Ministère de l'enseignement supérieur et de recherches scientifiques
Institut Supérieur des Etudes Technologiques de Radès
Département Technologies de l' information

Chapitre 2 :
Les Bases de Données NoSQL

Objectifs

Objectifs

- Nouveaux besoins en Gestion des données
- Théorème de CAP
- Connaîtres les différentes typologies des bases de données NoSQL
- Quelques différences entre SQL et NoSQL
- 10 meilleurs outils des bases de données NoSQL

Nouveaux besoins en gestion de données

Constat :

- essor des **très grandes plateformes et applications Web (Google, Facebook, Twitter, LinkedIn, Amazon, …)**
- **volume considérable de données à gérer par ces applications nécessitant une distribution des données et leur traitement sur de nombreux serveurs.**
- ces données sont souvent associées à des **d'objets complexes et hétérogènes**

⇒*Limites des **SGBD traditionnels (relationnels et transactionnels) basés sur SQL.***

D'où nouvelles approches de stockage et de gestion des données :

- permettant une **meilleure scalabilité dans des contextes fortement distribués.**
- permettant une **gestion d'objets complexes et hétérogènes sans avoir à** déclarer au préalable l'ensemble des champs représentant un objet
- regroupées derrière le terme **NoSQL (proposé par Carl Strozzi), les SGBD NoSQL** ne se substituant pas aux SGBD Relationnels mais les **complètant en comblant leurs faiblesses .**

Nouveaux besoins en gestion de données

SGBD Relationnels sont généralement transactionnels :

=> gestion de transactions respectant les **contraintes ACID (Atomicity, Consistency, Isolation, Durability)**

Contexte fortement distribué :

Cette gestion a un coût considérable : il est nécessaire de distribuer les traitements de données entre différents serveurs

→ alors **difficile de maintenir les contraintes ACID à l'échelle du système distribué entier**

tout en maintenant des **performances correctes.**

⇒*la plupart des SGBD « NoSQL » relâchent les contraintes ACID, ou même ne proposent pas de gestion de transactions, pour favoriser l'efficacité.*

Nouveaux besoins en gestion de données

Théorème de CAP

Théorème de CAP (Brewer, 2000) :

En 2000, Eric Brewer a formalisé un théorème très intéressant reposant sur 3 propriétés fondamentales pour caractériser les bases de données (relationnelles, NoSQL et autres) :

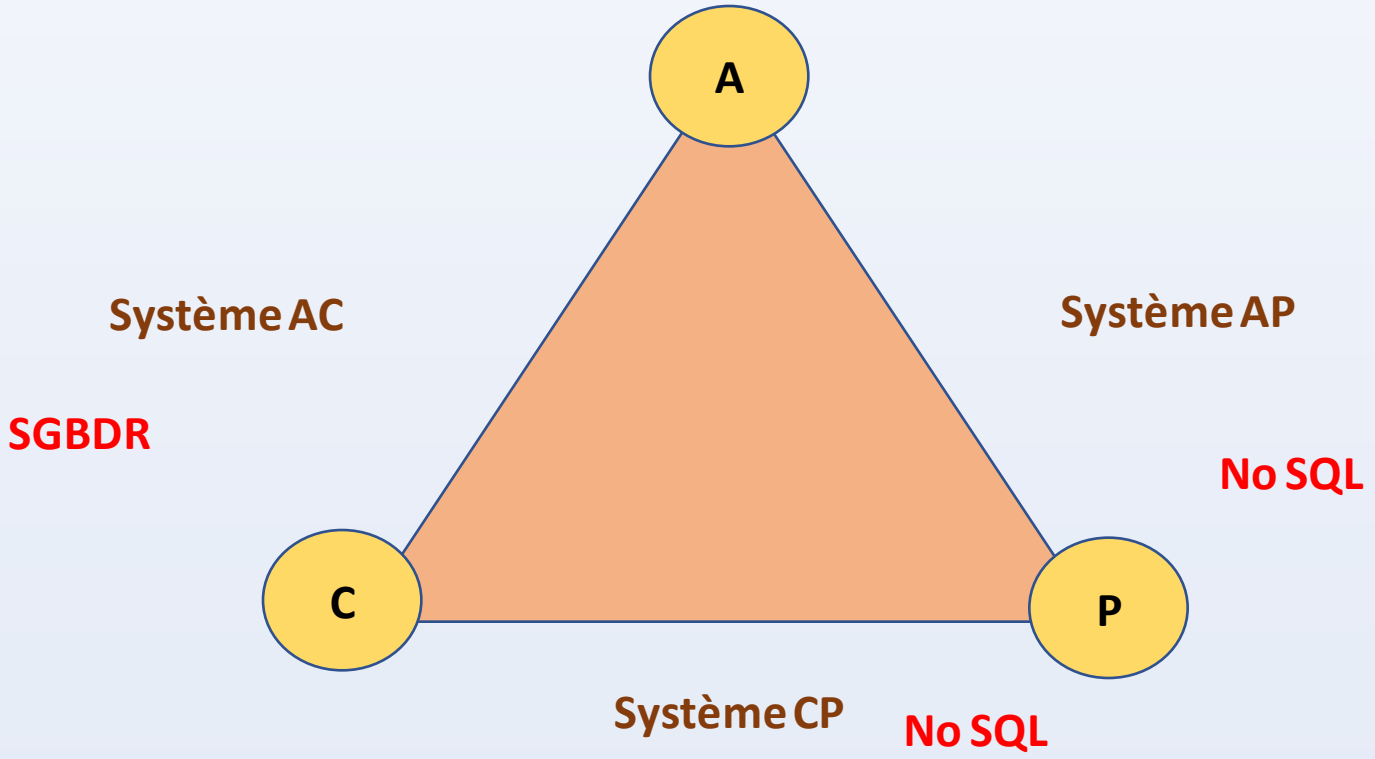
- **C*onsistency* (Cohérence)** : Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas.
- **A*vailability* (Disponibilité) :** Tant que le système tourne (distribué ou non), la donnée doit être disponible.
- **P*artition Tolerance* (Distribution) :** Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct

