NoSQL

1 - Nouveaux besoins en gestion des données

Le NoSQL, pour "**not only SQL**", désigne les bases de données qui ne sont pas fondées sur l'architecture classique des bases de données relationnelles. Développé à l'origine pour gérer du **big data**, l'utilisation de base de données NoSQL a explosée depuis quelques années. Mais qu'est-ce que réellement le NoSQL ?

Le big data est caractérisé par les 3V (Gardner)

- **Volume – high volume**
  Le volume décrit la quantité de données générées par des entreprises ou des personnes. Le Big Data est généralement associé à cette caractéristique. Les entreprises, tous secteurs d'activité confondus, devront trouver des moyens pour gérer le volume de données en constante augmentation qui est créé quotidiennement.
- **Vitesse (rapidité) – high Velocity**
  La vitesse décrit la fréquence à laquelle les données sont générées, capturées ,visualisées et partagées. Du fait des évolutions technologiques récentes, les consommateurs mais aussi les entreprises génèrent plus de données dans des temps beaucoup plus courts.
- **Variété- high variety**
  La prolifération de types de données provenant de sources comme les médias sociaux, les interactions *Machine to Machine* et les terminaux mobiles, crée une très grande diversité au-delà des données transactionnelles traditionnelles. Les données ne s'inscrivent plus dans des structures nettes, faciles à consommer.
  Les nouveaux types de données incluent contenus, données géo spatiales, points de données matériels, données de géolocalisation, données de connexion, données générées par des machines, données de mesures, données mobiles, points de données physiques, processus, données RFID, données issues de recherches, données de confiance, données de flux, données issues des médias sociaux, données texte et données issues du Web.

A ce concept, on peut ajouter 4 nouveaux paramètres

- **Véracité - veracity**
  Avoir beaucoup de données dans des volumes différents à grande vitesse est sans valeur si ces données sont incorrectes. Des données incorrectes peuvent causer beaucoup de problèmes aux organisations et aux consommateurs. Par conséquent, les organisations doivent s'assurer que les données sont correctes et que les analyses effectuées sur les données sont correctes. Surtout dans la prise de décision automatisée, où aucun humain n'est plus impliqué
- **Variabilité - variability**

  le big data est extrêmement variable. On définit la variabilité comme la "variance de sens dans le lexique". Pour effectuer des analyses de sentiment appropriées, les algorithmes doivent être capables de comprendre le contexte et de déchiffrer la signification exacte d'un mot dans ce contexte. C'est toujours très difficile.

- **Visualisation - visualisation**

1

C'est la partie difficile des données volumineuses. Faire en sorte que toutes ces données soient compréhensibles d'une manière facile à comprendre et à lire. Avec les bonnes analyses et visualisations, les données brutes peuvent être utilisées pour des données brutes, sinon les données brutes restent essentiellement inutiles.

- **Valeur - value**

  Toutes ces données disponibles créeront beaucoup de valeur pour les organisations, les sociétés et les consommateurs. Le Big Data signifie que les grandes entreprises et tous les secteurs tireront profit des données volumineuses .

1 – 1 - Les bases de données NoSQL

Représentent principalement les données du big data

Les propriétés **BASE** ont été proposées pour caractériser les bases NoSQL :

- **B*asically Available*** : quelle que soit la charge de la base de données (données/requêtes), le système garantie un taux de disponibilité de la donnée
- ***Soft-state*** : La base peut changer lors des mises à jour ou lors d'ajout/suppression de serveurs. La base NoSQL n'a pas à être cohérente à tout instant
- **E*ventually consistent*** : À terme, la base atteindra un état cohérent

Avantages :

- L'évolutivité se fait de manière horizontale (pour augmenter les performances on ajoute des nouvelles machines)
- Les données sont distribuées sur plusieurs machines (sharding) de ce fait on évite les goulets d'étranglements lors de la récupération des données (fortes performances de lecture)
- La représentation des données est notable par l'absence de schéma (schemaless)
- La majorité des solutions est Open Source, néanmoins il existe des Support Pro pour répondre aux besoins des entreprises.

Inconvénients :

- Il n'existe pas de langage d'interrogation standardisé : chaque éditeur a mis en place le sien
- La mise en œuvre d'un environnement fortement transactionnel (fort besoin d'écriture) où le séquencement des écritures est primordial, reste complexe puisque l'architecture est distribuée compliquant l'atomicité et la cohérence des transactions
- L'écriture de requêtes complexes est difficile à mettre en œuvre
- L'offre NoSQL est segmentée en plusieurs familles où chacune répond à un besoin précis.

1 – 2 - Les bases de données relationnelles

2

Avantages :

- La technologie est mature (création il y a plusieurs dizaines d'années) ce qui fait qu'aujourd'hui le SQL est un langage standard et normalisé
- On a une garantie que les transactions sont atomiques, cohérentes, isolées et durables — principe **ACID (Atomic, Consistent, Indépendant, Durable)**
    - 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 été validée
    - D***urabilité*** : Une fois la transaction validée, l'état de la base est permanent (non affecté par les pannes ou autre)

- La possibilité de mettre en œuvre des requêtes complexes (croisement multiple des données)
- Du fait du nombres d'années d'existence, un large support est disponible et il existe également de fortes communautés.

Inconvénients :

- La modification du modèle établi peut être couteuse
- L'évolutivité des performances est privilégiée de manière verticale (augmentation des ressources du serveur) bien qu'une évolutivité horizontale soit possible, cette dernière reste plus coûteuse (environnement type cluster)
- Sur un très grand volume de données (centaines-milliers de Téraoctets) le modèle peut atteindre des limites en termes de performance
- Pour certains éditeurs, le prix de licence est élevé.

