Les bases de données NoSQL

Dr. H. EL BOUHISSI BRAHAMI

M2 GL

Introduction

À l'heure du Big Data, les bases de données relationnelles ne sont plus adaptées. Pour prendre en charge les immenses volumes de données, les stocker et les analyser, il est impératif de s'en remettre à de nouvelles solutions.

Une base de données NoSQL est une base de données "non relationnelle". Il est possible d'y stocker des données sous une forme non structurée, sans suivre de schéma fixe. Les jointures ne sont plus nécessaires, et le scaling est facilité.

Définition

Le NoSQL regroupe de nombreuses bases de données, récentes pour la plupart, qui se caractérisent par une logique de représentation de données non relationnelle et qui n'offrent donc pas une interface de requêtes en SQL.

NoSQL signifie Not OnlySQL et non pas No SQL, il s'agit de compléments aux SGBDR pour des besoins spécifiques et non de solutions de remplacement.

Les bases NoSQL se fondent sur une approche dite ***schema-less***, c'est à dire sans schéma logique défini a priori.

Définition

NoSQL est différent de SQL, il ne nécessite pas de schéma et n'a pas de relations de table, il est donc plus flexible.

Les bases de données NoSQL continuent d'augmenter en nombre d'utilisations, en particulier dans les implémentations de données volumineuses et les applications Web en temps réel.

Sa popularité n'a cessé d'augmenter au début de ce siècle millénaire, déclenchée par les besoins des entreprises basées sur le Web 2.0 et des applications gérées.

