

# Technologies du Web

Laure Soulier — laure.soulier@lip6.fr

Sorbonne Université

19 février 2019

## **Introduction *BIG DATA***

**Licence Informatique – Technologies du Web**

*Laure Soulier*

**2017-2018**

# Plan du chapitre 1

---

- Historique des Bases de Données (BD)
- Emergence des *Big Data*
- Vocabulaire autour des *Big Data*
- Aperçu des approches et solutions *Big Data*

# Donnée, information, connaissance (1)

## □ De la donnée à l'information

- Une donnée est l'enregistrement d'une **observation, objet, fait** destiné à être interprété, traité par l'homme. La donnée est généralement **objective**

*Exemples :*

- température = $35^{\circ}$
- âge = 2 mois

- Une information est le **signifiant attaché à la donnée** ou à un ensemble de données par association. L'information est généralement **subjective, définie selon un contexte**

*Exemples*

- (température= $35^{\circ}$ ) : temps chaud
- (âge=2 mois) : nourrisson

# Donnée, information, connaissance (2)

## □ De l'information à la connaissance

- Une connaissance est une information nouvelle, apprise par association d'informations de base, de règles, de raisonnement, d'expérience, d'expertise, etc. La donnée est généralement objective, peut être subjective.

*Exemple :*

*- temps chaud et enfant nourrisson alors risque de déshydratation*

# Bases de Données (BD) : c'est quoi ?

## □ Des fichiers à la base de données

- Un fichier est un ensemble d'enregistrements physiques qui représentent des données manipulées par plusieurs utilisateurs ayant une vue unique de ces données
- Une base de données est un ensemble de données construit selon un schéma, d'où peuvent être dérivés différentes vues manipulées par plusieurs utilisateurs

## □ De la base de données à la banque de données

- Une base de données est ensemble structuré de données, destinées à être exploitées par des applications cibles  
*Exemple*  
*Base de données personnel université (suivi de carrière, paie, ..)*
- Une banque de données comporte les données de référence associées à un domaine donné; elle est généralement structurée en un ensemble de bases de données.

# Bases de Données (BD) : c'est quoi ? (2)

## ❑ Base de Données (BD) ?

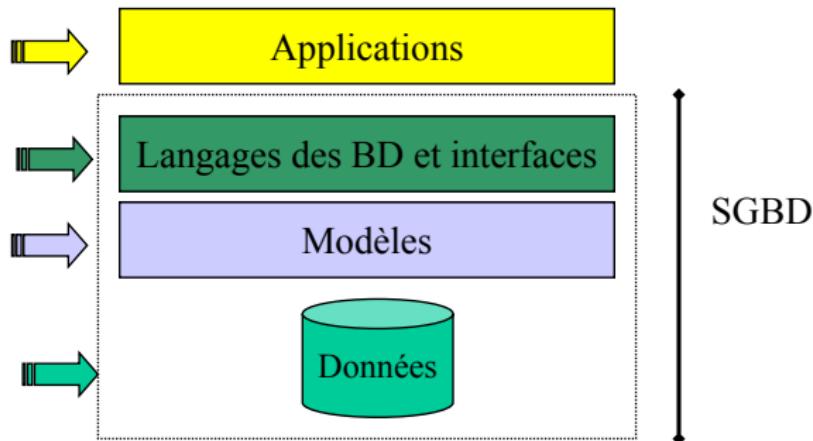
- Un ensemble structuré d'informations agrégées ou élémentaires accessibles par une communauté d'utilisateurs [Chrismont,08]
- Une collection de données qui intègre :
  - une structure intégrée,
  - des liaisons sémantiques entre données,
  - des contraintes d'intégrité,
  - des vues de différents utilisateurs.
- Une collection de données qui supporte des opérations de manipulation et de recherche de données :
  - cohérente,
  - sécurisée,
  - pérenne.

## BD et SGBD : Historique

### ❑ + 50 années de réflexion sur la gestion des données

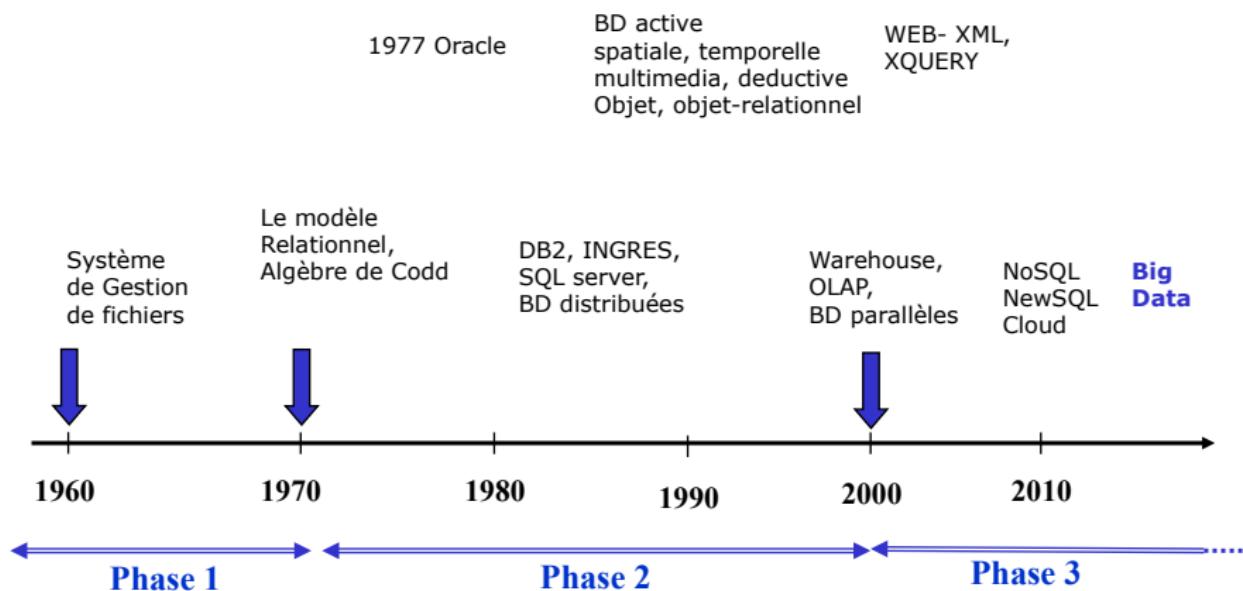
- Phase 1 : période préhistorique 1960 – 1969
- Phase 2 : période phare 1970 – 2000
- Phase 3 : période nouvelle 2000 – ...

*Evolution au niveau :*



Eléments extraits de Tutoriel,  
M. Adiba, EDBT 2013

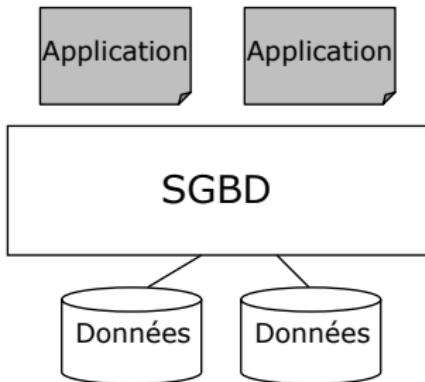
## BD et SGBD : Historique



## BD et SGBD : quelques repères historiques (60-69)

### □ Introduction des principes de bases des BD/SGBD

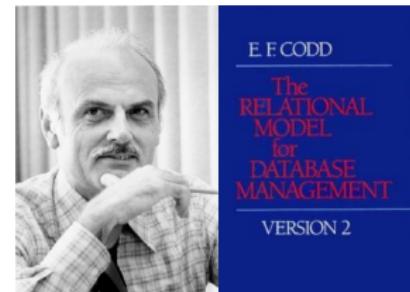
- Plusieurs applications partagent des données
- Séparation des données et des traitements sur les données
- Données gérées par un serveur central
- Minimisation de la redondance et de l'inconsistance
- Amélioration du contrôle des données
- Accès par langage standardisé (COBOL)



# BD et SGBD : quelques repères historiques (70-2000)

## □ Modèle relationnel

- Inventé par *Edgar Franck Codd* en 1970
- Fondements :
  - Algèbre relationnelle, logique de prédictat de 1<sup>er</sup> ordre
  - Indépendance des données, vue tabulaire
  - Langages : SQL, QUEL, QBE
  - Dépendances fonctionnelles, formes normales
- Prototypes SGBDR (1975)
- SGBDR commercialisés en 1980



# BD et SGBD : quelques repères historiques (70-2000)

## ❑ Notion de transaction et propriétés, J. Gray 1975

- Langage relationnel de requêtes
  - Langage déclaratif, non procédural
  - Indépendance des données/traitements
  - Introduction du principe d'optimisation des requêtes
- Propriétés ACID
  - Atomicité : principe de TOUT ou RIEN, une transaction est exécutée intégralement ou pas du tout
  - Cohérence : l'exécution de toute transaction assure le passage de la base d'un état cohérent vers un autre état cohérent
  - Isolation : une transaction est exécuté indépendamment des autres qui s'exécutent simultanément
  - Durabilité : les modifications opérées dans la base par une transaction sont pérennes
- Impact
  - Théorique : protocoles pour la gestion de la concurrence (exclusif, partagé, à deux phases,...)
  - Pratique : modules de gestion de la concurrence, contrôleurs de concurrence, algorithmes de reprise, gestion d'inter blocage

# BD et SGBD : quelques repères historiques (70-2000)

## ❑ Vers de nouveaux types de données...

- Temporel
  - Extensions SQL : SQL2, TempSQL, TQUEL, TSQL2,..*R. Snodgrass*, 1985, 1986
  - Introduction de propriétés temporelles, SQL2011  
(<http://www.sigmod.org/publications/sigmod-record/1209/pdfs/07.industry.kulkarni.pdf>)
- Spatial
  - Extension de SQL à des objets spatiaux, *M. EgenHofer*, 1994
  - Opérations et relations spatiales
  - Requêtes interactives : localisation des régions par l'utilisateur
- Multimédia
  - Introduction texte, image, audi, vidéo, *M. Adiba*, 1996
  - Extensions SQL (temps, données continues,...)

