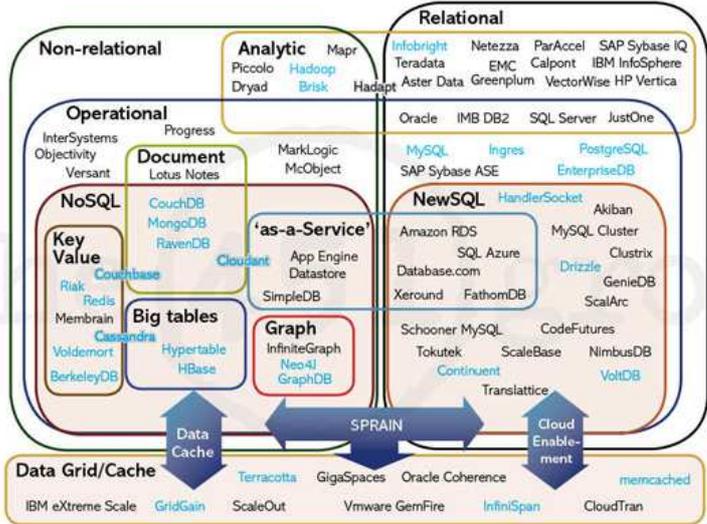


Plan

- Généralités
 - SGBD Relationnel
 - Théorème *CAP* ; Propriétés *BASE* ; Protocole Paxos
 - SGBD *NoSQL*
- Modèles de SGBD *NoSQL*
 - Classement des SGBD
 - 4 modèles : Logiciels *NoSQL* (modèles et langages) ; Comparaison entre SGBDR et modèles de SGBD *NoSQL* ; Utilisation et logiciels ; Présentation
 - Modèles : clé-valeur, colonne, document, graphe
- Bibliographie ~ Webographie

Généralités



Cf. blogs.the451group.com/information_management/2011/04/15/nosql-newsq-and-beyond/, 4
NoSQL, NewSQL and Beyond: The answer to SPRAINed relational databases, M. Aslett, 15/4/2011

SGBD relationnel (SGBDR)

- SGBDR généralement transactionnel : réalise des transactions atomiques, cohérentes, isolées et durables (propriétés ACID) c.-à-d. \forall concurrence, toutes les opérations de mise à jour des données (insertion, modification, suppression) sont prises en compte et sérialisées tout en préservant l'intégrité des données
N. B. : logiciel de gestion des transactions ACID (notamment dans le cas distribué) complexe ce qui peut peser sur les performances
- SGBDR moins efficace pour des données non structurées
N. B. : modèle *Entity-Attribute-Value* (ex. : gestion de tables « creuses ») : mise à jour du schéma des données flexible mais peu performant et coûteux en temps de développement
- SGBDR exige des logiciels et des ordinateurs coûteux ainsi que des compétences en optimisation peu répandues pour gérer de très gros volumes de données (devant donc être distribuées)
N. B. : MapReduce : distribution des données sur de nombreux serveurs vers lesquels on pousse les programmes (car plus efficace de transférer un petit programme sur le réseau plutôt qu'un grand volume de données)

5

Théorème CAP

S. Gilbert et N. Lynch (2002), conjecture d'E. Brewer

- Un système informatique de calcul distribué ne peut garantir à un instant donné qu'au plus deux des trois contraintes suivantes :
 - Cohérence (*Consistency*) : tous les nœuds du système voient exactement les mêmes données au même moment
 - Disponibilité (*Availability*) : la perte de nœuds n'empêche pas le système de fonctionner correctement (chaque client peut toujours lire et écrire)
 - Tolérance au partitionnement (*Partition tolerance*) : aucune panne moins importante qu'une coupure totale du réseau ne doit empêcher le système de répondre correctement (en cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome)

Corollaire : transactions ACID impossibles dans un environnement distribué

- Illustration

Après qu'un premier utilisateur modifie une valeur sur l'un des nœuds du système, un second utilisateur voulant lire cette valeur sur un autre nœud doit attendre leur synchronisation pour garantir la cohérence. Or, ce temps incompressible d'attente, sur un système très chargé et très vaste, va considérablement influencer la disponibilité et la résistance au morcellement.

