Introduction au BIG DATA

Big Data

- Introduction à cette formation
  - Votre formateur … Et Vous
  - Le matériel
    - Le support de cours
    - La distribution Anaconda
  - L’organisation – horaires
    - Formation de 3 jours
  - La forme : cours, exercices et TP

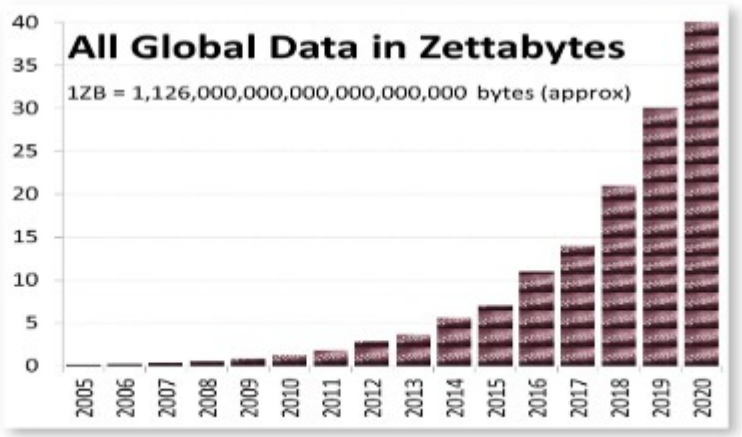
Big Data

- Des liens utiles
  - https://openclassrooms.com/fr/courses/4467491-concevez-des-architectures-big-data
  - https://chewbii.com/wp-content/uploads/2015/11/NoSQL_intro_handout.pdf
  - https://perso.liris.cnrs.fr/emmanuel.coquery/dokuwiki/lib/exe/fetch.php?media=enseignement:bdav:cm-postsgbdr.pdf

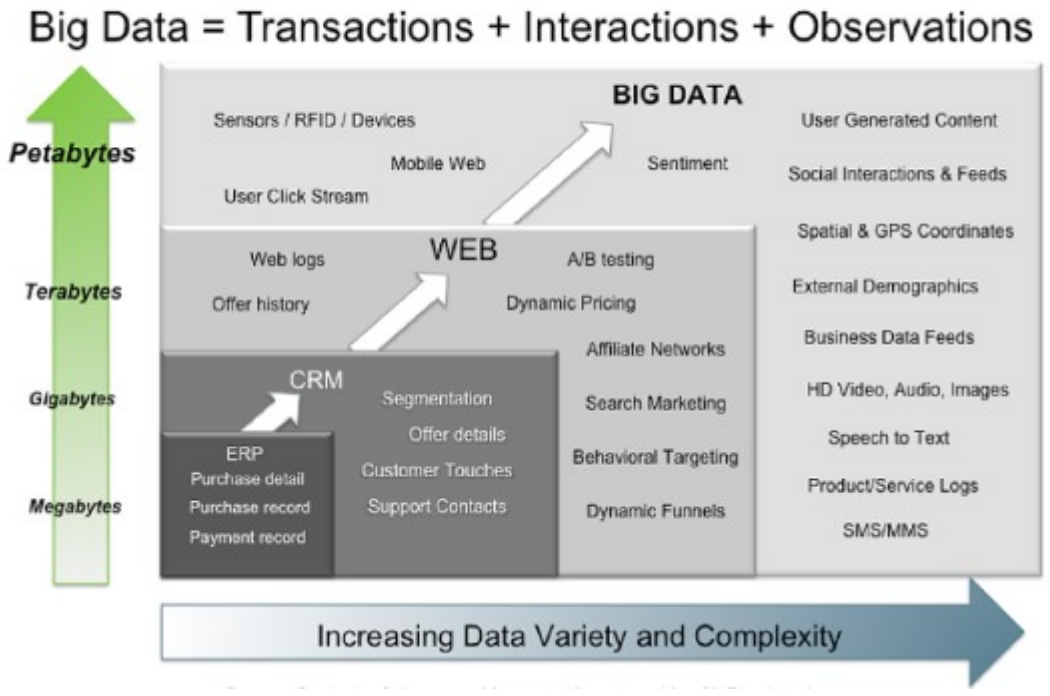
Sommaire

- Présentation de l’architecture Big Data
- Le sharding
- Architecture lambda
- Aperçu des bases NoSQL
- MongoDB
- MongoDB et Python