# BD et SGBD : quelques repères historiques (70-2000)

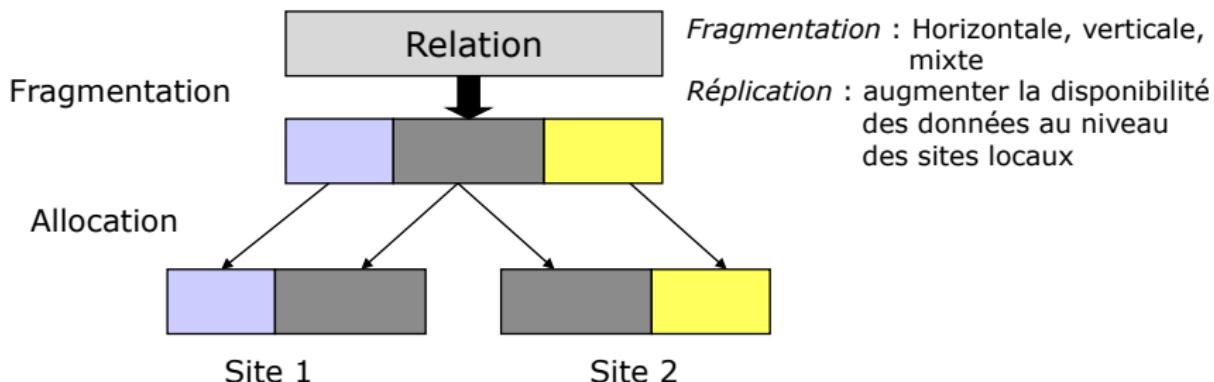
## ❑ Vers de nouveaux types de données...

- Document
  - Données + documents semi-structurés (XML)
  - GML (1971), SGML (1986), HTML (W3C, 1986), XML (W3C, 1998)
  - Introduction Xpath, Xquery, ..
- Objet, modèle NF2 (1985, 1986)
  - Non First Normal Form
  - Introduction des concepts classe, méthode, héritage
  - Extensions SQL : SQL2, OSQL
- Multimédia (1995)
  - Introduction texte, image, audi, vidéo, *M. Adiba*, 1996
  - Extensions SQL (temps, données continues,...)

# BD et SGBD : quelques repères historiques (70-2000)

## □ Données et systèmes distribués : milieu des années 70

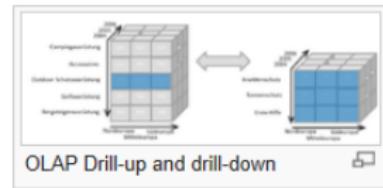
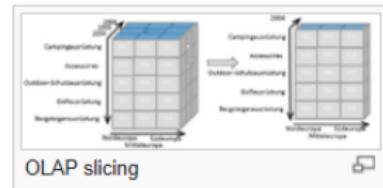
- Partition et replication (duplication) sur différents sites
  - Données
  - Schémas et catalogues de données
  - Système de contrôle (SGBD)
  - Infrastructure matérielle



# BD et SGBD : quelques repères historiques (70-2000)

## □ Data Warehouse & OLAP, 2000...

- OLAP : collection de données orientées « sujet », historisées, non volatiles consolidées dans une BD unique pour des besoins de gestion, prise de décision
- Schémas de données multidimensionnelles  
OLAP CUBE est une abstraction de l'opérateur relationnel de projection
- Requêtes décisionnelles plus complexes que pour une BD classique. Opérations de synthèse sur les données : rotation, *slicing*, *dicing*, forage vers le haut (*drill-up*), forage vers le bas (*drill-down*)



## BD et SGBD : quelques repères historiques (70-2000)

## Vers les *BIG DATA* 2010...

- **Volume, Variété, Vélocité** des données (**3V**)
  - Données peu (pas) structurées
  - Solutions open-source
  - Paradigme MAP REDUCE
  - Infrastructures pour la gestion des big data : HADOOP, Cassandra, ..



# Big Data, c'est quoi ? (1)

## □ Quelques définitions

- Définition 1 : « *data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges* » Oxford English Dictionary, « *données de très grande taille, dont la manipulation et gestion présentent des enjeux du point de vue logistiques* »
- Définition 2 : « *an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using on-hand data management tools or traditional data processing applications* » Wikipédia, « *englobe tout terme pour décrire toute collection de données tellement volumineuse et complexe qu'il devient difficile de la traiter en utilisant des outils classiques de traitement d'applications* »
- Définition 3 : « *datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze* » McKinsey, 2011, « *collections de données dont la taille dépasse la capacité de capture, stockage, gestion et analyse des systèmes de gestion de bases de données classiques* »

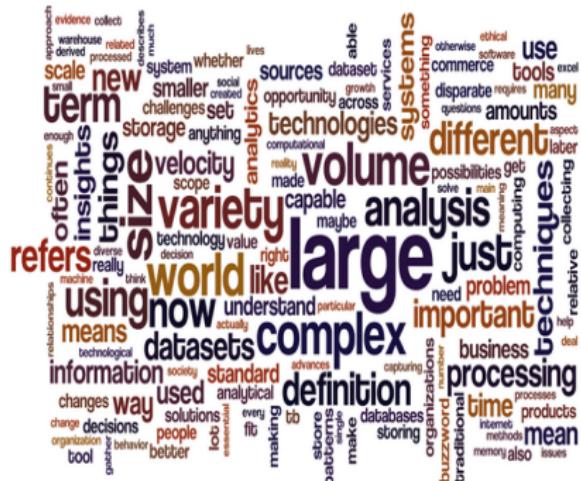
# Big Data, c'est quoi ? (2)

## □ Bien d'autres définitions encore...

<http://datascience.berkeley.edu/what-is-big-data/>

## □ Ce qu'on retient ...

Volume des données,  
Complexité,  
Limites des outils classiques  
de gestion des données,  
Passage à l'échelle  
..



Top recurring themes in our thought leaders' definitions (word cloud via Wordle □)

# Big Data, pourquoi ? (1)

## □ **Explosion des volumes des données générées sur le web, web mobile...**

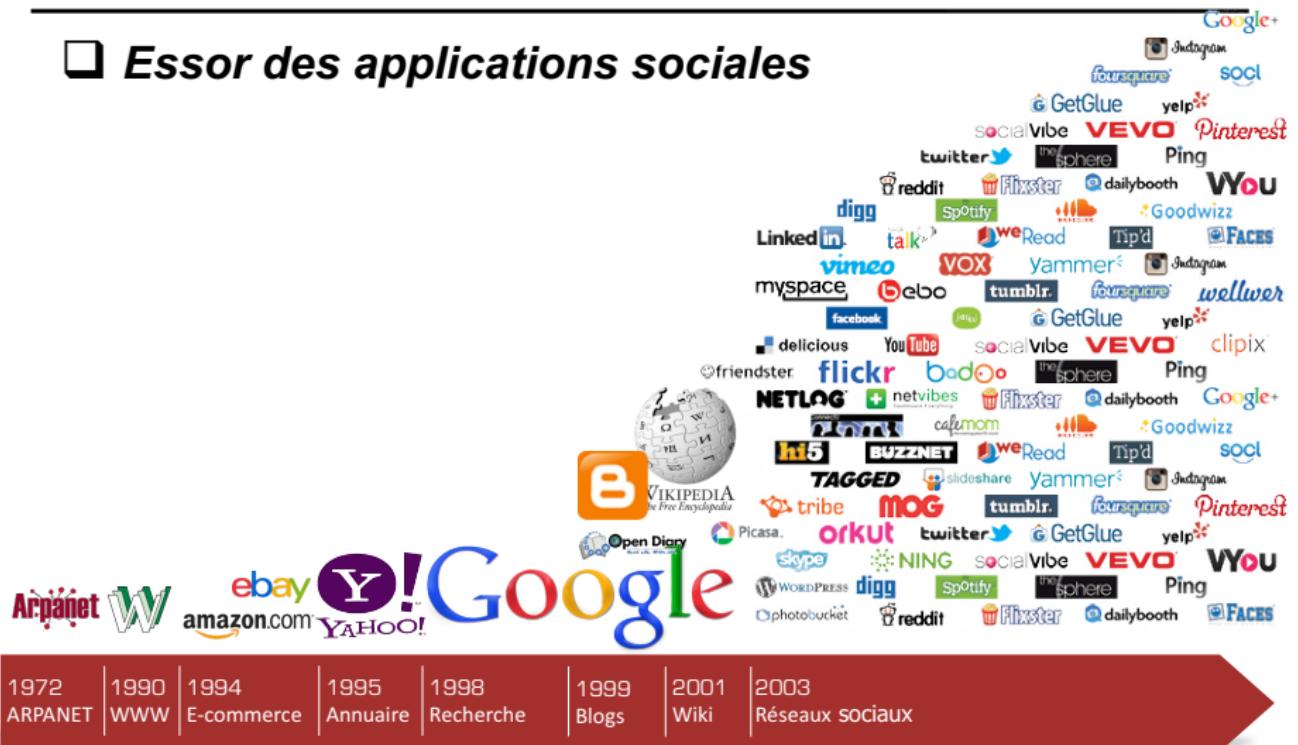
- Réseaux sociaux : Facebook, Twitter,..
- Moteurs de recherche : Google, Yahoo, Bing
- Internet des objets
- Sites commerciaux
- Appareils mobiles
- Capteurs
- Systèmes d'information des entreprises

## □+ **Disponibilité, ouverture des données**

- *Open data* : données ouvertes au grand public
  - Gouvernement
  - Industries
  - Services : transports, météo, ...
  - ...