1 – 3 – Le stockage

Constat :

• essor des très grandes plateformes et applications Web (Google, Facebook, …)

• volume considérable de données à gérer par ces applications nécessitant une distribution des données et leur traitement sur de nombreux serveurs : « **Data Centers** »

• ces données sont souvent associées à des objets complexes et

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

3

• permettant une meilleure scalabilité dans des contextes fortement distribués

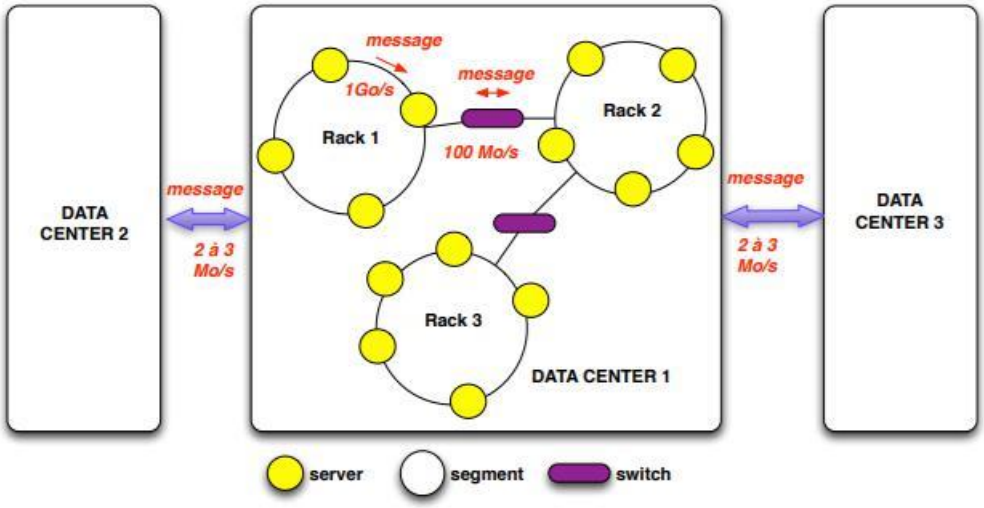
• permettant une gestion d'objets complexes et hétérogènes sans avoir à déclarer au préalable l'ensemble des champs représentant un objet

• regroupées derrière le terme NoSQL (proposé par Carl Strozzi), ne se substituant pas aux SGBD Relationnels mais les complétant en comblant leurs faiblesses (Not Only SQL)

Data Centers

• Utilisent des LANs (Local Area Networks) avec 3 niveaux de communication :

- Les **serveurs** sont regroupés en " **Racks** " : liaison réseau rapide, environ 1Go/sec
- un " **Data center** " consiste en un grand nombre de " **racks ",** interconnectés par des routeurs (switches) : liaison à 100 Mo/sec
- Entre différents " **Data centers "** : communication internet à 2-3 Mo/sec
- Les serveurs communiquent par envoi de messages, ils ne partagent pas de disque ni de ressource de traitement = architecture " shared nothing "



La distribution de données sur plusieurs serveurs organisés en " data centers "

- Gestion de volumes de données très importants,
- Assurer une continuité de service en cas d'indisponibilité de service sur un serveur

Trois techniques de répartition des données

L'élasticité

L'*élasticité*, 'est la capacité du système à s'adapter automatiquement en fonction du nombre de serveurs qu'il dispose et de la quantité de données à répartir. Par exemple, vous avez 1To de données sur 1000 serveurs (1Go par serveur), lors d'un pic d'activité (soldes, période de

4

Noël…), il serait utile de rajouter 1000 serveurs pour répartir la charge de calcul avec 500 Mo par serveur. L'élasticité va permettre de **répartir** uniformément les données sur les 2000 serveurs (déplacement de la moitié des données), et inversement lorsque le pic d'activité s'achève. De même, si le volume de données augmente et atteint 2To, l'élasticité garantira une répartition uniforme de 2Go par serveur.

Le sharding

Le *sharding* est une technique permettant de distribuer des **chunks** (morceaux de fichiers) sur un ensemble de serveurs, avec la capacité de gérer l'élasticité (serveurs/données) et la tolérance aux pannes. Trois familles de distribution pour le NoSQL existent : *HDFS* (basé sur la distribution), *le clustered index* (basé sur le BTree) et le *consistent hashing* (basé sur les tables de hachage). Regardons comment elles fonctionnent, elles nous permettront d'orienter nos choix d'architecture.

HDFS

HDFS (*Hadoop Distributed File System*) est une technique de distribution de fichiers volumineux. Chaque fichier sera découpé en "chunk" de 64Mo pour être distribué sur le réseau. Chaque serveur de ce réseau est un *datanode* contenant plusieurs chunks. La répartition de ces chunks est définie par le serveur central, le *namenode*.

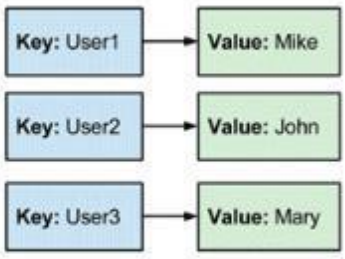
Chaque chunk est répliqué sur 3 serveurs distincts ; de fait, si un serveur tombe en panne, la donnée peut tout de même être récupérée. Nous pouvons constater également que le **namenode** est lui-même répliqué avec un s*econdary namenode*. Celui-ci permet au serveur central de redémarrer rapidement en cas de panne.

