UNIVERSITE BADJI MOKHTAR-ANNABA

DEPARTEMENT D'INFORMATIQUE
M2 SID

Structuration et Manipulation des Bases de Données Complexes

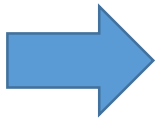
par

Dr. Samira LAGRINI

Année universitaire:2024/2025

Introduction

- Le web scraping permet d'extraire des données de diverses sources web, souvent sous des formats structurés comme CSV, JSON, ou XML.
- Ces formats facilitent une analyse initiale des données.
- Lorsque les données deviennent volumineuses, ou contiennent des relations complexes (des interactions sociales), les formats simples ne suffisent plus.



Pour quoi????

Introduction

- Les bases de données relationnelles (SQL) sont limitées par :
  - Leur schéma rigide
  - Capacité réduite à gérer des données non structurées ou semi-structurées.
  - Besoin des jointures coûteuses en ressources pour analyser des relations complexes,

- Les bases de données complexes **(NoSQL)** offrent plus de flexibilité permettant une analyse efficace des données extraites via le web scraping.

Flux de Données dans le Web Mining

Nettoyage des Données après Extraction

- **Suppression des doublons (**entrées dupliquées) : cela aide à réduire la taille des données et améliore la précision de l'analyse.
- **Gestion des valeurs manquantes** (NaN) : choisir de les supprimer ou de les remplacer par une valeur par défaut.
- **Correction des erreurs typographiques et les incohérences**: (ex. : "NY" et "New York" doivent être harmonisés).
- **Normalisation des données** : la mise dans un format cohérent (ex: les dates dans un format uniforme).

Nettoyage des Données après Extraction

Outils :

- **Python (Pandas)**

  Outil puissant pour manipuler les données tabulaires.

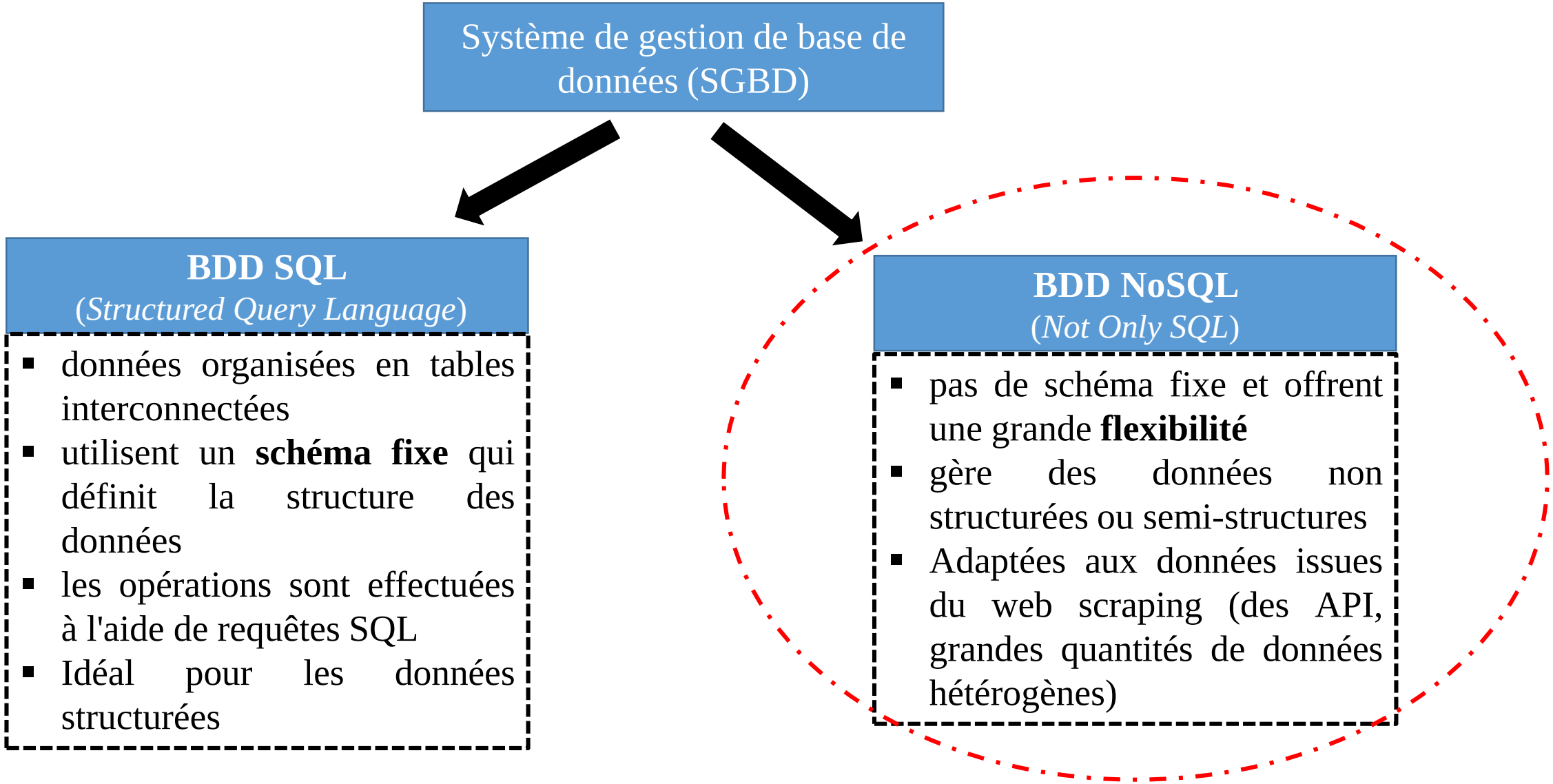
- **OpenRefine** :