# Big Data, pourquoi ? (2)

## ❑ Essor des applications sociales



# Big Data, pourquoi ? (3)

## □ Chiffres à l'appui : utilisateurs des réseaux sociaux

### Monthly Active Users (MAUs)

In Millions

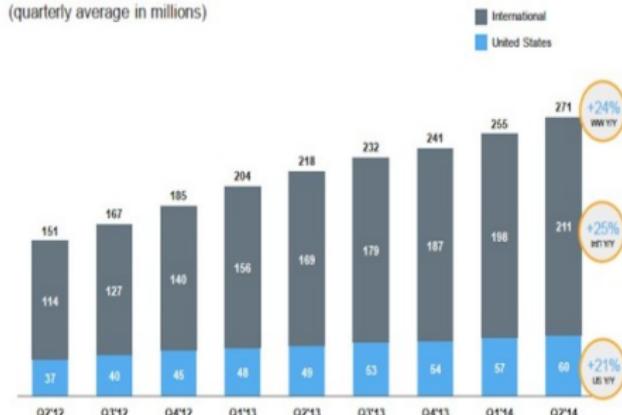
- Rest of World
- Asia
- Europe
- US & Canada



Please see Facebook's most recent quarterly or annual report filed with the SEC for definitions of user activity used to determine the number of MAUs, mobile MAUs, and mobile MAUs. The number of MAUs, mobile MAUs, and mobile MAUs do not include Instagram users unless they would otherwise qualify as such users, respectively, based on their other activities on Facebook.

### Monthly active users

(quarterly average in millions)

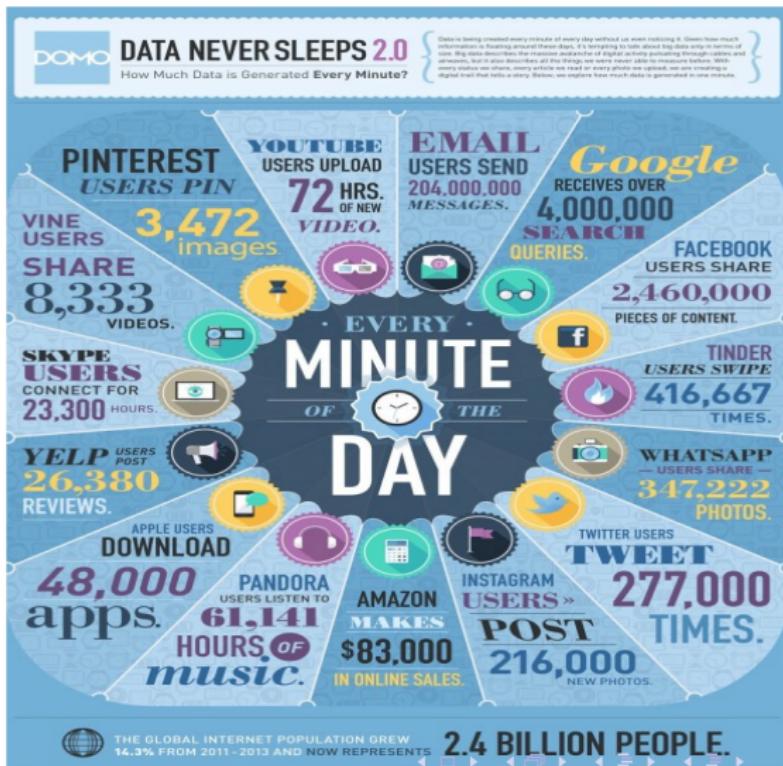


<http://www.blogdumoderateur.com/reseaux-sociaux/facebook/chiffres-facebook/>

<http://www.blogdumoderateur.com/reseaux-sociaux/twitter/chiffres-twitter/>

# Big Data, pourquoi ? (4)

- Chiffres à l'appui : volumes de données par minute sur le web



<http://www.blogdumoderateur.com/60-seconde-internet-2014/>

# *Big Data, pourquoi ? (5)*

## ***Explosion des volumes des données générées sur le web, web mobile...***

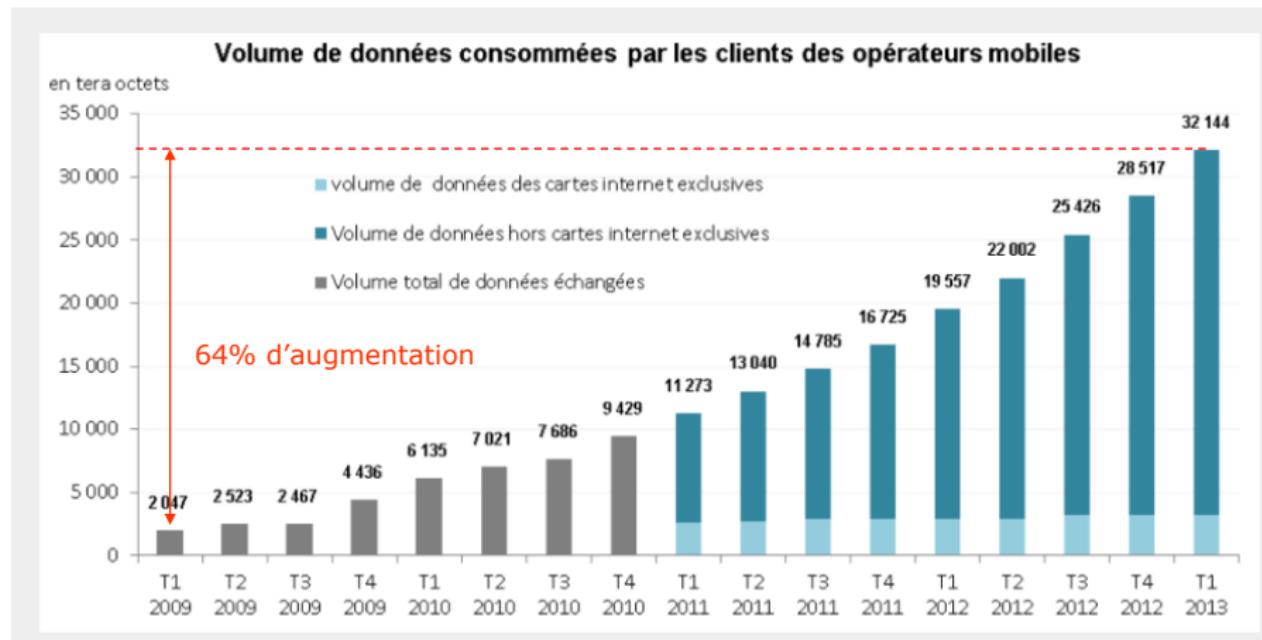
- Réseaux sociaux : Facebook, Twitter,..
- Moteurs de recherche : Google, Yahoo, Bing
- Internet des objets
- Sites commerciaux
- Appareils mobiles
- Capteurs
- Systèmes d'information des entreprises

## + ***Disponibilité, ouverture des données***

- *Open data* : données ouvertes au grand public
  - Gouvernement :
  - Industries
  - Services : transports, météo, ...
  - ...

# Big Data, pourquoi ? (6)

Croissance des volumes de données générées par les appareils mobiles en France



<http://thecallr.com/fr/blog/2013/07/29/une-augmentation-de-32-des-communications-telephoniques-mobiles-depuis-2012/>

# L'explosion des données

1 Zo = 1 000 000 000 000 000 000 octets, soit un milliard de Teraoctets

Produire 5 Mds de Go

L'humanité jusqu'en 2003

1 jour en 2011

10 minutes en 2013

90% des données produites  
ces 2 dernières années

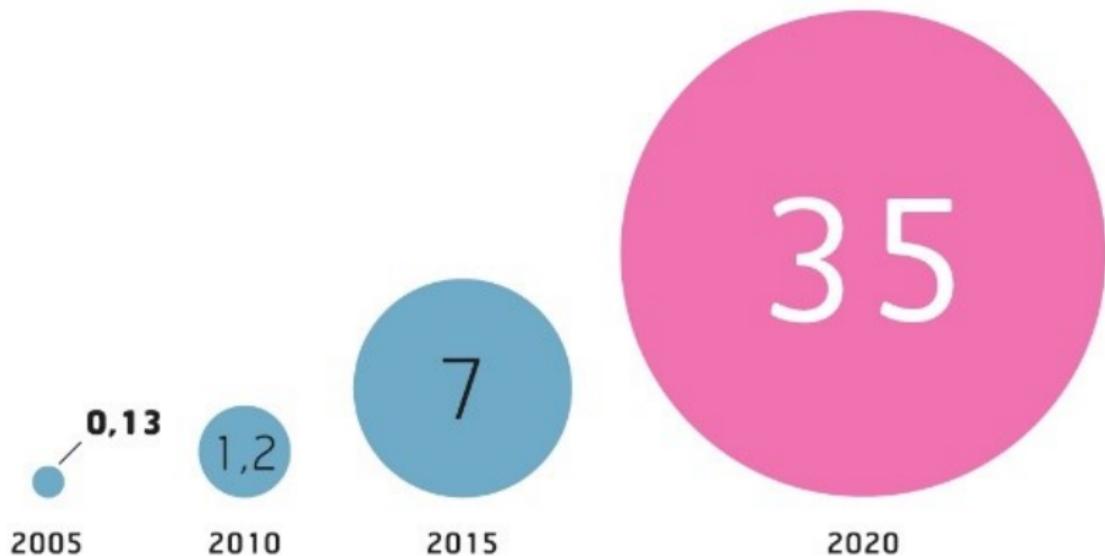


40 Zo à horizon 2020

<http://fr.slideshare.net/Affinity-Engine/big-data-43666013>

## LE VOLUME MONDIAL DE DONNÉES CRÉÉES ET RÉPLIQUÉES

EN ZETTAOCTETS (1 ZETTAOCTET = 1 TRILLION DE GIGAOCTETS)



« LES ÉCHOS » / SOURCE : IDC

Copyright L.Tamine-Lechani & L. Soulier

# Big Data, pourquoi ? (5)

## □ ..+ Variété des données, peu de structure...

- Image
- Vidéo,
- *Logs*,
- *Graphes*,
- *Son*,
- ..