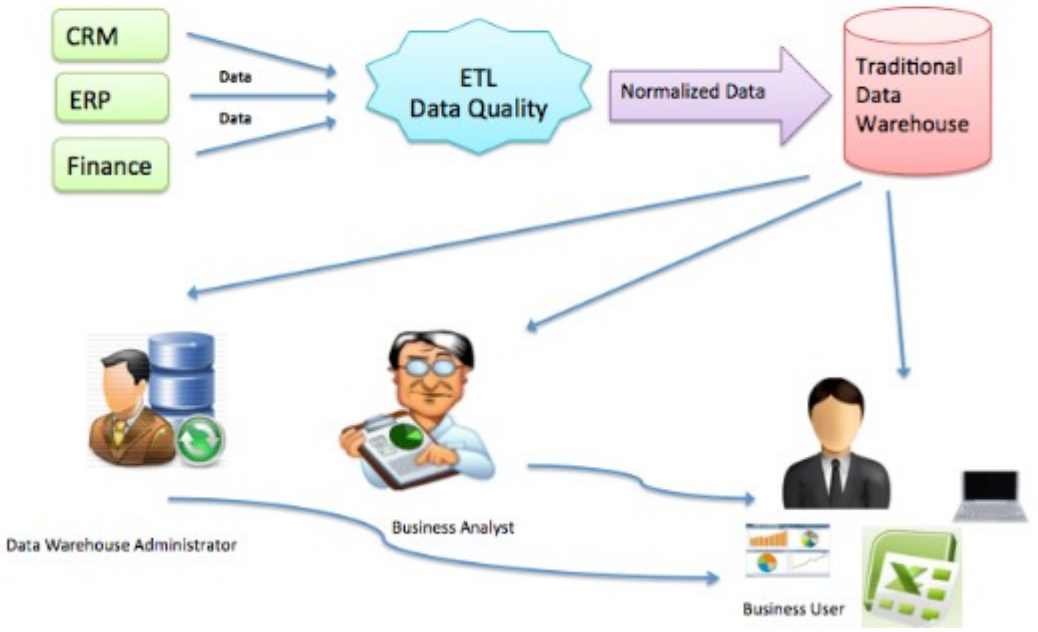
- La quantité de données digitales produites double tous les 2 ans.
- En d'autres termes, on a produit autant de données digitales ces 2 dernières années que tout ce qui a été produit auparavant.



- **Volume et Variété**



- **Décisionnel : ancienne méthode**



- ## Décisionnel vs 3V

  - L'approche classique incompatible avec les 3V du BigData :
  - Le **Volume**: les entrepôts sont conçus pour gérer des Go ou To de données alors que la croissance exponentielle des données nous conduit aux Po ou Eo
  - Le **type (Variety)**: le nombre de types, incluant les données textuelles semi ou non structurées, augmente
  - La **vitesse (Velocity)**: les données sont créées de plus en plus vite et nécessitent des traitements en temps-réel

-

SGBDR vs Distribution

- Fonctionnalités
  - Jointures entre les tables
  - Langage d'interrogation riche
  - Contraintes d'intégrité solides

- Limites dans le contexte distribué :
  - **Comment distribuer/partitionner les données**
  - Liens entre entités -> Même serveur
  - Mais plus on a de liens, plus le placement des données est complexe

- # SGBDR vs Distribution

ACID vs BASE

- Propriétés **ACID** pour les transactions
  - ***A**tomicité* : une transaction s'effectue entièrement ou pas du tout
  - ***C**ohérence* : le contenu d'une base doit être cohérent au début et à la fin d'une transaction
  - ***I**solation* : les modifications d'une transaction ne sont visibles/modifiables que quand celle-ci a validé
  - ***D**urabilité* : une fois la transaction validée, l'état de la base est permanent (non affecté par les pannes ou autre)
- Systèmes distribués : modèle **BASE**
  - ***B**asically **A**vailable* : garantie minimale pour taux de disponibilité face grande quantité de requêtes
  - ***S**oft-state* : l'état du système peut changer au cours du temps même sans nouveaux inputs (cela est du au modèle de consistance).
  - ***E**ventually consistent* : tous les réplicas atteignent le même état, et le système devient à un moment consistant, si on stoppe les inputs

-

NoSQL : une solution

- NoSQL : **Not Only SQL**
  - Nouvelle approche de stockage et de gestion de données
  - Permet le passage à l'échelle via un contexte hautement distribué
  - Gestion de données complexes et hétérogènes
    - Pas de schéma pour les objets
- Ne remplace pas les SGBDR !!
  - Quantité de données énorme (PétaBytes)
  - Besoin de temps de réponse
  - Cohérence de données faible

- Le traitement des mégadonnées – le Big Data – nécessite des architectures matérielles et logicielles adaptées
  - Pour traiter le volume
  - Pour répondre dans les meilleurs délais
  - Pour s’adapter aux changements

- Les critères à atteindre pour une telle architecture :
  - La tolérance aux pannes
    - Matériels de qualité avec MTBF (mean time between failures) élevé
    - Redondance d’équipements
  - Maintenabilité facile
    - Le système a du être livré avec peu de dette technique
    - Conçu de façon modulaire
    - Scalable – l’augmentation en capacité et puissance est possible

- Les critères à atteindre pour une telle architecture (suite) :
  - Un coût maîtrisé
    - Choix de conception pour utiliser des composants, bibliothèques et framework existants
    - Concevoir le système avec le critère de scalabilité pris en compte. Augmenter la puissance des machines n’est pas toujours le moins cher.

