420-KBK – Veille technologique
Module Bases de données

Présentation de la portion BD

Objectif

- Explorer les bases de données non-relationnelles (NoSQL) les plus populaires
- Le but étant d'être en mesure de choisir une solution NoSQL adaptée aux besoins.

1. Introduction aux bases de données NoSQL
2. Quelques types de stockages de données
   - Gestion de documents
   - Clé/valeur
   - Lignes vers les colonnes
   - Orienté graphe

3. ACID vs BASE
4. Théorème de CAP
5. Présentation du travail

Historique

- Les données représentent une ressource organisationnelle et informationnelle cruciale que l'entreprise se doit de maîtriser. La majorité des organisations ne saurait réussir sans connaître les données exactes de leur entreprise et de leur environnement externe
- Pendant très longtemps, les systèmes d'information des entreprises structuraient leurs données sous forme de fichiers, chaque application utilisait donc son propre fichier. Nous ne pouvons pas citer tous les problèmes qui sont engendrés par ce type de stockage de l'information.
- Vers la fin des années 60 et le début des années 70, sont apparus les premiers SGBDs hiérarchiques et **réseaux**.
- Chaque type de SGBD Hiérarchique ou réseau a ses propres limites.

Historique

- Vers le milieu des années 70, nous avons vu naître les SGBDs relationnels, qui utilisent le modèle relationnel de Edgar Frank « Ted » Codd. Aujourd'hui les SGBD relationnels sont présents dans la majorité des entreprises et représente la plus grosse part du marché des SGBD. https://db-engines.com/en/ranking
- Le succès des SGBDR est sans doute parce qu'ils tirent leur fondement des mathématiques: La théorie des ensemble et l'algèbre relationnelle.
- Les géants comme Oracle, Microsoft et IBM ont une importante part du marché dans ces SGBDR.
- SQL est le langage de haut niveau pour interroger des DB relationnelles.

Introduction

- Not Only SQL propose de laisser de côté certaines contraintes des bases de données relationnelles. (dénormalisation, pas de FK)
- Dans ce contexte, il est plus intéressant d'avoir un langage de haut niveau pour exploiter les bases de données.
- Contrairement aux BD SQL, qui fonctionnent toutes sous le même principe, il existe plusieurs types de BD No SQL
    - Clé/Valeurs: Redis(VmWare) , SimpleDB (Amazon)
    - Des lignes vers les colonnes: le stockage des données est sous forme de colonne plutôt que de lignes. BigTable(Google), HBase
    - Gestion de documents: MongoDB, Cassandra.
    - Orienté Graph:Neo4J
- Les bases de données NoSQL sont des bases de données réparties: Sur plusieurs serveurs.

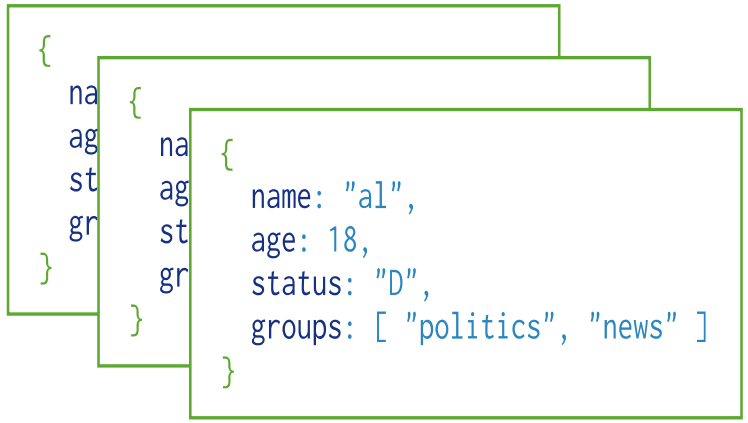
NoSQL: Documents

- Dans une BD orientée document, « un enregistrement » est un document, qui est une structure de données composée de paires de champs et de valeurs.
- Un document est encapsulé dans des accolades {...}, pouvant contenir des listes de clés/valeurs
- Les documents sont similaires aux objets JSON.
- Une valeur peut être un type scalaire (entier, nombre, texte, booléen, null), des listes de valeurs [...], ou des documents imbriqués
- Le but de ce stockage est de manipuler des documents contenant des informations avec une structure complexe (types, listes, imbrications)

```
{
  name: "sue",                    ← field:value
  age: 26,                        ← field:value
  status: "A",                    ← field:value
  groups: [ "news", "sports" ]    ← field:value
}
```

