Adoption d'une solution NoSQL dans l'entreprise

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Matteo DI MAGLIE

Conseiller au travail de Bachelor :

Christian STETTLER, Chargé de cours HES

Carouge, 12 septembre 2012

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

1 Déclaration

Ce mémoire de Bachelor est réalisé dans le cadre de l'examen final de la Haute École de Gestion, en vue de l'obtention du titre de Bachelor en Informatique de Gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celle citées dans la bibliographie. »

Fait à Carouge, le 12.09.2012

Matteo Di Maglie

2 Remerciements

Tout d'abord, je souhaite remercier toutes les personnes qui m'ont aidé d'une manière ou d'une autre à réaliser ce travail de Bachelor.

Je tiens à remercier tout particulièrement Monsieur Christian Stettler pour son suivi ainsi que ses conseils durant les deux mois de réalisation de ce travail.

Je remercie également tous les intervenants de la conférence « NoSQL Roadshow » qui a eu lieu à Bâle pour leurs présentations ainsi que les nombreuses personnes du groupe « NoSQL » présentes sur LinkedIn pour leurs réponses à mes questions.

Un grand merci à Madame Marlène Poyet et Madame Pamela Di Maglie qui ont eu la gentillesse de lire et corriger l'orthographe de ce document.

Enfin, je souhaite remercier toute ma famille qui m'a soutenu tout au long de ce travail.

3 Résumé

Ce mémoire traite de l'adoption d'une solution NoSQL en entreprise, il est essentiellement destiné au responsable IT (CIO) dans une société qui envisagerait l'adoption d'une solution NoSQL, ce document étant destiné à lui servir de guide.

La première partie de ce document est consacrée à un rappel de certaines définitions relatives aux bases de données en général.

En seconde partie suit une introduction au NoSQL, ce qui permet d'avoir une rétrospective historique du mouvement, d'en saisir les enjeux ainsi qu'un descriptif du fonctionnement. Vous trouverez également un aperçu du marché grâce à une description de quelques produits NoSQL.

Ensuite, vous serez propulsés au cœur du sujet de mon travail de recherche. En effet, la lecture de ce chapitre renseignera le CIO sur les avantages d'une solution NoSQL dans l'entreprise, les cas d'utilisation de celle-ci et ceux ou une telle solution ne s'y adapte pas, les difficultés qu'il devra surmonter, ainsi que les caractéristiques techniques et non techniques auxquelles il doit être attentif lors du choix du produit.

La dernière partie de ce travail est une liste de questions permettant l'identification des produits NoSQL qui répondront le mieux aux besoins de l'entreprise.

Mots clés : NoSQL, SGDB non relationnel, Enterprise, Adoption, Big data, « Scalabilité » horizontale, Tolérance à la panne, Schéma dynamique, MangoDB, Cassandra, Riak

4 Table des matières

5 Liste des tableaux

6 Liste des figures

7 Introduction

Cela fait plus de vint-cinq ans que les systèmes de bases de données relationnelles règnent en maître quand les données d'une application sont concernées. Cependant, ces dernières années, le Web a connu une révolution avec l'avènement des sites web à fort trafic tels que Facebook, Amazon et LinkedIn. Ces grands acteurs du web ont été rapidement limités par les dits systèmes pour deux raisons majeures :

- Les gros volumes de données
- Les montées en charge.

N'ayant pas trouvé de solution sur le marché répondant à ces problèmes, ils décidèrent de développer chacun à l'interne leurs propres SGBD. Ces produits développés de manière distincte sont connus sous le nom de base de données NoSQL ou de base de données non relationnelle.

Les besoins en performance, lors de traitement de gros volumes de données ainsi que d'augmentation de trafic, ne touchent pas seulement les fameux sites mentionnés ci-dessus, mais aussi de nombreuses entreprises de tous types d'industries confondues.

Du fait que le NoSQL est assez récent et pour le moment peu répandu dans le monde des entreprises, peu de personnes le connaissent vraiment. Donc, lorsqu'un responsable IT veut faire le grand pas vers l'adoption d'une solution NoSQL, il se retrouve avec une multitude de difficultés à surmonter, la plus importante étant de trouver la solution qui réponde le mieux aux besoins de l'entreprise. En effet, il se retrouve face à une technologie qui regroupe près de 122 solutions NoSQL dont chacune d'entre elles comporte des caractéristiques spécifiques.

8 Notions préliminaires

8.1 Base de données

Une base de données (BDD) est un dispositif dans lequel il est possible de stocker des données de manière structurée et avec le moins de redondances possibles. Ces données sont souvent susceptibles d'être utilisées par des programmes qui sont employés simultanément par plusieurs utilisateurs. C'est pourquoi la notion de base de données est généralement couplée à celle de réseau, afin de pouvoir mettre en commun ces informations.

Afin que les utilisateurs puissent procéder à une consultation, une saisie ou faire une mise à jour des données, la base de données les leur met à disposition, tout en s'assurant des droits accordés respectifs.

Une base de données peut être soit locale, soit répartie. Par locale, on entend qu'elle est utilisable sur une machine par un seul utilisateur, tandis que pour une base de données répartie, les informations sont stockées sur des machines distantes et accessibles par le réseau.

L'avantage majeur des bases de données réparties est la possibilité de pouvoir être accédées simultanément par plusieurs utilisateurs.

8.2 Système de gestion de base de données

Un système de gestion de base de données, abrégé SGBD, est un logiciel qui permet la gestion des bases de données. Il englobe de nombreux services tels que l'accès aux données de façon simple, l'insertion de données, l'autorisation des accès aux informations simultanément à de multiples utilisateurs ainsi que la manipulation (modification et suppression) des données présentes dans la base.

Un SGBD se décompose en trois sous systèmes :

- Le système de gestion de fichier permettant le stockage d'informations sur support physique.
- Le SGBD interne gérant l'ordonnance des informations.
- Le SGBD externe représentant l'interface avec l'utilisateur.

Les niveaux d'abstraction d'un SGBD sont définis par l'architecture ANSI/SPARC décrite ci-dessous :

- Le niveau interne (ou physique) définit les méthodes de stockage de données ainsi que les manières d'y accéder.
- Le niveau conceptuel définit l'organisation des données au sein des informations dans la base de données.
- Le niveau externe définit les vues des utilisateurs.

Cette architecture à trois niveaux permet d'avoir une indépendance entre les données et les traitements.

8.2.1 Modèle hiérarchique

Dans une base de données hiérarchique, les enregistrements sont structurés hiérarchiquement, selon une arborescence descendante où chaque enregistrement ne possède qu'un seul enregistrement parent. Ces structures de données hiérarchiques furent utilisées dans les premiers systèmes de gestion de bases de données de type mainframe. Le modèle hiérarchique a vite montré ses limites, surtout lorsque la structure devient complexe et qu'elle doit répondre à des besoins plus pointus.

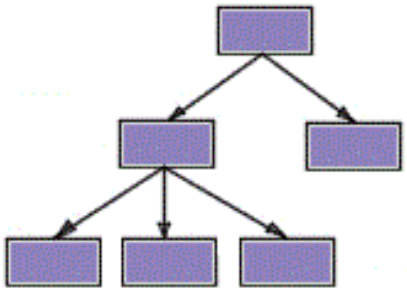


Figure 1 – Modèle hiérarchique

8.2.2 Modèle réseau

Le modèle de données de type réseau a vu le jour dans le but de résoudre certaines difficultés que posait le modèle hiérarchique. Il permet de faire des relations entre tous types de données et les liens entres objets peuvent exister sans restriction. Afin de retrouver une donnée dans ce type de modèle, il est nécessaire de connaître le chemin d'accès vers l'information, cela tendant à rendre les programmes dépendants de la structure de données.

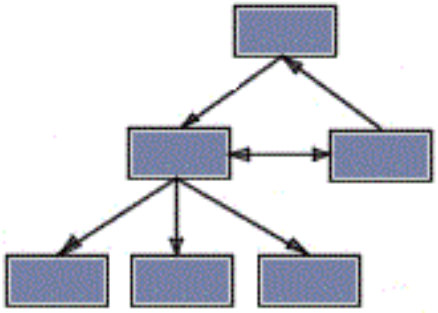


Figure 2 – Modèle réseau

8.2.3 Modèle relationnel

Le SGBD relationnel est le plus répandu à l'heure actuelle. Dans ce modèle, les informations sont décomposées et organisées dans des matrices à deux dimensions (ligne-colonne) appelées relations ou tables. La manipulation de ces données se fait selon la théorie mathématique des opérations d'algèbre relationnelle telles que l'intersection, la jointure ou le produit cartésien.

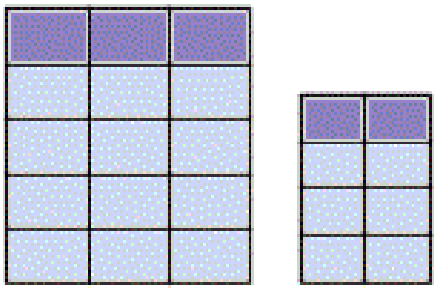


Figure 3 – Modèle relationnel

8.3 Les propriétés ACID

Lorsque des transactions sont effectuées, les SGBD de type hiérarchique, réseau et relationnel fonctionnent selon les contraintes dites ACID. ACID est un acronyme qui veut dire « Atomicity », « Consistancy », « Isolation » et « Durability ». Une transaction est réalisée avec succès si elle respecte ces quatre contraintes. Voici une description détaillée de celles-ci[1] :

- **« Atomicity »** (Atomicité) :
  Lorsqu'une transaction est effectuée, toutes les opérations qu'elle comporte doivent être menées à bien : en effet, en cas d'échec d'une seule des opérations, toutes les opérations précédentes doivent être complètement annulées, peu importe le nombre d'opérations déjà réussies. En résumé, une transaction doit s'effectuer complètement ou pas du tout. Voici un exemple concret : une transaction qui comporte 3000 lignes qui doivent être modifiées ; si la modification d'une seule des lignes échoue, alors la transaction entière est annulée. L'annulation de la transaction est toute à fait normale, car chaque ligne ayant été modifiée peut dépendre du contexte de modification d'une autre, et toute rupture de ce contexte pourrait engendrer une incohérence des données de la base.

- **« Consistancy »** (Cohérence) :
  Avant et après l'exécution d'une transaction, les données d'une base doivent toujours être dans un état cohérent. Si le contenu final d'une base de données contient des incohérences, cela entraînera l'échec et l'annulation de toutes les opérations de la dernière transaction. Le système revient au dernier état cohérent. La cohérence est établie par les règles fonctionnelles.

- **« Isolation »** (Isolation) :
  La caractéristique d'isolation permet à une transaction de s'exécuter en un mode isolé. En mode isolé, seule la transaction peut voir les données qu'elle est en train de modifier, c'est le système qui garantit aux autres transactions exécutées en parallèle une visibilité sur les données antérieures. Ce fonctionnement est obtenu grâce aux verrous système posés par le SGBD. Prenons l'exemple de deux transactions A et B : lorsque celles-ci s'exécutent en même temps, les modifications effectuées par A ne sont ni visibles, ni modifiables par B tant que la transaction A n'est pas terminée et validée par un « commit ».

- **« Durability »** (Durabilité) :
  Toutes les transactions sont lancées de manière définitive. Une base de données ne doit pas afficher le succès d'une transaction pour ensuite remettre les données modifiées dans leur état initial. Pour ce faire, toute transaction est sauvegardée dans un fichier journal afin que, dans le cas où un problème survient empêchant sa validation complète, elle puisse être correctement terminée lors de la disponibilité du système.

[1] Source : Transaction informatique http://fr.wikipedia.org/wiki/Transaction_informatique

8.4 « Scalabilité »

La « scalabilité » est le terme utilisé pour définir l'aptitude d'un système à maintenir un même niveau de performance face à l'augmentation de charge ou de volumétrie de données, par augmentation des ressources matérielles.

Il y a deux façons de rendre un système extensible : la « scalabilité » horizontale (externe) ainsi que la « scalabilité » verticale (interne).

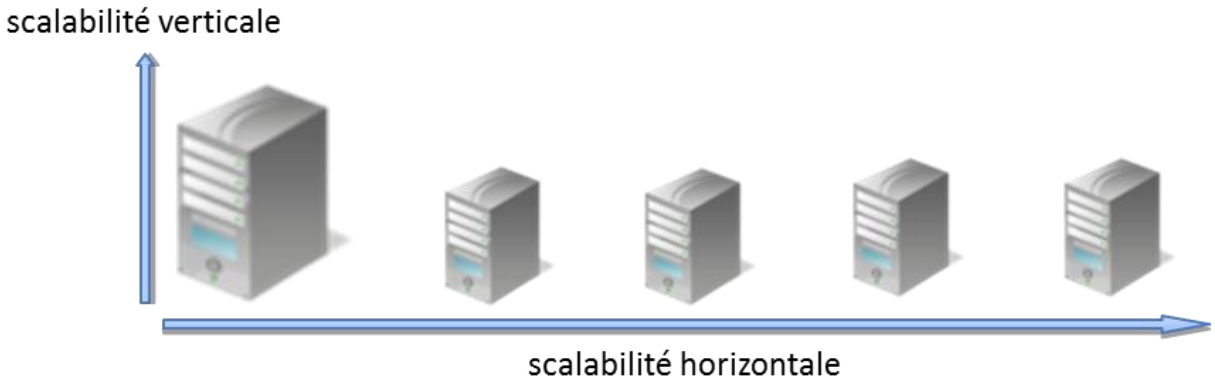


Figure 4 - "Scalabilité" horizontale et verticale

8.4.1 « Scalabilité » horizontale

Le principe de la « scalabilité » horizontale consiste à simplement rajouter des serveurs en parallèle, c'est la raison pour laquelle on parle de croissance externe. On part d'un serveur basique et on rajoute des nouveaux serveurs identiques afin de répondre à l'augmentation de la charge.

Les avantages :

- Achat de serveurs quand le besoin s'en fait sentir.
- Une panne de serveur ne pénalise pas le système.
- Flexibilité du système.
- Possibilité de mises à jour sans interruption de service.

Les inconvénients :

- Achat de licences pour chaque serveur.
- Difficulté de trouver des serveurs identiques sur le long terme.

8.4.2 « Scalabilité » verticale

Le principe de la « scalabilité » verticale consiste à modifier les ressources d'un seul serveur, comme par exemple le remplacement du CPU par un modèle plus puissant ou par l'augmentation de la mémoire RAM. C'est ce que l'on appelle la croissance interne. On a une machine et on l'a fait évoluer dans le temps.

Les avantages:

- Achat d'une seule licence.
- Administration simplifiée du serveur.

Les inconvénients :

- Aucune tolérance à la panne.
- Achat d'une machine coûteuse.
- Le système est limité dans la flexibilité.
- Impossibilité de faire des mises à jour sans interrompre le système.

9 Introduction au NoSQL

Cette première partie est une introduction au NoSQL. J'ai jugé que cette étape était primordiale car le NoSQL est relativement récent et peu de personnes le connaissent vraiment bien. En effet, selon Shashank Tiwari, « le monde du développement informatique, est probablement divisé en trois groupes de personnes quand intervient le NoSQL »[2] :

- Les personnes qui l'adulent : celles-ci analysent comment les technologies NoSQL pourraient rendre une application plus performante grâce à leurs différents outils. Elles l'utilisent quand il est nécessaire.
- Les personnes qui le nient : elles ne voient que les aspects négatifs du NoSQL et ne vont pas encourager son utilisation.
- Les personnes qui ne le connaissent pas : celles-ci en ont vaguement entendu parler mais attendent que la technologie NoSQL devienne plus mature, ou pensent que c'est ce qu'on appelle dans le monde de la technologie un « fad » et attendent la fin du « hype cycle », ou n'ont jamais eu de temps à y consacrer.

Avant la réalisation de ce travail de Bachelor, je me situais dans le troisième groupe.

9.1 Historique du mouvement NoSQL

En 1998, le monde entend pour la première fois le terme NoSQL. Terme inventé et employé par Carlo Strozzi pour nommer son SGBD relationnel Open Source léger qui n'utilisait pas le langage SQL. Ironiquement, la trouvaille de M. Strozzi n'a rien à voir avec la mouvance NoSQL que l'on connaît aujourd'hui, vu que son SGBD est de type relationnel. En effet, c'est en 2009, lors d'un rassemblement de la communauté des développeurs des SGBD non-relationnels, que le terme NoSQL a été mis au goût du jour pour englober tous les SGBD de type non-relationnel.

NoSQL est une combinaison de deux mots : No et SQL. Qui pourrait être mal interprétée car l'on pourrait penser que cela signifie la fin du langage SQL et qu'on ne devrait donc plus l'utiliser. En fait, le « No » est un acronyme qui signifie « Not only », c'est-à-dire en français, « non seulement » ; c'est donc une manière de dire qu'il y a autre chose que les bases de données relationnelles.