- Le Data Architect est né !
  - https://www.fichemetier.fr/metiers/data-scientist
  - https://www.orientation.com/metiers/data-architect#:~:text=Salaire.%202600%20%E2%82%AC%20-%205200%20%E2%82%AC%20net%2Fmois.%20Le,stockage%2C%20de%20manipulation%20et%20de%20restitution%20des%20donn%C3%A9es
    .
  - Il met à disposition des Data Scientist des lots de données de plus en plus gros, de plus en plus vite !
  - Il conçoit et maîtrise l'architecture matérielle et logicielle du système

Problématique

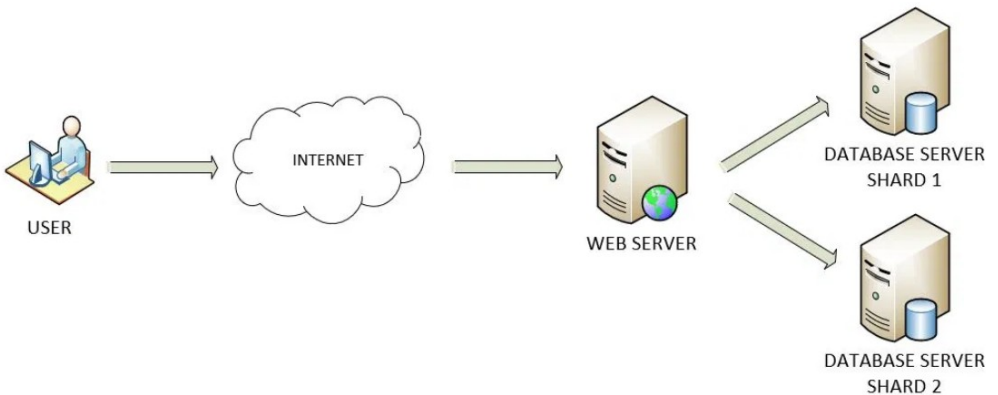
- Sur un site Web très utilisé on souhaite réaliser des métriques et des enquêtes, pour améliorer la force de vente
  - Un « Google Analytics » amélioré
  - Des Data Scientists ont été recrutés
  - Ils ont besoin de données

- Des pistes :
  - Créer des tables spécifiques MySQL ou Oracle dans notre application
  - Notre serveur Web réalise les transactions pour ces tables

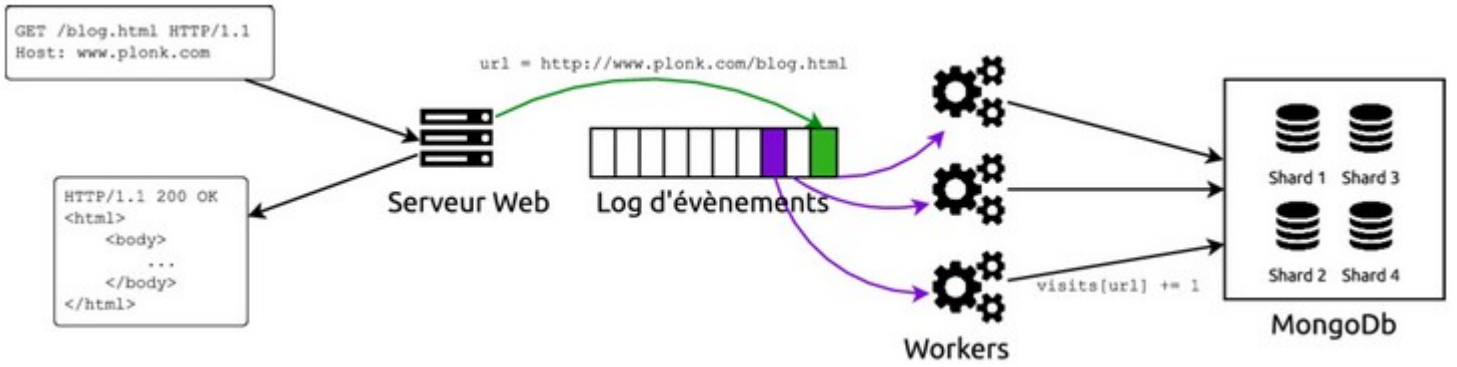
- Techniques connues
- Fait partie de l’application

- Les bases de données relationnelles ne sont pas faites pour de nombreuses écritures concurrentes et rapides
- Le volume va être très vite important

- La technique du Sharding
  - Sharding : éclatement, partitionnement pour une base de données
  - La partie de la BD qui pose problème (volume et temps d'accès) est tronçonnée et répartie sur plusieurs bases
  - Les « data shards » - jeux de données partitionnés - sont sur plusieurs serveurs, de coûts inférieurs à un gros serveur performant.
  - Si un des serveurs est en maintenance, l'ensemble des services peut être indisponible ...



- Pour implémenter ce sharding, il faut utiliser des gestionnaires de base de données qui l'intègrent :
  - Base de données NoSQL : plus rapides et gèrent de gros volumes
  - Gèrent le sharding : MongoDB, Elasticsearch