Le théorème de CAP dit :

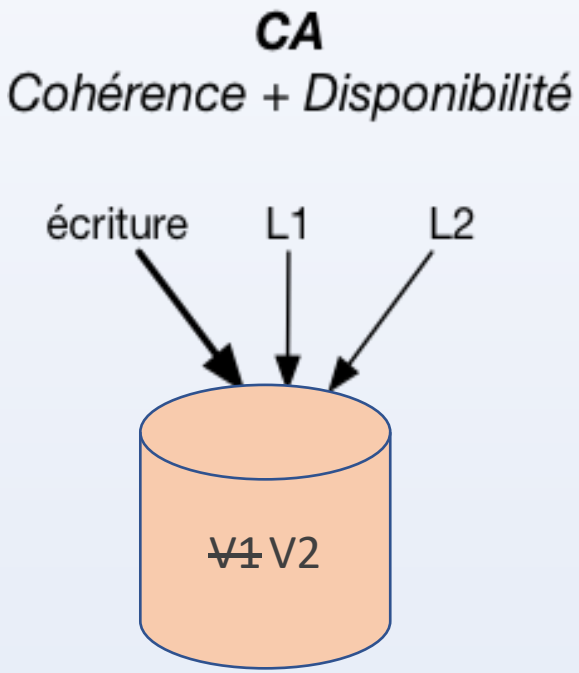
- **Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la cohérence, la disponibilité et la distribution.**

Théorème de CAP

- Les **SGBDR assurent les propriétés de *Consistance et de Disponibilité (Availability) => système AC***
- Les **SGBD « NoSQL » sont des *systèmes CP (Cohérent et Résistant au partitionnement) ou AP (Disponible et Résistant au partitionnement).***

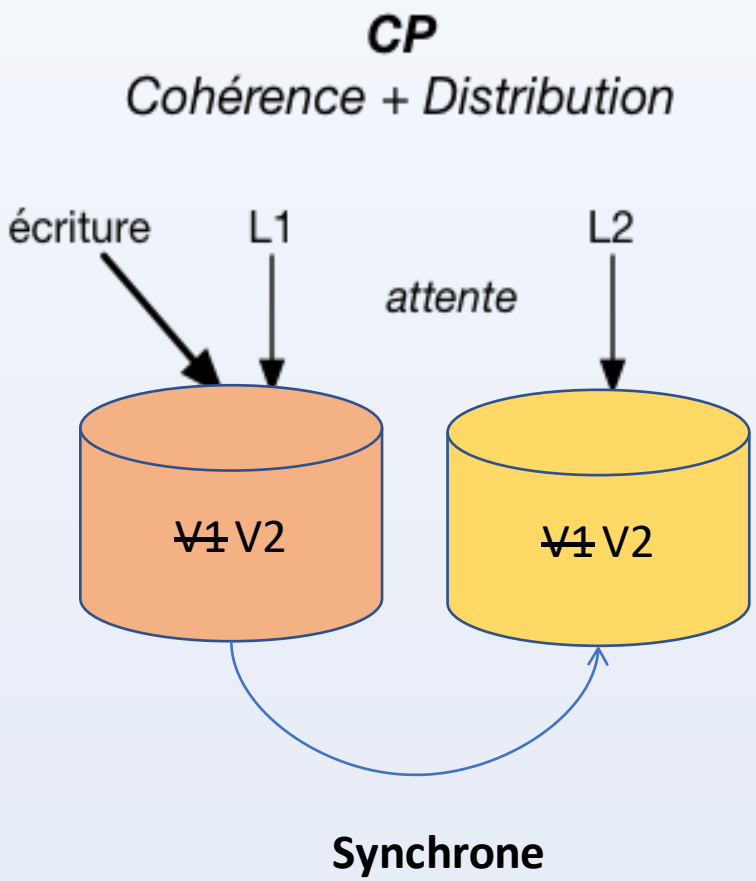


Théorème de CAP



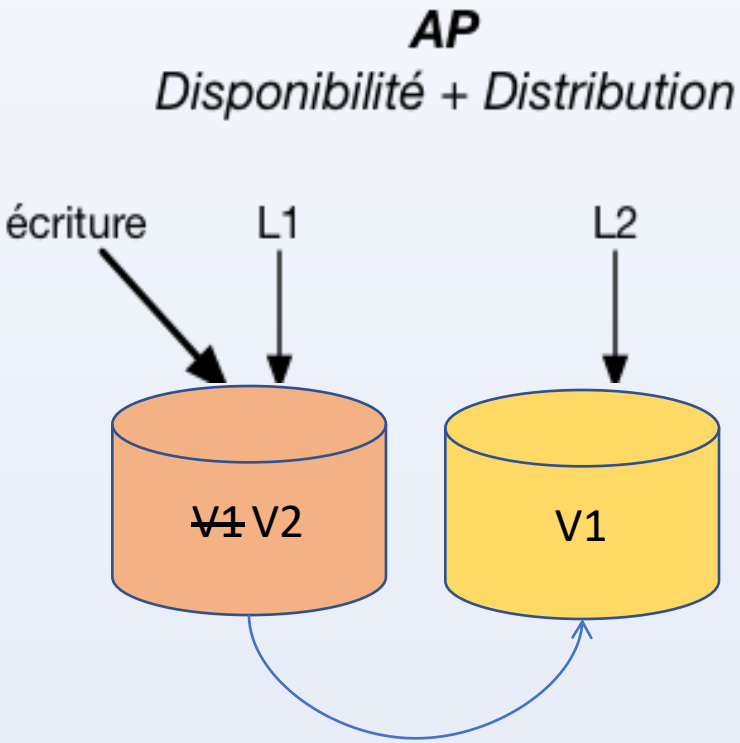
Le couple **AC** (Consistency-Availability), il représente le fait que lors d'opérations concurrentes sur une même donnée, les requêtes L1 et L2 retournent la nouvelle version (v2) et sans délai d'attente.

