université PARIS-SACLAY

Master 1 Informatique & Miage

Bases de Données Approfondies – 2e partie

Théorie et pratique de NoSQL

Philippe Declercq 2019/2020

Préliminaire

- Ce cours est dispensé dans de le cadre de l'enseignement de Master 1 Informatique & Miage « Bases de Données Approfondies ». Le volume horaire est de 18 heures, réparties en 3 séances de Cours et 3 séances de TD.
- Ce cours est une suite logique et complémentaire à la première partie (dispensée par Serenella Cerrito) qui a abordé les notions suivantes :
  - Rappel des bases de données relationnelles, objet, relationnelle-objet,
  - Manipulations XML (Xpath, Xquery, …),
  - Bases XML.
- Cours et TD associent des parties théoriques, des mises en application sur des cas concrets et des manipulations d'outils.

Références

[1]: Bases de données documentaires et distribuées (Ph. Rigaux)
[2]: Introduction aux systèmes NoSQL (B. Espinasse)
[3]: NoSQL, une nouvelle approche du stockage et de la manipulation des données (SMILE)
[4]: Blog Xebia : http://blog.xebia.fr/category/nosql/
[5]: MongoDB Documentation
[6]: NoSQL systems : sharding, replication and consistency
[7]: RDBMS to MongoDB Migration Guide
[8]: Introduction to MapReduce, MSc Software and Systems
[9]: Introduction aux bases de données NoSQL (K. Tannir)
[10]: Les bases de données NoSQL et le Big Data (R. Bruchez)

Plan du cours et des TD

1	**Introduction au NoSQL**
2	**Stockage et manipulation de données avec NoSQL**
3	**Réplication des données**
4	**Partitionnement des données**
5	**Recherches et calculs en NoSQL**
6	**Et l'avenir ?**
TD1	**Installation et manipulations simples NoSQL**
TD2	**Réplication et partitionnement**
TD3	**Recherche d'information – Moteurs de recherche**

Université d'Évry-Val d'Essonne

université PARIS-SACLAY

Théorie et pratique de NoSQL

Chapitre 1

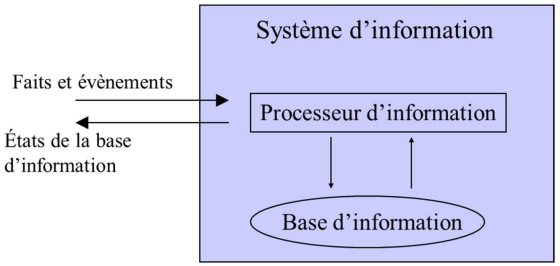
Introduction au NoSQL

Philippe Declercq

2019-2020

Introduction

- Le besoin d'organiser les données, potentiellement de grandes quantités de données, afin d'en optimiser la conservation et la restitution, a toujours été au cœur de l'informatique.
- Exemple de représentation d'un Système d'Information :



Introduction

- Le modèle relationnel, apparu dans les années 1980, est régi par des règles précises, énoncées par Codd (IBM) en 1985 :
  - séparation logique et physique,
  - langage déclaratif,
  - structuration forte des données,
  - représentation tabulaire,
  - contraintes définies au niveau du moteur,
  - cohérence transactionnelle forte, etc.
- Les bases de données relationnelles font la preuve de leur efficacité depuis longtemps !

Introduction

- Ces bases relationnelles sont bien adaptées au stockage et à la manipulation d'informations bien structurées, décomposables en unités simples et représentables sous forme de tableaux.
- Cependant … depuis quelques années, ce modèle est remis en cause … pour des raisons que nous allons (tenter) d'expliquer à partir de deux constats.

Introduction - Constats

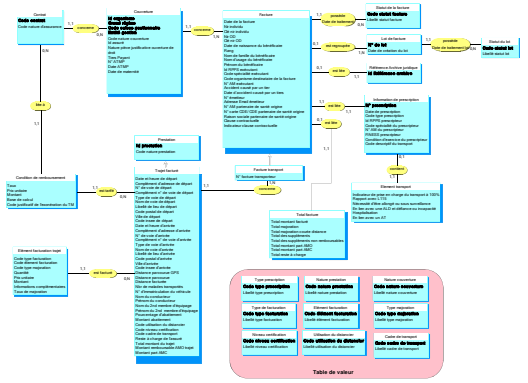
- Constat n°1 : le modèle relationnel présente certaines limites présentées en première partie du cours :

1. On ne peut pas imbriquer les informations.
   - On multiplie le nombre de tables pour représenter conceptuellement le même objet.

2. La structure du schéma est très rigide.
   - une base a un nombre fixé de tables, une table a un nombre fixe d'attributs, …
   - Le modèle doit être défini tôt dans le processus de développement, il est souvent difficile et couteux de le faire évoluer.

Introduction - Constats

- Exemple : représentation de l'objet « Facture Transport » transmis par les entreprises de Transport de malades à l'Assurance Maladie :



```
create table  TT_FACTURE (…) ;
create table  TT_CERTIFICAT (…) ;
create table  TT_LOT (…) ;
create table  TT_COUVERTURE (…) ;
create table  TT_CONTRAT (…) ;
create table  TT_REMBOURSEMENT (…) ;
create table  TT_PRESTATION (…) ;
create table  TT_TRAJET (…) ;
create table  TT_ELEMENT_TRAJET (…) ;
create table  TT_PRESCRIPTION (…) ;
create table  TT_TOTAL (…) ;
```

- 11 tables pour stocker un objet facture !

