R5.01 : Bases de données NoSQL

BUT3 SD

S1 2026-2027

Alexandre Louvet

Quelques informations utiles

Objectifs

- Découverte de l'approche NoSQL et positionnement par rapport aux BD relationnelles.
- Découverte des familles de schémas de données.
- Mise en œuvre sur des technologies NoSQL adaptée à un problème donné.

Organisation

- Cours:
- TP:
- Evaluation:

1. Rappels sur les SGBD relationnels

Le modèle relationnel

Les données sont organisées en relations:

- **Tables**: représentation des
  Un coureur a un nom, un prénom, une nationalité, une équipe et un nombre de points UCI
- **Colonnes**: représentation des
  Nom, Prénom, équipe, nombre de points
- **Lignes**: tuple de valeurs.
  France, SEIXAS, Paul, Decathlon CMA CGM, 4800.7

Rider	Team	Points
POGAČAR Tadej	UAE Team Emirates - XRG	11321.8
EVENEPOEL Remco	Red Bull - BORA - hansgrohe	6956.6
VINGEGAARD Jonas	Team Visma | Lease a Bike	6866.4
DEL TORO Isaac	UAE Team Emirates - XRG	6833.5
SEIXAS Paul	Decathlon CMA CGM Team	4800.7
PIDCOCK Tom	Pinarello Q36.5 Pro Cycling Team	4535.9
PHILIPSEN Jasper	Alpecin - Premier Tech	4141.6
SKJELMOSE Mattias	Lidl - Trek	3293.8
PEDERSEN Mads	Lidl - Trek	3229.1
VAN DER POEL Mathieu	Alpecin - Premier Tech	3140.6
MARTINEZ Lenny	Bahrain - Victorious	3087.4
SCHMID Mauro	Team Jayco AlUla	2927.8
DE LIE Arnaud	Lotto Intermarché	2788
GALL Felix	Decathlon CMA CGM Team	2683
HINDLEY Jai	Red Bull - BORA - hansgrohe	2667.8
MAGNIER Paul	Soudal Quick-Step	2599
SCARONI Christian	XDS Astana Team	2564
VAN AERT Wout	Team Visma | Lease a Bike	2479.6

Le modèle relationnel (ii)

coureur	Pays	Nom	Prenom	Nom_Equipe	Points
	Slovenie	POGACAR	Tadej	UAE	11321
	Belgique	EVENEPOEL	Remco	Red Bull - Bora	6956
	Danemark	VINGEGAARD	Jonas	Visma	6866
	Mexique	DEL TORO	Isaac	UAE	6833
	France	SEIXAS	Paul	Decathlon CMA CGM	4800

Chaque tuple est et il est identifié par un attribut ou une combinaison d'attributs appelés

"France, SEIXAS, Paul, Decathlon CMA CGM, 4800" a pour *"SEIXAS Paul"*

SQL et algèbre relationnelle

SQL (Structured Query Language) est le langage normalisé servant à exploiter des bases de données relationnelles.

En SQL, on décrit que l'on souhaite obtenir pas de l'obtenir.

```
SELECT * FROM coureur WHERE pays='France'
```

On n'indique pas qu'il faut parcourir la table (`for`) et sélectionner la ligne si (`if`) l'attribut Pays a pour valeur France.

SQL et algèbre relationnelle (ii)

Les jointures

On utilise principalement l'opération de pour recomposer l'information entre différentes tables.

```
SELECT * FROM coureur JOIN equipe on
Nom_Equipe WHERE pays='France' and
loc='France'
```

equipe	Nom_Equipe	loc
	UAE	Emirats Arabes Unis
	Red Bull - Bora	Allemagne
	Visma	Pays-Bas
	Decathlon CMA CGM	France

Ici on joint la table contenant la localisation des équipes et on sélectionne les coureurs français courant pour des équipes françaises.

Normalisation

Une forme normale caractérise le fait que la structure de la base respecte

Le respect par une base de données d'une forme normale permet de garantir certaines propriétés.

