NoSQL

Késako ?

Bruno PINAUD

Support enseignement IPB. Ce support est incomplet. Il faut compléter les trous.

Bibliographie

A First Course in DATABASE SYSTEMS
3rd edition
J. D. Ullman, J. Widom
Pearson, Prentice Hall, 2008

NoSQL Distilled
A Brief Guide to the Emerging World of Polyglot Persistence
P. J. Sadalage, M. Fowler
Addison-Wesley, 2013

Documentation en ligne PostgreSQL
http://www.postgresql.org/docs/

Origine du mot

- **NoSQL** : reconnu maintenant comme *Not Only SQL*.

Mais, que signifie vraiment « Not Only » ??? Pas grand chose... Pas formellement défini

- **NoSQL** →

Historique

Le terme « NoSQL » dans sa signification actuelle vient du nom d'un séminaire (pendant une conférence sur Hadoop) qui a eu lieu en 2009 sur les nouveaux types de bases de données qui n'utilisent pas SQL. Il fallait un nom qui fasse un bon hashtag sur Twitter : court, facile à mémoriser, et non populaire sur Google pour espérer pouvoir remonter dans les premiers résultats.

Les systèmes du type NoSQL sont en fait utilisés depuis longtemps :

Rappels sur les bases de données relationnelles

Les principales opérations (LDD et LMD, SQL–92) :

Un **Système de Gestion de Bases de Données** (SGBD) se charge de la mise en œuvre des transactions.

Régies par le **Modèle relationnel** (Codd, 70) : tous les SGBD fonctionnent à peu près de la même façon (au moins pour le LDD et LMD) : « One Size Fits All ».

Rappels sur les bases de données relationnelles

Propriétés ACID des transactions :

- **A**
- **C**
- **I**
- **D**

Rappels sur les bases de données relationnelles

- **A**

  Toute transaction doit s'exécuter entièrement sans erreur ou pas du tout (ne doit alors laisser aucune trace de son existence).

- **C**

  La base de données doit toujours être dans un état cohérent entre deux transactions (même si une transaction ne se termine pas correctement).

Rappels sur les bases de données relationnelles

- **I**

  Toute transaction doit s'exécuter comme si elle était la seule sur le système. Aucune dépendance possible entre les transactions.

- **D**

  Quand la transaction est terminée, la base de données doit être dans un état stable et durable dans le temps (persistance).

Rappels sur les bases de données relationnelles

Problèmes à éviter lors du déroulement des transactions

- Une transaction lit des données écrites par une autre transaction non validée.

- Une transaction relit des données (donc déjà lues précédemment) mais les données ont été modifiées (par une autre transaction validée entre les deux lectures).

- Une même requête exécutée dans deux transactions ne renvoie pas le même résultat.

Limites des bases de données relationnelles

Relation

Contient un ensemble de *n*-uplets (les lignes) qui contiennent des attributs (les colonnes). Le domaine de chaque attribut est fixé et doit être composé de valeurs atomiques simples, *i.e.* non décomposables (1FN).
Une requête SQL de type *select* retourne toujours une relation.

Comment stocker des structures plus complexes ?

Possible dans la plupart des langages de programmation (notamment à objets) mais pas avec le modèle relationnel afin de respecter la normalisation et la définition d'une relation.
Un changement de modèle et une traduction des données sont donc nécessaires.

Limites des bases de données relationnelles

- Une BD relationnelle est une couche spécifique pour accéder aux données (architecture à trois couches (niveaux, tiers) ; 3-tier en anglais).
- De nos jours, intégration de la base de données dans l'application pour simplifier l'architecture du système et accéder plus rapidement aux données (pas de serveur dédié).
- Les BD relationnelles sont conçues pour fonctionner sur un serveur unique avec des volumes de données modérés (maintenance des index, maintien de la cohérence).
- Contraintes ACID impossibles à respecter sur un cluster (et *a fortiori* sur Internet) pour conserver un fonctionnement correct (haute disponibilité, rapidité d'exécution et cohérence permanente de la base, partition accidentelle du cluster).