Introduction - Constats

- Constat n°2 : croissance exponentielle des données générées par le web.

- Ces masses de données apportent des opportunités d'analyses plus larges et plus fines ainsi que de nouveaux usages de l'information.

- Les données sont devenues le principal actif de certaines entreprises !

- Le phénomène « Big data » !

- Les systèmes de gestion des BD relationnelles n'ont pas été conçus pour gérer de tels volumes de données.

Introduction - Constats

- Les premiers acteurs à avoir été confrontés à ces deux constats et avoir buté sur les limites des modèles relationnels furent les fournisseurs de services en ligne les plus populaires, tels que Yahoo, Google, ou plus récemment les acteurs du web social comme Facebook, Twitter ou LinkedIn.

- Ces grands acteurs du web ont alors conçu de nouveaux systèmes leur permettant de s'affranchir de ces limites.

- C'est la naissance du phénomène NoSQL !

- NoSQL : No SQL ?

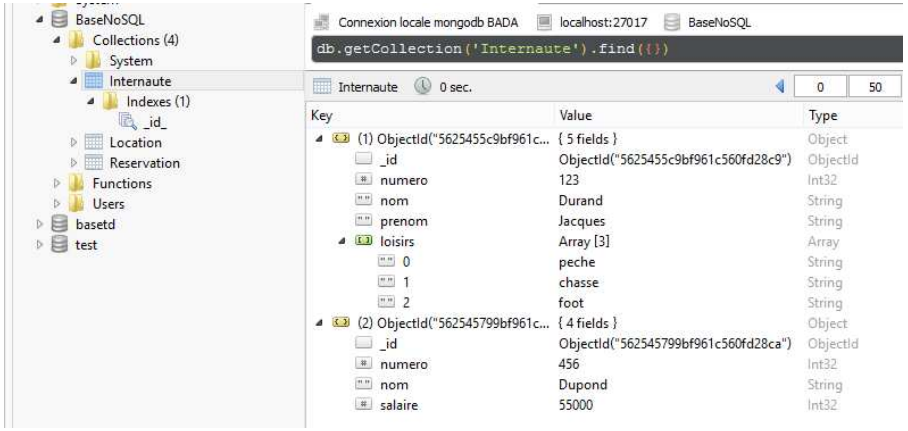
- Faut il jeter le modèle relationnel ?

Introduction – La réponse NoSQL

- Réponse au constat n°1 : les modèles non-structurés ou semi-structurés, plus adaptés à de nombreux cas de collecte et de manipulation des données…
- Pourquoi ?
- XML est un modèle semi-structuré, étudié en première partie du cours.
- Les bases NoSQL manipulent les données à l'aide de modèles non ou semi-structurées : par exemple XML, mais ça n'est pas le seul format de modèles semi-structurés qui existe…

Introduction – La réponse NoSQL

- Exemple : insertion de documents dans la collection « Internaute » de la base Mongodb :

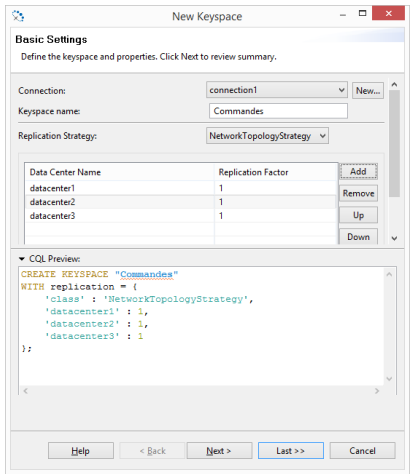


Introduction – La réponse NoSQL

- Réponse au constat n°2 : gérer de grands volumes de données nécessite de mettre en place des systèmes distribués ….
- Les capacités de stockage sont démultipliées par la possibilité de répartir les données sur plusieurs nœuds.
- Les systèmes relationnels ne sont pas adaptés aux environnements distribués.
- Pourquoi ?

Introduction – La réponse NoSQL

- Les solutions NoSQL implémentent naturellement des mécanismes qui permettent une distribution des données.
- Exemple : création d'un « keyspace » sous Cassandra :



Introduction – La réponse NoSQL

- Les bases NoSQL ne remplacent pas les BD relationnelles mais en sont une alternative, un complément apportant des solutions plus intéressantes dans certains contextes.

Système	NoSQL	Relationnel
Capacité de stockage	Très élevée (> 1 To)	Modérée (< 1 To)
Architecture	Distribuée	Centralisée
Modèle de données	Destructuré	Relationnel (tabulaire)
Réponse à la charge	Lecture et écriture	Lecture en majorité
Scalabilité	Horizontale (nombre)	Verticale (puissance)
Moteur de requêtes	Propre au système	SQL
Principales caractéristiques	BASE	ACID
Ancienneté de la technologie	Récente	Eprouvée

Introduction – La réponse NoSQL

- Classement des utilisations des bases de données :

