

MOOCEBFR4 : Ingénierie des Données du Big Data : SGBD NoSql et Lacs de Données avec Big Data SQL par la pratique

Gabriel MOPOLO-MOKE
Professeur chargé d'enseignements
Université Côte d'Azur (UCA)

2023 / 2024

Plan Général

➤ Plan

- *Module M4.1 : Rappel sur les concepts du Big Data et des SGBD NOSQL*
- *Module M4.2 : Introduction à Oracle NOSQL*
- *Module M4.3 : Oracle NoSql et le Modèle Key/Document*
- **Module M4.4 : INTRODUCTION A MONGODB ET LE MONGO SHELL**
- **Module M4.5 : INTRODUCTION A MONGODB ET SON API JAVA**
- *Module M4.6 : Architectures Big data et construction de lacs de Données avec Big Data SQL par la pratique*

Module M4.4 : INTRODUCTION A MONGODB ET LE MONGO SHELL

G. Mopolo-Moké
Professeur chargé d'enseignements
Université Côte d'Azur (UCA)

2022 / 2023

Module M4.4 : INTRODUCTION A MONGODB ET LE MONGO SHELL

➤ Plan

- **Module M4.4, section 1 : Objectif et carte de visite MongoDB**
- **Module M4.4, section 2 : Installation, outils architecture MongoDB**
- **Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB**
- **Module M4.4, section 4 : Gestion des Documents MongoDB**
- **Module M4.4, section 5 : Indexation et recherches de documents MongoDB**
- **Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB**
- **Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB**

Module M4.4, section 1 : Objectif et carte de visite MongoDB

Module M4.4, section 1 : Objectif et carte de visite

MongoDB

➤ Plan

- Objectif
- Carte de visite MongoDB

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Objectifs du cours

- Comprendre l'architecture d'une instance mongoDB
- Comprendre les principales caractéristiques de MongoDB et savoir positionner cette base de données par rapport aux autres BD NoSQL
- Pouvoir gérer des bases de données
- Pouvoir gérer des collections
- Pouvoir gérer des documents
- Comprendre le rôle du format JSON pour ce moteur
- Comprendre sa stratégie de réPLICATION
- Savoir poser des index secondaires
- Savoir accéder à un ou plusieurs aux objets (via la clé, via des prédictats, via un index)
- Pouvoir effectuer des jointures, des groupements, etc...

Module M4.4, section 1 : Objectif et carte de visite

MongoDB

➤ Carte de visite de MongoDB

- Editeur
- Versions initial (Nr. De version et date)
- Versions actuelle (Nr. De version et date)
- Modèles de données supportés(Clé/Valeur, Orienté document, Orienté Colonnes, Orienté Graphe)
- Gestion du schéma (sans schéma, dynamique, statique, mixte)
- Typage (none, static, dynamique)
- Support de SQL (DDL, DML)
- Support d'indexes secondaires
- Langage de développement du sgbd nosql
- Support / Pérénitité (communauté , etc....)
- API Supportés
- Théorème CAP (CP, AP, AC)
- Méthode de partitionnement
- Méthode de réPLICATION
- Concept de consistance
- Concept de durabilité
- Clés étrangères
- Support de références (REF)
- Licences et prix

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Carte de visite MongoDB

- Différents types de versions (communautaire, entreprise, ...)
- Audience dans le marché
- Tables et tables filles
- Clé avec major et minor key
- Gestion des utilisateurs
- Gestion des droits
- Gestion des namespaces ou databases
- Systèmes d'exploitations supportés
- Disponible en mode DBaaS
- Applications communautaires l'utilisant
- Domaines d'applications
- Architecture du moteur NoSql
- Montée en charge
- Gestion de la disponibilité

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Carte de visite MongoDB

- **Editeur :** MongoDB inc.
- **Versions initial (Nr. De version et date) :** 2009
- **Versions actuelle (Nr. De version et date) :** 4.0.8, March 2019
- **Modèles de données supportés(Clé/Valeur, Orienté document, Orienté Colonnes, Oriénté Graphe):** Orienté document
- **Gestion du schéma (sans schéma, dynamique, statique, mixte) :** Sans schéma
- **Typage (none, static, dynamique):** dynamique

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Carte de visite MongoDB

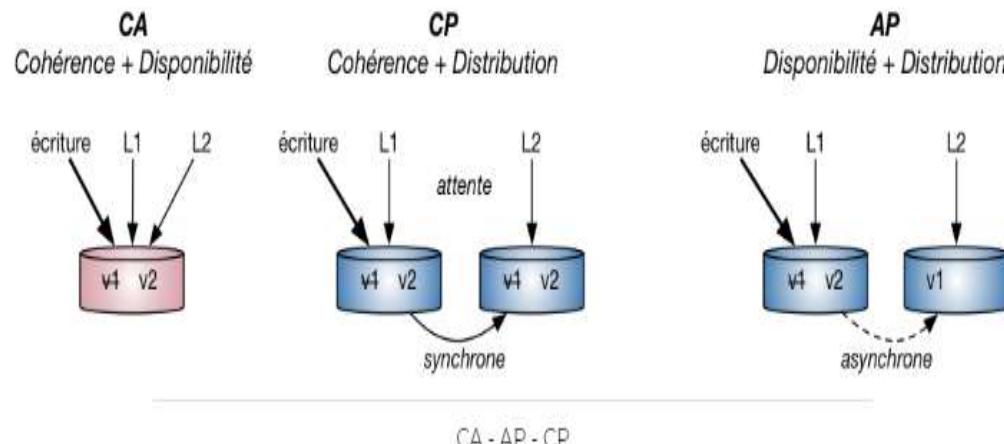
- **Support de SQL (DDL, DML)** : NON, Read-only SQL queries via the MongoDB Connector for BI
- **Support d'indexes secondaires** : OUI
- **Langage de développement du sgbd nosql** : C++
- **Support / Pérénité (communauté , etc....)** : MongoDB Inc., Communauté
- **License** : Open Source
- **API Supportées :**
 - Actionscript, C, C#, C++, Clojure, ColdFusion, D,
 - Dart, Delphi, Erlang, Go, Groovy, Haskell,
 - Java, JavaScript, Lisp, Lua, MatLab, Perl, PHP,
 - PowerShell, Prolog, Python, R, Ruby, Scala, Smalltalk

Module M4.4, section 1 : Objectif et carte de visite

MongoDB

➤ Théorème CAP (CP, AP, AC)

- MongoDB supporte : CP
- CP : Propose de distribuer les données sur plusieurs serveurs en garantissant la tolérance aux pannes (réplication). Exemple : MongoDB, etc. Attention il y a des attentes. L2 doit attendre que V2 soit r

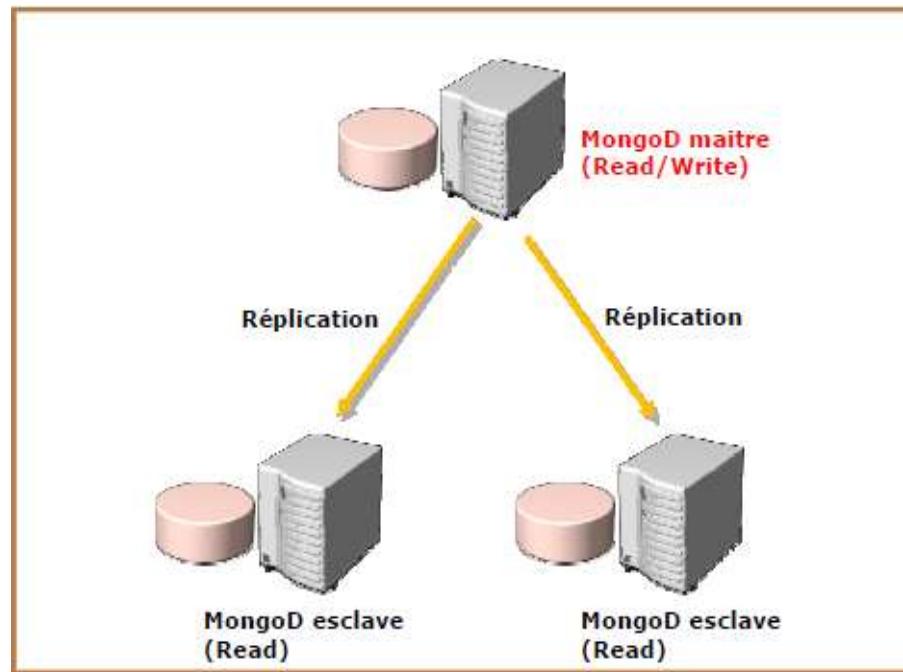


Module M4.4, section 1 : Objectif et carte de visite MongoDB

- Méthode de partitionnement : SHARDING
- Une clé de Sharding permet de repartir les données des collections sur plusieurs SHARD

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Méthode de réPLICATION : Master-slave replication

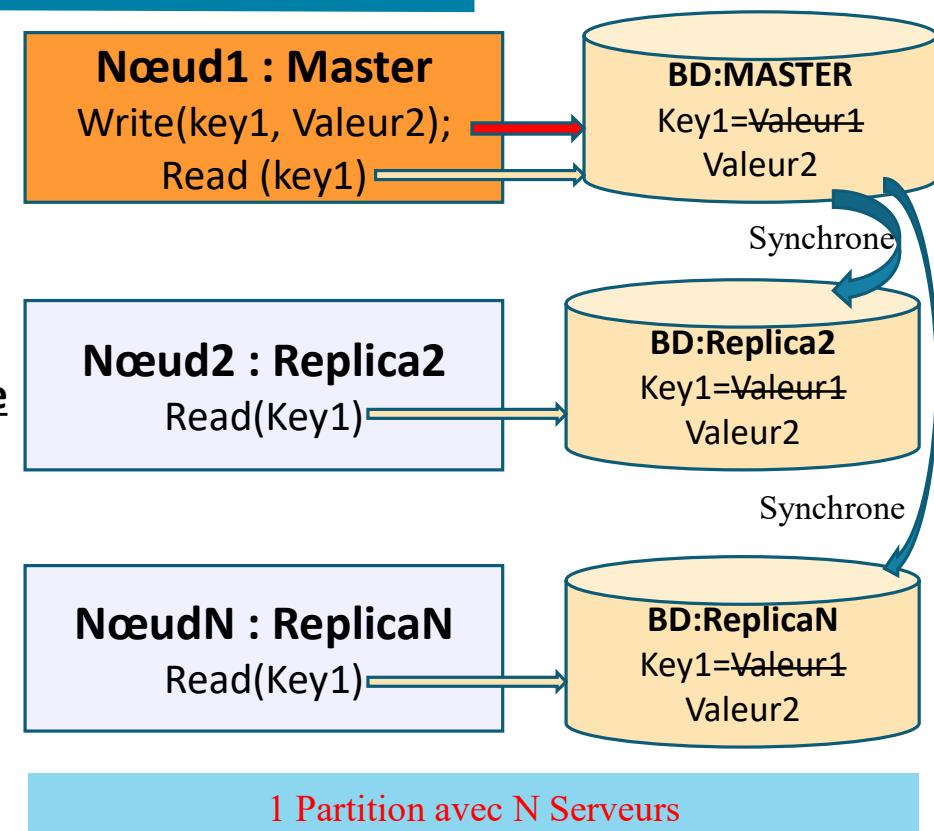


Module M4.4, section 1 : Objectif et carte de visite

MongoDB

➤ Concept de consistance

- CP : **Consistency and Partition (Cohérence et Distribution)**: Permet de **lire la même chose** sur **l'ensemble des nœuds** d'une partition
 - Nœud1 MASTER : READ/**WRITE :réPLICATION SYNCHRONE**
 - Nœud2, ..., NœudN Replicas: READ ONLY
 - Lecture
 - Nœud1, Nœud2 et NœudN lisent la nouvelle version de la valeur **Valeur2** associée à **Key1 (réPLICATION EN MODE SYNCHRONE)**.
 - Exemple : MongoDB, HBASE, BIGTABLE



Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ La durabilité dans MongoDB

- Lorsqu'une opération d'écriture est effectuée, elle est d'abord **enregistrée dans un journal de transactions**
- Le journal de transactions **est écrit sur disque pour garantir la durabilité des données.**
- Si un incident se produit, **MongoDB utilise le journal de transactions pour restaurer les données jusqu'au point le plus récent où elles ont été enregistrées**
- Les données sont ensuite mises à jour **dans la base de données pour refléter les opérations en attente**

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ La durabilité dans MongoDB

■ Recouvrement

- Avec la journalisation, le processus de recouvrement se déroule comme suit:
 1. Cherche dans les fichiers de données l'identifiant du **dernier point de sauvegarde** (checkpoint)
 2. Recherche dans les fichiers journaux l'enregistrement qui correspond à l'identifiant du dernier point de sauvegarde
 3. Applique les opérations trouvées dans les fichiers journaux depuis le dernier point de sauvegarde

Module M4.4, section 1 : Objectif et carte de visite MongoDB

- Clés étrangères : NON mais gestion par l'utilisateur pour lier des documents
- Support de références (REF) : OUI

Module M4.4, section 1 : Objectif et carte de visite MongoDB

- Licences et prix : OpenSource, N/A
- Différents types de versions (communautaire, entreprise, ...): Les deux
- Audience dans le marché : Le plus populaires des moteur NoSQL

Rank	Mar 2019	Feb 2019	Mar 2018	DBMS	Database Model
1.	1.	1.	1.	Oracle +	Relational, Multi-model ↗
2.	2.	2.	2.	MySQL +	Relational, Multi-model ↗
3.	3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ↗
4.	4.	4.	4.	PostgreSQL +	Relational, Multi-model ↗
5.	5.	5.	5.	MongoDB +	Document
6.	6.	6.	6.	IBM Db2 +	Relational, Multi-model ↗
7.	↑ 9.	9.	7.	Microsoft Access	Relational
8.	↓ 7.	7.	8.	Redis +	Key-value, Multi-model ↗
9.	↓ 8.	8.	9.	Elasticsearch +	Search engine, Multi-model ↗
10.	↑ 11.	10.	↑ 11.	SQLite +	Relational
11.	11.	↑ 10.	10.	Cassandra +	Wide column
12.	12.	↑ 15.	15.	MariaDB +	Relational, Multi-model ↗
13.	13.	13.	13.	Splunk	Search engine
14.	14.	↓ 12.	12.	Teradata +	Relational
15.	15.	↑ 18.	18.	Hive +	Relational
...	Search engine

Module M4.4, section 1 : Objectif et carte de visite MongoDB

- **Tables et tables filles** : Pas de tables
- **Clé avec major et minor key** : Pas vu

Module M4.4, section 1 : Objectif et carte de visite MongoDB

- **Gestion des utilisateurs** : OUI, par défaut non, il est possible d'en créer
- **Gestion des droits** : OUI, uniquement si on a créé des utilisateurs :
`db.createUser(userDocument)`
- **Gestion des namespaces ou databases** : OUI, il est possible de créer des bases de données et des collections

Module M4.4, section 1 : Objectif et carte de visite MongoDB

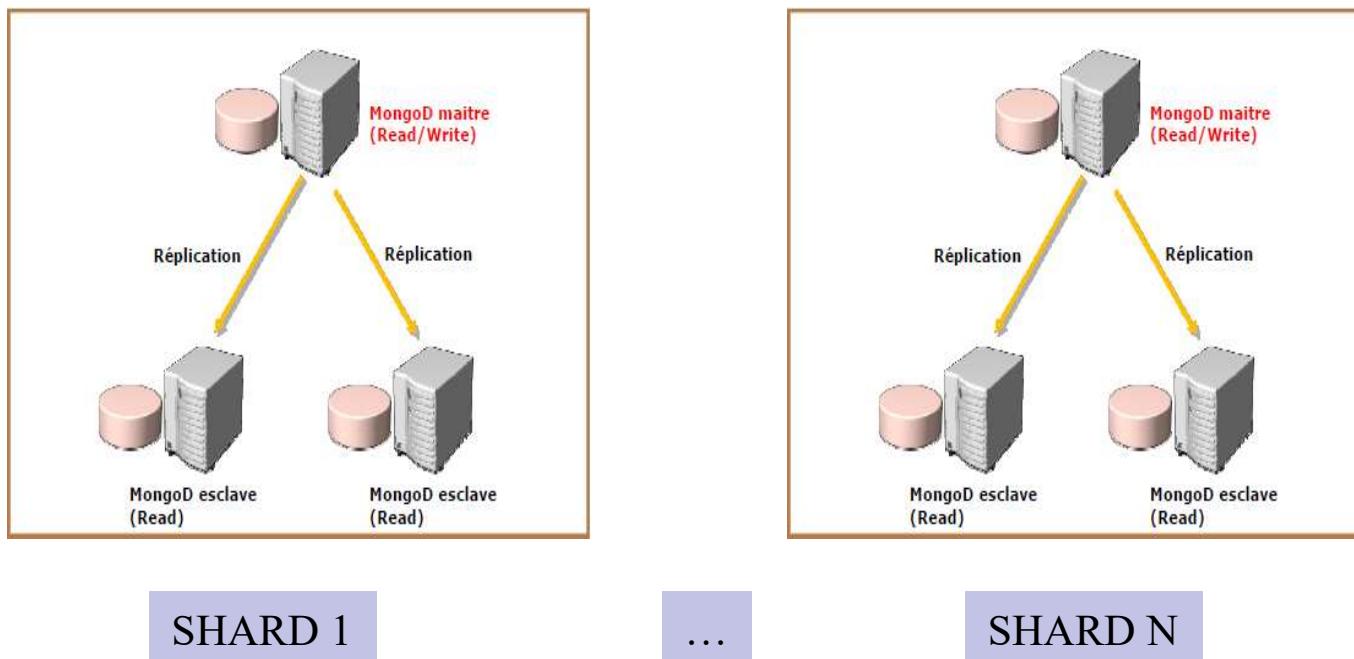
- **Systèmes d'exploitations supportés** : Linux, Windows et MacOS
- **Disponible en mode Dbaas** : OUI, disponible en mode cloud computing

Module M4.4, section 1 : Objectif et carte de visite MongoDB

- Applications communautaires l'utilisant
- Domaines d'applications

Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Architecture distribuée / répliquée (SYNCHRONE, MONGODB)



Module M4.4, section 1 : Objectif et carte de visite MongoDB

➤ Architecture du moteur NoSql :

- Organisé en SHARD
- Une SHARD contient un serveur maître et des serveurs esclave
- Une instance MongoDB comprend 1 ou plusieurs Bases de Données

- Une base de données comprends 0, 1 ou plusieurs collections
- Une collections contient 0,1 ou plusieurs documents

➤ Montée en charge : Possibilité d'ajout de nœud

➤ Gestion de la disponibilité : en cas de panne du maître, élection d'un nouveau maître parmi les nœud esclave

Module M4.4, section 1 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

Module M4.4, section 2 : Installation, outils architecture MongoDB

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Plan

- Installation community edition sous Windows
- Installation community edition sous Linux Ubuntu / Debian
- Installation community edition sous macOS
- Installation des utilitaires MongoDB
- Installation du shell mongoDB sous windows
- Les outils et utilitaires MongoDB
- Lancement du shell mongoDB sous windows
- Lancement des utilitaires d'import/export
- Architecture d'une Instance MongoDB

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community édition sous Windows

Etape 1 : Télécharger l'archive à partir de <https://www.mongodb.com/download-center/community>

Etape 2 : Décompresser l'archive

Etape 3 : Création de l'arborescence de stockage de l'instance par défaut

Etape 4 : Configuration du path

Etape 5 : Création du service d'arrêt/démarrage de Windows

Etape 6 : Démarrage le serveur MongoDB

Etape 7: Lancer le client shell mongosh

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

■ Pour les deux distributions

1) Importer la clé publique utilisée par le système de gestion des paquets

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
9DA31620334BD75D9DCB49F368818C72E52529D4
```

2) Créer un fichier "mongodb-org-4.0.list" pour MongoDB dans le dossier /etc/apt/sources.list.d/

```
$ cd /etc/apt/sources.list.d/  
$ touch mongodb-org-4.0.list
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

■ Pour UBUNTU

➤ Pour Ubuntu 18.04 (Bionic) exécuter ce qui suit :

```
$ echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
```

- Pour Ubuntu 16.04 (Xenial) exécuter ce qui suit :

```
$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
```

- Pour Ubuntu 14.04 (Trusty) exécuter ce qui suit :

```
$ echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

■ Pour DEBIAN

- Pour Debian 8 (Jessie) exécuter ce qui suit :

```
$ echo "deb http://repo.mongodb.org/apt/debian jessie/mongodb-org/4.0 main" | sudo tee  
/etc/apt/sources.list.d/mongodb-org-4.0.list
```
- Pour Debian 9 (Stretch) exécuter ce qui suit :

```
$ echo "deb http://repo.mongodb.org/apt/debian stretch/mongodb-org/4.0 main" | sudo tee  
/etc/apt/sources.list.d/mongodb-org-4.0.list
```
- Remarque: Actuellement, des paquets sont disponibles pour Debian 8 "Jessie" et Debian 9 "Stretch".

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

- Pour les deux distributions

- 1) Recharger la base de données des paquets locaux

```
$ sudo apt-get update
```

- 2) Installer les paquets MongoDB. Pour installer la dernière version stable, exécutez ce qui suit:

```
$ sudo apt-get install -y mongodb-org
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

■ Répertoire mongoDB et localisation de certains fichiers

- Repertoire de mongoDb

Si vous l'avez installé via le gestionnaire de paquets, le répertoire de données /var/lib/mongodb et le répertoire log /var/log/mongodb sont créés pendant l'installation.