## □ ..+ Dynamicité des données...

- Flux de d'images (TV stream),..
- Flux de *tweets*
- Flux de données des capteurs
- ...

## □ ..+ Variété des sources

- Mobiles
- Machine-Machine
- Machine-Homme
- Homme-Homme

# *Big Data, pourquoi ? (5)*

---

## ..+ Limites des SGBD

- Capacités de stockage / traitement des SGBD
  - 1980 : Teradata database machine
  - 2010 : Oracle Exadata Database machine
- Nature/type des données
  - Structurée ou semi-structurées
- Vitesse de stockage
  - Temps de stockage ne suit pas le progrès en termes de vitesse des réseaux

## ...Passage à l'échelle des SGBD à quel coût ?

# Big Data, pourquoi ? (6)

**Exercice :** Quel est le coût de stockage de 48 heures de vidéo extraites de Youtube dans une base ORACLE Exadata vs. système Big Data dédié

## Software License Costs for Reproducing YouTube Using Oracle

### Database Software

#### Per Machine:

CPUs per Exadata Server	16
* Cores per CPU	8
* Core Factor (Intel 7560)	0.5
* Price of Database Enterprise Edition w/ options	\$107,000
= Cost per Exadata Machine (\$M)	\$6.8
* Number of Exadata Machines	26
= Cost of Database Licenses (\$M)	\$178

### Middleware Software

#### Per Machine:

CPUs per Exalogic Server	30
* Cores per CPU	12
* Core Factor (Intel 7560)	0.5
* Price of WebLogic Enterprise Edition	\$25,000
= Cost per Exalogic Machine (\$M)	\$4.5
* Number of Exalogic Machines	22
= Cost of Middleware Licenses (\$M)	\$99

### Operating System Software

Total servers	70.0
* Price of Linux Ultimate Edition	\$2,299
= Total Linux Support Costs	\$0

### Exadata Storage Software

Total Exadata Cells	1,373
* Disks per Cell	12
= Total Disks	16,476
* Price Exadata Software/Disk	\$10,000
= Cost of Exadata Storage Licenses (\$M)	\$165

### Total License Cost (\$M)

### Maintenance Cost

Annual Maintenance Cost (\$M)

597

## Software Support Costs for Reproducing YouTube Using Open Source Software

### Data Management Software

Number of Data Management Server Racks	2
* Servers per Rack	8
* Annual Subscription per Server, Datameer Hadoop	\$12,000
= Cost of Database Support (\$M)	\$0.2

### Middleware Software

Number of App Server Racks	54
* Servers per Rack	8
* Number of CPUs	2
* Price of Jboss per CPU	\$1,406
= Cost of Middleware Support (\$M)	\$1.2

### Operating System Software

Number of Racks	222
* Servers per Rack	8
* Price of Red Hat Enterprise Linux Support	\$6,498
= Cost of Database Support (\$M)	\$11.5

### Total Support Cost (\$M)

\$12.9



# *Big Data, à quoi ça sert ? (1)*

## *Explosion des domaines d'application utilisant les Big Data*

- Médical
- Marketing
- Politique
- Economie,
- ...



*Pour ?*

- L'aide à la décision
- La prévision
- La découverte de nouvelles connaissances,...

# Big Data, à quoi ça sert ? (2)

## ❑ Quelques cas d'étude

- **Prédire les conflits mondiaux**

*L'outil GDELT, développé par l'université de Georgetown et accessible de manière open source, compile toutes les actualités (communiqués de presse, articles, discours...) parues depuis 1979. Il applique ensuite des techniques d'analyse sémantique et des algorithmes auto-apprenants pour faciliter la compréhension des événements récents et des principes de cause à effet pour arriver à prédire les conflits mondiaux –*

- **Gérer les catastrophes naturelles**

*En utilisant des outils de tracking, d'analyse sémantique et de visualisation en temps réel, l'Organisation Mondiale de la Migration a pu assister les forces locales en dégageant les urgences sanitaires, la localisation des ressources clés et en optimisant l'allocation des ressources sur le terrain lors du typhon qui a frappé les Philippines en 2013*

- **Faire de la veille sanitaire**

*Des scientifiques de l'université de Bringhma Youns essaient de simuler la localisation des mouches tsé-tsé dans le but d'aider à contrôler la propagation d'épidémies. De la même manière, la police de Chicago utilise le Big Data et la visualisation de données pour contrôler les populations de rats dans la ville.*

# Big Data, à quoi ça sert ? (2)

## ❑ Autres cas d'étude

- Cibler les clients sur le web

*Dans le marketing web par exemple, le phénomène d'enchères en temps réel (Real-Time-Bidding – RTB), s'appuie sur de la data en mouvement pour proposer une publicité spécifique en fonction de l'utilisateur qui se connecte au site. L'entreprise Turn par exemple, classe l'utilisateur dans un segment lorsqu'il se connecte au site, en fonction de son historique de navigation et des informations issues de réseaux sociaux et lui affiche la publicité de l'annonceur ayant fait la meilleure enchère pour ce segment...en moins de 10 millisecondes - <http://www.data-business.fr/big-data-definition-enjeux-etudes-cas/#sthash.kRSvs3hq.dpuf>*

- Bien d'autres...

- Secteur des Telecom. : analyse de la qualité de service en temps réel
- Secteur des banques : prévention des fraudes et gestion du risque
- Secteur des transports : optimisation de trafics et des taux de remplissage
- Secteur de l'éducation : au travers des Massive Open Online Courses : pour comprendre les comportements des apprenants, et adapter les programmes
- ...

# Big Data, situation actuelle ? (1)

## L'investissement en solutions « big data » en France

Part des entreprises, en %, en septembre 2012

- Utilise déjà des solutions « big data » → 10%
- Va commencer dans les prochains mois → 1%
- Etudie l'opportunité d'investir dans ces solutions → 18%
- Ni projets, ni réflexions → 70%

## L'évolution du marché mondial du « big data »

En milliards de dollars



« LES ÉCHOS » / SOURCE : IDC / PHOTO : DR

# Big Data, situation actuelle ? (2)

## □ En France... (source 01Business, 17/07/14)

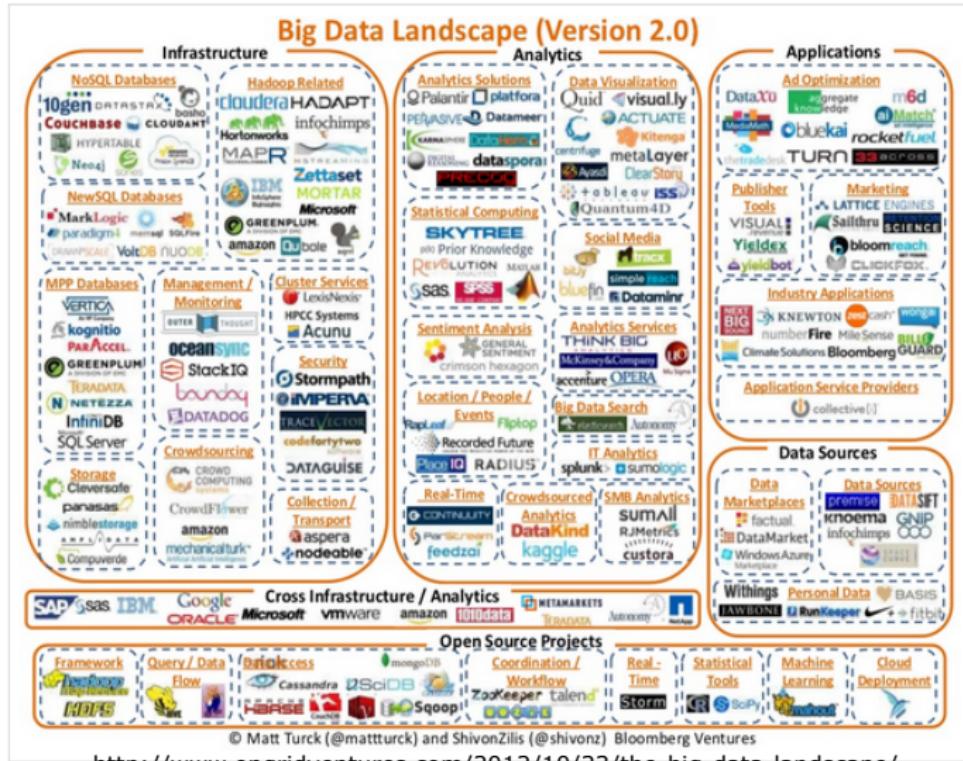
« **Environ 10 % des entreprises françaises** en utiliseraient déjà (une solution Big Data) selon une étude de Steria de 2013, contre un tiers au niveau mondial.