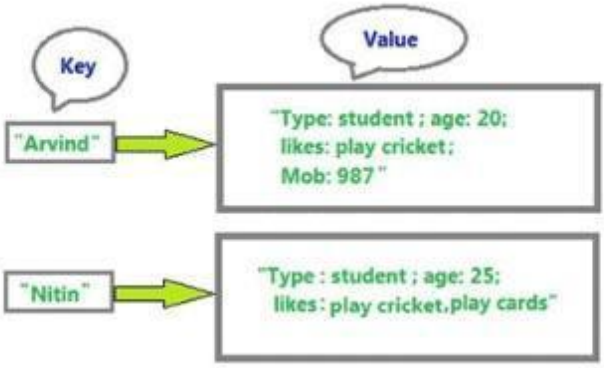
II – Type de base de données NoSQL

- Clé-valeur
- Documents
- Colonnes
- Graphes

Un objet **Blob** représente un objet, semblable à un fichier, qui est immuable et qui contient des données brutes. Les **blobs** (pour B inary L arge Ob jects ...)

Les données sont stockées en clé-valeur : une clé plus un BLOB (dans lequel on peut mettre : nombre, date, texte, JSON,XML, photo, vidéo, structure objet).





5

Les plus

- Facilement sécable
- Temps d'écriture/lecture très bas
- disponibilité

Les moins

- Mise à jour compliqué
- Requêtes rudimentaires : interrogation sur une clé

Implémentations

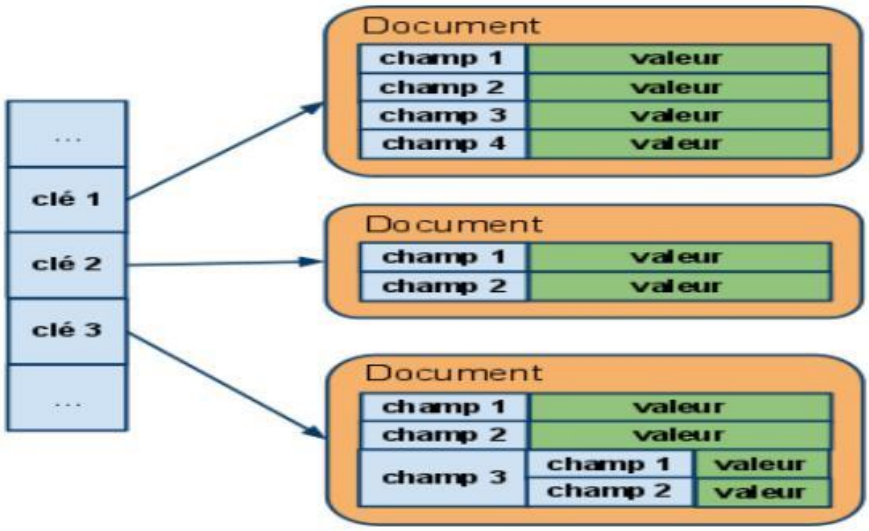
- Amazon : simpleDB
- DynamoDB
- Redis (VMware)
- Azure CosmosDB (Microsoft)
- Memcached (Danga)
- Voldemort (développement LinkedIn puis open source)

Les seules opérations de type **CRUD** peuvent être utilisées :

- Create (key,value)
- Read (key)
- Update (key,value)
- Delete (key)

II – 2 – Type 2 : Bases orientées Document

Ces bases de données stockent des données semi-structurées : le contenu est formaté JSON ou XML, mais la structure n'est pas contrainte.



6

Les plus

- Modèle de données simple mais puissant
- Requêtes plus complètes
- Flexibilité
- Evolutif au cours du temps

Les moins

- Duplication des données
- Cohérence difficile (pas de modèles interconnectées)
- Modele limité à des clés

Implémentations

- MongoDB
- CouchDB (apache)
- Cassandra (facebook $\longrightarrow$ Apache))
- DynamoDB (amazon)

utilisation

- Enregistrement d'événements
- Systèmes de gestion de contenu
- Web analytique ou analytique temps-réel
- Catalogue de produits
- Systèmes d'exploitatio
- …

II – 3 :Bases orientées COLONNE

Ces bases de données se rapprochent des bases de données relationnelles, à ceci près qu'elles permettent de remplir un nombre de colonnes variable.

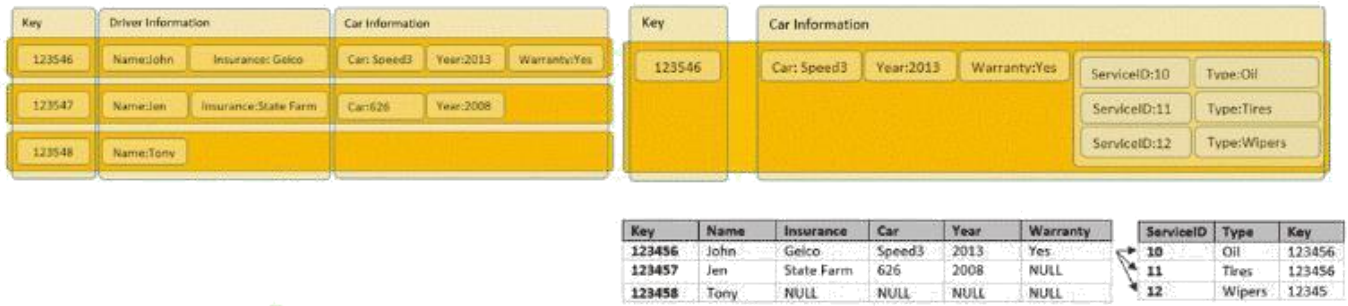
Les principaux concepts associés sont les suivants :

- **Colonne** :
  - Entité de base représentant un champ de donnée
  - Chaque colonne est définie par un couple clé / valeur
  - Une colonne contenant d'autres colonnes est nommée **supercolonne**.

