

Mongo DB

Toute performance, toute modularité

peut contenir
tous plusieurs documents
fond au court

collection = 1 table MySQL
par nature

différence entre mongo db et RDBMS (ref, ...)

grande détails

Mongo DB

- Base de données

- Collection ↘ equivalent ↗
- Document ↘
- champs
- document intégré
- Clef primaire (générée par mongo)

base de données
de l'appli

contenu
objet
de données

RDBMS

- Base de données

- table
- ligne
- colonne
- Jointure ↗ produit
- Clef primaire (donnée générée par table)

I. exemple

restitution primaire de valeurs

{
 " _id " : ObjectId (" 5bf5367a405d ")
 " title " : " cours DB + Aperçu "
 " description " : " donne un aperçu de Mongo "
 " article " : " Qu'est ce que Mongo DB "
 " author " : ObjectId (" 54750c1e173697 ")
 " document_id "

extraction

Dynamique

insertion de données dans une base de données.

⚠️ n'oublie pas mongoDB

- Création d'une base de données

Sur CLI

- use : pour aller ou se déplacer dans le DB (green)
- db : pour dans quel db on se trouve
- show dbs : pour afficher toutes les db

collection (table)

développeur

insertion : db = nom de la collection : insert [{ 'nom' : 'tuto' }]
la bd sélectionnée avec une

Delete db : db = dropDatabase () → supprimer la bd

Sur collections

2 paramètres

createCollection

- db.createCollection (nom du db , option)
(trunc) → nom de la collection 4 option

1° option : capped : true / false

{ capped : true / false , size : 10 }

2

pour définir le
nombre de ligne
(d'entrée dans la
collection)

db.createCollection ('name', { capped : true / false , size : 10 , max : 10 },
autoIndexId (ajouter automatiquement des index sur Id
dans chacune des entrées))

pour définir le
nombre de document
maximal

`show collections` : pour afficher les collections
du db

touche Tab permet d'auto-compléter

`db.nom de la collection.drop()` = pour supprimer
la collection

Les types de données

`integer` : chain de caract

`string` : entier

`double` : (true or false) decimal

`boolean` : true or false

`array` : pour les listes

`object` : à l'index 3

`null` : nulle

`timestamp` :

`object id` :

`Date`

`BsonType Date`

→ - Insertion document dans mongo db - ←

`insert` peut ou non

`db.nom collection.insert({
objet de données
})`

pour recuperer les données

`db.nom de la collection.find()`

Exemple :

db.Etudiant.find()

.pretty()

permet de formater le code en structure → JSON
(apparence)

ex : db.Etudiant.find().pretty()

Il permet aussi du filtre

db.nomCollection.find({ " " : " " })

ex : db.Etudiant.find({ "nom" : "Emane Randri" })

Pour ajouter des paramètres supplémentaires il suffit d'ajouter
le " et " (et) (aussi)

ex : db.nomCollection.find({ " " : " " })
de valeur

Condition ← pour faire la recherche

and or
and \$or

\$or : []

Ex : db.nomCollection.find([

\$or : [{ "cle" : "valeur1" }, { "cle2" : "valeur2" }]

⚠

il faut respecter
les cases

Majuscule

et Min

opérateur

!= ne pas égal !=

> gt > greater than >

>= gte > greater than equal >=

== equal =

< lt < less than <

<= lte < less than equal <=

Ex :

db.nomCollection.find({ "age" : { \$eq : 20 }, \$or : [

{ "nom" : "Emane Randri" }, { "nom" : "Fidjana" }]

Valuer say Et ne retourne que

la première condition

ou

qui permet de vérifier
si la condition correspond
à qui correspond

à la seconde condition

Update

Rechercher puis à jour
id (object id)

(db.collection.update({}))

(db.update) mise à jour du document existant

db.collection.update()

ex:

db.Etudiant.update({ id: ObjectId("...") }, { \$set: { "de":

la partie bind

"valeur" })

matchedCount :

document correspondant

modifiedCount :

nombre de modifications

insertedCount :

nombre d'insertions

modification de plusieurs documents

{ multi : true }

db.collection.update()

ex

db.Etudiant.update({ "age": 20 }, { \$set: { "sexe": "feminin" } })

Résultat tout ce que on connait d'âge d

âge = 20 auront comme update la

modification sexe = feminin

Delete

db. nomcollection.remove()

→ suppression d'un document

db. nomcollection.remove({ "_id": ObjectId("...") })

Projection Limite Sort

lorsque "sortir dans un autre que l'on
se rapporte" → résultats

where **Projection** : renvoyer les documents demandés
find() → select (req)

db. nomcollection.find({ "clé1": valeur1, "clé2": valeur2 })

retour : { Object id, "clé1": valeur, "clé2": valeur }

tout les
Document dont toutes les
conditions
sont remplies

△ : db. nomcollection.find({ "clé": valeur }, { "clé": valeur })
- id : 0 → ne renvoie pas l'identifiant

Limit : permet d'indiquer le nombre de document
que l'on veut renvoyer

find().limit()

ex : db. nomcollection.find({ "clé": valeur }).limit(1)
return → 1 seul document

• non mais tout le information moins celle
• skip(1) les premières sur skip(1)

parce que le nombre d'item que l'on va afficher
ex : db.nomcollection.find().skip(1)
retour → document 2 parce que le doc 1 a été passé

le skip peut être utilisé avec le limit
ex : db.nomcollection.find({}).limit(1).skip()

db.nomcollection.find({}).limit(1).skip(1)
retour → document 2

Sort : permettre de définir un ordre dans lequel
on va afficher mes information

db.nomcollection.find().sort({})

Les valeur sont 1 0 -1 pour Asc et Desc

1 -1

db.nomcollection.find().sort({})

meilleur afficher par ordre croissant

le nombre de la clé string :

string number :

date

orange de 21 afficher

le nombre de la clé

pas dans document