- Complément :

infoq.com/fr/articles/cap-twelve-years-later-how-the-rules-have-changed,⁶
E. Brewer (traduction : O. Bourgain, 5/8/2013)

Propriétés *BASE*

- SGBD issus de la nuagique (*cloud computing*) et des systèmes distribués :
 - privilégiant la haute disponibilité des données (distribuées), la rapidité, la simplicité
 - au détriment de la cohérence, de l'exactitude de la réponse
- Propriétés *BASE* :
 - *Basically Available* : le système doit toujours être accessible (ou indisponible sur de courtes périodes)
 - *Soft state* : l'état de la BD n'est pas garanti à un instant donné (les mises à jour ne sont pas immédiates : cf. cohérence à terme)
 - *Eventual consistency* : la cohérence des données à un instant donné n'est pas primordiale (mais assurée à terme : verrouillage optimiste en reportant à plus tard la vérification de l'intégrité)

7

Protocole Paxos

- Paxos
 - Famille de protocoles pour résoudre le consensus (processus permettant de parvenir à une décision sur un résultat) dans un réseau de nœuds faillibles
- Protocole Paxos
 - Informations gérées : échanges entre les nœuds, temps entre chaque réponse à un message avant de prendre une décision, niveau d'activité des participants, nombre de messages envoyés, types de pannes
 - Chaque acteur a au moins un (plusieurs pour permettre de baisser la latence entre les messages) rôle (*client*, *acceptor*, *proposer*, *learner*, *leader*) à un instant donné
 - Pas d'incohérence possible, et les conditions qui peuvent l'empêcher de progresser (s'exécuter jusqu'à apporter une réponse valable) sont rares
Théorème : aucun algorithme de consensus résistant aux pannes ne permet de garantir de progresser sur un réseau asynchrone
N. B. : intégrité garantie si moins de la moitié des processus en panne
 - Très efficace pour la lecture dans un environnement distribué, beaucoup moins pour l'écriture/modification ; ne gère pas les transactions ACID
 - Cas d'utilisation : application devant assurer la durabilité

8

Protocole Paxos : rôles

- *Client*
 - Le *client* envoie des requêtes au système distribué et attend une réponse
- *Acceptor* (Votant)
 - Les *acceptors*, regroupés en quorums, servent de mémoire résistante au panne
 - Chaque message envoyé à un *acceptor* doit l'être au quorum entier
- *Proposer*
 - Un *proposer* pousse la requête du *client* : il doit convaincre les *acceptors* de tomber d'accord, et agit comme coordinateur pour avancer quand un conflit se présente
- *Learner*
 - Les *learners* servent à la réplication : une fois qu'une requête d'un *client* a été acceptée par les *acceptors*, le *learner* peut agir (c.-à-d. exécuter une requête et envoyer la réponse au *client*)
 - N. B. : pour augmenter la disponibilité, on peut ajouter des *learners*
- *Leader*
 - *Proposer* spécifique pour avancer
 - Plusieurs processus peuvent croire être le *leader*, mais le protocole ne garantit l'avancement que si l'un d'eux est choisi (si plusieurs processus croient qu'ils sont *leaders*, ils peuvent bloquer le protocole en envoyant continuellement des propositions conflictuelles, mais l'intégrité des données est cependant toujours préservée dans ce cas)

9

SGBD *NoSQL* (1/3)

- Définition
 - SGBD non fondé sur l'architecture des SGBDR, *open source*, distribué, *horizontally scalable* (montée en charge par ajout de serveurs)
- Origine
 - « Les SGBDR en font trop, alors que les produits *NoSQL* font exactement ce dont vous avez besoin » selon J. Travis (lors de la rencontre *meetup NoSQL* de San Francisco du 11/6/2009)
 - Gestion des BD géantes des sites web de très grande audience
 - Exemple des SGBD d'annuaires : grande majorité des accès aux BD consistant en lectures sans modification (ainsi, seule la persistance doit être vérifiée)
- « Consensus » actuel
 - Les SGBD *NoSQL* ne remplacent pas les SGBDR mais les complètent en palliant leurs faiblesses
- *UnQL* (*Unstructured Query Language*)
 - 2011 : début d'une spécification d'un langage de manipulation standardisé (pour formaliser le requêtage des collections des BD *NoSQL*)

