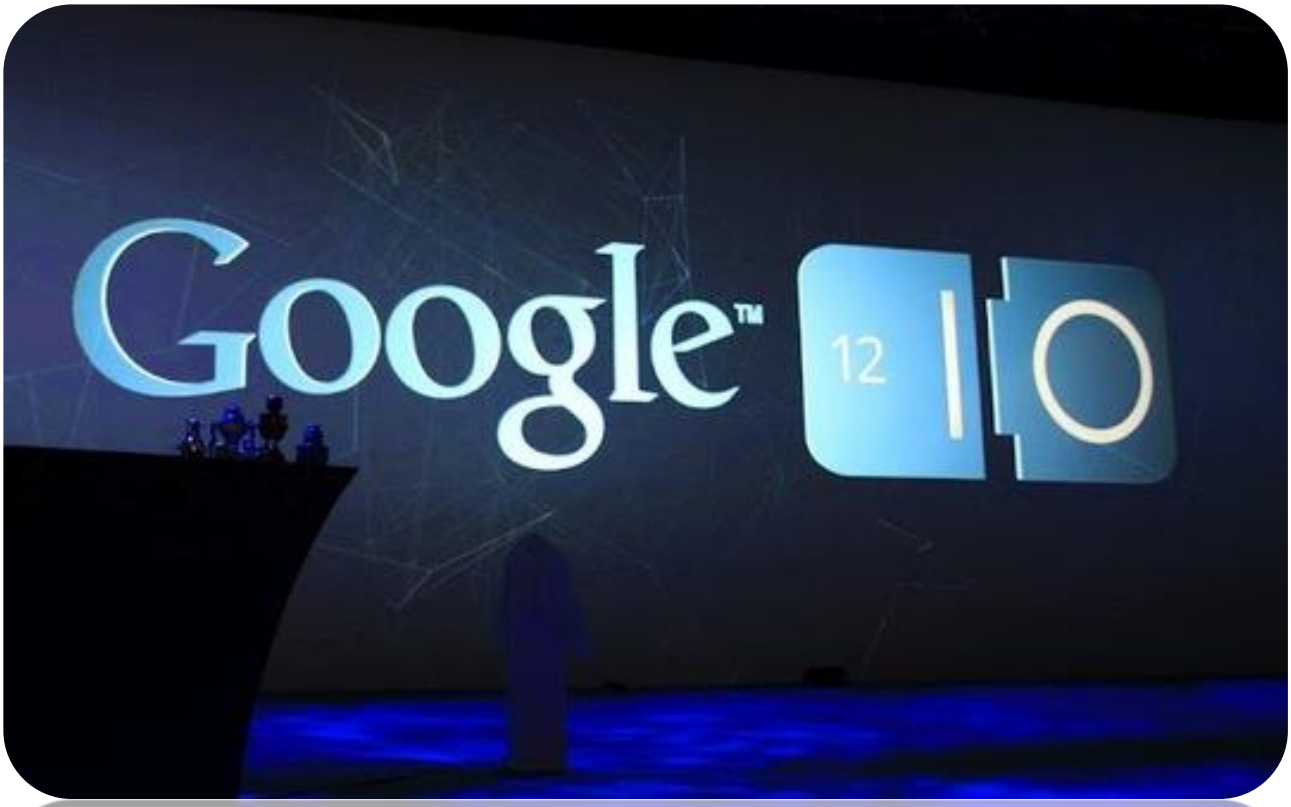
# Résumé Conference Google
# SQL vs NoSQL :
# Battle of the Backends

Alexandre LANZERAY - Julie CLERE - Quentin FIGUERAS
ICOM

13/12/2013



# Sommaire

# Introduction

Cette synthèse résume les différents points et concepts abordés lors de la conférence de Google présenté par Ken Ashcraft et Alfred Fuller nommé « SQL vs NoSQL : Battle of the Backends ». Cette dernière traite de deux types de stockage de données (Cloud SQL et Datastore) et donc par conséquent deux langages (SQL et NoSQL) utilisés par Google App Engine.