346 systems in ranking, October 2018

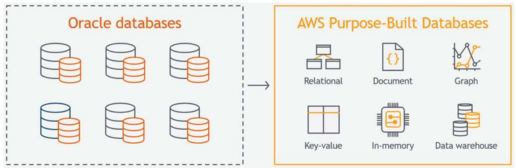
Rank Oct 2018	Rank Sep 2018	Rank Oct 2017	DBMS	Database Model	Score Oct 2018	Score Sep 2018	Score Oct 2017
1.	1.	1.	Oracle	Relational DBMS	1319.27	+10.15	-29.54
2.	2.	2.	MySQL	Relational DBMS	1178.12	-2.36	-120.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1058.33	+7.05	-151.99
4.	4.	4.	PostgreSQL	Relational DBMS	419.39	+12.97	+46.12
5.	5.	5.	MongoDB	Document store	363.19	+4.39	+33.79
6.	6.	6.	DB2	Relational DBMS	179.69	-1.38	-14.90
7.	↑ 8.	↑ 9.	Redis	Key-value store	145.29	+4.35	+23.24
8.	↓ 7.	↑ 10.	Elasticsearch	Search engine	142.33	-0.28	+22.09
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	136.80	+3.41	+7.35
10.	10.	↓ 8.	Cassandra	Wide column store	123.39	+3.83	-1.40
11.	11.	11.	SQLite	Relational DBMS	116.74	+1.29	+4.76
12.	12.	12.	Teradata	Relational DBMS	78.63	+1.25	-1.45
13.	13.	↑ 16.	Splunk	Search engine	76.90	+2.87	+12.54
14.	14.	↑ 18.	MariaDB	Relational DBMS	73.13	+2.49	+16.73
15.	15.	↓ 13.	Solr	Search engine	61.31	+1.11	-9.82
16.	16.	↑ 19.	Hive	Relational DBMS	61.10	+1.47	+9.66
17.	17.	↓ 15.	HBase	Wide column store	60.67	+2.21	-3.72
18.	18.	↓ 14.	SAP Adaptive Server	Relational DBMS	58.57	+0.52	-8.67
19.	19.	↓ 17.	FileMaker	Relational DBMS	56.04	+0.74	-5.02
20.	20.	↑ 22.	Amazon DynamoDB	Multi-model	54.47	+1.12	+16.87
21.	21.	↓ 20.	SAP HANA	Relational DBMS	54.37	+1.64	+4.28
22.	22.	↓ 21.	Neo4j	Graph DBMS	42.65	+2.56	+4.71
23.	23.	23.	Couchbase	Document store	35.92	+1.37	+2.55
24.	24.	↑ 25.	Memcached	Key-value store	30.56	-0.98	+2.92
25.	25.	↑ 27.	Microsoft Azure SQL Database	Relational DBMS	26.27	+1.02	+4.42
26.	26.	↓ 24.	Informix	Relational DBMS	26.24	+1.33	-1.66

http://db-engines.com/en/ranking

Introduction – La réponse NoSQL

- La diversité des solutions techniques de stockage de données n'a jamais été aussi importante !
- Les entreprises utilisent aujourd'hui plusieurs solutions de stockage, selon leurs besoins.

- Exemple : Amazon



"The migration gave each internal team the freedom to choose the purpose-built AWS database service that best fit their needs" (Jeff Barr, Chief Evangelist for AWS)

https://aws.amazon.com/fr/blogs/aws/migration-complete-amazons-consumer-business-just-turned-off-its-final-oracle-database/

Introduction - Résumé

- « Entities are king » vs. « Queries are king » ?
- Les bases NoSQL adoptent une représentation de données non relationnelle.
- Les bases NoSQL apportent une plus grande performance dans le contexte des applications Web avec des volumétries de données très importantes.
- Les bases NoSQL utilisent une très forte distribution de ces données et des traitements associés sur de nombreux serveurs.

Introduction - Résumé

- Une proposition de définition de NoSQL :

« Tout système de gestion de données
sacrifiant des fonctionnalités du relationnel
pour faciliter la scalabilité par distribution »

- Ou celle de http://nosql-database.org :

“ Next Generation Databases
mostly addressing some of the points : being non-relational, distributed, open-source and horizontally scalable.”

Introduction - Résumé

- La suite de ce cours est dédié à l’étude des systèmes NoSQL et leurs capacités à répondre aux deux constats établis précédemment :
  - Le stockage et manipulation de données semi-structurées
    → chapitre 2 du cours
  - La mise en place de solutions distribuées pour gérer de grands volumes de données
    → chapitres 3 & 4 du cours
  - Les mécanismes de recherche et de calcul sur de grands volumes de données, en environnement distribué
    → chapitre 5 du cours

Université d'Évry-Val d'Essonne

université PARIS-SACLAY

Théorie et pratique de NoSQL

Chapitre 2

Stockage et manipulation de données avec NoSQL

Philippe Declercq 2019-2020

Stockage et manipulation de données - JSON

- En introduction de ce cours, nous avons vu que les bases NoSQL manipulent des données déstructurées ou semi-structurées, plus adaptées à de nombreux cas de collecte et de manipulation des données.
- Quels sont les formats de manipulation de données « semi-structurées » ?
- En fait, il en existe une multitude, comme il existe une multitude de bases NoSQL !
  - XML, Xquery, XML-Schema : très répandu !
  - JSON : la valeur qui monte !
  - Autres : YAML, … : moins répandus

- JSON : JavaScript Object Notation. http://www.json.org
- Format de données textuelles dérivé de la notation des objets du langage JavaScript. Permet de représenter de l'information structurée comme le permet XML.
- Dérivé de la représentation littérale d'un objet en Javascript.
- JSON est plus simple que XML, et il est très facile à associer à un langage de programmation.
- JSON est massivement utilisé dans les applications web (AJAX), les web-services (REST), et … les bases de données NoSQL !

