Brief MongoDB

Creation de base de donnée et les collections.

use globaldb

var collectionList = ["all-sales", "categories", "customers", "employees", "employeeterritories", "orderdetails", "orders", "regions", "products", "shippers", "suppliers", "territories"];

collectionList.forEach(function(collectionName) {db.createCollection(collectionName)})

Importer les fichiers csv ci-joint dans votre base de données.

mongoimport --type csv -d globaldb -c allsales --headerline --drop All-sales.csv

mongoimport --type csv -d globaldb -c categories --headerline --drop categories.csv

mongoimport --type csv -d globaldb -c customers --headerline --drop customers.csv

mongoimport --type csv -d globaldb -c employees --headerline --drop employees.csv

mongoimport --type csv -d globaldb -c employee-territories --headerline --drop employee-territories.csv

mongoimport --type csv -d globaldb -c order-details --headerline --drop order-details.csv

mongoimport --type csv -d globaldb -c orders --headerline --drop orders.csv

mongoimport --type csv -d globaldb -c regions --headerline --drop regions.csv

mongoimport --type csv -d globaldb -c products --headerline --drop products.csv

mongoimport --type csv -d globaldb -c shippers --headerline --drop shippers.csv

mongoimport --type csv -d globaldb -c suppliers --headerline --drop suppliers.csv

mongoimport --type csv -d globalDb -c territories --headerline --drop territories.csv

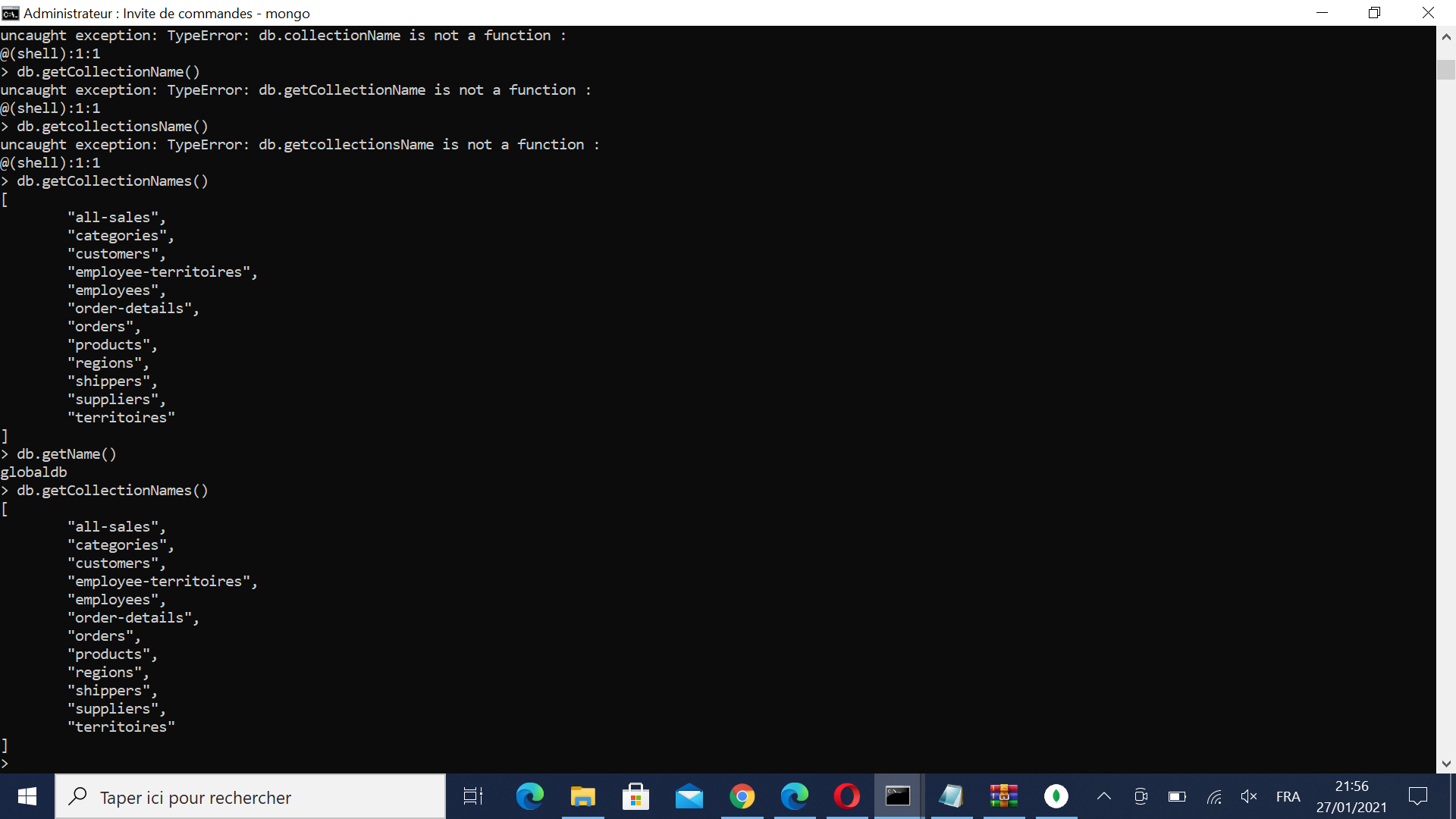
Vérifier que les éléments de votre base de données respectent les conventions de nommage suivantes :