Google a créé un Cloud très utilisé pour son ergonomie et son efficacité. En effet, ce Cloud possède un système de récupération automatique des erreurs qui permet, lors de l'apparition de l'une d'elles, à l'utilisateur de continuer son travail sans aucun dérangement, sans même connaitre l'existence de l'erreur. Ce système implique donc peu de maintenance, les mises à jour étant faites automatiquement. De plus, le Google Cloud est accessible partout et existe même en environnement local pour travailler sans connexion internet.

Google App Engine est un outil permettant de construire des applications sur l'infrastructure Cloud de Google en se servant de celle-ci comme une plateforme de service (Platform as a Service - PaaS) permettant à l'utilisateur de se focaliser sur l'application en elle-même.
Google App Engine possède plusieurs types de stockage disponibles : Cloud SQL, Datastore et Cloud Storage.

On va surtout s'intéresser aux deux premiers. Le stockage « Datastore » représente littéralement l'infrastructure de stockage de Google, notamment pour ses applications comme par exemple pour Gmail. Ce stockage est une solution NoSQL entièrement géré et faite pour une utilisation en masse (2 billion d'opérations par mois).

Contrairement à Datastore, Cloud SQL fonctionne en MySQL.

Les deux présentateurs comparent ensuite ces deux modèles de stockage à travers six différents aspects :

- Requêtes
- Transactions
- Cohérence
- Capacité d'adaptation
- Management
- Schémas

# I. Requêtes

Les requêtes sont importantes, en effet l'utilisateur doit disposer d'un puissant langage de requêtes afin d'accéder facilement et rapidement aux données désirées.

Le Cloud SQL fonctionne à travers les langages de manipulation de données SQL standards et les requêtes structurées permettant ainsi à l'utilisateur d'utiliser le CRUD sur les bases de données. Contrairement au Cloud SQL, le NoSQL ne supporte pas toutes les spécificités du langage SQL, seulement un sous-ensemble de celui-ci. Par exemple, NoSQL supporte que depuis peu le « OR » pour Java et Python ou encore les requêtes index. Cependant, ce langage va au-delà des opérateurs de SQL, permettant ainsi d'écrire des requêtes contenant davantage de propriétés et l'agrandissement de la taille de données traitées ne changera en rien le temps de réponse de ces requêtes.

Cependant, le Cloud SQL de Google permet d'effectuer des agrégations (group by) alors que le NoSQL doit passer par une méthode quelque peu dérivée : MapReduce. Ce dernier effectue des calculs parallèles de données volumineuses en les distribuant dans un cluster de machines pour être traitées, le tout en deux fonctions : Map() et Reduce(). La première analyse le problème, le découpe en sous-problème et délègue à d'autres nœuds et ainsi de suite. La deuxième va faire remonter les résultats des nœuds les plus bas au nœud parent. A la fin du processus, le nœud d'origine peut recomposer une réponse. De plus, ce système a une option permettant de traquer les différents processus de la requête et de les afficher.

Enfin, le Cloud SQL permet d'utiliser des jointures pour toute requête compliquée, ce que ne peut supporter le NoSQL de Datastore.

## II. Transactions

Chaque tentative de créer, mettre à jour ou supprimer une entité prend place dans le contexte d'une transaction. Une transaction unique peut inclure n'importe quel nombre de ces opérations. Afin de maintenir la cohérence des données, la transaction permet de s'assurer que l'ensemble des opérations qu'il contient sont appliquées sur la base de données. Si l'une opération échoue, alors aucune d'entre elles n'aient alors appliqués.

Celles-ci sont parfaitement intégrées au langage MySQL de Cloud SQL tandis que pour Datastore effectuer une transaction est possible mais plus limité. En effet, la solution Datastore utilise « Entity Group » : Il faut déterminer quelles sont les entités dont nous avons besoin pour être en mesure de traiter la même transaction, lorsque ces entités sont créées, les placer dans le même groupe d'entités en les déclarants avec un ancêtre commun. Elles seront donc toutes dans le même groupe d'entités et nous serons en mesure de mettre à jour de façon transactionnelle. Cependant, la transaction ne peut s'appliquer qu'à cinq groupes d'entités maximum à la fois.