- **Famille de colonnes** :
  - Permettent de regrouper plusieurs colonnes (ou supercolonnes)
  - Les colonnes sont regroupées par ligne
  - Chaque ligne est identifiée par un identifiant unique (assimilées aux tables dans le modèle relationnel) et sont identifiées par un nom unique

- **Supercolonnes** :
  - Situées dans les familles de colonnes sont souvent utilisées comme les lignes d'une table de jointure dans le modèle relationnel.

7



Les plus

- Capacité de stockage accrue
- Accès rapide aux données

Les moins

- Requêtes limitées
- Limitation du nombre de types d' objets

Implémentation

- Apache :HBASE
- BigTable (Google)
- Spark SQL (Apache)
- ElasticSearch (elastic)

Utilisation

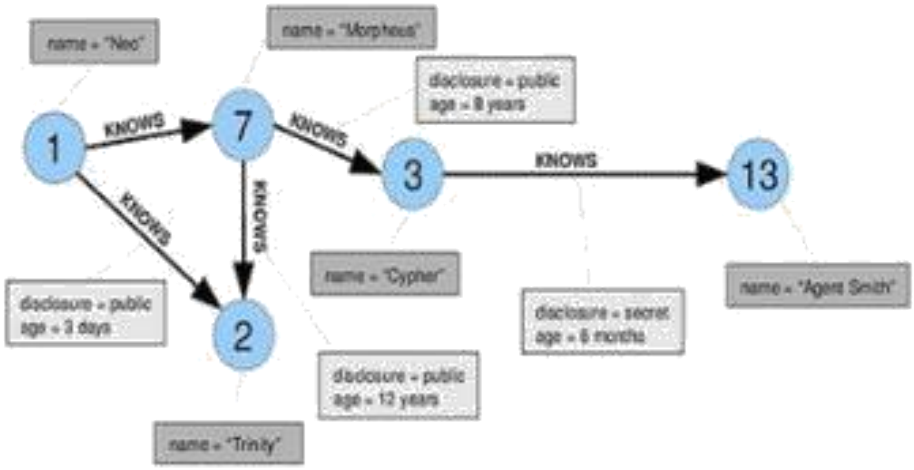
Les BD NoSQL type « Colonne » sont principalement utilisées pour :

- 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
- § …

II – 4 – Bases de données orientées GRAPHE'

Ces bases de données, basées sur la théorie des graphes, sont gérées par nœuds, relations et propriétés. Elles gèrent des données spatiales, sociales ou financières (dépôts/retraits).

8



<u>Utilisation en Web Semantique avec magasins de Triplet</u> (codage RDF sujet-prédicat-objet)

Les plus

- Adaptées à la gestion des données relationnelles
- Architecture modulable

Les moins

- Limitées à certains cas

Implémentations

- TITAN
- Neo4j
- OrientDB (Apache)

Utilisation

• Les BD NoSQL type « Graphe » sont principalement utilisées pour :

- Moteurs de recommandation
- Business Intelligence (BI)
- Web sémantique
- Social computing
- Données géospatiales
- Généalogie
- Web of things
- Catalogue des produits
- Sciences de la Vie et calcul scientifique (bio-informatique, …)
- Données liées, données hiérarchiques § Services de routage, d'expédition et de géolocalisation
- Services financiers : chaîne de financement, dépendances, gestion des risques, détection des fraudes, …

III <u>– THEOREME CAP (Brewer)</u>

9

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

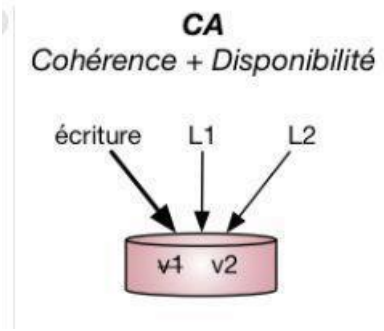
1. **C***onsistency* (**Cohérence**) : Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas
2. **A***vailability* (**Disponibilité**) : Tant que le système tourne (distribué ou non), la donnée doit être disponible
3. **P***artition Tolerance* (**Distribution**) : Quel que soit le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

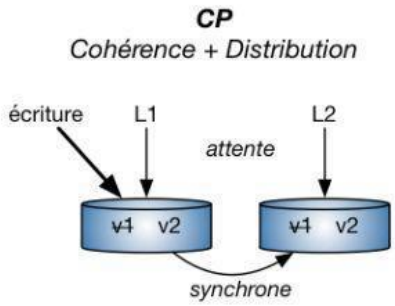
Cela s'illustre assez facilement avec les bases de données relationnelles, elles gèrent la cohérence et la disponibilité, mais pas la distribution.

Couple CA (Consistency – Available)



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. Cette combinaison n'est possible que dans le cadre de bases de données transactionnelles telles que les SGBDR.