- Eviter la
- Interdire les venant des redondances dont une partie seulement a été mise à jour (différentes versions d'une même information, sans que l'on sache laquelle est valide) .
- Diminuer la
- Limiter qui sont des processus bloquants (écritures).

5 formes normales standard (1NF - 5NF) + forme normale de Boyce-Codd (FNBC).

Le respect d'une forme normale de niveau supérieur implique le respect des formes normales des niveaux inférieurs.

Propriétés ACID dans un SGBD relationnel

Les propriétés ACID sont un ensemble de propriétés qui garantissent qu'une transaction informatique est

- **Atomicité**: Garantit que chaque transaction est traitée comme une seule "unité", qui : si l'une des déclarations constituant une transaction échoue, la transaction entière échoue et la base de données reste inchangée.
- **Cohérence**: Garantit qu'une transaction ne peut faire passer la base de données que d'un état cohérent à un autre, en de la base de données : toute donnée écrite dans la base de données doit être valide selon toutes les règles définies, y compris les contraintes, les rollbacks, les déclencheurs et toute combinaison de ceux-ci.
- **Isolation**: Garantit que l'exécution simultanée des transactions laisse la base de données dans le même état que celui qui aurait été obtenu si les transactions avaient été exécutées
- **Durabilité**: Garantit qu'une fois qu'une transaction a été validée, elle le restera (par exemple, une panne de courant ou un crash).

Ces garanties sont pensées pour un à savoir des ordinateurs uniques.

2. Le NoSQL (Not only SQL)

Pourquoi remettre en cause le modèle relationnel ?

Big Data

Désigne les ressources d'informations dont les caractéristiques imposent l'utilisation de technologies et de méthodes analytiques particulières, et qui dépassent et nécessitent des traitements

Les 3V du Big Data

- La quantité de données générées est telle qu'elle ne peut plus être traitée et stockée sur un unique ordinateur.
  - Twitter générait en janvier 2013, 7 téraoctets de données chaque jour et Facebook 10 téraoctets.
  - Le radiotélescope Square Kilometre Array produira 50 téraoctets de données analysées par jour.
- La fréquence à laquelle les données sont engendrées, capturées ou mise à jour est telle qu'elles ne peuvent plus être traitée ou stockée sur un unique ordinateur.
  - Le radiotélescope Square Kilometre Array produira des données brutes à un rythme de 7 000 téraoctets par seconde.

Pourquoi remettre en cause le modèle relationnel ? (ii)

- La variété de la nature des données et des traitements nécessaires rend impossible le stockage et le traitement sur un même ordinateur.

Limites concrètes des SGBD relationnels

- On ne peut pas rendre une machine plus performante indéfiniment (et ça coûte de plus en plus cher).
- Schéma rigide.
- Jointures coûteuses à grande échelle (complexité au moins linéaire sans pré-traitement).

Scalabilité verticale vs horizontale

Scalabilité (Extensibilité)

Désigne la capacité d'un produit à s'adapter à un changement d'ordre de grandeur de la demande (montée en charge), en particulier sa capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande.

- Scalabilité verticale: On augmente
  Amélioration des composants, remplacement par des machines plus puissantes
- Scalabilité horizontale: On augmente
  Répartition de charge, traitements parallélisés

Pourquoi le NoSQL est pensé nativement pour être scalable horizontalement?

- Données dont l'exploitation ne dépendent pas de `JOIN`.
- Facilement entre plusieurs serveurs.

Le théorème CAP

Propriétés des systèmes de calcul distribués

- Cohérence (Consistency): Tous les nœuds du système voient exactement les

- Disponibilité (Availability): Toutes les requêtes reçoivent
- Tolérance au partitionnement (Partition tolerance): En cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner

Théorème (Brewer)

On ne peut garantir que propriétés simultanément pour les systèmes distribués.

⇒ C'est donc le cas des systèmes distribués de base de données.