- Fichier de configuration

Le package officiel de MongoDB comprend un fichier de configuration (/etc/mongod.conf)

Ces paramètres (tels que les spécifications du répertoire de données et du répertoire de journaux) prennent effet au démarrage.

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

■ Démarrage, Vérification, Arrêt de MongoDB

1) Démarrer le serveur MongoDB

\$ sudo service mongod start

2) Vérifier que le serveur MongoDB a bien démarré avec succès

Vérifiez que le processus mongod a bien démarré en vérifiant le contenu du fichier journal dans /var/log/mongodb/mongodod.log pour une lecture de ligne:
[initandlisten] waiting for connections on port 27017

3) Arrêter le serveur MongoDB

\$ sudo service mongod stop

4) Redémarrer le serveur MongoDB

\$ sudo service mongod restart

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous Linux Ubuntu / Debian

- Lancer le shell mongoDB

```
$ mongosh
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous macOS

- MongoDB ne supporte que les versions MacOS 10.11 et ultérieures sur Intel x86-64.
- **Prérequis:** Si vous n'avez pas encore ajouté la source référentiel du dépôt de formules officielles de mongoDB, ajouter la liste des formules à partir d'un terminal, en tapant ce qui suit :
`$ brew tap mongodb/brew`

➤ Installer ensuite MongoDB

```
$ brew install mongodb-community@4.0
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation community edition sous macOS

- Démarrer le serveur MongoDB

- 1) Démarrer MongoDB en mode manuel

```
$ mongod --config /usr/local/etc/mongod.conf
```

- 2) Démarrer MongoDB en mode automatique

```
$ brew services start mongodb-community@4.0
```

- Lancer un shell MongoDB

```
$ mongo
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation des utilitaires MongoDB

- Télécharger les utilitaires pour votre OS en suivant le lien

https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools

Vous obtenez

mongodb-database-tools-windows-x86_64-100.6.1.zip ou équivalent

- Dezipper le fichier téléchargé

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation des utilitaires MongoDB

- Dans le dossier bin se trouve les utilitaires suivants

mongofiles.exe

mongotop.exe

mongostat.exe

mongoexport.exe

mongoimport.exe

mongorestore.exe

mongodump.exe

bsondump.exe

- Déplacer ces fichiers dans le dossier

C:\Program Files\MongoDB\Server\6.0\bin

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation du shell mongoDB sous windows

- Télécharger le mongoShell en suivant ce lien

https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools

- mongosh-1.6.1-win32-x64.zip
- Dézipper le fichier téléchargé

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Installation du shell mongoDB sous windows

- Dans le dossier bin se trouve les utilitaires suivants

Mongosh.exe

- Déplacer ces fichiers dans le dossier
C:\Program Files\MongoDB\Server\6.0\bin

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Les outils et utilitaires MongoDB

- mongod (le moteur de base)
- mongosh (le shell javascript)
- mongos (le contrôleur de Sharding)
- Les outils d'import/export et de sauvegarde: mongoexport.exe, mongoimport.exe, mongorestore.exe, mongodump.exe, bsondump.exe
- mongofiles (l'utilitaire GridFS)
- mongostat (visualisation des stats d'une instance mongoDB)
- mongosniff (le tcpdump de mongo)
- mongotop, mongoperf

Note : Sous windows, vous devez modifier la variable d'environnement PATH (variables systèmes) en ajoutant le chemin vers les exécutables MongoDB
C:\Program Files\MongoDB\Server\6.0\bin

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Lancement du shell mongoDB sous windows

1. S'assurer que le server MongoDB a été lancé

2. Ouvrir un CMD

Exécuter ensuite MONGO

C:\>mongosh

>

Note : Sous Linux ou MacOS on lance la même commande.

Voir aussi le chapitre installation Linux et MacOS

```
C:\Users\mondi>mongosh
Current Mongosh Log ID: 63aa34fea8b29e247c444b74
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=5000
Using MongoDB:      6.0.2
Using Mongosh:      1.6.0
```

For mongosh info see: <https://docs.mongodb.com/mongodb-shell/>

```
The server generated these startup warnings when booting
2022-12-23T09:28:03.987+01:00: Access control is not enabled for the database. Read and write a
llegation is unrestricted
```

```
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
```

```
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
```

```
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
```

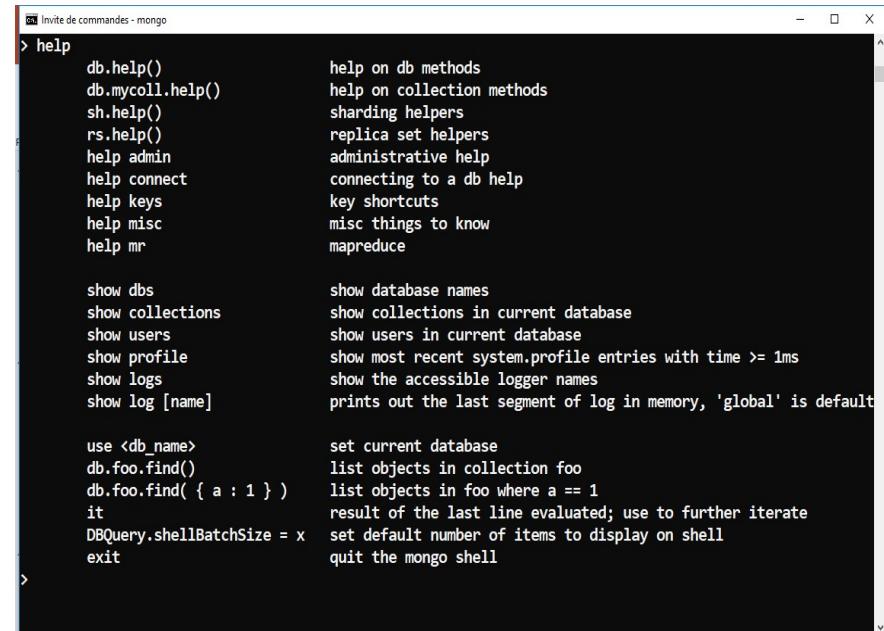
```
Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> |
```

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Lancement du shell mongoDB sous windows

- Connaitre les commandes disponibles

> help



The screenshot shows a terminal window titled "Invite de commandes - mongo". The user has entered the command "> help" and the shell has displayed a list of available commands and their descriptions. The list includes:

- db.help() help on db methods
- db.mycoll.help() help on collection methods
- sh.help() sharding helpers
- rs.help() replica set helpers
- help admin administrative help
- help connect connecting to a db help
- help keys key shortcuts
- help misc misc things to know
- help mr mapreduce
- show dbs show database names
- show collections show collections in current database
- show users show users in current database
- show profile show most recent system.profile entries with time >= 1ms
- show logs show the accessible logger names
- show log [name] prints out the last segment of log in memory, 'global' is default
- use <db_name> set current database
- db.foo.find() list objects in collection foo
- db.foo.find({ a : 1 }) list objects in foo where a == 1
- it result of the last line evaluated; use to further iterate
- DBQuery.shellBatchSize = x set default number of items to display on shell
- exit quit the mongo shell

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Lancement des utilitaires d'import/export

```
mongoimport --port=27017 --host=localhost --db= nomBase --collection=nomCollection --  
file=D:\TPMONGO\2Exercices\0_5Json_collection_Import_Vols_Airbase.json –jsonArray  
ou
```

```
mongoimport --port=27017 --host=localhost --db=nomBase --collection= nomCollection --  
file=nomFichier.json –jsonArray
```

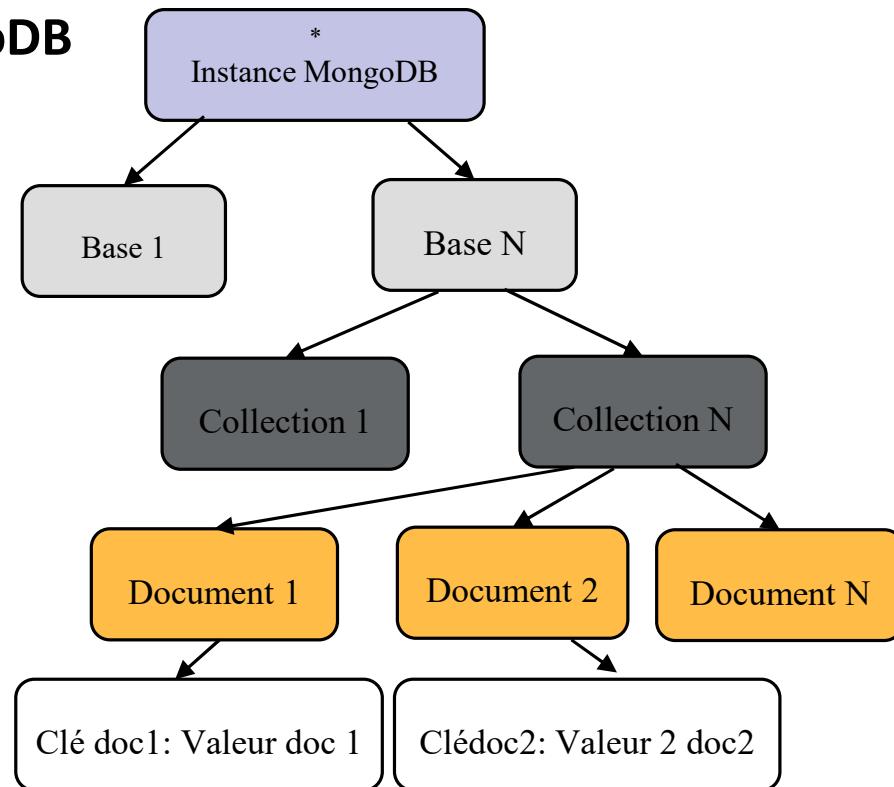
```
C:\> mongoexport –port=NrPort –host nomHost --db=nomBase --collection=nomCollection --  
out=nomFichier.json –query {field:value, ...}
```

Notes :

- Plusieurs autres paramètres sont disponibles. Voir la documentation en ligne
- Port par défaut : 27017

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Architecture d'une Instance MongoDB



Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Architecture d'une Instance MongoDB

■ Caractéristiques d'une instance

- Un port d'écoute (par défaut 27017)
- Un processus serveur
- Un répertoire racine de stockage
- Un fichier de log
- Un fichier de configuration: mongod.conf

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Architecture d'une Instance MongoDB

■ Arrêt / démarrage d'une instance sous Linux

- **Démarrage**

/etc/init.d/mongod start (démarrage de mongoDB via script)
mongod --dbpath <racine_instance> (démarrage de mongoDB)
mongod -f <fichier.conf>

- **Arrêt**

/etc/init.d/mongod stop
/etc/init.d/mongod restart (arrêt/relance)
Commande db.shutdownServer() à partir du client mongo et la base admin

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Architecture d'une Instance MongoDB

- Arrêt / démarrage d'une instance sous Windows. Via les services Windows ou en mode ligne de commandes

- Via les services Windows

Services (local)			
Nom	Description	État	
MessagingService_a691e	Service prenant en charge les envois de S...		
Mettre à jour le service Orchestrator	Gère les mises à jour Windows. Si cette fo...	En cours d'exécution	
Micro Star SCM		En cours d'exécution	
Microsoft App-V Client	Manages App-V users and virtual applicat...		
Microsoft Edge Elevation Service (MicrosoftEdgeElevation...	Keeps Microsoft Edge up to update. If thi...		
Microsoft Edge Update Service (edgeupdate)	Keeps your Microsoft software up to date...		
Microsoft Edge Update Service (edgeupdateam)	Keeps your Microsoft software up to date...		
Microsoft Office Click-to-Run Service	Gérer la coordination des ressources, le té...	En cours d'exécution	
Microsoft Passport	Assure l'isolation des processus pour les c...		
Mode incorporé	Le service Mode incorporé active les scén...		
Modules de génération de clés IKE et AuthIP	Le service IKEEXT héberge les modules de...		
MongoDB Server (MongoDB)	MongoDB Database Server (MongoDB)	En cours d'exécution	

Module M4.4, section 2 : Installation, outils architecture MongoDB

➤ Architecture d'une Instance MongoDB

- Arrêt / démarrage d'une instance sous Windows. Via les services Windows ou en mode ligne de commandes

- Via le mode ligne de commandes

D:\>net start MongoDB

D:\>net stop MongoDB

```
Administrator : Invite de commandes
Microsoft Windows [version 10.0.17134.706]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\Windows\system32>d:

D:\>net start MongoDB
Le service demandé a déjà été démarré.

Vous obtiendrez une aide supplémentaire en entrant NET HELPMSG 2182.

D:\>net stop MongoDB
Le service MongoDB Server s'arrête.
Le service MongoDB Server a été arrêté.

D:\>
D:\>net start MongoDB
Le service MongoDB Server démarre..
Le service MongoDB Server a démarré.

D:\>
```

Module M4.4, section 2 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

➤ Plan

- Gestion des Bases de données MongoDB: Lister les bases de données existantes
- Gestion des Bases de données MongoDB: Switcher sur une base de données existantes
- Gestion des Bases de données MongoDB: Identifier la base courante
- Gestion des Bases de données MongoDB: Créer une nouvelle base de données
- Gestion des Bases de données MongoDB: Supprimer une base de données
- Gestion des Collections MongoDB: Définition
- Gestion des Collections MongoDB: Créer une collection dans une base
- Gestion des Collections MongoDB: Visualiser des collections dans une base
- Gestion des Collections MongoDB: Supprimer une collection dans une base
- La fonction load de mongoDB

Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

➤ Gestion des Bases de données MongoDB: Lister les bases de données existantes

> show dbs

```
>
> show dbs
2019-04-21T16:28:55.751+0200 I NETWORK  [js] trying reconnect to 127.0.0.1:27017 failed
2019-04-21T16:28:55.752+0200 I NETWORK  [js] reconnect 127.0.0.1:27017 ok
admin    0.000GB
airbase   0.000GB
config    0.000GB
local     0.000GB
test      0.000GB
>
```

Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

➤ Gestion des Bases de données MongoDB: Switcher sur une base de données existantes

>use <NomBase>
>use airbase

```
> show dbs
2019-04-21T16:28:55.751+0200 I NETWORK [js] trying reconnect to 127.0.0.1:27017 failed
2019-04-21T16:28:55.752+0200 I NETWORK [js] reconnect 127.0.0.1:27017 ok
admin    0.000GB
airbase   0.000GB
config    0.000GB
local     0.000GB
test      0.000GB
>
>
>
> use airbase
switched to db airbase
>
```

Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

➤ Gestion des Bases de données MongoDB: Identifier la base courante

- La commande à lancer est : db

airbase>db

- Attention à partir de la version 4.5 le **nom de la base est affichée par défaut**

Note:

Le **prompt** est paramétrable

```
> show dbs
2019-04-21T16:28:55.751+0200 I NETWORK [js] trying reconnect to 127.0.0.1:27017 failed
2019-04-21T16:28:55.752+0200 I NETWORK [js] reconnect 127.0.0.1:27017 ok
admin      0.000GB
airbase    0.000GB
config     0.000GB
local      0.000GB
test       0.000GB
>
>
>
> use airbase
switched to db airbase
>
>
> db
airbase
>
```

Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

➤ Gestion des Bases de données MongoDB: Créer une nouvelle bases de données

- Crédit implicite (il faut au moins y ajouter une collection afin de la créer définitivement)

- ```
>use airbase2
>show dbs
Ou >show databases
>db.createCollection('PILOTE');
```

```
Sélection Invité de commandes - mongo
> show dbs
admin 0.000GB
airbase 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>
> use airbase2
switched to db airbase2
>
> show dbs
admin 0.000GB
airbase 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>
> db.createCollection('PILOTE');
{ "ok" : 1 }
>
>
> show dbs
admin 0.000GB
airbase 0.000GB
airbase2 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Bases de données MongoDB: Créer une nouvelle bases de données

- Crédit implicite lors du lancement du client ligne de commande mongo

```
C:\> mongo airbase3
```

```
-- il faut moins une collection pour qu'elle persiste
```

```
>dbs
>show dbs
>db.createCollection('PILOTE3');
>show dbs
```

```
> db
airbase3
> show dbs
admin 0.000GB
airbase 0.000GB
airbase2 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
> db.createCollection('PILOTE3');
{ "ok" : 1 }
> show dbs
admin 0.000GB
airbase 0.000GB
airbase2 0.000GB
airbase3 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Bases de données MongoDB: Supprimer une base de données

- Approche 1 pour supprimer une base

C:\> mongo

```
> use nomBase
> use airbase3
> db.dropDatabase()
```

```
> show dbs
admin 0.000GB
airbase 0.000GB
airbase2 0.000GB
airbase3 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>
>
> db.dropDatabase()
{ "dropped" : "airbase3", "ok" : 1 }
> db
airbase3
> show dbs
admin 0.000GB
airbase 0.000GB
airbase2 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Bases de données MongoDB: Supprimer une base de données

- Approche 2 pour supprimer une base

C:\> mongo

```
> use nomBase
> use airbase2
> db.runCommand({ dropDatabase: 1 })
```

```
> show dbs
admin 0.000GB
airbase 0.000GB
airbase2 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
> use airbase2
switched to db airbase2
> db.runCommand({ dropDatabase: 1 })
{ "dropped" : "airbase2", "ok" : 1 }
> show dbs
admin 0.000GB
airbase 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Collections MongoDB: Définition

- Une collection est un container de documents
- **Une collection en mongoDB équivaut à peu près à une TABLE dans une BD relationnelle**
- Il existe deux approches de création de collection en MongoDB : IMPLICITE ou EXPLICITE

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Collections MongoDB: Créer une collection dans une base

- Crédit implicite d'une collection lors de l'insertion d'un document

Syntaxe :

```
db.<collectionName>.insert()
```

```
C:\> mongo
>use airbase
>db.pilote.insert({plnum:2, plnom: "Milou" , adr:"Nice" });
-- Une collection appelé pilote est créée
```

```
> use airbase
switched to db airbase
> db.pilote.insert({plnum:2, plnom:"Milou" , adr:"Nice" });
WriteResult({ "nInserted" : 1 })
> db.getCollectionNames();
["pilote"]
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Collections MongoDB: Créer une collection dans une base

- Crédit **explicite** d'une collection

Syntaxe :

```
db.createCollection(nomCollection);
```

```
C:\> mongo
>use airbase
>db.createCollection("avion");
```

-- Une collection appelé avion est créée

```
> use airbase
switched to db airbase
> db.pilote.insert({plnum:2, plnom:"Milou" , adr:"Nice" });
WriteResult({ "nInserted" : 1 })
> db.getCollectionNames();
["pilote"]
> db.createCollection("avion");
{ "ok" : 1 }
> db.getCollectionNames();
["avion", "pilote"]
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Collections MongoDB: Visualiser des collections dans une base

- A travers la commande : `show collections`  
>`show collections`
- Ou l'appel de la fonction `db.getCollectionNames`  
>`db.getCollectionNames`

```
>
> show collections
avion
pilote
>
>
> db.getCollectionNames();
["avion", "pilote"]
>
>
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ Gestion des Collections MongoDB: Supprimer une collection dans une base

- La commande de suppression d'une collection est : db.<collectionName>.drop()

```
C:\> mongo
>use airbase
>show collections
>db.avion.drop();
>show collections
```

```
>
> show collections
avion
pilote
> db.avion.drop()
true
> show collections
pilote
>
```

## Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB

### ➤ La fonction load de mongoDB

- **Syntaxe**

> **load("chemin/nomFichier.js");**

Cette fonction permet d'exécuter les commandes javaScripts MongoDB contenues dans le fichier **nomFichier.js**

- **NOTES :** Les commandes dans le fichier peuvent être des commandes de manipulation

- Des bases de données,
- Des collections
- Des documents
- ...

## Module M4.4, section 3 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

## Module M4.4, section 4 : Gestion des documents MongoDB

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Plan

- Définition
- Les formats JSON/BSON
- Approche de modélisation : Imbrication ou partage des objets
- Schéma exemple
- Validation du schéma (contraintes)
- Insertion de documents
- Modification de documents
- Suppression de documents
- Ajout d'un champ dans un document
- Suppression d'un champ dans un document
- Renommage d'un champ dans un document

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Définition

- Les **documents** sont des objets **au format Json**
- Ils sont **comparables aux enregistrements d'une table** dans une base de données relationnelle.
- Tout document **appartient à une collection** et une seule
- Un champ **appelé \_id permet d'identifier de façon unique un document** dans une base de données MongoDB
- MongoDB enregistre les documents sur le disque sous **un format BSON (JSON binaire)**.