Théorème de CAP



Le couple **CP** (Consistency-Partition Tolerance) propose maintenant de distribuer les données sur plusieurs serveurs en garantissant la tolérance aux pannes (réplication). En même temps, il est nécessaire de vérifier la cohérence des données en garantissant la valeur retournée malgré des mises à jour concurrentielles. La gestion de cette cohérence nécessite un protocole de synchronisation des réplicas, introduisant des délais de latence dans les temps de réponse (L1 et L2 attendent la synchronisation pour voir v2). C'est le cas de la base NoSQL *MongoDB*.

Théorème de CAP



Le couple **AP** (Availability-Partition Tolerance) à contrario s'intéresse à fournir un temps de réponse rapide tout en distribuant les données et les réplicas. De fait, les mises à jour sont asynchrones sur le réseau, et la donnée est "*Eventually Consistent*" .C'est le cas de *Cassandra* dont les temps de réponses sont appréciables, mais le résultat n'est pas garanti à 100% lorsque le nombre de mises à jour simultanées devient important.

Le grand paysage des BD

Face aux faiblesses des SGBDR en termes de **performance, d'évolutivité et de flexibilité des besoins de traitement à grande échelle des données, plisieurs alternatives** ont vu le jour avec des **architectures distribuées :**

- **NoSQL BD : BD avec schéma dynamique ou sans schéma, BD de clés-valeur, BD de documents , etc…**
- **NewSQL DB :** un stockage distribué - utilise SQL - offre des capacités modernes et une architecture moderne - Destinés aux entreprises qui gèrent beaucoup de données et qui nécessitent une évolutivité.-Gére les contraintes ACID.

Report « NoSQL, NewSQL and Beyond: The answer to SPRAINed relational databases », 451 Group, April 15th, 2011

SPRAINed database

SPRAIN : acronyme qui désigne les 6 facteurs clés de l'adoption de technologies de gestion de données alternatives aux traditionnelles BDR.

- ces BD alternatives doivent permettre de traiter de **très grands volumes de données, supporter des applications hautement distribuées ou très complexes.**
- Ces **6 facteurs clés sont :**

Scalability (évolutivité) –hardware economics (économie de matériel)

Performance –BDR limitations

Relaxed consistency –CAP theorem (cohérence relachée)

Agility –polyglot persistence (agilité, persistance polyglotte)

Intricacy (intrication quantique) –big data

Necessity (nécessité) –Vaste communité open source

Présentation

Le terme NoSQL désigne une catégorie de systèmes de gestion de base de données destinés à manipuler des bases de données volumineuses pour des sites de grande audience. Les bases données NoSQL sont scalables, elles permettent de traiter les données d'une façon distribuée.

L'adoption croissante des bases NoSQL par des grands acteurs du Web (Google, FaceBook, …) => multiplication des offres de systèmes NoSQL

Caratéristiques des BD NoSQL

- Pas de relations
  - Pas de schéma physiques ou dynamiques
- Données complexes
  - Imbrication, tableaux
- Distribution de données (milliers de serveurs)
  - Parallélisation des traitements (Map/Reduce)
- Replication des données
  - Disponibilité vs Cohérence (pas de transactions)
  - Peu d'écritures, beaucoup de lectures

Typologie des BD NoSQL

Stocker les informations de la façon la mieux adaptée à leur représentation => différents types de BD NoSQL :

- **type « Clé-valeur / Key-value » : basique, chaque objet est identifié par une clé unique constituant la seule manière de le requêter**

  Voldemort, Redis, Riak, …

- **type « Colonne / Column » : permet de disposer d'un très grand nb de valeurs sur une même ligne, de stocker des relations « one-to-many », d'effectuer des requêtes par clé (adaptés au stockage de listes : messages, posts, commentaires, ...)**

  HBase, Cassandra, Hypertable, …

- **type « Document » : pour la gestion de collections de documents, composés chacun de champs et de valeurs associées, valeurs pouvant être requêtées (adaptées au stockage de profils utilisateur)**

  MongoDB, CouchDB, Couchbase, …

- **type « Graphe » : pour gérer des relations multiples entre les objets (adaptés au données issues de réseaux sociaux, …)**

  Neo4j, OrientDB, …

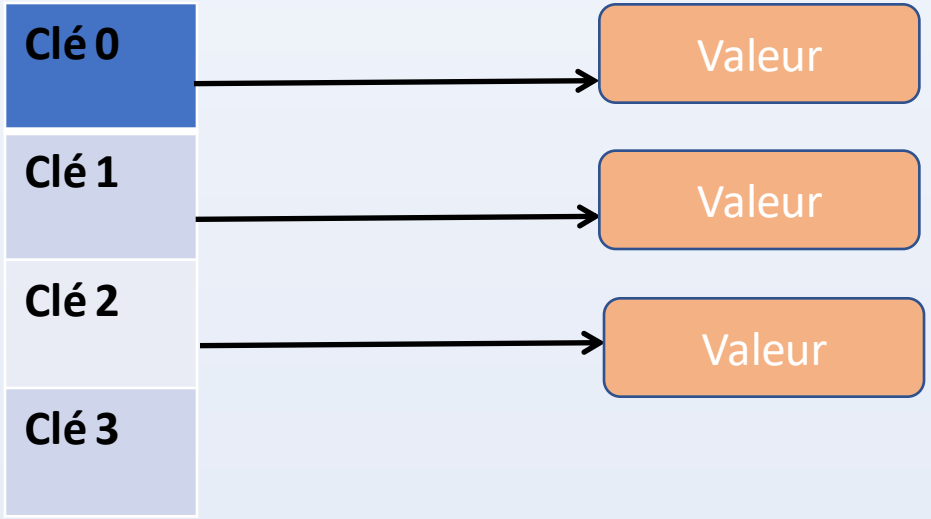
Typologie des BD NoSQL

BD NoSQL “clé/valeur”