• Le nom de base de données est un mot en minuscule et se termine par db.

• Le nom d’une collection est un mot en pluriel et en minuscule.

db.getName() ;

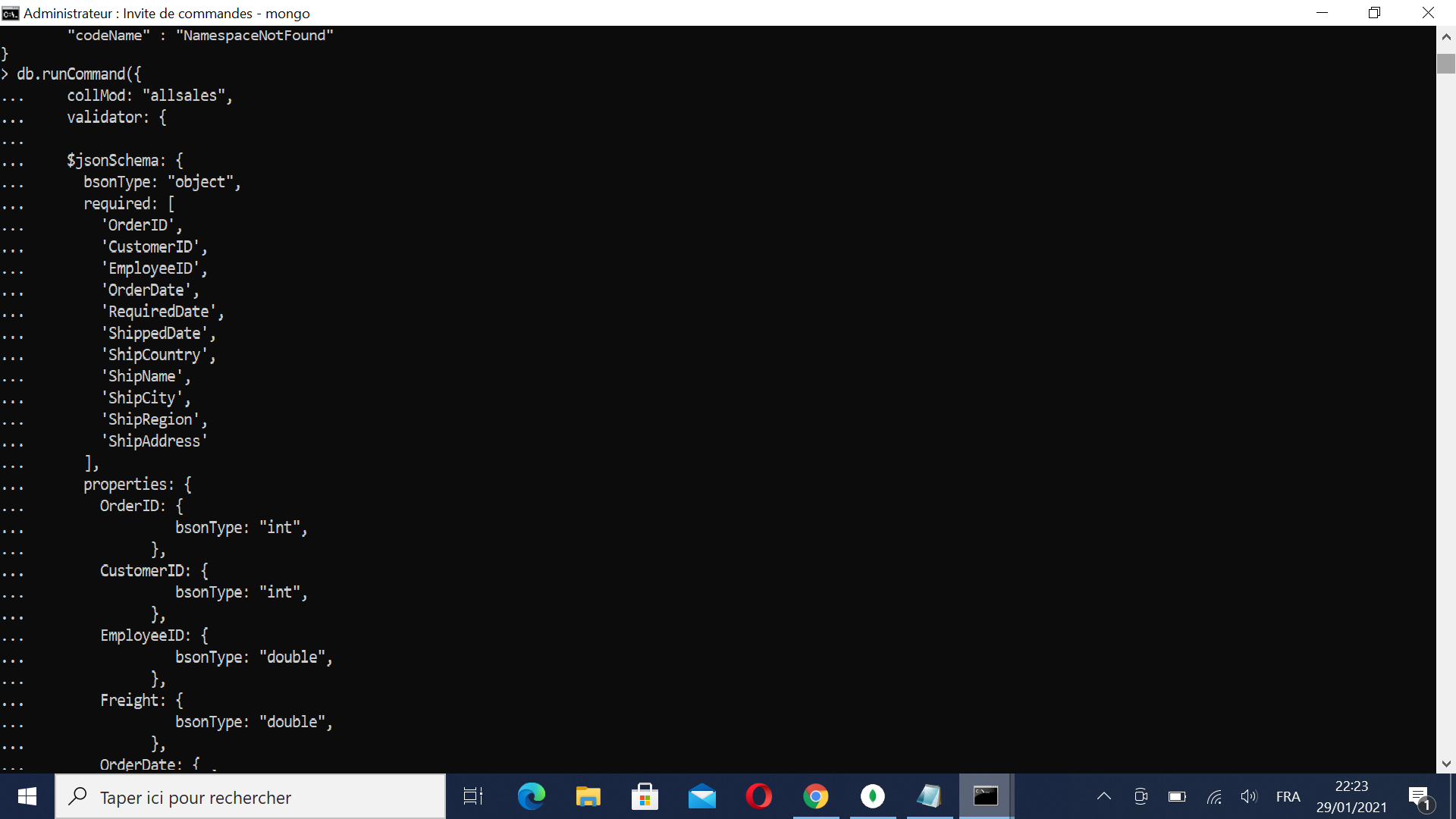
db.getCollectionNames() ;