  Logiciel open-source pour nettoyer des données complexes.

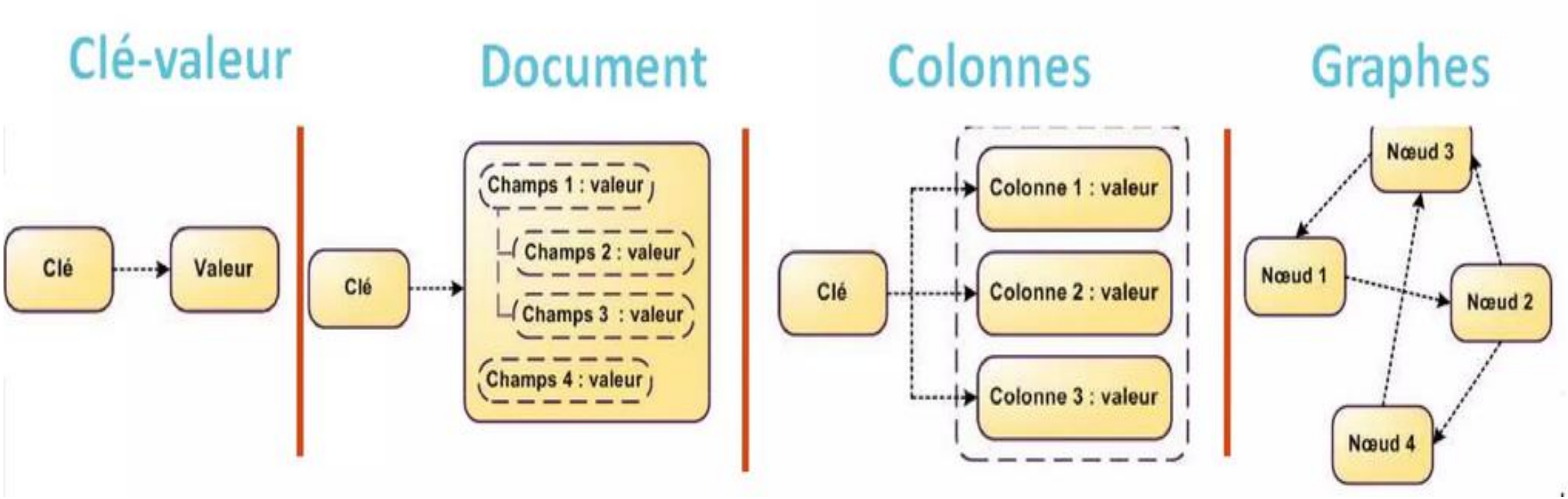
Exemple

```
import pandas as pd
import numpy as np
df = pd.read_csv('avis_produits.csv') // Étape 1 : Charger les données
# Étape 2 : Suppression des doublons
df = df.drop_duplicates()
# Étape 3 :  Suppression des lignes où les colonnes critiques (utilisateur, produit) sont manquantes
df = df.dropna(subset=['utilisateur', 'produit'])
# Remplacement des notes manquantes par la moyenne des notes
df['note'] = df['note'].fillna(df['note'].mean())
# Remplacement des dates manquantes par une date par défaut
df['date'] = df['date'].fillna('2024-01-01')
# Remplacement des villes manquantes par 'Inconnue'
df['ville'] = df['ville'].fillna('Inconnue')
# Étape 4 : Correction des erreurs typographiques
# Harmoniser les noms de villes ("NY" et "New York" doivent être identiques)
df['ville'] = df['ville'].replace({'NY': 'New York'})
# Étape 5 : Normalisation des données
df['date'] = pd.to_datetime(df['date'], errors='coerce', format='%Y-%m-%d')
# Étape 6 : Sauvegarder les données nettoyées dans un nouveau fichier CSV
df.to_csv('avis_produits_nettoyes.csv', index=False)
print("\nLes données nettoyées ont été sauvegardées dans 'avis_produits_nettoyes.csv'.")
```