ACID vs BASE

ACID : Atomicité, Cohérence, Isolation, Durabilité (relationnel)

Pour garantir les propriétés ACID, il faut garantir les propriétés de disponibilité (Availability) et de cohérence (Consistency) on parle donc de systèmes **CA**. Ils ne sont donc pas résilients à la chute d'un nœud réseau.

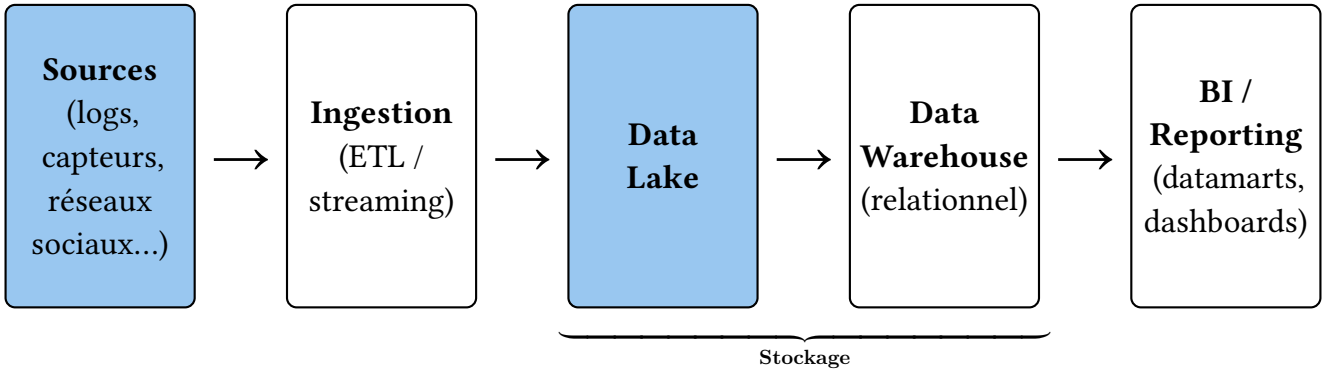
BASE : Basically Available, Soft state, Eventually consistent (NoSQL)

- Basically Available: Le système mais la réponse peut correspondre à un ou être
- Soft state: Le système peut être dans des états dans lequel il
- Eventually consistent: A terme le système dans un état cohérent.

Systèmes CP	**Systèmes AP**
Cohérence + Partitionabilité	Disponibilité + Partitionabilité
MongoDB, HBase, Redis	Neo4j, CouchDB, Cassandra, DynamoDB

Positionnement dans la chaîne décisionnelle

- La donnée suit un parcours :
- Les bases NoSQL s'insèrent à plusieurs endroits de cette chaîne, en complément du relationnel.



- Stockage natif pour les données brutes produites par les sources.
- Entre on trouve souvent aujourd'hui une zone de stockage intermédiaire, brute et peu structurée, portée par des technologies NoSQL.

3. Systèmes distribués appliqués aux BDD

Partitionnement horizontal (sharding)

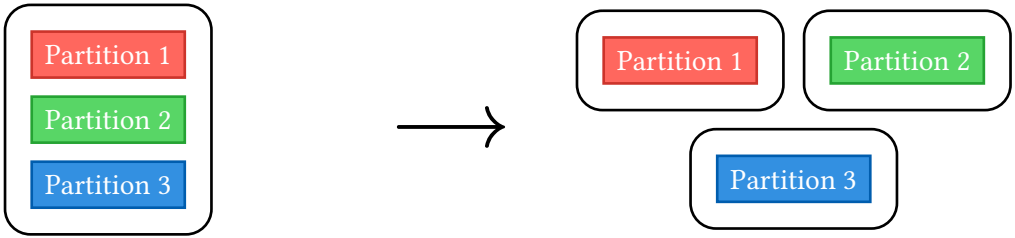
Le but est de répartir les données sur plusieurs nœuds selon une clé de partitionnement. Les lignes de la base de données sont séparées en morceaux appelés . Chaque shard peut alors être stockée sur différents serveurs et/ou à différents endroits physiques.