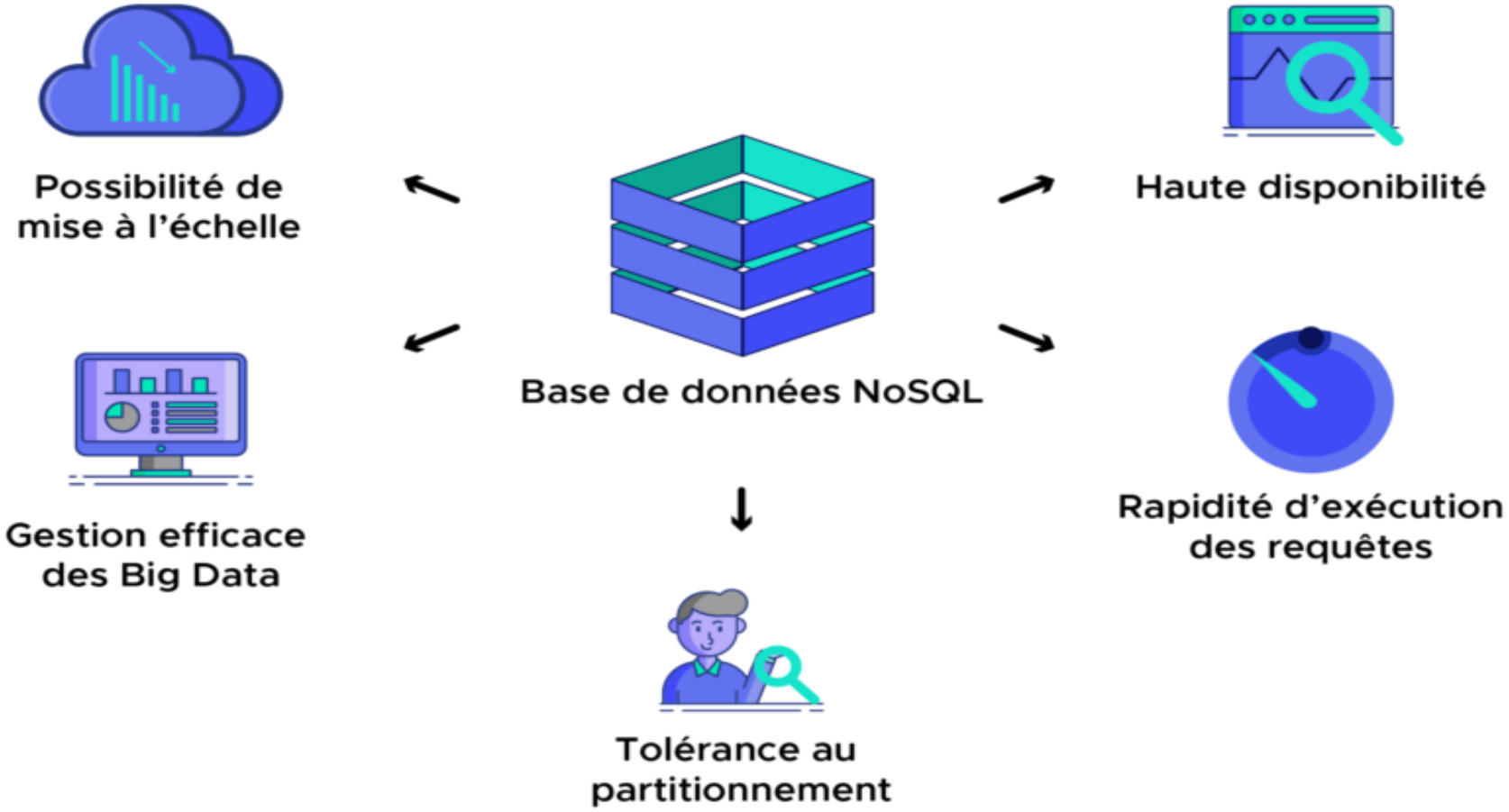
Historique

Le terme NoSQL a été publié pour la première fois par Carlo Strozzi en 1998 pour nommer la base de données qu'il développait « Strozzi NoSQL open-source relational database ». « NoREL », qui fait référence au terme « No Relational ».

Fin 2000, le développement de NoSQL a repris, dans le but de dépasser les limites de SQL, notamment en termes de scalabilité et de potentiel de collecte de données multi-structurées.

Début 2009, Johan Oskarsson, l'un des développeurs de Last.fm, a réintroduit le terme NoSQL lorsqu'il a organisé un événement pour discuter des « bases de données distribuées non relationnelles open source ».

Pourquoi le NoSQL?



De point de vue métier, utiliser un environnement Big Data et NoSQL fournit un avantage compétitif certain