Structuration des Données (SQL vs NoSQL)



Type Bases de Données NoSQL



Bases de Données Clé-Valeur (ex: Redis)

- Stocke des paires clé-valeur, permettant un accès rapide aux données.
- La valeur associée à une clé peut être une simple chaine de caractère ou encore un objet beaucoup plus complexe pouvant contenir une multitude d'information.

Cas d'Utilisation :

- le stockage temporaire des résultats de requêtes pour accélérer les performances d'une application web.
- Stockage des sessions utilisateur dans une application web.

Bases de Données Clé-Valeur (ex: Redis)

Avantages

- ❑ Très rapide pour les opérations de lecture et d'écriture.
- ❑ Simple et efficace pour le stockage temporaire des données.

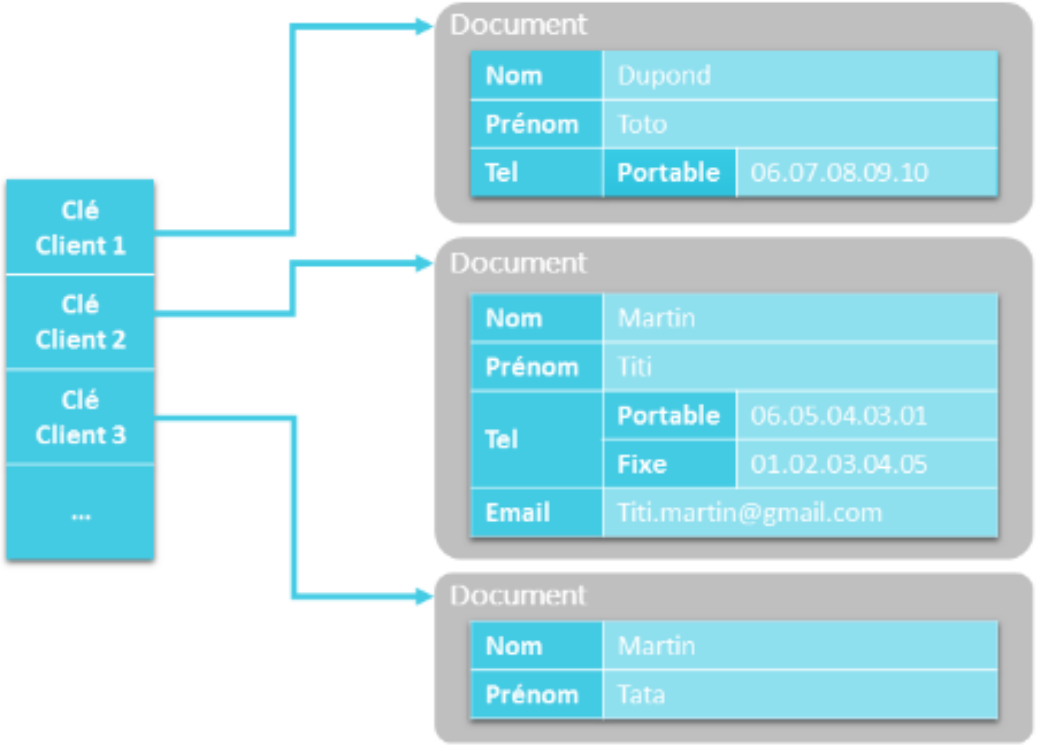
Inconvénients

- ❑ Limité pour les requêtes complexes et les relations entre données.

Bases de Données Orientées Document

- Stockent les données sous forme de documents (**JSON**, **BSON** (Binary JSON), ou **XML)**.
- Chaque document est un enregistrement autonome contenant des champs et des valeurs, ce qui permet une grande flexibilité.