Les créateurs du mot NoSQL voulaient probablement dire « No SGBD relationnel » ou « No relational », mais ils ont été séduits par le terme NoSQL qui fait plus marketing. Certaines personnes ont proposé de remplacer le terme NoSQL par NoRel, toutefois sans succès.

Aujourd'hui, le terme NoSQL englobe tous les SGBD qui ne suivent pas la tendance de type relationnel. Cela signifie que le NoSQL n'est pas un seul produit ou même une technologie unique. En effet, il représente de nombreux produits ainsi que plusieurs concepts de stockage et de manipulation de données.

[2] TIWARI, Shashank. Professional: NoSQL. Indianapolis: Wrox, 2011. P. 3

9.2 Pourquoi le NoSQL?

A l'heure de la révolution du Web 2.0 et vu l'usage actuel d'Internet, le nombre de données disponibles sur la toile augmente d'année en année de manière exponentielle. Voici quelques exemples d'acteurs concernés par ces gros volumes de données.

Le réseau social Facebook c'est[3] :

- Plus de 845 millions de comptes à travers le monde.
- Plus de 2,7 milliards de « J'aime » et de messages postés par jour.
- Plus de 250 millions de photos « uploadées » chaque jour.
- Plus de 100 milliards de connexions par an.

La célèbre plateforme de réseautage Twitter c'est[4] :

- Plus de 465 millions de comptes à travers le monde.
- Plus de 175 millions de « tweets » par jour.
- Plus de 33 milliards de « tweets » par an.

Le terme usuel pour définir ce type de données est « Big Data » (en français gros volume de données). Pour manipuler et stocker ces données, les professionnels ont longtemps misé sur les outils traditionnels que sont les SGBD relationnels. Malgré l'optimisation poussée au maximum de ceux-ci, les limites sont atteintes, surtout en termes de performances quant à la rapidité de lecture et d'écriture des données. En effet, garantir la cohérence des données coûte cher en temps de réponse et est souvent incompatible avec les performances.

Ces problèmes de performance liés à l'augmentation de charge et de volumétrie de données ne sont pas le fruit du hasard. En effet, les SGBD relationnels doivent respecter les principes ACID (cf. 8.3 Les propriétés ACID). Donc, il devient très difficile de faire du « Scale out » dans leur cas. Pour cela, il faudrait que les données soient simultanément présentes sur tous les serveurs.

Prenons un exemple concret : une base de données qui effectuerait par exemple 1000 transactions par minute, dont chacune d'elle correspondrait à environ 2 Ko de données. Répartissons-la sur 10 serveurs. Cela fait environ 300 Ko/s de communications à envoyer entre les différents serveurs. De plus, il faut prendre en compte la gestion de l'intégrité, de la cohérence et de la disponibilité des données. Face à un tel trafic de données le système est alors incapable de gérer la situation et s'écroule très rapidement.

Nous pouvons donc constater que les SGBD relationnels ne peuvent que difficilement reproduire les mécanismes de répartition de charge dont sont capables les serveurs Web. Afin de rester performant malgré l'augmentation de charge et de volumétrie de données, la seule solution est de faire du « Scale in », c'est-à-dire opter pour l'obtention d'un serveur de base de données fortement évolutif. Mais à quel prix ? Et surtout est-ce que cette machine super puissante qui réglera tous les problèmes

[3] Source : Les statistiques de Facebook en 2012 http://www.passion-net.fr/infographie-les-statistiques-de-facebook-en-2012/

[4] Source : Les statistiques de Twitter en 2012 http://www.passion-net.fr/infographie-les-statistiques-de-twitter-en-2012/

existe-t-elle vraiment sur le marché ? Et si la solution venait du changement de SGBD ?

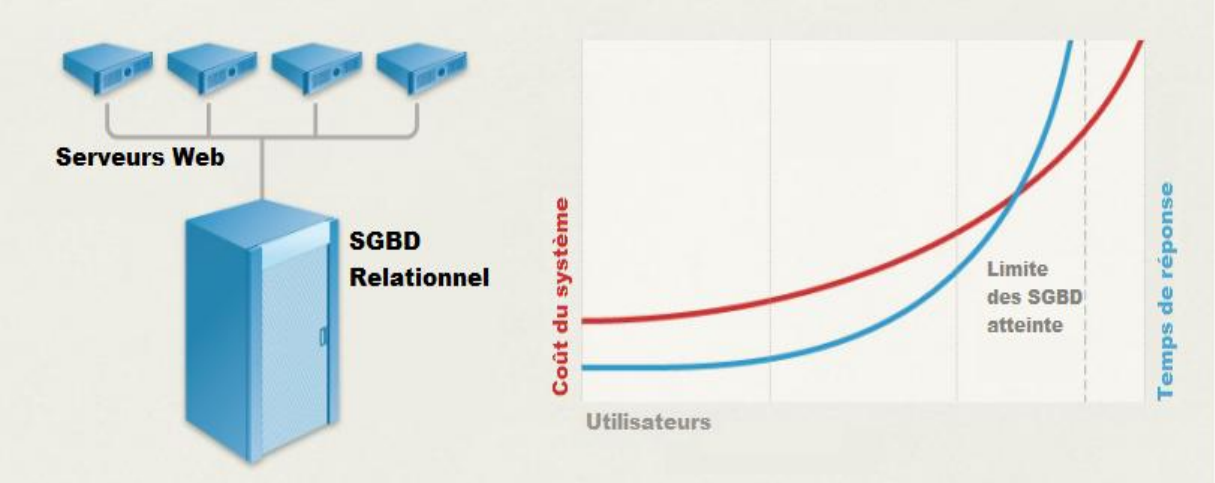


Figure 5 - "Scalabilité" verticale

Les bases de données relationnelles ont fondamentalement été conçues pour gérer des charges de travail stables ainsi que l'extraction de données complexes, ce qui n'est plus si fréquent dans les applications modernes. Pour ces dernières, la capacité à gérer de grands ensembles de données tout en maintenant la rapidité et l'extensibilité est devenue primordiale. C'est de ce constat qu'est né le mouvement NoSQL. Les SGBD NoSQL sont basés sur des systèmes innovants permettant la « scalabilité » horizontale et dont la plupart d'entre eux sont Open Source, ils sont conçus pour répondre à des besoins spécifiques et assurer une extensibilité extrême sur de très grands ensembles de données.

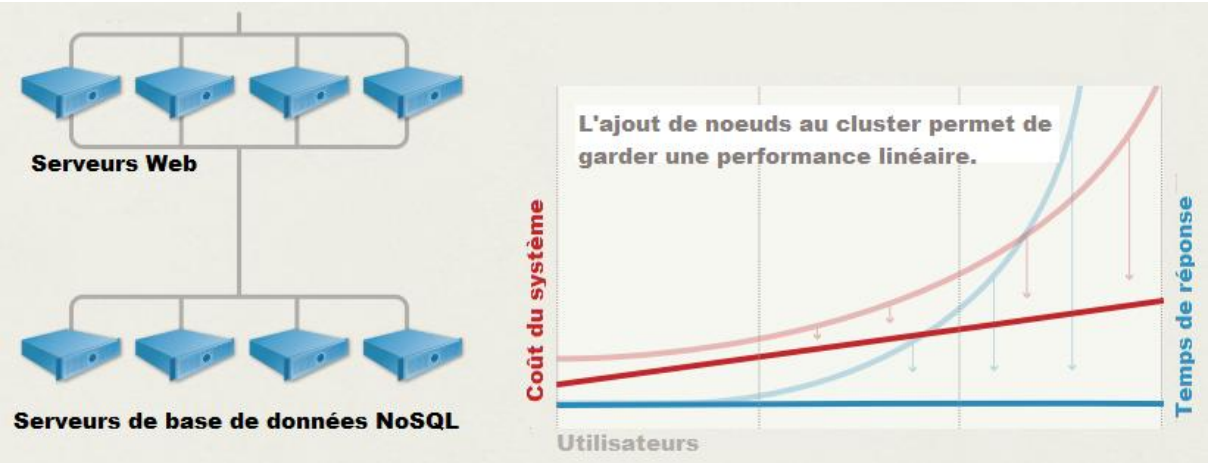


Figure 6 - "Scalabilité" horizontale

9.3 Comment le NoSQL fonctionne ?

Ce sont les grands acteurs du Web tels que Google, Amazon, Linkedin et Facebook qui ont solutionné leurs problèmes en développant, en parallèle de leurs activités primaire, leurs propres SGBD NoSQL. A l'heure actuelle, la quasi-totalité des sites à fort trafic utilisent des SGBD NoSQL.

9.3.1 Théorème de CAP

Le théorème de CAP est l'acronyme de « Coherence », « Availability » et « Partition tolerance », aussi connu sous le nom de théorème de Brewer. Ce théorème, formulé par Eric Brewer en 2000 et démontré par Seth Gilbert et Nancy Lych en 2002[5], énonce une conjecture qui définit qu'il est impossible, sur un système informatique de calcul distribué, de garantir en même temps les trois contraintes suivantes :

- **« Coherence »** (Cohérence) :
  Tous les clients du système voient les mêmes données au même instant.

- **« Availibility »** (Haute disponibilité) :
  Un système est dit disponible si toute requête reçue par un nœud retourne un résultat. Bien évidemment le nœud en question ne doit en aucun cas être victime de défaillance.

- **« Partition tolerance »** (Tolérance à la partition) :
  Un système est dit tolérant à la partition s'il continue à répondre aux requêtes de manière correcte même en cas de panne autre qu'une panne totale du système.

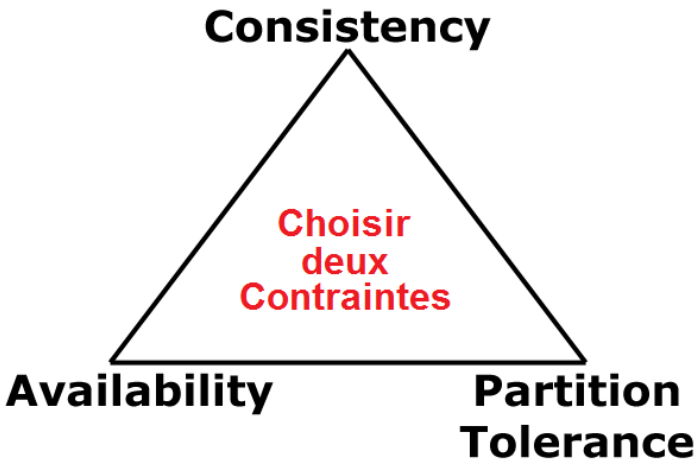


Figure 7 - Théorème de CAP

Seuls deux des trois contraintes peuvent être respectées en même temps. Un des premiers buts des systèmes NoSQL est de renforcer la « scalabilité » horizontale, il faut pour cela que le principe de tolérance au partitionnement soit respecté, ce qui exige l'abandon soit de la cohérence, soit de la haute disponibilité.

[5] Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services http://lpd.epfl.ch/sgilbert/pubs/BrewersConjecture-SigAct.pdf
Brewer's CAP theorem
http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

9.3.2 Haute disponibilité et tolérance à la partition

Dans bien des cas, c'est le principe de haute disponibilité qui est privilégié : c'est notamment le cas pour Amazon et, cela vient du fait que ce principe est plus important (pour Amazon) que la cohérence des données.

En effet, cela voudrait dire que les utilisateurs n'ont pas forcément tous la même vue à un moment donné. Dans certains cas cela peut poser des problèmes, mais bien souvent ce n'est pas une nécessité. Prenons l'exemple de deux utilisateurs qui sont amis sur Facebook, Marcel et Pierre. Si Marcel partage une photo sur son « mur » et que Pierre au même instant ne la voit pas dans sa « fil d'actualité » elle apparaîtra dans les prochaines secondes. Dans ce cas précis, est-ce un problème pour Pierre d'attendre environ une minute pour voir la photo que vient de publier son ami Marcel ?

Si toutefois, la cohérence est une chose importante pour certaines données, il est d'usage de l'isoler dans des bases relationnelles et de placer les autres données dans des bases dites NoSQL.

Dans certains cas la cohérence des données nous parait logique, mais cela est souvent dû à notre passivité d'esprit. Prenons le cas d'un « shop » en ligne, une valeur indique le stock d'un article. Si deux serveurs ayant des valeurs non cohérentes, il peut arriver qu'un serveur considère qu'il reste un article disponible alors que l'autre serveur avec une mise à jour plus récente indique qu'il n'en reste plus. Si un client malchanceux commande un article qui n'est plus en stock, le magasin en ligne a deux options : remboursement immédiat et donc mettre un terme à la commande du client ou réapprovisionnement du stock afin de conclure avec succès la commande.

Il est clair que cet exemple démontre certains désagréments, mais entre :

- faire ses achats sur un site très lent (quelques dizaines de secondes de latence par page) mais où le stock est en temps réel entre les milliers de serveurs pour tous les produits du site.
- une éventualité de stock non cohérente mais avec une autre disponibilité

Il n'y a pas photo : les sites ont vite fait leurs choix. Le géant de la vente en ligne Amazon a constaté lors de diverses études concernant le temps d'attente, qu'un dixième de seconde de latence leur fait diminuer les ventes de 1%[6].

Afin de garantir un minimum de cohérence dans les données des systèmes de type hautement disponible, un mécanisme de fusion des données est lancé. Les conflits sont résolus par un protocole de résolution très similaire à celui d'un système de gestion des versions (comme Subversion ou Git) sauf que dans ce cas la gestion du conflit est entièrement automatisée.

9.3.3 Cohérence et tolérance à la partition

Quand on regarde la définition d'un système cohérent et tolérant à la partition, on pourrait croire que, du fait que la disponibilité soit compromise, le système ne soit jamais disponible ; en d'autres termes un système qui ne sert à rien. Bien entendu, cela n'est pas le cas.

[6] Source: http://mashable.com/2011/04/06/site-speed/

La plupart du temps, les systèmes NoSQL qui garantissent une forte cohérence des données sont architecturés en maître/esclave. Cela veut dire que toutes les écritures doivent être faites sur le serveur maître, et en cas de panne de ce dernier, les opérations d'écriture, de modification et de suppression ne deviennent plus disponibles. Seule la lecture des données sur les serveurs esclaves est possible.

Comme ces systèmes utilisent la plupart du temps une réplication asynchrone, seul le serveur maitre est cohérent : cela veut dire que pour une courte période de temps des anciennes données sont encore accessibles en lecture sur les serveurs esclaves.

9.3.4 Les propriétés de BASE

Dans la première partie consacrée aux notions principales, nous avons vu les propriétés ACID (cf. 8.3 Les Propriétés ACID) auxquelles doivent répondre les SGBD de type relationnel. Les SGBD NoSQL qui, selon le théorème CAP, privilégient la disponibilité ainsi que la tolérance à la partition plutôt que la cohérence, répondent aux propriétés de BASE.

Le principe de BASE est le fruit d'une réflexion menée par Eric Brewer (cf. 9.3.1 Théorème de CAP). Les caractéristiques de BASE sont fondées sur les limites que montrent les SGBD relationnelles. Voici sa description[7] :

- **Basically Available** (Disponibilité basique) :
  Même en cas de désynchronisation ou de panne d'un des nœuds du cluster, le système reste disponible selon le théorème CAP.

- **Soft-state** (Cohérence légère) :
  Cela indique que l'état du système risque de changer au cours du temps, sans pour autant que des données soient entrées dans le système. Cela vient du fait que le modèle est cohérent à terme.

- **Eventual consistancy** (Cohérence à terme) :
  Cela indique que le système devient cohérent dans le temps, pour autant que pendant ce laps de temps, le système ne reçoive pas d'autres données.

[7] Source : Eventual consistency: http://en.wikipedia.org/wiki/Eventual_consistency

9.4 Les types de bases NoSQL

Dans la mouvance NoSQL, il existe une diversité d'approches classées en quatre catégories. Ces différents systèmes NoSQL utilisent des technologies forts distinctes. Les différents modèles de structure sont décrits comme suit :

9.4.1 Paradigme clé/valeur

Souvent assimilé à une « hashmap » distribuée, le système de base de données de type clé/valeur est probablement le plus connu et le plus basique que comporte la mouvance NoSQL. Son principe est extrêmement simple : chaque objet (valeur) est identifié par une clé unique. Celle-ci représente la seule manière de solliciter l'objet. Dans la plupart des cas, la structure de l'objet est libre et donc à la charge du développeur de l'application (XML, JSON, BSON, etc…), en général la base de données ne gérant que des chaînes d'octets. L'absence de structure ou de typage a un impact important sur la manière de faire les requêtes à la base de données. Cela vient du fait qu'auparavant toute l'intelligence était portée par les requêtes SQL, alors que dans ce système c'est au niveau de l'applicatif que l'interrogation de la base de données a lieu. Toutefois, la communication avec la base de données se résume aux opérations basiques que sont PUT, GET, UPDATE et DELETE. La plupart des bases de données de type clé/valeur disposent d'une interface HTTP REST qui permet de procéder très facilement à des requêtes depuis n'importe quel langage de développement. Ces systèmes affichent des performances exceptionnellement élevées en lecture et en écriture, ainsi qu'une « scalabilité » horizontale étendue, cela vient du fait que ces types de bases sont réduits à un simple accès disque.