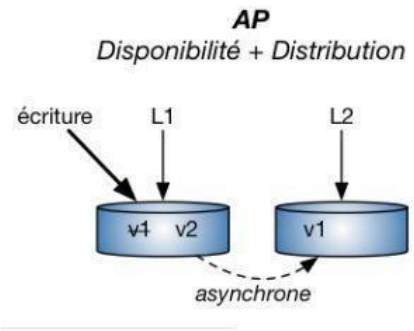
Couple CP (Consistency-partition**)**



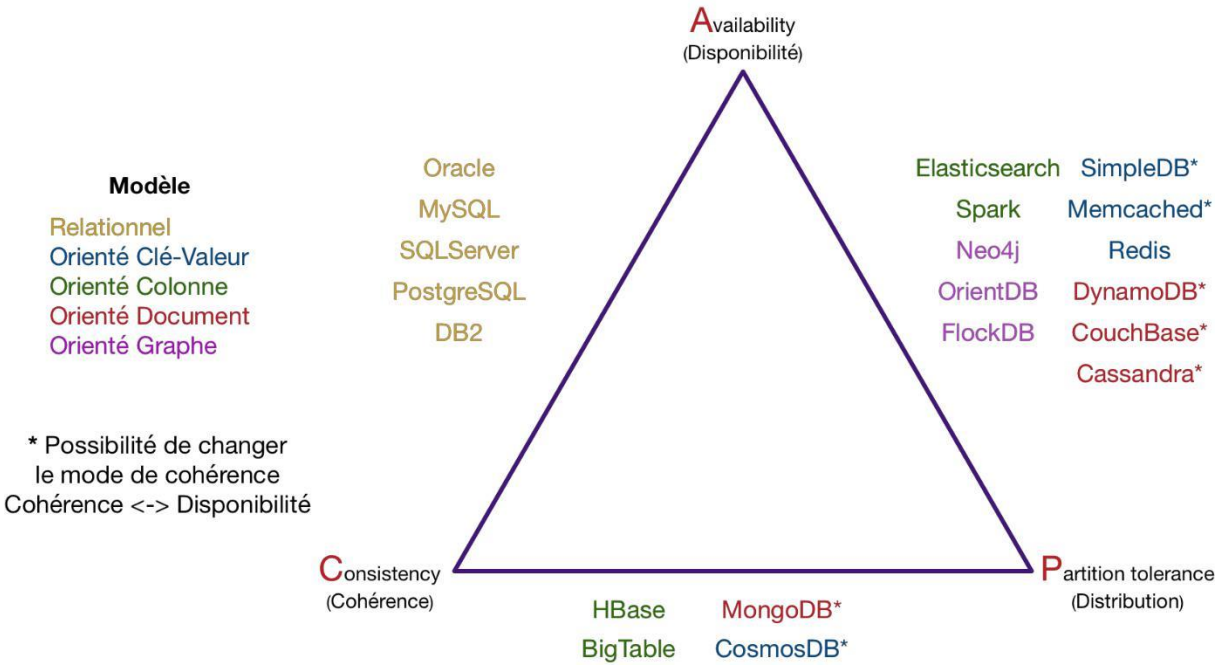
Distribution sur plusieurs serveurs en garantissant la tolérance aux pannes (réplication). En même temps, il est nécessaire de vérifier la cohérence des données en garantissant la valeur retournée malgré des mises à jour concurrentielles. La gestion de cette cohérence nécessite un protocole de synchronisation des réplicas, introduisant des délais de latence dans les temps de réponse (L1 et L2 attendent la synchronisation pour voir v2).

10

Couple AP (Available-partition**)**



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*" (L1 voit la version v2, tandis que L2 voit la version v1). 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.



***Extrait de Openclassroom.com**

III – Principaux fondements des systèmes NnSQL

- **Le « Sharding** » : un partitionnement des données sur plusieurs serveurs un ensemble de techniques qui permet de répartir les données sur plusieurs machines pour assurer la scalabilité de l'architecture.
- Le **« Consistent hashing**» : un partitionnement des données sur plusieurs serveurs eux-mêmes partitionnés sur un segment, Mécanisme de partitionnement (horizontal) dans lequel les objet-données sont stockés sur des nœuds-serveurs différents en utilisant la même fonction de hachage à la fois pour le hachage des objets et le hachage des nœuds :
- Le « **Map Reduce** » : un modèle de programmation parallèle permettant de paralléliser tout un ensemble de tâches à effectuer sur un ensemble de données,

11

- Le « **MVCC** » : pour « Contrôle de Concurrence Multi-Version », est un mécanisme permettant d'assurer le contrôle de concurrence,

  Méthode de contrôle de concurrence couramment utilisée par les SGBD pour gérer des accès simultanés à la base de données avec mises à jour
- Le « **Vector-Clock**» : ou horloges vectorielles permet des mises à jours concurrentes en datant les données par des vecteurs d'horloge

  Les ensembles de données répartis sur nœuds peuvent être lus et modifiés sur chaque nœud et aucune cohérence stricte n'est assurée par des protocoles de transactions distribuées

V – Evolution

V – 1 – NewSQL : amélioration des performances grâce à de nouveaux moteurs de stockage, des technologies transparentes de fragmentation, de nouveaux logiciels et matériels : des BD radicalement nouvelles

NewSQL est une catégorie de bases de données SQL qui s'attaque aux problèmes de performances et d'évolutivité que posent les bases de données relationnelles pour le traitement transactionnel (OLTP).

Ces systèmes ont pour but d'atteindre l'évolutivité des systèmes **NoSQL** tout en conservant les attributs **ACID** (Atomicité, Cohérence, Isolation et Durabilité) que garantissent les bases de données classiques.

Les bases de données NewSQL sont essentiellement destinées aux entreprises qui gèrent des données sensibles et stratégiques et qui ont besoin d'évolutivité, mais aussi d'une cohérence supérieure à ce qu'apportent les bases NoSQL.

Si les diverses bases de données NewSQL diffèrent par leurs architectures internes, elles exploitent toutes le modèle de données relationnel et s'exécutent toutes en langage SQL.