## III. Cohérence

Selon la théorie des bases de données, les propriétés ACID sont les quatre principaux attributs d'une transaction de données. Il s'agit là d'un des concepts les plus anciens et les plus importants du fonctionnement des bases de données : spécifier quatre buts à atteindre pour toute transaction. La Cohérence signifie que les modifications apportées à la base doivent être valides, en accord avec l'ensemble de la base et de ses contraintes d'intégrité. S'il arrive qu'un changement risque de perturber l'intégrité des données, alors soit le système doit modifier les données dépendantes, soit la transaction doit être interdite.

Datastore utilise Megastore Replication pour que la cohérence se fasse au sein de la base de données. Megastore utilise les groupes d'entités afin d'effectuer toutes les transactions en même temps. Les données sont dupliquées pour chaque groupe d'entités, donc si l'un d'eux est entrain de se mettre à jour, l'utilisateur peut tout de même effectuer des requêtes sur la base de données à partir d'une des répliques. Si l'on souhaite travailler sur un groupe d'entités avec les dernières mises à jour, tandis que d'autres transactions sont toujours entrain de se faire, celui-ci supporte des requêtes le permettant (Get et Ancestor Query).

La solution Cloud SQL fonctionne peu ou prou pareil mais ils utilisent des datacenters afin que la base de données soit toujours accessible.

## IV. Capacité d'adaptation

La scalabilité ou capacité d'adaptation désigne la capacité d'un produit à s'adapter à un changement d'ordre de grandeur de la demande, en particulier sa capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande.

Cloud SQL, bien qu'il permette de maintenir de bonnes performances en cas de grande influence, plus de 30 000 personnes en même temps par exemple, est cependant beaucoup plus limité que le Datastore (plusieurs millions d'utilisateurs). En effet, ce dernier se repose sur Megastore qui se repose sur Bigtable qui lui-même fonctionne avec GFS v2(Google File System, ce dernier fournissant une immense capacité de données. Bigtable, quant à lui, divise les données selon leur poids sur les différentes machines libres. Lors d'une grosse influence soudaine sur une partie des données, Bigtable va dupliquer les données sur deux machines différentes afin que les requêtes n'arrivent plus au même endroit.

## V. Management

Les deux présentateurs nous montrent comment créer, gérer et configurer notre application par l'infrastructure de Google App Engine. Google App Engine gère automatiquement les patchs, les installations et les serveurs. Néanmoins, en utilisant Cloud SQL il faut configurer beaucoup plus de détails qu'avec Datastore. En effet, celui-ci est prêt à l'utilisation dès la première seconde de création de l'application.

## VI. Schémas

Les schémas permettent de structurer les données en définissant leur type et leur lien.

MySQL permet de créer et changer la structure des données de façon simple et intuitive par la commande « Alter ». Cependant, cette commande a le défaut de rendre indisponibles les tables le temps de la modification. Cloud SQL a trouvé une sorte de parade à ce défaut en copiant la table altérée afin que les données soient toujours disponibles durant le changement. Si un changement est fait sur l'ancienne table, un trigger est automatiquement généré afin de mettre à jour la nouvelle table une fois prête.

En ce qui concerne NoSQL, le changement de structure se produit dans l'application et donc est instantané.

## Conclusion

Le tableau suivant résume les différents avantages de chacun des systèmes Cloud SQL et Datastore :

| | Datastore | Cloud SQL |
|---|---|---|
| Queries | ✔ | ✔+ |
| Transactions | ✔ | ✔+ |
| Consistency | ✔ | ✔+ |
| Scalability | ✔+ | ✔ |
| Management | ✔+ | ✔ |
| Schema | ✔+ | ✔ |

On peut voir que chaque solution a ses avantages et ses inconvénients, tout dépend bien entendu de l'usage recherché. NoSQL est à utilisé pour des applications avec de très grandes influences, cependant c'est un langage peut être plus complexe et moins connu, il faut donc un petit temps d'adaptation.

Une solution hybride représenterait donc l'idéal afin de compenser les faiblesses de l'un par les forces de l'autre.