On peut alors utiliser différentes stratégies pour construire les S shards:

- Basées sur : Soit $H : K \mapsto [1, S]$ une fonction de hachage qui transforme une clé primaire $k \in K$ d'une ligne l de la BD. On stocke alors l dans la shard $H(k)$.
- Basées sur une : on choisit un attribut et on sépare la plage des valeurs possibles de cet attribut en valeurs qu'on répartit entre différentes shards.
  Pour un liste de produit entre 0 et 1000€, on stocke sur la shard 1 les produits entre 0 et 100€, sur la shard 2 entre 100.01€ et 300€ et le reste sur la shard 3.
- Basées sur : on maintient un table qui indique sur quelle shard est stockée chaque donnée représentée par une clé.

Partitionnement horizontal (sharding) (ii)

Méthode	+	-
Hachage	Répartition uniforme entre les shards	Pas de contrôle sur la shard sur laquelle une ligne donnée est stockée
Plage de valeur	Préserve la localité pour les requêtes par exemple dans le cas de séries temporelles	Potentiellement mauvais équilibrage de la charge entre les shards
Annuaire	Garantit un contrôle total sur le choix de la shard de stockage	Demande une (longue) table supplémentaire



Élasticité et équilibrage de charge

Dans un système distribué de base de données, il est possible de retirer ou d'ajouter des nœuds.

- Ajouter ou retirer un nœud doit redistribuer le
- Le comportement dépend directement de choisie.

Partitionnement par hachage *(Cassandra, DynamoDB)*

- En utilisant des fonctions de hachage particulières , on assure que le changement de topologie ne déplace qu'une fraction des nœuds.

Partitionnement par plage *(HBase, BigTable)*

- Rééquilibrage par : une plage trop chargée est coupée en deux, une moitié migre vers un autre nœud ou : fusion de plages sous-utilisées.

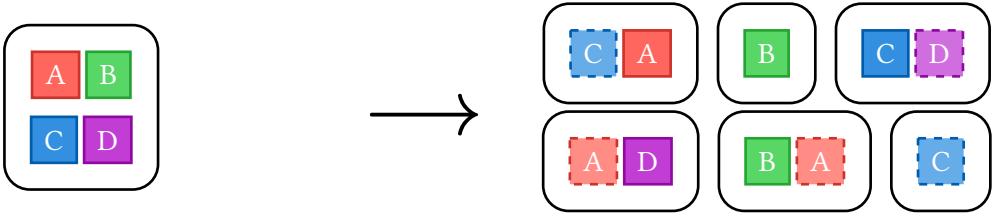
Partitionnement par annuaire *(MongoDB)*

- Rééquilibrage : il choisit quelles données déplacer, puis met à jour la table.
- Flexible, mais introduit un point de coordination central.

Réplication des données

La réplication est un processus de partage d'informations pour assurer la de données entre plusieurs sources , pour améliorer la fiabilité, la tolérance aux pannes, ou la disponibilité.

- Modèle maître-esclaves: Dans le modèle maître-esclave, nœud accepte les écritures et les esclaves se mettent à jour en lisant le maître.
  - Avantage: Simplifie la mise en cohérence car les modifications ne peuvent venir que d'une seule source.
- Modèle multi-maîtres: Dans le modèle multi-maîtres, acceptent les écritures et s'envoient mutuellement les mises à jour.
  - Avantage: Réduit la latence d'écriture en permettant à la source la plus proche de l'utilisateur d'accepter les modifications.



Tolérance aux pannes et élection de leader

Pour rester tolérant aux pannes, notamment dans le cadre de la réplication maître-esclave, il faut que les systèmes soient capables de choisir de manière autonome un nouveau maître. Il faut alors établir entre nœuds pour rester cohérent malgré les pannes.

