COMPOSANTS ACCES BDD

JOUR 3 : MONGODB

PRESENTE PAR NATACHA DESSE

AFEC

DEROULE SEMAINE

- Composants d’accès SQL
- Validation des données
- **MongoDB**
- Composants d’accès aux données NoSQL

MONGODB

LES OBJECTIFS

Comprendre le fonctionnement et l'intérêt de MongoDB par rapport aux bases relationnelles traditionnelles

Manipuler des bases, collections et documents en utilisant les commandes principales de MongoDB.

Savoir modéliser des données efficacement dans un système orienté documents.

Utiliser les index et les pipelines d'agrégation pour optimiser et exploiter les données.

LE NO SQL

NUAGE DE MOTS

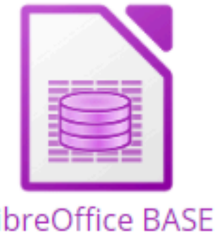
Que vous évoque le noSQL ?

https://www.menti.com/aldpp4773piz

RAPPEL BASE DE DONNEES

Une base de données est un système de stockage structuré permettant d'enregistrer, de retrouver et de manipuler des données de façon rapide et cohérente.

- Les **SGBDR** (Systèmes de Gestion de Bases de Données Relationnelles) dominent depuis les années 1980 : Oracle, MySQL, PostgreSQL, SQL Server.
- Modèle basé sur des tables, relations, jointures, ACID.

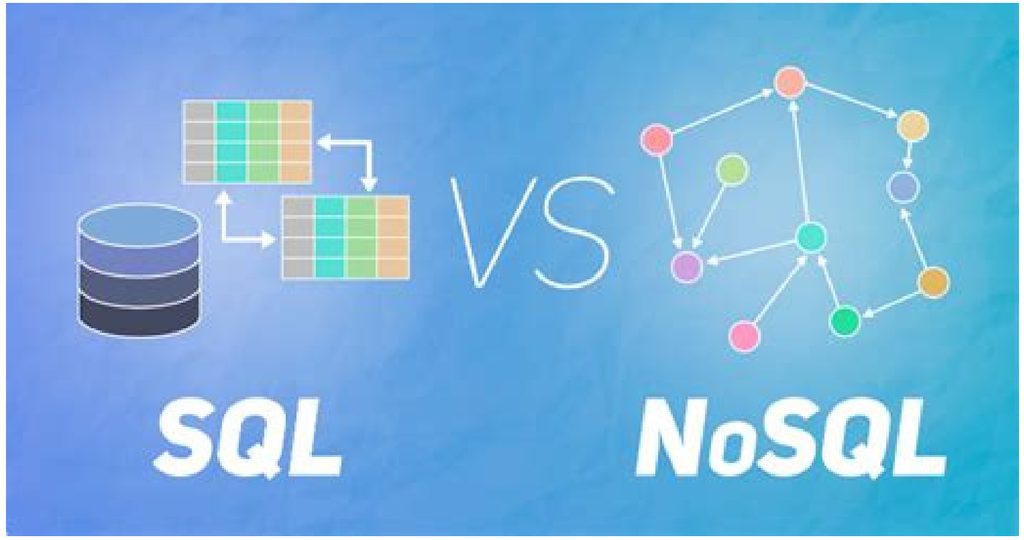


QU’EST CE QUE LE NOSQL ?

Le terme NoSQL (pour Not Only SQL) désigne une famille de bases de données non relationnelles, apparue à la fin des années 2000 pour répondre aux limites des SGBD classiques dans des contextes modernes : Big Data, applications web à fort trafic, cloud computing, etc.

Contrairement aux bases relationnelles, les bases NoSQL :

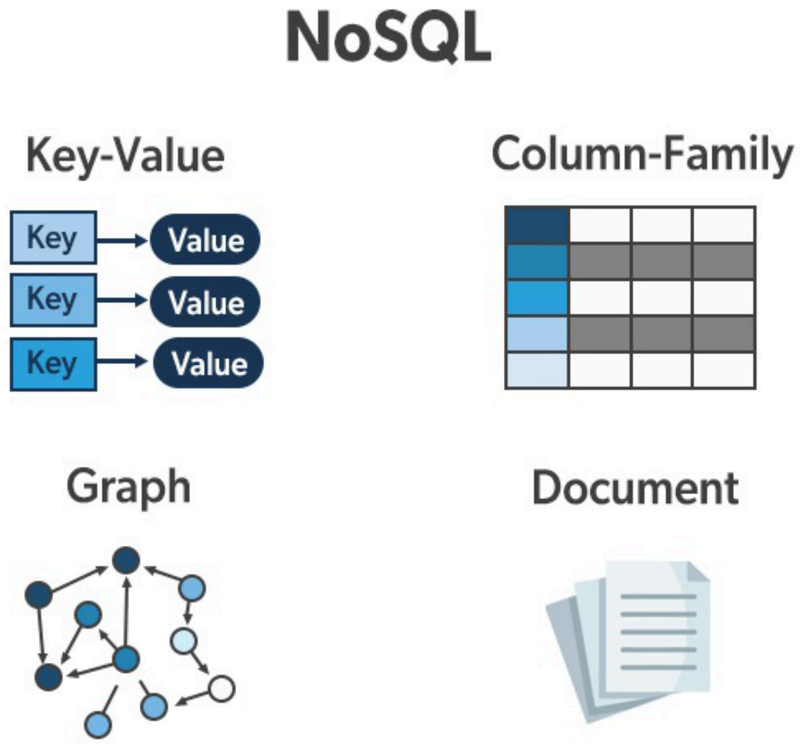
- ne reposent pas sur des tables et schémas fixes,
- ne nécessitent pas de jointures complexes,
- sont conçues pour scaler horizontalement (ajout de serveurs),
- et privilégient souvent la souplesse et la performance à la rigueur du modèle relationnel.