- les données sont simplement **représentées par un couple clé/valeur.**
- la valeur peut être une **simple chaîne de caractères, ou un objet sérialisé.**
- cette absence de structure ou de typage ont un impact important sur le requêtage : toute l’intelligence portée auparavant par les requêtes SQL devra être portée par **l’applicatif qui interroge la BD.**
- Implémentations les plus connues :
  - ✓ **Amazon Dynamo (Riak en est l'implémentation Open Source)**
  - ✓ **Redis (projet sponsorisé par VMWare)**
  - ✓ **Voldemort (développé par LinkedIn en interne puis passage en open source).**

Typologie des BD NoSQL
BD NoSQL “clé/valeur”

- Chaque objet est identifié par une **clé unique qui est la seule façon de le requêter**



Typologie des BD NoSQL
BD NoSQL “clé/valeur”

Exemple:

- En voici un exemple, codé en JSON:

"nom": "philippe"

- Et le même, codé en XML:

<nom>philippe**</nom>**

Typologie des BD NoSQL

BD NoSQL “clé/valeur”

Key	Value
Book Title	Business Intelligence and Analytics: Systems for De-cision Support
Author (set)	Ramesh Sharda
	Dursun Delen
	Efraim Turban
Publication Date	2015
Edition	10th
Publisher	Pearson
…	…

Exemple: Commet une paire clé,valeur est stockée dans une BD NoSQL

Typologie des BD NoSQL

BD NoSQL “clé/valeur”

- Ces bases sont principalement faites pour le stockage temporaire et ne permettent que 4 opérations :
  - ✓La création : créer un nouveau couple (clé,valeur). Selon la base choisie, la valeur peut être n’importe quel objet.
  - ✓La lecture : lire un objet en connaissant sa clé
  - ✓La modification : mettre à jour l’objet associé à une clé
  - ✓La suppression : supprimer un objet connaissant sa clé
- elles disposent généralement d’une simple **interface de requêtage HTTP REST accessible depuis n’importe quel langage de développement.**
- ont des **performances très élevées en lecture et en écriture .**

Typologie des BD NoSQL

BD NoSQL “clé/valeur”

Forces :

- **modèle de données simple**
- **évolutivité (scalable)**
- **disponibilité**
- **pas de maintenances requises lors d'ajout/suppression**

Faiblesses :

- **modèle de données TROP simple :**

  pauvre pour les données complexes

  interrogation seulement sur clé

  déporte une grande partie de la complexité de l'application sur la **couche application elle-même**

Typologie des BD NoSQL
BD NoSQL “clé/valeur”

Pourquoi une base NoSQL orientée clé valeur?

Elles sont beaucoup utilisées en tant que cache, pour conserver les sessions d’un site web et plus généralement pour toutes les données que l’on ne souhaite conserver que pendant un certain laps de temps, pouvant aller de quelques secondes à quelques jours.

Exemple :

- gestion de panier d’achat (Amazon)
- collecte d’événements (jeu en ligne**)**

Typologie des BD NoSQL
BD NoSQL “document ”

- Elles stockent une collection de "**documents**"
- elles sont basées sur le modèle « clé-valeur » mais la valeur est un **document en format semi-structuré hiérarchique de type JSON ou XML (possible aussi de stocker n'importe quel objet, via une sérialisation)**
- les **documents n'ont pas de schéma, mais une structure arborescente : ils contiennent une liste de champs, un champ a une valeur qui peut être une liste de champs, ...**
- elles ont généralement une **interface d’accès HTTP REST permettant d’effectuer des requêtes (plus complexe que l’interface CRUD des BD clés/valeurs)**
- Implémentations les plus connues :

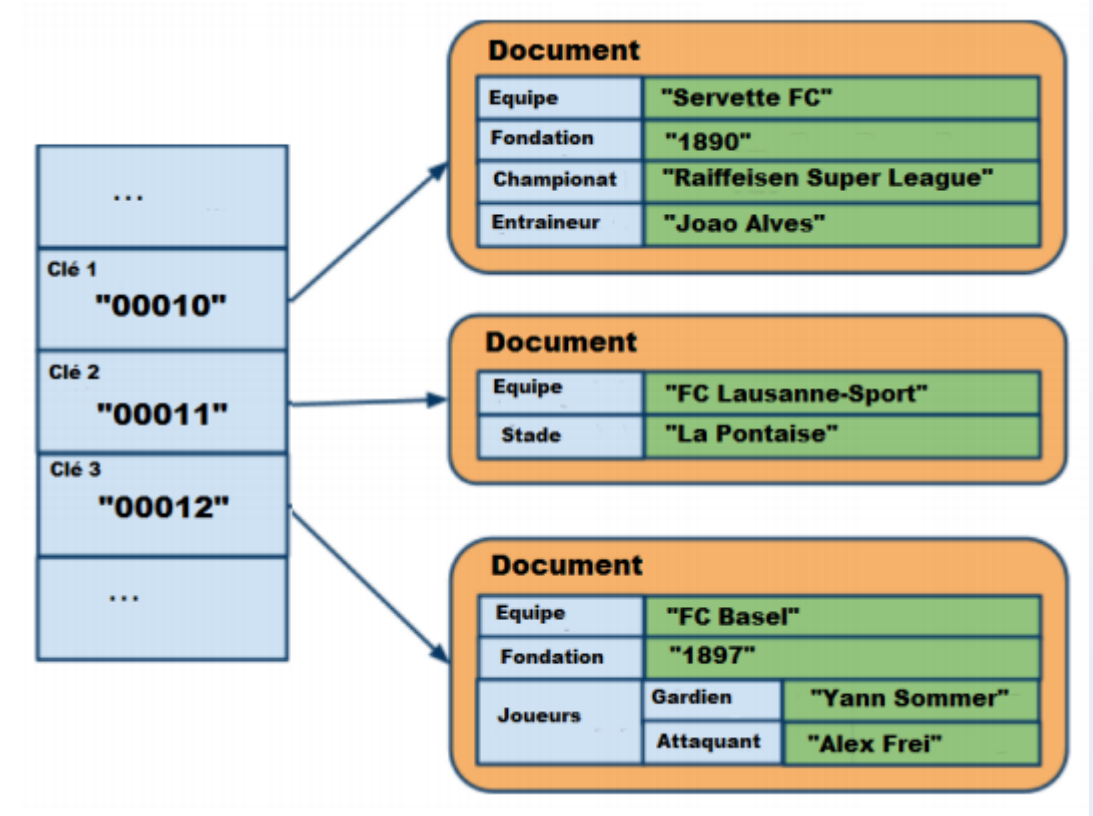
  CouchDB (fondation Apache)