« *De nombreuses structures ont commencé à réaliser des POC (Proof of Concept), mais peu ont déroulé un projet de A à Z pour en tirer des enseignements et un retour sur investissement clair* »

Gilbert Grenié, associé de l'activité conseil au sein de PWC, partenaire de l'EBG pour le livre blanc Big Data

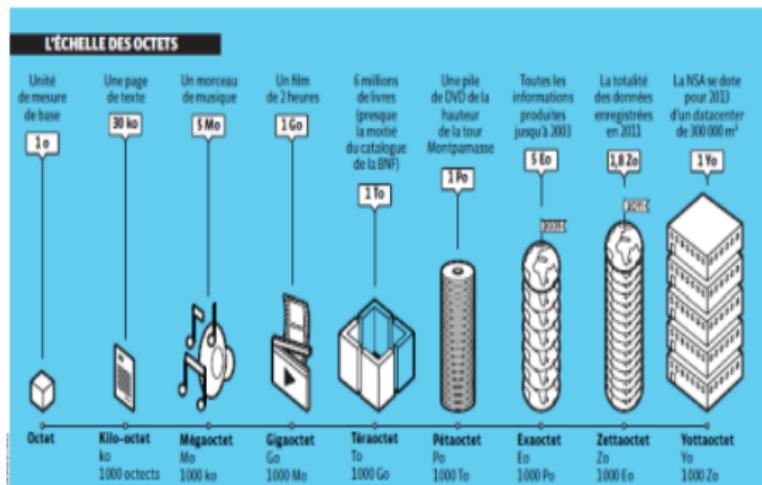
**Principales causes** : manque de compétences autour des big data : informatique pour les données massives, statistique, ...

# Mots autour des Big Data



## Vocabulaire de base : unités de mesure de capacité de stockage

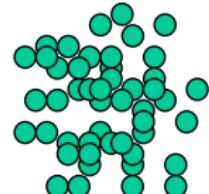
Unité de mesure	Eq. Octets
GigaOctets	$10^9$
TeraOctets	$10^{12}$
PetaOctets	$10^{15}$
Exaoctets	$10^{18}$
ZetaOctets	$10^{21}$



## Vocabulaire de base : Dimensions des Big Data ou les Big V

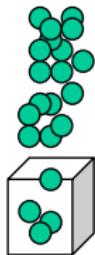
### □ **Volumétrie**

- Grande quantité de données
- Difficultés : stockage, recherche, partage, analyse, visualisation,...



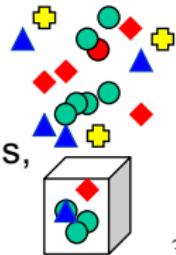
### □ **Vélocité**

- Flux continus de données : capteurs, appareils mobiles, réseaux sociaux...
- Difficultés : analyse et traitement des données à la volée, sans les avoir en intégralité (*one-pass processing*)



### □ **Variété**

- Différents formats : séquences, graphes, ..
- Difficulté d'intégration (jointure, association) par le sens, l'échelle, la qualité, ...



## Vocabulaire de base

Mot	Brève
MAP REDUCE	Principe de programmation qui consiste à distribuer et paralléliser le traitement sur plusieurs nœuds
HADOOP, HDFS (Hadoop Distributed File System)	Hadoop est une plate-forme informatique open-source de la fondation Apache, capable de gérer/traiter des big data sur une architecture distribuée. HDFS est le système de gestion de fichier de base qui supporte Hadoop
NOSQL	Technologie qui se différencie à la notion relationnelle des données, adaptée à des données peu structurées (nombre dynamique de colonnes, document, graphes,..)
HBase, Cassandra, MongoDB, NE04J, Couche DB, Redis	SGBD qui supportent l'approche d'interrogation des données NOSQL
SAS, Talend, R, Python	Outils et ou environnements de programmation et analyse adaptés aux Big Data
Cloud computing	Ensemble de processus permettant d'offrir un espace de stockage sous forme de serveurs, accessibles à distance, sous forme de location. Utilile pour les entités (entreprises) qui ne souhaitent pas investir dans les infrastructures de stockage

# Quelles solutions pour le Big Data ?

## ❑ Direction majeure

Exploiter le parallélisme sur une architecture multi-processeurs

## ❑ Comment ?

- Machines de bases de données
  - Pour les données massives, structurées, semi-structurées
  - Permet de pérenniser les solutions BD existantes => préservation des acquis, économie d'argent
  - Solutions propriétaires : ORACLE, MySQL, ...: amélioration des services à moindre coût
- Environnement de programmation parallèle
  - MAP REDUCE , inventé par Google
  - Version logiciel libre (Open source) par Hadoop
  - Adapté aux données dynamiques, irrégulières, sans schéma qui sont inadaptées pour SQL, Xquery
- Systèmes de Gestion de Bases de Données NoSQL
  - Pour les données non structurées : graphes, textes, ..



Avec possibilités de combinaisons de ces solutions

## Bases de données NoSQL

# NoSQL

## Historique

### Historique

- NoSQL : Not Only SQL
- 2004 : Big Table Google
- 2008 : Cassandra Facebook
- 2009 : lancement d'une communauté de développement logiciels open source NoSQL

### Spécificités

- Manipulation de données peu (pas) structurées, schéma flexible, sans modèle pré-défini
- Manipulation de données volumineuses
- Haute disponibilité
- Capacités élevées de lectures/écritures

# SQL vs. NoSQL

	Système SQL	Système NoSQL
<b>Finalité</b>	Système de gestion de bases de données	Système de gestion de données
<b>Indépendance données/programmes</b>	Garantie (séparation du niveau logique/physique)	Non garantie
<b>Schéma des données</b>	Prédéfini, relativement fixe, peu flexible	Pas besoin de fixer le schéma, schéma dynamique, schema-less, schema-later
<b>Eclatement/répartition des données</b>	Eclatement en respect du principe de normalisation, choix de la répartition des données difficile	Principe de proximité physique des données reliées logiquement
<b>Transaction</b>	Respect des propriétés ACID	Relâchement des propriétés ACID
<b>Valeurs d'attributs</b>	Trois états : vrai, faux, NULL	Variable : pas de valeurs, pas d'état NULL, ...

# NoSQL

## NoSQL

NoSQL (Not Only SQL), apparu en 2009, et comme son nom l'indique une alternative au langage SQL et au modèle relationnel de base de données que vous avez l'habitude d'utiliser. L'idée est de proposer une architecture souple et puissante avec une forte disponibilité et de faibles contraintes.

Particulièrement à la mode, la majorité des grandes entreprises du web abandonnent leurs bases de données traditionnelles et portent leur propre projet NoSQL : Facebook et Twitter utilisent Cassandra (de la Fondation Apache), Amazon SimpleDB, LinkedIn Voldemort, etc.

# Théorème CAP

Propriétés fondamentales pour les systèmes distribués

- Coherence : tous les noeuds du système voient exactement les mêmes données au même moment
- Availability (Disponibilité) : garantie que toutes les requêtes reçoivent une réponse ;
- Partition tolerance (Résistance au partitionnement) : sauf coupure totale du système, aucune panne ne peut empêcher le système de fonctionner normalement

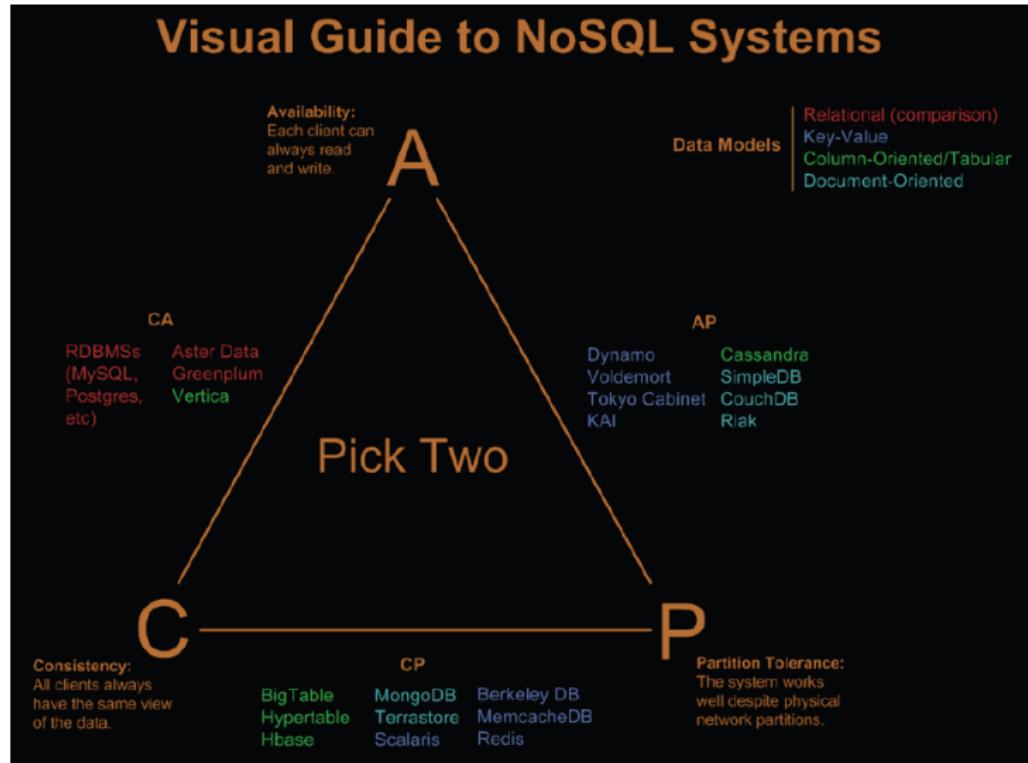
Dans un système distribué, seules deux propriétés sur trois peuvent être assurées à un instant  $t$ .