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Les formats JSON/BSON

- **JSON (JavaScript Object Notation)**

- Permet de **représenter** de l'information **semi-structurée**
- **Format de données textuelle**
- Utilise une notation JavaScript
- Décrit par la RFC 4627
- **Un document JSON, ne comprend que deux éléments structurels :**
  - Des ensembles de paires nom / valeur ;
  - Des listes ordonnées de valeurs.

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Les formats JSON/BSON

- BSON est la représentation binaire des objets JSON
  - Ajoute des améliorations et des capacités supplémentaires
    - ✓ expressions régulières
    - ✓ Bytes Arrays
    - ✓ Date et TimeStamps
    - ✓ Stockage de blocs de code ou de fonctions JavaScript
  - LE FORMAT JSON d'origine ne prend pas en charge les données binaires
  - Données binaires (plus petite taille) : **Stockage COMPACT donc RAPIDE d'Accès**

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Les formats JSON/BSON

#### ■ Avantage du format JSON

- Format abstrait pour une **représentation simplifiée** dans les différents langages
- **Indépendant du langage de programmation**
- **Simple et complet pour la représentation des objets**
- **Utilise des types de données connus** et simples à décrire
- Une bonne **lisibilité de la syntaxe**
- Temps de traitement très proche de celui des fichiers XML
- Moins volumineux en terme de stockage

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Les formats JSON/BSON

#### ▪ Types disponibles en JSON (voir aussi json.org)

- string
- number
- Object (un JSON)
- array
- true
- false
- Null

#### ➤ Types supplémentaires BSON

- Bytes Arrays
- Date
- TimeStamps

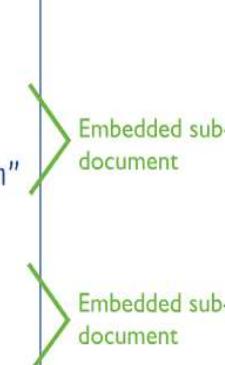
## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Approche de modélisation : Imbrication ou partage des objets

#### ■ Imbrication

- **permet d'emboiter des documents.** Lorsqu'il n'y a pas de risque de redondance, il s'agit d'une approche très efficace qui permet de charger d'un seul document et ses sous-documents

```
{
 _id: <ObjectId1>,
 username: "123xyz",
 contact: {
 phone: "123-456-7890",
 email: "xyz@example.com"
 },
 access: {
 level: 5,
 group: "dev"
 }
}
```

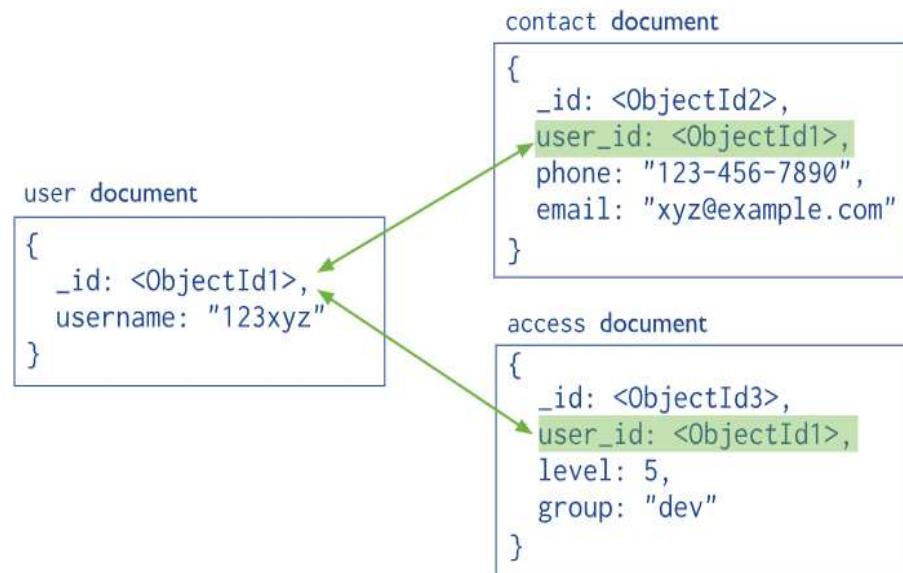


## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Approche de modélisation : Imbrication ou partage des objets

#### ■ Partage des objets

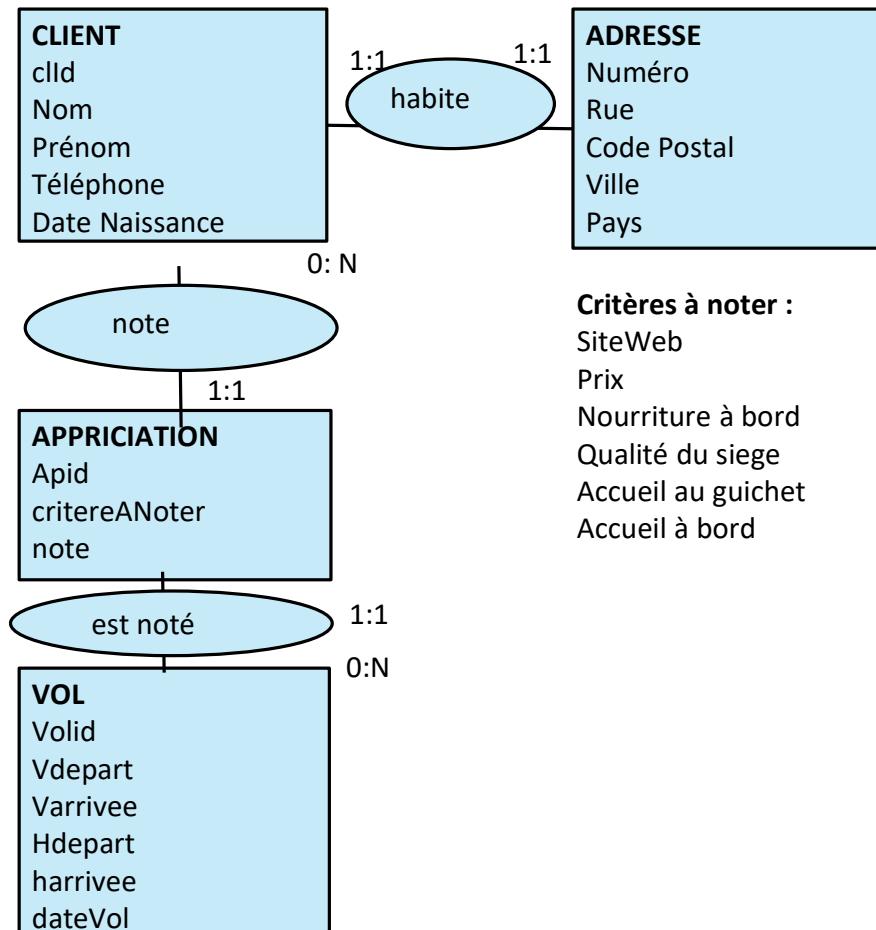
- Très utile s'il y a risque de redondance non souhaité. Plus couteux dans le traitements



## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Schéma exemple

#### ▪ Schéma Merise



**Critères à noter :**

- SiteWeb
- Prix
- Nourriture à bord
- Qualité du siège
- Accueil au guichet
- Accueil à bord

**Notes :**

- EXCELLENT
- TRES\_BIEN
- BIEN
- MOYEN
- PASSABLE
- MEDIOCRE

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Schéma exemple

#### ▪ Schéma JSON/BSON

- Document Client

- ✓ Client 1 : ERZULIE
- ✓ Vue qu'il y a un lien 1:1 entre Adresse et Client, Adresse devient un sous-document

```
{
 _id: 1,
 nom: "ERZULIE",
 prenom:["Maria", "Frida"],
 telephone:"00509232472",
 DateNaiss:"01/01/1880",
 adresse: {
 numero: 11,
 rue:"Rue des miracles",
 codePostal: "HT8110",
 ville: "PORT-AU-PRINCE",
 pays: "HAITI"
 }
}
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Schéma exemple

#### ■ Schéma JSON/BSON

- Document Client

- ✓ Client 2 : BARON
- ✓ Vue qu'il y a un lien 1:1 entre Adresse et Client, Adresse devient un sous-document

```
{
 _id: 2,
 nom: "BARON",
 prenom: ["Samedi"],
 telephone: "00509232488",
 DateNaiss: "01/01/1870",
 adresse: {
 numero: 1,
 rue: "Rue des lwa",
 codePostal: "HT8111",
 ville: "PORT-AU-PRINCE",
 pays: "HAITI"
 }
}
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Schéma exemple

- Schéma JSON/BSON

- Document VOL (avec `_id` explicite)

Y inclure un Array contenant les appréciations

Les appréciations d'un vol sont attachées à lui et lui seul

Les appréciations sont regroupés par clients

A chaque client il y a l'Array de ces appréciations

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Schéma exemple

- Schéma JSON/BSON: Document VOL

```
{
 "_id: "100",
 villeDepart: "Nice",
 villeArrivee:"Paris",
 heureDepart:"10:10",
 heureArrivee:"11:30",
 dateVol:"12/12/2018"
 appreciations:
 [
 {idClient:1, notes:[
 {apid:11, critereANoter:"SiteWeb", note:"BIEN"},
 {apid:12, critereANoter:"Prix", note:"BIEN"},
 {apid:13, critereANoter:"Nourriture à bord", note:"BIEN"},
 {apid:14, critereANoter:"Qualité siège", note:"MOYEN"},
 {apid:15, critereANoter:"Accueil guichet", note:"TRES_BIEN"},
 {apid:16, critereANoter:"Accueil à bord", note:"EXCELLENT"}
]},
 {idClient:2, notes:[
 {apid:21, critereANoter:"SiteWeb", note:"TRES_BIEN"},
 {apid:22, critereANoter:"Prix", note:"MEDIocre"},
 {apid:23, critereANoter:"Nourriture à bord", note:"BIEN"},
 {apid:24, critereANoter:"Qualité siège", note:"MOYEN"},
 {apid:25, critereANoter:"Accueil guichet", note:"TRES_BIEN"},
 {apid:26, critereANoter:"Accueil à bord", note:"BIEN"}
]}
]
}
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Validation du schéma (contraintes)

- MongoDB est normalement un SGBD schema less
- On peut ajouter dans une collection les BSON qu'on veut
- Il est maintenant possible d'introduire **des contraintes d'intégrités** à faire respecter sur les schéma BSON d'une collection
  - Donner la liste des champs obligatoires
  - Donner la valeur minimale et maximale
  - Vérifier qu'un champ est d'un type donné
  - Définir un énuméré (liste des valeurs possibles pour un champ)
  - Définir des expressions régulières
  - Interdire l'ajout de nouveaux champs dans un Record

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Validation du schéma (contraintes)

- Exemple

```
db.createCollection("students", {
 validator: [
 {
 $jsonSchema: {
 bsonType: "object",
 required: ["name", "year", "major", "gpa", "address.city", "address.street"]
 properties: {
 name: {
 bsonType: "string",
 description: "must be a string and is required"
 },
 gender: {
 bsonType: "string",
 description: "must be a string and is not required"
 },
 year: {
 bsonType: "int",
 minimum: 2017,
 maximum: 2017,
 exclusiveMaximum: false,
 description: "must be an integer in [2017, 2017] and is required"
 },
 major: {
 enum: ["Math", "English", "Computer Science", "History", null],
 description: "can only be one of the enum values and is required"
 },
 gpa: {
 bsonType: ["double"],
 minimum: 0,
 description: "must be a double and is required"
 },
 "address.city" : {
 bsonType: "string",
 description: "must be a string and is required"
 },
 "address.street" : {
 bsonType: "string",
 description: "must be a string and is required"
 }
 }
 }
 }
]
})
```

## Module M4.4, section 4 : Gestion des documents MongoDB

```
db.myCollection4.drop();
db.createCollection("myCollection4", {
 validator: {
 $jsonSchema: {
 bsonType: "object",
 required: ["a", "b", "c", "d", "e"],
 properties: {
 a: {
 bsonType: "number"
 },
 b: {
 bsonType: "string"
 },
 c: {
 enum: ["abc", "def", "ghi"],
 description:"Liste des valeurs autorisées"
 },
 d: {
 bsonType: "date",
 description: "La date doit être au format ISO : ISODate('1990-03-02T00:00:00Z')"
 },
 e: {
 bsonType: "number",
 minimum:1,
 maximum:1000,
 description: "La date doit être au format ISO : ISODate('1990-03-02T00:00:00Z')"
 }
 }
 }
 });
});
```

### ➤ Validation du schéma (contraintes)

- Exemple, champs obligatoires, valeurs obligatoires

-- Document correcte  
`db.myCollection4.insert({ _id:1, a:1, b:"b", c:"abc", d:ISODate('1990-03-02T00:00:00Z'), e:1 })`

-- Valeur erronée pour le champ : C  
`db.myCollection4.insert({ _id:2, a:2, b:"b", c:"aaa", d:ISODate('1990-03-02T00:00:00Z'), e:1001 })`

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Validation du schéma (contraintes)

- Exemple, interdiction d'ajout de champs

```
-- interdiction d'ajout de champs
db.myCollection5.drop();
db.createCollection("myCollection5", {
 validator: {
 $jsonSchema: {
 bsonType: "object",
 additionalProperties: false,
 required: ["a", "b", "c", "d", "e"],
 properties: {
 a: {bsonType: "number"},
 b: {bsonType: "string"},
 c: {
 enum: ["abc", "def", "ghi"],
 description:"Liste des valeurs autorisées"
 },
 }
 }
 }
});
```

```
d: { bsonType: "date",
 description: "La date doit être au format ISO : ISODate('1990-03-
02T00:00:00Z')"
},
e: {bsonType: "number",
 minimum:1,
 maximum:1000,
 description: "La date doit être au format ISO : ISODate('1990-03-
02T00:00:00Z')"
} }});
});
```

```
-- Erreur : "errmsg" : "Document failed validation"
db.myCollection5.insert(
{_id:1, a:1, b:"b", c:"abc", d:ISODate('1990-03-02T00:00:00Z'), e:1});
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Validation du schéma (contraintes)

#### ▪ Exemple, interdiction d'ajout de champs

```
-- interdiction d'ajout de champs, _id doit être dans la liste des
champs obligatoires
db.myCollection6.drop();
db.createCollection("myCollection6", {
 validator: {
 $jsonSchema: {
 bsonType: "object",
 additionalProperties: false,
 required: ["_id", "a", "b", "c", "d", "e"],
 properties: {
 _id: {bsonType: "number"},
 a: {bsonType: "number"},
 b: {bsonType: "string"},
 c: {enum: ["abc", "def", "ghi"]},
 description:"Liste des valeurs autorisées"
 },
 }
 }
});
```

```
d: {bsonType: "date",
 description: "La date doit être au format ISO : ISODate('1990-03-
02T00:00:00Z')"
},
e: {
 bsonType: "number",
 minimum:1,
 maximum:1000,
 description: "La date doit être au format ISO : ISODate('1990-03-
02T00:00:00Z')"
} }}});
```

```
-- Erreur : "errmsg" : "Document failed validation"
db.myCollection6.insert(
 {_id:1, a:1, b:"b", c:"abc", d:ISODate('1990-03-02T00:00:00Z'), e:1, f:1 }
);
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Insertion de documents

- Trois méthodes permettent d'effectuer des insertions de documents

- db.collection.insertOne(): Insère un document dans la collection
- db.collection.insertMany(): Insère plusieurs documents dans la collection
- db.collection.insert(): Insère indifféremment un ou plusieurs documents dans la collection

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Insertion de documents

- D'autres méthodes peuvent servir à l'insertion de documents. Voir la document pour leur usage

- db.collection.update() when used with the upsert: true option.
- db.collection.updateOne() when used with the upsert: true option.
- db.collection.updateMany() when used with the upsert: true option.
- db.collection.findAndModify() when used with the upsert: true option.
- db.collection.findOneAndUpdate() when used with the upsert: true option.
- db.collection.findOneAndReplace() when used with the upsert: true option.
- db.collection.save().
- db.collection.bulkWrite().

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Insertion de documents

- Insertion des clients du client 1

```
c:\>mongosh
> use airbase
> db.createCollection("clients");
> db.createCollection("vols");
> document={
 _id: 1,
 nom: "ERZULIE",
 prenoms:["Maria", "Freda"],
 telephone:"00509232485",
 DateNaiss:"01/01/1881",
 adresse: {
 numero: 11,
 rue:"Rue des miracles",
 codePostal: "HT8110",
 ville: "PORT-AU-PRINCE",
 pays: "HAITI"
 }
}
>db.clients.insert(document);
WriteResult({ "nInserted" : 1 })
```

La collection clients est aussi créés par la même occasion.

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Insertion de documents

#### ▪ Insertion des clients du client 2

```
c:\>mongo
> use airbase
> db.clients.insertOne(
{
 _id: 2,
 nom: "BARON",
 prenoms:["Samedi"],
 telephone:"00509232488",
 DateNaiss:"01/01/1870",
 adresse: {
 numero: 1,
 rue:"Rue des Iwa",
 codePostal: "HT8111",
 ville: "PORT-AU-PRINCE",
 pays: "HAITI"
 }
});
{ "acknowledged" : true, "insertedId" : 2 }
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Insertion de documents

#### ■ Insertion d'un VOL

```
c:\>mongo
> use airbase
> document =
{
 _id: "100",
 villeDepart: "Nice",
 villeArrivee: "Paris",
 heureDepart: "10:10",
 heureArrivee: "11:30",
 dateVol: "12/12/2018",
 appreciations:[
 {idClient:1,
 notes:[
 {apid: 11, critereANoter: "SiteWeb", note: "BIEN"},
 {apid: 12, critereANoter: "Prix", note: "BIEN"},
 {apid: 13, critereANoter: "Nourriture à bord", note: "BIEN"},
 {apid: 14, critereANoter: "Qualité siège", note: "MOYEN"},
 {apid: 15, critereANoter: "Accueil guichet", note: "TRES_BIEN"},
 {apid: 16, critereANoter: "Accueil à bord", note: "EXCELLENT"}
]
 },
 ...
}
```

... Voir la suite page suivante

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Insertion de documents

- Insertion d'un vol suite

```
{idClient:2,
 notes:[
 {apid: 21, critereANoter: "SiteWeb", note: "TRES_BIEN"},
 {apid: 22, critereANoter: "Prix", note: "MEDIOCRE"},
 {apid: 23, critereANoter: "Nourriture à bord", note: "BIEN"},
 {apid: 24, critereANoter: "Qualité siège", note: "MOYEN"},
 {apid: 25, critereANoter: "Accueil guichet", note: "TRES_BIEN"},
 {apid: 26, critereANoter: "Accueil à bord", note: "BIEN" }
]
}
}

>db.vols.insertOne(document);
{ "acknowledged" : true, "insertedId" : 100 }
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Modification de documents

#### ■ Les méthodes suivantes permettent d'effectuer des mises à jour

- db.collection.updateOne(<filter>, <update>, <options>)
- db.collection.updateMany(<filter>, <update>, <options>)
- db.collection.replaceOne(<filter>, <update>, <options>)

**Filter** : Expression de recherche

**Update** : Modification à faire

**Options** : Divers options (voir la documentation)

```
{
 upsert: <boolean>,
 writeConcern: <document>,
 collation: <document>,
 arrayFilters: [<filterdocument1>, ...]
}
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Modification de documents

- Exemple de modification de documents

```
> db.clients.updateOne({ nom: "ERZULIE" }, { $set: {DateNaiss: "12/12/1900" } },{});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

- Attention il faut respecter la casse sur le nom des champs et des valeurs

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Suppression de documents

- Les méthodes suivantes permettent d'effectuer des suppressions
  - db.collection.deleteMany()
  - db.collection.deleteOne()

```
db.collection.deleteOne(
 <filter>,
 {
 writeConcern: <document>,
 collation: <document>
 }
)
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Suppression de documents

#### ■ Exemple de suppression d'un document

- Supprimer le client ERZULIE

```
> db.clients.deleteOne({nom:"ERZULIE"},{});
```

- Vérification

```
> db.clients.deleteOne({nom:"ERZULIE"},{});
{ "acknowledged" : true, "deletedCount" : 1 }
> db.clients.find();
{ "_id" : 2, "nom" : "BARON", "prenoms" : ["Samedi"], "telephone" : "00509232488", "DateNaiss" : "01/01/1870", "adresse"
: { "numero" : 1, "rue" : "Rue des lwa", "codePostal" : "HT8111", "ville" : "PORT-AU-PRINCE", "pays" : "HAITI" } }
```

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Ajout d'un champ dans un document

- L'opérateur \$set permet d'ajouter un champ dans un ou plusieurs documents. Si le champ existe une mise à jour est effectuée
- { \$set: { <field1>: <value1>, ... } } à combiner avec update
- Exemple
  - Ajouter le champ **voyageurAssidu** dans chaque document de la collection clients
   