- JSON se base sur deux structures :
  - Une collection de couples nom/valeur, appelée **objet**
  - Une liste de valeurs ordonnées, appelée **tableau**.
- Exemples :

```
"director": {
    "last_name": "Fincher",
    "first_name": "David",
    "birth_date": 1962
}

"actors": ["Eisenberg", "Mara", "Garfield", "Timberlake"]
```

objet

couples nom/valeur

tableau

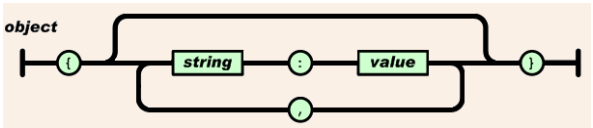
- XML est plus bavard que JSON !
- En JSON :

```
{"nom": "BADA",
"enseignant": "P. Declercq",
"heures": 18,
"seances" : [
  {"sujet" : "Introduction", "date" : "10/01"},
  {"sujet" : "Manip. NoSQL", "date" : "15/01"}
  ]
}
```

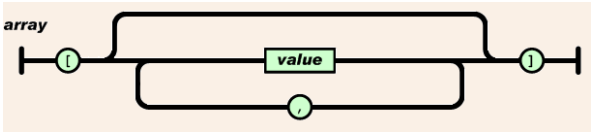
- L'équivalent en XML :

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <nom>BADA</nom>
  <enseignant>P. Declercq</enseignant>
  <heures>18</heures>
  <seances>
    <sujet>Introduction</sujet>
    <date>10/01</date>
  </seances>
  <seances>
    <sujet>Manip. NoSQL</sujet>
    <date>15/01</date>
  </seances>
</root>
```

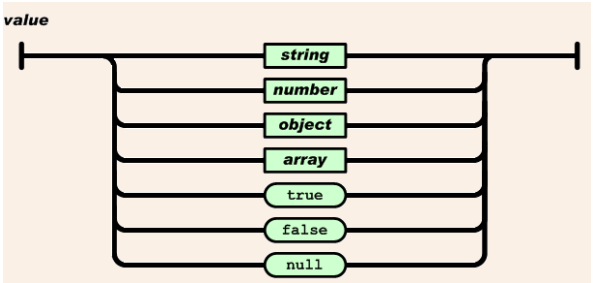

- L'objet est un ensemble de couples *nom/valeur* non ordonné :



- Un tableau est une collection de valeurs ordonnées :



- Une valeur peut être une chaîne de caractères, un nombre, true ou false ou null, un objet, un tableau :



- Les structures peuvent bien sûr être imbriquées !

```
"MonFilmCulte" {
        "title": "The Social network",
        "summary": "On a fall night in 2003, Harvard undergrad and \n
        programming genius Mark Zuckerberg sits down at his \n
        computer and heatedly begins working on a new idea. (...)",
        "year": 2010,
        "director": {       "last_name": "Fincher",
                            "first_name": "David"
        },
        "actors": [
                {"first_name": "Jesse", "last_name": "Eisenberg"},
                {"first_name": "Rooney", "last_name": "Mara"}
        ]
 }
```


- JSON est un concurrent plus que sérieux à XML !
  - JSON est plus léger et intuitif que XML,
  - JSON est facile à parser pour n'importe quel langage de programmation.

- Extrait de http://www.json.org :

Most of the excitement around XML is around a new role as an interchangeable data serialization format. XML provides two enormous advantages as a data representation language : it is text-based, and it is position-independent.

These together encouraged a higher level of application-independence than other data-interchange formats. The fact that XML was already a W3C standard meant that there wasn't much left to fight about (or so it seemed).

Unfortunately, XML is not well suited to data-interchange, much as a wrench is not well-suited to driving nails. It carries a lot of baggage, and it doesn't match the data model of most programming languages. When most programmers saw XML for the first time, they were shocked at how ugly and inefficient it was. It turns out that that first reaction was the correct one. There is another text notation that has all of the advantages of XML, but is much better suited to data-interchange. That notation is JavaScript Object Notation (JSON).

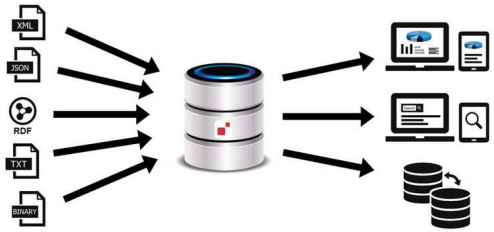
The most informed opinions on XML suggest that XML has big problems as a data-interchange format, but the disadvantages are compensated for by the benefits of interoperability and openness.

JSON promises the same benefits of interoperability and openness, but without the disadvantages.

- JSON est plus jeune que XML. Un certain nombre de caractéristiques d'XML ne disposent pas d'équivalent bien établi chez JSON :
  - Pas d'équivalent XSD / DTD pour formater ou contrôler un flux
    - Une initiative JSON-schéma en version draft…
  - Pas d'équivalent XQuery / XPath pour la recherche d'informations
    - JSONiq, the « The JSON Query Language » …
  - Pas d'équivalent pour les transformations XSLT
    - Une initiative isolée JSONT …

Stockage et manipulation de données - JSON

- XML et JSON sont les deux formats documentaires les plus répandus dans les solutions NoSQL du marché.

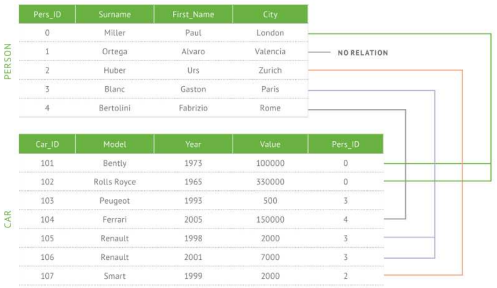


- Solutions du marché :
  - Solutions NoSQL XML : baseX, Berkeley DB XML, eXist, …
  - Solutions NoSQL JSON : mongoDB, couchDB, …
  - Solutions multiples : MarkLogic Server, …

Stockage et manipulation de données – Oubliez vos références !

- Que ce soit en XML ou en JSON, la conception d'une base NoSQL diffère de la conception d'une base relationnelle.

A un modèle de tables unies par des jointures…

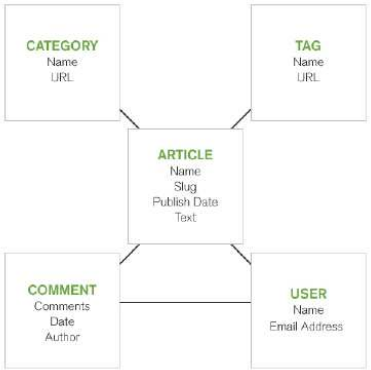


…va succéder un modèle imbriqué où l'ensemble des informations sera accessible sans opération de jointure …