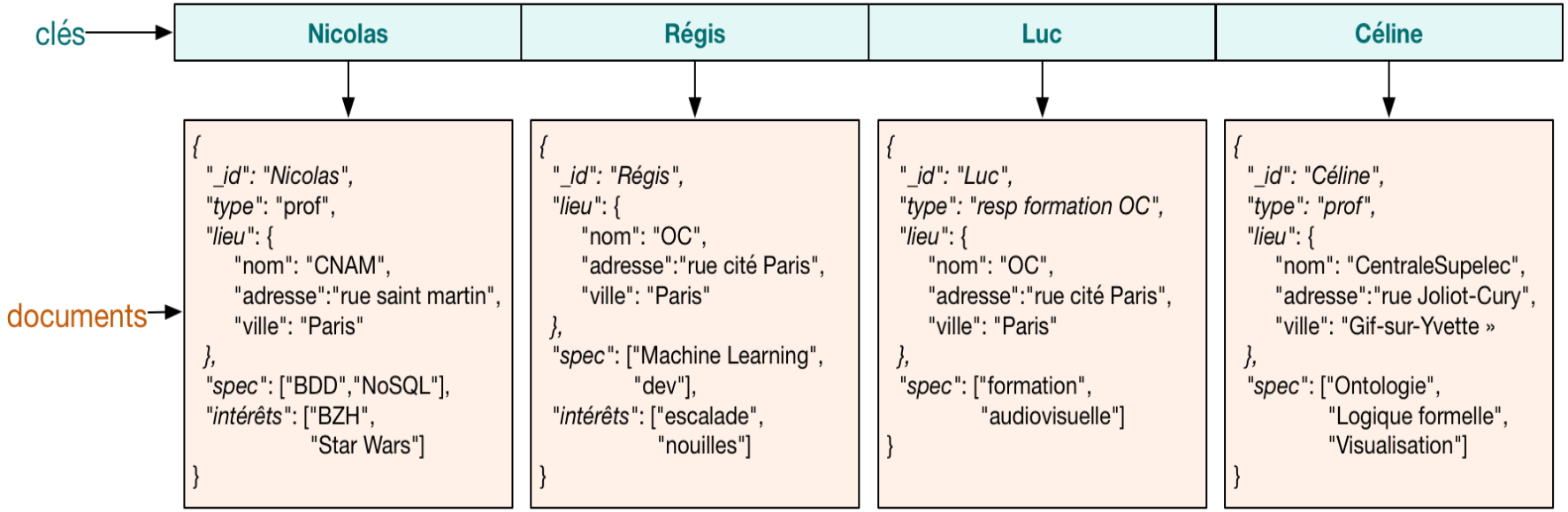
NoSQL: Documents

- Une collection est un ensemble de documents.
- C’est comme une table dans une base de données relationnelle.
- Les collections se trouvent dans une base de données



NoSQL: Documents

Source :https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462433-choisissez-votre-famille-nosql



NoSQL: Documents

Types d'applications:

- gestion de contenu (bibliothèques numériques,
- gestion des historiques d'utilisateurs sur réseaux sociaux
- collections de produits, dépôts de logiciels, collections multimédia, etc.),

BD connues:

MongoDB: ADP, Adobe, Bosch, Cisco, eBay, Expedia

CouchDB (Apache, Hadoop) : AT&T, Disney, PayPal

Amazon DocumentDB

NoSQL: Clé-Valeur

Le but de la famille clé-valeur est l'efficacité et la simplicité.
Un système clé-valeur agit comme une énorme table de hachage distribuée sur le réseau. Tout repose sur le couple Clé/Valeur

La base de données clé-valeur repose sur un tableau comportant seulement deux colonnes :

- La clé qui identifie la donnée de manière unique et permet de la gérer.

- La valeur qui contient n'importe quel type de données (différents formats)
  - des valeurs simples : chaînes de caractères, des entiers, etc.
  - des objets complexes : références à des fichiers dans la base de données, des tuples (groupe de valeurs), etc.