Deux exemples de bases NewSQL sont **NuoDB et VoltDB.**

Caractéristiques

Le NewSQL est une architecture qui rprend donc les avantages du NoSQL et comble son principal désavantage. Il palie à l'éventuel cohérence des données de ce dernier grâce au support de transaction ACID via un langage pour les requêtes le SQL. Ci-dessous quelques-unes de ces caractéristiques :

- Le SQL comme langage commun de requétage
- Transaction ACID
- Un mécanisme qui évite la pause de verrous lors d'opérations concurrentes de lecture avec les opérations d'écritures. La lecture en temps réel en est ainsi facilitée (moins de perte de temps).
- Une architecture qui a de meilleures performances par nœud que les solutions classiques de type SGBDR.
- Architecture distribuée
- la plupart utilise des base de données en mémoire

12

Au vu de ce qui précède, on peut penser que ce type d'architecture sera celle adoptée demain en entreprise. En revanche, c'est une technologie assez récente et plutôt jeune et n'a pas encore pu faire suffisamment ses preuves. Par conséquent, les entreprises sont encore réticentes à l'adoption de cette toute nouvelle architecture

V – 2 - SPRAINed Databases (entorse)

BD alternatives pouvant de traiter de très grands volumes de données, et supporter des applications hautement distribuées ou très complexes

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

- **Scalability** (évolutivité) – hardware economics (économie de matériel)
- **Performance** – BDR limitations
- **Relaxed consistency** – CAP théorème (cohérence relachée)
- **Agility – polyglot persistence** (agilité, persistance polyglotte)
- **Intricacy (intrication)** – big data, total data § N ecessity (nécessité) – open source
- **Necessity (nécessité)** – open source

VI – Classement par usage

DB-Engines has been created and is maintained by **solid IT.(https://db-engines.com/en/about**

solid IT is an Austrian IT consulting company with a special focus on software development, consulting and training for database-centric applications.

solid IT also operates Project Management Zone, a project management portal.

DB-Engines est une initiative pour collecter et présenter des informations sur les systèmes de gestion de bases de données (SGBD). En plus des SGBD relationnels établis, on met l'accent sur les systèmes et les concepts de la zone NoSQL croissante.

Le classement DB-Engines est une liste de SGBD classés par leur popularité actuelle. La liste est mise à jour mensuellement.

Les propriétés les plus importantes de nombreux systèmes sont affichées dans l'aperçu des systèmes de gestion de base de données.

6-1-DB-Engines Ranking of Key-value Stores

Rank			DBMS	Database Model	Score		
Sep 2018	Aug 2018	Sep 2017			Sep 2018	Aug 2018	Sep 2017
1.	1.	1.	Redis	Key-value store	140.94	+2.37	+20.54
2.	2.	2.	Amazon DynamoDB	Multi-model	53.34	+1.69	+15.52
3.	3.	3.	Memcached	Key-value store	31.54	-1.38	+2.60

4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model	19.18	-0.35	+7.95
5.	5.	5.	Hazelcast	Key-value store	8.78	+0.18	+0.05
6.	↑ 7.	↑ 7.	Riak KV	Key-value store	6.39	+1.08	-0.66
7.	↓ 6.	↓ 6.	Ehcache	Key-value store	6.37	-0.17	-0.77
8.	8.	8.	OrientDB	Multi-model	5.48	+0.57	-0.42
9.	9.	9.	Aerospike	Key-value store	5.17	+1.10	+0.84
10.	↑ 11.	10.	ArangoDB	Multi-model	4.05	+0.71	+1.05
11.	↓ 10.	↑ 15.	Ignite	Multi-model	3.71	+0.26	+1.11
12.	12.	↑ 14.	InterSystems Caché	Multi-model	2.77	+0.51	+0.03
13.	↑ 16.	↓ 11.	Oracle NoSQL	Key-value store	2.58	+0.66	-0.37
14.	↑ 15.	↓ 12.	Oracle Berkeley DB	Multi-model	2.58	+0.54	-0.27
15.	↓ 13.	↑ 18.	LevelDB	Key-value store	2.49	+0.28	+0.37
16.	↓ 14.	16.	Infinispan	Key-value store	2.49	+0.33	+0.08
17.	17.	17.	Amazon SimpleDB	Key-value store	2.33	+0.52	+0.08
18.	18.	↓ 13.	Oracle Coherence	Key-value store	2.15	+0.34	-0.69
19.	19.	↑ 20.	RocksDB	Key-value store	1.80	+0.32	+0.44
20.	20.	↓ 19.	GridGain	Multi-model	1.55	+0.46	+0.12
21.	21.	21.	Geode	Key-value store	1.32	+0.29	+0.32
22.	22.		FoundationDB	Multi-model	1.08	+0.21	
23.	23.	23.	XAP	Key-value store	0.82	+0.01	+0.08
24.	24.	24.	GT.M	Key-value store	0.80	+0.07	+0.07
25.	↑ 28.	↑ 26.	WiredTiger	Key-value store	0.78	+0.27	+0.13
26.	↓ 25.	↓ 22.	NCache	Key-value store	0.76	+0.17	+0.01
27.	↓ 26.	↑ 31.	Tarantool	Key-value store	0.67	+0.11	+0.24
28.	↓	↓	ZODB	Key-value	0.53	-0.00	-0.19