## Exemples (wikipedia)

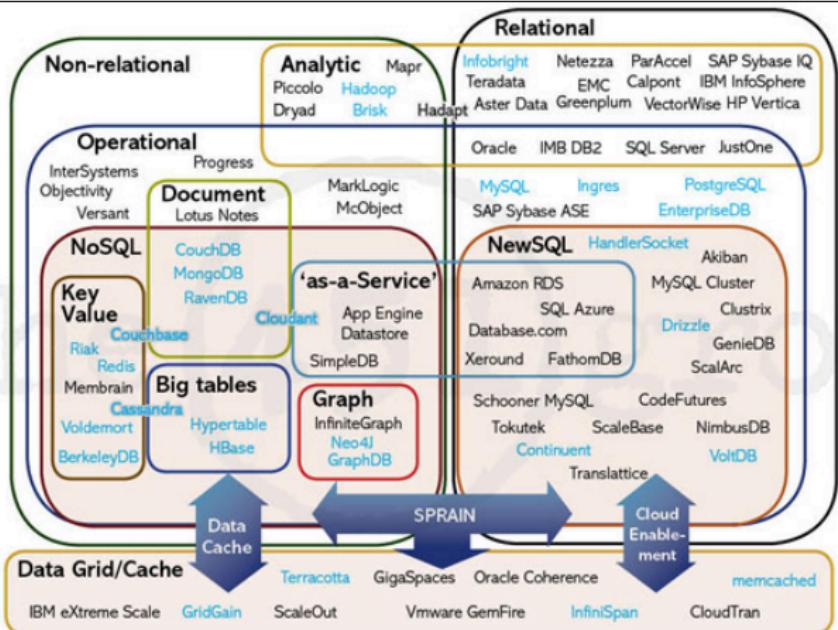
Soit A et B deux utilisateurs, soit N1 et N2 deux noeuds.

- Si A modifie une valeur sur N1, alors pour que B voie cette valeur sur N2 il faut attendre que N1 et N2 soient synchronisés (cohérence).

*Exemple d'arbitrage* : deux utilisateurs qui font une même recherche peuvent obtenir des résultats différents, mais cet inconvénient est moins grave que de ne pas avoir de résultats du tout.



# Typologie des modèles de données NoSQL



[http://blogs.the451group.com/information\\_management/2011/04/15/nosql-newsql-and-beyond/](http://blogs.the451group.com/information_management/2011/04/15/nosql-newsql-and-beyond/)

# Typologie des modèles de données NoSQL

## ❑ Bases de données clé-valeur (1/4)

- Similaire à un tableau associatif / table de hachage
- Clé : permet d'accéder à la valeur
- Valeur :
  - Simple chaîne de caractères ou objet sérialisé
  - Absence de structure ou de typage

Clé	Valeur
k1	v1
k2	v2
...	...

Exemples :

- Réseaux sociaux :  
id utilisateur → liste de ses amis
- Livres :  
Id livre → informations (auteur, éditeur, ...)
- Journaux  
Date → liste des événements

# Typologie des modèles de données NoSQL

## ❑ Bases de données clé-valeur (2/4)

- Opérations (CRUD)
  - Create : Création d'un nouvel objet
  - Read : Lecture d'un objet à partir de la clé
  - Update : Mise à jour d'une valeur à partir de la clé
  - Delete : Suppression d'un objet à partir de la clé
- Indexation
  - Index primaire sur la clé
  - Pas d'indexation secondaire (autres attributs)

# Typologie des modèles de données NoSQL

## ❑ Bases de données clé-valeur (3/4)

- Avantages
  - Performance élevée en lecture et écriture
  - Modèle de données simple
  - Passage à l'échelle facile
- Inconvénients
  - Interrogation limitée : obligation de connaître la clé
  - Structure trop simple pour les données complexes
  - Complexité déportée sur l'application : les traitements faits de la requête SQL doivent maintenant être faits par l'application
- Solutions : Amazon Dynamo, FundationDB, Redis, Riak, ...



# Typologie des modèles de données NoSQL

## ❑ Bases de données clé-valeur (4/4)

- Exemple :

Gestion du serveur 'azteka' de la fac avec redis



- Créer un objet : stocker le nom du serveur  
`SET server:name "azteka"`
- Lire l'objet : récupérer le nom du serveur  
`GET server:name` → "azteka "
- Supprimer un objet  
`DEL server:name`
- Modification d'un objet :  
`SET server:name "azteka2"` → écrase la valeur précédente

# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées colonne (1/5)

- Modèle le plus d'un SGDB relationnel
- Nombre de colonnes dynamiques (pas de valeurs nulles)
- Une « clé cachée » (*rowkey*) pour chaque objet, triée par ordre lexicographique
- Les colonnes sont traitées indépendamment
- Les lignes peuvent être reconstituées plus tard

# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées colonne (2/5)

Nom (2Go)	Note (1Go)
Durand	10
Dupond	15
Dupoint	14

BD orientée ligne :

1:Durand,10 ; 2:Dupond,15 ; 3:Dupoint,14

BD orientée colonne :

Durand:1 ; Dupond:2 ; Dupoint:3

10:1 ; 15:2 ; 14:3

→ Traitement par tableau des valeurs de colonnes

→ Améliore les performances lorsque traitement sur un nombre réduit de colonnes

Exemple : Moyenne des notes

- BD orientée ligne : chargement de toute la table (3Go)
- BD orientée colonne : chargement de l'attribut Note (1Go)

# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées colonne (3/5)

- Terminologie :
  - Clés
    - » Point d'entrée de la donnée.
  - Familles de colonnes
    - » Structure définie
    - » Regroupement par lignes de plusieurs colonnes
    - » Utilisées pour l'indexation, par exemple
  - Colonnes
    - » Entités de base (attribut)
    - » Associées à une paire clé-valeur
    - » Pas de structure du schéma de la valeur
  - Timestamp
    - » Associé aux colonnes
    - » Gestion de la cohérence de la BD par rapport aux versions
- Pour accéder à une donnée  
Clé → Famille → Colonne → Timestamp (par défaut : le plus récent)

# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées colonne (4/5)

- Avantages
  - Bon passage à l'échelle
  - Indexation naturelle (colonnes)
  - Gestion des versions → résultats en temps réels
- Inconvénients
  - Lecture difficile pour les données complexes
  - Maintenance : ajout/suppression/regroupement des colonnes
  - Les requêtes dépendent des données
- Solutions : Hbase (Open Source de BigTable utilisé par Google), Cassandra (Facebook), Hyperbase (Amazon)



# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées colonne (5/5)

- Exemple :
  - Définir la structure de la table  
`CREATE 'serveurs', {NAME => 'nom', VERSIONS => 3}, {NAME => 'config', VERSIONS => 2} , {SPLITS => ['1000','2000','3000','4000']}`
  - Créer un objet : stocker le serveur azteka  
`PUT 'serveurs', '1', 'nom', 'azteka', 'admin:nom', 'soulier', 'config', 'unix'`
  - Lire l'objet : récupérer le nom du serveur tel qu'il a été inséré lors de la version 1  
`GET 'serveurs', '1' ,{COLUMN => 'nom' , VERSIONS =1}`
  - Supprimer un objet  
`DELETEALL 'serveurs' , '1'`
  - Suppression de la table (désactivation, puis suppression)  
`DISABLE 'serveurs'`  
`DROP 'serveurs'`

APACHE  
HBASE

# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées document (1/3)

- Stockage d'une collection de documents
- Basé sur les paires clés-valeurs
  - Clé : identifie le document
  - Valeur : document en format semi-structuré hiérarchique (XML, JSON)

### • Exemple de fichier JSON

```
{  
  "card": "2",  
  "numbers": {  
    "Conway": [1,11,21],  
    "Fibonacci": [0,1]  
  }  
}
```

### • Exemple de fichier XML

```
<xml>  
  <card>2</card>  
  <numbers>  
    <Conways>  
      <Conway>1</Conway>  
      <Conway>11</Conway>  
      <Conway>21</Conway>  
    </Conways>  
    <Fibonaccis>  
      <Fibonacci>0</Fibonacci>  
      <Fibonacci>1</Fibonacci>  
    </Fibonaccis>  
  </numbers>  
</xml>
```

# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées document (2/3)

- Avantages
  - Modèle de données simple mais puissant (documents hiérarchiques)
  - Pas de maintenance pour l'ajout/suppression des documents
  - Requêtes assez complexes (simplifie l'application)
- Inconvénients
  - Lenteur pour les grandes requêtes
  - Interconnexion des données difficile
- Solutions : MongoDB, CouchDB, CouchBase



Copyright L.Tamine-Lechani & L. Soulier



# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées document (3/3)

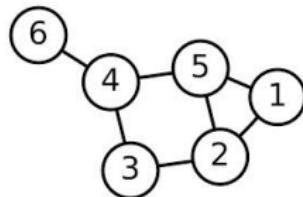
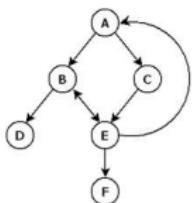
- Exemple avec mongoDB
  - Création d'une collection (table en SGBDR)  
`db.createCollection("macollection")`
  - Insertion d'un document  
`db.macollection.insert({id : '1', titre : 'cours M2', année : '2014-2015'})`
  - Lecture d'un document  
`db.macollection.find({},{'titre':'cours M2'})`
  - Mise à jour d'un document  
`db.macollection.update({'titre': 'cours M2'},{$set:{'année : '2015-2016'}})`
  - Suppression d'un document  
`db.macollection.remove({'titre': 'cours M2'})`
  - Suppression d'une collection  
`db.macollection.drop()`



# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées graphe (1/3)

- Stockage de données complexes avec des relations
- Basé sur la théorie des graphes
  - Nœuds : moteur de stockage
  - Relations : description des arcs
- Application : traitement des données des réseaux sociaux



# Typologie des modèles de données NoSQL

## ❑ Bases de données orientées graphe (2/3)

- Avantages
  - Modèle de données puissant
  - Stockage et requêtage efficace pour les données complexes et interconnectées
- Inconvénients
  - Fragmentation difficile
- Solutions : Neo4j, OrientDB, InfiniteGraph, ...