Exemple : MongoDB



Bases de Données Orientées Document

Cas d'Utilisation

- Gestion de contenu web (CMS) où chaque document représente un article avec des champs (titre, auteur, contenu, tags).
- Applications de réseaux sociaux où les profils d'utilisateurs ont des champs variés.

Avantages

- Flexibilité des documents sans schéma fixe.
- Performances élevées pour les requêtes sur des documents JSON

Inconvénients

- Moins efficace pour les opérations complexes de jointure.

Bases de Données Orientées Colonne

- Stockent les données dans des colonnes au lieu de lignes.
- Les colonnes peuvent varier en nombre et en nom d'une ligne à l'autre, et peuvent changer dans le temps.
- Il n'y a pas d'espace mémoire consommé par des valeurs NULL, contrairement aux cas des BdD relationnelles

Product ID	Name	Price 1	Price 2	Price 3
1	Liquide vaisselle	**date**: 01/02/2020 **price**: 2.42€	**date**: 12/05/2020 **price**: 2.48€	
2	Shampooing	**date**: 08/05/2020 **price**: 1.56€	**date**: 12/09/2020 **price**: 1.12€	**date**: 19/09/2020 **price**: 1.56€
3	Fromage blanc	**date**: 12/05/2020 **price**: 2.02€		

Exemple :Apache Cassandra, HBase

Bases de Données Orientées Colonne

Cas d'Utilisation

- Idéales pour des entreprises qui analysent des données massives en temps réel.
- les séries temporelles (données générées régulièrement), et les traitements de Big Data

Avantages

- Excellente performance pour des requêtes sur des ensembles de données volumineux.

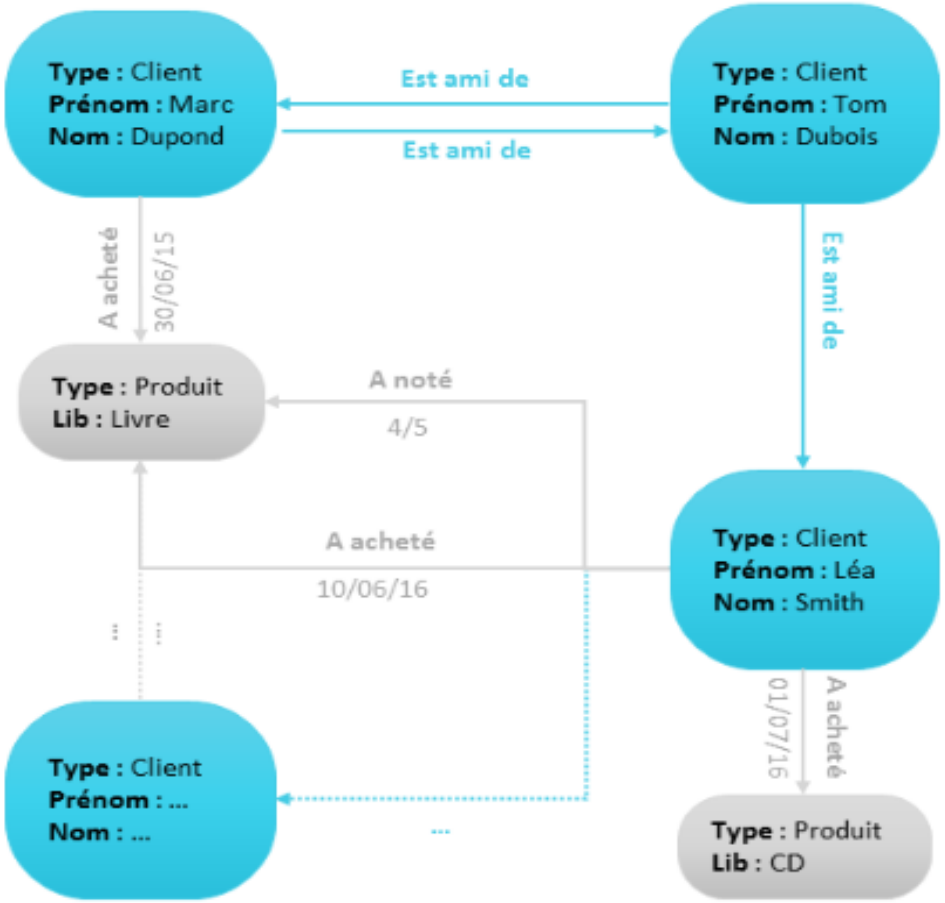
Inconvénients

- Complexité de modélisation des données.
- Moins efficace pour les requêtes nécessitant des jointures complexes.