Du fait que les opérations possibles soient basiques (simple CRUD), le besoin en « scalabilité » verticale est fortement réduit. Ces systèmes sont souvent utilisés comme dépôts de données si toutefois les besoins en termes de requêtes restent de niveau simple et que l'intégrité relationnelle des données est non significative.

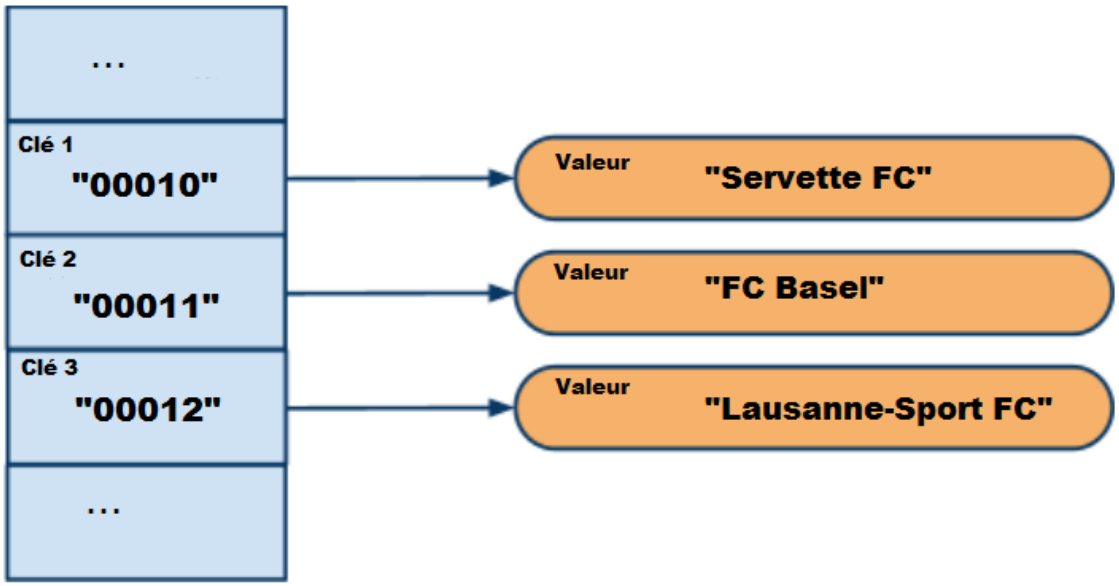


Figure 8 - Clé/valeur

9.4.2 Bases documentaires

Les systèmes de type documentaire sont composés de collections de documents. La représentation en document est une sorte d'extension du concept clé/valeur. La valeur est représentée sous forme de document contenant des données. Des champs et des valeurs associées composent le document. Ces valeurs associées peuvent soit être de type simple (integer, string, date, …), soit de type composé, c'est-à-dire de plusieurs couples clé/valeur. Bien que les documents soient organisés de manière hiérarchique à l'image de ce que permettent les fichiers de type XML ou JSON, ces bases sont dites « Schemaless » ce qui signifie sans schéma défini. Cela veut tout simplement dire qu'il n'est pas nécessaire de définir au préalable les champs dans le document : on peut très bien en rajouter en cours de développement. Les documents peuvent être très différents les uns des autres au sein de la base. Le fait que les documents soient structurés permet d'effectuer des requêtes sur le contenu des objets.

Une interface d'accès HTTP REST est aussi disponible afin de permettre d'effectuer des requêtes sur la base. Mais cette interface est un peu plus complexe que l'interface CRUD des bases clé/valeur. Le plus souvent, ce sont les formats de données JSON et XML qui sont utilisés afin de garantir le transfert des données sérialisées entre l'application et la base.

Ce type de système accorde une popularité importante dans le milieu du développement de CMS ; cela vient du fait que ces bases disposent de fonctionnalités très intéressantes en termes de flexibilité du modèle documentaire par rapport aux bases de données relationnelles.

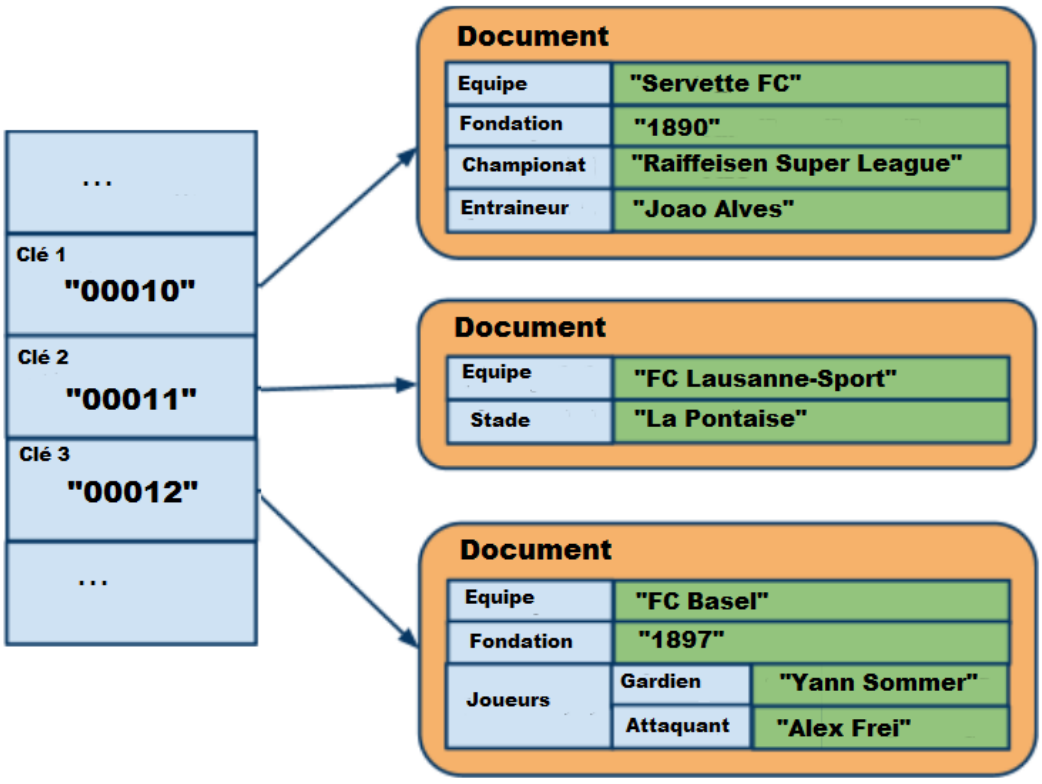


Figure 9 - Orienté document

9.4.3 Bases orientées colonnes

Aussi connues sous le nom de représentation à modèle extensible. Bien que le schéma soit relativement proche des bases orientées documents, les bases de données orientées colonnes sont probablement les plus complexes à comprendre parmi la mouvance NoSQL.

Comme il existe différents types de produits orientés colonnes, on trouve donc quelques variations selon le produit considéré. Le concept de base est décrit comme suit :

- Column : c'est l'entité de base qui représente un champ de données. Toutes les colonnes sont définies par un couple clé/valeur
- Super column : c'est une colonne qui contient d'autres colonnes
- Column family : elle est considérée comme un conteneur de plusieurs colonnes ou super-colonnes
- Row : c'est l'identifiant unique de chaque ligne de colonne
- Value : c'est le contenu de la colonne. Cette valeur peut très bien être une colonne elle-même.

Ce type de structure permet d'être plus évolutif et flexible ; cela vient du fait qu'on peut ajouter à tout moment une colonne ou une super-colonne à n'importe quelle famille de colonnes.

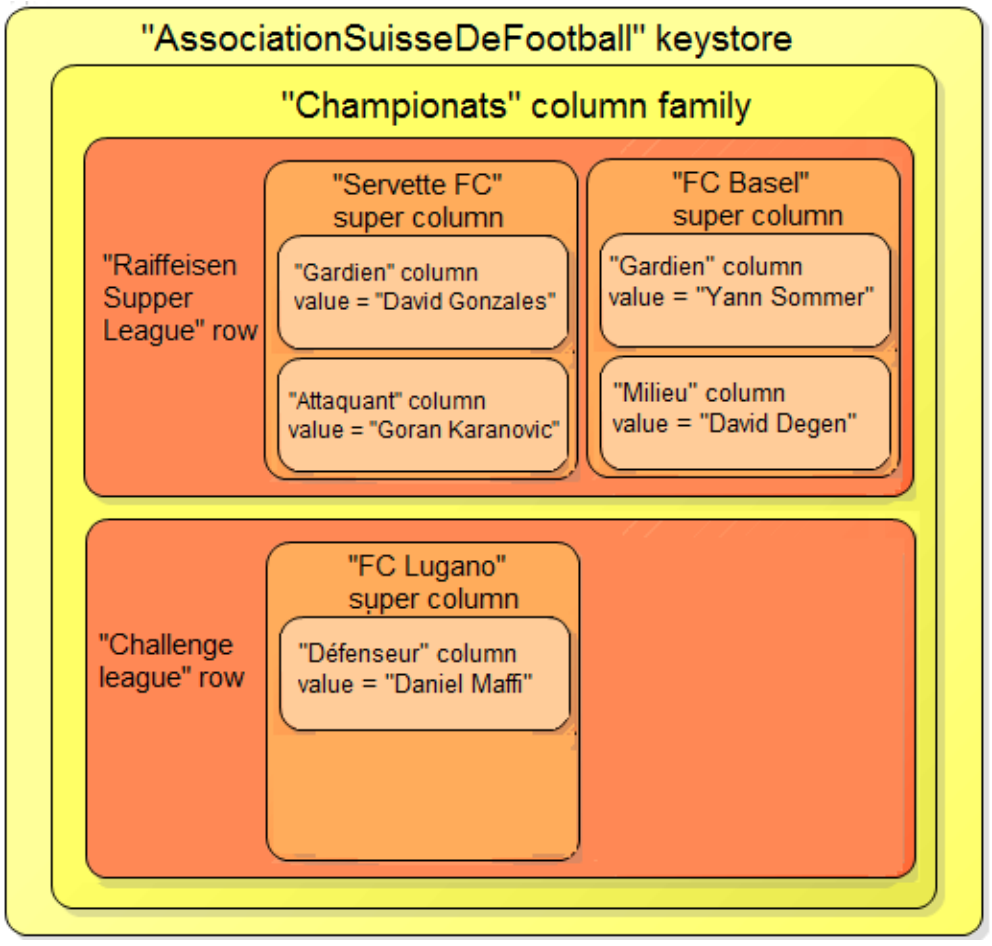


Figure 10 - Orienté colonne

9.4.4 Bases orientées graphe

Les bases orientées graphe sont les moins connues de la mouvance NoSQL. Ces bases permettent la modélisation, le stockage ainsi que le traitement de données complexes reliées par des relations. Ce modèle est composé d'un :

- Moteur de stockage pour les objets : c'est une base documentaire où chaque entité de cette base est nommé nœud
- Mécanisme qui décrit les arcs : c'est les relations entre les objets, elles contiennent des propriétés de type simple (integer, string, date..).

Pour le traitement des problèmes liés aux réseaux (cartographie, relations entre personnes sur les réseaux sociaux), les bases de données orientées graphe sont nettement plus efficaces que les SGBD de type relationnel. Cela vient du fait que lorsque l'on utilise le modèle relationnel pour obtenir des résultats, cela nécessite un grand nombre d'opérations dites complexes.

Le cas d'utilisation classique pour ce type de base de données est le stockage des informations des réseaux sociaux (relations d'amis sur Facebook, connaissances sur LinkedIn) ; néanmoins les bases orientiées graphe peuvent aussi être apte à modéliser de nombreuses autres situations qui ne pourraient être représentées que de manière réductrice dans une base de données relationnelles.

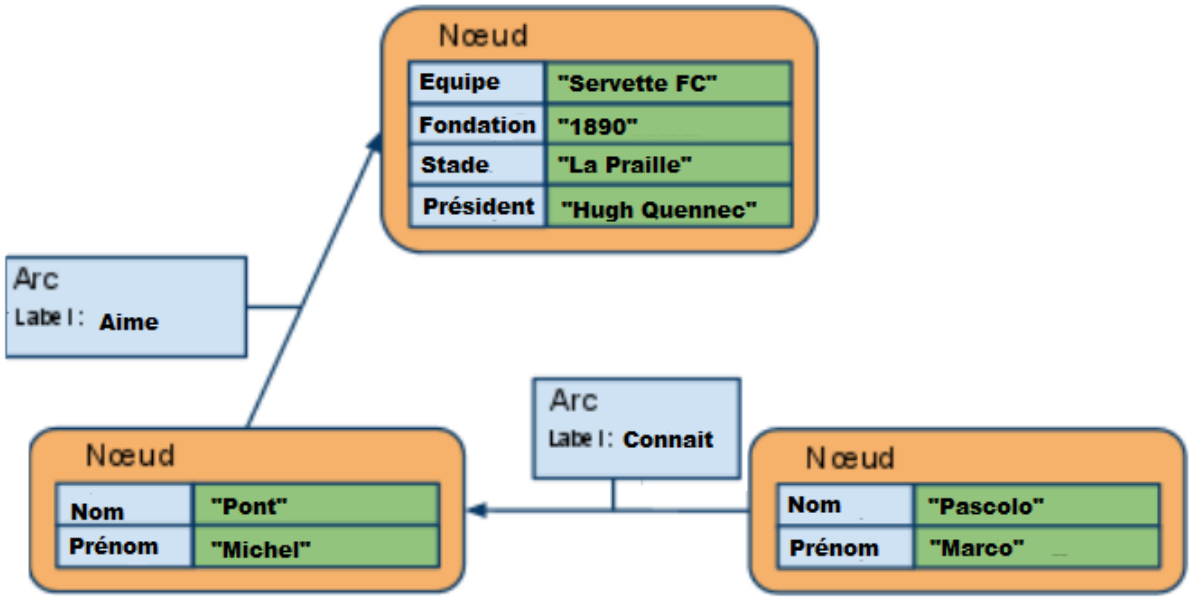


Figure 11 - Orienté graphe

9.5 Le marché

A l'heure actuelle, il y a plus de 122 solutions NoSQL tous types confondus (Clé/valeur, Document, Colonne et Graphe) sur le marché. La plupart d'entre elles sont soit sous licence Open Source ou soit proposées en SaaS. Les principaux contributeurs (Facebook, Google, Linkedin,…) de ces produits ont développé leurs outils à l'interne avec pour unique but de répondre à leurs propres besoins et non pas dans le but de les commercialiser. Lorsqu'ils se sont trouvés dans un état d'avancement très important (mise en production à l'interne), ces fameux SGBD ont été libérés et mis à disposition du grand public. Il ne faut pas oublier que la fondation Apache joue un grand rôle dans ces divers projets.

Je ne vais pas faire une description exhaustive de chacune des solutions sur le marché, mais simplement citer quelques unes des plus connues. Si toutefois vous souhaiter avoir un plus large aperçu, je vous invite à consulter le site *nosql-database.org* qui est le site de référence répertoriant tous les produits de type NoSQL.

9.5.1 Redis

Redis est un projet Open Source de type clé/valeur sous licence BSD. Il dispose de plus de fonctionnalités que la majeure partie des autres solutions du même type. Notamment par rapport au fait que Redis permet la manipulation des chaînes ainsi que les stockages de collections. Le principe de Redis est simple[8] :

- Tout est stocké en mémoire
- De temps en temps un « snapshot » des données est écrit sur le disque
- Au démarrage du service, il récupère le dernier « snapshot » persisté et le replace en mémoire.

En revanche, un des seuls points négatifs à noter est que Redis ne fournit pas de réel mécanisme de partitionnement, mais uniquement une réplication de type maître/esclave. Malgré cela Redis reste très performant. Il faut aussi ajouter que le projet est sponsorisé par la société VM Ware, ce qui rend la solution crédible et pérenne.

9.5.2 Voldemort

Voldemort est un projet Open Source de type clé/valeur sous licence Apache 2.0. Il a été nommé d'après un personnage du célèbre film « Harry Potter » et offre les caractéristiques suivantes[9] :

- Les données sont automatiquement répliquées sur de nombreux serveurs
- Les données sont automatiquement partitionnées afin que chacun des serveurs obtienne un sous-ensemble d'un ensemble de données
- L'échec d'un serveur est géré de manière transparente
- Les données sont « versionnées »
- Chaque nœud est indépendant des autres. De plus, chaque nœud du système atteint de hauts niveaux de performance de l'ordre des 20Ko d'opérations par seconde.

