornes

db.tweets.find({"user.followers\_count":{$gte : 41, $lte : 43}})

# Entreprises de Crunchbase (Collection "companies")

1. Lister toutes les entreprises ayant des bureaux à Seattle

db.companies.find({"offices":{$elemMatch: {"city":"Seattle"}}})

db.companies.find({"offices.city":"Seattle"})

1. Lister toutes les entreprises ayant fait au moins une levée de fonds depuis 2010

db.companies.find({"funding\_rounds":{$elemMatch: {"funded\_year": {$gte: 2010}}}})

db.companies.find({"funding\_rounds.funded\_year": {$gte: 2010}})

1. Lister les noms de toutes les entreprises ayant "social" pour code de catégorie

db.companies.find({"category\_code": "social"}, {"name":1})

1. Lister les entreprises de la catégorie "social" et ayant plus de 1000 employés

db.companies.find({"category\_code": "social", "number\_of\_employees":{$gt: 1000}})

1. Lister les entreprises qui n'ont pas levé de fonds

db.companies.find({"funding\_rounds": []})

1. Lister les entreprises qui ont été introduites en bourse, mais n'ont jamais levé de fonds

db.companies.find({"funding\_rounds": [], "ipo.valuation\_amount":{$ne:null}})

1. Lister les entreprises qui ont levé des fonds et qui ont été introduites en bourse

db.companies.find({"funding\_rounds": {$ne:[]}, "ipo.valuation\_amount":{$gt:0}})

1. Lister les entreprises qui ont levé des fonds ou qui ont été introduites en bourse

db.companies.find({$or:[{"funding\_rounds": {$ne:[]}}, {"ipo.valuation\_amount":{$gt:0}}]})

1. Lister les noms de toutes les entreprises achetées en 2004, 2005, 2006 et 2013

db.companies.find({"acquisition.acquired\_year": {$in:[2004, 2005, 2006, 2013]}},{name:1})

1. Lister le top 10 des entreprises ayant le plus de salariés (à classer de la plus peuplée à la moins peuplée)

db.companies.find({"number\_of\_employees":{$ne:null}}).limit(10).sort({"number\_of\_employees":-1})

1. Lister les nom, adresse e-mail, numéro de téléphone des entreprises dont le nom commence par "Tech"

db.companies.find({name: /^Tech/},{"name" : 1,"email\_address":1,"phone\_number":1});

1. Lister les nom et bureaux de toutes les entreprises contenant le mot "France" dans leur nom

db.companies.find({name: /.\*France.\*/},{name:1, offices:1});

1. Lister les entreprises qui ont des bureaux en Californie et qui ont un seul produit

db.companies.find({"offices.state\_code": "CA", "products":{$size:1}});

1. Lister les entreprises qui ont plus de 10 produits

db.getCollection('companies').find({ $where: "this.products.length > 10" })

# Posts de blog (Collection "posts")

1. Insérer les posts de blog suivants en une seule commande :

{"\_id": "blogpost\_1", "title": "DotT", "author":"Hoagie", "body":"Great, I'm stuck in colonial times, tentacles are taking over the world, and now the toilet's backing up.", "views":42, "toBeRemoved": false, "score": 98.0}

{"\_id": "blogpost\_2", "title": "Monkey Island", "author":"Guybrush", "body":"Look behind you, a Three-Headed Monkey!", "views":1337, "toBeRemoved": false, "score": 95.0}

{"\_id": "blogpost\_3", "title": "Mon 3e post de blog", "author":"Me", "body":"Ceci est mon 3e post de blog.", "views":10}

{"\_id": "blogpost\_4", "title": "Mon 4e post de blog", "author":"Myself", "body":"Ceci est mon 4e post de blog.", "views":10, "toBeRemoved": true, "score": 56.0}

{"\_id": "blogpost\_5", "title": "Mon 5e post de blog", "author":"You", "body":"Ceci est mon 5e post de blog.", "views":10, "toBeRemoved": true}

{"\_id": "blogpost\_6", "title": "Mon 6e post de blog", "author":"You", "body":"Ceci est mon 6e post de blog.", "views":8, "score": 90.0}

{"\_id": "blogpost\_7", "title": "Mon 7e post de blog", "author":"Myself", "body":"Ceci est mon 7e post de blog.", "views":5, "toBeRemoved": false, "score": 98.0}

db.posts.insertMany([

{"\_id": "blogpost\_1", "title": "DotT", "author":"Hoagie", "body":"Great, I'm stuck in colonial times, tentacles are taking over the world, and now the toilet's backing up.", "views":42, "toBeRemoved": false, "score": 98.0},

{"\_id": "blogpost\_2", "title": "Monkey Island", "author":"Guybrush", "body":"Look behind you, a Three-Headed Monkey!", "views":1337, "toBeRemoved": false, "score": 95.0},

{"\_id": "blogpost\_3", "title": "Mon 3e post de blog", "author":"Me", "body":"Ceci est mon 3e post de blog.", "views":10},

{"\_id": "blogpost\_4", "title": "Mon 4e post de blog", "author":"Myself", "body":"Ceci est mon 4e post de blog.", "views":10, "toBeRemoved": true, "score": 56.0},