Le fait d'avoir n'importe quoi comme valeur, implique qu'il n'y ait ni schéma, ni structure pour le stockage. Il n'y a pas la possibilité d'exploiter ni de contrôler la structure des données. En ce sens, si vous savez ce que vous cherchez (la clé) et que vous manipulez directement la valeur il n’y a pas de problème à retrouver l’information

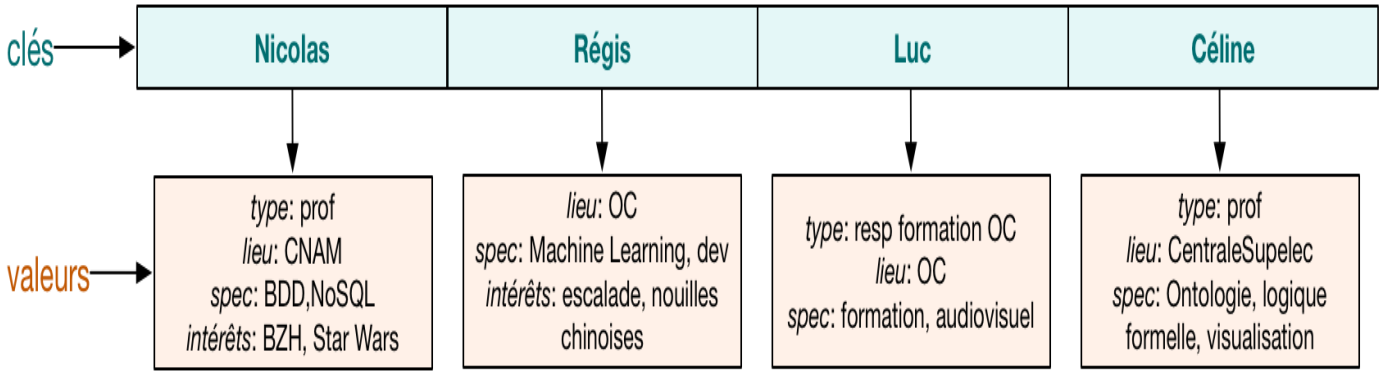
NoSQL: Clé-Valeur

Types d'applications:

- Détection d'infections d'un virus en temps réel
- Détection des fraudes en temps réel
- Fichiers de logs
- IoT (Internet des objets)
- Chat

BD Connues

- Redis (VMWare) : Trip Advisor, Nokia, Samsung, Docker
- Memcached : Wikipédia, Wordpress
- Azure Cosmos DB (Microsoft) :
- SimpleDB (Amazon)



NoSQL: Orienté vers la colonne

Contrairement au BD relationnelle où les données sont représentées dans des lignes, le stockage orienté colonne change ce paradigme en se focalisant sur chaque attribut et
en les distribuant.
Il est alors possible de focaliser les requêtes sur une ou plusieurs colonnes,
sans avoir à traiter les informations inutiles (les autres colonnes).

Modèle relationnel:

id	nom	Adresse	Téléphone	Courriel
1	Patoche	123, du ciel	450-2223598	Patoche@gmaol.com
2	Saturne	null	null	saturne@hotmail.com
3	Bien	578 mars	null	Bien@gmail.com
4	Henry	null	514-2256689	null

NoSQL: Orienté vers la colonne

Modèle orienté colonnes: seules les valeurs non nulles sont stockées

Id =1	**Id=2**	**Id =3**	**Id=4**
Nom: Patoche@gmaol.com Patoche	Nom: saturne	Nom: bien	Nom: Henry
Adresse :123, du ciel	Courriel: saturne@hotmail.com	Adresse:578 mars	Téléphone: 514-2256689
Téléphone: 450-2223598		Courriel :Bien@gmail.com	
Courriel Patoche@gmaol.com			

NoSQL: orienté vers la colonne.

Types d'applications:

- Cette solution est très adaptée pour effectuer des traitements sur des colonnes comme les agrégats (comptage, moyennes, co-occurences...).
- Gros calculs analytiques : Analyse des données

BD Connues

- Elasticsearch
- BigTable *de Google*
- HBase
- Cassandra (Facebook -> Apache) : NY Times, eBay

Cassandra Query Language (CQL) est le langage d'interrogation pour Cassandra

NoSQL: Graphe

Les structures graphiques ont pu être représentées dans des bases de données de modèles de réseaux à partir de la fin des années **1960**. (le NoSQL, n'est pas nouveau) .Dès le milieu des années 1980, les graphiques étiquetés pouvaient être représentées dans des graphic databases, comme le modèle logique de données.