 > db.clients.update({}, { \$set: { voyageurAssidu: "OUI" } });
   
**Cette action n'ajoute le champ que dans le premier document**
  - Pour ajouter le champ dans tous les documents d'une collection. Deux possibilités
   
 > db.clients.updateMany({}, { \$set: { voyageurAssidu: "OUI" } });
   
 > db.clients.update({}, { \$set: { voyageurAssidu: "OUI" } }, {**multi:true**});

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Suppression d'un champ dans un document

- L'opérateur **\$unset** permet de supprimer des champs dans un ou plusieurs documents.  
{ \$set: { <field1>: <value1>, ... } } à combiner avec update
- Exemple
  - Supprimer le champ **voyageurAssidu** dans chaque document de la collection clients  
> db.clients.update({}, { \$unset: { voyageurAssidu: 1 } });  
\*\*Cette action ne mofie que le premier document
  - Pour supprimer le champ dans l'ensemble des document. Deux possibilité :  
> db.clients.updateMany({}, { \$unset: { voyageurAssidu: 1 } });  
> db.clients.update({}, { \$unset: { voyageurAssidu: 1 } }, {multi:true});

## Module M4.4, section 4 : Gestion des documents MongoDB

### ➤ Renommage d'un champ dans un document

- L'opérateur **\$rename** permet **de renommer des champs** dans un ou plusieurs documents.  
{ \$rename: { <field1>: <newName1>, ... } } à combiner avec update

- Exemple

- Renomme le champ DateNaiss en dateNaissance **dans un document**

```
> db.clients.update({_id:1}, { $rename: {"DateNaiss": "dateNaissance"} });
```

- Renomme le champ DateNaiss en dateNaissance **dans tous les documents**

```
> db.clients.updateMany({}, { $rename: {"DateNaiss": "dateNaissance"} });
```

```
> db.clients.update({}, {$rename : {"dateNaiss": "dateNaissance"} }, {multi:true});
```

## Module M4.4, section 4 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

# Module M4.4, section 5 : Indexation et recherches de documents MongoDB

## ➤ Plan

- Indexation de documents MongoDB: Création d'index
- Indexation de documents MongoDB: Affichage d'un plan d'exécution d'une requête
- Indexation de documents MongoDB: Affichage des indexes créés sur une collection
- Indexation de documents MongoDB: Suppression d'un index
  
- Recherche de documents MongoDB : Les méthodes pour effectuer les recherches
- Recherche de documents MongoDB : Recherche de tous les documents d'une collection
- Recherche de documents MongoDB : Recherche de documents connaissant la valeur d'une propriété
- Recherche de documents MongoDB : Recherche de documents connaissant la valeur plusieurs propriétés
- Recherche de documents MongoDB : Recherche de documents avec l'opérateur >, >=, <, ...
- Recherche de documents MongoDB : Recherches de documents avec le connecteur logique OR
- Recherche de documents MongoDB : Recherche de documents via une colonne indexée

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Indexation de documents MongoDB : Création d'index

- Les méthodes suivantes permettent de poser des index
  - db.collection.createIndex()
  - db.collection.createIndex( <key and index type specification>, <options> )
- Il existe différents types d'index
  - Des index mono-colonnes
  - Des index multicolonnes
  - Des index ascendants
  - Des index descendants
  - Des index uniques ou non unique
- Il faut avoir une bonne stratégie d'indexation. Elle doit dépendre des requêtes à lancer

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Indexation de documents MongoDB: Création d'index

- <key and index type specification>

- Document contenant la liste des champs à indexer {**champ1:1, champ2:-1**}
  - 1:** Index ascendant; **-1:** Index descendant

- <options>

- **Ce document est optionnel.** Il permet de spécifier entre autre le nom de l'index. Voici les champs possibles de ce document:

- background** : true ou false (si true construit l'index sans bloquer les activités)

- name** : nom index (nom explicite)

- unique** : true ou false (true si index unique)

- **Si aucun nom d'index est donné explicitement, mongoDB donne un nom implicite**

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

➤ Indexation de documents MongoDB: Création d'index

- Crédit d'un index sur la colonne NOM de la collection CLIENTS (**nom index implicite**)

```
>db.clients.createIndex({nom:1});
```

```
airbase> db.clients.createIndex({nom:1});
nom_1
```

# Module M4.4, section 5 : Indexation et recherches de documents MongoDB

## ➤ Indexation de documents MongoDB: Création d'index

- Cr ation d'un index sur la colonne DateNaiss de la collection CLIENTS (nom index explicite)

```
> db.clients.createIndex({DateNaiss :1}, {background:false, unique:false, name:"idx_clients_DateNaiss" });
```

```
airbase> db.clients.createIndex({DateNaiss :1}, {background:false, unique:false, name:"idx_clients_DateNaiss" });
idx_clients_DateNaiss
```

# Module M4.4, section 5 : Indexation et recherches de documents MongoDB

## ➤ Indexation de documents MongoDB: Affichage d'un plan d'exécution d'une requête

- Exemple : db.clients.find({nom:"Leroy"}).explain();

```
airbase> db.clients.find({nom:"Leroy"}).explain();
{
 explainVersion: '1',
 queryPlanner: {
 namespace: 'airbase.clients',
 indexFilterSet: false,
 parsedQuery: { nom: { '$eq': 'Leroy' } },
 queryHash: 'F1CB349A',
 planCacheKey: '1E5DB9DD',
 maxIndexedOrSolutionsReached: false,
 maxIndexedAndSolutionsReached: false,
 maxScansToExplodeReached: false,
 winningPlan: {
 stage: 'FETCH',
 inputStage: {
 stage: 'IXSCAN',
 keyPattern: { nom: 1 },
 indexName: 'nom_1',
 isMultiKey: false,
 multiKeyPaths: { nom: [] },
 isUnique: false,
 isSparse: false,
 isPartial: false,
 indexVersion: 2,
 direction: 'forward',
 indexBounds: { nom: ["[\"Leroy\", \"Leroy\"]"] }
 }
 },
 rejectedPlans: []
 }
},
```

```
command: { find: 'clients', filter: { nom: 'Leroy' }, '$db': 'airbase' }
serverInfo: {
 host: 'PC-GABRIEL',
 port: 27017,
 version: '6.0.2',
 gitVersion: '94fb7dfc8b974f1f5343e7ea394d8d9deedba50e'
},
serverParameters: {
 internalQueryFacetBufferSizeBytes: 104857600,
 internalQueryFacetMaxOutputDocSizeBytes: 104857600,
 internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
 internalDocumentSourceGroupMaxMemoryBytes: 104857600,
 internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
 internalQueryProhibitBlockingMergeOnMongoS: 0,
 internalQueryMaxAddToSetBytes: 104857600,
 internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
},
ok: 1
}
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Indexation de documents MongoDB: Affichage d'un plan d'exécution d'une requête

- Nom index explicite : db.clients.find({DateNaiss:"01/01/1870"}).explain();

```

airbase> db.clients.find({DateNaiss:"01/01/1870"}).explain();
{
 explainVersion: '1',
 queryPlanner: {
 namespace: 'airbase.clients',
 indexFilterSet: false,
 parsedQuery: { DateNaiss: { '$eq': '01/01/1870' } },
 queryHash: 'A1D78EC3',
 planCacheKey: '445DEC89',
 maxIndexedOrSolutionsReached: false,
 maxIndexedAndSolutionsReached: false,
 maxScansToExplodeReached: false,
 winningPlan: {
 stage: 'FETCH',
 inputStage: {
 stage: 'IXSCAN',
 keyPattern: { DateNaiss: 1 },
 indexName: 'idx_clients_DateNaiss',
 isMultiKey: false,
 multiKeyPaths: { DateNaiss: [] },
 isUnique: false,
 isSparse: false,
 isPartial: false,
 indexVersion: 2,
 direction: 'forward',
 indexBounds: { DateNaiss: ["[\"01/01/1870\", \"01/01/1870\"]"] }
 }
 },
 rejectedPlans: []
 },
 command: {
 find: 'clients',
 filter: { DateNaiss: '01/01/1870' },
 '$db': 'airbase'
 },
 serverInfo: {
 host: 'PC-GABRIEL',
 port: 27017,
 version: '6.0.2',
 gitVersion: '94fb7dfc8b974f1f5343e7ea394d0d9deedba50e'
 },
 serverParameters: {
 internalQueryFacetBufferSizeBytes: 104857600,
 internalQueryFacetMaxOutputDocSizeBytes: 104857600,
 internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
 internalDocumentSourceGroupMaxMemoryBytes: 104857600,
 internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
 internalQueryProhibitBlockingMergeOnMongoS: 0,
 internalQueryMaxAddToSetBytes: 104857600,
 internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
 },
 ok: 1
}

```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Indexation de documents MongoDB: Affichage des indexes créés sur une collection

- La méthode `db.collection.getIndexes()` permet d'afficher tous les indexes créés sur une collection

#### ➤ Exemple

```
> use airbase
>db.clients.getIndexes();
```

```
> db.clients.getIndexes();
[{"v": 2, "key": {"_id": 1}, "name": "_id_", "ns": "airbase.clients"}, {"v": 2, "key": {"DateNaiss": 1}, "name": "idx_clients_DateNaiss", "ns": "airbase.clients", "background": false}, {"v": 2, "key": {"nom": 1}, "name": "nom_1", "ns": "airbase.clients"}]
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Indexation de documents MongoDB: Suppression d'un index

- Pour supprimer un index il faut appeler la méthode **db.collection.dropIndex(index)**

- Exemple :

Suppression de l'index nom\_1

```
> db.clients.dropIndex("nom_1");
```

```
airbase> db.clients.dropIndex("nom_1");
{ nIndexesWas: 3, ok: 1 }
airbase> db.clients.getIndexes();
[
 {
 v: 2,
 key: { _id: 1 },
 name: '_id_',
 background: false
 }
]
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Les méthodes pour effectuer les recherches

- Les méthodes suivantes permettent de rechercher des documents :

- **db.collection.find()** : renvoie tous les documents d'une collections
- **db.collection.find(query, projection)** : renvoie les documents qui matchent avec "query" et affiche les colonnes indiquées dans "projection"

✓ Query contient des tests et évaluation:

Opérateur de comparaison : \$eq, \$gt, \$in, \$lt, \$lte, \$ne, \$nin

Opérateur logique : \$and, \$not, \$nor, \$or

Vérification d'éléments : \$exists, \$type

Evaluation d'expressions: \$expr,\$jsonSchema, \$mo, \$regex, \$text, \$where

✓ Projection contient des expressions :

{ field1: <value>, field2: <value> ... }

Si **value=1** champ retourné, si **value =0** non

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Recherche de tous les documents d'une collection

#### ■ Rechercher tous les documents de la collection CLIENTS

```
> db.clients.find();
```

```
> db.clients.find();
{ "_id" : 1, "nom" : "ERZULIE", "prenoms" : ["Maria", "Frida"], "telephone" : "00509232472", "DateNaiss" : "01/01/1880", "adresse" : { "numero" : 11, "rue" : "Rue des miracles", "codePostal" : "HT8110", "ville" : "PORT-AU-PRINCE", "pays" : "HAITI" } }
{ "_id" : 2, "nom" : "BARON", "prenoms" : ["Samedi"], "telephone" : "00509232488", "DateNaiss" : "01/01/1870", "adresse" : { "numero" : 1, "rue" : "Rue des lwa", "codePostal" : "HT8111", "ville" : "PORT-AU-PRINCE", "pays" : "HAITI" } }
>
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Recherche de tous les documents d'une collection

-- Pour un meilleurs affichage

```
> db.clients.find().pretty();
```

```
> db.clients.find().pretty();
{
 "_id" : 1,
 "nom" : "ERZULIE",
 "prenoms" : [
 "Maria",
 "Frida"
],
 "telephone" : "00509232472",
 "DateNaiss" : "01/01/1880",
 "adresse" : {
 "numero" : 11,
 "rue" : "Rue des miracles",
 "codePostal" : "HT8110",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}

{
 "_id" : 2,
 "nom" : "BARON",
 "prenoms" : [
 "Samedi"
],
 "telephone" : "00509232488",
 "DateNaiss" : "01/01/1870",
 "adresse" : {
 "numero" : 1,
 "rue" : "Rue des Iwa",
 "codePostal" : "HT8111",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

➤ Recherche de documents MongoDB :  
Recherche de documents connaissant la  
valeur d'une propriété

- Recherche de clients de nom ERZULIE

```
> db.clients.find({nom:"ERZULIE"}).pretty();
```

```
> db.clients.find({nom:"ERZULIE"}).pretty();
{
 "_id" : 1,
 "nom" : "ERZULIE",
 "prenoms" : [
 "Maria",
 "Frida"
],
 "telephone" : "00509232472",
 "DateNaiss" : "01/01/1880",
 "adresse" : {
 "numero" : 11,
 "rue" : "Rue des miracles",
 "codePostal" : "HT8110",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}
>
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

➤ **Recherche de documents MongoDB :**  
**Recherche de documents connaissant la valeur de plusieurs propriétés**

- Recherche de clients de nom BARON et ayant le telephone suivant :  
 :"00509232488"

```
> db.clients.find({nom:"BARON",
 telephone:"00509232488"}).pretty();
```

```
> db.clients.find({nom:"BARON", telephone:"00509232488"}).pretty();
{
 "_id" : 2,
 "nom" : "BARON",
 "prenoms" : [
 "Samedi"
],
 "telephone" : "00509232488",
 "DateNaiss" : "01/01/1870",
 "adresse" : {
 "numero" : 1,
 "rue" : "Rue des lwa",
 "codePostal" : "HT8111",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}
>
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

➤ Recherche de documents MongoDB :  
Recherche de documents avec l'opérateur >,  
>=, <, ...

- Recherche de clients nés après 1870

```
>db.clients.find({DateNaiss:{"$gt":"01/01/1871"}
}).pretty();
```

```
> db.clients.find({DateNaiss:{"$gt":"01/01/1871"}).pretty()
{
 "_id" : 1,
 "nom" : "ERZULIE",
 "prenoms" : [
 "Maria",
 "Frida"
],
 "telephone" : "00509232472",
 "DateNaiss" : "01/01/1880",
 "adresse" : {
 "numero" : 11,
 "rue" : "Rue des miracles",
 "codePostal" : "HT8110",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}
>
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Recherche de documents avec le connecteur logique OR

- Recherche des clients nées après 1870 ou qui habitent au numéro 11.

```
> db.clients.find({"$or":
 [{DateNaiss:{$gt:"01/01/1871"}}, {"adresse.
 numero":11}]}).pretty();
```

```
> db.clients.find({"$or":
 [{DateNaiss:{$gt:"01/01/1871"}}, {"adresse.
 numero":11}]}).pretty();

{"_id": 1,
 "nom": "ERZULIE",
 "prenoms": [
 "Maria",
 "Frida"
>],
 "telephone": "00509232472",
 "DateNaiss": "01/01/1880",
 "adresse": {
 "numero": 11,
 "rue": "Rue des miracles",
 "codePostal": "HT8110",
 "ville": "PORT-AU-PRINCE",
 "pays": "HAITI"
> }
}
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Recherche de documents avec projection

- Recherche tous les clients et afficher uniquement le champs nom

```
>db.clients.find({}, {nom:1}).pretty();
```

```
> db.clients.find({}, {nom:1}).pretty();
{
 "_id" : 1, "nom" : "ERZULIE"
}
{
 "_id" : 2, "nom" : "BARON"
}
>
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Recherche de documents avec projection

- Recherche tous les clients de nom ERZULIE et afficher tous les champs sauf dateNaiss et telephone
- > db.clients.**find({nom:"ERZULIE"}, {DateNaiss:0, telephone:0})**.pretty();

```
> db.clients.find({nom:"ERZULIE"}, {DateNaiss:0, telephone:0}).pretty();
{
 "_id" : 1,
 "nom" : "ERZULIE",
 "prenoms" : [
 "Maria",
 "Frida"
],
 "adresse" : {
 "numero" : 11,
 "rue" : "Rue des miracles",
 "codePostal" : "HT8110",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}
>
```

## Module M4.4, section 5 : Indexation et recherches de documents MongoDB

### ➤ Recherche de documents MongoDB : Recherche de documents via une colonne indexée

- Recherche tous les clients via l'**index posé sur le champ NOM**

```
> db.clients.find({nom:"ERZULIE"}, {DateNaiss:0, telephone:0}).pretty();
```

```
> db.clients.find({nom:"ERZULIE"}, {DateNaiss:0, telephone:0}).pretty();
{
 "_id" : 1,
 "nom" : "ERZULIE",
 "prenoms" : [
 "Maria",
 "Frida"
],
 "adresse" : {
 "numero" : 11,
 "rue" : "Rue des miracles",
 "codePostal" : "HT8110",
 "ville" : "PORT-AU-PRINCE",
 "pays" : "HAITI"
 }
}
>
```

# Module M4.4, section 5 : Indexation et recherches de documents MongoDB

## ➤ Recherche de documents MongoDB : Recherche de documents via une colonne indexée

### ■ Recherche tous les clients via l'**index** posé sur le champ NOM

```
airbase> db.clients.find({nom: "ERZULIE"}, {DateNaiss:0, telephone:0}).explain();
{
 explainVersion: '1',
 queryPlanner: {
 namespace: 'airbase.clients',
 indexFilterSet: false,
 parsedQuery: { nom: { '$eq': 'ERZULIE' } },
 queryHash: 'F1CB349A',
 planCacheKey: '1E5DB9DD',
 maxIndexedOrSolutionsReached: false,
 maxIndexedAndSolutionsReached: false,
 maxScansToExplodeReached: false,
 winningPlan: {
 stage: 'PROJECTION_DEFAULT',
 transformBy: { DateNaiss: 0, telephone: 0 },
 inputStage: {
 stage: 'FETCH',
 inputStage: {
 stage: 'IXSCAN',
 keyPattern: { nom: 1 },
 indexName: 'nom_1',
 isMultiKey: false,
 multiKeyPaths: { nom: [] },
 isUnique: false,
 isSparse: false,
 isPartial: false,
 indexVersion: 2,
 direction: 'forward',
 indexBounds: { nom: ['[\"ERZULIE\", \"ERZULIE\"]'] }
 }
 },
 rejectedPlans: []
 }
 },
 command: {
 find: 'clients',
 filter: { nom: 'ERZULIE' },
 projection: { DateNaiss: 0, telephone: 0 },
 '$db': 'airbase'
 },
 serverInfo: {
 host: 'PC-GABRIEL',
 port: 27017,
 version: '6.0.2',
 gitVersion: '94fb7dfc8b974f1f5343e7ea394d0d9deedba50e'
 },
 serverParameters: {
 internalQueryFacetBufferSizeBytes: 104857600,
 internalQueryFacetMaxOutputDocSizeBytes: 104857600,
 internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
 internalDocumentSourceGroupMaxMemoryBytes: 104857600,
 internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
 internalQueryProhibitBlockingMergeOnMongoS: 0,
 internalQueryMaxAddToSetBytes: 104857600,
 internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
 },
 ok: 1
}
```

## Module M4.4, section 5 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Plan

- Les différentes étapes d'agrégation
- Pipe des étapes d'agrégation
- Etape d'agrégation \$addFields
- Etape d'agrégation \$bucket
- Etape d'agrégation \$bucketAuto
- Etape d'agrégation \$collStats
- Etape d'agrégation \$count
- Etape d'agrégation \$facet
- Etape d'agrégation \$graphLookup

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Les différentes étapes d'agrégation

- **MongoDB propose plusieurs étapes d'agrégations** : \$addFields, \$bucket, \$bucketAuto, \$collStats, \$count, \$facet, \$graphLookup, \$group, \$indexStats, \$limit, \$listSessions, \$lookup, \$match, \$out, \$project, \$redact, \$replaceRoot, \$sample, \$skip, \$sort, \$sortByCount, \$unwind, ...
- Ces étapes sont des **paramètres de la fonction db.collection.aggregate([{etape 1}, ..., {etape N}])**
- Une **étape** peut être appelée **zéro, une ou plusieurs fois**
- Ces étapes **reçoivent en entrée des documents** et renvoient après leur action des documents
- Nous allons définir certaines de ces étapes par la suite.