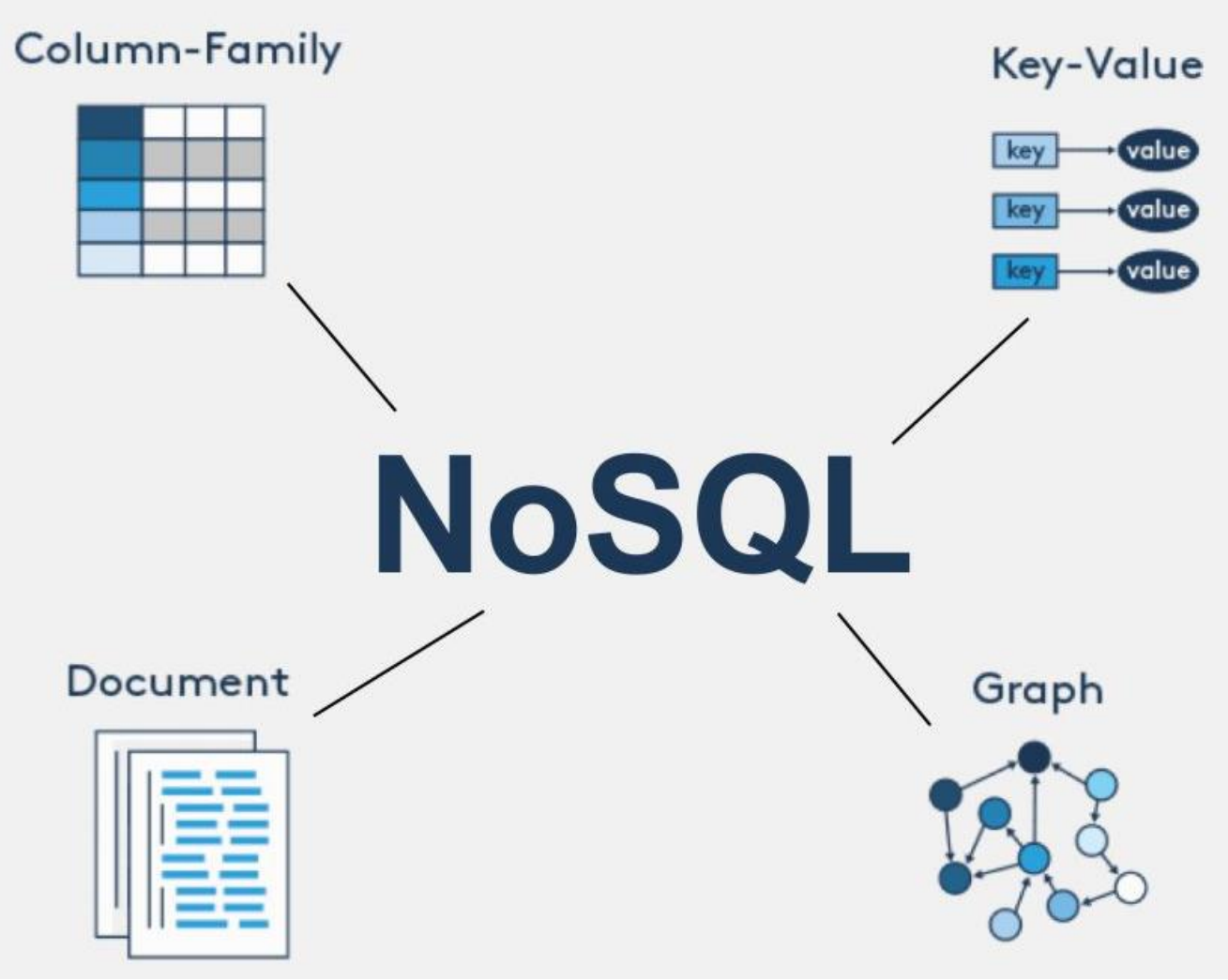
Acteurs du NoSQL

Les entreprises du WEB 2.0 avaient besoin de solutions technologiques plus adaptées à leurs besoins :

Développement des systèmes NoSQL propriétaires

- Facebook → Cassandra, Hbase
- Google → BigTable
- LinkedIn → Projet Voldemort
- Amazon → DynamoDB, SimpleDB
- Twitter → Cassandra

Taxonomie des BD NoSQL



Toutefois aucun de ces quatre types de bases de données ne permet de résoudre n'importe quel problème.

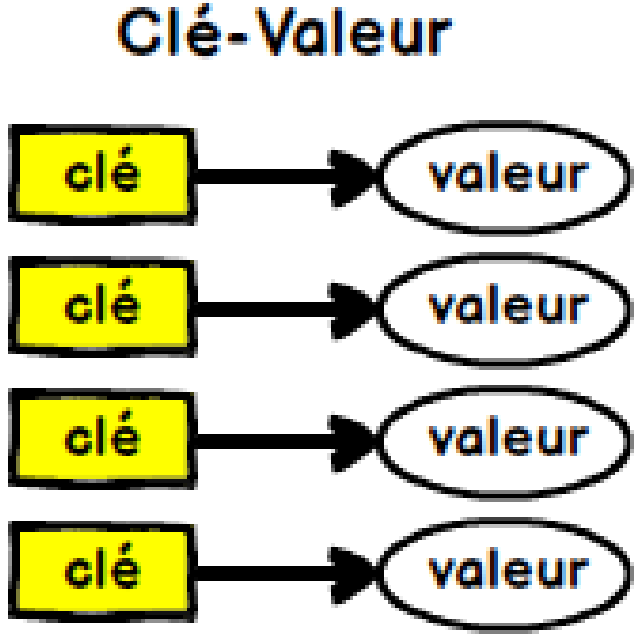
Il est nécessaire de **choisir la base de données adéquate** en fonction du cas d'usage.

1/ BD NoSQL Clé-Valeurs

Type basique

Principes: Représentation des données sous forme de clé/valeur. Les valeurs peuvent être de simple chaînes de caractères ou des objets complexes.

•Utilisation: dépôt de données avec besoins de requêtage simples (préférences d'utilisateur, . . .)



Exploitation basée sur 4 opérations : Accès par la clé

•Create: Création d'un objet.

•Read: Lecture d'un objet.

•Update: Mise à jour d'un objet.

•Delete: Suppression un objet.

2/ Base de données orientées document

Principes: c'est une variante des SGBD clé/valeur, où la valeur est un document de type XML ou JSON.