[8] Redis.io http://redis.io/
[9] Project Voldemort http://project-voldemort.com

Le projet Voldemort développé par les ingénieurs de LinkedIn a été mis en production à l'interne en 2009. Chez LinkedIn, Voldemort est utilisé pour certaines opérations de stockage de gros volumes de données, ce qui exige des performances accrues auxquelles les solutions de stockage simple ne peuvent répondre.

9.5.3 DynamoDB

DynamoDB est un SGBD de type clé/valeur développé par le géant de la vente en ligne Amazon. DynamoDB est un service basé sur le « cloud ». Grâce à cette solution, les clients s'affranchissent totalement de la gestion des serveurs, qui eux sont gérés par l'AWS (Amazon Web Service). Voici les caractéristiques de cette solution[10] :

- performance : les serveurs d'Amazon sont munis de disques durs SSD afin de garantir une certaine performance en lecture/écriture sur les disques.
- gestion : une console d'administration est mise à disposition pour la création de tables et la configuration de la capacité de requêtes souhaitées.
- API : utilisation des API d'Amazon pour la lecture/écriture.
- coût : les clients obtiennent 100 Mo de stockage gratuits et des performances permettant 5 écritures par seconde et 10 lectures par seconde. En cas de dépassement de cette capacité, des tarifs sont appliqués (prix d'entrée d'environ 1$ le Gigaoctet par mois). Pour plus de détails, je vous prie de consulter la liste des prix de DynamoDB qui se trouve sur le site officiel[11].
- sécurité : les données sont répliquées de manière synchrone sur 3 zones de disponibilités dans les data centers d'Amazon. L'accès aux données est intégré dans les outils globaux d'Amazon.
- EMR : l'intégration d'Elastic Map Reduce permet de gérer la distribution des données sur un nombre massif d'instances et de faciliter leur analyse.

DynamoDB permet de s'affranchir du coût de l'infrastructure et de sa maintenance. Cela permet de se concentrer sur le core-business de la société et ainsi laisser la gestion de ses infrastructures à des spécialistes. De plus, les serveurs sont équipés de disques durs SSD qui permettent d'augmenter la performance de manière drastique.

9.5.4 MangoDB

MangoDB est un système de la catégorie orientée documents, le plus populaire de sa catégorie. Il est écrit en C++ et Open Source sous licence AGPL v3.0. Cette solution a été développée par la société 10gen. Voici les caractéristiques du produit[12] :

- flexibilité du schéma : chaque document est libre de suivre sa propre structure.
- format de document JSON.
- garantit la « scalabilité » horizontale (réplication et « sharding »).
- support de recherche full-text, géo-spatiale et Map-Reduce.
- requêtes sur le contenu des documents.
- une large palette de pilotes est disponible pour divers langages de programmation.
- bonne documentation.

[10] DynamoDB http://aws.amazon.com/fr/dynamodb/
[11] Tarification DynamoDB http://aws.amazon.com/fr/dynamodb/#pricing
[12] MangoDB http://www.mangodb.org

- support après-vente bien développé.

MangoDB est en production dans plusieurs entreprises de grande envergure telles que gitHub, sourceforge et EA. Cela garantit la crédibilité du produit.

9.5.5 Cassandra

Cassandra est une solution de type orienté colonnes. Mise en Open Source par Facebook en 2008, c'est la fondation Apache qui a repris le flambeau et l'a mise sous licence Apache 2.0. Les caractéristiques de cette solution sont les suivantes[13] :

- flexibilité du schéma : chaque « table » est libre de suivre sa propre structure.
- représentation spatiale de la donnée (cela permet de traiter un grand nombre de lignes)
- « scalabilité » horizontale : s'il y a besoin de plus de puissance, on rajoute un serveur au cluster.
- plusieurs niveaux de stratégie de cohérence des données en écriture.
- réplication multi data-center.

Cassandra est utilisée par de nombreux acteurs de premier plan comme Facebook, Twitter et Cisco. Ceci garantit une certaine pérennité du produit. De plus une vaste communauté d'utilisateurs fait partager ses connaissances du produit et le font évoluer.

9.5.6 Neo4j

Neo4j est un système de base de données orienté graphe. C'est un projet Open Source sous licence GPLv3. Sa première version est sortie en 2007. Ce type de solution est utilisé dans le monde des réseaux sociaux (Ex : amis sur Facebook). Voici quelques caractéristiques du produit[14] :

- transaction complètement ACID.
- disponibilité haute.
- possibilité d'avoir plus de 64 milliards de nœuds/relations/propriétés sur une JVM.
- solution robuste (7ans en production sans interruption).
- intégration d'API (librairie Java).

Neo Technologie, qui est la société de développement de la base Neo4j, propose un excellent support technique ainsi qu'une documentation complète.

[13] Cassandra http://cassandra.apache.org/
[14] Neo4j http://neo4j.org/

9.6 Les avantages et les inconvénients du NoSQL

9.6.1 Les avantages

9.6.1.1 La « scalabilité » horizontale

Au lieu d'avoir misé sur une « scalabilité » horizontale, pendant longtemps les administrateurs de bases de données ont opté pour la « scalabilité » verticale. Ceci vient du fait qu'il est difficile d'adapter cette stratégie avec une base de données relationnelle.

Aujourd'hui, la rapidité en lecture/écriture ainsi que la haute disponibilité sont devenues des critères indispensables. C'est pourquoi les bases de données NoSQL répondent entièrement à ce besoin. Elles ont été conçues pour répandre les données de manière transparente sur plusieurs nœuds et ainsi former un cluster. Les performances qu'offre la « scalabilité » horizontale peuvent même être atteintes avec des serveurs bas de gamme, ce qui rend la structure plus économique.

9.6.1.2 Gros volume de données/« Big data »

Au cours de la dernière décennie, le volume de données à stocker a augmenté de manière massive. Les bases de données relationnelles ont augmenté leurs capacités afin de suivre la tendance. Mais avec cette constante augmentation de volume de données, il est devenu quasi impossible, pour un unique serveur de base de données relationnelle, de répondre aux exigences des entreprises en terme de performance. Aujourd'hui, ces gros volumes de données ne sont plus un problème pour les SGBD de type NoSQL, même le plus grand des SGBD relationnel ne peut rivaliser avec une base NoSQL.

9.6.1.3 Administrateur de bases de données (DBA) moins indispensable

Malgré les nombreuses améliorations vantées pendant des années par les fournisseurs de SGBD relationnels concernant la gestion de bases de données, un SGBD relationnel haut de gamme demande une certaine expertise pour sa maintenance. C'est la raison pour laquelle on fait généralement appel à un DBA, ce qui représente donc un certain coût. Les DBA sont intimement impliqués dans la conception, l'installation ainsi que dans le réglage des SGBD relationnels haut de gamme.

Il est erroné de dire que la présence d'un DBA n'est plus requise pour gérer une base de données NoSQL. Mais ses tâches seront facilitées notamment grâce à la distribution des données et surtout grâce à la simplicité du schéma de données. Il y aura toujours une personne responsable des performances ainsi que de la disponibilité quand des données critiques sont en jeu.

9.6.1.4 Solution économique

Les bases de données NoSQL ont tendance à utiliser des serveurs bas de gamme dont le coût est moindre afin d'équiper les « clusters », tandis que les SGBD relationnels, eux, tendent à utiliser des serveurs ultra puissants dont le coût est extrêmement élevé. De ce fait, les systèmes NoSQL permettent de revoir à la baisse les coûts d'une entreprise en termes de gigabytes ou de transactions par seconde. Cela permet de stocker ainsi que de manipuler plus d'informations à un coût nettement inférieur.

9.6.1.5 Modèle de données flexible

Changer le modèle de données est une vraie prise de tête dans une base de données relationnelle en production. Même une petite modification doit être maniée avec précaution et peut nécessiter l'arrêt du serveur pendant la modification ou limiter les niveaux de services.

Les systèmes NoSQL sont plus souples en termes de modèles de données, comme dans les catégories clé/valeur et documentaire. Même les modèles un peu plus stricts comme dans la catégorie orientée colonne permettent d'ajouter une colonne sans trop de problème.

9.6.2 Les inconvénients

9.6.2.1 Maturité

Les SGBD relationnels sont là depuis plus de 30 ans. Ainsi, cette pérennité est un signe de réconfort pour les responsables IT. Les SGBD relationnels sont de nature stable et riches en fonctionnalités. En comparaison, la plupart des SGBD NoSQL sont en pré-production et ont encore beaucoup de fonctionnalités futures à implémenter. Les spécialistes parlent d'une attente d'une bonne dizaine d'années avant de voir ce concept utilisé fréquemment par des applications critiques.

9.6.2.2 Support

Les entreprises veulent être rassurées quand c'est leur système de base de données qui est en jeu. Si le système subit une défaillance, ils veulent un support rapide et efficace. Les principaux fournisseurs de SGBD relationnels offrent un haut niveau de support pour les entreprises. Bien que certains fournisseurs proposent du support de qualité, ils ne peuvent bien évidement pas rivaliser avec un support mondial comme peuvent le faire Oracle, IBM ou Microsoft.

9.6.2.3 Administration

Un des buts des bases de données NoSQL était d'en simplifier l'administration, mais la réalité est différente. Aujourd'hui, les solutions NoSQL demandent beaucoup de compétences pour leur installation ainsi que des efforts conséquents lors de leur maintenance.

9.6.2.4 Expertise

Il y a dans le monde des millions de développeurs, opérant, dans différents businesses, qui sont familiers des concepts ainsi que de la programmation dans un environnement de bases de données relationnelles. Dans le monde NoSQL, presque tous les développeurs sont en apprentissage avec la technologie. Avec le temps, cette situation risque de changer, mais pour le moment, il est plus simple de trouver une personne ayant une grande expérience des bases de données relationnelles qu'un expert NoSQL.

10 Adoption d'une solution NoSQL dans l'entreprise

En lisant le chapitre précédent, on aurait pu croire que le problème de « Big data » ne concernait que les entreprises de type Facebook, LinkedIn, Google, eBay et Amazon. En effet, ces sociétés sont en grande partie les créateurs de nombreuses solutions NoSQL et l'utilisent en production.

En 2011, lors d'une tournée chez leurs plus gros clients de tous horizons confondus, les ingénieurs de VMware, grande société spécialisée dans la virtualisation, effectuent un sondage[15] dont la question principale, était : « Faites-vous du big data ? ». Les réponses furent claires, quasi aucune de ces sociétés n'était préoccupée par les gros volumes de données. Le sondage fut reconduit une année plus tard (2012) et les réponses étaient alors touts autres : en effet, la majorité d'entre elles s'étaient penchées sur le problème et certaines avaient même déjà déployé en production une solution NoSQL pour traiter leurs gros volumes de données.

Le « Big data » touche/intéresse de plus en plus de sociétés de tous horizons tels que les constructeurs automobiles, les grandes compagnies d'assurances et les grandes surfaces. Toutes les entreprises concernées par le « Big data » se penchent sur le problème en se posant les questions « comment l'exploiter ? » et surtout « comment stocker/lire ces gros volumes de données ? » ; c'est là que la technologie NoSQL entre en jeu. Les experts en stockage d'informations disent que 2012 est clairement l'année de l'adoption des technologies NoSQL par les entreprises dont les bases de données relationnelles ne peuvent plus répondre à leurs besoins.

L'adoption du NoSQL dans l'entreprise ne veut pas forcément dire abandonner les bases de données MySQL ou Oracle, mais plutôt procéder à une cohabitation des deux systèmes, c'est la raison même du terme NoSQL (Not only SQL, Pas seulement SQL). Il est ahurissant d'entendre un CIO d'une entreprise vouloir faire la transition de toute sa base de données relationnelles pour adopter une solution NoSQL uniquement. Cela est tout à fait contre-productif et peut mener à mal un projet. Il faut voir les solutions NoSQL comme un outil complémentaire et non comme une solution unique. L'entreprise est trop grande et complexe pour s'appuyer sur une solution « one-size-fits-all ».

Cette partie du dossier est destinée au CIO d'une entreprise qui souhaiterait faire le grand saut dans le monde du NoSQL, soit dans le cas d'une transition d'un projet existant ou tout simplement pour un nouveau projet.

Figure 12 - NoSQL?

[15] NoSQL & Big Data: Scaling the Enterprise into the New Age http://www.youtube.com/watch?v=1U73ns1L74E

10.1 Ai-je vraiment besoin du NoSQL dans mon entreprise ?

De nombreux CIOs font le choix de solutions NoSQL pour leurs entreprises juste à cause de la « hype » autour de ces nouvelles technologies. L'astuce, comme avec n'importe quel outil, c'est de savoir où l'on met les pieds. Selon Julian Browne, « Lead IT architect » au sein de la célèbre compagnie de télécommunications britannique O2, « Pour des raisons évidentes, il ne vaut vraiment pas la peine d'entreprendre un projet avec n'importe quel produit NoSQL, sauf si vous en avez besoin pour résoudre pour résoudre un problème ». Les technologies NoSQL répondent à de nombreux problèmes, mais de manière générale aux deux cas suivant :

- Gros volumes de données :
  Problème : Comment prend-t-on des montagnes de données afin de les stocker pour les manipuler et analyser ?
- Montées en charge :
  Problème : Comment gère-t-on des millions d'utilisateurs qui accèdent aux données en lecture et écriture simultanément ?

Lors d'une enquête menée par le groupe Unisphere Research concernant les gros volumes de données dans l'entreprise, 87% des entreprises ont affirmé que la croissance des données avait des répercussions négatives sur les performances de leurs applications. C'est une des raisons pour lesquelles nombre d'entre elles ont adopté des solutions NoSQL.

Vous n'avez certainement pas les montées en charge de Google ou d'Amazon, mais avez tout de même un nombre conséquent d'utilisateurs connectés simultanément à votre application et génèrent des transactions qui font augmenter le temps de latence. Alors, pour cette raison, une solution NoSQL pourrait être envisagée.

Si aucun de ces deux problèmes n'est identifié dans votre application actuelle ou future, il serait erroné de partir sur des conclusions type « le NoSQL ne s'adapte pas à mon application, donc je continue uniquement avec du relationnel ». En effet, il faut avoir un esprit visionnaire et penser sur le long terme afin de pas être victime de son propre succès et devoir ainsi réparer les pots cassés. Ceci peut engendrer plusieurs coûts qui ne sont pas des moindres : l'exemple le plus concret serait l'achat d'un nouveau serveur plus puissant afin de remplacer celui qui est essoufflé. Le nouveau serveur tient la route ? Oui, mais jusqu'à quand ?

Cependant, une étude menée en décembre 2011, auprès de 1300 développeurs par la société de développement CouchBase - qui fournit la base de données orientée document CoucheBase Server - démontre que le plus gros problème qui pousserait les développeurs à adopter une solution NoSQL vient de l'inflexibilité du schéma des bases de données relationnelles[16].

[16] NoSQL adoption is on the Rise : http://www.infoq.com/news/2012/02/NoSQL-Adoption-Is-on-the-Rise/

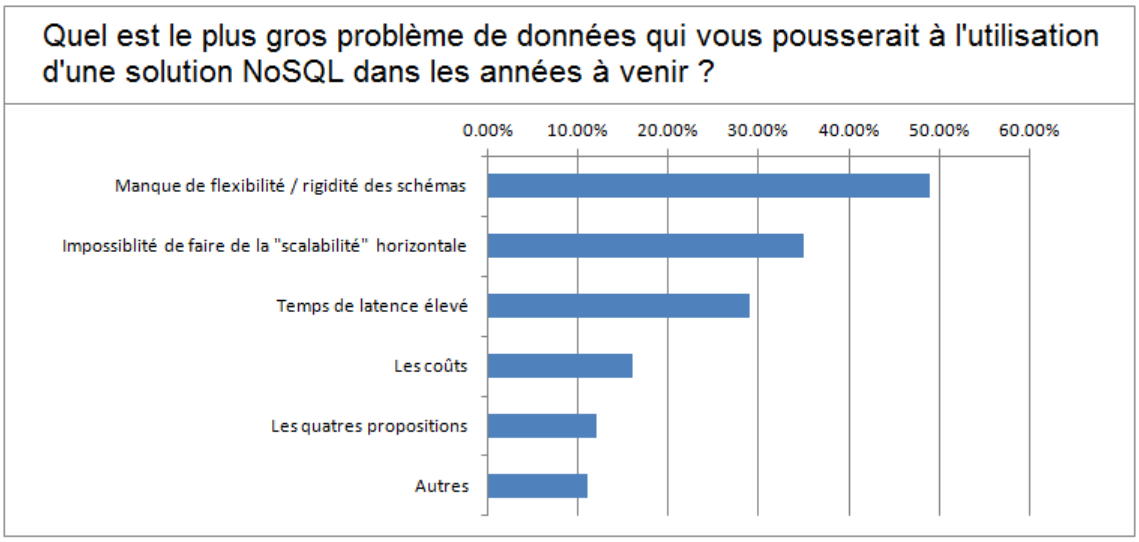


Figure 13 - Graphique - Les raisons d'adoption d'une solution NoSQL[17]

En effet, la flexibilité du schéma qui est proposé par les solutions NoSQL représente un grand avantage pour les sociétés dont certains modèles évoluent dans le temps. De plus, il est quasiment impossible, dans la plupart des entreprises, de définir un schéma qui répondrait aux besoins futurs. La gestion des changements dans les tables d'un modèle relationnel est une vraie prise de tête.