Raft est un algorithme permettant d'obtenir efficacement dans un système distribué.

- Le maître envoie un "battement de cœur" régulier aux esclaves pour signaler qu'il "est en vie".
- Chaque esclave a un délai aléatoire qui définit la fréquence à laquelle il s'attend à recevoir le battement de cœur du maître.
- Si le maître tombe en panne, les esclaves vont:
  - soit se porter candidat si ils n'ont rien reçu dans leur délai. Dans ce cas ils envoient un message aux autres esclaves pour signaler leur candidature au rôle de maître.
  - soit recevoir un/des message(s) de candidature d'autres esclaves avant la fin de leur délai. Dans ce cas ils donnent leur voix au premier candidat dont ils ont reçu la candidature.
- Si un candidat obtient une majorité de voix il est élu comme le nouveau maître, sinon une nouvelle élection prend place.

Cohérence des données d'un système distribué

Rappel:

- Cohérence forte (ACID): Garantit qu'une transaction ne peut faire passer la base de données que d'un état cohérent à un autre.
- Cohérence éventuelle (BASE): Garantit qu'à terme le système retournera dans un état cohérent.

La cohérence réglable

Dans certains SGBD NoSQL (ex: Cassandra), il est possible de choisir le nombre minimum de lectures et d'écritures qui doivent s'effectuer correctement pour valider l'opération. On appelle ces valeurs

et

Si N représente le nombre de nœuds, en choisissant les quorum de lecture Q_l et d'écriture Q_e tel que $Q_l + Q_e > N$. On garantit ainsi que chaque lecture permettra de voir la dernière écriture même si tous les nœuds ne sont pas à jour individuellement.

Avantages:

- Permet de faire varier le compromis cohérence/disponibilité selon l'opération.
- Apporte de la flexibilité qui évite d'être bloqué par un nœud lent ou de manquer des mises à jour.

PACELC, une extension de CAP

Le théorème CAP peut-être étendu pour inclure des propriétés supplémentaires en cas de non-partitionement. C'est le théorème

Ce théorème indique que même en l'absence de partition (E pour Else), il y a un compromis entre
et

Ainsi on obtient 4 familles de bases de données:

- PA/EL: Priorise la disponibilité et la faible latence.
- PA/EC: Quand le système est partitionné, priorise la disponibilité, sinon priorise la cohérence.
- PC/EL: Quand le système est partitionné, priorise la cohérence, sinon priorise la faible latence.
- PC/EC: Priorise la cohérence.

PA/EL	PA/EC	PC/EL	PC/EC
Cassandra (faible quorum), DynamoDB	Cassandra (haut quorum), MongoDB	PNUTS (Yahoo)	Neo4j

4. Les familles de bases NoSQL

Bases orientées clés-valeurs

Base de données clé-valeur

Une base de données clé-valeur, est un paradigme de stockage de données conçu pour stocker, récupérer et gérer des tableaux associatifs, et une structure de données souvent appelée ou

- Tableau associant une clé à des espaces mémoires où sont stockés des valeurs.
- Une valeur peut-être un objet complexe.
  Redis: chaînes de caractères, tableaux associatifs, listes, ensembles et ensembles ordonnés.
- Accès aux valeurs exclusivement par le biais de la clé: les requêtes portant sur le contenu des valeurs ne sont pas possibles.
  4 opérations possibles (CRUD): Insertion (Create), Lecture (Read), Mise à jour (Update), Suppression (Delete).

Bases orientées clés-valeurs (ii)

Avantages

- Accès très efficace aux données (O(1)).
- API simple permettant une intégration à un grand nombre d'outils.

Inconvénients

- Modèle de données (trop) simple.
- Fait reposer la quasi-intégralité de la complexité du traitement des données sur l'application car ne permet pas les requêtes complexes.