# Typologie des modèles de données NoSQL

## ❑ **Bases de données orientées graphe (3/3)**

- Exemple avec Neo4j
  - Création d'un nœud  
`CREATE n ={nom : 'Milou', ville : 'Paris'};`
  - Lecture d'un nœud  
`START n=node(1) return n;`
  - Création d'un nœud avec une relation  
`START Milou = node(1)  
CREATE  
Tintin = {nom : 'Tintin', ville : 'Paris'},  
Castafiore = {nom : 'Castafiore', ville : 'Paris'},  
Milou-[r:AMI]->Tintin  
RETURN  
Tintin;`



## MongoDB (Wikipédia)

MongoDB est un système de gestion de base de données :

- orientée documents,
- libre,
- montant bien en puissance (scalable),
- à performance raisonnable,
- ne nécessitant pas de schéma prédéfini des données,
- écrit avec le langage de programmation C++.

Il fait partie de la mouvance NoSQL et vise à fournir des fonctionnalités avancées. Utilisée par Foursquare, bit.ly, Doodle, Disqus

# MongoDB



## Autres propriétés

- Document-oriented storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Rich, document-based queries.
- Map/Reduce
- GridFS
- Commercial Support

# MongoDB

## Drivers

- C
- C++
- Erlang
- Haskell
- Java
- Javascript
- .NET (C# F#, PowerShell, etc)
- Perl
- PHP
- Python
- Ruby
- Scala

# MongoDB : Principes

- Une instance de Mongodbd peut contenir une ou plusieurs base de données
- Une base de données peut avoir une ou plusieurs **collections**
  - ▶ Equivalent des tables dans les bases classiques
- Une collection peut avoir de zéro à plusieurs **documents**
  - ▶ Les documents d'une même collection n'ont pas obligatoirement le même "schéma" (champs)
  - ▶ Les documents correspondent aux enregistrement (tuples)
  - ▶ Des documents peuvent contenir des documents (structure hiérarchique)
  - ▶ Chaque document possède une clef unique
- MongoDB stockent l'information au format BSON (Binary JSON)

# MongoDB : Propriétés

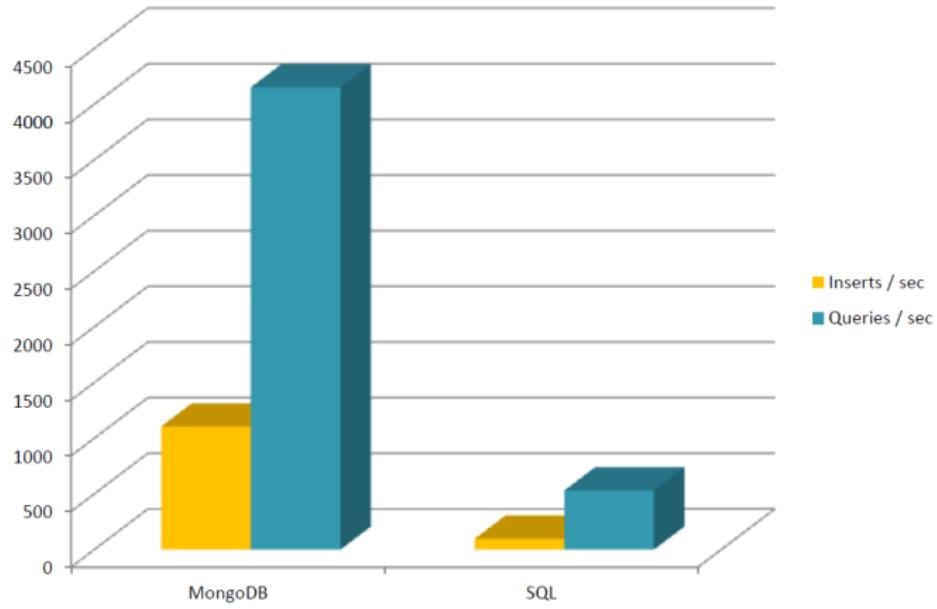
## Avantages

- Requêtes "faciles"
- Fait sens avec la plupart des applications Web
- Facilité d'intégration des données

## Inconvénients

Pas bien adapté pour les systèmes transactionnels complexes

# MongoDB



# MongoDB

Un document :

```
user = {  
    name: "Z",  
    occupation: "A scientist",  
    location: "New York"  
}
```

# MongoDB

Un document :

```
{  
  "name": "John Smith",  
  "address": {  
    "street": "Lily Road",  
    "number": 32,  
    "city": "Owatonna",  
    "zip": 55060  
  },  
  "hobbies": ["yodeling", "ice skating"]  
}
```

# MongoDB

Une collection :

```
{ "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
  "Last Name": "DUMONT",
  "First Name": "Jean",
  "Date of Birth": "01-22-1963" },
{ "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
  "Last Name": "PELLERIN",
  "First Name": "Franck",
  "Date of Birth": "09-19-1983",
  "Address": "1 chemin des Loges",
  "City": "VERSAILLES" }
```

*Obligatory, and automatically generated by MongoDB*

# MongoDB

Imaginons que l'on veuille modéliser un blog :

## Mauvais design

```
post = {
    id: 150,
    author: 100,
    text: 'This is a pretty awesome post.',
    comments: [100, 105, 112]
}
author = {
    id: 100,
    name: 'Michael Arrington'
    posts: [150]
}
comment = {
    id: 105,
    text: 'Whatever this sux.'
}
```

## Autre design

```
post = {  
    author: 'Michael Arrington',  
    text: 'This is a pretty awesome post.',  
    comments: [  
        'Whatever this post sux.',  
        'I agree, lame!'  
    ]  
}
```

Pourquoi celui-ci est-il meilleur ?

# MongoDB

```
location1 = {  
    name: "10gen HQ",  
    address: "17 West 18th Street 8th Floor",  
    city: "New York",  
    zip: "10011",  
    latlong: [40.0,72.0],  
    tags: ["business", "cool place"],  
    tips: [  
        {user:"nosh", time:6/26/2010, tip:"stop by for office hours on  
        Wednesdays from 4-6pm"},  
        {.....},  
    ]  
}
```

## Example queries:

- db.locations.find({latlong:{\$near:[40,70]}})
- db.locations.find({zip:"10011", tags:"business"})
- db.locations.find({zip:"10011"}).limit(10)

# MongoDB

```
user1 = {  
    name: "nosh"  
    email: "nosh@10gen.com",  
    .  
    .  
    .  
    checkins: [{ location: "10gen HQ",  
        ts: 9/20/2010 10:12:00,  
        ...},  
        ...  
    ]  
}
```

# MongoDB

- SQL

- `INSERT INTO t (fn, ln) VALUES ('John', 'Doe')`

- MongoDB

- `db.t.insert({fn:'John', ln: 'Doe'})`

# MongoDB

- SQL

- UPDATE t SET ln='Smith' WHERE ln='Doe'

- MongoDB

- db.t.update({ln:'Doe'}, {\$set:{ln:'Smith'}})

- Only 1 doc. is updated by default

- Specify `{multi:true}` for multi-doc updates

- db.t.update({ln:'Doe'}, {\$set:{ln:'Smith'}}, {multi:true})

# MongoDB

- SQL

- `DELETE FROM t WHERE fn='John'`

- MongoDB

- `db.t.remove({fn:'John'})`

# MongoDB

- SQL
  - `DROP TABLE t`
- MongoDB
  - `db.t.drop()`

# MongoDB

- SQL

- `SELECT * FROM t`

- MongoDB

- `db.t.find()`

- SQL

- `SELECT id, name FROM t`

- MongoDB

- `db.t.find({}, {name:1})`

# MongoDB

- SQL

- `SELECT * FROM t WHERE fn='John'`

- MongoDB

- `db.t.find({ln:'John'})`

- SQL

- `SELECT id, age FROM t WHERE fn='John' AND ln='Smith'`

- MongoDB

- `db.t.find({fn:'John',ln:'Smith'}, {age:1})`

# MongoDB

- SQL

- ```
SELECT country FROM t
WHERE fn='John'
ORDER BY age DESC LIMIT 10
```

- MongoDB

- ```
db.t.find(
  {fn:'John'},
  {country:1, _id:0}
).sort({age:1}).limit(10)
```