Il existe plusieurs types de graphiques qui peuvent être classés par catégories:

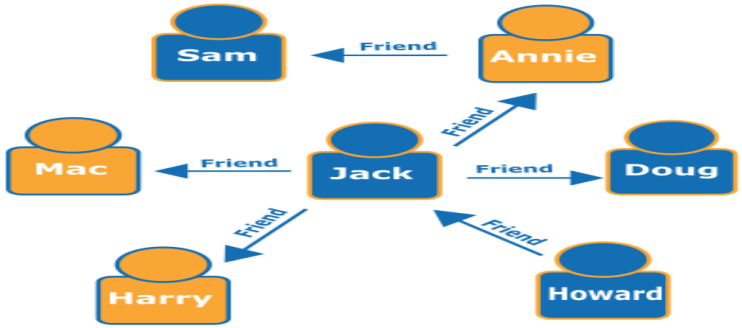
- Graphique social : il s'agit des liens entre les personnes ; par exemple, Facebook
- Graphique d'intention : il traite du raisonnement et de la motivation. (détection de Fraude)
- Graphique de consommation : également appelé "graphique de paiement", le graphique de consommation est très utilisé dans le secteur du commerce de détail pour suivre la consommation des clients individuels.
- Graphique des intérêts : ce graphique représente les intérêts d'une personne et est souvent complété par un graphique social.

NoSQL: Graphe

Dans la base orientée graphe, les données stockées sont : les nœuds, les liens et des propriétés sur ces nœuds et ces liens. Les requêtes que l'on peut exprimer sont basées sur la gestion de chemins, de propagations, d'agrégations, voire de recommandations.

Les nœuds sont identifiés de manière unique. Ce type de stockage n’est pas évident
Les graphes suivants montrent un exemple de graphe pour un réseau social. Au vu des personnes (nœuds) et de leurs relations (périphéries), vous pouvez déterminer qui sont les « amis des amis » d'une certaine personne : par exemple, les amis des amis d'Howard.

Exemple de langage : Gremlin Query Language



https://aws.amazon.com/fr/nosql/graph/

NoSQL: Orienté Graphe

Types d’applications:

- Relations entre utilisateurs dans les réseaux sociaux (Twitter ou Instagram)
- Analyse du comportement d'achat des clients dans les magasins en ligne
- L'analyse des risques, la détection des fraudes et la recherche de pannes.

BD Connues

- Neo4j (open source)
- FlockDB (*Twitter)*
- OrientDB (*Appache*)
- Oracle NoSQL

NoSQL, Avantages et inconvénients

Avantages

- Permet de gérer rapidement des tonnes de données (grand volume à une vitesse rapide).
- Schémas dynamiques pour les données non structurées (évolutifs, n'a pas à être connu d'avance).
- Plusieurs façons de stocker des données.
- Moins coûteux (ajout de serveurs).

Inconvénients

- La cohérence des données n'est pas garantie.
- Pas de langage de requête abstrait partagé, donc un travail de programmation spécifique plus important.

ACID vs BASE

Quand utilise-t-on le SQL ?

- Les données doivent être structurées. L'organisation est connue (ou pourrait être connue) d'avance.
- **L'intégrité des données doit-être respectée.**
- Les transactions sont importantes. Le principe ACID est important
- Faire des requêtes complexes avec un langage de haut niveau (SQL)

- A: Atomicité
- C: Cohérence
- I: Isolation
- D: Durabilité.

Cette propriété ACID ne peut pas être garantie dans un contexte distribué comme les BD NoSQL.

Une base de données répartie sur plusieurs serveurs ne peut pas garantir simultanément la cohérence, la disponibilité et la tolérance au partitionnement.

Vous n'avez qu'à penser à une transaction de 6 opérations réparties sur 4 serveurs. Comment garantir l'atomicité, l'isolation et la cohérence d'une telle transaction ? (il faudrait une parfaite synchronisation entre les 4 serveurs)

ACID vs BASE

Quand utilise-t-on le NoSQL ?