LES GRANDES FAMILLES DE NOSQL

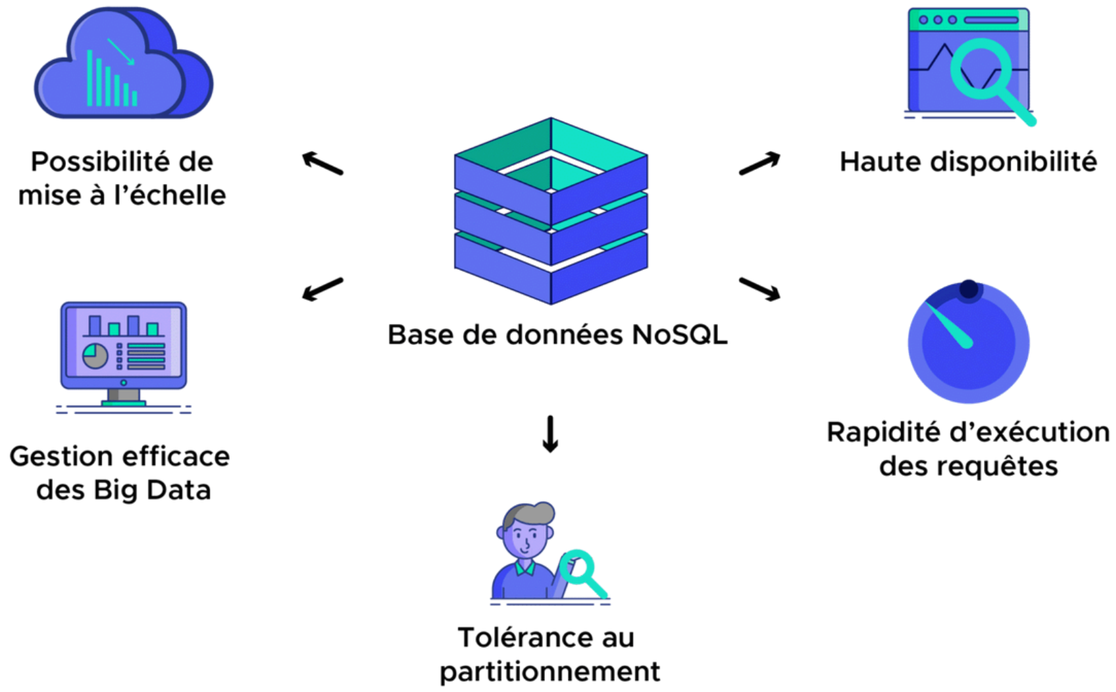
Il existe plusieurs grandes familles de bases NoSQL :

- Orientées documents (comme MongoDB) : stockent les données sous forme de documents JSON/BSON.
- Clé-valeur (comme Redis) : extrêmement rapides pour des recherches simples.
- Colonnes (comme Cassandra) : optimisées pour le Big Data.
- Graphes (comme Neo4j) : idéales pour les relations complexes entre entités (ex. réseaux sociaux).



LES AVANTAGES

Le NoSQL ne remplace pas les bases relationnelles, mais les complète : il s'agit de choisir l'outil adapté à la structure des données, aux besoins de performance et à l'évolution du projet



CAS D’UTILISATION

Les bases de données NoSQL sont particulièrement bien adaptées aux systèmes qui doivent gérer de gros volumes de données, des structures flexibles ou des charges hautement distribuées. Elles sont souvent utilisées dans les contextes suivants