  RavenDB (pour plateformes « .NET/Windows » -LINQ)

  MongoDB, Terrastore, …)

Typologie des BD NoSQL

BD NoSQL “document ”



Typologie des BD NoSQL
BD NoSQL “document ”

Exemple:

Voici un second exemple JSON, montrant un document :

{"nom": "Philippe Rigaux", "tél": 2157786, "email": "philippe@cnam.fr"}

La représentation équivalente en XML est donnée ci-dessous:

<personne>

<nom>Philippe Rigaux**</nom>**

<tel>2157786**</tel>**

<email>philippe@cnam.fr**</email>**

</personne>

Typologie des BD NoSQL
BD NoSQL “document ”

Exemple:

Voici la table des artistes en notation JSON.

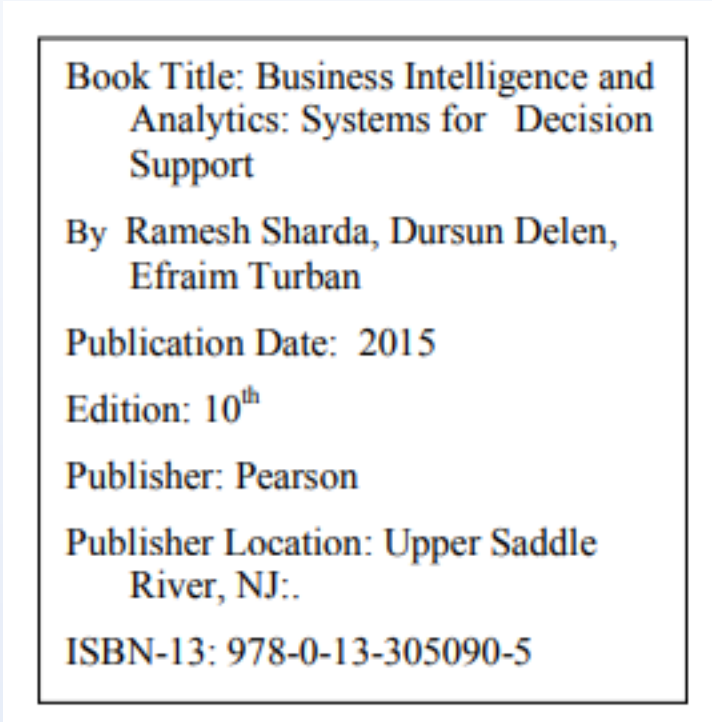
```
[
    artiste: {"id": 11, "nom": "Travolta", "prenom": "John"},
    artiste: {"id": 27, "nom": "Willis", "prenom": "Bruce"},
    artiste: {"id": 37, "nom": "Tarantino", "prenom": "Quentin"},
    artiste: {"id": 167, "nom": "De Niro", "prenom": "Robert"}, *
    artiste: {"id": 168, "nom": "Grier", "prenom": "Pam"}
]
```

Voici la table artiste en relationnel.

Id	Nom	Prénom
11	Travolta	John
12	Willis	Bruce
13	Tarantino	Quentin
167	De Niro	Robert
168	Grier	Pam

Typologie des BD NoSQL

BD NoSQL “document ”



Exemple: Commet un document est stockée dans une BD NoSQL

Typologie des BD NoSQL

BD NoSQL “document ”

Forces :

- **modèle de données simple mais puissant (expression de structures imbriquées)**
- **pas de maintenance de la BD requise pour ajouter/supprimer des données**
- **forte expressivité de requêtage (requêtes assez complexes sur des structures imbriquées)**

Faiblesses :

- **inadaptée pour les données interconnectées**
- **modèle de requête limitée à des clés (et indexes)**
- peut alors être **lent pour les grandes requêtes (avec MapReduce)**

Typologie des BD NoSQL
BD NoSQL “document ”

Les BD NoSQL type « document» sont principalement utilisées pour :

- Enregistrement d’événements
- Systèmes de gestion de contenu
- Catalogue de produits

Typologie des BD NoSQL
BD NoSQL “colonne ”

- Elles sont les plus **complexes à appréhender des BD NoSQL, même si au final on a un schéma assez proche des bases documentaires**
- elles sont très utilisées pour les **traitements d’analyse de données et dans les traitements massifs (notamment via des opérations de type *MapReduce).***
- elles offrent **plus de flexibilité que les BD relationnelles:**
  - ✓ Il est possible **d’ajouter une colonne ou** une **super colonne à n’importe quelle ligne** d’une **famille de colonnes, colonnes ou supercolonne à tout instant.**

Typologie des BD NoSQL

BD NoSQL “colonne ”

Typologie des BD NoSQL

BD NoSQL “colonne ”

Business Intelligence and Analytics: Systems for Decision Support	
Book Details (includes authors, year, edition, publisher, etc.)	
Ramesh Sharda	
Dursun Delen	
Efraim Turban	
2015	
10th	
Pearson	

Exemple: Commet une colonne est stockée dans une BD NoSQL

Typologie des BD NoSQL

BD NoSQL “colonne ”

Forces :