• Les champs de documents sont en « camel case ».

db.all-sales.validate()

db.adminCommand( { renameCollection: "globalDb.all-sales", to: "globalDb.allsales" } )

Créer des règles de validation.

db.runCommand(

{ collMod: "allsales", validator:

{ $jsonSchema: {

bsonType: "object"

, required: [ 'OrderID',

'CustomerID',

'EmployeeID',

'OrderDate',

'RequiredDate',

'ShippedDate',

'ShipCountry',

'ShipName',

'ShipCity', 'ShipRegion', 'ShipAddress' ], properties: {

OrderID: { bsonType: "int", }, CustomerID:

{ bsonType: "int", },

EmployeeID: {

bsonType: "double", }, Freight: { bsonType: "double", }, OrderDate: { bsonType: "int", }, ShipName: { bsonType: "string", }, ShipAddress: { bsonType: "string", }, ShipCity:

{ bsonType: "string", },

ShipRegion: { bsonType: "string", },

ShipPostalCode: { bsonType: "int", },

ShipCountry: { bsonType: "string", },

ShipVia:

{ bsonType: "int", },

} }

}

});

• Implémenter les bonnes pratiques de sécurité pour votre base de données, à l’instar :

Activer les control d’accès et exiger une authentification.

Use Admin

db.createUser(

{

user: "Admin",

pwd: "Admin",

roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]

}

)

Db.shutdownServer()

mongod –auth

dans le fichier configuration nous avons activé l’authentification

Sécurity:

autorization: enabled

Authentification : db.auth(’Admin’, ’Admin’)

• Configurer des rôles.

db.createRole(

{

role: "globalUser",

privileges: [

{ resource: { db: "globaldb", collection: "" }, actions: [ "find" ,"createCollection","collStats"] },

{ resource: { db: "globaldb", collection: "allsales" }, actions: [ "update", "insert"] },

{ resource: { db: "globaldb", collection: "categories" }, actions: [ "update", "insert"] },

{ resource: { db: "", collection: "" }, actions: [ "find" ] }

],

roles: []

}

)

• Bonnes pratiques de sauvegarde et restauration.

mongodump --db=globaldb --out=C:\data\backup\dump --authenticationDatabase "admin" -u "Admin" -p "Admin"

mongorestore dump/ --authentificationDatabase “admin” –u “Admin” –p “Admin

Créer des utilisateurs dans les règles de l’art des normes du sécurité (e.g. des utilisateurs qui administre les utilisateurs et d’autre qui administrent les bases de données) et affecter à eux les rôles convenables.

db.createUser(

{

user: "user",

pwd:"987654",

roles: [ { role: "globalUser", db: "admin" }]

}

)

Vous devez faire des veilles TEC sur database fine tuning, sharding, replication et capped collections sous MongoDB.