Redis

Amazon DynamoDB

Riak

Bases orientées documents

Base de données orientée documents

Une base de données orientée documents est un système de stockage de données conçu pour stocker, récupérer et gérer des informations orientées documents, également appelées

- L'unité de stockage est un document, généralement encodé en JSON ou en son équivalent binaire BSON et organisés dans
- Aucun schéma n'est imposé au moment de l'écriture : deux documents d'une même collection peuvent avoir
- Il est possible de lier les documents entre-eux par le biais de
- Il faut choisir entre dupliquer des informations pour optimiser la lecture ou les référencer pour éviter les incohérences.
  Exemple: Stocker les commentaires d'une publication dans le même document que celle-ci ou dans un document différent.

Bases orientées documents (ii)

Avantages

- Modèle très flexible qui ne requiert pas de modélisation en amont.
- Permet le stockage de contenus hétérogènes.
- Passage à l'échelle simple.
- Possibilité de faire des requêtes complexes portant sur le contenu des documents.

Inconvénients

- Les requêtes complexes peuvent être lentes car la base n'a pas de structure pour les optimiser
- Fait reposer la cohérence des documents sur l'application.

MongoDB

Couchbase

Apache CouchDB

Bases orientées colonnes

Base de données orientée colonnes

Une base de données orientée colonnes est un système de gestion de base de données (SGBD) qui stocke les tableaux de données par colonne et non par ligne.

- L'orientation colonne permet un accès plus efficace aux données pour interroger un (en éliminant le besoin de lire les colonnes qui ne sont pas pertinentes).
- Le stockage par colonne permet de d'une colonne en ne stockant qu'une seule fois les valeurs uniques.

La conception d'un schéma orienté colonnes part et non des entités métier comme en relationnel. C'est une inversion complète de la démarche de modélisation habituelle.

Une même donnée peut ainsi être dupliquée dans plusieurs tables, chacune conçue pour répondre efficacement à un type de requête précis.

Bases orientées colonnes (ii)

Avantages

- Efficace pour les données sparses (pas de `NULL`).
- Ajout de colonnes simple.
- Efficaces pour les séries temporelles, données de capteurs IoT, journalisation à très grande échelle : des cas où le volume d'écriture est massif et les patterns de lecture sont connus à l'avance.

Inconvénients

- Ajout de lignes plus complexe qu'en relationnel.
- Les requêtes non anticipées lors de la modélisation sont difficiles voire impossibles à exprimer efficacement.

Cassandra

Apache HBase

ScyllaDB

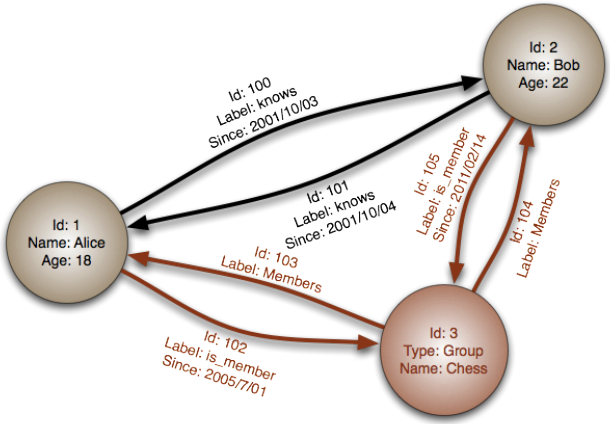
Bases orientées graphes

Graphe (orienté)

Un graphe (orienté) est composé de nœuds et de relations entre ces nœuds modélisées par des arcs.

Une base de données orientée graphe correspond à un système de stockage capable de fournir : chaque voisin d'une entité est accessible grâce à un pointeur physique.

- Les nœuds et les relations peuvent être munis de propriétés.
- On parcourt la base de données comme on parcourt un graphe: Cela permet de faire dépendre la complexité des requêtes du voisinage exploré.



source: wikipedia

Bases orientées graphes (ii)

Avantages

- Très efficace pour traiter les données sous forme de réseau (cartographie, réseaux sociaux).
- Adapté à des très grands volumes de données grâce à une exploration locale.