- **Modèle de données supportant des données semi-structurées**
- **naturellement indexé (colonnes)**
- on peut voir les résultats de requêtes en temps réel

Faiblesses :

- A éviter pour des données interconnectés : si les relations entre les données sont aussi importantes que les données elles-mêmes (comme la distance ou calculs de la trajectoire)
- à éviter pour les lectures de données complexes
- exige de la maintenance -lors de l'ajout / suppression de colonnes et leur regroupements
- les requêtes doivent être pré-écrites

Typologie des BD NoSQL

BD NoSQL “colonne ”

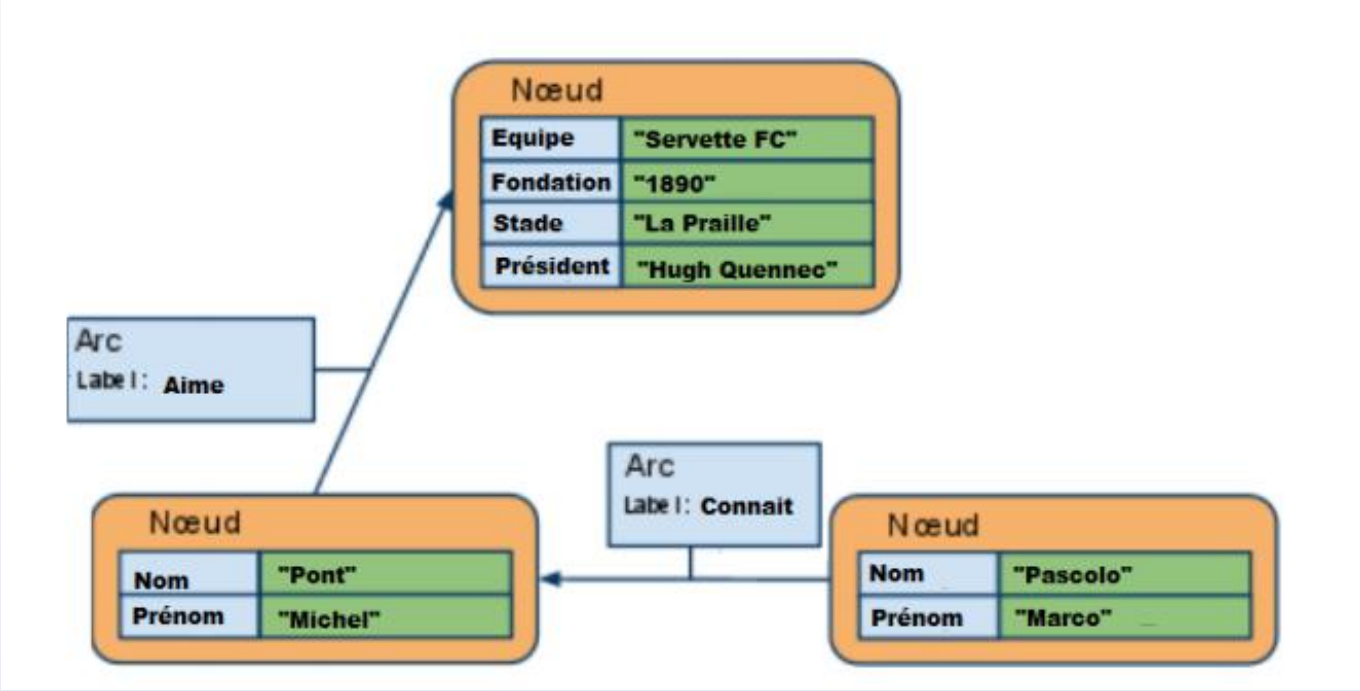
Les BD NoSQL type « Colonne » sont principalement utilisées pour :

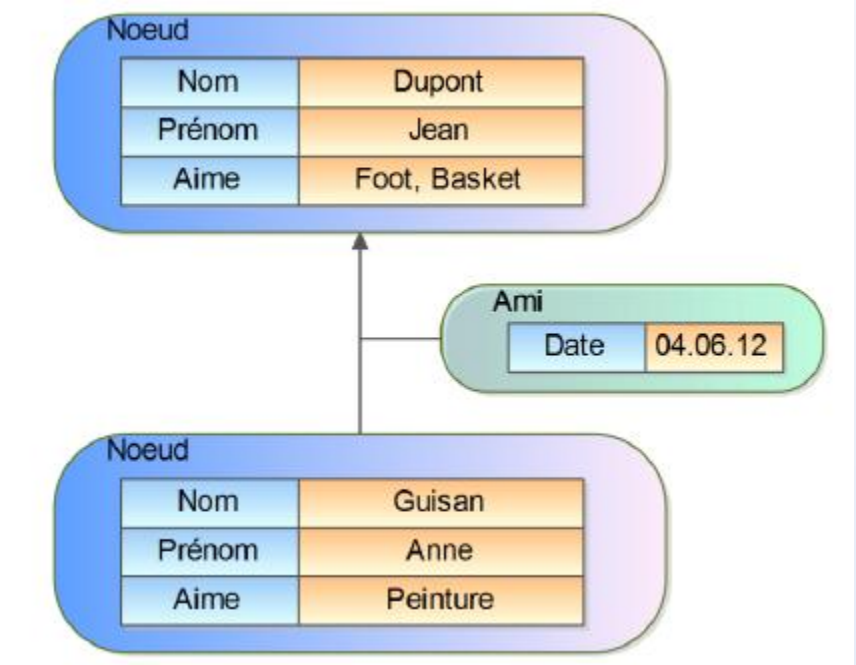
- ✓ Netflix l'utilise notamment pour le ***logging et l'analyse de sa clientèle***
- ✓ Ebay l'utilise pour ***l'optimisation de la recherche***
- ✓ Adobe l'utilise pour le ***traitement des données structurées et de Business Intelligence (BI)***
- ✓ Des sociétés de TV l’utilisent pour ***cerner leur audience et gérer le vote des spectateurs (nb élevé d'écritures rapides et analyse de base en temps réel (Cassandra)***
- ✓peuvent être de ***bons magasins d'analyse des données semi-structurées***
- ✓utilisé pour la ***journalisation des événements et pour des compteurs***