Les systèmes NoSQL

- *Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable* (http://nosql-database.org/).
- Pas un Système de Gestion de Bases de Données mais plus un système de stockage de données.
- Pas de modèle pré-défini, pas de contrainte stricte comme en relationnel, tout doit rester simple pour aller vite. L'ajout d'un serveur doit améliorer les performances.
- Maintien de la cohérence simple. Exemple du QUORUM :

Les systèmes NoSQL : CAP

Théorème CAP (proposé par Brewer, 2000 et démontré par Gilbert et Lynch 2002)

- Dans un environnement distribué, il n'est pas possible de respecter simultanément :
  - **C**
  - **A**
  - **P**

- On peut, en revanche, respecter deux contraintes.
- BD relationnelles :

Les systèmes NoSQL

À l'opposé de ACID, les systèmes NoSQL sont parfois présentés comme BASE :

Basically Available, Soft state, Eventually consistent

- **Eventual consistency :**
- **Soft state :**
- **Basically available** :

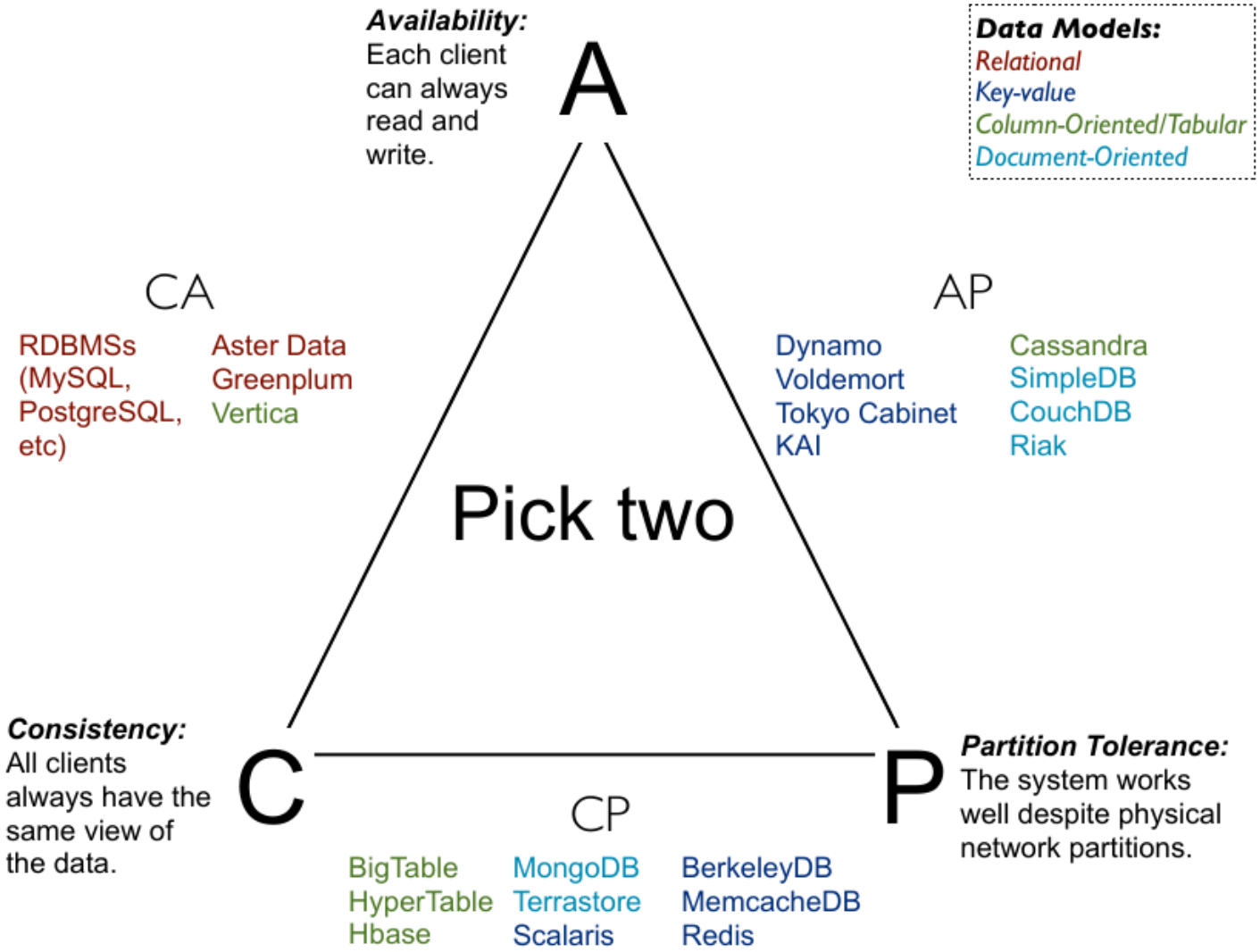
Les systèmes NoSQL : choix

4 grands types de systèmes

- Orienté clé-valeurs
- Orienté documents
- Orienté colonnes
- Orienté graphes

Les systèmes NoSQL : choix

http://blog.nahurst.com/visual-guide-to-nosql-systems



Les systèmes NoSQL : clé-valeurs

- Rapidement : stockage simple et direct d'un tableau associatif (hash-table). Possible en relationnel avec une table à deux attributs dont les accès se font uniquement par la clé primaire.

- Système le plus simple : lecture, écriture et suppression. Ne fait que stocker des données en rapport avec une clé.

- Implique pas de relation entre les données. Le système ne connaît pas les données ou ce qu'elles représentent (sémantique associée).

- Interrogation uniquement par la clé. On récupère une donnée (blob notamment) associée à une clé.

- **Exemples de systèmes** : Riak, Project Voldemort, DynamoDB, Redis, Berkeley DB

- **Exemples d'usages** :

Les systèmes NoSQL : Documents

- Basé sur un système clé-valeur
- Mais la valeur est une structure que le système connaît et peut donc parcourir.
- Pour la plupart des systèmes pas d'opérations croisées entres plusieurs documents.

- **Exemples de systèmes** : MongoDB, CouchDB, RavenDB
- **Exemples d'usages** :

Les systèmes NoSQL : colonnes

- « Opposé » d'une BD relationnelle : en relationnel, le modèle est orienté *n*-uplets dans des tables, le nombre de colonnes est fixé. Avec un modèle orienté colonnes, on ne stocke que les colonnes qui ont des valeurs non nulles regroupées dans des familles de colonnes.