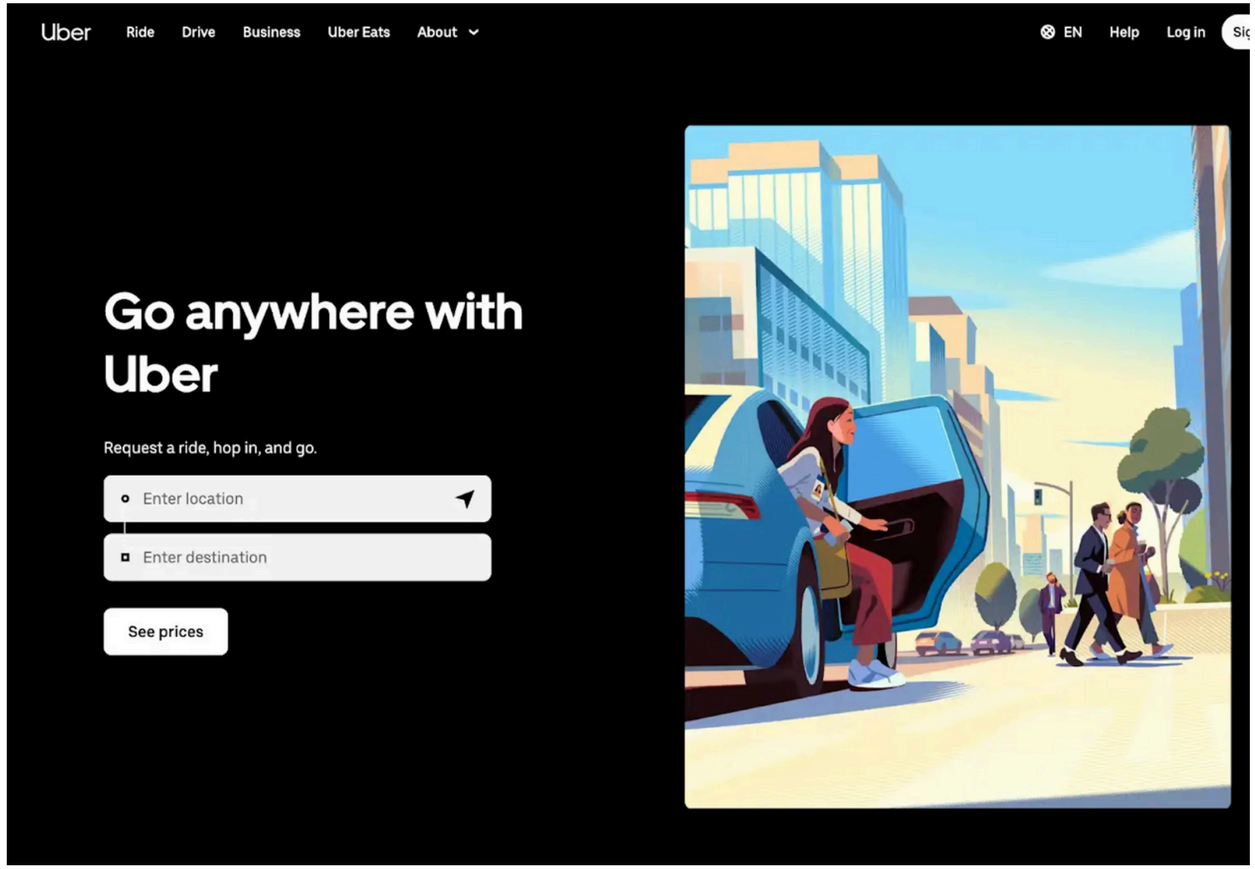
- 🌐 **Applications web** à **fort trafic** : plateformes e-commerce, réseaux sociaux, systèmes de gestion d’utilisateurs (ex. Amazon, Facebook, Netflix).
- 📱 **Applications mobiles** ou **SaaS** : les données changent fréquemment, le modèle doit rester flexible et évolutif.
- 📊 Analyse de **données massives** (Big Data) : les bases NoSQL permettent un stockage massif sans rigidité de schéma.
- 🛰️ Systèmes **IoT** ou **temps réel** : ingestion rapide de données hétérogènes et en grande quantité.
- 🔎 **Moteurs de recherche**, catalogues produits, recommandations : besoin d’indexation rapide, de structures imbriquées, et de performance.

QUI UTILISE DU NOSQL ?

Netflix utilise des bases de données NoSQL pour stocker et gérer :

- profil utilisateurs
- historique visionnage
- recommandation

QUI UTILISE DU NOSQL ?



Uber utilise des bases de données NoSQL pour gérer les **volumes massifs** de données générés par sa plateforme de covoiturage, notamment :

- les **profils** des conducteurs et des passagers,
- l'**historique** des trajets,
- les données de **géolocalisation** en temps réel.

Les bases NoSQL offrent la **scalabilité** et la **flexibilité** nécessaires pour absorber un trafic très élevé et s'adapter à des modèles de données en constante évolution.

QUI UTILISE DU NOSQL ?

Airbnb utilise des bases de données NoSQL pour stocker et gérer les données de sa plateforme de réservation, notamment :

- les **annonces** de logements,
- les **profils** des voyageurs,
- les **historiques** de **réservation**.

Airbnb doit traiter de grands volumes de données **non structurées** et, grâce à sa base NoSQL, les utilisateurs peuvent accéder **rapidement** et de manière fiable aux données, même à travers un réseau distribué.