[17] Source: CoucheBase NoSQL Survey Decembre 2011

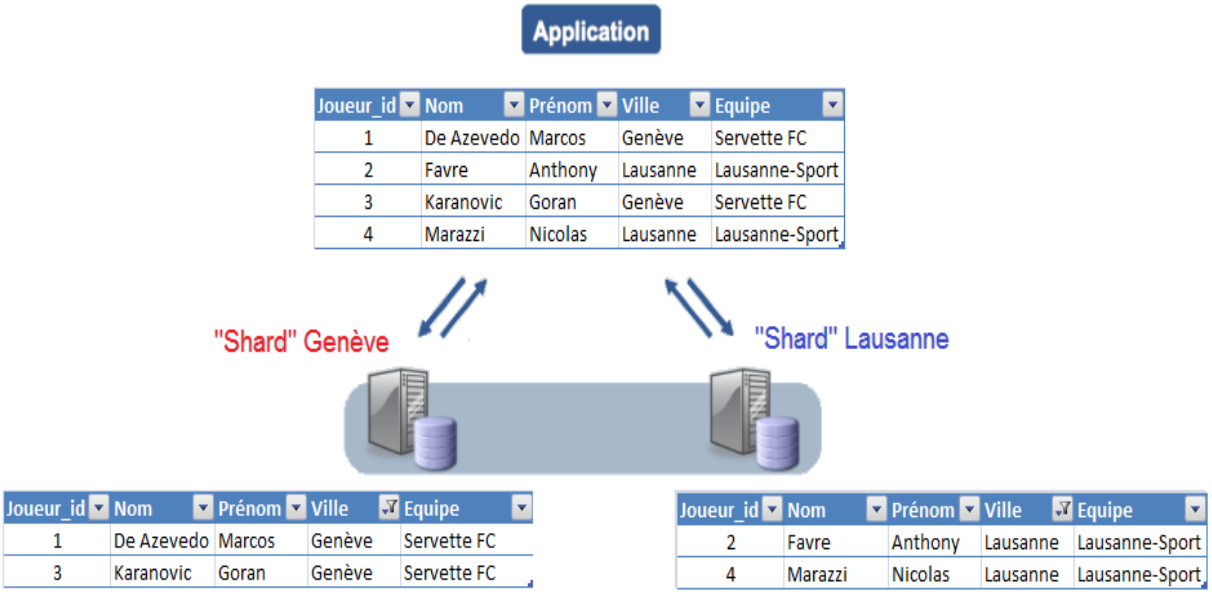
10.2 Comment les entreprises ont-elles pallié les problèmes de gros volumes de données et de montées en charges avec les SGBD relationnels ?

Il est bien connu que la plupart des humains n'aiment pas le changement. C'est la raison pour laquelle de nombreuses tactiques ont vu le jour, afin de permettre aux entreprises de continuer à utiliser des bases de données de type relationnel même dans des situations où cette technologie n'est peut-être plus aussi pertinente qu'elle ne l'était auparavant. Il faut également souligner que ces tactiques comportent elles aussi leurs défauts.

En voici quelques-unes communément utilisées dans les entreprises :

10.2.1 « Sharding »

Le modèle de données ainsi que le mécanisme transactionnel des bases de données relationnelles rendent nécessaire l'utilisation d'un serveur centralisé. Si un serveur unique est incapable de gérer simultanément les données en entrées/sorties de plusieurs utilisateurs, alors une tactique nommée « sharding » est fréquemment utilisée. Le principe est simple : les données d'une base sont réparties à travers plusieurs serveurs. Dans cette architecture, les serveurs deviennent des « shards ». Par exemple, les utilisateurs qui vivent à Genève vont avoir leurs données stockées sur un serveur, tandis que ceux vivant à Lausanne auront leurs données stockées sur un autre serveur.



Joueur_id	Nom	Prénom	Ville	Equipe
1	De Azevedo	Marcos	Genève	Servette FC
2	Favre	Anthony	Lausanne	Lausanne-Sport
3	Karanovic	Goran	Genève	Servette FC
4	Marazzi	Nicolas	Lausanne	Lausanne-Sport

Joueur_id	Nom	Prénom	Ville	Equipe
1	De Azevedo	Marcos	Genève	Servette FC
3	Karanovic	Goran	Genève	Servette FC

Joueur_id	Nom	Prénom	Ville	Equipe
2	Favre	Anthony	Lausanne	Lausanne-Sport
4	Marazzi	Nicolas	Lausanne	Lausanne-Sport

Figure 14 - Exemple de "Sharding"

Malgré le fait que cette tactique fonctionne très bien pour ce qui est des problèmes de montées en charge, elle comporte des conséquences indésirables :

- « Shard » rempli :
  Quand un « shard » est rempli, c'est toute la stratégie du « sharding » qui doit être modifiée. Reprenons l'exemple des deux « shards » : le premier contient les données des utilisateurs de Genève et le deuxième celles des utilisateurs

de Lausanne. Si le « shard » contenant les données des utilisateurs de Genève atteint la limite de sa capacité, cela voudra dire qu'il faudra changer la stratégie et donc modifier l'application. Auparavant, l'application devait seulement savoir si l'utilisateur venait de Genève et puis le redirigeait vers le bon serveur ; maintenant, avec la nouvelle stratégie, l'application doit savoir s'il vient de Genève et s'il habite sur la rive gauche pour taper dans le bon serveur.

- Perte d'importants avantages fournis par le modèle relationnel :
  Les jointures à travers plusieurs « shards » ne sont pas possibles. Par exemple, si vous voulez retrouver tous les clients ayant acheté un maillot de football, mais qui n'ont jamais acheté de chaussures, vous devez exécuter une requête sur chacun des serveurs et rassembler le tout dans l'application.

- Maintenir les schémas sur tous les « shards » :
  Sur le long terme, il est probable que de nouvelles informations auront besoin d'être collectées. Dans une telle situation, les schémas de bases de données devront être modifiés sur tout les « shards ». Suite à cela, il faudra une nouvelle fois normaliser, « tuner » et reconstruire les tables. Ce qui était difficile à faire sur un seul serveur devient un vrai cauchemar sur plusieurs. C'est pour cette raison que dans une architecture dite « sharding », il faut absolument éviter les changements dans le schéma de données.

10.2.2 Dénormalisation

Avant de stocker des données dans un SGBD relationnel, un schéma doit précisément définir les types de données qu'il va stocker ainsi que les relations entre les données. On dit alors que la base de données est dans sa « forme normale ». Afin de modifier un enregistrement, toutes les tables concernant ce dernier doivent être verrouillées et mises à jour de manière atomique. Cette approche limite considérablement le débit des mises à jour concurrentielles : de plus, il est impossible de l'implémenter à travers plusieurs serveurs.

JOUEUR	
id	nom
1	De Azevedo
2	Streller

JOUE	
id_joueur	id_equipe
1	2
2	1

EQUIPE	
id	nom
1	FC Basel
2	Servette FC

Figure 15 - "Forme normale"

Afin de pallier ces problèmes, les données sont fréquemment stockées dans une « forme dénormalisée ». Cette approche consiste à dupliquer certaines colonnes afin d'éliminer les jointures. Cette solution requiert la mise à jour de plusieurs tables lorsqu'une donnée d'une colonne dupliquée subit une modification.

JOUEUR	
id	nom
1	De Azevedo
2	Streller

JOUE	
joueur	equipe
De Azevedo	Servette FC
Streller	FC Basel

EQUIPE	
id	nom
1	FC Basel
2	Servette FC

Figure 16 - "Forme dénormalisée"

10.2.3 « Caching » distribué

Une autre tactique régulièrement adoptée par les entreprises est l'utilisation des technologies de « caching » distribué, tel que le fameux outil Memcached. Memcached est aujourd'hui devenu un outil incontournable dans les architectures de la plupart des applications web.

Situé entre les serveurs d'application et le SGBD relationnel, le « caching » distribué permet de garder en mémoire les données récemment accédées à travers un ou plusieurs serveurs (suivant la configuration). Quand une application veut accéder aux données, au lieu de directement aller chercher dans la base de données, elle va d'abord vérifier dans la couche « caching » pour voir si les données sont disponibles. Dans le cas contraire, l'application doit aller les lire les données dans le SGBD ; suite à cela, les données lues sont stockées dans la couche « caching » afin qu'elles soient accessibles plus rapidement lors d'une future requête de ces mêmes données.

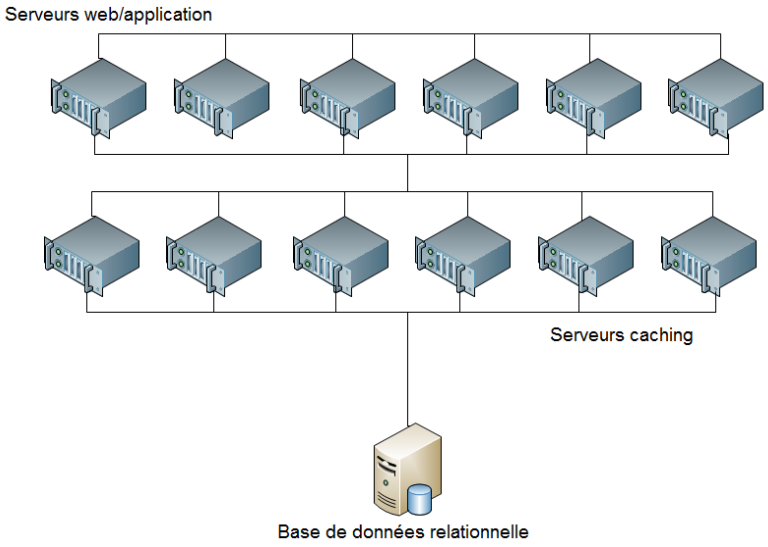


Figure 17 – "Caching" distribué

Malgré le fait que cette tactique de « caching » distribué est à la fois utile et efficace, elle comporte quelques petites défaillances :

- Elle augmente seulement les performances en lecture des données :
  Le « caching » distribué améliore uniquement les performances en lecture. Pour ce qui est des écritures, aucune amélioration n'est faite car, comme avant, les données doivent être stockées directement sur le SGBD centralisé avec toutes les contraintes que cela générait auparavant.

- La panne d'un serveur peut avoir de lourdes conséquences sur les performances :
  En cas de panne d'un des serveurs du cluster de « caching », cela peut avoir un sérieux impact sur les performances de l'application. En effet, si plusieurs utilisateurs recherchent des données qui étaient présentes sur le serveur tombé en panne, les requêtes seront redirigées vers le SGBD, cela ayant ainsi pour effet des augmentations de délais de réponse aussi bien en lecture qu'en écriture.

10.3 Les caractéristiques techniques essentielles d'une solution NoSQL pour l'entreprise

Nous avons vu qu'il existe une multitude de solutions NoSQL sur le marché (plus de 122 solutions), mais pas toutes répondent vraiment aux besoins de l'entreprise. Ici, je vous délivre les caractéristiques techniques qu'un CIO doit prendre en compte lors du choix d'un outil de type NoSQL. Certaines de ces caractéristiques ne sont pas forcement nécessaires suivant l'activité de l'entreprise.

10.3.1 Source de données primaires et secondaires

Une des caractéristiques les plus importantes pour une entreprise qui souhaite faire l'acquisition d'une solution NoSQL est le fait qu'elle soit capable d'accepter des données primaires provenant de diverses applications (interne, client, partenaire). Comme les données proviennent de diverses applications, l'outil doit pouvoir assimiler de manière très rapide tout type de données (structurées, semi-structurées et non-structurées). La solution NoSQL doit pouvoir offrir la possibilité de faire des requêtes de haute performance sur ces données.

Une fois que les données primaires sont dans la base de données, il convient de les analyser. C'est pourquoi la solution NoSQL doit pouvoir être « interfaçable » avec des outils de business intelligence afin d'analyser les données primaires et de stocker le résultat des analyses (données secondaires) dans la solution NoSQL

10.3.2 Fiabilité lors du stockage de données dites critiques

Certaines solutions NoSQL, lors du traitement des données, ont la caractéristique d'être cohérentes dans le temps (« eventual consistency »). En effet la cohérence des données est souvent compromise pour faire place à la « haute disponibilité » et à la « tolérance au partitionnement ». La conséquence de cette caractéristique tend à rendre les gens sceptiques voire même réticents quant à l'adoption d'une solution NoSQL dans l'entreprise, car elle ne permet pas de garantir une protection suffisante des données dites critiques.

Cependant, cela n'est pas le cas de toutes les solutions NoSQL. En effet, plusieurs d'entre elles, telles que Riak et Cassandra, permettent de configurer le niveau de cohérence sur les opérations désirées, ce qui représente, en termes de flexibilité, un avantage majeur. Ainsi les développeurs/architectes peuvent décider, suivant la situation, si la cohérence doit être forte ou faible. En termes de cohérence, ces deux solutions peuvent se comporter comme des SGBD relationnels quand c'est nécessaire ou bien peuvent fournir une « cohérence finale » quand le cas le permet.

10.3.3 Haute disponibilité

Dans une entreprise, le système de base de données doit être disponible en permanence, même en cas de panne d'un des serveurs du cluster. La configuration doit être de type « No Single Point of Failure » (aucun point de défaillance). C'est la raison pour laquelle une solution NoSQL, qui intégrera parfaitement l'entreprise, doit posséder le principe de haute disponibilité. En effet, les employés doivent pouvoir travailler aussi bien en lecture de données qu'en écriture.

10.3.4 Distribution des données entre plusieurs « data centers »

La réplication des données est une caractéristique que possèdent la plupart des SGBD relationnels, mais aucune d'entre elle n'offre une méthode simple avec des performances satisfaisantes, quant à la distribution de données entre plusieurs « data centers ». En effet, à l'heure actuelle, la majorité des entreprises ont besoin que leurs bases de données soient distribuées entre plusieurs « data centers » souvent situés dans des régions différentes.

Une solution NoSQL adoptable dans une entreprise doit permettre de façon simple la configuration de la distribution des données entre plusieurs « data centers ». Cela est possible grâce à un compromis entre la performance et la cohérence des données.

10.3.5 Réplication des données de façon transparente dans un environnement distribué

Un des gros problèmes lors de la distribution des données sur les SGBD relationnels ainsi que sur plusieurs solutions NoSQL est qu'ils sont dépendants d'une architecture de type « sharding » ou « maître/esclave ». Dans ces types d'architecture, la machine maître devient très souvent un goulot d'étranglement quand des opérations d'écritures intempestives sont exécutées, ce qui provoque une augmentation des temps de latence sur les machines esclaves.

Une bonne solution NoSQL permet l'écriture d'informations sur n'importe quel nœud du cluster. Suite à cette écriture, le système va répliquer l'information de manière complètement transparente sur tous les nœuds du cluster (peu importe leur localisation) qui ne la possèdent pas encore.

10.3.6 Disponibilité sur le « Cloud »

Malgré la réticence de quelques entreprises quant au fait de stocker des données confidentielles sur le « Cloud », les services « Cloud » sont de plus en plus utilisés. En effet, selon une étude menée par le cabinet IDC, ses experts prédisent que d'ici 2015 près de 20% des données totales seront rattachées d'une manière ou d'une autre aux services « Cloud » et qu'environ 10% de celles-ci seront stockées sur le « Cloud »[18].

Pour le moment votre entreprise ne stocke aucune donnée sur le « Cloud », mais peut-être dans un avenir proche ; dans un tel cas, la solution NoSQL adoptée devra être compatible « Cloud ».

10.3.7 Gestion des gros volumes de données

Vous n'avez peut-être pas le volume de données d'une entreprise comme Facebook, qui se calcule à plus de 4 milliards de données toutes confondues stockées par jour. La gestion des gros volumes de données ne concerne pas seulement les entreprises comme Facebook. En effet de nombreuses sociétés amassent des gigas, des téras voir même des pétas de données diverses et cela ne fait qu'augmenter avec le temps.

Les solutions NoSQL ne sont pas limitées à travailler avec des données de type « Big data ». En effet, la solution NoSQL choisie par l'entreprise doit lui permettre de gérer aussi bien des téras que des pétas de données quand il lui sera demandé. On ne sait

[18] Extracting Value from Chaos, de John Gantz et David Reinsel, IDC

pas de quoi demain sera fait alors il faut éviter à tout prix d'être victime de son propre succès.