Typologie des BD NoSQL
BD NoSQL “graphe”

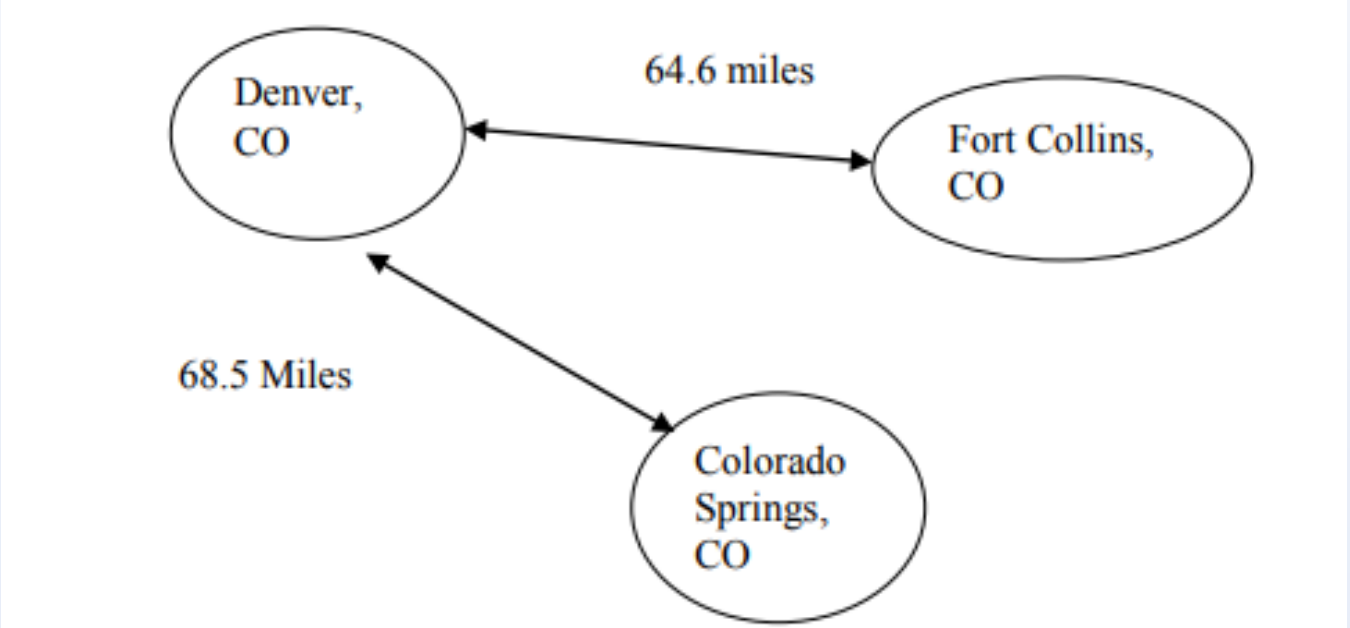
- Elles permettent la **modélisation, le stockage et la manipulation de données complexes liées par des relations**
- modèle de représentation des données basé sur la **théorie des graphes**
- s’appuie sur les notions de **noeuds, de relations et de propriétés qui leur sont rattachées.**
- Implémentations les plus connues :
  - ✓**Neo4J**
  - ✓**OrientDB (fondation Apache)**

Typologie des BD NoSQL
BD NoSQL “graphe”





Typologie des BD NoSQL
BD NoSQL “graphe”



Exemple: Commet un graphe qui représente la distance entre des cités dans une BD NoSQL

Typologie des BD NoSQL
BD NoSQL “graphe”

- **Elles utilisent :**
  - ✓un **moteur de stockage pour les objets (similaire à une base documentaire, chaque entité de cette base étant nommée noeud)**
  - ✓un **mécanisme de description d’arcs (relations entre les objets), arcs orientés et avec propriétés (nom, date, ...)**
- elles sont **bien plus efficaces que les BDR pour traiter les problématiques liées aux réseaux (cartographie, relations entre personnes, ...)**

Typologie des BD NoSQL

BD NoSQL “graphe”

Forces :

- **modèle de données puissant**
- rapide pour les **données liées, bien plus rapide que SGBDR**
- **modèles d’interrogation bien établis et performants : (fournit un ensemble commun d'interfaces permettant aux différentes technologies informatiques graphiques de travailler ensemble, que le développeur utilise en cas de besoin), SPARQL et Cypher**

Faiblesses :

- **Fragmentation (sharding) :**
  - ✓Même si elles peuvent évoluer assez bien
  - ✓Pour certains domaines, on peut aussi fractionner.

Typologie des BD NoSQL
BD NoSQL “graphe”

! **Moteurs de recommandation**

! Social computing

! Données géospatiales

! Web of things

! Catalogue des produits

! Services financiers : chaîne de financement, dépendances, gestion des risques, détection des fraudes, …

Difference entre SQL et No SQL

« LES TABLES SQL » VS « LES DOCUMENTS NOSQL »

	SQL	No SQL
Données	Tables liées entre elle	Documents
Modèle de données	Stricte	Plus flexible
Insertion des données	Strictes -> BD Cohérente	N'importe où-> pb de cohérences

Difference entre SQL et No SQL

« LE SCHÉMA SQL » VS « LA LOGIQUE NOSQL »

SQL	No SQL
Schéma: tables, indexes, séquences, contraintes, fonctions , procédures stockées. Schéma : doit être conçu et mis en œuvre avant que toute logique métier puisse être développée pour manipuler des données. Il est possible de faire des mises à jour plus tard, mais de gros changements peuvent être compliqués.	Les données peuvent être ajoutées n'importe où, à tout moment. Il n'est pas nécessaire de spécifier une conception de document à l'avance. La représentation des données en collection et le résultat en flux JSON des requêtes permet de consommer les données très rapidement et facilement par les applications front (web, mobile) .