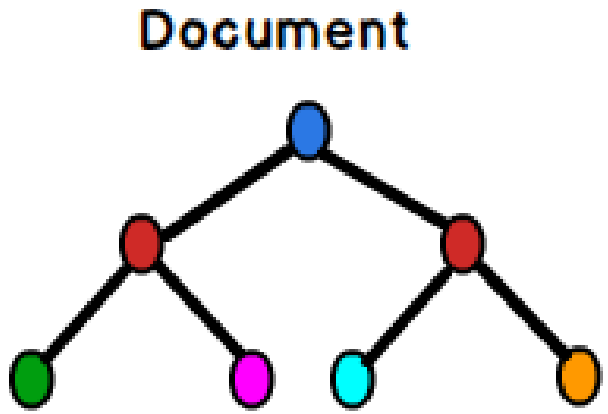
–Les documents ont une structure arborescente, ils sont composés de champs et des valeurs associées.

Ce type de SGBD permet d'effectuer des requêtes sur le contenu des documents. Ce type de base de données offre une flexibilité accrue.

Utilisé pour les systèmes CMS, les plateformes de blogging, ou les applications de e-commerce

Ne convient pas pour les transactions complexes nécessitant des opérations ou des requêtes multiples sur des structures agrégées variables.

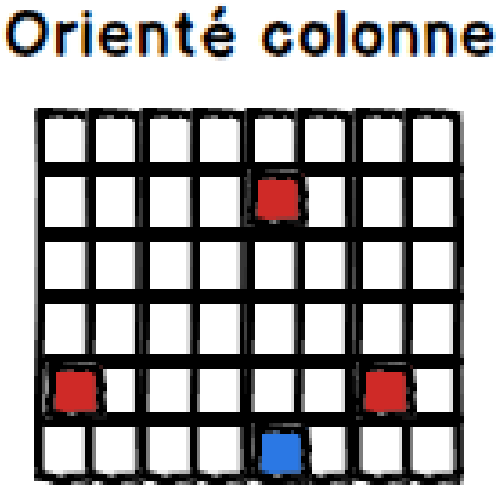
mongoDB



3/ Base de données orientées colonnes

Principes: Repose sur des colonnes. Proche du relationnel. Mais le stockage des donnés se fait par colonne et non par ligne.
Chaque colonne est traitée séparément, et les valeurs sont stockées de façon contigüe.
Ajout de colonnes facile et dynamique.
Possibilité de compression des données

•Utilisation: Cette catégorie de base de données offre de hautes performances pour les requêtes d'agrégation comme SUM, COUNT, AVG et MIN. Pour cause, les données sont déjà disponibles et prêtes dans une colonne.



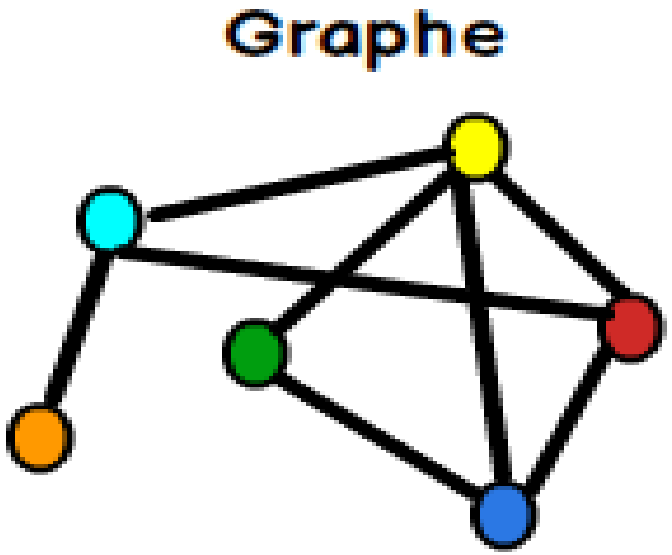
4/ Base de données orientées graphe

Principes:

–Les données sont représentées sous-forme de graphe : Des nœuds pour les entités, des arcs pour les relations entre les entités.