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Pipeline des étapes d'agrégation

- La fonction `db.collection.aggregate( [ { <étape> }, ... ] )` permet d'appliquer les opérateurs sur une ou plusieurs collections
- Lors de l'appel de `db.collection.aggregate()`, un tableau des étapes est passé en argument  
    > `db.collection.aggregate( [ { <étape1> }, ..., { <étapeN> } ] )`
- Le tableau contient une ou plusieurs étapes
- Les étapes sont **appliquées dans leur ordre d'apparition** dans le tableau.
- Une étape reçoit en entrée des documents et fournit en sortie des documents à l'étape suivante. **ON APPELLE CELA AUSSI PIPELINE DES ETAPES**
- **Les pipeline des étapes ressemblent au PIPE Unix/Linux**

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Pipeline des étapes d'agrégation

- Exemple de pipeline d'agrégation
  - Considérons les commandes de Pizzas ci-dessous

```
db.orders.insertMany([
 { _id: 0, name: "Pepperoni", size: "small", price: 19, quantity: 10, date: ISODate("2021-03-13T08:14:30Z") },
 { _id: 1, name: "Pepperoni", size: "medium", price: 20, quantity: 20, date: ISODate("2021-03-13T09:13:24Z") },
 { _id: 2, name: "Pepperoni", size: "large", price: 21, quantity: 30, date: ISODate("2021-03-17T09:22:12Z") },
 { _id: 3, name: "Cheese", size: "small", price: 12, quantity: 15, date: ISODate("2021-03-13T11:21:39.736Z") },
 { _id: 4, name: "Cheese", size: "medium", price: 13, quantity: 50, date: ISODate("2022-01-12T21:23:13.331Z") },
 { _id: 5, name: "Cheese", size: "large", price: 14, quantity: 10, date: ISODate("2022-01-12T05:08:13Z") },
 { _id: 6, name: "Vegan", size: "small", price: 17, quantity: 10, date: ISODate("2021-01-13T05:08:13Z") },
 { _id: 7, name: "Vegan", size: "medium", price: 18, quantity: 10, date: ISODate("2021-01-13T05:10:13Z") }
]);
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Pipeline des étapes d'agrégation

#### ■ Exemple de pipeline d'agrégation

- Nous souhaitons effectuer les actions suivantes
  - ✓ Restreindre sur les commandes effectuées entre deux dates
  - ✓ Calculer le **total** des commandes et la quantité **moyenne** des commandes groupés par nom de produit
  - ✓ Trier en ordre **croissant** sur le total des commandes calculées

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Pipeline des étapes d'agrégation

- Exemple de pipeline d'agrégation
- Schéma du pipeline

ORDERS  
Collection

```
[{"_id": 0, "name": "Pepperoni", "size": "small", "price": 19, "quantity": 10, "date": ISODate("2021-03-13T08:14:30.000Z")}, {"_id": 1, "name": "Pepperoni", "size": "medium", "price": 20, "quantity": 20, "date": ISODate("2021-03-13T09:13:24.000Z")}, {"_id": 2, "name": "Pepperoni", "size": "large", "price": 21, "quantity": 30, "date": ISODate("2021-03-17T09:22:12.000Z")}, {"_id": 3, "name": "Cheese", "size": "small", "price": 12, "quantity": 15, "date": ISODate("2021-03-13T11:21:39.736Z")}, {"_id": 4, "name": "Cheese", "size": "medium", "price": 13, "quantity": 50, "date": ISODate("2022-01-12T21:23:13.331Z")}, {"_id": 5, "name": "Cheese", "size": "large", "price": 14, "quantity": 10, "date": ISODate("2022-01-12T05:08:13.000Z")}, {"_id": 6, "name": "Vegan", "size": "small", "price": 17, "quantity": 10, "date": ISODate("2021-01-13T05:08:13.000Z")}, {"_id": 7, "name": "Vegan", "size": "medium", "price": 18, "quantity": 10, "date": ISODate("2021-01-13T05:10:13.000Z")}]
```

\$MATCH

```
[{"_id": 0, "name": "Pepperoni", "size": "small", "price": 19, "quantity": 10, "date": ISODate("2021-03-13T08:14:30.000Z")}, {"_id": 1, "name": "Pepperoni", "size": "medium", "price": 20, "quantity": 20, "date": ISODate("2021-03-13T09:13:24.000Z")}, {"_id": 2, "name": "Pepperoni", "size": "large", "price": 21, "quantity": 30, "date": ISODate("2021-03-17T09:22:12.000Z")}, {"_id": 3, "name": "Cheese", "size": "small", "price": 12, "quantity": 15, "date": ISODate("2021-03-13T11:21:39.736Z")}]
```

\$GROUP

```
{ "_id": { "nom": "Pepperoni" }, "totalOrderValue": 1220, "averageOrderQuantity": 20}, {"_id": { "nom": "Cheese" }, "totalOrderValue": 180, "averageOrderQuantity": 15}
```

\$SORT

```
{ "_id": { "nom": "Cheese" }, "totalOrderValue": 180, "averageOrderQuantity": 15}, {"_id": { "nom": "Pepperoni" }, "totalOrderValue": 1220, "averageOrderQuantity": 20}
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Pipeline des étapes d'agrégation

### ■ Exemple de pipeline d'agrégation

- Nous souhaitons effectuer les actions suivantes

```
db.orders.aggregate([
```

```
 // Etape 1, $MATCH : Restriction sur les Pizzas commandées entre deux dates
```

```
 { $match: { "date": { $gte: new ISODate("2021-01-30"), $lt: new ISODate("2021-12-30") } } },
```

```
 // Etape 2, $GROUP : Grouper les commandes restantes issues de l'étape 1 sur name
```

```
 { $group: { _id: { nom: "$name" } },
```

```
 totalOrderValue: { $sum: { $multiply: ["$price", "$quantity"] } },
```

```
 averageOrderQuantity: { $avg: "$quantity" } },
```

```
 // Etape 3, $SORT : Trier les documents issues de l'étape 2 ordre décroissant sur totalOrderValue
```

```
 { $sort: { totalOrderValue: 1 } });
```

```
{ _id: { nom: 'Cheese' }, totalOrderValue: 180, averageOrderQuantity: 15 },
{ _id: { nom: 'Pepperoni' }, totalOrderValue: 1220, averageOrderQuantity: 20 }
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$addFields

- Ajoute de nouveaux champs aux documents. **\$addFields** produit des documents qui contiennent tous les champs existants et les champs nouvellement ajoutés.
- Syntaxe  
`{ $addFields: { <newField>: <expression>, ... } }`

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$addFields

- Exemple

```
> use airbase
> db.createCollection("scores"); // collection scores homework : travail à la maison
> db.scores.insert({
 _id: 1,
 student: "Maya",
 homework: [10, 5, 10],
 quiz: [10, 8],
 extraCredit: 0
});

> db.scores.insert({
 _id: 2,
 student: "Ryan",
 homework: [5, 6, 5],
 quiz: [8, 8],
 extraCredit: 8
});
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$addFields

### ➤ Exemple

```
> db.scores.aggregate([
 {
 $addFields: {
 totalHomework: { $sum: "$homework" } ,
 totalQuiz: { $sum: "$quiz" }
 }
 },
 {
 $addFields: { totalScore:
 { $add: ["$totalHomework", "$totalQuiz", "$extraCredit"] } }
 }
])

// $add: additionne des nombres ou des dates avec des nombres
```

```
[
 {
 _id: 1,
 student: 'Maya',
 homework: [10, 5, 10],
 quiz: [10, 8],
 extraCredit: 0,
 totalHomework: 25,
 totalQuiz: 18,
 totalScore: 43
 },
 {
 _id: 2,
 student: 'Ryan',
 homework: [5, 6, 5],
 quiz: [8, 8],
 extraCredit: 8,
 totalHomework: 16,
 totalQuiz: 16,
 totalScore: 40
 }
]
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$bucket (plage)

- **Classement des documents entrants en groupes, appelés « buckets»**
  - Les groupes sont construits en fonction **d'une expression**
  - Une **liste de bornes** est fournie
  - **Une borne identifie un groupe de documents** compris entre une borne inférieure incluse et la borne supérieure non incluse
  - **Un document est produit pour chaque bucket** (plage) Chaque document de sortie contient un champ \_id dont la valeur correspond à la limite inférieure du groupe
  - **Default** : implique les documents inférieurs à la borne basse ou haute
  - L'**option de sortie** spécifie les champs inclus dans chaque document de sortie.
- **\$bucket** ne produit que les documents de sortie pour les **bornes** qui contiennent au moins un document d'entrée.

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$bucket (plage)

### ■ Syntaxe

```
{
 $bucket: {
 groupBy: <expression>,
 boundaries: [<lowerbound1>, <lowerbound2>, ...],
 default: <literal>,
 output: {
 <output1>: { <$accumulator expression> },
 ...
 <outputN>: { <$accumulator expression> }
 }
 }
}
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$bucket (plage)

### ➤ Exemple

```
db.artists.insertMany([
 { "_id" : 1, "last_name" : "Bernard", "first_name" : "Emil", "year_born" : 1868, "year_died" : 1941, "nationality" : "France" },
 { "_id" : 2, "last_name" : "Rippl-Ronai", "first_name" : "Joszef", "year_born" : 1861, "year_died" : 1927, "nationality" :
 "Hungary" },
 { "_id" : 3, "last_name" : "Ostroumova", "first_name" : "Anna", "year_born" : 1871, "year_died" : 1955, "nationality" : "Russia" },
 { "_id" : 4, "last_name" : "Van Gogh", "first_name" : "Vincent", "year_born" : 1853, "year_died" : 1890, "nationality" : "Holland" },
 { "_id" : 5, "last_name" : "Maurer", "first_name" : "Alfred", "year_born" : 1868, "year_died" : 1932, "nationality" : "USA" },
 { "_id" : 6, "last_name" : "Munch", "first_name" : "Edvard", "year_born" : 1863, "year_died" : 1944, "nationality" : "Norway" },
 { "_id" : 7, "last_name" : "Redon", "first_name" : "Odilon", "year_born" : 1840, "year_died" : 1916, "nationality" : "France" },
 { "_id" : 8, "last_name" : "Diriks", "first_name" : "Edvard", "year_born" : 1855, "year_died" : 1930, "nationality" : "Norway" }
]);
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$bucket(plage)

#### ■ Exemple

```
db.artists.aggregate([
 // First Stage
 {$bucket: {
 groupBy: "$year_born", // Field to group by
 boundaries: [1840, 1850, 1860, 1870, 1880], // Boundaries for the buckets
 default: "Other", // Bucket id for documents which do not fall into a bucket
 output: { // Output for each bucket
 "count": { $sum: 1 },
 "artists" :{$push: {"name": { $concat: ["$first_name", " ", "$last_name"] }, "year_born": "$year_born"} }
 }
 },
 // Second Stage
 {$match: { count: { $gt: 0} } }
]
);
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$bucket (plage)

### ■ Exemple

```
[
 {
 _id: 1840,
 count: 1,
 artists: [{ name: 'Odilon Redon', year_born: 1840 }]
 },
 {
 _id: 1850,
 count: 2,
 artists: [
 { name: 'Vincent Van Gogh', year_born: 1853 },
 { name: 'Edvard Diriks', year_born: 1855 }
]
 },
 {
 _id: 1860,
 count: 4,
 artists: [
 { name: 'Emil Bernard', year_born: 1868 },
 { name: 'Joszef Rippl-Ronai', year_born: 1861 },
 { name: 'Alfred Maurer', year_born: 1868 },
 { name: 'Edvard Munch', year_born: 1863 }
]
 },
 {
 _id: 1870,
 count: 1,
 artists: [{ name: 'Anna Ostroumova', year_born: 1871 }
 }]
]
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$bucketAuto

- **Catégorise les documents entrants en un nombre spécifique de groupes, appelés bucket, sur la base d'une expression spécifiée.** Les limites de ces groupes sont automatiquement déterminées afin de répartir les documents de manière uniforme dans le nombre de groupes spécifié.
- **Chaque groupe est représenté comme un document dans la sortie.** Le document de chaque groupe contient un champ `_id`, dont la valeur spécifie la limite inférieure inclusive et la limite supérieure exclusive du groupe, et un champ de comptage qui contient le nombre de documents dans le groupe. Le champ "count" est inclus par défaut lorsque la sortie n'est pas spécifiée.

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$bucketAuto (plages auto)

- **groupBy expression** : Une expression pour regrouper les documents. Pour spécifier un chemin d'accès à un champ, préfixez le nom du champ par le signe \$ et mettez-le entre guillemets.
- **buckets integer** : Un entier positif de 32 bits qui spécifie le nombre de plages dans lesquelles les documents d'entrée sont groupés
- **Buckets** : nombre de plages
- **Output, document de sortie** : Facultatif. Un document qui spécifie les champs à inclure dans les documents de sortie en plus du champ \_id. Pour spécifier le champ à inclure, vous devez utiliser des expressions d'accumulation (de groupe)

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$bucketAuto (plages auto)

### ➤ Syntaxe

```
{
 $bucketAuto: {
 groupBy: <expression>,
 buckets: <number>,
 output: {
 <output1>: { <$accumulator expression> },
 ...
 }
 granularity: <string>
 }
}
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$bucketAuto (plages auto)

### ■ Exemple

```
db.artwork.insertMany([
 {
 "_id": 1, "title": "The Pillars of Society", "artist": "Grosz", "year": 1926, "price": NumberDecimal("199.99"),
 "dimensions": { "height": 39, "width": 21, "units": "in" },
 { "_id": 2, "title": "Melancholy III", "artist": "Munch", "year": 1902, "price": NumberDecimal("280.00"),
 "dimensions": { "height": 49, "width": 32, "units": "in" },
 { "_id": 3, "title": "Dancer", "artist": "Miro", "year": 1925, "price": NumberDecimal("76.04"), "dimensions": {
 "height": 25, "width": 20, "units": "in" },
 { "_id": 4, "title": "The Great Wave off Kanagawa", "artist": "Hokusai", "price": NumberDecimal("167.30"),
 "dimensions": { "height": 24, "width": 36, "units": "in" },
 { "_id": 5, "title": "The Persistence of Memory", "artist": "Dali", "year": 1931, "price": NumberDecimal("483.00"),
 "dimensions": { "height": 20, "width": 24, "units": "in" },
 { "_id": 6, "title": "Composition VII", "artist": "Kandinsky", "year": 1913, "price": NumberDecimal("385.00"),
 "dimensions": { "height": 30, "width": 46, "units": "in" },
 { "_id": 7, "title": "The Scream", "artist": "Munch", "price": NumberDecimal("159.00"), "dimensions": { "height": 24,
 "width": 18, "units": "in" },
 { "_id": 8, "title": "Blue Flower", "artist": "O'Keefe", "year": 1918, "price": NumberDecimal("118.42"), "dimensions": {
 "height": 24, "width": 20, "units": "in" }
 }
]);
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$bucketAuto (plages auto)

#### ■ Exemple

```
db.artwork.aggregate([
 {
 $bucketAuto:
 groupBy: "$price",
 buckets: 4
 }
]
```

```
[
 {
 _id: { min: Decimal128("76.04"), max: Decimal128("159.00") },
 count: 2
 },
 {
 _id: { min: Decimal128("159.00"), max: Decimal128("199.99") },
 count: 2
 },
 {
 _id: { min: Decimal128("199.99"), max: Decimal128("385.00") },
 count: 2
 },
 {
 _id: { min: Decimal128("385.00"), max: Decimal128("483.00") },
 count: 2
 }
]
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$collStats

### ■ Renvoie des statistiques concernant une collection

- Nombre/durée de lectures (**read**), Nombre/durée d'écritures (**write**), ... dans la collection
- Les espaces disques, La mémoire consommée, les indexes, etc.

### ■ Syntaxe

```
{
 $collStats:
 {
 latencyStats: { histograms: <boolean> },
 storageStats: { scale: <number> },
 count: {}
 }
}
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$collStats

- Exemple

```
db.orders.insertMany([
 { "_id" : 1, "item" : "abc", "price" : 12, "quantity" : 2, "type": "apparel" },
 { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "type": "electronics" },
 { "_id" : 3, "item" : "abc", "price" : 10, "quantity" : 5, "type": "apparel" }
]);
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$collStats

### ■ Exemple

```
db.orders.aggregate([{ $collStats: {
 latencyStats: { histograms: true } } }])
```

### ■ Interprétation :

micros: Long("32"), count: Long("8")

8 lectures ont durées 32 microsecondes

```
[
 {
 ns: 'airbase.orders',
 host: 'PC-GABRIEL:27017',
 localTime: ISODate("2023-02-11T12:19:01.744Z"),
 latencyStats: {
 reads: {
 histogram: [
 { micros: Long("32"), count: Long("8") },
 { micros: Long("64"), count: Long("3") },
 { micros: Long("4096"), count: Long("1") }
],
 latency: Long("5674"),
 ops: Long("12")
 },
 writes: {
 histogram: [{ micros: Long("8192"), count: Long("1") }],
 latency: Long("10458"),
 ops: Long("1")
 },
 commands: {
 histogram: [{ micros: Long("8192"), count: Long("1") }],
 latency: Long("11425"),
 ops: Long("1")
 },
 transactions: { histogram: [], latency: Long("0"), ops: Long("0") }
 }
 }
]
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$collStats

### ■ Exemple

```
db.orders.aggregate([{ $collStats: {
 storageStats: {} } }])
```

### ■ Interprétation :

- ✓ Infos sur le cache mémoire
- ✓ Les indexes/ btree
- ✓ Le stockage

```
airbase> db.orders.aggregate([{ $collStats: { storageStats: {} } }])
[
 {
 ns: 'airbase.orders',
 host: 'PC-GABRIEL:27017',
 localTime: ISODate("2023-02-11T12:34:37.292Z"),
 storageStats: {
 size: 217,
 count: 3,
 avgObjSize: 72,
 numOrphanDocs: 0,
 storageSize: 20480,
 freeStorageSize: 0,
 btree: {
 'maximum internal page size': 4096,
 'maximum leaf page key size': 2867,
 'maximum leaf page size': 32768,
 },
 cache: {
 'bytes currently in the cache': 1568,
 'bytes dirty in the cache cumulative': 882,
 'bytes read into cache': 0,
 'bytes written from cache': 321,
 },
 },
 },
]
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$count

- Passe à l'étape suivante un document qui contient une décompte du nombre de documents entrés à l'étape précédente.
- Syntaxe

```
{ $count: <outPutFieldName string> }
```

### ■ Exemple

```
db.scores.insertMany([
 { "_id" : 1, "subject" : "History", "score" : 88 },
 { "_id" : 2, "subject" : "History", "score" : 92 },
 { "_id" : 3, "subject" : "History", "score" : 97 },
 { "_id" : 4, "subject" : "History", "score" : 71 },
 { "_id" : 5, "subject" : "History", "score" : 79 },
 { "_id" : 6, "subject" : "History", "score" : 83 }]);
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$count

#### ■ Exemple

Compte les documents dont **score** est > 80

```
db.scores.aggregate(
 [
 {$match: {score: { $gt: 80} }},
 {$count: "passing_scores"}
]
);
{ "passing_scores" : 4 }
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$facet

- **Traite plusieurs pipelines d'agrégation en une seule étape sur le même ensemble de documents d'entrée.**
  - Chaque sous-pipeline a son propre champ dans le document de sortie où ses résultats sont stockés sous forme de tableau de documents.
- **L'étape \$facet permet de créer des agrégations à facettes multiples qui caractérisent les données dans plusieurs dimensions, ou facettes, au cours d'une seule étape d'agrégation.**
  - Les agrégations à facettes multiples fournissent des filtres et des catégorisations multiples pour guider la navigation et l'analyse des données. Les détaillants utilisent couramment les facettes pour restreindre les résultats de recherche en créant des filtres sur le prix des produits, le fabricant, la taille, etc.
- **Les documents d'entrée ne passent qu'une seule fois à l'étape des facettes**
  - \$facet permet de faire plusieurs agrégations sur un même ensemble de documents d'entrée, sans avoir besoin de récupérer les documents d'entrée plusieurs fois

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$facet

#### ■ Syntaxe

```
{ $facet:
 {
 <outputField1>: [<stage1>, <stage2>, ...],
 <outputField2>: [<stage1>, <stage2>, ...],
 ...
 }
}
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$facet

#### ➤ Exemple

```
db.inventory.insertMany([
 { "_id" : 1, "title" : "The Pillars of Society", "artist" : "Grosz", "year" : 1926, "price" : NumberDecimal("199.99"), "tags" : ["painting",
 "satire", "Expressionism", "caricature"] },
 { "_id" : 2, "title" : "Melancholy III", "artist" : "Munch", "year" : 1902, "price" : NumberDecimal("280.00"), "tags" : ["woodcut",
 "Expressionism"] },
 { "_id" : 3, "title" : "Dancer", "artist" : "Miro", "year" : 1925, "price" : NumberDecimal("76.04"), "tags" : ["oil", "Surrealism", "painting"
],
 { "_id" : 4, "title" : "The Great Wave off Kanagawa", "artist" : "Hokusai", "price" : NumberDecimal("167.30"), "tags" : ["woodblock",
 "ukiyo-e"] },
 { "_id" : 5, "title" : "The Persistence of Memory", "artist" : "Dali", "year" : 1931, "price" : NumberDecimal("483.00"),
 "tags" : ["Surrealism", "painting", "oil"] },
 { "_id" : 6, "title" : "Composition VII", "artist" : "Kandinsky", "year" : 1913, "price" : NumberDecimal("385.00"), "tags" : ["oil",
 "painting", "abstract"] },
 { "_id" : 7, "title" : "The Scream", "artist" : "Munch", "year" : 1893, "tags" : ["Expressionism", "painting", "oil"] },
 { "_id" : 8, "title" : "Blue Flower", "artist" : "O'Keefe", "year" : 1918, "price" : NumberDecimal("118.42"), "tags" : ["abstract",
 "painting"] }
]);
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$facet