INTRODUCTION MONGODB

UN PEU D'HISTOIRE

Contexte historique

MongoDB a été créé en **2007** par Dwight Merriman, Eliot Horowitz et Kevin Ryan, fondateurs de la société 10gen (devenue MongoDB Inc.).

Pourquoi ?

Les bases relationnelles montraient leurs limites face aux nouveaux usages du Web :

- Données non **structurées** ou **semi-structurées**.
- Volumes **massifs** (Big Data).
- Forte variabilité du **schéma** (ajouts, suppressions de colonnes).
- Besoin de **scalabilité horizontale**.

MongoDB a donc été conçu pour :

- Être orienté **documents JSON**.
- Offrir un modèle **souple** et **scalable**.
- Fonctionner de façon haute disponibilité dès le départ.

COMPARATIF MONGODB/BDD RELATIONNELLES

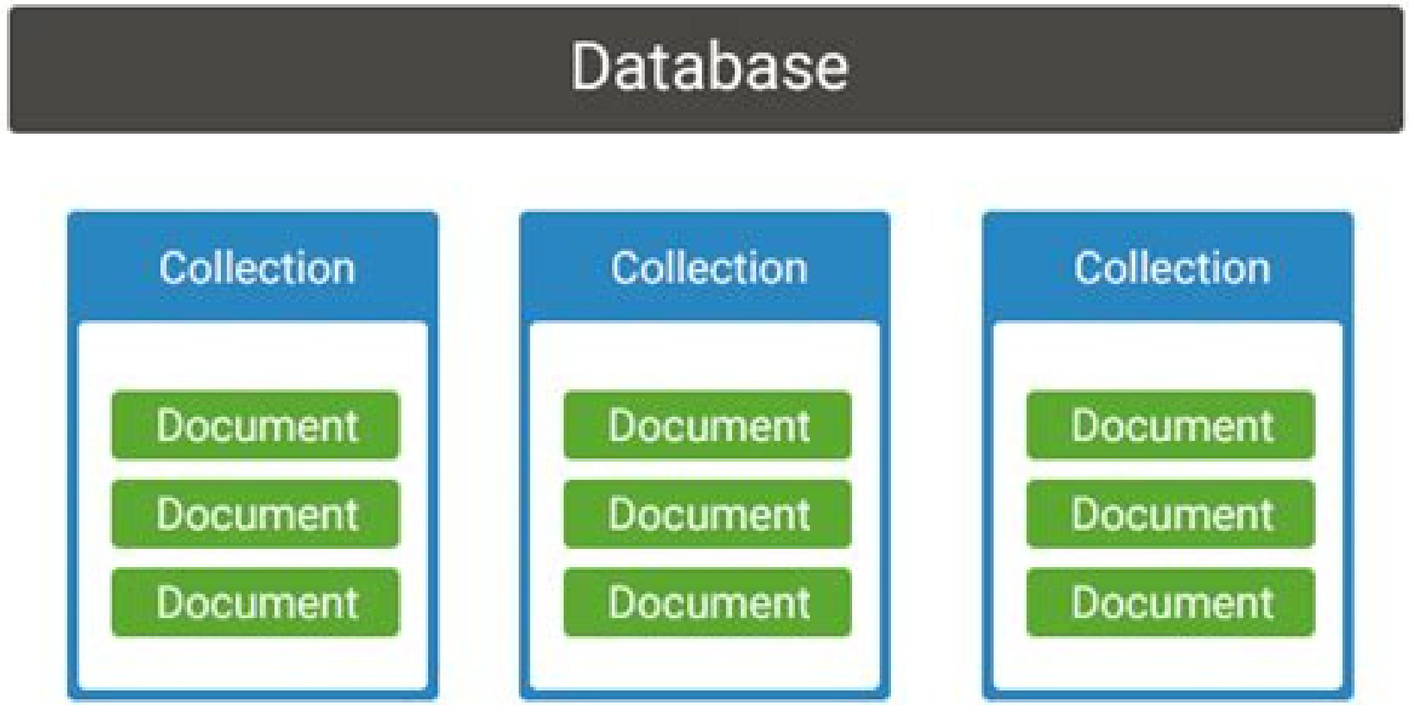
Caractéristique	SQL / SGBDR	MongoDB / NoSQL Documentaire
Structure	Tables, colonnes	Collections, documents
Langage	SQL	Syntaxe de type JSON
Schéma	Strict, figé	Flexible, dynamique
Relations	Clés étrangères, jointures	Embedding, référence manuelle
Transactions	ACID	BASE (eventuellement ACID)
Scalabilité	Verticale (scale up)	Horizontale (scale out)
Données hiérarchiques	Complexes, via jointures	Natives, via objets imbriqués

- **ACID** (Atomicité, Cohérence, Isolation, Durabilité) : rigueur, fiabilité.
- **BASE** (Basically Available, Soft state, Eventually consistent) : plus souple, tolérant aux retards de synchronisation dans des systèmes distribués.

MODELES DE DONNEES DE MONGODB

MongoDB utilise un modèle orienté document, avec :

- **Collections** : équivalent des tables.
- **Documents** : objets de type JSON (internement BSON).
- **Champs** : équivalent des colonnes, mais dynamiques.