```
{
first_name: "Paul",
surname: "Miller",
city: "London",
location: [45.123,47.232],
cars: [
   { model: "Bentley",
     year: 1973,
     value: 100000, ….},
   { model: "Rolls Royce",
     year: 1965,
     value: 330000, ….},
]
}
```

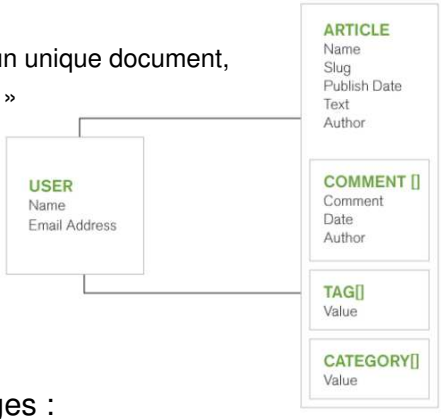

Stockage et manipulation de données – Oubliez vos références !

- Et dans cet exemple ?



Stockage et manipulation de données – Oubliez vos références !

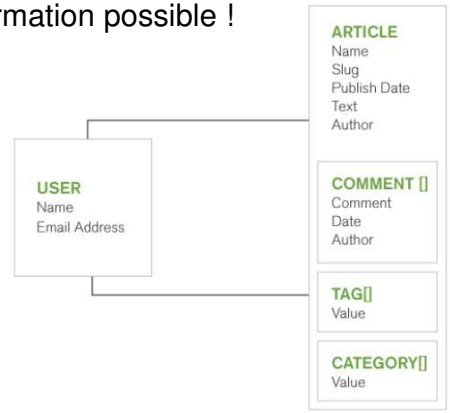
- En NoSQL :
  - Agrégation dans un unique document,
  - « Pre-joining data »



- Quelques avantages :
  - Accès aux informations en un seul appel et une seule lecture sur un serveur,
  - L'information ainsi regroupée simplifie la répartition des données et la scalabilité dans le cadre d'un système distribué.

Stockage et manipulation de données – Oubliez vos références !

- Attention : redondance d'information possible !
  - le nom de l'auteur !



- Doit on éviter à tout prix la redondance d'informations ? Quels inconvénients cela engendre t il ?

Stockage et manipulation de données – Oubliez vos références !

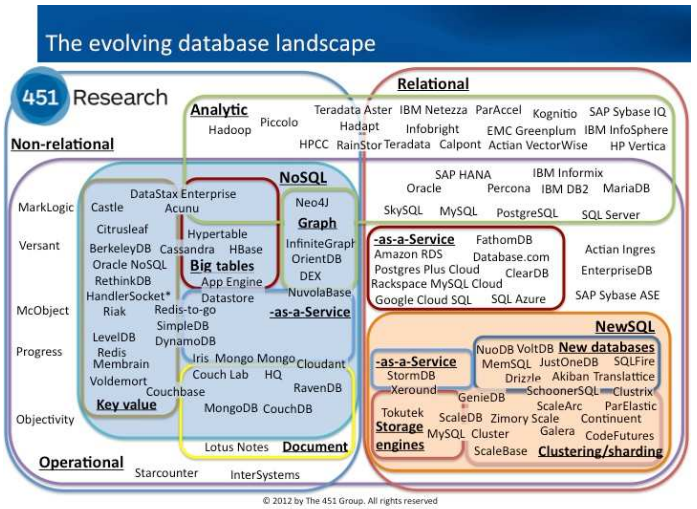
- La conception d'une base NoSQL peut paraitre à première vue plus complexe qu'une base relationnelle.
- L'étude des usages est prépondérante.
- Les éléments fondamentaux à prendre en compte pour concevoir une base NoSQL :
  - Le ratio des accès en lecture et écriture sur la base,
  - Le type de requêtes et mises à jour sur la base,
  - Le cycle de vie des données et le taux de croissance du volume des données.
- Pour une base relationnelle soumise à de fortes sollicitations, on devrait se poser les mêmes questions…

Stockage et manipulation de données – Oubliez vos références !

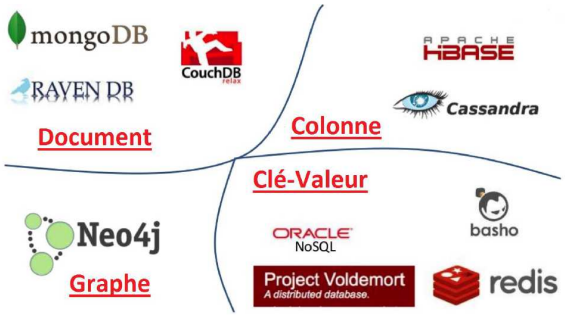
- Contrairement aux bases relationnelles, il n'existe pas (encore) de langage standard pour l'interrogation des bases NoSQL.
- Exemples :
  - Cassandra : CQL Cassandra Query Langage, basé sur SQL,
  - Couchdb : JSON,
  - eXist, BaseX : Xquery,
  - Mongodb : langage spécifique,
  - …

Stockage et manipulation de données – Solutions NoSQL

- Il existe un très grand nombre de solutions NoSQL :

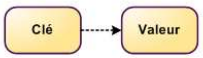


- Les solutions NoSQL sont généralement classées en 4 catégories selon le schéma ou la structure de données qu'ils manipulent :
  1. Bases clé/valeur,
  2. Bases orientées colonnes,
  3. Bases orientées documents,
  4. Bases orientées graphes.



- Le choix d'une catégorie ou d'une autre, d'une solution ou d'une autre, doit se faire en analysant :
  - La structure des données à conserver,
  - L'usage des données (modes de lectures, mises à jour, …),
  - Le besoin de performance,
  - Le volume de données à traiter,
  - …

- **Les bases « clé/valeur » :**
  - Il s'agit de la catégorie de base de données la plus basique. Dans ce modèle chaque objet est identifié par une clé unique qui constitue la seule manière de le requêter.
  - La structure de l'objet est libre et le plus souvent laissé au choix du développeur de l'application (XML, JSON, ...), la base ne gérant généralement que des chaînes d'octets.
  - Le système de stockage ne connait pas la structure de l'information qu'il manipule.



BDD Clé-Valeur

- Exemples d'implémentation : Voldemort, **Redis**, Riak

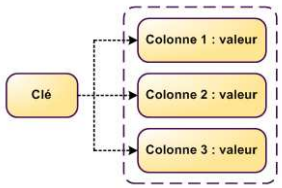
- Intérêts :
  - L'approche volontairement très limitée de ces systèmes sur le plan fonctionnel est radicale et leur permet d'afficher des performances exceptionnellement élevées en lecture et en écriture.
  - Scalabilité horizontale considérable.
  - Pas de maintenance en cas d'évolution de nouveaux types d'informations.

- Limites :
  - Peu de possibilités de requêtage : uniquement sur les clés.
  - Le système n'ayant pas d'indice sur la structure de l'information qu'il stocke, tous les traitements (extraction du contenu, application de filtres…) doivent être effectués par le client.

- Usages :
  - dépôt de données avec besoins de requêtage très simples,