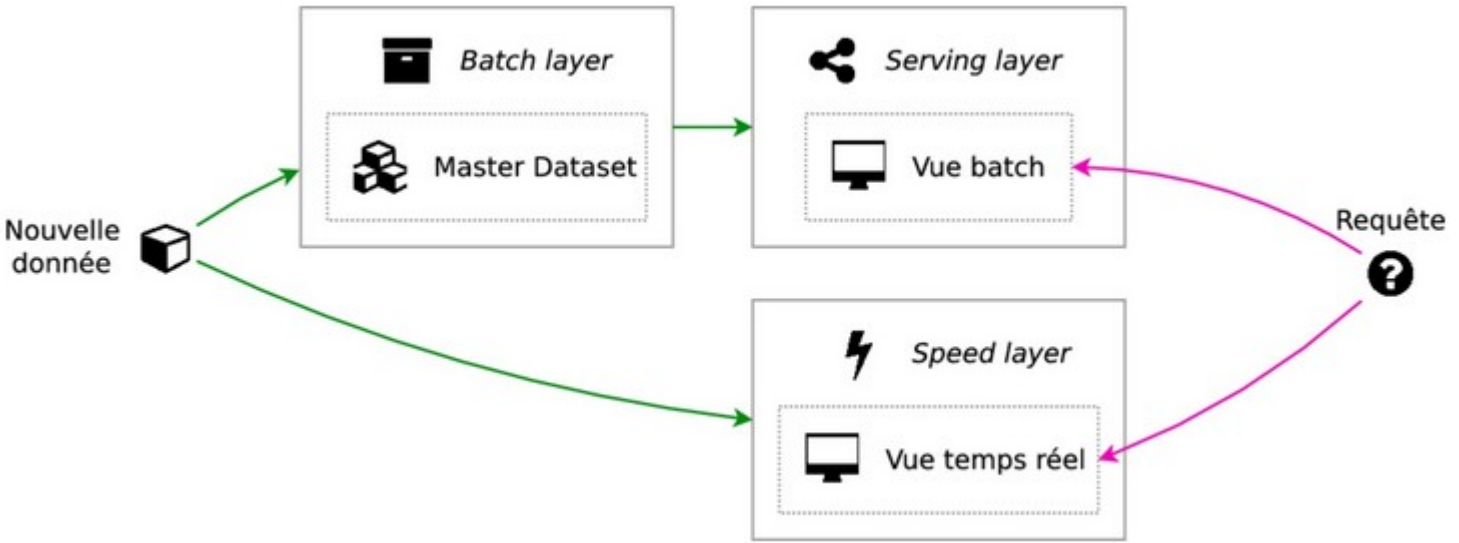


- Avantages :
  - Passage à l’échelle (agrandissement) possible sans problème :
    - Ajout de nouveaux workers
    - Augmenter le sharding
- Inconvénients
  - Sur bug logiciel ayant conduit à fausser un shard, pas de moyen de revenir en arrière

=> Les bases NoSQL ne sont qu’une partie de la solution

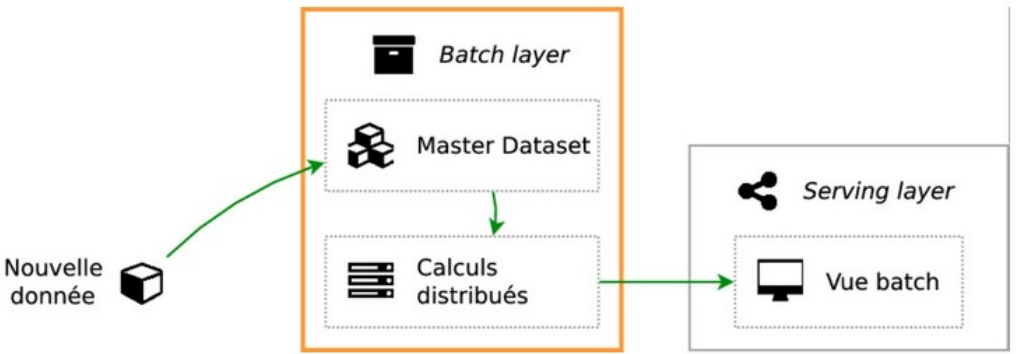
- L’architecture lambda propose une solution d’ensemble pour les architectures Big Data
  https://www.manning.com/books/big-data
- Elle intègre les contraintes suivantes :
  - Passage à l’échelle : l’architecture doit pouvoir passer à l’échelle de manière horizontale, en ajoutant des serveurs
    (la manière verticale signifie augmenter la capacité et performance d’un seul serveur, de plus en plus coûteux)
  - Facilité de maintenance : choix de composants standards, architecture évolutive
  - Facilité d’exploitation des données : stocker un très gros volume de données mais aussi les mettre à disposition à d’autres applications, avec des contraintes de performances élevées

- Décomposition en 3 couches

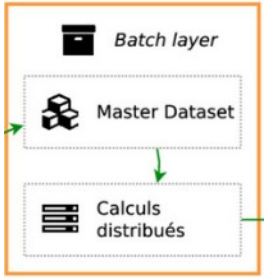


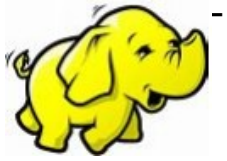
Source openclassroom

- Le Batch Layer
  - Stocke les données massives (nombreuses et fréquentes) récoltées
  - Effectue des calculs sur ces données



- Le Master Dataset est un référentiel des données brutes
  - C'est un des composants d'un Data Lake
    https://openclassrooms.com/fr/courses/4467481-creez-votre-data-lake
    https://soat.developpez.com/tutoriels/bigdata/datalakes-architecture-big-data/





  - Pour être efficace et bon marché, l'implémentation du Dataset ne se fait pas par un SGBD (trop lent) ou par du NoSQL
    L'implémentation se fait par le système de fichiers distribués HDFS – Hadoop Distributed File System

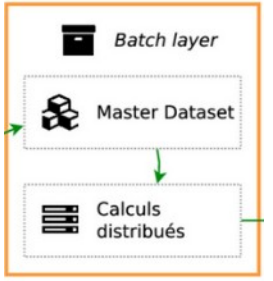
  - Les données stockées dans le HDFS sont sérialisées – Apache Avro permet ceci. Embarque le schéma des données. https://avro.apache.org/
  - Les données du Dataset ne peuvent pas être supprimées ou modifiées

- Master DataSet (suite)
  - Les données stockées doivent elles être normalisées ou dénormalisées ?
    Normalisées : pas de duplication de champ, utilisation de foreign key
    Dénormalisées : tout est “à plat” quitte à répéter des informations.