- La structure de données n'est pas importante. Évolutive et pas connue d'avance
- Gestion de beaucoup de données structurées, et non structurée.
- BASE: (contrairement à ACID)
  - **B**a*sically* ***A****vailable* : quelle que soit la charge de la base de données, le système garantie la disponibilité des données.
  - **S***oft-state* : La base peut changer lors des mises à jour ou lors d'ajout/suppression de serveurs. La base NoSQL n'a pas à être cohérente à tout instant

Théorème de CAP

Le théorème **de Brewer** peut vous éclaire en stipulant qu'un système distribué (soit ici, une base de données répartie sur plusieurs serveurs ne peut pas garantir simultanément la cohérence, la disponibilité et la tolérance au partitionnement.

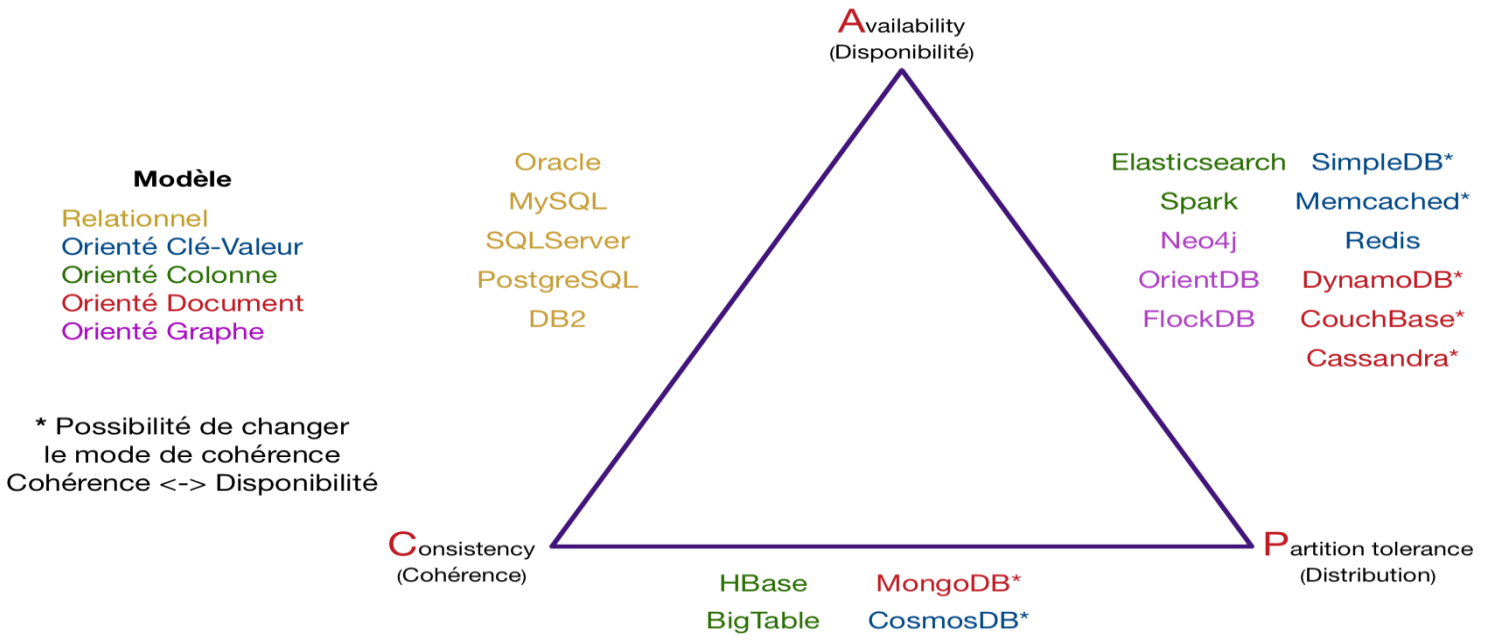
Théorème de Brewer dit "théorème de CAP, 2000:

Indique qu'il est impossible, pour un système distribué, de garantir en même temps les trois contraintes suivantes:

- **Cohérence (*C**onsistency*)**: Tous les noeuds du système voient les mêmes données au même moment.
- **Disponibilité (*A**vailability*)** : Toutes les requêtes reçoivent une réponse.
- **Tolérance au partitionnement (*P**artition Tolerance*)** : Aucune panne ne doit empêcher le système de répondre correctement (sauf une coupure complète du réseau).