Bien entendu, cela va au-delà du simple stockage de gros volumes de données. La solution NoSQL doit en effet fournir des performances en termes de vélocité (vitesse de lecture/écriture), variété des données (type de données) ainsi qu'en fonction de la complexité (données distribuées – multi data centers).

10.3.8 Performance linéaire lors d'ajout de serveur(s) au cluster

Afin de garder une performance linéaire lors de montées en charge, une solution NoSQL doit permettre la « scalabilité » horizontale, c'est-à-dire de pouvoir ajouter un ou plusieurs serveurs basiques au cluster sans interrompre le système. Sur certains systèmes, l'ajout de serveur(s) dans une configuration réduit parfois les performances. Dans une bonne solution NoSQL, c'est clairement le contraire qui se produit.

10.3.9 Flexibilité du schéma

La flexibilité du schéma de données est une caractéristique que possèdent la plupart des solutions NoSQL. Cette caractéristique est très importante pour les entreprises, car cela leur permet de traiter des données diverses (structurées, semi-structurées ou non-structurées) provenant de plusieurs sources différentes malgré le fait que la structure de données ne soit pas connue à l'avance. De plus, une modification de schémas dans une structure peut se faire à n'importe quel moment sans avoir besoin de mettre le système hors-ligne. Cela garantit une certaine disponibilité même lors d'une modification de schéma.

Il faut aussi noter que les schémas de données diffèrent suivant la catégorie (clé/valeur, orientée colonne et orientée document) à laquelle appartient la solution. Certains schémas s'adapteront probablement beaucoup mieux à votre entreprise que d'autres.

10.3.10 Système d'exploitation et langages de développement

Bien évidemment, une solution NoSQL adaptée à l'entreprise doit être compatible avec les systèmes d'exploitation les plus couramment utilisés. Il faudra aussi veiller à ce que la solution fournisse les librairies des langages (Clients API) qui sont utilisés dans votre entreprise. Cependant, si la solution est Open Source et que vous possédez des développeurs capables de programmer l'API, alors ceci ne sera pas un problème.

La plupart des développeurs ont sûrement tous une grande expérience avec les bases de données de type relationnel dont le langage de requête est standard (SQL) ; un plus serait que la solution possède un langage-type SQL pour les requêtes afin de faciliter la transition.

10.3.11 Facilité d'implémenter, de maintenir et d'évoluer

La solution NoSQL idéale pour l'entreprise ne doit pas être un casse-tête lors de son implémentation. Elle doit être simple à implémenter et à utiliser, mais elle doit quand même fournir un certain nombre de fonctionnalités capables de prendre en main de « grosses applications » d'entreprise.

La solution NoSQL fournie devrait aussi offrir de bons outils de gestion, tels que :

- une interface permettant le monitoring afin d'avoir un œil sur le comportement du système.
- une interface permettant la gestion des tâches administratives afin d'augmenter la capacité d'un cluster.

Evidemment, la solution NoSQL devrait permettre une certaine évolutivité sans qu'il soit nécessaire de procéder à des modifications dans l'application.

10.3.12 Vaste communauté Open Source

Si votre choix se porte sur une solution NoSQL Open Source, il est important qu'elle possède une vaste communauté active sur Internet (blogs/forums/mailings/gitHub). Cela permet de faire évoluer le produit grâce à différentes corrections de bugs et d'implémenter de nouvelles fonctionnalités. De plus, une vaste communauté permet d'avoir une garantie en termes de qualité ainsi qu'une pérennité de l'outil.

Il existe plusieurs indicateurs pour savoir si la communauté Open Source est vaste ou non. Il suffit pour cela de faire un tour sur les réseaux sociaux ainsi que les blogs et forums. L'organisation de conférences sur le produit est aussi un indicateur fiable.

10.4 Les caractéristiques non techniques essentielles d'une solution NoSQL pour l'entreprise

La solution NoSQL choisie par l'entreprise doit non seulement répondre à ses besoins techniques mais également à certaines caractéristiques non techniques ; qui sont les suivantes :

10.4.1 Pérennité du fournisseur

La pérennité du fournisseur est un facteur très important lors du choix d'une solution NoSQL. En effet, bien que cette dernière bénéficie d'une large communauté Open Source avec tous les avantages que cela comporte (si toutefois la solution NoSQL est Open Source), si elle ne dispose pas d'un éditeur solide, cela représente un grand risque sur le long terme pour les données de l'entreprise.

10.4.2 Service et support de qualité

Lors du choix d'une solution NoSQL pour l'entreprise, il faut également penser au support technique fourni par l'entité commerciale. En effet, si un problème technique survient sur la base de données en production, le problème doit évidemment être réglé de suite. La dernière chose qu'une entreprise devrait être obligée de faire dans de tels moments critiques est de n'avoir d'autre alternative que de demander de l'aide sur un forum, ceci sans garantie d'obtenir une réponse solutionnant le problème.

Les fournisseurs de solutions NoSQL doivent pouvoir offrir aux entreprises un support professionnel de qualité, généralement proposé sous forme d'abonnements annuels (« packages ») régis par des « Service Level Agreements », ainsi que des services de consultation pré- et postproduction. Ceux-ci permettent d'obtenir une aide ciblée lors de la pré- et postproduction.

Afin que le personnel IT de votre entreprise maîtrise le nouveau produit, il faudra veiller à ce que le fournisseur propose des séances de formation aussi bien théoriques que pratiques.

10.4.3 Documentation exhaustive et de qualité

Un des points qui est souvent négligé par les fournisseurs est la mise à disposition d'une documentation exhaustive et de qualité. Cette dernière doit être consultable en ligne et régulièrement mise à jour.

10.4.4 Références de clients

Les fournisseurs devraient être en mesure de prouver que leurs solutions NoSQL sont utilisées avec succès en production chez plusieurs clients de secteurs différents. Cela signifiera que l'outil touche ne touche pas un secteur d'activité restreint mais bien une grande variété de domaines d'activités.

10.4.5 Coût de la licence

En général, les solutions NoSQL proposent des prix de licence avantageux aux entreprises. Ceci est un avantage indéniable car lors de l'acquisition d'une licence d'un SGBD relationnel (Oracle, Microsoft) le coût en est très élevé, d'autant plus qu'il faut y ajouter un montant non négligeable pour la maintenance annuelle.

10.5 Cas d'utilisation dans l'entreprise

On a vu les types de problèmes auquels répondaient les technologies NoSQL. Maintenant je vais vous décrire quelques exemples de cas d'utilisation du NoSQL dans une entreprise[19] :

- Extraction de logs :
  Stockage de différents types de fichiers logs générés par les nœuds d'un cluster (Logs du serveur, logs d'applications, logs d'utilisateurs). Un outil de « Log mining » est idéal pour analyser ces fichiers logs et résoudre certains problèmes de production. Un outil de « Log mining » peut facilement être interfacé avec une base de données NoSQL.

- « Enterprise social software » :
  À l'heure actuelle, de nombreuses entreprises fournissent une plate-forme de communication afin que les employés, les clients et partenaires commerciaux puissent communiquer à travers des messages (forums/blogs/mp). Le stockage et l'exploration de ces données non structurées sont d'une grande importance pour les entreprises pour qu'elles puissent en tirer des informations et les exploiter en vue d'améliorer leurs services. Pour ce type de besoin, les bases de données NoSQL fonctionnent parfaitement.

- Intégration de flux de données externe :
  Dans bien des cas, les entreprises doivent faire face à l'intégration de flux de données externe venant la plupart du temps de partenaires commerciaux afin de récolter diverses informations. Les entreprises recevant ces flux ont peu de contrôle sur le format des données, et très souvent ce format change suivant les partenaires commerciaux. Ce problème peut vite être résolu avec l'adoption d'une solution NoSQL lors du développement/customisation d'une solution « Extract Transform Load » (ETL).

- Gros volume de données Intégration d'application d'entreprise (IAE) :
  À travers leur système IAE, la majorité des entreprises récolte de gros volumes de données. Ces informations, qui rentrent dans le système IAE, doivent être conservées avec attention afin de pouvoir les consulter, en cas d'audit par exemple. Encore une fois, les bases de données de type NoSQL peuvent tout à fait couvrir ces besoins du fait que les informations ne sont pas structurées de la même façon et que ce sont aussi de gros volumes de données que les technologies NoSQL gèrent très bien.

- « Content Management Service » (CMS) d'entreprise :
  Actuellement, les CMS sont utilisés dans tous les secteurs des entreprises (vente, marketing, ressources humaines, production) pour divers objectifs. Grâce à leur flexibilité sur les schémas de données, les solutions NoSQL sont d'excellents outils pour résoudre le problème de rassembler dans un même CMS les flux d'informations provenant de ces divers secteurs.

[19] Source: NoSQL in the enterprise http://www.infoq.com/articles/nosql-in-the-enterprise

10.6 Quels sont les cas ou le NoSQL n'est pas conseillé ?

Dans certains cas, il est fortement déconseillé d'adopter une solution NoSQL comme moteur de base de données. Selon Max Schireson, président de la société 10gen qui a développé MangoDB, il existe deux cas dans lesquels il déconseillerait fortement d'adopter une solution NoSQL[20] :

- Quand les schémas de données du système sont de nature complexe ; c'est-à-dire que les tables sont reliées par des jointures que l'on peut difficilement supprimer afin de tout mettre dans la même table : alors, dans ce cas, le NoSQL n'est vraiment pas conseillé.

- Quand le système est obligé de faire des transactions à instructions multiples. Par exemple, une transaction qui comporte plusieurs instructions de mise à jour, ce qui affecte plusieurs champs et que l'on confirme par un « commit » ou que l'on annule par un « rollback ».

Si l'un de ces deux cas est identifié, alors le CIO ne doit pas insister sur l'adoption d'une solution NoSQL mais rester sur un système de gestion de base de données relationnelle.

[20] NoSQL & Big Data: Scaling the Enterprise into the New Age http://www.youtube.com/watch?v=1U73ns1L74E

10.7 Les avantages du NoSQL dans l'entreprise

Dans cette section, nous allons voir les avantages dont peuvent bénéficier les entreprises grâce à l'adoption d'une solution NoSQL au lieu d'un SGBD relationnel.

10.7.1 « Business agility »

Les bases de données de type NoSQL peuvent aider les entreprises à devenir « Business agile ». En d'autres termes, elles leur permettent de répondre à des changements de manière rapide, à moindre coût et surtout sans paralyser le système. Et tout cela grâce aux deux caractéristiques suivantes :

- Schéma libre :
  Le fait que le schéma des données ne soit pas prédéfini permet de répondre rapidement aux changements avec quasiment aucun impact sur l'application et les fonctionnalités existantes.

- « Scalabilité » horizontale :
  Cela permet de pouvoir rajouter des serveurs bas de gamme au cluster sans pour autant interrompre le fonctionnement du système, ceci afin de répondre à une augmentation de charge de manière rapide et économe.

10.7.2 Satisfaction des utilisateurs finaux

Dans les entreprises d'aujourd'hui, un des indicateurs de qualité des applications est la satisfaction des utilisateurs finaux, ce qui est tout-à-fait normal, car ce sont eux qui vont passer la majorité de leur temps à travailler sur l'application. Les bases de données NoSQL aident les entreprises à atteindre la satisfaction des usagers finaux à plusieurs niveaux :

- Performance :
  Les bases de données NoSQL permettent d'augmenter drastiquement la performance des applications. On atteint cette performance grâce au concept de distribution des données. Ces réductions de temps de latence sont très satisfaisantes pour les utilisateurs finaux, cela évite des prises de tête devant l'écran et améliore aussi le rendement de travail effectué.

- Haute disponibilité :
  L'autre aspect qui concerne la satisfaction des usagers est la haute disponibilité des données. Les utilisateurs finaux doivent pouvoir accéder à l'application à n'importe quel moment de la journée afin de pouvoir travailler dessus quand ils le souhaitent. La non-disponibilité de l'application est une chose qu'il faut absolument éviter, au risque de mettre la productivité des utilisateurs finaux en péril. Afin de bénéficier de cette haute disponibilité, il faut s'assurer que la solution NoSQL choisie soit basée sur le concept de cohérence éventuelle des données.

10.7.3 Réduction des coûts

Dans les entreprises modernes, le budget pour l' « IT » est en permanence examiné avec attention afin d'éviter qu'elles soient victimes de dépenses inutiles. Les avantages des bases de données NoSQL résident dans le fait qu'elles peuvent réduire d'une

manière considérable les coûts et même permettre de réaliser des économies, le système fonctionnant en continu.

- « Scalabilité » horizontale :
  Grâce au concept de « scalabilité » horizontale, l'entreprise peut se permettre l'achat de serveurs dits « low cost ». Ceci ne va pas seulement réduire les coûts au niveau du hardware, mais aussi en termes d'électricité et de maintenance.

- Transaction de « Big data » :
  Sur le long terme, les coûts d'exploitation liés à la maintenance des SGBD relationnels ne sont pas des moindres, surtout lorsqu'ils traitent de gros volumes de données ; en effet cela demande l'intervention de spécialistes dont les compétences ont un certain coût. Par contre, pour les bases de données NoSQL, les gros volumes de données sont gérés systématiquement, même en cas d'augmentation drastique du volume de données. Cela permet aux développeurs de se consacrer uniquement au code de l'application et non au « tuning » de la base de données.

- Haute disponibilité :
  Certaines bases de données NoSQL disposent du principe de haute disponibilité. Les entreprises peuvent ainsi disposer d'une base de données qui est capable de tourner 24h/24h. Cela leur permet de n'avoir aucune interruption du « business » liée à des problèmes de base de données. Si un nœud du cluster tombe en panne, les employés peuvent continuer à travailler comme si de rien n'était, le chiffre d'affaires n'étant ainsi pas mis en péril.

10.8 Les difficultés à surmonter lors de l'adoption dans l'entreprise

Malgré tous les bénéfices que représentent les bases de données NoSQL pour l'entreprise, le CIO sera confronté à diverses difficultés qu'il devra inévitablement affronter afin que la solution soit adoptée avec succès.

10.8.1 Obtenir un financement

Dans le climat financier actuel, il est devenu très délicat pour un CIO d'obtenir des fonds afin de financer un nouveau projet. Billy Bosworth, CEO de la société DataStax, classe ce défi comme le plus difficile à surmonter[21]. En effet, la majorité des responsables de budget IT sont moins enclins à financer des projets, surtout dans ce climat d'incertitude financière au niveau macro-économique. Le CIO doit répondre à cela de manière efficace et en démontrer la rentabilité sur le long terme.

10.8.2 Fonctions et procédures stockées qui intègrent de la logique métier

Une question très importante qu'il faut se poser avant une transition du relationnel vers le NoSQL est de se demander : « Est-ce que dans mon application actuelle ma base de données intègre beaucoup de fonctions et de procédures stockées qui contiennent de la logique métier ? ». Donc, par exemple, si vous possédez un système de gestion de base de données de type *Oracle*, et qu'elle contient de nombreuses fonctions, procédures stockées et de triggers, qui eux-mêmes contiennent une grande partie de la logique métier, cela peut rendre la transition vers le NoSQL beaucoup plus difficile que prévu. La solution serait de déplacer le code de la couche base de données vers la couche applicative, donc de redévelopper le tout dans l'application. Tout cela a un certain coût suivant le nombre de procédures qu'il faudra redévelopper. Mais, dans le cas contraire où l'application a bien été architecturée, la transition devient nettement plus simple.

10.8.3 Identifier les données de l'application pouvant être gérées par une solution NoSQL.

Il est certain que dans un modèle de données d'une entreprise, une partie des données n'a besoin ni de relations, ni de contraintes ACID. Cependant, après des années de loyaux services entre les SGBD relationnels et les entreprises, il est devenu très difficile pour celles-ci d'identifier quelles données peuvent s'alléger dans une base de données NoSQL. La plupart du temps, les responsables IT n'ont pas une idée claire de la partie du modèle qu'ils vont devoir adapter, ce qui les contraint à rester sur du SGBD relationnel et donc d'abandonner la solution NoSQL.

10.8.4 Choisir le bon outil dans le bon contexte

Le choix de la solution peut être un sacré défi à surmonter. Effectivement, comme mentionné dans le chapitre précédent (cf. 9.5 Le marché), la technologie NoSQL regroupe plus de 122 solutions ayant différentes caractéristiques à travers 4 dimensions (clé-valeur, orienté document, orienté colonne et graphe). Parmi toutes les solutions qui sont proposées, il est très difficile d'en choisir une qui réponde à tous les besoins. Cela a même mené, dans certaines entreprises, à l'adoption de plusieurs solutions. Celle-ci étant devenues rapidement ingérables car non standardisées, la plupart des entreprises en question ont aussitôt laissé tomber les technologies NoSQL

[21] NoSQL for Big Data in the Enterprise http://www.youtube.com/watch?v=PoMsXvHkHG4

et sont vite revenues aux bons vieux SGBD relationnels. C'est la raison pour laquelle il faut bien connaître ses propres besoins ainsi que les caractéristiques que proposent les différents produits issus de la mouvance NoSQL.