    Le format Normalisé est choisi en général pour éviter les duplications.

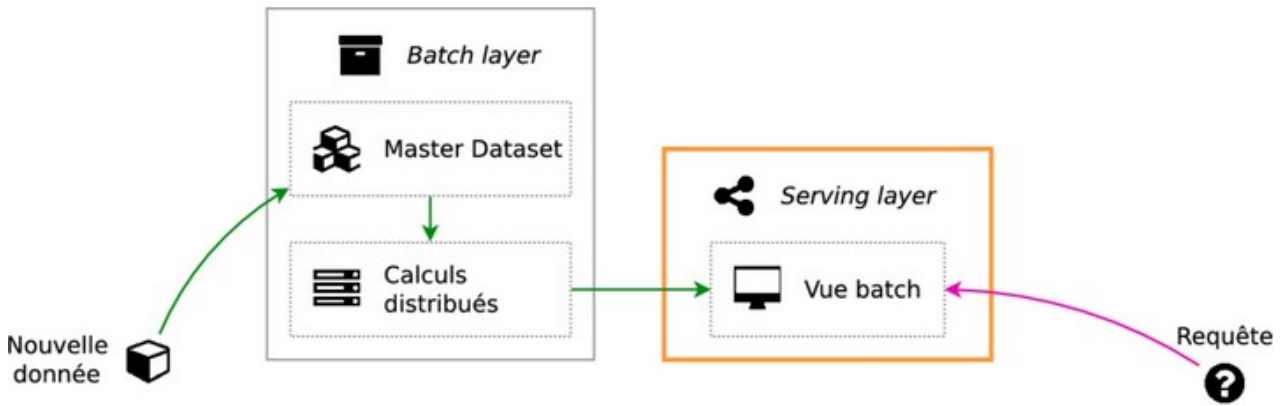
  - Le stockage des données est séquentiel (écriture en bout, en append) et non pas en aléatoire (comme une BD SQL par exemple)

- Calculs distribués
  - Ces processus vont lire le Master Dataset – utilisant Hadoop et Apache Avro – et réaliser des calculs distribués
  - Traitement par lots (suite à un volume reçu), sur échéances (toutes les n Heures).
  - Sur grossissement du volume un passage à l'échelle doit pouvoir se faire de façon horizontale (ajout de machines plutôt que de grossir les capa/perf d'une machine). L'architecture MapReduce permet de traiter de très gros volumes avec des clusters de machines. Hadoop et Spark sont des supports



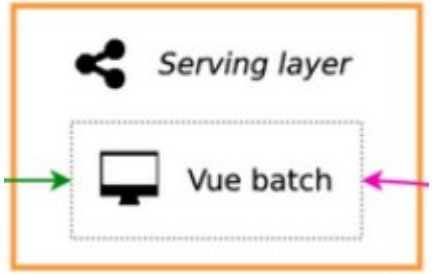
https://openclassrooms.com/fr/courses/4297166-realisez-des-calculs-distribues-sur-des-donnees-massives

- Serving Layer
  - Stocke et met à disposition aux utilisateurs les données traitées par lots par la couche Batch layer, avec le traitement “Calculs distribués”