### ➤ Exemple

```
db.inventory.aggregate([
 {
 $facet: {
 "categorizedByTags": [{ $unwind: "$tags"}, { $sortByCount: "$tags" }],
 "categorizedByPrice": [
 // Filter out documents without a price equal greater, _id: 7
 { $match: { price: { $exists: 1 } } },
 {
 $bucket: {groupBy: "$price", boundaries: [0, 150, 200, 300, 400], default: "Other",
 output: {"count": { $sum: 1 }, "titles": { $push: "$title" }}}
 }
],
 "categorizedByYears(Auto)": [{$bucketAuto: {groupBy: "$year", buckets: 4}}]
 }
 }
]);
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$facet

### ■ Exemple

```
[{ $facet: { categorizedByTags: [{ _id: 'painting', count: 6 }, { _id: 'oil', count: 4 }, { _id: 'Expressionism', count: 3 }, { _id: 'abstract', count: 2 }, { _id: 'Surrealism', count: 2 }, { _id: 'caricature', count: 1 }, { _id: 'ukiyo-e', count: 1 }, { _id: 'woodblock', count: 1 }, { _id: 'woodcut', count: 1 }, { _id: 'satire', count: 1 }], categorizedByPrice: [{ _id: 0, count: 2, titles: ['Dancer', 'Blue Flower'] }, { _id: 150, count: 2, titles: ['The Pillars of Society', 'The Great Wave off Kanagawa'] }, { _id: 200, count: 1, titles: ['Melancholy III'] }, { _id: 300, count: 1, titles: ['Composition VII'] }, { _id: 'Other', count: 1, titles: ['The Persistence of Memory'] }], categorizedByYears(Auto): [{ _id: { min: null, max: 1902 }, count: 2 }, { _id: { min: 1902, max: 1918 }, count: 2 }, { _id: { min: 1918, max: 1926 }, count: 2 }, { _id: { min: 1926, max: 1931 }, count: 2 }] } }
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$graphLookup

- Effectue une recherche récursive (**hiérarchique**) sur une collection, avec des options permettant de restreindre la recherche par profondeur de récursion et filtre de requête.
- Ressemble aux requêtes hiérarchiques en SQL

#### ▪ Syntaxe

```
{
 $graphLookup: {
 from: <collection>,
 startWith: <expression>,
 connectFromField: <string>,
 connectToField: <string>,
 as: <string>,
 maxDepth: <number>,
 depthField: <string>,
 restrictSearchWithMatch: <document>
 }
}
```

## Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$graphLookup

#### ■ Exemple

```
db.employees.insertMany(
 [
 { "_id" : 1, "name" : "Dev" },
 { "_id" : 2, "name" : "Eliot", "reportsTo" : "Dev" },
 { "_id" : 3, "name" : "Ron", "reportsTo" : "Eliot" },
 { "_id" : 4, "name" : "Andrew", "reportsTo" : "Eliot" },
 { "_id" : 5, "name" : "Asya", "reportsTo" : "Ron" },
 { "_id" : 6, "name" : "Dan", "reportsTo" : "Andrew" }
]
);
```

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$graphLookup

### ■ Exemple

```
db.employees.aggregate([
 {
 $graphLookup: {
 from: "employees",
 startWith: "$reportsTo",
 connectFromField: "reportsTo",
 connectToField: "name",
 as: "reportingHierarchy"
 }
 }
]);
```

1. Pour chaque document d'entrée, la recherche commence par la valeur désignée par **startWith**.
2. **\$graphLookup fait correspondre la valeur startWith avec le champ désigné par connectToField dans les autres documents de la collection from.**
3. Pour chaque document **correspondant en 2**, **\$graphLookup prend la valeur de connectFromField et vérifie chaque document dans la collection from pour une valeur connectToField correspondante.**
4. Pour chaque **correspondance en 3**, **\$graphLookup ajoute le document correspondant dans la collection from à un champ de tableau nommé par le paramètre as.**

L'étape s'exécute récursivement jusqu'à ce qu'il n'y ai plus de document qui matche.

# Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB

## ➤ Etape d'agrégation \$graphLookup

### ■ Exemple

```
[
 { _id: 1, name: 'Dev', reportingHierarchy: [] },
 {
 _id: 2,
 name: 'Eliot',
 reportsTo: 'Dev',
 reportingHierarchy: [{ _id: 1, name: 'Dev' }]
 },
 {
 _id: 3,
 name: 'Ron',
 reportsTo: 'Eliot',
 reportingHierarchy: [
 { _id: 2, name: 'Eliot', reportsTo: 'Dev' },
 { _id: 1, name: 'Dev' }
]
 },
]
```

```
{
 _id: 4,
 name: 'Andrew',
 reportsTo: 'Eliot',
 reportingHierarchy: [
 { _id: 2, name: 'Eliot', reportsTo: 'Dev' },
 { _id: 1, name: 'Dev' }
]
},
{
 _id: 5,
 name: 'Asha',
 reportsTo: 'Ron',
 reportingHierarchy: [
 { _id: 3, name: 'Ron', reportsTo: 'Eliot' },
 { _id: 2, name: 'Eliot', reportsTo: 'Dev' },
 { _id: 1, name: 'Dev' }
]
},
{
 _id: 6,
 name: 'Dan',
 reportsTo: 'Andrew',
 reportingHierarchy: [
 { _id: 1, name: 'Dev' },
 { _id: 2, name: 'Eliot', reportsTo: 'Dev' },
 { _id: 4, name: 'Andrew', reportsTo: 'Eliot' }
]
}
```

## Module M4.4, section 6, partie 1 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

# Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

## ➤ Plan

- Etape d'agrégation \$group (Group By)
- Etape d'agrégation \$indexStats
- Etape d'agrégation \$limit
- Etape d'agrégation \$listSessions
- Etape d'agrégation \$lookup (jointure)
- Etape d'agrégation \$match
- Etape d'agrégation \$out
- Etape d'agrégation \$project
- Etape d'agrégation \$replaceRoot
- Etape d'agrégation \$sample
- Etape d'agrégation \$skip
- Etape d'agrégation \$sort
- Etape d'agrégation \$sortByCount
- Etape d'agrégation \$unwind (unnest)

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$group (Group By)

- Regroupe les documents en entrée par l'expression `_id` spécifiée et, pour chaque regroupement distinct, produit un document. C'est l'équivalent du GROUP BY en SQL
- Le champ `_id` de chaque document de sortie contient la clé du groupe unique par valeur.
- Les documents de sortie peuvent également contenir des champs calculés qui contiennent les valeurs d'une expression de groupe

#### ▪ Syntaxe

```
{
 $group:
 {
 _id: <expression>, // Group By Expression
 <field1>: { <accumulator1> : <expression1> },
 ...
 }
}
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$group (Group By)

#### ■ Exemple

```
db.books.insertMany([
 { "_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 },
 { "_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 },
 { "_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 },
 { "_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 },
 { "_id" : 7020, "title" : "Iliad", "author" : "Homer", "copies" : 10 }
])
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$group (Group By)

#### ■ Exemple

```
db.books.aggregate([
 { $group : { _id : "$author", books: { $push: "$title" } } }
])
[
 {
 _id: 'Dante',
 books: ['The Banquet', 'Divine Comedy', 'Eclogues']
 },
 { _id: 'Homer', books: ['The Odyssey', 'Iliad'] }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$indexStats

- Renvoie les statistiques sur l'utilisation des indexes d'une collection. S'il fonctionne avec un contrôle d'accès, l'utilisateur doit avoir des priviléges qui incluent l'action indexStats.

- **Syntaxe**

```
{ $indexStats: {} }
```

- **Exemple**

```
db.orders.insertMany([
 { "_id" : 1, "item" : "abc", "price" : 12, "quantity" : 2, "type": "apparel" },
 { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "type": "electronics" },
 { "_id" : 3, "item" : "abc", "price" : 10, "quantity" : 5, "type": "apparel" }
]);
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$indexStats

#### ■ Exemple : création d'indexes

```
db.orders.createIndex({ item: 1, quantity: 1 })
db.orders.createIndex({ type: 1, item: 1 })
```

#### ■ Exemple : quelques actions

```
db.orders.find({ type: "apparel" })
[
 { _id: 1, item: 'abc', price: 12, quantity: 2, type: 'apparel' },
 { _id: 3, item: 'abc', price: 10, quantity: 5, type: 'apparel' }
]

db.orders.find({ item: "abc" }).sort({ quantity: 1 })
[
 { _id: 1, item: 'abc', price: 12, quantity: 2, type: 'apparel' },
 { _id: 3, item: 'abc', price: 10, quantity: 5, type: 'apparel' }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$indexStats

#### ■ Exemple : index stats

```
db.orders.aggregate(
 [
 { $indexStats: {} }
]
);
```

```
[
 {
 name: 'item_1_quantity_1',
 key: { item: 1, quantity: 1 },
 host: 'PC-GABRIEL:27017',
 accesses: { ops: Long("1"), since: ISODate("2022-12-29T14:14:23.125Z") },
 spec: { v: 2, key: { item: 1, quantity: 1 }, name: 'item_1_quantity_1' }
 },
 {
 name: '_id_',
 key: { _id: 1 },
 host: 'PC-GABRIEL:27017',
 accesses: { ops: Long("0"), since: ISODate("2022-12-29T11:36:15.101Z") },
 spec: { v: 2, key: { _id: 1 }, name: '_id_' }
 },
 {
 name: 'type_1_item_1',
 key: { type: 1, item: 1 },
 host: 'PC-GABRIEL:27017',
 accesses: { ops: Long("1"), since: ISODate("2022-12-29T14:14:25.697Z") },
 spec: { v: 2, key: { type: 1, item: 1 }, name: 'type_1_item_1' }
 }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$limit

- Limite le nombre de documents qui passent à l'étape suivante. Renvoie les N premiers documents

- Syntaxe

```
{ $limit: <positive integer> }
```

- Exemple

```
db.orders.aggregate([
 { $limit : 2 }
];])
```

The output of the aggregation pipeline is:

```
[
 { _id: 1, item: 'abc', price: 12, quantity: 2, type: 'apparel' },
 { _id: 2, item: 'jkl', price: 20, quantity: 1, type: 'electronics' }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$listSessions

- Liste toutes les sessions stockées dans la collection `system.sessions` dans la base de données «**config**». Ces sessions sont visibles par tous les membres l'instance MongoDB.

- Syntaxe

```
{ $listSessions: <document> }
```

- Exemple

```
use config
db.system.sessions.aggregate([
 { $listSessions: {} }
])
```

```
[{
 "_id": {
 "id": new UUID("c1b3b37a-c5e4-4784-88af-68d68a5f4586"),
 "uid": Binary(Buffer.from("e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855", "hex"), 0)
 },
 "lastUse": ISODate("2022-12-29T14:37:12.577Z")
}]
```

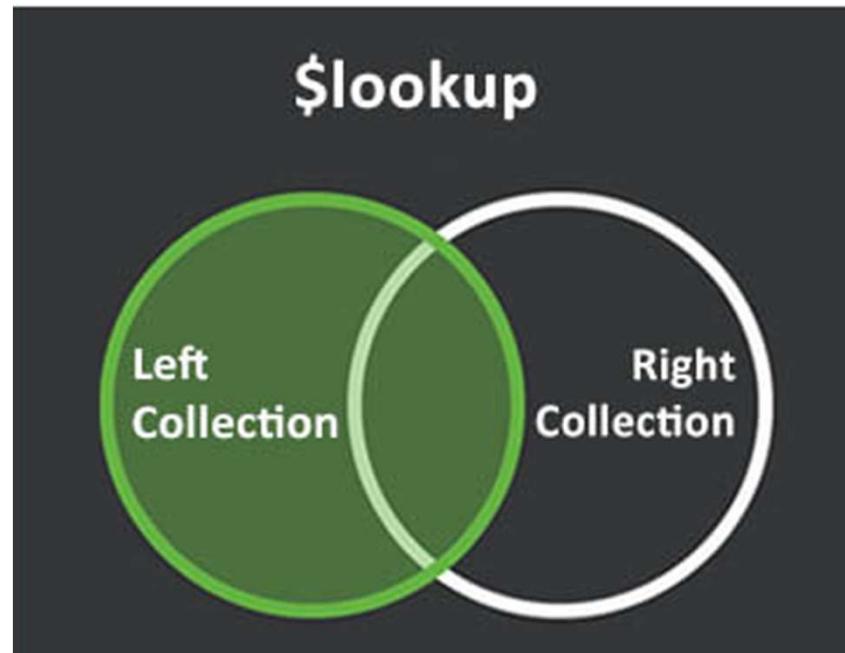
## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$lookup (jointure)

- Cette étape de recherche (**lookup**) permet d'effectuer l'équivalent d'une jointure externe dans un système de gestion de bases de données relationnelles (SGBDR) et ne peut opérer que sur des collections non shardées situées dans la même base de données
- À chaque document saisi, l'étape **\$lookup ajoute un nouveau champ de tableau** dont les éléments sont les documents correspondants de la collection "jointe".
- L'étape **\$lookup** fait passer ces documents remodelés à l'étape suivante.
- Nous ne traitons ici que l'équijointure dans un premier temps
- Il s'agit d'une jointure **externe gauche**

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

- Etape d'agrégation \$lookup (jointure)



## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$lookup (jointure)

#### ▪ Equijointure

- Pour effectuer une correspondance d'égalité entre un champ des documents d'entrée et un champ des documents de la collection "jointe", l'étape \$lookup a la syntaxe suivante :

```
db.localCollection.aggregate([
 $lookup:
 {
 from: <collection to join>,
 localField: <field from the input documents from localCollection>,
 foreignField: <field from the documents of the "from" collection>,
 as: <output array field>
 }
])
}
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$lookup (jointure)

- Equijointure

- Equivaut à

```
SELECT *, <output array field>
FROM collection
WHERE <output array field>
IN (SELECT *
FROM <collection to join>
WHERE <foreignField>= <collection.localField>);
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$lookup (jointure)

- Equijointure : Exemple

- Creation de a collection **orders2** avec les documents suivants :

```
db.orders2.insertMany([
 { "_id" : 1, "item" : "almonds", "price" : 12, "quantity" : 2 },
 { "_id" : 2, "item" : "pecans", "price" : 20, "quantity" : 1 },
 { "_id" : 3, "item" : "amandes" }
]);
```

- Creation d'une autre collection **inventory2** avec les documents suivants :

```
db.inventory2.insertMany([
 { "_id" : 1, "sku" : "almonds", "description": "product 1", "instock" : 120 },
 { "_id" : 2, "sku" : "bread", "description": "product 2", "instock" : 80 },
 { "_id" : 3, "sku" : "cashews", "description": "product 3", "instock" : 60 },
 { "_id" : 4, "sku" : "pecans", "description": "product 4", "instock" : 70 },
 { "_id" : 5, "sku": null, "description": "Incomplete" },
 { "_id" : 6 }
]);
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$lookup (jointure)

- Equijointure : Exemple

```
db.orders2.aggregate([
 {
 $lookup:
 {
 from: "inventory2",
 localField: "item",
 foreignField: "sku",
 as: "inventory_docs"
 }
 }
])
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$lookup (jointure)

#### ■ Equijointure : Exemple

```
[
 {
 _id: 1,
 item: 'almonds',
 price: 12,
 quantity: 2,
 inventory_docs: [
 {
 _id: 1,
 sku: 'almonds',
 description: 'product 1',
 instock: 120
 }
],
 {
 _id: 2,
 item: 'pecans',
 price: 20,
 quantity: 1,
 inventory_docs: [
 { _id: 4, sku: 'pecans', description: 'product 4', instock: 70 }
]
 },
 { _id: 3, item: 'amandes', inventory_docs: [] }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$match

➤ **Filtre** les documents pour ne transmettre que les documents qui correspondent à la (aux) condition(s) spécifiée(s), à l'étape suivante du pipeline

### ➤ Syntaxe

```
{ $match: { <query> } }
```

### ➤ Exemple

```
db.orders2.aggregate([
 {
 $match: { "_id": 1 }
 }
]).pretty();
```

```
testgm> db.orders2.aggregate([
... {
... $match: { "_id": 1 }
... }
...]).pretty();
[{ _id: 1, item: 'almonds', price: 12, quantity: 2 }]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$out

➤ Prend les documents retournés par le pipeline d'agrégation et les écrit dans une collection spécifique dont le nom apparaît derrière \$out. L'opérateur \$out doit être la dernière étape du pipeline. L'opérateur \$out permet au cadre d'agrégation de renvoyer des ensembles de résultats de n'importe quelle taille.

#### ■ Syntaxe

```
{ $out: "<output-collection>" }
```

#### ■ Exemple

```
db.orders2.aggregate([
 {
 $match: { "_id": 1 },
 },
 {$out :"Commande1"}
]).pretty();
```

```
testgm> db.Commande1.find();
[{ _id: 1, item: 'almonds', price: 12, quantity: 2 }]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$project

- Transmet les documents avec les champs demandés à l'étape suivante du processus. Les champs spécifiés peuvent être des champs existants dans les documents de saisie ou des champs nouvellement calculés. **Correspond à la projection SQL.**

#### ■ Syntaxe

```
{ $project: { <specification(s)> } }
```

#### ■ Exemple

```
db.orders2.aggregate([
 {"$project": {
 "item": 1
 }
}).pretty();
```

```
[{
 "_id": 1, item: 'almonds' },
 { _id: 2, item: 'pecans' },
 { _id: 3, item: 'amandes' }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$replaceRoot

- Remplace le document en entrée par le document spécifié
- L'opération remplace tous les champs existants dans le document de saisie, y compris le champ \_id.
- Vous pouvez promouvoir un document intégré existant au niveau supérieur, ou créer un nouveau document à des fins de promotion (voir exemple).
- Syntaxe

{ \$replaceRoot: { newRoot: <replacementDocument> } }

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$replaceRoot

#### ■ Exemple

```
db.personnes.insertMany([
 { "_id": 1, "name" : { "first" : "John", "last" : "Backus" } },
 { "_id": 2, "name" : { "first" : "John", "last" : "McCarthy" } },
 { "_id": 3, "name": { "first" : "Grace", "last" : "Hopper" } }
]);
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$replaceRoot

#### ➤ Exemple

```
db.personnes.aggregate([
 { $replaceRoot: { newRoot: "$name" } }
])

[
 { first: 'John', last: 'Backus' },
 { first: 'John', last: 'McCarthy' },
 { first: 'Grace', last: 'Hopper' }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$replaceRoot

#### ➤ Exemple 2 : créer et sauver la nouvelle collection

```
db.personnes.aggregate([
 { $replaceRoot: { newRoot: "$name" } },
 {$out:"newpers" }
])
```

```
[{
 _id: ObjectId("63adc27a849d50bef823c869"),
 first: 'John',
 last: 'Backus'
},
{
 _id: ObjectId("63adc27a849d50bef823c86a"),
 first: 'John',
 last: 'McCarthy'
},
{
 _id: ObjectId("63adc27a849d50bef823c86b"),
 first: 'Grace',
 last: 'Hopper'
}]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$sample

- Sélectionne au hasard (aléatoirement) le nombre de documents spécifié dans l'expression
- **Syntaxe**  
`{$sample: { size: <positive integer> } }`
- **Exemple**

```
db.sampleSource.insertMany([
 { "_id" : 1, "name" : "david", "q1" : true, "q2" : true },
 { "_id" : 2, "name" : "Zorro", "q1" : false, "q2" : false },
 { "_id" : 3, "name" : "Annie", "q1" : true, "q2" : true },
 { "_id" : 4, "name" : "Allan", "q1" : true, "q2" : false },
 { "_id" : 5, "name" : "Zembla", "q1" : false, "q2" : true },
 { "_id" : 6, "name" : "Zoé", "q1" : true, "q2" : true },
 { "_id" : 7, "name" : "Sonia", "q1" : false, "q2" : true }
]);
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$sample

- Exemple

```
db.sampleSource.aggregate(
 [{ $sample: { size: 3 } }]
)
```

- Résultat

```
[
 { _id: 1, name: 'david', q1: true, q2: true },
 { _id: 5, name: 'Zembla', q1: false, q2: true },
 { _id: 7, name: 'Sonia', q1: false, q2: true }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$skip

- **Sauter le nombre spécifié de documents et faire passer les documents restants à l'étape suivante**
- Permet de **sauter N documents**
- **Syntaxe**  
{ \$skip: <positive integer> }
- **Exemple :** sauter les 5 premiers documents  
`db.sampleSource.aggregate({ $skip : 5 });`

```
[
 { _id: 6, name: 'Zoé', q1: true, q2: true },
 { _id: 7, name: 'Sonia', q1: false, q2: true }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$sort

- trie tous les documents en entrée et les renvoie au pipeline dans l'ordre spécifié
- **Syntaxe**

```
{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }
```

#### ■ Exemple

```
db.sampleSource.aggregate(
 { $sort :{ "q1": 1 , name : 1 } }
);
```

```
[
 { _id: 7, name: 'Sonia', q1: false, q2: true },
 { _id: 5, name: 'Zembla', q1: false, q2: true },
 { _id: 2, name: 'Zorro', q1: false, q2: false },
 { _id: 4, name: 'Allan', q1: true, q2: false },
 { _id: 3, name: 'Annie', q1: true, q2: true },
 { _id: 6, name: 'Zoé', q1: true, q2: true },
 { _id: 1, name: 'david', q1: true, q2: true }
]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$sortByCount

- Regroupe les documents entrants en fonction de la valeur d'une expression donnée, puis calcule le nombre de documents dans chaque groupe distinct.
- Chaque document en sortie contient deux champs : un champ `_id` contenant la valeur du groupe distinct, et un champ de comptage contenant le nombre de documents appartenant à ce groupe ou à cette catégorie.
- Les documents sont triés par nombre dans l'ordre décroissant.