10.8.5 Se projeter sur le long terme

Une autre difficulté qui touche le CIO, est le fait de pouvoir se projeter dans le futur. Il y a une grande différence entre le début d'un projet et son évolution deux ans plus tard en un immense cluster rempli de données. Le CIO doit se poser les questions suivantes « A quoi va ressembler le projet dans deux ans ? », « Quel sera le degré de complexité de la gestion des données ?, « Qu'en est-il de la gestion de la complexité », « Est-ce que la solution peut facilement exploiter le « cloud » ? ».

10.8.6 Formation du personnel

Certains ne la considèrent pas comme une vraie difficulté car un bon développeur doit savoir s'adapter rapidement aux nouvelles technologies grâce à une formation faite par le fournisseur pour les développeurs. Avec l'adoption d'une solution NoSQL, c'est la manière de penser le développement qui va changer. Par exemple, la gestion de l'intégrité des données se fera du côté application, donc cela signifie que les développeurs doivent être capables et accepter de faire du travail en plus.

Le vrai problème est quand le personnel montre sa réticence au changement et ne veut pas investir des années de formation dans une technologie qui n'est pas encore considérée comme « mainstream ».

10.8.7 Trouver le bon projet pilote

Une des difficultés souvent identifiée dans les grandes entreprises est le fait de trouver un bon projet pilote permettant de démontrer que les solutions NoSQL sont stables et rentables à long terme.

11 Check-list pour le CIO

Afin d'orienter le CIO vers la solution NoSQL la mieux adaptée aux besoins de l'entreprise, voici une liste de questions qu'un responsable IT doit se poser avant de faire un choix de solution. Les questions sont basées sur les caractéristiques techniques et non techniques des solutions proposées ci-dessous :

- Apache Cassandra[22] - DataStax[23] (Orienté colonne)
- MangoDB[24] – 10gen[25] (Orienté document)
- Riak - Basho[26] (Clé/Valeur)
- Project Voldemore[27] (Clé/valeur)

J'ai choisi ces bases de données de types différents (Clé/valeur, Orienté Colonne et Orienté document) afin de démontrer que ce n'est pas le type de base de données qui définit si c'est une solution adaptée à l'entreprise ou non.

Attention, Information importante :
Le CIO doit regarder de manière attentive le type de base de données NoSQL (cf. 9.4 Les types de base de données NoSQL) qui répond le mieux aux besoins de l'application. L'expérience peut être reconduite avec des solutions de même type.

11.1 Check-list : caractéristiques techniques :

La solution est-t-elle capable de stocker des données primaires ?

Apache Cassandra – DataStax	Oui, sans problème.
MangoDB – 10gen	Oui, sans problème.
Riak – Basho	Oui, sans problème.
Project Voldemort	Oui, sans problème.

Tableau 1 - Données primaires

La solution permet-elle l'interfaçage avec un outil d'analyse ?

Apache Cassandra – DataStax	Oui, il est possible d'interfacer Hadoop Analytics avec Hive et Pig.
MangoDB - 10gen	Oui, il est possible d'interfacer Pentaho Big Data Analytics.
Riak - Basho	Non, pas actuellement (il est difficile de faire de l'analyse sur les données provenant d'une base de type clé/valeur).
Project Voldemort	Non, pas actuellement (il est difficile de faire de l'analyse sur les données provenant d'une base de type clé/valeur).

Tableau 2 - Outil d'analyse

[22] Apache Cassandra http://cassandra.apache.org/
[23] DataStax http://www.datastax.com/
[24] MangoDB http://www.mongodb.org/
[25] 10gen http://www.10gen.com/
[26] Basho - Riak http://basho.com/
[27] Project Voldemort http://www.project-voldemort.com/voldemort/

La solution permet-elle de stocker de manière fiable des données critiques ?

Apache Cassandra – DataStax	Oui, malgré le fait que la solution privilégie la haute disponibilité à la cohérence, Cassandra permet de configurer le niveau de cohérence des opérations.
MangoDB – 10gen	Oui, MangoDB comporte le principe de cohérence.
Riak - Basho	Oui, malgré le fait que la solution privilégie la haute disponibilité par rapport à éa cohérence, Riak permet de configurer le niveau de cohérence des opérations.
Project Voldemort	Non, le Voldemort privilégie la haute disponibilité à la cohérence, Voldemort ne permet en aucun cas de configurer le niveau de cohérence des opérations.

Tableau 3 - Fiable pour les données critiques

La solution est-elle tolérante à la panne?

Apache Cassandra – DataStax	Oui, le cluster ne comprend aucun nœud de type « master ». En cas de panne d'un nœud du cluster, les autres nœuds du cluster sont toujours disponibles en lecture ainsi qu'en écriture
MangoDB – 10gen	Oui, grâce à la configuration de MangoDB en mode « replica-set » qui permet de représenter un « shard » par un cluster et cela de manière transparente. Le « replica-set » est représenté par plusieurs nœuds. Un seul nœud est de type « Master », les autres étant de type « Secondary ». En cas de panne d'un nœud de type « Master », un procédé d'élection est fait afin de déterminer le nouveau master. C'est un des nœuds de type « Secondary » qui devient « Master ».
Riak – Basho	Oui, le cluster ne comprend aucun nœud de type « Master ». En cas de panne d'un nœud du cluster, les autres nœuds sont toujours disponibles en lecture ainsi qu'en écriture.
Project Voldemort	Oui, en cas de panne d'un des serveurs, les écritures et les lectures sont redirigées vers un autre serveur disponible du cluster.

Tableau 4 - Tolérance à la panne

La solution permet-elle la distribution des données via plusieurs « data centers » ?

Apache Cassandra – DataStax	Oui, Cassandra permet la distribution à travers n « data center(s) »
MangoDB – 10gen	Oui, via plusieurs types de configuration, il est possible de faire de la distribution des données à travers plusieurs « data centers»
Riak – Basho	Oui, la version Enterprise de Riak permet la distribution à travers plusieurs « data centers»
Project Voldemort	Oui, Voldemort permet la distribution à travers plusieurs « data centers »

Tableau 5 - Plusieurs "data centers"

La solution peut-elle être intégrée dans le « cloud » sans difficulté ?

Apache Cassandra – DataStax	Oui, moyennant quelques configurations, Cassandra s'intègre facilement dans le « cloud »
MangoDB – 10gen	Oui, MangoDB peut facilement être déployé sur le cloud.
Riak – Basho	Oui, Basho, la société ayant développé Riak propose une solution cloud-based nommé Riak, CS compatible Amazon S3.
Project Voldemort	Non, pas actuellement.

Tableau 6 - Intégration dans le "cloud"

La solution est-elle capable de gérer de gros volumes de données ?

Apache Cassandra – DataStax	Oui, Cassandra peut stocker des petabytes de données.
MangoDB – 10gen	Oui, MangoDB peut stocker des petabytes d'information.
Riak – Basho	Oui, Riak peut stocker des petabytes de données.
Project Voldemort	Oui, Voldemort peut stocker des petabytes de données.

Tableau 7 - Gros volumes de données

La solution dispose-t-elle d'une performance linéaire lors de l'ajout de nœuds dans le cluster ?

Apache Cassandra – DataStax	Oui, grâce à l'ajout de nœuds dans le cluster, la performance augmente de manière linéaire. Aucune interruption du système n'est requise lors de l'ajout d'un nœud dans le cluster.
MangoDB – 10gen	Oui, du fait de sa configuration en auto-sharding, MangoDB permet de garantir une performance linéaire lors de l'ajout de serveurs et cela sans interrompre le fonctionnement du système.
Riak – Basho	Oui, Riak permet sans problème l'ajout de nœuds dans le cluster. Lors de l'ajout de nœuds, la performance reste linéaire.
Project Voldemort	Oui, Voldemort permet l'ajout de nœuds dans un cluster qui tourne sans pour autant interrompre le système.

Tableau 8 - Ajout de nœuds dans le cluster

La solution dispose-t-elle d'une certaine flexibilité au niveau du schéma de données ?

Apache Cassandra – DataStax	Oui, les schémas de données sont flexibles, cela permet de s'adapter rapidement au changement des besoins.
MangoDB – 10gen	Oui, MangoDB possède la propriété de schéma dynamique permettant une évolution du schéma.
Riak – Basho	Oui, Riak possède aussi la propriété de schéma flexible.
Project Voldemort	Oui, Voldemort permet aussi les schémas de données libres.

Tableau 9 - Flexibilité du schéma de données

La solution est-t-elle compatible avec les systèmes d'exploitation couramment utilisés ?

Apache Cassandra – DataStax	Cassandra est compatible sur les plateformes Windows, Mac OS X et Linux (Ubuntu, Red Hat, CentOS,) qui comportent une JVM récente
MangoDB – 10gen	MangoDB est compatible sur les plateformes Windows, Mac OS X, Linux (Debian, Ubuntu, Fedora, CentOS et Gentoo) et Solaris.
Riak – Basho	Riak est compatible sur les plateformes Mac OSX et Linux (Debian, Ubuntu, Rhel, CentOS et SUSE).
Project Voldemort	Voldemort est compatible sur les plateformes Windows et Linux (Debian) qui comportent une JVM récente.

Tableau 10 - Systèmes d'exploitation compatibles

La solution possède-t-elle des librairies client pour les langages utilisés dans votre entreprise ?

Apache Cassandra – DataStax	Librairie disponible pour les langages de développement suivants : C#, C++, Erlang, Java, JavaScript, OCaml, Perl, Php, Python, Ruby et XSD.
MangoDB – 10gen	Librairie disponible pour les langages de programmation suivants : C, C++, C#, Erlang, Haskell, Java, JavaScript, Perl, Php, Python, Ruby et Scala.
Riak – Basho	Librairie disponible pour les langages de développement suivants : C, C++, Erlang, Java, JavaScript, Php, Python et Ruby.
Project Voldemort	Librairie disponible pour les langages de développement suivants : Java, Php, Python et Ruby.

Tableau 11 - Librairies client

La solution possède-t-elle un langage de requête type SQL ?

Apache Cassandra – DataStax	Oui, Cassandra possède un langage de requête type SQL nommé CQL.
MangoDB – 10gen	Oui, MangoDB possède un équivalent de requête type SQL nommé Query Expression Objects.
Riak – Basho	Non, Riak ne possède pas de langage de requête type SQL, ce qui est normal car Riak est une base de données de type clé/valeur et il n'est donc pas possible de faire des requêtes complexes sur les valeurs. Riak offre tout de même quatre autres moyens de rechercher les valeurs de la base.
Project Voldemort	Non, Voldemort ne possède pas de langage de requête type SQL ce qui est normal car Voldemort est une base de données de type clé/valeur et il n'est donc pas possible de faire des requêtes complexes sur les valeurs.

Tableau 12 - Langage de requête type SQL

La solution possède-t-elle une interface graphique permettant le monitoring du système ainsi que la gestion des tâches administratives?

Apache Cassandra – DataStax	Dans la version DataStax Enterprise, une console graphique d'administration ainsi que de monitoring est disponible. La console se nomme DataStax OpsCenter.
MangoDB – 10gen	10gen, la société ayant développé MangoDB, n'offre pas de console graphique de monitoring et de gestion ; cependant la communauté Open Source de MangoDB a développé divers interfaces de monitoring et d'administration.
Riak – Basho	Riak offre une console graphique nommée Riak Contrôle permettant le monitoring ainsi que la gestion des nœuds du cluster.
Project Voldemort	Voldemort n'offre pas de console graphique, mais dispose d'une interface en ligne de commande permettant l'administration ainsi que le monitoring des nœuds du cluster.

Tableau 13 – Interface graphique de gestion/monitoring

La solution possède-t-elle une vaste communauté Open Source ?

Apache Cassandra – DataStax	Il existe une vaste communauté Open Source très active qui utilise Cassandra.
MangoDB – 10gen	La communauté MangoDB est très active sur les forums/blogs ; Il existe de nombreux composants, tels que patchs, API, librairies client développés par la communauté MangoDB.
Riak – Basho	La communauté est un peu moins développée que celle de Cassandra ou de MangoDB, mais elle est quand même présente sur la toile à travers GitHub, les blogs, IRC.
Project Voldemort	Voldemort possède une petite communauté Open Source. Il est possible de contribuer au projet par le biais de GitHub, IRC et Google groups.

Tableau 14 - Grande communauté Open Source

11.2 Check-list : caractéristiques non-techniques

Le fournisseur de la solution propose-t-il un support de qualité ?

Apache Cassandra – DataStax	DataStax propose un support de qualité pour les entreprises, avec un support 24h/24h, 7j/7j, 365 jours par an (par email, web, téléphone, dépannage sur site et développement de patches) avec des temps de réponse de maximum 1heure.
MangoDB – 10gen	10gen fournit aussi un support de qualité pour les entreprises avec différents types de package (Silver et Gold). Le package Gold prévoit un support de 24h/24h, 7j/7j, 365 jours par an, des temps de réponse de maximum 1heure, support par téléphone, email et possibilité de développer des patches d'urgence.
Riak – Basho	Basho offre aussi un support assez robuste pour les entreprises. Elle propose à celle-ci un support de 24h/24h, 7j/7j, 365 jours par an.
Project Voldemort	Non, il n'existe aucune entité commerciale derrière le Project Voldemort qui puisse offrir un support du produit.

Tableau 15 - Support de qualité

Le fournisseur de la solution propose-t-il des séances de formation ?

Apache Cassandra – DataStax	DataStax propose des séances de formation à travers tout le territoire des USA. Mais sur demande, il est possible d'organiser une séance à l'interne de l'entreprise.
MangoDB – 10gen	10gen propose différents types de séances de formation (Développeur, Administrateur, Essentiel, Atelier). Elles sont effectuées aux quatre coins du monde. Il est aussi possible, sur demande, d'effectuer des séances de formation à l'interne.
Riak – Basho	Basho propose aussi des séances de formation sur demande.
Project Voldemort	Voldemort n'ayant pas d'entité commerciale, il est très rare que des séances de formation aient lieu.

Tableau 16 - Séances de formations

Le fournisseur de la solution fournit-il une documentation exhaustive et de qualité ?

Apache Cassandra – DataStax	Une documentation exhaustive et de qualité est disponible en ligne sur le site de la solution.
MangoDB – 10gen	Une documentation exhaustive et de qualité est disponible en ligne sur le site de la solution
Riak – Basho	Une documentation exhaustive et de qualité est disponible en ligne sur le site de la solution.
Project Voldemort	Une petite documentation est disponible en ligne.

Tableau 17 - Documentation exhaustive et de qualité

Le fournisseur de la solution possède-t-il des clients/références de domaine d'activité hétérogène ?

Apache Cassandra – DataStax	Oui, voici une liste non-exhaustive de clients qui utilisent DataStax – Apache Cassandra en production: Backupify, Digital Reasoning, Formspring, HealthCare Anytime, Next Big Sound etc...
MangoDB – 10gen	Oui, voici une liste non-exhaustive de clients qui utilisent MangoDB en production : Disney, Craigslist, Forbes, O2, Struq, MTV network, Wordnik, etc…
Riak – Basho	Oui, voici une liste non-exhaustive de clients qui utilisent Riak en production : BestBuy, GitHub, Comcast, BrainTree, Yammer, Voxer, etc…
Project Voldemort	Il y a peu d'entreprises qui utilisent Voldemort en production ; voici la liste des références : LinkedIn (ce sont les développeurs du Project Voldemort), Gilt Group et KaChing.

Tableau 18 - Clients/références de domaine d'activité hétérogène

Le fournisseur de la solution propose-t-il des licences à prix avantageux?

Apache Cassandra – DataStax	Le prix des licences d'entreprises est uniquement disponible par email.
MangoDB – 10gen	Le prix des licences d'entreprises est uniquement disponible par mail, mais l'achat du package de type « Gold » octroie une licence ($4000/serveur/par an).
Riak – Basho	Le prix des licences d'entreprises est uniquement disponible par email.
Project Voldemort	Gratuite.

Tableau 19 - Coût de la licence

11.3 Résultats obtenus

Sans grande surprise, les deux solutions ayant répondu de manière positive à toutes les questions sont Apache Cassandra - DataStax et MangoDB – 10gen. Ces deux cadors de la mouvance NoSQL s'adaptent très bien à l'entreprise. Talonné derrière par Riak - Basho qui s'adapte lui aussi très bien à l'entreprise malgré quelques réponses négatives, tandis que la solution Project Voldemort est fortement déconseillée à mettre en production dans une entreprise. Ceci est notamment dû au fait que cette solution ne dispose d'aucune entité commerciale pouvant fournir un support qui fait alors cruellement défaut.