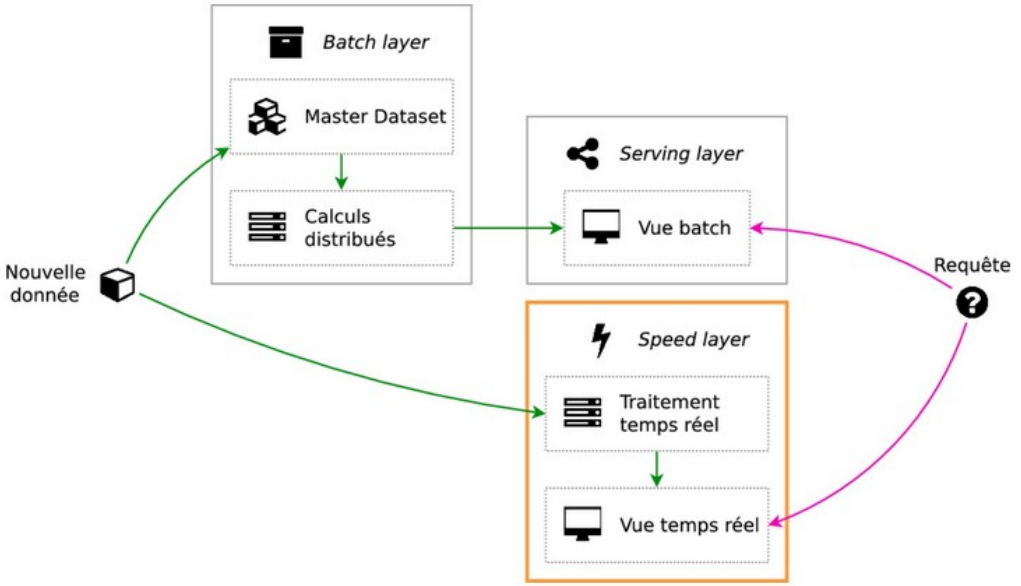


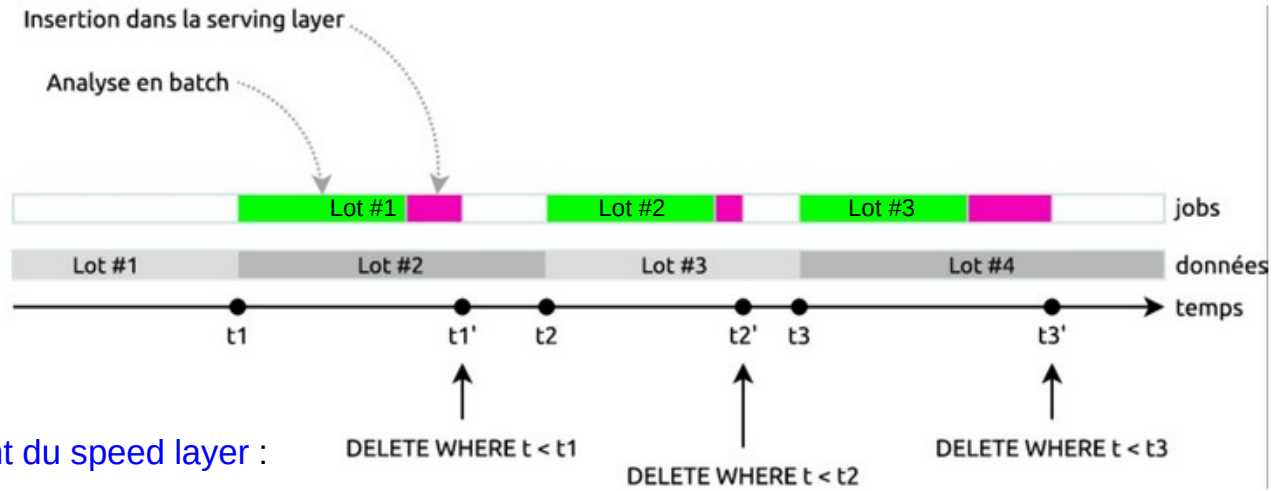
- Le support base de données ne peut pas être une BD SQL :
  Pas de passage à l'échelle horizontale pour ce type, elle nécessite d'augmenter la puissance du serveur SGBD
- Une base NoSQL est à choisir
  Cassandra, MongoDB, ElasticSearch

  https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql



- Le traitement par lots – à échéances convenues - est fait par le Batch layer et met à disposition les données au Serving layer
- Les données les plus récentes, pas encore dans l'échéance, peuvent être traitées par le Speed layer





Arrivée dans le DataSet:

1

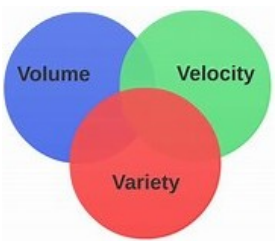
Enlèvement du speed layer :

3

- Données de la Speed Layer
  - Elle peuvent être agrégées - alors que les données du Master Dataset sont brutes
    « Agrégé » signifie que le dernier état d'une info est sauvegardé, un calcul entre plusieurs infos est fait …
  - Elle peuvent être dénormalisées
  - Une base NoSQL est un support possible

- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql

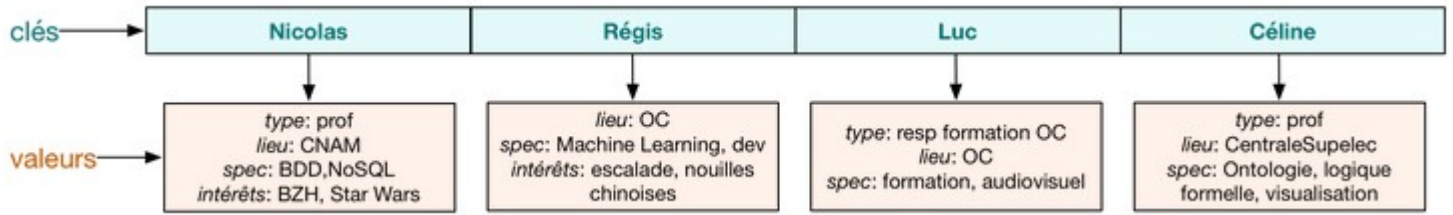
- Not Only SQL : il n’y a pas que le SQL
- Face aux 3V – Volume, Velocity, Variety – le modèle relationnel n’est pas adapté