**database tuning** : décrit un groupe d'activités utilisées pour optimiser et homogénéiser les performances d'une [base de données](https://en.wikipedia.org/wiki/Database) . Il chevauche généralement [le](https://en.wikipedia.org/wiki/Query_language) réglage des [requêtes](https://en.wikipedia.org/wiki/Query_language) , mais se réfère à la conception des fichiers de base de données, à la sélection de l'application du [système de gestion de base de données](https://en.wikipedia.org/wiki/Database_management_system) (SGBD) et à la configuration de l'environnement de la base de données ( [système d'exploitation](https://en.wikipedia.org/wiki/Operating_system) , [CPU](https://en.wikipedia.org/wiki/CPU) , etc.).

Le réglage de la base de données vise à maximiser l'utilisation des ressources du système pour effectuer le travail aussi efficacement et rapidement que possible. La plupart des systèmes sont conçus pour gérer leur utilisation des ressources système, mais il reste encore beaucoup à faire pour améliorer leur efficacité en personnalisant leurs paramètres et leur configuration pour la base de données et le SGBD.

**Le sharding** : est une technique pour pouvoir augmenter le nombre d’utilisateurs en améliorant le point souvent critique qui est la capacité de la base de données. Le principe est assez simple et peut se résumer en « diviser pour mieux régner ». Pour une table donnée, on a souvent une donnée que l’on connait avant d’aller la chercher. Ainsi sur une table utilisateur pour un use case d’identification, on connait l’email de l’utilisateur que l’on a demandé. Pareil quand on souhaite créer un utilisateur ou le modifier, on connait son email (de toute façon pour le sharding on a pas le choix, il faut une clé dont on connait tout le temps la valeur).

**la réplication** est un processus de partage d'informations pour assurer la cohérence de données entre plusieurs sources de données redondantes, pour améliorer la fiabilité, la [tolérance aux pannes](https://fr.wikipedia.org/wiki/Tol%C3%A9rance_aux_pannes), ou la [disponibilité](https://fr.wikipedia.org/wiki/Disponibilit%C3%A9). On parle de *réplication de données* si les mêmes données sont dupliquées sur plusieurs périphériques.

La réplication n'est pas à confondre avec une [sauvegarde](https://fr.wikipedia.org/wiki/Sauvegarde) : les données sauvegardées ne changent pas dans le temps, reflétant un état fixe des données, tandis que les données répliquées évoluent sans cesse à mesure que les données sources changent.

**Capped collections** : nous pouvons créer des collections dans mongoDb sur lesquelles nous pouvons appliquer une limite de taille. Ces types spéciaux de collections sont appelés **collections plafonnées** . Il s'agit d'une sorte de files d'attente circulaires, dans lesquelles si la limite de taille allouée est atteinte, elle fait de la place pour de nouveaux documents en écrasant les documents les plus anciens de la collection.

**Comment créer des collections plafonnées:**  
db.createCollection(nameOfCollection, {capped: Boolean, autoIndexId: Boolean, size: Number, max : Number})  
Ici, les paramètres spécifient:

* nameOfCollection: le nom de la collection plafonnée
* Option "plafonnée": définie sur true au cas où créer une collection plafonnée
* Option «size»: spécifiez la taille limite de la collection en octets (obligatoire en cas de collection plafonnée)
* Option «max»: spécifiez la limite du nombre maximal de documents autorisés dans la collection (l'option de taille a la préférence sur l'option max)
* Option «autoIndexId»: définie sur false en cas de collections plafonnées pour désactiver la création automatique d'index