{"\_id": "blogpost\_5", "title": "Mon 5e post de blog", "author":"You", "body":"Ceci est mon 5e post de blog.", "views":10, "toBeRemoved": true},

{"\_id": "blogpost\_6", "title": "Mon 6e post de blog", "author":"You", "body":"Ceci est mon 6e post de blog.", "views":8, "score": 90.0},

{"\_id": "blogpost\_7", "title": "Mon 7e post de blog", "author":"Myself", "body":"Ceci est mon 7e post de blog.", "views":5, "toBeRemoved": false, "score": 98.0}

])

1. Vérifier que les documents ont bien été insérés

db.posts.find({\_id: /^blogpost/})

1. Mettre à jour tous les posts de blog ayant toBeRemoved à true en leur attribuant un score de zéro (en une seule commande)

db.posts.updateMany({toBeRemoved: true}, {$set:{score: 0.0}})

1. Supprimer tous les posts ayant un score inférieur à 20 (en une seule commande)

db.posts.deleteMany({score: {$lt:20}})

# Trajets à vélo (Collections "trips")

1. Trouver les 12 premiers voyages classés par ordre alphabétique inversé de nom de station de fin de voyage, et, pour une même station, par durée de voyage croissante

db.trips.find({}).sort({"end station name": -1,tripduration:1}).limit(12)

1. Trouver tous les voyages démarrés entre le 01/01/2016 00:15 et le 01/01/2016 00:20

db.trips.find({"start time": {$gte: ISODate("2016-01-01T00:15:00.000Z"), $lt: ISODate("2016-01-01T00:20:00.000Z")}})

1. Lister, sous forme de documents, les stations de départ (« start station name » pour \_id) avec, pour chacune d’entre-elles, le nombre de voyages partis de la station

db.trips.aggregate([

{ $group : { \_id : '$start station name', totaldocs : { $sum : 1 } } }

])

1. Répéter la requête précédente en stockant le résultat dans une nouvelle collection « groupingResult ».

db.trips.aggregate([

{ $group : { \_id : '$start station name', totaldocs : { $sum : 1 } } },

{ $out : 'groupingResult' }

])

1. Afficher la liste des documents de *groupingResult*

db.groupingResult.find()

1. Créer un document comportant la moyenne des durées de tous les voyages

db.trips.aggregate([{$group:{"\_id":null,trip\_dur\_avg: { $avg: "$tripduration" }}}])

1. Créer le même document sans le champ \_id

db.trips.aggregate([{$group:{"\_id":null,trip\_dur\_avg: { $avg: "$tripduration" }}},

{ $project:{"\_id": 0,trip\_dur\_avg: "$trip\_dur\_avg" }}

])

1. Trouver, pour chaque genre, l’année de naissance moyenne des voyageurs

db.trips.aggregate(

[

{

$group:{"\_id":"$gender",birthyeardbl: { $avg: "$birth year" }}},

{ $project:{"\_id": "$\_id",avg\_birth\_year: {$toInt:{ $ceil: "$birthyeardbl" }}}

}

])

1. Trouver les 10 vélos les plus utilisés (en nombre de trajets) par les utilisateurs de type Subscriber

db.trips.aggregate([

{ $match : { usertype : 'Subscriber' } },

{ $group : { \_id : '$bikeid', totaltrips : { $sum : 1 } } },

{ $sort : { 'totaltrips' : -1 } },

{ $limit : 10 }

])

1. Compter le nombre d’arrivées dans une station par quart d’heure (l’identifiant du document doit comporter les année, mois, jour, heure et minute de début du quart d’heure)

db.trips.aggregate([

{

"$group": {

"\_id": {

"year": { "$year": "$stop time" },

"month": { "$month": "$stop time" },

"day": { "$dayOfMonth": "$stop time" },

"hour": { "$hour": "$stop time" },

"minute": {

"$subtract": [

{ "$minute": "$stop time" },

{ "$mod": [{ "$minute": "$stop time"}, 15] }

]

}

},

"count": { "$sum": 1 }

}

}

])

# Entreprises de Crunchbase (Collections "companies")

1. Compter le nombre de produits de la société Wetpaint

db.companies.aggregate( [

{ $match : { name : 'Wetpaint' } },

{ $project: {\_id: 0, productCount: {$size: "$products"}} }

] )

1. Afficher un document par produit de la société Wetpaint avec, pour chaque document : le nom de la société, le nom du produit

db.companies.aggregate( [

{ $match : { name : 'Wetpaint' } },

{ $unwind: "$products" },

{ $project: {\_id: 0, name: "$name", product\_name: "$products.name"} }

] )

1. Afficher un document par société avec, pour chaque document : le nom de la société, un champ *founded\_before\_2000* qui prendra pour valeur l’année de création de la société si elle a été créé avant 2000 et « FAUX » sinon

db.companies.aggregate([

{

$project: {

\_id: 0,

name: 1,

founded\_before\_2000: { $cond: { if: {$lt: [ "$founded\_year", 2000 ]}, then: "$founded\_year", else: "FAUX"} }

}

}

] )