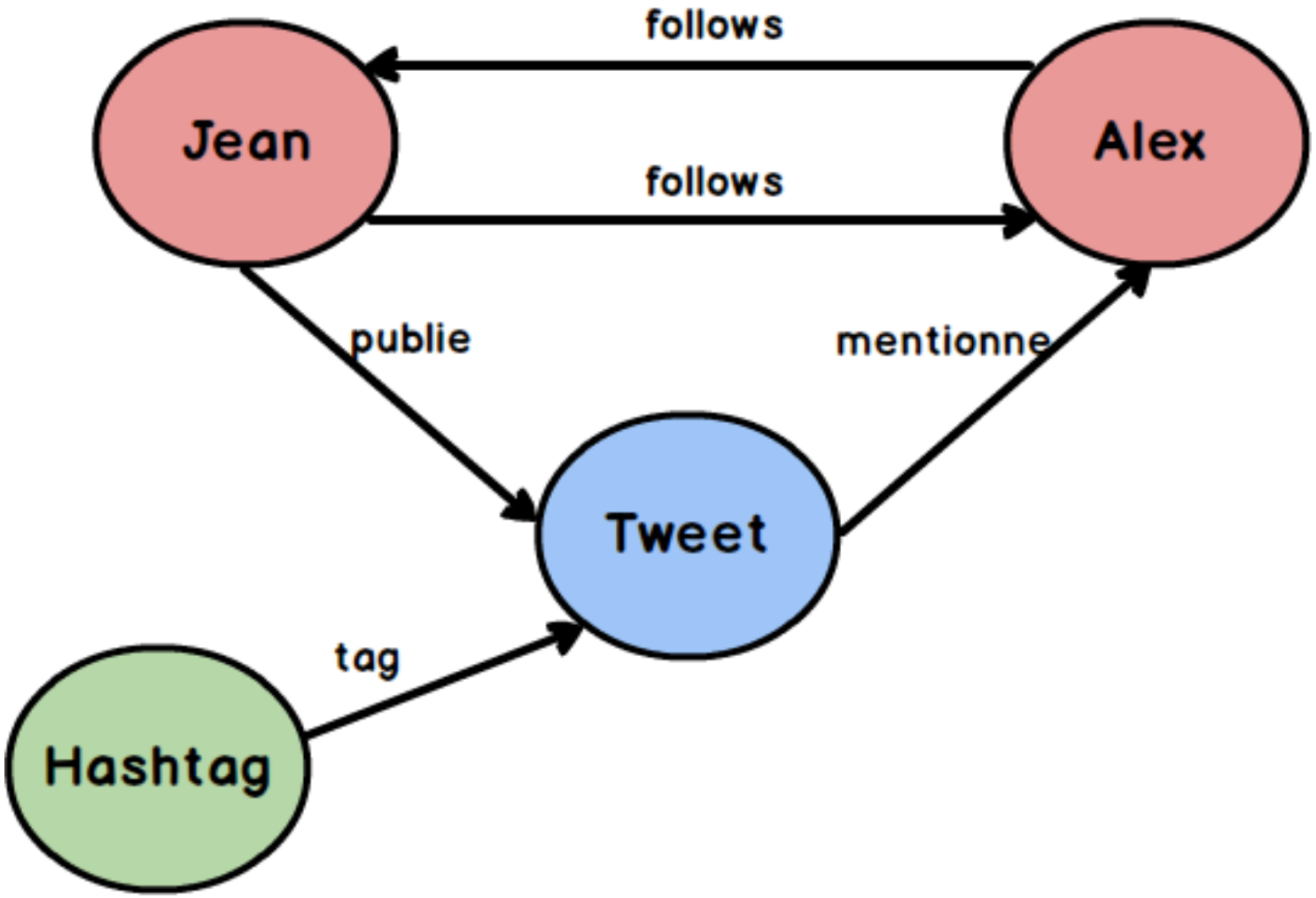
–Ce type de SGBD est adapté à la manipulation de données fortement connectées

- Utilisation: Systèmes de recommandations, Réseaux sociaux, Systèmes de transport . .



neo4j

4/ Base de données orientées graphe



BDD NoSQL - Atouts

- Principalement utilisé sur des clusters de serveurs
- Permet un modèle qui peut s'étendre plus facilement (scalability)
- Assouplit les contraintes habituellement présentes sur les bases de données relationnelles
- Permet de gérer rapidement des tonnes de données

BDD NoSQL - Exemples

- **Cassandra** est une base de données NoSQL en colonnes, optimisée pour le stockage et le traitement de jeux de données volumineux.
- La base de données documentaire **CouchDB** exploite une API intuitive HTTP/JSON et peut être utilisée dans n'importe quel domaine d'application, que ce soit pour des Big Data et des applications mobiles ou encore Web.
- Avec **Neo4j**, vous pouvez intégrer vos données au sein d'une base de données graphique NoSQL basée sur le Cloud. En première ligne, vous pouvez ainsi représenter dans des graphiques les relations entre les différents jeux de données et reconnaître des modèles définis.
- La **base de données Clé-Valeur redis** est ce que l'on appelle une base de données en mémoire : elle enregistre directement les données dans le cache et vous garantit ainsi de meilleures performances.