variety signifie ici la diversité des données vs rigidité d’un modèle SQL

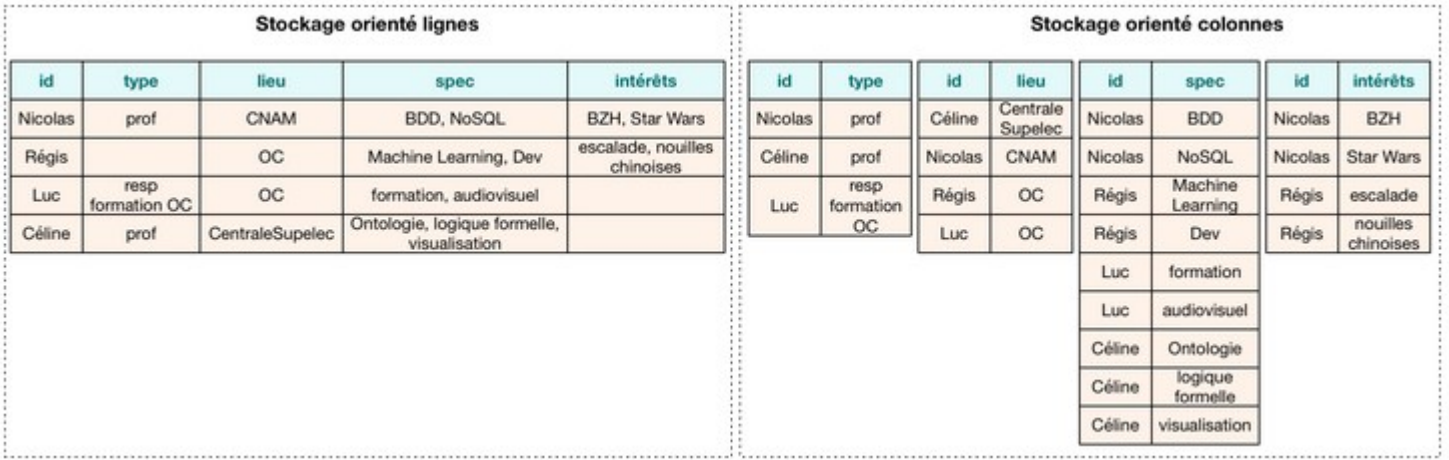
Les types de BD NoSQL :

- Les clés-valeurs : simple et efficace, sorte de grande table de hachage distribuée sur un réseau, entre N serveurs.
  Aucun schéma, aucune structure, pas de langage SQL



SimpleDB (Amazon)
Azure Cosmos DB (Microsoft)
...

- Stockage orienté colonnes



Stockage orienté lignes

id	type	lieu	spec	intérêts
Nicolas	prof	CNAM	BDD, NoSQL	BZH, Star Wars
Régis		OC	Machine Learning, Dev	escalade, nouilles chinoises
Luc	resp formation OC	OC	formation, audiovisuel	
Céline	prof	CentraleSupelec	Ontologie, logique formelle, visualisation	

Stockage orienté colonnes

id	type
Nicolas	prof
Céline	prof
Luc	resp formation OC

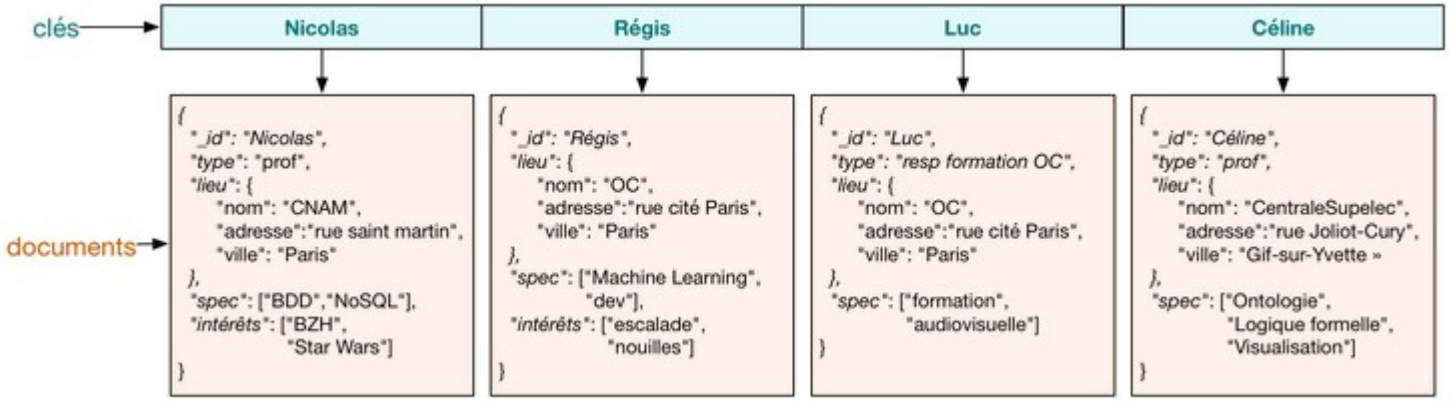
id	lieu
Céline	Centrale Supelec
Nicolas	CNAM
Régis	OC
Luc	OC

id	spec
Nicolas	BDD
Nicolas	NoSQL
Régis	Machine Learning
Régis	Dev
Luc	formation
Luc	audiovisuel
Céline	Ontologie
Céline	logique formelle
Céline	visualisation

id	intérêts
Nicolas	BZH
Nicolas	Star Wars
Régis	escalade
Régis	nouilles chinoises