### ➤ Syntaxe

```
{ $sortByCount: <expression> }
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$sortByCount

#### ■ Exemple

```
db.sampleSource.aggregate(
 { $sortByCount : "$q1" }
);
```

```
[{ _id: true, count: 4 }, { _id: false, count: 3 }]
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$unwind (unnest)

- Accepte en entrée un document ayant un Array.
- Produit en sortie le dit document répété avec chaque éléments de l'Array si unwind sur l'Array
- Converti un ARRAY en une TABLE (équivalent de l'opérateur TABLE en SQL3)
- Exemple {A, [B,C, D]} => devient après \$UNWIND [B,C, D]

A, B

A, C

A, D

#### ■ Syntaxe

{ \$unwind: <field path> }

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$unwind (unnest)

- Exemple

```
db.oeuvres.insertMany([
 { "_id" : 1, "title" : "The Pillars of Society", "artist" : "Grosz", "year" : 1926, "tags" : ["painting", "satire",
 "Expressionism", "caricature"] },
 { "_id" : 2, "title" : "Melancholy III", "artist" : "Munch", "year" : 1902, "tags" : ["woodcut", "Expressionism"] },
 { "_id" : 3, "title" : "Dancer", "artist" : "Miro", "year" : 1925, "tags" : ["oil", "Surrealism", "painting"] },
 { "_id" : 4, "title" : "The Great Wave off Kanagawa", "artist" : "Hokusai", "tags" : ["woodblock", "ukiyo-e"] },
 { "_id" : 5, "title" : "The Persistence of Memory", "artist" : "Dali", "year" : 1931, "tags" : ["Surrealism", "painting",
 "oil"] },
 { "_id" : 6, "title" : "Composition VII", "artist" : "Kandinsky", "year" : 1913, "tags" : ["oil", "painting", "abstract"] },
 { "_id" : 7, "title" : "The Scream", "artist" : "Munch", "year" : 1893, "tags" : ["Expressionism", "painting", "oil"] },
 { "_id" : 8, "title" : "Blue Flower", "artist" : "O'Keefe", "year" : 1918, "tags" : ["abstract", "painting"] }
]);
```

## Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB

### ➤ Etape d'agrégation \$unwind (unnest)

#### ■ Exemple

```
db.oeuvres.aggregate([{$sample:{size:2}}, { $unwind : "$tags" }]);
```

-- Application de **unwind** sur un **échantillon de 2 documents** de la collection **oeuvres**

```
[
 {_id: 7, title: 'The Scream', artist: 'Munch', year: 1893, tags: 'Expressionism'},
 {_id: 7, title: 'The Scream', artist: 'Munch', year: 1893, tags: 'painting'},
 {_id: 7, title: 'The Scream', artist: 'Munch', year: 1893, tags: 'oil'},
 {_id: 6, title: 'Composition VII', artist: 'Kandinsky', year: 1913, tags: 'oil'},
 {_id: 6, title: 'Composition VII', artist: 'Kandinsky', year: 1913, tags: 'painting'},
 {_id: 6, title: 'Composition VII', artist: 'Kandinsky', year: 1913, tags: 'abstract'}
]
```

## Module M4.4, section 6, partie 2 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

# Module M4.4 : INTRODUCTION A MONGODB ET LE MONGO SHELL

## ➤ Bilan

- **Module M4.4, section 1 : Objectif et carte de visite MongoDB**
- **Module M4.4, section 2 : Installation, outils architecture MongoDB**
- **Module M4.4, section 3 : Gestion des Bases de Données et des Collections MongoDB**
- **Module M4.4, section 4 : Gestion des Documents MongoDB**
- **Module M4.4, section 5 : Indexation et recherches de documents MongoDB**
- **Module M4.4, section 6, partie 1 : Les agrégats dans MongoDB**
- **Module M4.4, section 6, partie 2 : Les agrégats dans MongoDB**

## ➤ Exercices

# Module M4.5 : INTRODUCTION A MONGODB ET SON API JAVA

G. Mopolo-Moké  
Professeur chargé d'enseignements  
Université Côte d'Azur (UCA)

2022 / 2023

# Module M4.5 : INTRODUCTION A MONGODB ET SON API

## JAVA

### ➤ Plan

- **Module M4.5, section 1 : Prise en main de l'API Java**
- **Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple**
- **Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple**

## Module M4.5, section 1 : Prise en main de l'API Java

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Plan

- Principales classes de la nouvelle API Java MongoDB
- Nouvelle API Java MongoDB versus ancienne API
- Installation du driver MongoDB JDBC
- Connexion à la base
- Création d'une collection
- Chargement d'une collection
- Insertion de documents dans une collection
- Recherche de document dans une collection
- Modification de documents
- Suppression de documents

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Principales classes de la nouvelle API Java MongoDB

| Nom de la classe          | Description                                                                                        |
|---------------------------|----------------------------------------------------------------------------------------------------|
| MongoClient               | Permet de créer un pointeur vers une instance MongoDB qui tourne 1 machine                         |
| MongoCredential           | Classe fournissant les fonctions d'authentification dans une base MongoDB                          |
| MongoDatabase             | Fournit des fonctions qui permettent d'agir dans 1 base MongoDB. Par exemple créer des collections |
| MongoCollection<Document> | Fournit des fonctions manipulation des collections (find, aggregate, ...)                          |
| Document                  | Classe permettant de construire tout type de document                                              |
| FindIterable<Document>    | Fournit un itérateur lié à la méthode find d'une collection                                        |

Lien vers l'API JAVA MONGODB (suivre Modern API): <https://mongodb.github.io/mongo-java-driver/>

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Principales classes de la nouvelle API Java MongoDB

| Nom de la classe            | Description                                                              |
|-----------------------------|--------------------------------------------------------------------------|
| UpdateResult                | Résultat d'une opération de mise à jour                                  |
| AggregateIterable<Document> | Fournit un itérateur lié à la méthode aggregate d'une collection         |
| Aggregates                  | Classe fournissant les fonctions d'agrégation : find, match, unwind, ... |
| IndexOptions                | Fournit les options lors de la création d'un index                       |
|                             |                                                                          |
|                             |                                                                          |
|                             |                                                                          |

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Nouvelle API Java MongoDB versus ancienne API

| Ancienne API        | Nouvelle API              |
|---------------------|---------------------------|
| MongoClientOptions  | MongoClientSettings       |
| MongoClient.getDB() | MongoClient.getDatabase() |
| DB                  | MongoDatabase             |
| new MongoClient()   | MongoClients.create()     |
| DBCollection        | MongoCollection<>         |
| DBCursor            | MongoCursor               |

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Nouvelle API Java MongoDB versus ancienne API

| Ancienne API                 | Nouvelle API                                                                                                      |
|------------------------------|-------------------------------------------------------------------------------------------------------------------|
| DBCollection.findOne()       | MongoCollection.find().first()                                                                                    |
| DBCollection.insert()        | MongoCollection.insertOne(), MongoCollection.insertMany()                                                         |
| DBCollection.update()        | MongoCollection.updateOne(), MongoCollection.updateMany(),<br>MongoCollection.replaceOne()                        |
| DBCollection.remove(),       | MongoCollection.deleteOne(), MongoCollection.deleteMany()                                                         |
| DBCollection.count()         | MongoCollection.countDocuments(),<br>MongoCollection.estimatedDocumentCount()                                     |
| DBCollection.findAndModify() | MongoCollection.findOneAndUpdate(),<br>MongoCollection.findOneAndReplace(),<br>MongoCollection.findOneAndDelete() |

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Installation du driver MongoDB JDBC

- Nous supposons que vous avez déjà installé java sur votre machine
- Créer un dossier tpmongodb sur un disque ou vous avez de la place. Dans ce dossier créer ensuite les sous-dossiers suivants :
  - mongojar
  - tp
- Télécharger mongo-java-driver-3.12.X.jar depuis ici :  
<http://central.maven.org/maven2/org/mongodb/mongo-java-driver/3.12.X/>. Placez ce jar dans le dossier tpmongodb\mongojar
- Placez tous vos programmes dans tp
- Dans chacun de vos programmes, il faut définir tp comme package

## Module M4.5, section 1 : Prise en main de l'API Java

### ■ Installation du driver MongoDB JDBC

- Assurez que **la variable d'environnement PATH contient le path java** : ..\java\jdk-x.x.x\bin

- Définir une variable d'environnement comme suit :

Set MYPATH=D:\tpmongodb

- Ligne de compilation

```
javac -g -cp %MYPATH%\mongojar\mongo-java-driver-3.12.X.jar;%MYPATH%
%MYPATH%\tp\nomFichier.java
```

- Ligne d'exécution

```
java -Xmx256m -Xms256m -cp %MYPATH%\mongojar\mongo-java-driver-3.12.X.jar;%MYPATH%
tp.nomFichier
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Connexion à la base

```
package tp;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;

public class ConnectToDB {
 public static void main(String args[]) {
 // Creating a Mongo client
 MongoClient mongo = new MongoClient(
 "localhost" , 27017);
 }
}
```

```
// Creating Credentials
MongoCredential credential;
credential =
 MongoCredential.createCredential("sampleUser",
"myDb",
 "password".toCharArray());
System.out.println("Connected to the database
successfully");

// Accessing the database
MongoDatabase database =
mongo.getDatabase("myDb");
System.out.println("Credentials ::"+ credential);
 }
}
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤Création d'une collection

```
package tp;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;

public class CreatingCollection {

 public static void main(String args[]) {

 // Creating a Mongo client
 MongoClient mongo = new MongoClient("localhost" ,
27017);
```

```
// Creating Credentials
MongoCredential credential;
credential = MongoCredential.createCredential("sampleUser",
"myDb",
"password".toCharArray());
System.out.println("Connected to the database successfully");

//Accessing the database
MongoDatabase database = mongo.getDatabase("myDb");

//Creating a collection
database.createCollection("sampleCollection");
System.out.println("Collection created successfully");
}
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Chargement d'une collection

```
package tp;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;

public class SelectingCollection {

 public static void main(String args[]) {

 // Creating a Mongo client
 MongoClient mongo = new MongoClient("localhost" , 27017);
```

```
// Creating Credentials
MongoCredential credential;
credential = MongoCredential.createCredential("sampleUser", "myDb",
 "password".toCharArray());
System.out.println("Connected to the database successfully");

// Accessing the database
MongoDatabase database = mongo.getDatabase("myDb");

// Retieving a collection
MongoCollection<Document> collection =
 database.getCollection("myCollection");
System.out.println("Collection myCollection selected successfully");
}
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Insertion de documents dans une collection

```
package tp;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class InsertingDocument {
 public static void main(String args[]) {
 // Creating a Mongo client
 MongoClient mongo = new MongoClient("localhost" , 27017);
 // Creating Credentials
 MongoCredential credential;
 credential = MongoCredential.createCredential("sampleUser",
 "myDb","password".toCharArray());
 System.out.println("Connected to the database successfully");
 // Accessing the database
 MongoDatabase database = mongo.getDatabase("myDb");
 }
}
```

```
// Retrieving a collection
MongoCollection<Document> collection =
database.getCollection("sampleCollection");
System.out.println("Collection sampleCollection selected
successfully");
Document document = new Document("title", "MongoDB")
.append("id", 1)
.append("description", "database")
.append("likes", 100)
.append("url", "http://www.tutorialspoint.com/mongodb/")
.append("by", "tutorialspoint");
collection.insertOne(document);
System.out.println("Document inserted successfully");
}
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Rechercher tous les documents

```
package tp;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class RetrievingAllDocuments {
 public static void main(String args[]) {
 // Creating a Mongo client
 MongoClient mongo = new MongoClient("localhost" , 27017);
 // Creating Credentials
 MongoCredential credential;
 credential = MongoCredential.createCredential("sampleUser", "myDb",
 "password".toCharArray());
 System.out.println("Connected to the database successfully");
 }
}
```

```
// Accessing the database
MongoDatabase database = mongo.getDatabase("myDb");
// Retrieving a collection
MongoCollection<Document> collection =
database.getCollection("sampleCollection");
System.out.println("Collection sampleCollection selected successfully");
// Getting the iterable object
FindIterable<Document> iterDoc = collection.find();
int i = 1;
// Getting the iterator
Iterator it = iterDoc.iterator();
while (it.hasNext()) {
 System.out.println(it.next());
 i++;
}
}}
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Modification de documents

```
package tp;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Updates;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
public class UpdatingDocuments {
 public static void main(String args[]) {
 MongoClient mongo = new MongoClient("localhost" , 27017);
 MongoCredential credential =
 MongoCredential.createCredential("sampleUser", "myDb",
 "password".toCharArray());
 System.out.println("Connected to the database successfully");
 }
}
```

### // Accessing the database

```
MongoDatabase database = mongo.getDatabase("myDb");
// Retrieving a collection
MongoCollection<Document> collection =
 database.getCollection("sampleCollection");
System.out.println("Collection myCollection selected successfully");
collection.updateOne(Filters.eq("id", 1), Updates.set("likes", 150));
System.out.println("Document update successfully...");
// Retrieving the documents after updation
// Getting the iterable object
FindIterable<Document> iterDoc = collection.find();
int i = 1;
// Getting the iterator
Iterator it = iterDoc.iterator();
while (it.hasNext()) { System.out.println(it.next()); i++; }
}
```

## Module M4.5, section 1 : Prise en main de l'API Java

### ➤ Suppression du document

```
package tp;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;

public class DeletingDocuments {
 public static void main(String args[]) {
 MongoClient mongo = new MongoClient("localhost" , 27017);
 MongoCredential credential=MongoCredential.createCredential(
 "sampleUser" , "myDb" , "password".toCharArray());
 System.out.println("Connected to the database successfully");
 MongoDatabase database = mongo.getDatabase("myDb");
 }
}
```

```
// Retrieving a collection
MongoCollection<Document> collection =
 database.getCollection("sampleCollection");
System.out.println("Collection sampleCollection selected
successfully");
collection.deleteOne(Filters.eq("id", 1));
System.out.println("Document deleted successfully... ");
FindIterable<Document> iterDoc = collection.find();
int i = 1;
Iterator it = iterDoc.iterator();
while (it.hasNext()) {
 System.out.println("Inserted Document: "+i);
 System.out.println(it.next());
 i++;
}
}}
```

## Module M4.5, section 1 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

## Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

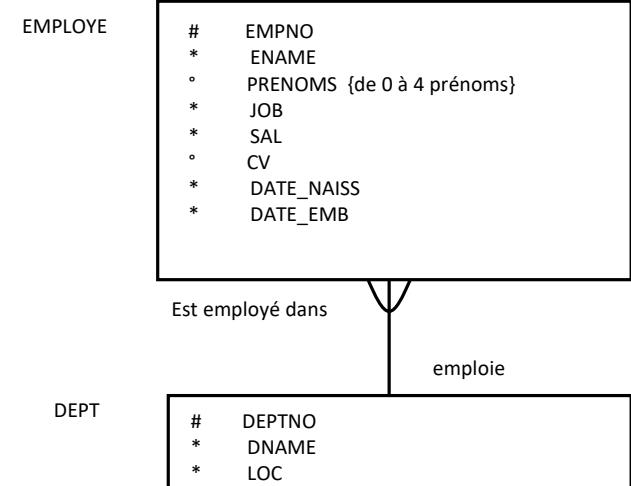
## ➤ Plan

- Exemple de référence
- Classes à importer
- Définition de la classe et Constructeur
- Création et suppression de collections
- Méthode d'insertion d'un document
- Méthode d'insertion de documents multiples
- Méthode de recherche d'un document identifié
- Méthode de recherche conditionnelle avec tri et projection
- Méthode de modification d'un ou plusieurs documents
- Méthodes de suppression de documents

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Exemple de référence

- Définition du schéma conceptuel de l'application de référence
- Un département peut avoir 1 ou N employés
- Un employé est lié à un et un seul département



# : indique un identifiant  
\* : attribut obligatoire  
○ : attribut facultatif

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Classes à importer

```
package tpjava;

import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.DBObject;
import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DB;
import org.bson.Document;
import java.util.Arrays;
import java.util.List;
import com.mongodb.client.FindIterable;
import java.util.Iterator;
import java.util.ArrayList;
import com.mongodb.client.result.UpdateResult;
import com.mongodb.client.model.UpdateOptions;
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Définition de la classe et Constructeur

```
public class Dept {
 private MongoDB database;
 private String dbName="RH";
 private String hostName="localhost";
 private int port=27017;
 private String userName="urh";
 private String passWord="passUrh";
 private String DeptCollectionName="colDepts";
}
```

```
/**
 * Constructeur Dept. Dans ce constructeur sont effectuées les activités suivantes:
 * - Crédation d'une instance du client MongoClient
 * - Crédation d'une BD Mongo appelé RH
 * - Crédation d'un utilisateur appelé
 * - Chargement du pointeur vers la base RH
 */
Dept(){
 // FD1 : Creating a Mongo client
 MongoClient mongoClient = new MongoClient(hostName , port);

 // Creating Credentials RH : Ressources Humaines
 MongoCredential credential= MongoCredential.createCredential(userName, dbName,
 passWord.toCharArray());
 System.out.println("Connected to the database successfully");
 System.out.println("Credentials ::"+ credential);

 // Accessing the database
 database = mongoClient.getDatabase(dbName);
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Crédit et suppression de collections

```
/**
 * createCollectionDept : Cette fonction permet de créer une
 * collection de nom nomCollection.
 */

public void createCollectionDept(String nomCollection){
 //Creating a collection
 database.createCollection(nomCollection);
 System.out.println("Collection Depts created successfully");
}
```

```
/**
 * dropCollectionDept : Cette fonction permet de supprimer une
 * collection
 * de nom nomCollection.
 */

public void dropCollectionDept(String nomCollection){
 //Drop a collection
 MongoCollection<Document> colDepts=null;

 System.out.println("\n\n***** dans dropCollectionDept *****");
 System.out.println("!!!! Collection Dept : "+colDepts);

 colDepts=database.getCollection(nomCollection);
 System.out.println("!!!! Collection Dept : "+colDepts);

 if (colDepts==null)
 System.out.println("Collection inexiste");
 else {
 colDepts.drop();
 System.out.println("Collection colDepts removed successfully !!!");
 }
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode d'insertion d'un document

```
/**
insertOneDept : Cette fonction permet d'insérer un
Département dans une collection.
*/

public void insertOneDept(String nomCollection,
 Document dept){

 //Drop a collection
 MongoCollection<Document>
 colDepts=database.getCollection(nomCollection);

 colDepts.insertOne(dept);

 System.out.println("Document inserted successfully");
}
```

```
/**
testInsertOneDept : Cette fonction permet de tester
la méthode insertOneDept.
*/

public void testInsertOneDept(){

 Document dept = new Document("_id", 50)
 .append("dname", "FORMATION")
 .append("loc", "Nice");

 this.insertOneDept(this.DeptCollectionName, dept);

 System.out.println("Document inserted successfully");
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ■ Méthode d'insertion de documents multiples

```
/**
 * insertManyDepts : Cette fonction permet d'insérer plusieurs Départements
 * dans une collection
 */

public void insertManyDepts(String nomCollection, List<Document> depts){

 //Drop a collection
 MongoCollection<Document>
 colDepts=database.getCollection(nomCollection);

 colDepts.insertMany(depts);

 System.out.println("Many Documents inserted successfully");
}
```

```
/**
 * testInsertManyDepts : Cette fonction permet de tester la fonction
 * insertManyDepts
 */

public void testInsertManyDepts(){

 List<Document> depts = Arrays.asList(
 new Document("_id", 10).append("dname",
 "ACCOUNTING").append("loc", "New York"),
 new Document("_id", 20).append("dname",
 "RESEARCH").append("loc", "Dallas"),
 new Document("_id", 30).append("dname", "SALES").append("loc",
 "Chicago"),
 new Document("_id", 40).append("dname",
 "OPERATIONS").append("loc", "Boston")
);

 this.insertManyDepts(this.DeptCollectionName, depts);
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de recherche d'un document identifié

```
/**
 * getDeptById : Cette fonction permet de rechercher un département dans une collection
 * connaissant son id.
 */
public void getDeptById(String nomCollection, Integer deptId){
 System.out.println("\n\n\n***** dans getDeptById *****");