Difference entre SQL et No SQL

« LA NORMALISATION SQL » VS « LA DENORMALISATION NOSQL »

- Par les termes « Normalization » et « Denormalisation » on veut préciser la façon dont les données sont ou pas dupliquées(noSQL) ou reliées par des clés étrangères (SQL) .

Difference entre SQL et No SQL

« LA LOGIQUE DE JOINTURE SQL » VS « PAS DE JOINTURE DANS NOSQL »

Les requêtes SQL offrent une puissante clause JOIN. Nous pouvons obtenir des données reliées dans plusieurs tables en utilisant une seule instruction SQL. Cela renvoie tous les titres de livres, auteurs et noms d'éditeurs associés (en supposant que l'un a été défini).
NoSQL n'a pas toujours d'équivalent de JOIN.
Il faut noter que d'autres SGDB ont implémenté un langage de requête proche du SQL autorisant l'usage de jointure. C'est le cas, par exemple de couchBase qui a créé le N1QL permettant de requêter une base noSQL suivant des principes proches du SQL.

Difference entre SQL et No SQL

« INTÉGRITÉ SQL » VS « NOSQL DATA INTEGRITY »

- La plupart des bases de données SQL vous permettent d'appliquer des règles d'intégrité de données à l'aide de contraintes de clés étrangères.
- Le schéma SQL applique ces règles qui prévient la création de données invalides ou d'enregistrements orphelins.
- Ces mêmes options d'intégrité de données ne sont pas disponibles dans les bases de données NoSQL. Vous pouvez stocker ce que vous voulez indépendamment de tout autre document. Idéalement, un seul document sera la seule source de toutes les informations sur un élément.

Difference entre SQL et No SQL

« TRANSACTION SQL » VS « TRANSACTION NOSQL »

- Dans les bases de données SQL, plusieurs mises à jour peuvent être exécutées au sein d'une même transaction afin de garantir le succès ou l'échec de l'exécution du code SQL. On n'imagine pas la vente d'un livre (ajout au journal de vente) sans décroitre le stock (décrémenter le stock du livre qui vient d'être vendu).
  Dans une base de données NoSQL, la modification d'un document unique est atomique (si vous mettez à jour trois valeurs dans un document, les trois sont mis à jour (Cohérence à terme) avec succès ou ils restent inchangés.

Difference entre SQL et No SQL

SQL VS NOSQL SCALABILITÉ

- Il est souvent constaté que les problématiques liées à la répartition de charge posent de réels challenges pour les entreprises (Théorème de CAP Consistency Availability Partition tolerance) .
- Le scalabilite sur un serveur SQL est complexe car dû à la nature même dont les données sont stockées, organisées et reliées entre elles. Cela a un impact direct sur les licences et donc le coût de ce type d'infrastructure.
- Les modèles de données NoSQL peuvent rendre le processus plus facile et beaucoup d'entre eux ont été conçus nativement avec des fonctionnalités de « scalabilité élastique ». L'organisation des données en documents et la « denormalization » des collections permettent le partitionnement et autorise une montée en charge de la base de données sur le matériel courant déployé sur site ou dans le Cloud. Cela permet une croissance pratiquement illimitée.

Difference entre SQL et No SQL

PERFORMANCE SQL VS NOSQL

- C’est sûrement la comparaison la plus controversée ! NoSQL est régulièrement cité comme étant plus rapide que SQL. Et ce n’est pas surprenant. Le principe de « denormalization » induit une représentation plus simple et permet donc de récupérer toutes les informations sur un élément spécifique dans une seule requête. Il n’y a donc pas besoin de liens JOIN ou de requêtes SQL complexes. Mais la redondance des informations alourdie considérablement les opérations de mise à jour.

MEILLEURS BASES DE DONNÉES NOSQL

MONGODB

- Il s'agit d'une base de données NoSQL open source orientée document. MongoDB utilise des documents de type JSON pour stocker toutes les données. Il est écrit en C++.

CASSANDRA

- Il a été développé sur Facebook pour la recherche dans les boîtes de réception. Cassandra est un système de stockage de données distribué pour le traitement de très grandes quantités de données structurées.

MEILLEURS BASES DE DONNÉES NOSQL

HBASE

- Il s'agit d'une base de données distribuée

et non relationnelle qui est conçue pour la base de données BigTable par Google.

NEO4J

- Neo4j est considéré comme une base de données de graphes native car il implémente efficacement le modèle de graphes de propriétés jusqu'au niveau du stockage.

MEILLEURS BASES DE DONNÉES NOSQL

ORACLE NOSQL

- Oracle NoSQL Database implémente une carte

allant des clés définies par l'utilisateur

aux éléments de données non structurées.

AMAZON DYNAMODB

- DynamoDB utilise un modèle de base de données NoSQL, qui n'est pas relationnel, ce qui permet d'avoir des documents, des graphiques et des colonnes parmi ses modèles de données.

MEILLEURS BASES DE DONNÉES NOSQL

COUCHBASE

- Couchbase Server est une base de données de documents NoSQL pour les applications Web interactives. Il dispose d'un modèle de données flexible, est facilement évolutif et offre des performances élevées et constantes.

COUCHDB

- C'est une base de données NoSQL Open Source qui utilise JSON pour stocker les informations et utilise JavaScript comme langage de requête.

MEILLEURS BASES DE DONNÉES NOSQL

GRAPHQL

- GraphQL est un langage de requête pour les APIs et un runtime pour répondre à ces requêtes avec vos données existantes. GraphQL fournit une description complète et compréhensible des données de votre API, donne aux clients le pouvoir de demander exactement ce dont ils ont besoin et rien de plus, facilite l'évolution des API dans le temps, et permet de puissants outils de développement.

Tendances des bases NoSQL

- https://db-engines.com/en/ranking_trend