BDD NoSQL - Exercice

On souhaite réaliser une base de données orientée documents pour gérer des cours et des étudiants, étant données les informations suivantes :

1. Un cours est décrit par les attributs code, titre, description, crédits et prérequis.
2. Les prérequis sont d'autres cours.
3. Un étudiant est décrit par les attributs nom, prénom, adresse.
4. Les adresses sont composées d'un numéro de rue, d'une rue, d'une ville et d'un code postal.
5. Un étudiant suit plusieurs cours et les cours sont suivis par plusieurs étudiants.

Sachant que l'objectif de l'application est de visualiser une liste des étudiants avec les cours que chacun suit, et d'accéder aux détails des cours uniquement lorsque l'on clique sur son code ou son titre, proposer une solution adaptée à ce problème.

BDD NoSQL - Solution

Étudiant


```
{
"nom":"Dupont",
"prenom":"Jean",
"adresse":
  {
  "num":8,
  "rue":"Gury"
  },
"cours":
  [
  {
    "_id":"30ae9e51-f5c8-4022-a68d-3f3948dbdcb1",
    "code":"api04",
    "titre":"DW et NOSQL"
  },
  {
    "_id":"cf936817-19fa-4635-b009-a383c90ab6d",
    "code":"api05",
    "titre":"DW et NOSQL, la suite"
  }
  ]
}
```


Cours


```
{
"_id":"30ae9e51-f5c8-4022-a68d-3f3948dbdcb1",
"code":"api04",
"titre":"DW et NOSQL",
"description":".....",
"credits":4,
"prerequis":
  [
  {
    "_id":"a1449020-9b24-44a1-b12d-84ef592f8853",
    "code":"api03"
  }
  ]
}
```

BDD NoSQL - Comparaison

Modèle	Performance	Évolutivité	Flexibilité	Complexité	Fonctionnalité
Clé/Valeur	Élevée	Élevée	Élevée	Aucune	Variable (Aucune)
Colonne	Élevée	Élevée	Modérée	Faible	Minimale
Document	Élevée	Variable	Élevée	Faible	Variable (Faible)
Graphe	Variable	Variable	Élevée	Grande	Théorie des graphes

Des illustrations

Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Représentation de ventes en colonne

Family Sales

#ticket	date	books
1	01/01/16	2212121504 2212141556
2	01/01/16	2212141556

Family Book

#isbn	title	a-surname	a-firstname
2212121504	Scenari		
2212141556	NoSQL	Bruchez	Rudi

Illustrations

Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Représentation de ventes en document

Collection Sales

#oid	
4d0407660766b236450b45a3	"ticket" : 1 "date" : "01/01/16" "books" : ["_id" : 2212121504 "_id" : 2212141556]
4d0407660766b236450b45a4	"ticket" : 2 "date" : "01/01/16" "books" : ["_id" : 2212141556]

Collection Book

#oid	
4d0407660766b236450b45a5	"isbn" : 2212121504 "title" : "Scenari"

4d0407660766b236450b45a6	"isbn" : 2212141556
	"title" : "NoSQL"
	"author" : {
	"surname" : Bruchez
	"firstname" : Rudi
	}

Illustrations

Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Représentation de ventes en graphe

Classe Sales

#oid		
4d0407660766b236450b45a3	*property*	ticket : 1
	property	date : 01/01/16
	relation	book : 4d0407660766b236450b45a5
	relation	book : 4d0407660766b236450b45a6
4d0407660766b236450b45a4	*property*	ticket : 2
	property	date : 01/01/16
	relation	book : 4d0407660766b236450b45a6

Classe Book

#oid		
4d0407660766b236450b45a5	*property*	title : Scenari
4d0407660766b236450b45a6	*property*	title : NoSQL
	relation	author : 4d0407660766b236450b45a8