 MongoCollection<Document> colDepts=database.getCollection(nomCollection);

 //BasicDBObject whereQuery = new BasicDBObject();
 Document whereQuery = new Document();

 whereQuery.put("_id", deptId);
 //DBCursor cursor = colDepts.find(whereQuery);
 FindIterable<Document> listDept=colDepts.find(whereQuery);

 // Getting the iterator
 Iterator it = listDept.iterator();
 while(it.hasNext()) {System.out.println(it.next());}
}
```

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();

 ...
 // Récupération des informations sur le département
 Nr 10
 dept.getDeptById(dept.DeptCollectionName, 10);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

\*\*\*\*\* dans getDeptById \*\*\*\*\*  
Document{{\_id=10, dname=ACCOUNTING, loc>New York}}

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de recherche conditionnelle avec tri et projection

```
/**
 * getDepts : Cette fonction permet de rechercher des départements dans une collection.
 * Le paramètre whereQuery : permet de passer des conditions de recherche
 * Le paramètre projectionFields : permet d'indiquer les champs à afficher
 * Le paramètre sortFields : permet d'indiquer les champs de tri.
 */
public void getDepts(String nomCollection,
Document whereQuery,
Document projectionFields,
Document sortFields){

 System.out.println("\n\n***** dans getDepts *****");

 MongoCollection<Document> colDepts=database.getCollection(nomCollection);

 FindIterable<Document> listDept=colDepts.find(whereQuery).sort(sortFields).projection(projectionFields);

 // Getting the iterator
 Iterator it = listDept.iterator();

 while(it.hasNext()) { System.out.println(it.next()); }
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de recherche conditionnelle avec tri et projection

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();

 ...
 // TD2 : Afficher tous les départements sans tri ni projection

 System.out.println("\n\nTD2 : ...");
 dept.getDepts(dept.DeptCollectionName,
 new Document(),
 new Document(),
 new Document());

 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TD2 : ...

```
***** dans getDepts *****
Document{{_id=50, dname=FORMATION, loc=Nice}}
Document{{_id=10, dname=ACCOUNTING, loc>New York}}
Document{{_id=20, dname=RESEARCH, loc=Dallas}}
Document{{_id=30, dname=SALES, loc=Chicago}}
Document{{_id=40, dname=OPERATIONS, loc=Boston}}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de recherche conditionnelle avec tri et projection

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();

 ...
 // TD3 : Afficher tous les Départements
 // Trié en ordre croissant sur le dname et la localité loc
 // Projeter sur _id, dname et loc

 System.out.println("\n\nTD3 : ...");
 dept.getDepts(dept.DeptCollectionName,
 new Document(),
 new Document("_id", 1).append("dname", 1).append("loc", 1),
 new Document("dname", 1).append("loc", 1)
);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TD3 : ...

\*\*\*\*\* dans getDepts \*\*\*\*\*  
Document{{\_id=10, dname=ACCOUNTING, loc>New York}}  
Document{{\_id=50, dname=FORMATION, loc=Nice}}  
Document{{\_id=40, dname=OPERATIONS, loc=Boston}}  
Document{{\_id=20, dname=RESEARCH, loc=Dallas}}  
Document{{\_id=30, dname=SALES, loc=Chicago}}

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de recherche conditionnelle avec tri et projection

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();
 ...
 // TD5 : Afficher tous les Départements
 // Dont le nom est "SALES" et la localité est "Chicago"
 // projection: tous les champs
 // tri : pas de tri

 System.out.println("\n\nTD5 : ...");

 dept.getDepts(dept.DeptCollectionName,
 new Document("dname", "SALES").append("loc", "Chicago"),
 new Document(),
 new Document()
);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TD5 : ...

\*\*\*\*\* dans getDepts \*\*\*\*\*  
Document{{\_id=30, dname=SALES, loc=Chicago}}

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de recherche conditionnelle avec tri et projection

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();
 ...
 // TD6: Afficher les Départements de nom : "FORMATION" ou "SALES"
 // Trier en ordre décroissant sur la localité.
 // Projeter sur dname, loc
 System.out.println("\n\nTD6 : ...");

 dept.getDepts(dept.DeptCollectionName,
 new Document("$or",
 Arrays.asList(new Document("dname","FORMATION"),
 new Document("dname","SALES"))),
 new Document("dname",1).append("loc", 1),
 new Document("loc", -1)
);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TD6 : ...

\*\*\*\*\* dans getDepts \*\*\*\*\*  
Document{{\_id=50, dname=FORMATION, loc=Nice}}  
Document{{\_id=30, dname=SALES, loc=Chicago}}

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de modification d'un ou plusieurs documents

```
/**
 * updateDepts : Cette fonction permet de modifier des départements dans une collection.
 * Le paramètre whereQuery : permet de passer des conditions de recherche
 * Le paramètre updateExpressions : permet d'indiquer les champs à modifier
 * Le paramètre UpdateOptions : permet d'indiquer les options de mise à jour :
 * .upsert : insère si le document n'existe pas
 */
public void updateDepts(String nomCollection,
Document whereQuery,
Document updateExpressions,
UpdateOptions updateOptions
{
 System.out.println("\n\n***** dans updateDepts *****");

 MongoCollection<Document> colDepts=database.getCollection(nomCollection);
 UpdateResult updateResult = colDepts.updateMany(whereQuery, updateExpressions);

 System.out.println("\nRésultat update :"
+"getUpdateId: "+updateResult
+"getMatchedCount : "+updateResult.getMatchedCount()
+"getModifiedCount : "+updateResult.getModifiedCount()
);
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de modification d'un ou plusieurs documents

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();
 ...
 // TD7 : modifier le département nr 30. Mettre sa localité à
 // "Saratoga"

 System.out.println("\n\nTD7 : ...");

 dept.updateDepts(dept.DeptCollectionName,
 new Document("_id", 30),
 new Document ("$set", new Document("loc", "Saratoga")),
 new UpdateOptions()
);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TD7 : ...

\*\*\*\*\* dans updateDepts \*\*\*\*\*

Résultat update : getUpdate id: AcknowledgedUpdateResult{matchedCount=1, modifiedCount=1, upsertedId=null} getMatchedCount : 1 getModifiedCount : 1

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthodes de suppression de documents

```
/**deleteDepts : Cette fonction permet de supprimer des départements dans une collection.
Le paramètre filters : permet de passer des conditions de recherche des employés à supprimer
*/
public void deleteDepts(String nomCollection, Document filters){
 System.out.println("\n\n\n***** dans deleteDepts *****");

 MongoCollection<Document> colDepts=database.getCollection(nomCollection);

 // Afficher les documents qui matchent avant suppression
 FindIterable<Document> listDept=colDepts.find(filters).sort(new Document("_id", 1));
 Iterator it = listDept.iterator(); // Getting the iterator
 while(it.hasNext()) { System.out.println(it.next()); }

 // Supprimer
 colDepts.deleteMany(filters);

 // Tenter d'afficher les documents après suppression
 listDept=colDepts.find(filters).sort(new Document("_id", 1));
 Iterator it = listDept.iterator(); // Getting the iterator
 while(it.hasNext()) { System.out.println(it.next()); }
}
```

# Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthodes de suppression de documents

```
public static void main(String args[]) {
 try{
 Dept dept = new Dept();
 ...
 // TD9 : Supprimer le département Nr 10
 System.out.println("\n\nTD9 : ...");
 dept.deleteDepts(dept.DeptCollectionName, new Document("_id", 10));
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

\*\*\*\*\* dans deleteDepts \*\*\*\*\*

```
Dans deleteDepts: avant suppression #####
Document{{_id=10, dname=ACCOUNTING, loc=Saratoga}}
```

```
Dans deleteDepts: Apres suppression
```

## Module M4.5, section 2, partie 1 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

## Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Plan

- Objectifs
- Classes à importer
- Définition de la classe et Constructeur
- Méthode d'insertion d'un document
- Méthode effectuant une jointure entre deux collections
- Méthode effectuant un group by
- Méthode de création d'indexes
- Méthode d'affichage des indexes

## Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

### ➤ Objectifs

- L'objectif ici est de créer une nouvelle classe appelée **Employe.java**. Cette classe contiendra entre autres les fonctions suivantes :
  - Méthode effectuant une jointure entre deux collections
  - Méthode effectuant un group by
  - Méthode de création d'indexes
  - Méthode d'affichage des indexes

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Classes à importer

```
package tpjava;

import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.DBObject;
import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DB;
import org.bson.Document;
import java.util.Arrays;
import java.util.List;
import com.mongodb.client.FindIterable;
import java.util.Iterator;
import java.util.ArrayList;
import com.mongodb.client.result.UpdateResult;
import com.mongodb.client.model.UpdateOptions;
import com.mongodb.client.model.Aggregates;
import com.mongodb.client.model.AggreateIterable;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Accumulators;
import com.mongodb.client.model.Indexes;
import com.mongodb.client.model.IndexOptions;
```

## Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

### ➤ Définition de la classe et Constructeur

```
public class Employe {
 private MongoDB database;
 private String dbName="RH";
 private String hostName="localhost";
 private int port=27017;
 private String userName="urh";
 private String passWord="passUrh";
 private String EmployeCollectionName="colEmployees";
```

```
Employe(){

 // Creating a Mongo client
 MongoClient mongoClient = new MongoClient(hostName , port);

 // Creating Credentials
 // RH : Ressources Humaines
 MongoCredential credential=MongoCredential.createCredential(
 userName, dbName, passWord.toCharArray());

 System.out.println("Connected to the database successfully");
 System.out.println("Credentials ::"+ credential);

 // Accessing the database
 database = mongoClient.getDatabase(dbName);
}
```

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode d'insertion d'un document

```
/**
 * insertOneEmploye : Cette fonction permet d'insérer un
 * employé dans une collection.
 */

public void insertOneEmploye(String nomCollection,
Document employe){

 //Drop a collection
 MongoCollection<Document> colEmployes=
 database.getCollection(nomCollection);

 colEmployes.insertOne(employe);

 System.out.println("Document inserted successfully");
}
```

```
/**
 * testInsertOneEmploye : Cette fonction permet de tester la méthode insertOneEmploye.
 */

public void testInsertOneEmploye(){
 Document employe = new Document("_id", 7950).append("ename", "ERZULIE")
 .append("prenoms", Arrays.asList("Marie", "Frida"))
 .append("adresse", new Document("numero", 6).append("rue", "Des Miracles")
 .append("ville", "Valbonne").append("codePostal", "06560"))
 .append("Pays", "France").append("job", "Salesman").append("mgr", null)
 .append("hiredate", "17-11-1981").append("sal", 2500)
 .append("comm", 7698).append("deptno", 30);

 this.insertOneEmploye(this.EmployeCollectionName, employe);

 System.out.println("Document inserted successfully");
}
```

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode effectuant une jointure entre deux collections

```
/**
joinLocalAndforeignCollections : Cette fonction permet d'effectuer une jointure
entre les collections dept et employe. Elle permet d'afficher les employes par
departement.
 ▪ Le parametre localCollectionName : nom de la 1ere collection a joindre
 ▪ Le parametre ForeignCollectionName : nom de la 2eme collection a joindre
 ▪ Le parametre localColJoinFieldName : champ de jointure de la 1 ere
 collection
 ▪ Le parametre foreigColJoinFieldName : champ de jointure de la 2 eme
 collection
 ▪ Le parametre fielterFieldsOnLocalCollection : Champ de restriction
 ▪ Le parametre namedJoinedElements : Nom des éléments de la
 foreignCollectionName qui matchent
*/
```

```
public void joinLocalAndforeignCollections(
 String localCollectionName,
 String foreignCollectionName,
 String localColJoinFieldName,
 String foreigColJoinFieldName,
 Document fielterFieldsOnLocalCollection,
 String namedJoinedElements
{
 AggregateIterable<Document> outPutColl;
 MongoCollection<Document> joinColl=
 database.getCollection(localCollectionName);
 System.out.println("\n\n***** dans joinLocalAndforeignCollections
FE13b *****");
 outPutColl= joinColl.aggregate(Arrays.asList(
 Aggregates.match(fielterFieldsOnLocalCollection),
 Aggregates.lookup(foreignCollectionName, localColJoinFieldName,
 foreigColJoinFieldName, namedJoinedElements)));

 for (Document colDoc : outPutColl){
 System.out.println(colDoc);
 }
}
```

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode effectuant une jointure entre deux collections

```
public static void main(String args[]) {
 try{
 Employe emp = new Employe();
 ...
 // TE4 : afficher les informations sur le département 10 et
 // et ses employés

 System.out.println("\n\nTE4 : ...");

 emp.joinLocalAndforeignCollections("colDepts","colEmployes", "_id", "deptno", new Document("_id", 10),
 "EmployesQuiMatchent");
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TE4 : ...

```
***** dans joinLocalAndforeignCollections FE13b *****
Document{{_id=10, dname=ACCOUNTING, loc=Saratoga, EmployesQuiMatchent=[Document{{_id=8000, ename=toto, prenoms=[lery, kingo], adresse=Document{{numero=5, rue=Traverse en Escalier, ville=Nice, codePostal=06160, Pays=France}}, job=President, mgr=null, hiredate=17-09-1981, sal=500, comm=null, deptno=10}], Document{{_id=7839, ename=KING, prenoms=[Leroi, Mokondji], adresse=Document{{numero=6, rue=Traverse en Escalier, ville=Valbonne, codePostal=06560, Pays=France}}, job=President, mgr=null, hiredate=17-11-1981, sal=5000, comm=null, deptno=10}], Document{{_id=7782, ename=CLARK, prenoms=[Jonson], adresse=Document{{numero=27, rue=Des Miracles, ville=Toulouse, codePostal=31000, Pays=France}}, job=Manager, mgr=7839, hiredate=9-06-1981, sal=2450, comm=null, deptno=10}], Document{{_id=7934, ename=MILLER, prenoms=[Rosa, Alison], adresse=Document{{numero=11, rue=Place des étoiles, ville=Biarritz, codePostal=64200, Pays=France}}, job=Clerk, mgr=7782, hiredate=23-01-1982, sal=1300, comm=null, deptno=10}]}}}}
```

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode effectuant un group by

```
/**
 * groupBy : Cette fonction permet d'effectuer des opérations
 * de groupes dans la collection de nom collectionName
 * ▪ Le paramètre groupOperator : L'opérateur de groupe doit
 * être l'agrégat "$group"
 * ▪ Le paramètre groupFields : documents contenant les
 * champs de regroupement, moyennes, sommes, etc.
 */
```

```
public void groupBy(String collectionName,
 String groupOperator, Document groupFields) {

 MongoCollection<Document> colEmployes =
 database.getCollection(collectionName);

 Document group = new Document(groupOperator, groupFields);

 AggregateIterable<Document> output =
 colEmployes.aggregate(Arrays.asList(group));

 for (Document document : output) {
 System.out.println(document);
 }
}
```

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode effectuant un group by

```
public static void main(String args[]) {
 try{
 Employe emp = new Employe();
 ...
 // TE5b: afficher les employes groupes par job
 System.out.println("\n\nTE5b : ...");

 emp.groupBy("colEmployes", "$group", new
 Document("_id", "$job").
 append("masseSalaire", new Document("$sum", "$sal")).
 append("moySalaire", new Document("$avg", "$sal")).
 append("countSalaire", new Document("$count", new
 Document())));
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TE5b : ...

\*\*\*\*\* dans groupBy \*\*\*\*\*  
Document{{\_id=Manager, masseSalaire=8560.0, moySalaire=2853.333333333335, countSalaire=3}}  
Document{{\_id=Analyst, masseSalaire=6000, moySalaire=3000.0, countSalaire=2}}  
Document{{\_id=Salesman, masseSalaire=6160.0, moySalaire=1540.0, countSalaire=4}}  
Document{{\_id=Clerk, masseSalaire=4245.0, moySalaire=1061.25, countSalaire=4}}  
Document{{\_id=President, masseSalaire=5500, moySalaire=2750.0, countSalaire=2}}

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode de création d'indexes

```
/**
 * createEmployeeIndexes : Cette fonction permet de créer un index
 * dans la collection.
 * ▪ Le paramètre localCollectionName : Nom de la collection
 * ▪ Le paramètre indexName : Nom de l'indexe
 * ▪ Le paramètre fieldNames : Nom des champs à indexer
 * ▪ Le paramètre isAscendingIndex : true si ascendant, false
 * sinon
 * ▪ Le paramètre indexUnique : true si index unique false sinon

 */
```

```
public void createEmployeeIndexes(String localCollectionName, String indexName,
List<String> fieldNames, boolean isAscendingIndex, boolean indexUnique){

 System.out.println("\n\n***** dans createEmployeeIndexes
*****");
 String returnedIndexName;

 MongoCollection<Document>
 colEmployees=database.getCollection(localCollectionName);
 IndexOptions indexOptions=new IndexOptions() ;

 if (indexName!= null)
 indexOptions.unique(indexUnique).name(indexName);
 else
 indexOptions.unique(indexUnique);

 if (isAscendingIndex==true)
 returnedIndexName=colEmployees.createIndex(Indexes.ascending(
 fieldNames), indexOptions);
 else
 returnedIndexName=colEmployees.createIndex(Indexes.descending(
 fieldNames), indexOptions);
 System.out.println("\n\nNom de l'index créé : "+ returnedIndexName);
}
```

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode d'affichage des indexes

```
public static void main(String args[]) {
 try{
 Employe emp = new Employe();
 ...
 // TE6: Creer sur le champ ename de la collection colEmployes
 // Son nom doit être : idx_colEmployes_ename
 // C'est un index ascendant: true et unique: false
 System.out.println("\n\nTE6 : ...");

 emp.createEmployeeIndexes(
 emp.EmployeCollectionName,
 "idx_"+emp.EmployeCollectionName+"_ename",
 new ArrayList<String> (Arrays.asList("ename")), true, false
);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

TE6 : ...

\*\*\*\*\* dans createEmployeeIndexes \*\*\*\*\*

Nom de l'index créé : idx\_colEmployes\_ename

# Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Méthode d'affichage des indexes

```
/**
 * getAllIndexesOfACollection : Cette fonction affiche les informations sur tous les indexes d'une collection.
 */
public void getAllIndexesOfACollection(
 String localCollectionName
{
 System.out.println("\n\n\n***** dans getAllIndexesOfACollection *****");

 MongoCollection<Document> colEmployes=database.getCollection(localCollectionName);
 ListIndexesIterable<Document> liIndexIter=colEmployes.listIndexes();

 Iterator it = liIndexIter.iterator();// Getting the iterator

 while(it.hasNext()) {System.out.println(it.next());}
}
```

## Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

### ➤ Méthode d'affichage des indexes

```
public static void main(String args[]) {
 try{
 Employe emp = new Employe();
 ...
 // TE7 : recuperer les informations sur tous les indexes
 // MongoDB d'une collection
 System.out.println("\n\nTE7 : ...");

 emp.getAllIndexesOfACollection(emp.EmployeCollectionName
);
 ...
 }catch(Exception e){e.printStackTrace();}
}
```

\*\*\*\*\* dans getAllIndexesOfACollection \*\*\*\*\*  
#### Dans getAllIndexesOfACollection: Liste des indexes de la collection colEmployes ####  
Document{{v=2, key=Document{{\_id=1}}, name=\_id\_}}  
Document{{v=2, key=Document{{ename=1}}, name=idx\_colEmployes\_ename}}

## Module M4.5, section 2, partie 2 : QUIZ

➤ **Question 1 :**

- A: ?

➤ **Question 2 :**

- A: ?

# Module M4.5 : INTRODUCTION A MONGODB ET SON API JAVA

## ➤ Bilan

- Module M4.5, section 1 : Prise en main de l'API Java
- Module M4.5, section 2, partie 1 : Mise en œuvre de l'API Java par l'exemple
- Module M4.5, section 2, partie 2 : Mise en œuvre de l'API Java par l'exemple

## ➤ Exercices

## Module M4.5 : INTRODUCTION A MONGODB ET SON API JAVA

### ➤ Webographie et Bibliographie

[1] Lien vers la documentation MongoDB

<https://docs.mongodb.com/manual/>

[2] MongoDB in Action, de Kyle Banker , Peter Bakkum, et al. | 7 avril 2016

[3] API Java MongoDB

<https://mongodb.github.io/mongo-java-driver/>

[4] Tutorial programmation Java avec MongoDB

[https://www.tutorialspoint.com/mongodb/mongodb\\_java.htm](https://www.tutorialspoint.com/mongodb/mongodb_java.htm)

[5] Lien vers le download des utilitaire MongoDB

[https://www.mongodb.com/try/download/database-tools?tck=docs\\_databasetools](https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools)

[6] API du Java MongoDB Driver 3.12

<https://mongodb.github.io/mongo-java-driver/3.12/javadoc/index.html>

[7] Migration de MongoDB vers PostGres

<https://eventuallycoding.com/2019/07/03/mongodb-vers-postgresql/>