  - système de stockage de cache ou de sessions distribuées,
  - profils, préférences d'utilisateurs,
  - données de panier d'achat,
  - données de capteur,
  - logs de données.
- Exemples :
  - Sur un site de réseau social, à partir d'un utilisateur (la clé), je veux obtenir une liste de ses amis (la valeur).
  - Dans un catalogue de livres, le numéro ISBN (la clé) donne accès à tous les détails sur le livre (la valeur).
  - Dans un journal d'activités, la date d'un événement (la clé) indexe les détails de ce qui s'est passé à ce moment (la valeur).

- Exemple de base NoSQL Clé-Valeur : REDIS
- "Open sources, in-memory data structure store, used as database, cache and message broker".
- REDIS est une solution qui a pour principal intérêt la performance. Dans cet esprit, les données sont par défaut conservées en RAM. Il est néanmoins possible de les persister.
- Implémentation très simple de systèmes de caches ou de sessions.

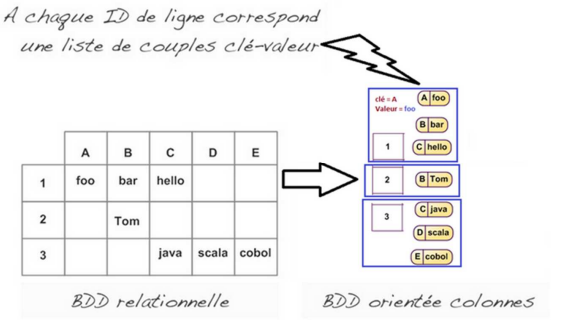
- **Les bases NoSQL orientées colonnes :**
  - La représentation orientée colonnes s'oppose à la représentation des tables dans les bases de données relationnelles. En effet, les SGBDR manipulent les colonnes d'une ligne d'une manière statique. Les bases de données orientées colonnes ont une vision plus flexible permettant d'avoir des colonnes différentes pour chaque ligne, de multiplier de manière conséquente le nombre de colonnes par ligne et d'optimiser le stockage des données associées.



BDD Orientée colonnes

- Exemples d'implémentation : Hbase, **Cassandra**, SimpleDB

- Optimisation du stockage par rapport à une solution relationnelle :



- Pas de place perdue lorsque des champs n'existent pas dans certaines lignes.
- L'accès aux données à partir de la clé sera beaucoup plus rapide, les données liées à une clé étant regroupées en terme de stockage.

- Exemple : gestion de données de Contacts :

Stockage en ligne :

ID	NOM	PRENOM	RUE	CODE POSTAL	VILLE
1	MAIGA	NULL	12 Rue Sosso	31000	TOUOUSE
2	LARBO	Jean	NULL	75012	PARIS
3	NULL	Laurent	NULL	78800	HOUILLES
4	CISSE	NULL	45 Rue Trole	75016	PARIS

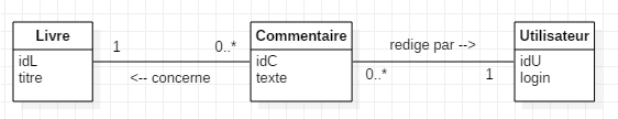
Stockage en colonnes :

1	NOM: MAIGA	RUE: 12 Rue Sosso	CODE POSTAL: 31000	VILLE: TOULOUSE
2	NOM: LARBO	PRENOM: Jean	CODE POSTAL: 75012	VILLE: PARIS
3	PRENOM: Laurent	CODE POSTAL: 78800	VILLE: HOUILLS	
4	NOM:CISSE	RUE: Rue Troie	CODE POSTAL: 75016	VILLE: PARIS

- Quelle est la solution la plus optimisée pour :
  - Rechercher tous les codes postaux des contacts ?
  - Rechercher toutes les propriétés du contact d'id 3 ?
  - Ajouter un nouveau contact ?

- Intérêts :
  - Conçues pour accueillir un grand nombre de colonnes (jusqu'à plusieurs millions) pour chaque ligne. Stockage optimisé de relations « one to many ».
  - Performances et scalabilité
- Limites :
  - Système de requêtes minimaliste
- Usages :
  - Traitements d'analyse de données et traitements massifs
  - Listes d'articles pour chaque utilisateur
  - Liste des actions effectuées par un utilisateur
  - Chronologie d'évènement maintenue et accédée en temps réel

- Exemple de base NoSQL orientée Colonne : CASSANDRA
- Soit le modèle de classes suivant :



- Comment stocker les informations de ces 3 classes de manière à optimiser leur lecture malgré un très grand volume de données ?
- Démarche en trois étapes :

- Les 2 cas d'usage avec exigences de temps de réponse les plus élevées sont :
  - Cas 1 : je veux accéder aux commentaires d'un livre donné (recherche à partir d'un titre donné)
  - Cas 2 : je veux accéder aux commentaires d'un utilisateur donné (recherche à partir d'un login donné)

- Le titre du livre et le login de l'utilisateur seront stockés dans la table Commentaire. On évite ainsi les jointures !

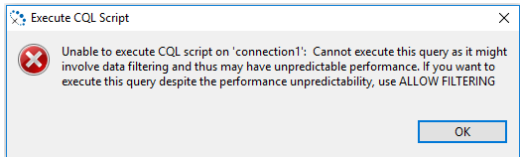
```
CREATE TABLE biblio.commentaire (
idCommentaire INT, texte VARCHAR, titreLivre VARCHAR,
loginUtilisateur VARCHAR,
PRIMARY KEY ( idCommentaire )
) ;
```

- Evidemment, cela crée de la duplication de données. Quelles conséquences ?

- Lorsque l'on exécute la commande :