L’ECOSYSTEME MONGODB

MongoDB Atlas
(Base de données cloud managée)

MongoDB Atlas est la plateforme cloud officielle proposée par MongoDB Inc. pour déployer, gérer et sécuriser des bases de données MongoDB.

- Disponible sur AWS, Azure et Google Cloud.
- Réplication automatique, sauvegardes intégrées, monitoring temps réel.
- Multi-régions, chiffrement des données, gestion fine des accès.
- Recommandée pour les projets modernes et les environnements de production.

➡️ C’est aujourd’hui la façon la plus courante d’utiliser MongoDB.

MongoDB Community Server
(Version open source classique)

- Le moteur MongoDB auto-hébergé, gratuit et open-source.
- Utilisable en local ou sur serveur privé (VM, Docker, etc.).
- Moins de services intégrés que dans Atlas (pas de sauvegardes auto, pas de scalabilité automatique).
- Idéal pour l'apprentissage, les POC ou les projets hébergés en interne

L’ECOSYSTEME MONGODB

MongoDB Compass
(Interface graphique)

- Outil GUI officiel pour interagir avec MongoDB.
- Permet d'explorer les collections, exécuter des requêtes, voir les index et visualiser les agrégations.
- Très utile pour les développeurs qui préfèrent une approche visuelle plutôt que ligne de commande.

MongoDB Shell
(mongosh)

- Shell moderne pour MongoDB, remplaçant le shell historique.
- Supporte la syntaxe JavaScript moderne, l'autocomplétion, les connexions sécurisées.
- Outil puissant pour les scripts d’administration, le debug, les requêtes manuelles.

L’ECOSYSTEME MONGODB

MongoDB Charts

- Outil de visualisation de données intégré à MongoDB Atlas.
- Permet de créer des dashboards directement à partir de collections MongoDB, sans ETL ni export.
- Requêtes dynamiques, graphiques interactifs, partage en lecture seule possible.

MongoDB Atlas Search

- Fonction de recherche plein texte intégrée, basée sur Apache Lucene.
- Permet d’ajouter des fonctionnalités de recherche avancée (autocomplétion, pertinence, facettes, etc.) sans infrastructure externe comme Elasticsearch.
- Déjà intégré à Atlas, sans besoin de synchronisation.

INSTALLATION

OUTILS QUE L’ON VA UTILISER

- **MongoDB Atlas** : On va se créer un compte sur mongoDB Atlas pour pouvoir générer un serveur pour notre base de données

- **MongoDB Compass** : Interface graphique pour pouvoir visualiser nos données.

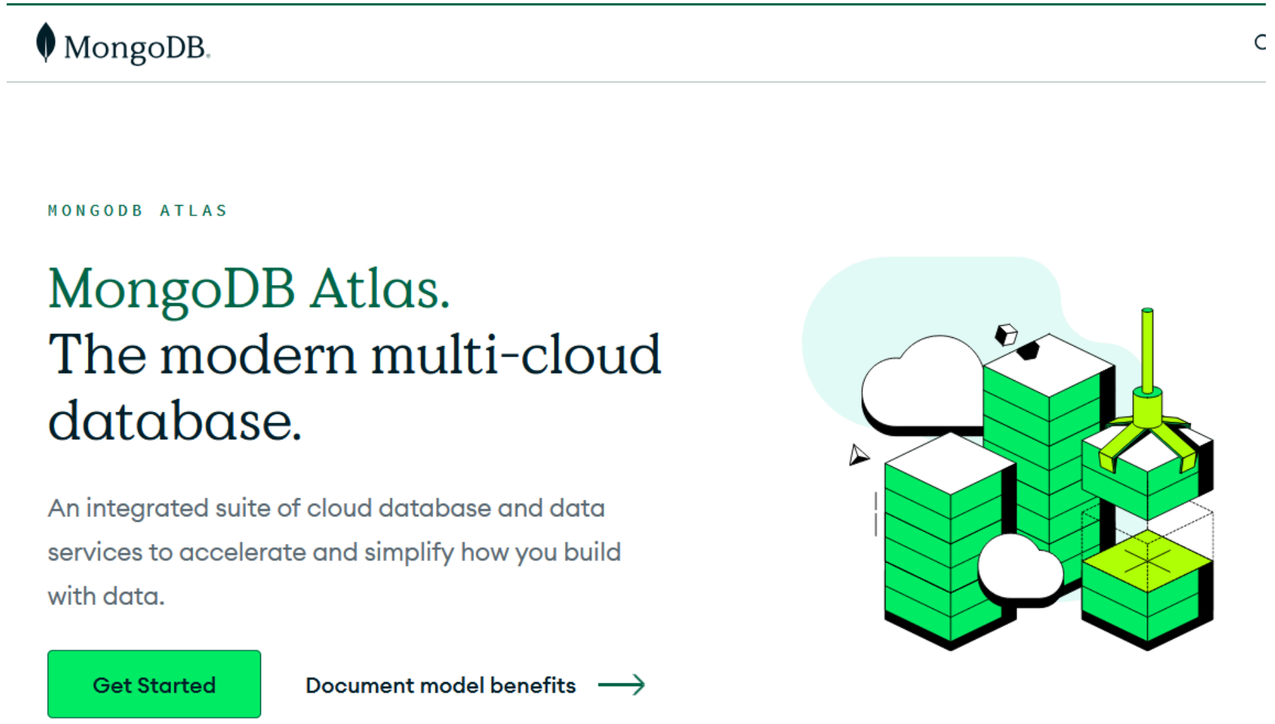
- **Mongosh Shell** : on va entrer nos requêtes sur le shell pour modifier les données