	27.	25.		store			
29.	29.	27.	WebSphere eXtreme Scale	Key-value store	0.48	-0.01	-0.10
30.	30.	30.	MapDB	Key-value store	0.42	+0.01	-0.06
31.	31.	28.	Tokyo Cabinet	Key-value store	0.40	+0.01	-0.13
32.	32.	29.	Sqrrl	Multi-model	0.34	-0.00	-0.17
33.	34.	34.	Project Voldemort	Key-value store	0.31	+0.02	-0.02
34.	33.	32.	Tokyo Tyrant	Key-value store	0.30	-0.00	-0.12
35.	36.	39.	BoltDB	Key-value store	0.30	+0.04	+0.11
36.	35.	35.	Graph Engine	Multi-model	0.29	+0.02	+0.02
37.	37.	40.	c-treeACE	Multi-model	0.26	+0.03	+0.09
38.	39.	36.	Kyoto Cabinet	Key-value store	0.20	-0.00	-0.06
39.	40.	37.	Hibari	Key-value store	0.20	+0.01	-0.02
40.	38.		YugaByte DB	Multi-model	0.19	-0.02	
41.	41.	38.	STSdb	Key-value store	0.19	+0.02	-0.02
42.			InterSystems IRIS	Multi-model	0.15		
43.	42.	33.	Scalaris	Key-value store	0.14	-0.00	-0.24
44.	43.	42.	GridDB	Multi-model	0.07	+0.00	-0.01
45.	45.	45.	ScaleOut StateServer	Key-value store	0.06	+0.03	-0.01
46.	44.	50.	Kyoto Tycoon	Key-value store	0.04	+0.00	+0.00
47.	47.	46.	TayzGrid	Key-value store	0.03	+0.00	-0.03
48.	47.	55.	LeanXcale	Multi-model	0.03	+0.00	+0.02
49.	50.	56.	TerarkDB	Key-value store	0.02	+0.01	+0.02
50.	49.	41.	GlobalsDB	Multi-model	0.02	+0.00	-0.07
51.	53.	47.	LedisDB	Key-value store	0.01	+0.01	-0.03
52.	46.	43.	Bangdb	Key-value store	0.01	-0.01	-0.07
53.			CodernityDB	Key-value	0.01	+0.00	-0.02

	52.	49.		store			
54.	51.	53.	InfinityDB	Key-value store	0.00	-0.01	-0.02
55.	54.	56.	Atos Standard Common Repository	Multi-model	0.00	±0.00	±0.00
55.	54.	56.	Badger	Key-value store	0.00	±0.00	±0.00
55.	54.	56.	BergDB	Key-value store	0.00	±0.00	±0.00
55.	54.	56.	Cachelot.io	Key-value store	0.00	±0.00	±0.00
55.	54.	48.	CortexDB	Multi-model	0.00	±0.00	-0.04
55.	54.	52.	Elliptics	Key-value store	0.00	±0.00	-0.03
55.	54.	56.	Helium	Key-value store	0.00	±0.00	±0.00
55.	54.	51.	HyperLevelDB	Key-value store	0.00	±0.00	-0.03
55.	54.	56.	Jaguar	Key-value store	0.00	±0.00	±0.00
55.	54.	56.	Nanolat	Key-value store	0.00	±0.00	±0.00
55.	54.	56.	Resin Cache	Key-value store	0.00	±0.00	±0.00
55.	54.		SwayDB	Key-value store	0.00	±0.00	
55.	54.	54.	TomP2P	Key-value store	0.00	±0.00	-0.02
55.	54.	44.	Upscaledb	Key-value store	0.00	±0.00	-0.08

6-2- DB-Engines Ranking of Document Stores

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the complete ranking showing only document stores.

Read more about the method of calculating the scores.

47 systems in ranking, September 2018

Rank			DBMS	Database Model	Score		
Sep	Aug	Sep			Sep	Aug	Sep

2018	2018	2017			2018	2018	2017
1.	1.	1.	MongoDB	Document store	358.79	+7.81	+26.06
2.	2.	2.	Amazon DynamoDB	Multi-model	53.34	+1.69	+15.52
3.	3.	3.	Couchbase	Document store	34.55	+1.59	+1.44
4.	4.	↑ 6.	Microsoft Azure Cosmos DB	Multi-model	19.18	-0.35	+7.95
5.	5.	↓ 4.	CouchDB	Document store	18.64	+0.21	-2.35
6.	6.	↓ 5.	MarkLogic	Multi-model	11.53	+0.32	-0.57
7.	7.	↑ 9.	Firebase Realtime Database	Document store	8.10	+0.59	+3.47
8.	8.	↓ 7.	OrientDB	Multi-model	5.48	+0.57	-0.42
9.	9.	↓ 8.	RethinkDB	Document store	4.30	+0.29	-0.33
10.	10.	↑ 14.	ArangoDB	Multi-model	4.05	+0.71	+1.05
11.	11.	↑ 13.	Google Cloud Datastore	Document store	3.80	+0.48	+0.74
12.	12.	↓ 10.	Cloudant	Document store	3.61	+0.33	-0.64
13.	13.	↓ 11.	RavenDB	Document store	3.35	+0.75	-0.14
14.	14.	↓ 12.	Apache Drill	Multi-model	2.95	+0.42	-0.26
15.	15.	15.	PouchDB	Document store	2.67	+0.32	-0.28
16.	16.		Google Cloud Firestore	Document store	2.43	+0.21	
17.	17.	↓ 16.	CloudKit	Document store	2.30	+0.72	+0.16
18.	18.	↓ 17.	Datameer	Document store	1.32	+0.20	-0.63
19.	19.	↓ 18.	Mnesia	Document store	1.11	+0.22	-0.06
20.	20.		FoundationDB	Multi-model	1.08	+0.21	
21.	↑ 23.		BigchainDB	Document store	0.88	+0.47	
22.	22.	↑ 24.	LiteDB	Document store	0.68	+0.26	+0.40
23.	↓ 21.	↓ 20.	MapR-DB	Multi-model	0.62	+0.01	+0.14
24.	24.	↓ 23.	AlaSQL	Multi-model	0.54	+0.14	+0.23
25.	25.	↓	Sqrrl	Multi-	0.34	-0.00	-0.17