Classe Author

#oid		
4d0407660766b236450b45a8	*property*	surname : Bruchez
	property	firstnam : Rudi

BDD SQL VS BDD NoSql

Critère	SQL	NoSQL
Définition	SGBDR ou bases de données relationnelles	Base de données non relationnelle ou base de *données distribuée*
Utilisation	Requête pour analyser et récupérer données	Traiter des données liées à des applications et des sites Web modernes de plus en plus complexes
Langage de requête	Langage de requête structuré (SQL)	Ne nécessite pas un langage de requête trop complexe (peut être avec Map, reduce)
Structure de la base de données	En forme de table	Valeur-clé, d'une colonne, d'un document et d'un graphique
Schéma	Besoin d'être déterminé d'abord	Schéma dynamique pour les données non structurées
Adapté pour	Requêtes complexes et intensives	Grande base de données, Big Data
Exemples	MySQL, Postgres, MS-SQL	Redis, Neo4j, MongoDB

Exemple : HBase

HBase est un système de stockage efficace pour des données très volumineuses
Modèle de données orienté colonne .

Il permet d'accéder aux données très rapidement même quand elles sont gigantesques.

HBase est utilisée par FaceBook pour stocker tous les messages SMS, email et chat…

HBase peut être utilisée à la fois comme:
–Base de données temps réel.
–Base de données pour une lecture intensive pour les systèmes décisionnels.

Clés	Colonnes et Valeurs	
isbn7615	colonne=auteur	valeur="Jules Verne"
isbn7615	colonne=titre	valeur="De la Terre à la Lune"
isbn7892	colonne=auteur	valeur="Jules Verne"
isbn7892	colonne=titre	valeur="Autour de la Lune"

Exemple : HBase

Pour obtenir une grande efficacité, les données des tables Hbase sont séparées en régions.

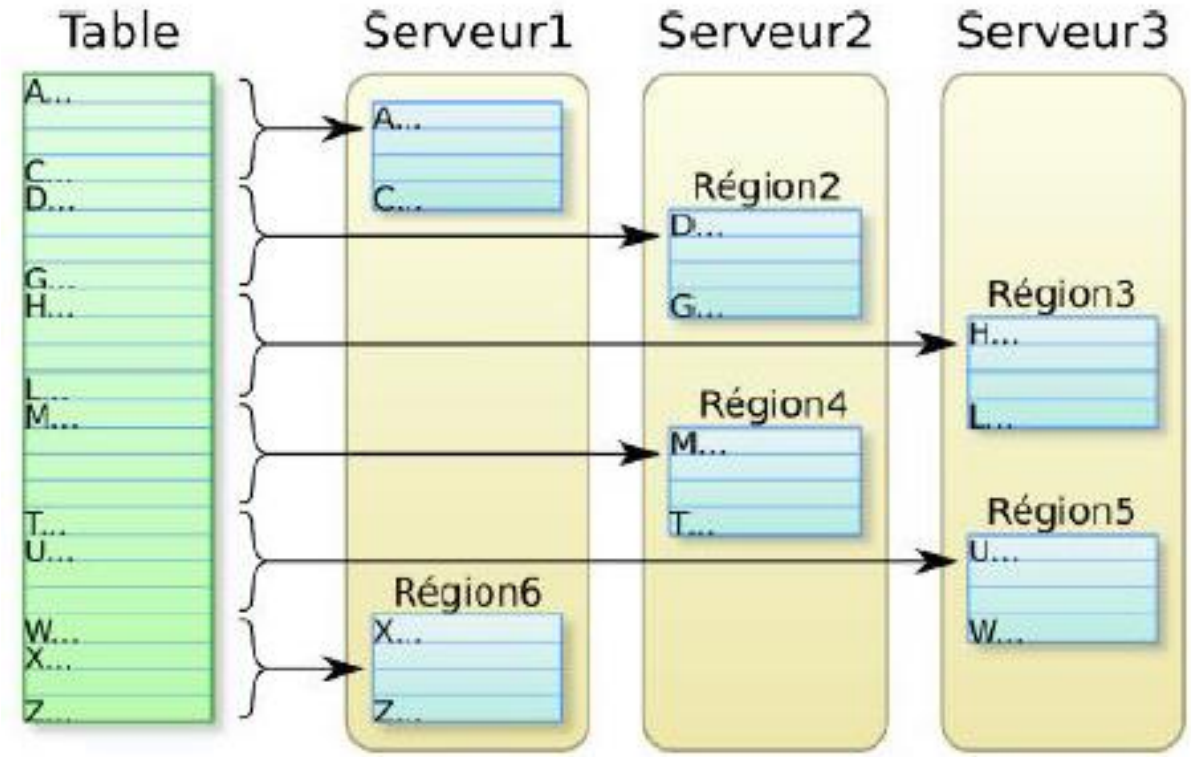
- Une région contient un certain nombre de n-uplets contigus (un intervalle de clés successives).
- Lorsqu'on crée une table, elle est mise dans une seule région.
- Lorsque la table dépasse une certaine limite, elle se fait couper en deux régions au milieu de ses clés. Et ainsi de suite si les régions deviennent trop grosses.
- Chaque région est gérée par un Serveur de Région (*RegionServer*).
- Ces serveurs sont distribués sur le cluster, ex: un par machine.
- Un même serveur de région peut s'occuper de plusieurs régions de la même table.