# MongoDB

- SQL

- ```
SELECT fn,ln FROM t
WHERE age > 30
```

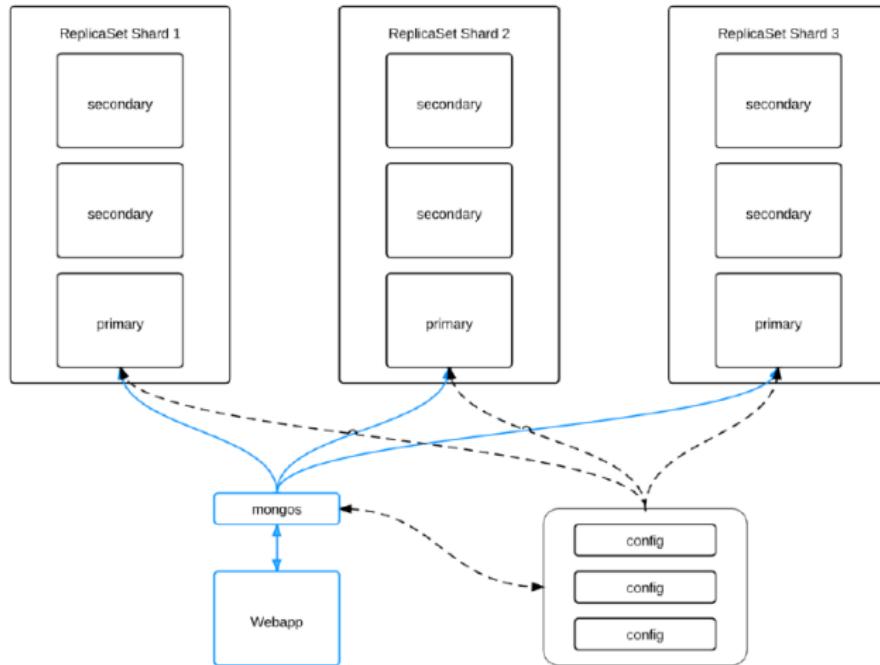
- MongoDB

- ```
db.t.find(
  {age:{$gt:30}},
  {fn:1, ln:1, _id:0}
)
```

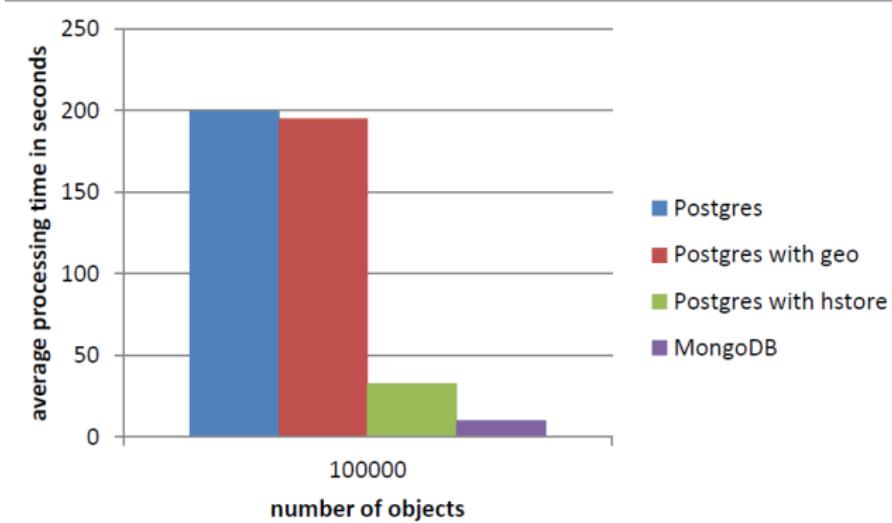
# Sharding

- Le sharding consiste en la répartition des données en plusieurs instances de bases de données, selon un critère donné.
- La montée en charge horizontale est plus simple à mettre en oeuvre : il est plus facile et moins coûteux d'acheter de nouveaux serveurs plutôt que d'augmenter la capacité d'un seul serveur.
- Un shard est un serveur mongo normal : il peut contenir des collections shardées ou non, et peut faire partie d'un replica set.
- Il est important de bien choisir ses critères de sharding pour éviter une trop grande disparité.

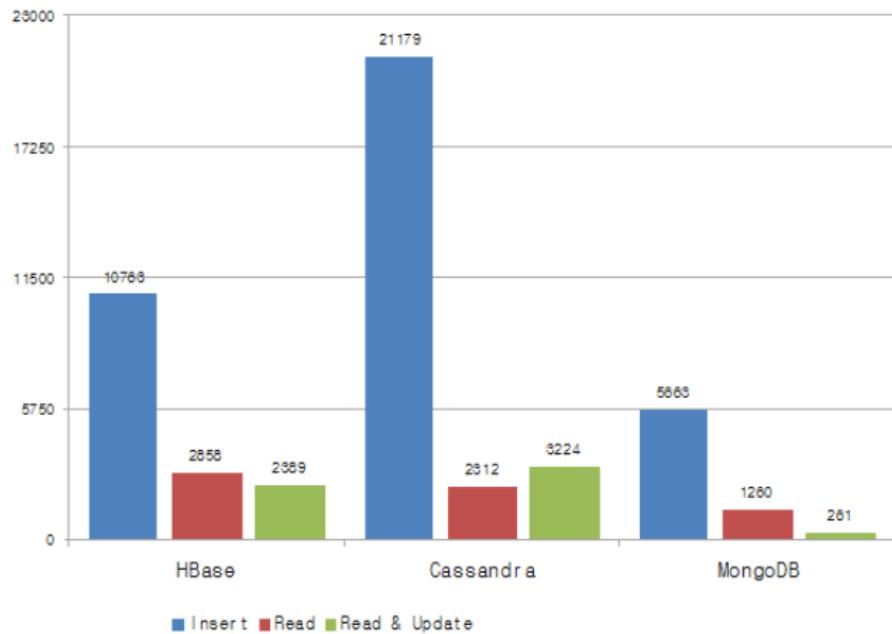
# MongoDB



# MongoDB



# MongoDB



## Nouvelle Connexion

```
1 import com.mongodb.client.MongoClients;
2 import com.mongodb.client.MongoDatabase;
3 import com.mongodb.client.MongoClient;
4
5 MongoClient mongo = MongoClients.create();
// or
7 MongoClient mongo = MongoClients.create("mongodb://host1");
8
9 MongoDatabase mDB = mongo.getDatabase("mydb");
```

Le Pooling de connection est automatiquement géré

# MongoDB en JAVA

## Liste des collections (tables)

```
1 for (String name : database.listCollectionNames()) {  
2     System.out.println(name);  
3 }  
4 }
```

## Utilisation des collections (tables)

```
1 DBCollection coll = db.getCollection("testCollection")
```

# MongoDB en JAVA

## Insertion

```
{  
    "name" : "MongoDB",  
    "type" : "database",  
    "count" : 1,  
    "info" : {  
        x : 203,  
        y : 102  
    }  
}
```

# MongoDB en JAVA

## Insertion

```
1 BasicDBObject doc = new BasicDBObject();
2
3     doc.put("name", "MongoDB");
4     doc.put("type", "database");
5     doc.put("count", 1);
6
7     BasicDBObject info = new BasicDBObject();
8
9         info.put("x", 203);
10        info.put("y", 102);
11
12        doc.put("info", info);
13
14        coll.insert(doc);
```

# MongoDB en JAVA

## Requête

```
1 Document doc = collection.findOne(); //get first document
2 System.out.println(doc);
```

# MongoDB en JAVA

## Requête

```
1 Document query = new Document();
2 query.append("number", 5);
3 FindIterable<Document> cursor = collection.find(query);
4 while(cursor.hasNext()) {
5     System.out.println(cursor.next());
6 }
7
8 //or
9 MongoCursor<Document> cursor = collection.find(query).iterator();
0 while(cursor.hasNext()) {
1     System.out.println(cursor.next());
2 }
```

# MongoDB en JAVA

## Requête

```
1 Document query = new Document();
2 List<Integer> list = new ArrayList<Integer>();
3 list.add(9);
4 list.add(10);
5 query.append("number", new Document("$in", list));
6 MongoCursor<Document> cursor = collection.find(query).iterator();
7 while(cursor.hasNext()) {
8     System.out.println(cursor.next());
9 }
```

# MongoDB en JAVA

## Requête

```
1 Document query = new Document();
2 query.append("number", new Document("$gt", 5));
3 MongoCursor<Document> cursor = collection.find(query).iterator();
4 while(cursor.hasNext()) {
5     System.out.println(cursor.next());
6 }
```

# MongoDB en JAVA

## Requête

```
1 Document query = new Document();
2 query.append("number", new Document("$gt", 5).append("$lt", 8));
3 MongoCursor<Document> cursor = collection.find(query).iterator();
4 while(cursor.hasNext()) {
5     System.out.println(cursor.next());
6 }
```

# MongoDB en JAVA

## Requête

```
1 Document query5 = new Document();
2 query5.append("number", new Document("$ne", 8));
3 MongoCursor<Document> cursor = collection.find(query).iterator();
4 while(cursor.hasNext()) {
5     System.out.println(cursor.next());
6 }
```

# MongoDB en JAVA

## Création d'un index

```
1 col.createIndex(Indexes.ascending("stars"));
2 // create index on "stars", ascending
```

# MongoDB en JAVA

## Requête

```
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import com.mongodb.*;

/**
 * Java MongoDB : Query document
 */
public class QueryDocumentApp {
    public static void main(String[] args) {
        try {
            MongoClient mongo = MongoClients.create("mongodb://host1");
            DB db = mongo.getDatabase("yourdb");

            // get a single collection
            MongoCollection<Document> message_collection = mongo.getCollection("my

            // insert number 1 to 10 for testing
            for (int i = 1; i <= 10; i++) {
                collection.insert(new Document().append("number", i));
            }
            //...
            System.out.println("Done");
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (MongoException e) {
            e.printStackTrace();
        }
    }
}
```

# Documentation

<http://mongodb.github.io/mongo-java-driver/3.10/driver/tutorials/>

# Conclusion

## Vous savez...

- ...utiliser MongoDB
- ...en JAVA

## Il faut...

- ...finir le MySQL (servlet User + Friends)
- ...intégrer MongoDB au site actuel (servlet MEssages)
  - ▶ ...Définir un "format" de stockage des commentaires
  - ▶ ...Implémenter les servlets de "recherche" des commentaires récents

## Vous ne savez pas...

- ...faire un moteur par recherche de mots-clefs
- à base de Map/Reduce
- ... plus tard