12 Conclusion

Au début de ce travail, je vous ai cité les trois types de personnes identifiés par l'auteur Shashank Tiwari (cf. 9.1 Introduction au NoSQL) et je disais que j'étais dans le groupe de celles qui connaissaient peu le NoSQL. A présent je me situe dans le groupe des personnes qui l'adulent. En effet, je suis persuadé que les entreprises ne peuvent que tirer un maximum de bénéfices d'une solution NoSQL, à condition qu'elle soit bien choisie et utilisée dans le bon contexte.

Il est certain que les bases de données relationnelles sont irremplaçables dans les entreprises et cela n'a jamais été le but des solutions NoSQL de les remplacer. Mais il est évident que, dans certains cas, les bases de données relationnelles sont utilisées dans des contextes pour lesquelles elles n'ont pas été créées, notamment pour gérer les changements de schéma de données, les gros volumes de données ainsi que les montées en charge. Donc, la solution n'est pas le remplacement du relationnel par du NoSQL, mais bien entendu une cohabitation des deux systèmes.

Je suis persuadé que d'ici quelques années, les technologies NoSQL deviendront des produits incontournables pour les applications d'entreprise. On remarque que de plus en plus d'entreprises sont intéressées par ces solutions NoSQL, et beaucoup en ont déjà adopté une. Pour preuve, il n'y a qu'à voir les nombreux clients de tous domaines confondus que possèdent les fournisseurs tels que Basho Technologies, 10gen et DataStax, qui utilisent une solution NoSQL en production.

Les bases de données NoSQL sont récentes, leurs fournisseurs n'ont en effet pas la même expertise que celle des fournisseurs de base de données relationnelles tels qu'Oracle. Cela va être le rôle des fournisseurs des solutions NoSQL de les rendre robustes aux yeux des entreprises qui hésitent à les adopter, ceci grâce à une couverture mondiale.

Pour conclure, ce document va ainsi permettre au CIO d'avoir un premier contact avec ces nouvelles technologies. Dans le cas où il pense que l'adoption d'une solution NoSQL se trouve être appropriée, la suite logique serait de commencer par un projet pilote avec la solution qui convienne le mieux. A cet égard, il convient de contacter d'abord le fournisseur.

13 Bilan personnel

Lors de la réalisation de ce mémoire de Bachelor, j'ai pris énormément de plaisir à travailler sur le sujet, d'autant plus qu'avant le début de ce travail, je n'avais quasiment aucune notion de ce qu'était le NoSQL. Ma première interaction avec le NoSQL fut très brève. C'était lors du projet d'intégration professionnelle (GREP) où mon groupe de projet et moi-même avions développé une application Web dans le Framework Symfony 2. Et lors d'un tutoriel que j'ai suivi sur ce dernier, on me demandait quel type de base de données j'allais utiliser pour réaliser mon application, MySQL ou MangoDB ? J'ai aussitôt tapé MangoDB dans le moteur Google search afin de voir ce que c'était. Et c'est la que j'ai découvert le mouvement NoSQL. A l'époque du GREP, je n'avais pas le temps de m'attarder sur le sujet, alors j'ai décidé que j'allais le garder pour mon travail de Bachelor.

De plus, le NoSQL est un sujet qui n'a jamais été traité lors de ma formation à l'HEG et, vu l'émergence des technologies NoSQL dans les entreprises, cela devient une plus-value pour mon CV.

Pendant les huit semaines de rédaction, j'ai aussi eu l'occasion d'assister à la conférence « NoSQL Roadshow » qui s'est déroulée dans la ville de Bâle le jeudi 30 Août 2012. Lors de cette conférence, de nombreux intervenants ont partagé leurs expériences du NoSQL. Cela fut très enrichissant pour moi.

J'aurais vivement souhaité expérimenter la partie technique du NoSQL, l'ayant prévu dans ma table des matières. Toutefois, le temps imparti pour la rédaction de mon mémoire n'était pas suffisant pour me permettre de le faire et je le regrette. Cependant, je ne compte pas en rester là et prévois de développer une petite application Web avec une base de données NoSQL, plus précisément MangoDB.

14 Glossaire

B

B.I. : Terme anglais qui veut dire Business Intelligence (en français l'informatique décisionnelle), désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider, modéliser et restituer les données, matérielles ou immatérielles, d'une entreprise en vue d'offrir une aide à la décision et de permettre à un décideur d'avoir une vue d'ensemble de l'activité traitée.

BSD (Licence) : (Berkeley software distribution license) est une licence libre, qui diffère de la licence GNU GPL, utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire.

C

CPU : Central Processing Unit (en français Microprocesseur)

Cluster : Ensemble logique de serveurs qui garantissent une haute disponibilité des ressources et une répartition des charges de traitement.

CMS : (Content Management System), ou SGC est un Système de Gestion de Contenu web permettant de construire très rapidement et simplement un site web statique ou dynamique (voire mixte).

CRUD : Désigne les quatre opérations de base pour le stockage d'informations en base de données. Soit : Create, Read, Update et Delete. C'est-à-dire : créer, lire, mettre à jour et supprimer.

E

EAI: C'est l'acronyme de « Enterprise Application Integration », qui est une architecture intergicielle permettant à des applications hétérogènes de gérer leurs échanges. On l'a place dans la catégorie des technologies informatiques d'intégration métier (Business Integration) et d'urbanisation. Sa particularité est d'échanger les données en pseudo temps réel.

ETL: C'est l'acronyme de « Extraction, Transformation and Load ». Ceci définit un logiciel assurant l'extraction de données depuis un système de production, ainsi que leur conversion et leur intégration dans le système décisionnel (SIAD) associé.

F

Fad: Mot anglais définissant une mode qui est passagère

G

Git : C'est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, le créateur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.

H

Hashmap : (collection associative sans répétition des clés) Représentation sous forme de tableau associatif de l'interface Map. Permet de stocker tout type de valeur (incluant null) et de clé (incluant null). Ne donne aucune garantie sur l'ordre de parcours des clés (ne se fait ni en fonction de leur ordre naturel, ni en fonction de leur ordre d'insertion). L'ordre des clés déjà stockées peut même changer en fonction des opérations effectuées.

Hype cycle : Le « hype cycle » est une courbe que suivent les nouvelles technologies. Cette courbe montre comment les nouvelles technologies sont acceptées, par quelles étapes elles passent et quand elles commencent à être rentables. La dimension de temps est très importante dans le « hype cycle ».

N

NSPOF : C'est l'acronyme de No Single Point Of Failure (En français Aucun point de défaillance). Un système informatique est dit NSPOF si la défaillance d'un de ses composants n'a aucun impact sur son fonctionnement.

O

Open Source : Mouvement prônant la diffusion libre des codes source. Pour les non-professionnels, surtout reconnu pour la gratuité résultante des logiciels dans de nombreux cas d'utilisation. Un logiciel Open Source n'est pas dans le domaine public, sa seule différence avec un logiciel propriétaire traditionnel est le fait que l'éditeur, ou plus généralement le titulaire des droits de propriété intellectuelle, a choisi de diffuser les sources de son logiciel, en permettant la libre utilisation.

S

SaaS : Software as a Service ou en francais logiciel en tant que service est un concept consistant à proposer un abonnement à un logiciel plutôt que l'achat d'une licence. Avec le développement des technologies de l'information et de la communication, de plus en plus d'offres SaaS se font au travers du web. Il n'y a alors plus besoin d'installer une application de bureau, mais d'utiliser un programme client-serveur.

Scale in : Le principe du scale veut dire que le système permet l'ajout de ressources à l'unique nœud du système, cela dans le but d'augmenter les performances.

Scale out : le principe du scale out veut dire que le système permet l'ajout de nœuds au système, cela dans le but d'augmenter les performances.

SPOF : C'est l'acronyme de Single Point Of Failure (en francais Un point unique de défaillance). Un SPOF est un point d'un système informatique dont le reste du système est dépendant et dont une panne entraîne l'arrêt complet du système.

Subversion : C'est un logiciel de gestion de versions, distribué sous licence Apache et BSD

T

Tweets : Nom donné aux messages limités à 140 caractères postés sur le réseau de micro-blogging Twitter. Un tweet utilisant l'intégralité des caractères est appelé un Twoosh.

Bibliographie

TIWARI, Shashank. Professional: NoSQL. Indianapolis: Wrox, 2011. P. 361

FOUCRET, Aurélien. NoSQL : Une nouvelle approche du stockage et de la manipulation des données [en ligne]. Paris : Smile Open SourceSolution, 2012. 55 p. http://www.smile.fr/Livres-blancs/Culture-du-web/NoSQL (consulté le 04.07.2012)

CATTELL, Rick. Scalable SQL and NoSQL Data Stores [en ligne]. 2010, vol. 39. 4, 16 p. http://www.cattell.net/datastores/Datastores.pdf (consulté le 12.07.2012)

15 Webographie

Wikipédia – Définition de la base de données
http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es (consulté le 10.07.2012)

Wikipédia – Définition de la base de données hiérarchique
http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_hi%C3%A9rarchique (consulté le 10.07.2012)

Wikipédia – Définition de la base de données réseau
http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_r%C3%A9seau (consulté le 10.07.2012)

Wikipédia – Définition de la base de données relationnelle
http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle (consulté le 10.07.2012)

Comment ça marche - Base de données
http://www.commentcamarche.net/contents/bdd/bddintro.php3 (consulté le 10.07.2012)

Your ultimate guide to Non-Relational Universe http://www.nosql-database.org (consulté le 04.07.2012)

Wikipédia – Transaction informatique
http://fr.wikipedia.org/wiki/Transaction_informatique (consulté le 11.07.2012)

Journal du net - Expliquez-moi... Les propriétés ACID d'une base de données
http://www.journaldunet.com/developpeur/tutoriel/theo/060615-theo-db-acid.shtml (consulté le 11.07.2012)

Wiktionary – Scalabilité http://fr.wiktionary.org/wiki/scalabilit%C3%A9 (consulté le 15.08.2012)

Developpez.com - SGBDR et répartion de charge : "scalabilité"
http://blog.developpez.com/sqlpro/p10383/langage-sql-norme/sgbdr-et-repartion-de-charge-scalabilite/ (consulté le 15.08.2012)

Neoxia.com – Scalabilité, le choix des armes http://blog.neoxia.com/scalabilite-choix-des-armes/ (consulté le 11.07.2012)

Neoxia.com – Scalabilité, oui. Mais élastique http://blog.neoxia.com/elasticite/ (consulté le 11.07.2012)

Neoxia.com – NoSQL : 5 Minutes pour comprendre http://blog.neoxia.com/nosql-5-minutes-pour-comprendre/ (consulté le 04.07.2012)

10 minutes pour comprendre le NoSQL
http://davidmasclet.gisgraphy.com/post/2010/06/09/10-minutes-pour-comprendre..NoSQL (consulté le 04.07.2012)

Bases de données: le noSQL menace 25 ans de certitudes http://www.indexel.net/infrastructure/bases-de-donnees-le-nosql-menace-25-ans-de-certitudes-3570.html (consulté le 04.07.2012)

NoSQL for beginners http://porcelli.com.br/archiv/nosql-for-beginners/ (consulté le 05.07.2012)

Octo.com - No :sql(eu) et NoSQL : qu'en est il ? http://blog.octo.com/nosqleu-et-nosql-quen-est-il/ (consulté le 05.07.2012)

Base de données : Petit état des lieux du NoSQL http://linuxfr.org/news/petit-etat-des-lieux-du-nosql (consulté le 13.07.2012)

NoSQL http://manurenaux.wp.mines-telecom.fr/2012/04/18/nosql/ (consulté le 13.07.2012)

Conception et dénormalisation de bases de données http://manuals.sybase.com/onlinebooks/group-asarc/asg1150f/perf/@Generic__BookTextView/1248 (consulté le 24.08.2012)

Big data des nouvelles opportunités http://www.touilleur-express.fr/2012/05/26/bigdata-de-nouvelles-opportunites/ (consulté le 19.07.2012)

Big data: Gesticulation marketing ou réalité? http://www.legrandbi.com/2011/08/bigdata-marketing-ou-realite/ (consulté le 19.07.2012)

CoucheBase - NoSQL Database Technology www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQL-Whitepaper.pdf (consulté le 24.08.2012)

Big data : what else ? http://www.osbi.fr/?p=3347#more-3347 (consulté le 19.07.2012)

10 things you should know about NoSQL databases http://www.techrepublic.com/blog/10things/10-things-you-should-know-about-nosql-databases/1772 (consulté le 23.07.2012)

NoSQL for the rest of us http://econsultancy.com/ch/blog/7812-nosql-for-the-rest-of-us (consulté le 23.07.2012)

Visual Guide to NoSQL Systems http://blog.nahurst.com/visual-guide-to-nosql-systems (consulté le 23.07.2012)

What the heck are you actually using NoSQL for? http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html (consulté le 23.07.2012)

NoSQL in the Enterprise: A Guide for Technology Leaders and Decision-Makers http://tek-tips.nethawk.net/nosql-in-the-enterprise-a-guide-for-technology-leaders-and-decision-makers/ (consulté le 23.07.2012)

NoSQL for the enterprise http://www.dbta.com/Articles/Editorial/Trends-and-Applications/NoSQL-for-the-Enterprise-80198.aspx (consulté le 23.07.2012)

Dbmsmusings - Problems with CAP, and Yahoo's little known NoSQL system http://dbmsmusings.blogspot.ch/2010/04/problems-with-cap-and-yahoos-little.html (consulté le 08.07.2012)

What you need to know to move form a relational to a NoSQL database http://fr.slideshare.net/Dataversity/what-you-need-to-know-to-move-from-a-relational-to-a-no-sql-database (consulté le 18.07.2012)

An introduction to NoSQL patterns http://architects.dzone.com/articles/introduction-nosql-patterns (consulté le 28.07.2012)

NoSQL adoption is on the rise, a new survey suggests http://www.infoq.com/news/2012/02/NoSQL-Adoption-Is-on-the-Rise/ (consulté le 28.07.2012)

James Philips on moving from relational to NoSQL Databases http://www.infoq.com/news/2011/12/relational-nosql-databases/ (consulté le 05.08.2012)

Le volume de données transactionnelles augmente http://www.informatica.com/fr/vision/harnessing-big-data/oltp-and-analytics/ (consulté le 19.07.2012)

The grill : Doug Cutting http://www.computerworld.com/s/article/9222758/The_Grill_Doug_Cutting (consulté le 06.08.2012)

The Database Tea Party: The NoSQL Movement http://kellblog.com/2010/02/24/the-database-tea-party-the-nosql-movement/ (consulté le 23.07.2012)

Brewer CAP theorem http://www.julianbrowne.com/article/viewer/brewers-cap-theorem (consulté le 15.07.2012)

Freedom form the tyranny of schemas http://www.julianbrowne.com/article/viewer/freedom-from-the-tyranny-of-schemas (consulté le 15.07.2012)

NoSQL in the enterprise http://www.julianbrowne.com/article/viewer/nosql-in-the-enterprise (consulté le 15.07.2012)

NoSQL in the enterprise http://www.infoq.com/articles/nosql-in-the-enterprise (consulté le 17.07.2012)

NoSQL en quête de performances extrêmes http://scale.af83.com/scale1.fr.html (consulté le 15.07.2012)

NoSQL databases' Adoption in the Enterprise World http://nosql.mypopescu.com/post/15614580003/nosql-databases-adoption-in-the-enteprise-world?30511670 (23.07.2012)

NoSQL Makes Big Inroads in Enterprise Development – New Evans Data Survey Shows http://www.prweb.com/releases/2011/6/prweb8609164.htm (consulté le 25.07.2012)

NoSQL Analysis of the strengths and weaknesses http://java.dzone.com/articles/nosql-analysis-strengths-and (consulté le 17.07.2012)

Putting NoSQL In Its Place – In the Enterprise http://www.jaspersoft.com/blog-entry/putting-nosql-in-its-place-%E2%80%93-in-the-enterprise (consulté le 24.08.2012)

Dénormalisation du modèle de données http://www.thecodingmachine.com/fr/d%C3%A9normalisation-du-mod%C3%A8le-de-donn%C3%A9es (consulté le 26.08.2012)

16 Webinar

The CIOs Guide to NoSQL http://www.dataversity.net/webinar-the-cios-guide-to-nosql-2/

NoSQL for Big Data in the Enterprise
http://www.youtube.com/watch?v=PoMsXvHkHG4

Webinar (Geeknet): Apache Cassandra: Real NoSQL Applications in the Enterprise Today http://www.youtube.com/watch?v=Eho-HCQ5iJE&feature=plcp

17 Vidéographie

Scaling the Web: Databases & NoSQL http://youtu.be/oL-A4JYwqH4

NoSQL & Big Data: Scaling the Enterprise into the New Age
http://www.youtube.com/watch?v=1U73ns1L74E

Julian Browne - NoSQL in the Enterprise - CodeKen 2011
http://www.youtube.com/watch?v=GhI3pHQrc1c

The NoSQL tapes http://nosqltapes.com