```
Select * from biblio.commentaire where titreLivre =
'Java en 1 heure' ;
```

- On obtient :



- Pour supporter l'augmentation de la volumétrie des données stockées, on met en place un partitionnement des données de la table Commentaire.

- On veut optimiser le stockage des commentaires qui ont le même titre et le stockage des commentaires qui ont le même login.

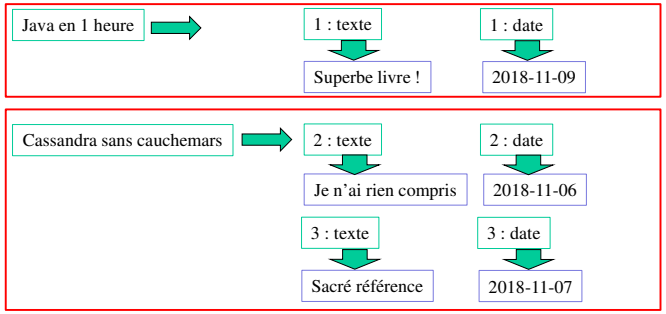
- Cela donne :
  - Pour le cas d'usage 1 :

```
CREATE TABLE biblio.CommentaireLivre (
idCommentaire INT, texte VARCHAR, titreLivre VARCHAR,
PRIMARY KEY ( (titreLivre), idCommentaire)) ;
```

  - Pour le cas d'usage 2 :

```
CREATE TABLE biblio.CommentaireLogin (
idCommentaire INT, texte VARCHAR, loginUtilisateur
VARCHAR,
PRIMARY KEY ( (loginUtilisateur), idCommentaire)) ;
```

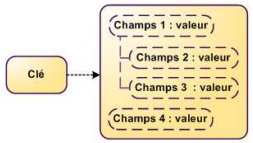
- Comment Cassandra a optimisé le stockage des données ?
- Insertion de 3 commentaires :



- Et cette fois, il est possible de rechercher les commentaires à partir du titre d'un livre :