Inconvénients

- Difficile à partitionner car il faut séparer des relations.

Neo4j

ArangoDB

Amazon Neptune

5. Synthèse

Tableau comparatif des 4 familles

	Clés-valeurs	Documents	Colonnes	Graphes
Modèle de données	Paire clé/valeur, valeur opaque	Documents semi-structurés (JSON/BSON)	Orienté colonne	Nœuds, relations, propriétés
Cohérence par défaut	Forte sur une clé donnée	Forte (via un maître)	Réglable par quorum	Forte (via un maître/quorum)
Scalabilité horizontale	Très bonne	Bonne	Excellente	Limitée
Types de requêtes	Accès par clé uniquement	Requêtes riches, agrégations	Requêtes anticipées à la conception	Traversées de graphes
Cas d'usage typique	Cache, sessions, compteurs	Catalogues, contenus hétérogènes	Séries temporelles, IoT, logs	Réseaux sociaux, recommandation, fraude

Comment choisir?

Questions à se poser :

- Quelle est la de mes données?
- Quel est le dominantes que je vais effectuer?
- Quels sont mes besoins en terme de ?
- Quelle de données vais-je devoir traiter?

Un même système d'information souvent plusieurs familles de bases de données ("polyglot persistence").

Le NewSQL ?

NewSQL

Le NewSQL regroupe un ensemble de technologies qui essayent d'allier et

Ils se basent sur des transactions avec les propriétés suivantes:

- Courtes (pas d'interactions bloquantes avec les utilisateurs)
- Impactant peu de données à la fois.
- Utilisant des annuaires plutôt que des scans de la BD.
- Faible variété des types de requêtes.

Utiles pour les acteurs ayant trop de données pour le SQL classique mais qui ont besoin d'une grande fiabilité et cohérence des transactions. *(Finance, Gestion de commandes)*

CockroachDB

Google Spanner

6. Traitement, formats et gouvernance

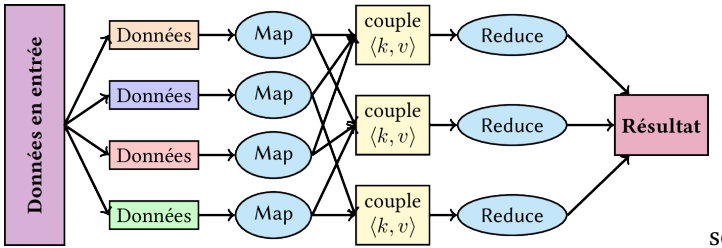
Du stockage au traitement

MapReduce

Framework développé par Google pour permettre l'écriture simple de programmes de calcul distribué.

Il est basé sur deux opérations et s'inspire du paradigme :

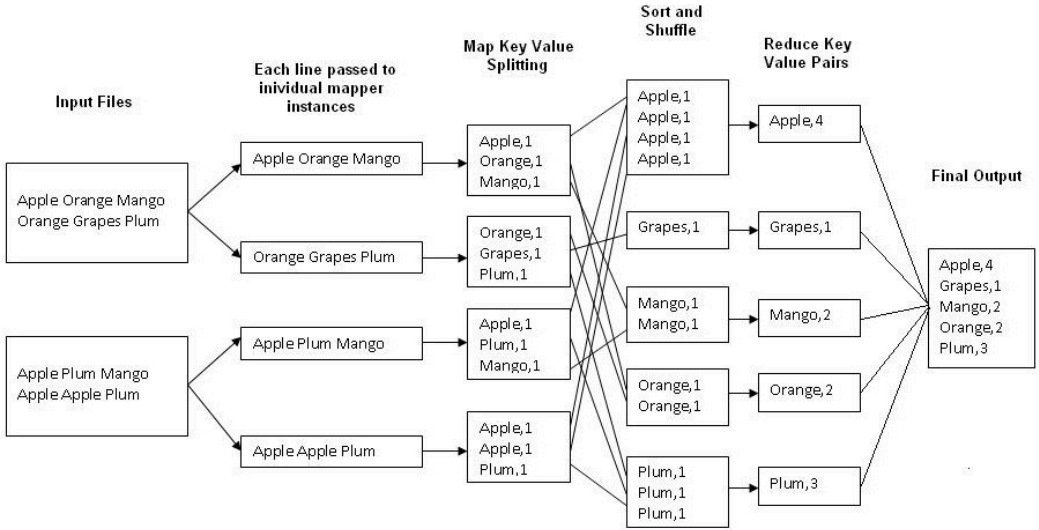
- Map (Diviser): le nœud analyse un problème, le découpe en , et le délègue à d'autres nœuds (qui peuvent en faire de même récursivement).
- Reduce (Recombiner): Les nœuds les plus bas font au nœud parent qui les avait sollicités. Celui-ci calcule un résultat partiel puis il remonte l'information à son tour.