		19.		model			
26.	27.	21.	Percona Server for MongoDB	Document store	0.28	+0.02	-0.17
27.	28.	25.	Sequoiadb	Document store	0.26	+0.04	-0.01
28.	26.	26.	LokiJS	Document store	0.26	-0.01	+0.05
29.	29.		YugaByte DB	Multi-model	0.19	-0.02	
30.	30.	27.	FaunaDB	Multi-model	0.17	+0.02	+0.00
31.			InterSystems IRIS	Multi-model	0.15		
32.	31.	28.	RaptorDB	Document store	0.13	+0.02	-0.01
33.	32.	29.	EJDB	Document store	0.10	+0.00	-0.02
34.	34.	32.	DensoDB	Document store	0.08	-0.00	-0.01
35.	35.	35.	SenseiDB	Document store	0.07	-0.01	+0.01
36.	33.	33.	NosDB	Document store	0.06	-0.02	-0.00
37.	36.	34.	FleetDB	Document store	0.06	+0.02	-0.01
38.	38.	31.	GlobalsDB	Multi-model	0.02	+0.00	-0.07
39.	37.	30.	iBoxDB	Document store	0.01	-0.01	-0.08
40.	41.	38.	SisoDb	Document store	0.01	+0.01	-0.04
41.	39.	39.	WhiteDB	Document store	0.00	-0.00	-0.04
42.	41.	42.	Atos Standard Common Repository	Multi-model	0.00	±0.00	±0.00
42.	41.	40.	CortexDB	Multi-model	0.00	±0.00	-0.04
42.	41.	37.	Djondb	Document store	0.00	±0.00	-0.05
42.	41.	41.	JasDB	Document store	0.00	±0.00	-0.00
42.	41.	42.	OrigoDB	Multi-model	0.00	±0.00	±0.00
42.	40.	36.	ToroDB	Document store			

- 6 – 3- DB-Engines Ranking of Graph DBMS

31 systems in ranking, September

Rank Sep 2018	Rank Aug 2018	Rank Sep 2017	DBMS	Database Model	Score Sep 2018	Score Aug 2018	Score Sep 2017
1.	1.	1.	Neo4j	Graph DBMS	40.10	-0.83	+1.67
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	19.18	-0.35	+7.95
3.	3.		Datastax Enterprise	Multi-model	7.76	+0.46	
4.	4.	↓ 3.	OrientDB	Multi-model	5.48	+0.57	-0.42
5.	5.	5.	ArangoDB	Multi-model	4.05	+0.71	+1.05
6.	6.	6.	Virtuoso	Multi-model	2.06	+0.01	+0.17
7.	↑ 8.		Amazon Neptune	Multi-model	1.12	+0.31	
8.	↓ 7.	↓ 7.	Giraph	Graph DBMS	1.02	+0.03	-0.05
9.	↑ 11.	↑ 16.	JanusGraph	Graph DBMS	0.90	+0.36	+0.68
10.	10.	↓ 9.	GraphDB	Multi-model	0.63	+0.06	+0.02
11.	↓ 9.	↓ 8.	AllegroGraph	Multi-model	0.60	+0.02	-0.04
12.	12.	↓ 10.	Stardog	Multi-model	0.54	+0.01	-0.04
13.	↑ 17.	13.	Dgraph	Graph DBMS	0.41	+0.17	+0.14
14.	↑ 15.	↑ 15.	Blazegraph	Multi-model	0.36	+0.08	+0.12
15.	↓ 13.	↓ 11.	Sqrrl	Multi-model	0.34	-0.00	-0.17
16.	16.	↓ 14.	Graph Engine	Multi-model	0.29	+0.02	+0.02
17.	↓ 14.	↓ 12.	InfiniteGraph	Graph DBMS	0.28	-0.02	-0.01
18.	18.		TigerGraph	Graph DBMS	0.22	+0.02	
19.	19.	19.	FaunaDB	Multi-model	0.17	+0.02	+0.00
20.	↑ 21.	↑ 22.	VelocityDB	Multi-model	0.14	+0.01	+0.03
21.	↓ 20.	↓ 17.	Sparksee	Graph DBMS	0.14	-0.00	-0.07
22.	22.	↓ 20.	HyperGraphDB	Graph DBMS	0.13	+0.01	-0.02
23.	↑	↓	FlockDB	Graph DBMS	0.13	+0.01	-0.04

	24.	18.					
24.	23.	21.	InfoGrid	Graph DBMS	0.12	-0.00	-0.02
25.	25.	24.	TinkerGraph	Graph DBMS	0.05	-0.01	-0.00
26.	26.	25.	GraphBase	Graph DBMS	0.04	-0.00	-0.00
27.	27.	27.	AgensGraph	Multi-model	0.04	+0.01	+0.04
28.	28.	26.	GRAKN.AI	Multi-model	0.03	+0.01	+0.02
29.	29.	23.	GlobalsDB	Multi-model	0.02	+0.00	-0.07
30.	30.		AnzoGraph	Graph DBMS	0.00	-0.00	
30.	31.	27.	HGraphDB	Graph DBMS	0.00	±0.00	±0.00

31 systems in ranking, September