```
Select * from biblio.CommentaireLivre where titreLivre
= 'Java en 1 heure' ;
```


- **Les bases NoSQL orientées documents :**
  - Constituées de collections de documents. Un document est composé de champs et des valeurs associées, ces dernières pouvant être requêtées.
  - Basées sur le modèle « clé-valeur » mais la valeur est un document en format semi-structuré hiérarchique de type JSON ou XML
  - Les documents n'ont pas de schéma, mais une structure arborescente : ils contiennent une liste de champs, un champ a une valeur qui peut être une liste de champs, ...



BDD Orientée document

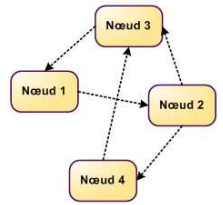
- Exemples d'implémentation : CouchDB, RavenDB, **MongoDB**

- Intérêts :
  - Capacité à effectuer des requêtes sur le contenu des objets
  - Modèle de données simple mais puissant
  - Scalabilité
  - Pas de maintenance de la BD requise pour ajouter/supprimer des «colonnes»
  - Permet de représenter les relations one-to-one et one-to-many. Ainsi un document pourra être sauvegardé et chargé sans aucun traitement de jointure.
- Limites :
  - Inadaptée pour les données interconnectées
  - Lent pour les grandes requêtes

- Usages :
  - Enregistrement d'événements
  - Systèmes de gestion de contenu
  - Web analytique ou analytique temps-réel
  - Catalogue de produits
  - Systèmes d'exploitation
  - Stockage de volumes très importants de données pour lesquelles la modélisation relationnelle aurait entraînée une limitation des possibilités de partitionnement et de réplication

- Exemple de solution NoSQL orientée documents : Mongodb
- Mongodb permet de stocker des données structurées, mais la structure n'a pas à être définie a priori.
- Le modèle des données structurées peut alors facilement évoluer, sans aucune modification du système de stockage.
- Exemple :
  - Stockage de données clients : nom, ville, téléphone.
  - Que se passe t il si un jour je veux gérer plusieurs n° de téléphone pour certains clients ?
  - Que devrais je faire avec un système relationnel ?

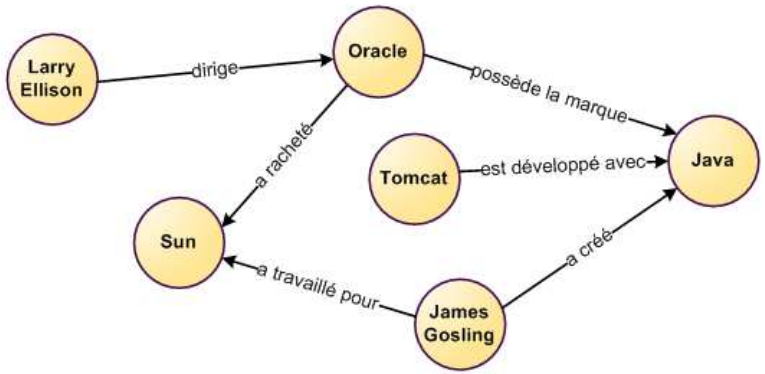
- Les bases NoSQL orientées graphes :
  - Permettent la modélisation, le stockage et la manipulation de données complexes et fortement connectées,
  - Performances en lecture et parcours des données connectées,
  - Utilisent un moteur de stockage pour les objets (similaire à une base documentaire, chaque entité de cette base étant nommée nœud), et un mécanisme de description d'arcs (relations entre les objets), arcs orientés et avec propriétés (nom, date, ...).



BDD Orientée graphe

- Exemples d'implémentation : Neo4J, OrientDB

- Exemple :



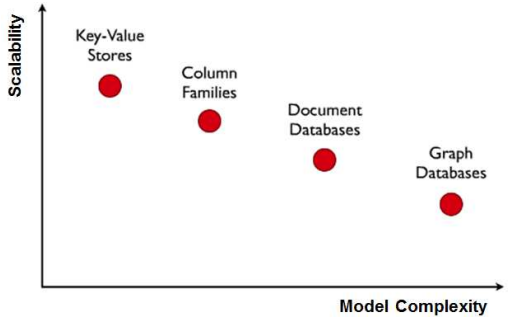
- Intérêts :
  - Plus efficaces que les bases relationnelles pour traiter les problématiques liées aux réseaux (cartographie, relations entre personnes, ...)
  - Modèle de données puissant
  - Performance d'accès pour les données liées, bien plus rapide qu'un modèle relationnel
- Limites :
  - Scalabilité moindre : Les mécanismes de synchronisation des écritures sur des graphes nécessite des opérations complexes. Ces systèmes sont donc fragiles au partitionnement des données sur des supports physiques distincts.

Stockage et manipulation de données – Solutions NoSQL

- Usages :
  - Manipulation d'objets complexes organisés en réseaux : cartographie, réseaux sociaux, ..
  - Moteurs de recommandations
  - Semantic Web
  - Social computing
  - Données géospatiales
  - Généalogie
  - Web of things
  - Catalogue des produits
  - Sciences de la Vie et calcul scientifique (bioinformatique, …)
  - Données liées, données hiérarchiques
  - Services de routage, d'expédition et de géolocalisation
  - Services financiers : chaîne de financement, dépendances, gestion des risques, détection des fraudes, …

Stockage et manipulation de données – Solutions NoSQL

- Pour choisir une technologie NoSQL, il faut se demander :
  - Quel type de données sont à manipuler ?
  - Comment les applications vont au final utiliser ces données ?
  - Quels sont les fréquences d'écriture, de lecture, de mise à jour ?
  - Quelle est la complexité des requêtes ?
  - Quels sont les prévisions d'augmentation du volume de données gérées ?



Stockage et manipulation de données – Résumé

- Les solutions NoSQL implémentent des solutions de stockage multiples : XML, JSON, …
- JSON est un format d'échange de données de plus en plus utilisé dans les solutions NoSQL.
- Les solutions NoSQL sont en rupture avec les solutions relationnelles, entre autres sur les points suivants :
  - la modélisation du stockage des données,
  - le langage d'accès aux données : pas de langage standard de manipulation des données.
- De nombreuses solutions NoSQL sont disponibles sur le marché. Ces solutions ont été classifiées en 4 grandes catégories, chacune apportant ses avantages et ses inconvénients et répondant à certains usages privilégiés.