source: wikipedia

Du stockage au traitement (ii)

Exemple, comptage de mots (source: wikipedia)



Du stockage au traitement (iii)

Pour utiliser l'algorithme MapReduce, il faut donc une solution de stockage de données flexible qui supporte l'utilisation de : le NoSQL.

Hadoop est l'implémentation open-source du framework MapReduce en Java distribué par la fondation Apache.

Apache Spark est le successeur de MapReduce améliorant les performances de Hadoop en effectuant les calculs en mémoire autant que possible, là où Hadoop écrit les données sur le disque après chaque opération.

Formats de sérialisation

Sérialisation

La sérialisation est le codage d'une information sous la forme d'une
pour, par exemple, sa sauvegarde ou son transport sur le réseau.

Il existe de nombreux formats de sérialisation qui correspondent à différents besoins. Par exemple,

- JSON: Utilisé pour obtenir une sérialisation de documents. Peu de compression et pas de description de la structure.
- Apache Parquet: Utilisé pour sérialiser des données sous forme de Bénéficie d'une forte compression et permet une lecture efficace en embarquant sa structure.
- Apache Avro: Utilisé pour sérialiser Contient un schéma décrivant l'organisation des données en JSON mais les données elles-même sont compressées.

Il faut choisir son format de sérialisation selon en prenant en compte les besoins de lisibilité, de compacité et selon le type de données à sérialiser.

Moteurs de recherche et indexation

En complément des technologies de NoSQL, notamment les bases orientées documents, il est commun d'utiliser des comme Elasticsearch ou Apache Solr.

Ces technologies permettent, en particulier, de mettre en place des qui permettent de rechercher efficacement tous les documents d'une base de données qui contiennent un terme donné.

- Permet de rendre les requêtes complexes reposant sur le contenu des documents plus efficaces en sous-traitant une partie de la requête au moteur d'indexation.
- Requiert plus de mémoire car les moteurs d'indexation en ont besoin pour fonctionner.

Sécurité et gouvernance des données

Le partitionnement et la réplication des données amené par les technologies NoSQL peuvent rendre certains aspects de la plus difficile.

Notamment, le RGPD:

- Rend obligatoire la possibilité pour les utilisateurs de voir leur données effacées définitivement
  - Certains mécanisme de NoSQL sont uniquement en écriture et la donnée n'est complètement effacée du disque que lorqu'on effectue une compaction du SGBD.
  - Du fait de l'absence de cohérence forte, il est possible d'avoir des qui auraient dû être effacées mais ont été répliquées pendant la phase intermédiaire de mise en cohérence du système.
- Exige d'être capable de connaître la d'une donnée.
  - La dénormalisation et l'absence de schéma de données rend plus difficile.
- Exige de pouvoir garantir qu'une donnée anonymisée ne puisse pas être ré-identifiée.
  - En croisant les différentes collections distribuées, on augmente les chances de réussir à ré-identifier les données en croisant les informations.