- Les colonnes sont stockées sous une forme clé-valeur. On peut ajouter une colonne dans un enregistrement aussi facilement que l'ajout d'un n-uplet en relationnel.

http://blog.xebia.fr/2010/05/04/nosql-europe-bases-de-donnees-orientees-colonnes-et-cassandra/

Les systèmes NoSQL : colonnes

- Coût de stockage d'une valeur nulle :
- Stockage de plusieurs millions de colonnes

- **Exemples de systèmes** : Cassandra, HBase, Hypertable
- **Exemples d'usages** :

Les systèmes NoSQL : Graphes

- On ne stocke plus un simple ensemble de données mais des relations.
- Stockage d'entités et de relations entre les entités. On peut ajouter des propriétés aux relations et aux entités.
- Cohérence forte des données. Pas d'arête pendante.
- Difficile de monter en charge (ajout de serveurs). Le stockage d'un graphe sur plusieurs serveurs est un problème difficile (problème de performance même pour un simple calcul de chemin si le graphe est réparti sur plusieurs serveurs).

- **Exemples de systèmes** : Neo4J, Infinite Graph, FlockDB (Twitter)
- **Exemples d'usages** :

Polyglot Persistence

- Nouveau concept lié au mouvement NoSQL et aux nouveaux choix technologiques.
- Les SGBDR sont maintenant une option parmi un panel de plus en plus large.
- Il faut apprendre à utiliser la base de données la plus adaptée aux données à gérer quitte à en utiliser plusieurs.

Conclusion

- Technologies jeunes et *a priori* plus pérennes que les bases de données objets.
- Les systèmes NoSQL ne remplaceront jamais les bases de données relationnelles. Il faut les voir comme une alternative après un quasi monopole du relationnel de plusieurs décennies (*One size fits all*).
- Les systèmes de gestion de données relationnelles sont maintenant une option parmi d'autres. Leur généricité leur permet d'être utilisés dans de nombreux cas mais les rends peu performants et complexes sur certains types de données (masse de données, attributs trop nombreux, …).
- Il faut savoir utiliser la solution la plus adaptée et pourquoi pas en utiliser plusieurs.
- Pas de schéma figé comme en relationnel, moins de contraintes. On déporte sur l'application la gestion du modèle de données et sa connaissance.
- Changement drastique des méthodes de développements notamment par l'absence de cohérence immédiate pour la tolérance au partitionnement et les évolutions du schéma.