CREATION COMPTE MONGODB ATLAS

https://www.mongodb.com/atlas

Etape 1 : Se créer un compte

1.. Se créer un compte avec son email



Etape 2 : Créer un cluster

1. Une fois connecté clique sur Create cluster
2. Choisis l'option gratuite (shared)
3. Vérifie que la région est la bonne
4. Laisse le nom Cluster0 par défaut
5. Garde les champs par défaut.

Attention enregistre ton mot de passe quelque part !!

Etape 3 : Génère l'URI de connexion

1. Va sur ton cluster et clique sur Connect
2. Choisis Connect sith MongoDB Shell
3. Copie l'url de connexion :
4. mongodb+srv://etudiant:password123@cluster0.xxxx.mongodb.net/

INSTALLER MONGO COMPASS

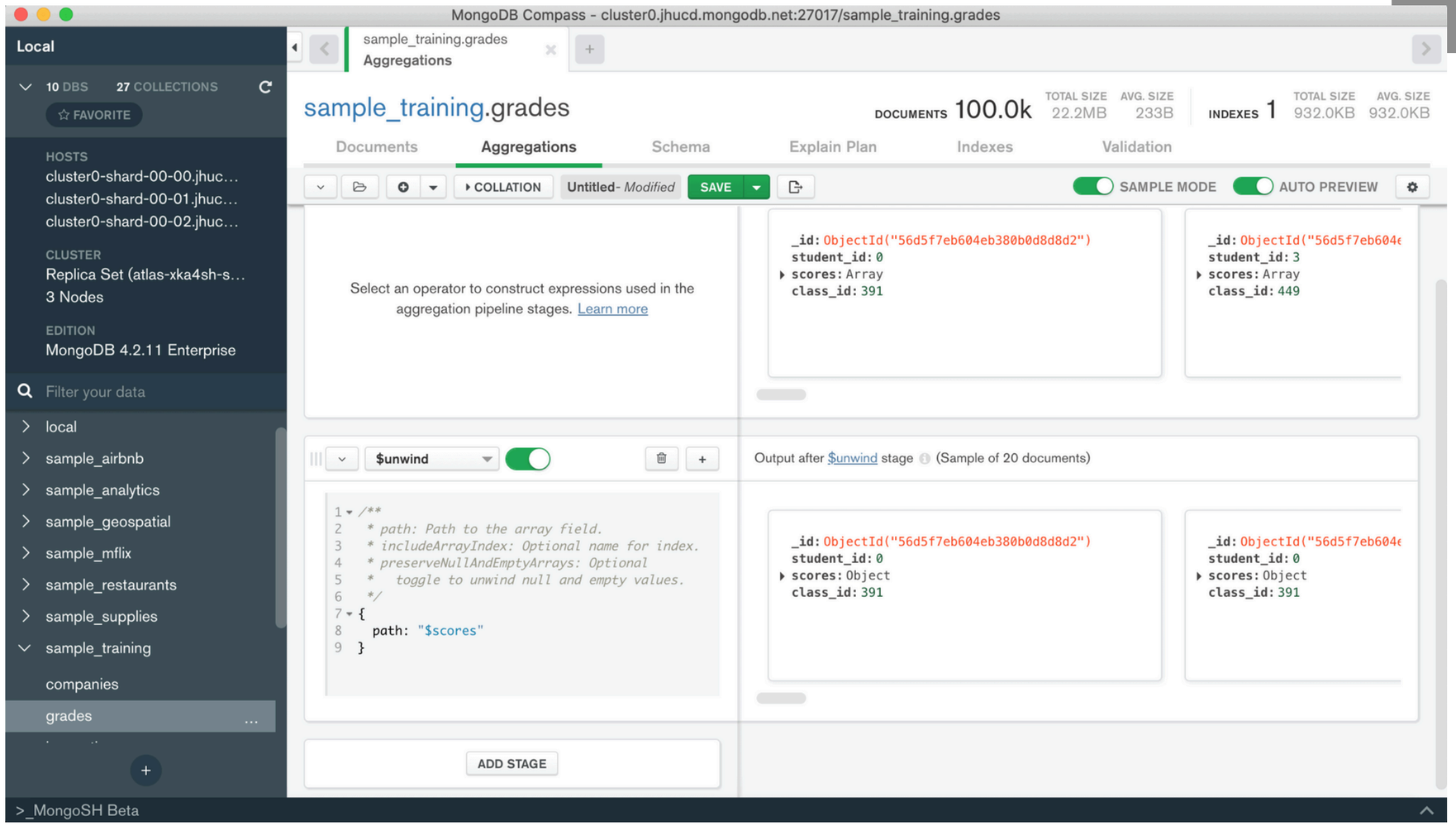
Lien officiel : https://www.mongodb.com/try/download/compass