Théorème de CAP

Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la cohérence, la disponibilité et la distribution.



Théorème de CAP

- Habituellement, un système de gestion de base de données relationnelles garantit la **cohérence** et la **disponibilité.**

- Il existe cependant un temps incompressible entre la mise à jour d'un noeud et sa synchronisation avec les autres. Ce temps peut avoir un grand impact sur un système très chargé.
- Les bases de données NoSQL tendent à privilégier la **disponibilité** et la **tolérance au partitionnement.**

    - Exemple : Il peut être préférable que deux personnes faisant la même recherche sur *Google* obtiennent des résultats différents que pas de réponses du tout (Facebook, Twitter, etc. utilisent le même principe).

Théorème de CAP

- Les bases de données NoSQL sont pratiques dans certaines situations :
  - Requiert une bonne tolérance au partitionnement;
  - Requiert une disponibilité à toute épreuve;
  - Gère un énorme trafic simultané sur un système distribué.

- *Les bases de données relationnelles peuvent aussi répondre à ces critères, mais souvent plus difficiles à mettre en place.*

Exemple d’utilisation NoSQL

Besoin : Le cas du coronavirus COVID-19 dans le monde, nécessite le traitement rapide des informations.

Nécessite une base de données qui peut :

1. stocker des grandes quantités d’informations en temps réel;
2. afficher les résultats en temps réel ;
3. faire des calculs d'analyse statiques rapides ;
4. fournir une carte (map) qui actualise les informations toutes les millisecondes

Exemple d’utilisation NoSQL

Solution : une combinaison des solutions de stockage NoSQL.

1. Clé-valeur pour les informations a accès rapide ;
2. Orienté lignes vers colonnes pour l'historique ;
3. Orienté document pour les informations personnelles ;
4. Graphes pour des analyses statistiques très avancées et efficaces.

SQL ou NoSQL ?

Les bases de données relationnelles privilégient la disponibilité des données et leur cohérence, mais un La propriété ACID est très importante

Les bases de données NoSQL tendent à privilégier la disponibilité et la tolérance au partitionnement.

Les systèmes NoSQL fonctionnent sur ce principe : Les données et l'état de la base de données seront **éventuellement cohérents** et consistants. L'important est que l'accès soit toujours permis.

Il peut être préférable que deux personnes faisant la même recherche sur Google obtiennent des résultats différents que pas de réponses du tout (Facebook, Twitter, etc. utilisent le même principe).

Ou encore ce n'est pas grave si votre page Facebook n'affiche pas exactement tout ce que vos amis viennent de publier, tant qu'éventuellement, vous êtes capable de voir ces publications?

SQL ou NoSQL ?

Finalement, la différence qui existe entre une base de données relationnelle et une base de données non relationnelle est la façon de stocker. L'une stocke les données dans des tables tandis que l'autre les stockent au format clé-valeur de manière à stocker davantage en termes de quantité.

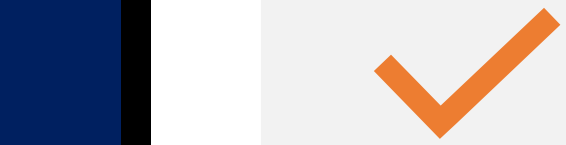
Les bases de données NoSQL ne sont pas en train de supplanter les bases relationnelles mais viennent les compléter, dans un contexte de croissance exponentielle de données. (Big Data)

Sources:

Sources:

- https://www.mongodb.com/docs/
- https://www.ibm.com/fr-fr/cloud/learn/cap-theorem
- https://www.oracle.com/fr/database/base-donnees-relationnelle-difference-non-relationnelle.html
- https://actualiteinformatique.fr/data/definition-nosql-not-only-sql-databas
- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462433-choisissez-votre-famille-nosql
- https://aws.amazon.com/fr/nosql/
- https://www.mongodb.com/docs/
- https://www.mongodb.com/blog/post/quick-start-c-sharp-and-mongodb-starting-and-setup
- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mongo-app?view=aspnetcore-6.0&tabs=visual-studio
- https://www.mongodb.com/docs/drivers/csharp/

BD NoSQL



Conclusion

Questions