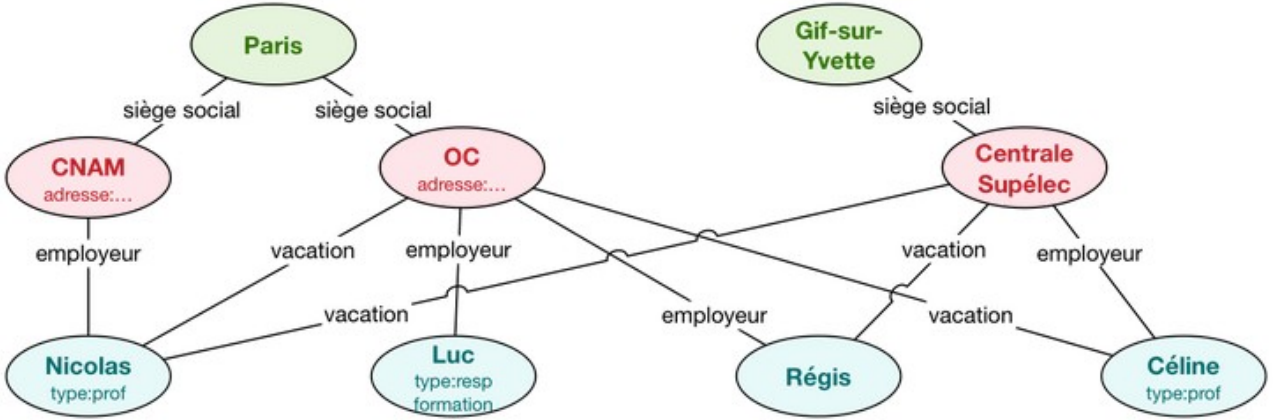
- Adapté pour des traitements sur les colonnes (somme, moyenne,…)
- BigTable (Google), Hbase (Apache, Hadoop), Elasticsearch (elastic)

- Bases orientées documents
  - Repose sur le type clé/valeur avec un document pour valeur



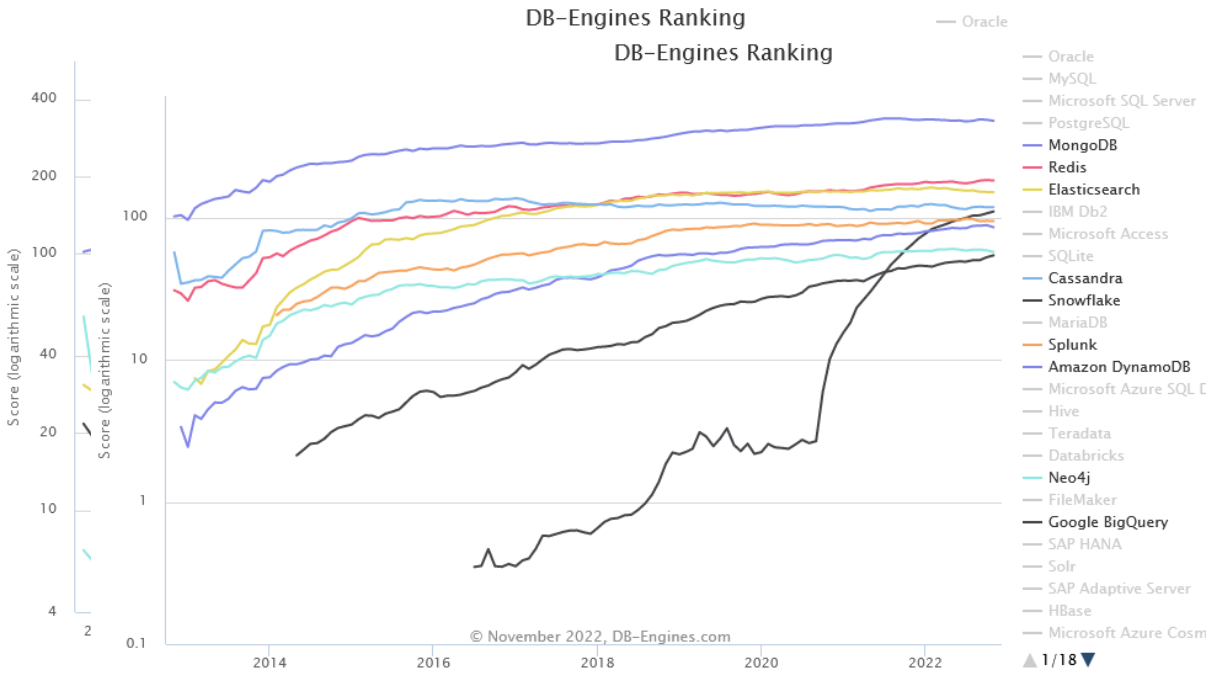
  - La valeur est structurée. Des langages d'interrogation peuvent exister
  - MongoDB, CouchBase, DynamoDB, Cassandra

- Les graphes



- Les données sont constituées par les nœuds et les liens.
- Requêtes d'interrogation basées sur la gestion des chemins
- Utilisés pour les réseaux sociaux, réseaux routier, électrique
- Neo4j, OrientDB, FlockDB

- Le choix d'une BD SQL se fait en fonction de son type et facilité de mise en œuvre, au regard du cas à traiter
  - Site https://db-engines.com/en/ranking_trend



- MongoDB est mis à disposition sous deux formes :
  - En local sur des machines
  - Sur le cloud : MongoDB Atlas
- Nous proposons une initiation à MongoDB en local:
  - Sa mise en place
  - Utilisation en mode console
  - Utilisation de MongoDB Compass
  - Interface avec Python

=> Utiliser le fichier Mongodb.pdf