Bases de Données Orientées Graphe

- Stockent des données sous forme de graphe (des nœuds pour les entités(ex. une personne, un produit, une ville) et des arcs pour les relations(ex. "AMI DE", "ACHÈTE", "SUIT")).
- Chaque nœud contient des propriétés (ou attributs), comme le nom, l'âge, ou la catégorie.
- Les arêtes peuvent également avoir des propriétés, comme la date ou l'intensité de la relation.

Exemple : Neo4j



Bases de Données Orientées Graphe

Cas d'Utilisation

- **Systèmes de Recommandation:**
  - Recommander des produits ou des services en fonction des relations et des préférences des utilisateurs (ex. "Les clients qui ont acheté ce produit ont aussi acheté...").
- **Analyse des Réseaux Sociaux**
  - Trouver des communautés, détecter des influenceurs, analyser les interactions entre utilisateurs.
- **Navigation et Réseaux de Transport**
  - Trouver les trajets les plus courts, modéliser des systèmes de navigation basés sur des graphes.

Bases de Données Orientées Graphe

Avantages

- ❑ Modélisation Naturelle des Relations Complexes
- ❑ Flexibilité du Schéma
  - Pas de schéma rigide, ce qui permet d'ajouter de nouveaux types de nœuds ou de relations sans modifier la structure existante.
  - Bien adaptées aux données hétérogènes et en évolution
- ❑ Adaptées aux Données Interconnectées

Inconvénients

- Nécessitent souvent une infrastructure spécialisée et peuvent être plus coûteuses à déployer et à maintenir par rapport aux bases de données relationnelles.

Comparaison entre les Types de BDD NoSQL

type	Cas d'utilisation	Exemple	Avantages	Inconvénients
Orientée Document	Données semi-structurées, API	MongoDB	Flexibilité, pas de schéma fixe	Moins efficace pour les jointures
Clé-Valeur	Mise en cache, sessions utilisateur	Redis, DynamoDB	Très rapide, simple	Limité pour les requêtes complexes
Orientée Colonne	Analyses de Big Data	Apache Cassandra	Scalabilité, performance	Complexité de modélisation
Orientée Graphe	Réseaux sociaux, recommandations	Neo4j	Modélisation des relations	Courbe d'apprentissage

Manipulation des Bases de Données Complexes

- La manipulation des données dans ces systèmes inclut un ensemble d'opérations :
- **Insertion**: ajouter de nouvelles données dans la base de données.
- **Mise à jour:** modifier des données existantes dans la base de données.
- **Suppression**: retirer des données de la base
- **interrogation des données:** récupérer des informations stockées dans la base de données en fonction de critères spécifiques

Conclusion

- les bases NoSQL offrent une flexibilité accrue pour gérer des données hétérogènes et semi-structurées.
- Le choix entre ses types de bases dépend également des besoins d'évolutivité et des caractéristiques des données
- Les bases de données MongoDB et Neo4j sont particulièrement adaptées aux données extraites via le web scraping, qui sont souvent non structurées et nécessitent une modélisation flexible des relations complexes

Travaux pratiques

TP

- Extraire des données web à l'aide d'un outil de scraping (BeautifulSoup ou Scrapy).
- Nettoyer et préparer les données extraites.
- Structurer et stocker les données dans une base de données NoSQL (MongoDB).
- Réaliser des requêtes simples pour manipuler et analyser les données stockées.

Etapes :

Étape 1 : scraper des informations à partir d'un site web de votre choix (par exemple, un site d'e-commerce pour collecter des informations sur les produits(nom du produit, prix, catégorie, note des utilisateurs).

Étape 2 : Nettoyage des Données

Étape 3 : Structuration et Stockage dans MongoDB

•Insérez les données nettoyées dans une base de données **MongoDB**.

•Assurez-vous d'avoir installé MongoDB sur votre machine locale ou utilisez MongoDB Atlas (service cloud).

Étape 4 : Manipulation et Requêtes dans MongoDB

• Effectuez des requêtes simples pour analyser les données stockées. Par exemple :

- •Trouver tous les produits dont le prix est supérieur à 50 $.
- •Compter le nombre de produits par catégorie.
- •…..


Merci de
votre
attention