Pour se connecter

1. Revenir sur Atlas
2. Connect sur cluster0
3. Choisir Compass
4. Faire la connexion

Attention à bien changer le mot de passe dans le lien de connexion.

Vous pouvez désormais parcourir votre database de façon plus visuelle



INSTALLER MONGO SHELL

Lien officiel : https://www.mongodb.com/try/download/shell

Pour se connecter

1. Revenir sur Atlas
2. Connect sur cluster0
3. Choisir Mongo Shell
4. Faire la connexion

Attention à bien changer le mot de passe dans le lien de connexion.

1. **Ouvrir cmd.exe**
2. **Taper mongosh -version**

```
C:\Users\Surface>mongodb+srv://natachadesse:HPVzXqJ5T76g7IMd@cluster0.apvkpnm.mongodb.net/
'mongodb+srv:' n’est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

C:\Users\Surface>mongosh "mongodb+srv://cluster0.apvkpnm.mongodb.net/" --apiVersion 1 --username natachadesse
Enter password: ***************
Current Mongosh Log ID: 684abc6a8b80c276ec73c5e7
Connecting to:          mongodb+srv://<credentials>@cluster0.apvkpnm.mongodb.net/?appName=mongosh+1.10.3
Using MongoDB:          8.0.9 (API Version 1)
Using Mongosh:          1.10.3

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

SYNTAXE DE BASE

db.collection.operation(filtre, miseAJour, options)

- **db.collection** : on cible la collection (exemple : db.pokemons)
- **operation** : ce qu'on veut faire, par exemple find(), updateOne(), etc.
- **filtre** (ou critère de recherche) : un objet entre accolades {}, qui sert à trouver les documents ciblés
- **miseAJour** : (pour les opérations de mise à jour) un objet qui décrit comment modifier le document
- **options** : paramètre optionnel pour préciser le comportement

```
db.pokemons.find({type: "Electric"})
```

```
db.pokemons.updateOne(
  { name: "Pikachu" },   // filtre : quel document on cible
  { $set: { level: 25 } }  // mise à jour : on crée ou modifie le champ 'level'
)
```

LES OPERATIONS CRUD DE BASE

Opération	Description	Commande MongoDB (mongosh)	Exemple
Create	Insérer un ou plusieurs documents	**insertOne**(document) / **insertMany**([documents])	db.pokemons.insertOne({name: "Pikachu", type: ["Electric"]})
Read	Lire un ou plusieurs documents	**find**(query) / **findOne**(query)	db.pokemons.find({type: "Electric"}) / db.pokemons.findOne({name: "Pikachu"})
Update	Mettre à jour un ou plusieurs docs	**updateOne**(filter, update) / **updateMany**(filter, update)	db.pokemons.updateOne({name: "Pikachu"}, {$set: {level: 25}})
Delete	Supprimer un ou plusieurs documents	**deleteOne**(filter) / **deleteMany**(filter)	db.pokemons.deleteOne({name: "Pikachu"})

LES OPERATEURS ARITHMETIQUES

Opérateur	Signification
$eq	Égal à
$ne	Différent de
$gt	Strictement supérieur à
$gte	Supérieur ou égal à
$lt	Strictement inférieur à
$lte	Inférieur ou égal à