Au final, les données sont stockées sur HDFS.

Exemple : HBase

Une table est découpée en régions faisant à peu près la même taille.

- Le découpage est basé sur les clés.
- Chaque région est gérée par un Serveur de région.
- Un même serveur peut gérer plusieurs régions.



Requetage NoSql

Dans le monde du NoSQL, il n'y a pas de langage standard comme SQL l'est dans le monde des bases de données relationnelles. L'interrogation des bases de données NoSQL se fait au niveau applicatif à travers principalement la technique dite de «MapReduce ».
Cette technique se décompose en deux grandes étapes:

- Étape du Mapping
- Étape du Reducing

Requetage NoSql

Étape du Mapping

Chaque item d'une liste d'objets clé-valeur passe par la fonction map qui va retourner un nouvel élément clé-valeur. Exemple de la fonction map : A chaque couple (UserId, User), on assigne le couple (Rôle, User). A l'issue de l'étape de mapping, on obtient une liste contenant les utilisateurs groupés par rôle.

Une opération faisant appel à la fonction « map » permet donc de lancer une fonction pour chaque élément dans une séquence afin d'obtenir en retour une séquence de taille égale des valeurs résultantes.

Requetage NoSql

Étape du Reducing

La fonction reduce est appelée sur le résultat de l'étape de mapping et permet d'appliquer une opération sur la liste Exemples de fonction reduce:

- Moyenne des valeurs contenues dans la liste.
- Comptabilisation des différentes clés de la liste.
- Comptabilisation du nombre d'entrées par clé dans la liste.

Une opération faisant appel à la fonction « reduce » permet donc d'accumuler le contenu d'une séquence dans une seule valeur de retour, en utilisant une fonction qui va combiner chaque élément dans la séquence avec la valeur de retour de l'itération précédente.

Requetage NoSql

L’étape de mapping peut être parallélisée en traitant l’application sur différents nœuds du système pour chaque couple clé-valeur.

L’étape de réduction n’est pas parallélisée et ne peut être exécutée avant la fin de l’étape de mapping.

Les bases de données NoSQL proposent diverses implémentations de la technique MapReduce permettant le plus souvent de développer les méthodes map et reduce en JavaScript ou en Java.

BDD NoSQL - Inconvénients

- Le modèle NoSQL n'est pas parfait et présente quelques inconvénients !!!!.
- Les bases NoSQL nécessitent beaucoup de stockage en espace disque.
- Une base NoSQL sera susceptible de peser beaucoup plus lourd que les mêmes données stockées dans une base relationnelle bien normalisée.
- Un autre problème plus petit est le fait que NoSQL n'est pas fait pour supprimer la duplication de données comme le fait SQL, et donc la taille des bases de données peut rapidement devenir massive.
- L'intégrité et la sécurité des données ne sont pas garanties pour le traitement des données volumineuses.

Conclusion

Le SQL et le NoSQL resteront tous les deux utilisés à l'avenir. Chaque langage a ses avantages. En revanche, les inconvénients peuvent amener, en fonction du besoin, à utiliser l'autre langage.