```
db.pokemons.find({
  level: { $gte: 20, $lte: 30 }
});
```

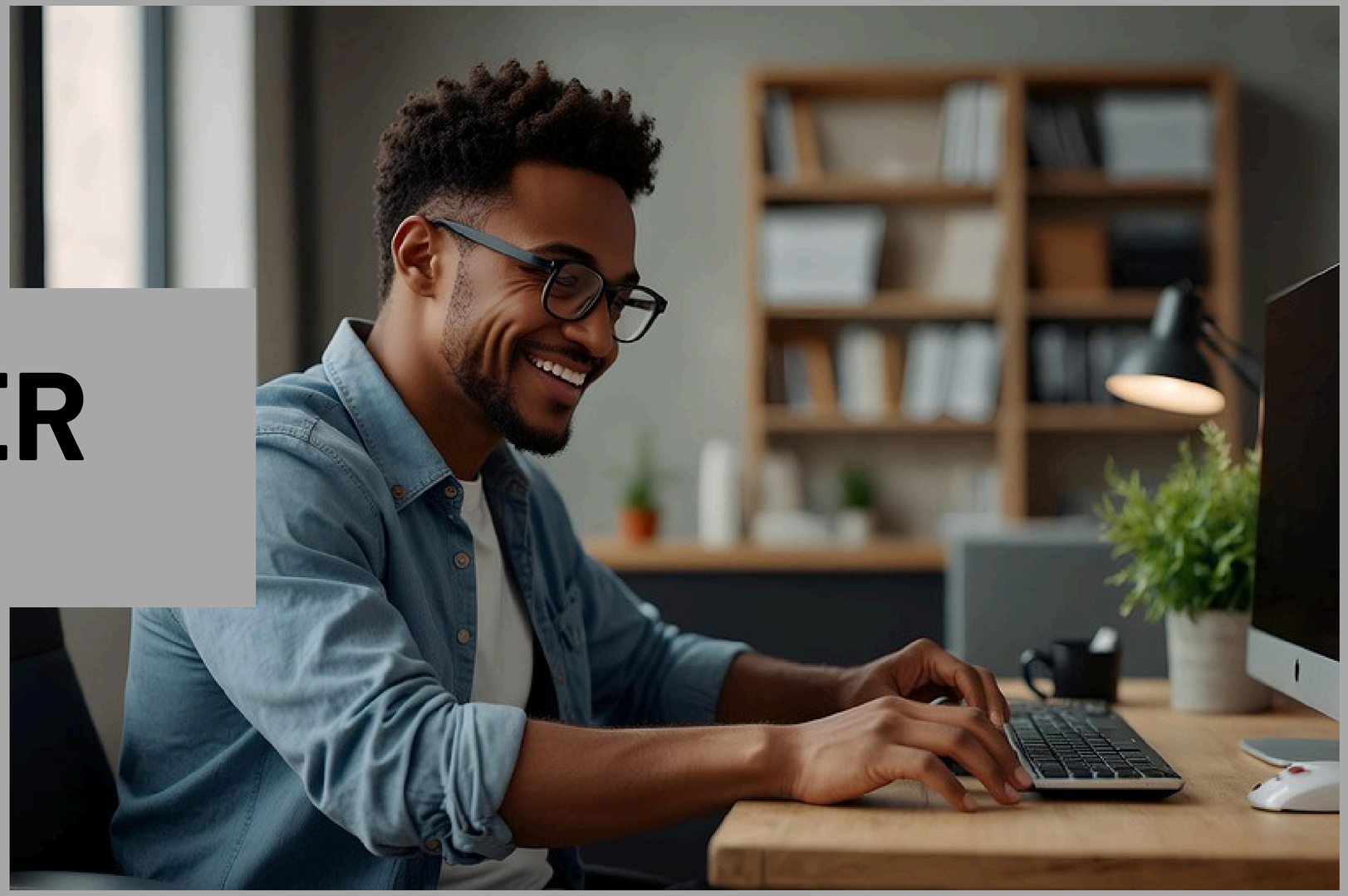
LES OPERATEURS LOGIQUES

Opérateur	Description	Exemple	Explication Résumée
$in	Appartient à une liste	{ type: { $in: ["Fire", "Electric"] } }	Type est Fire OU Electric
$nin	N'appartient pas à une liste	{ type: { $nin: ["Water", "Grass"] } }	Type différent de Water ET Grass
$and	Toutes les conditions sont vraies	{ $and: [{ type: "Water" }, { level: { $gte: 20 } }] }	Type Water ET niveau ≥ 20
$or	Au moins une condition est vraie	{ $or: [{ level: { $lt: 10 } }, { type: "Electric" }] }	Niveau < 10 OU type Electric
$exists	Champ présent ou non	{ attacks: { $exists: true } }	Tous les docs qui ont un champ attacks
$regex	Recherche textuelle (expression régulière)	{ name: { $regex: /^Pika/, $options: 'i' } }	Nom commençant par "Pika" (insensible à la casse)

```
db.pokemons.find({
  $and: [
    { type: "Water" },
    { level: { $gte: 20 } }
  ]
});
```

ATELIER

Des pokemons



DES POKEMONS

Bienvenue dans cet exercice pratique autour de MongoDB !

Nous allons créer ensemble une base de données nommée exercicesCRUD dédiée à l'univers des Pokémons.

Cette base contiendra trois collections distinctes :

- **pokemons** : qui regroupe toutes les informations sur les Pokémons (types, statistiques, évolutions...)
- **trainers** : qui liste les dresseurs avec leurs équipes, régions et badges
- **battles** : qui enregistre les combats entre dresseurs, leurs résultats et lieux

Votre mission sera de manipuler ces données avec les opérations de base de MongoDB (CRUD) :

- créer et insérer de nouveaux documents
- lire et filtrer des données selon différents critères
- modifier des champs précis dans les documents existants
- supprimer des documents ciblés

Cet exercice vous permettra de bien comprendre comment structurer, interroger et mettre à jour une base NoSQL en MongoDB. Prêt.e à attraper tous les Pokémons... et les requêtes ? C'est parti !