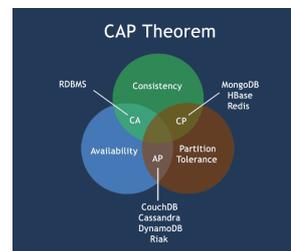
10

SGBD *NoSQL* (2/3)

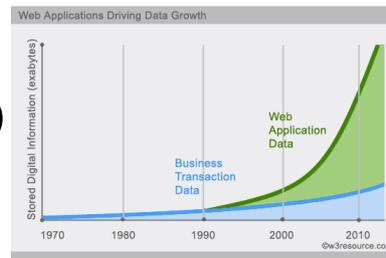
Simplification en renonçant aux fonctionnalités classiques des SGBDR :

- Redondance (via réplication)
- Pas forcément de schéma normalisé, initialement voire à terme
- Pas de tables mais des collections
- Rarement du *SQL* (LAG déclaratif, complet au sens de Turing depuis SQL-99) mais *API* simple ou langage spécialisé
- Pas forcément de jointure mais multiplication des requêtes, cache/réplication/données non normalisées, données imbriquées
- Transactions pas forcément ACID mais plutôt *BASE*
- *P* s'impose pour un système distribué :
AP (accepte de recevoir des données éventuellement incohérentes) voire *CP* (attendre que les données soient cohérentes)

Cf. w3resource.com/mongodb/nosql.php



SGBD *NoSQL* (3/3)



- Gestion des mégadonnées (*big data*) du web, des objets connectés, etc.
- Structure des données hétérogène et évolutive
- Données complexes et pas toujours renseignées
- Environnement distribué : données répliquées et accédées d'un peu partout (dans le monde), traitement répartis
- Techniques de partitionnement des BD : *sharding*, hachage cohérent (*consistent hashing*)
- Contrôle de concurrence multi-version (*Multi-Version Concurrency Control (MVCC)*)
 - Modification d'une donnée non par écrasement des anciennes données par les nouvelles mais en indiquant que les anciennes données sont obsolètes et en ajoutant une nouvelle version (seule la plus récente étant correcte) ... ce qui nécessite une purge régulière des données obsolètes
- Adaptation du protocole Paxos
- Performances linéaires avec la montée en charge (les requêtes obtiennent toujours aussi rapidement une réponse)

12

Modèles de SGBD *NoSQL*

Classement des SGBD

Rank			DBMS	Database Model	Score		
Apr 2021	Mar 2021	Apr 2020			Apr 2021	Mar 2021	Apr 2020
1.	1.	1.	Oracle	Relational, Multi-model	1274.92	-46.82	-70.51
2.	2.	2.	MySQL	Relational, Multi-model	1220.69	-34.14	-47.66
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1007.97	-7.33	-75.46
4.	4.	4.	PostgreSQL	Relational, Multi-model	553.52	+4.23	+43.66
5.	5.	5.	MongoDB	Document, Multi-model	469.97	+7.58	+31.54
6.	6.	6.	IBM Db2	Relational, Multi-model	157.78	+1.77	-7.85
7.	7.	8.	Redis	Key-value, Multi-model	155.89	+1.74	+11.08
8.	8.	7.	Elasticsearch	Search engine, Multi-model	152.18	-0.16	+3.27
9.	9.	9.	SQLite	Relational	125.06	+2.42	+2.87
10.	10.	10.	Microsoft Access	Relational	116.72	-1.41	-5.19
11.	11.	11.	Cassandra	Wide column	114.85	+1.22	-5.22
12.	12.	12.	MariaDB	Relational, Multi-model	96.37	+1.92	+6.47
13.	13.	13.	Splunk	Search engine	88.49	+1.56	+0.41
14.	14.	14.	Hive	Relational	78.50	+2.46	-5.56
15.	16.	23.	Microsoft Azure SQL Database	Relational, Multi-model	71.84	+0.96	+32.89
16.	17.	16.	Amazon DynamoDB	Multi-model	70.73	+1.84	+6.46
17.	15.	15.	Teradata	Relational, Multi-model	70.55	-0.88	-6.04
18.	20.	18.	SAP HANA	Relational, Multi-model	53.44	+2.45	+0.15
19.	19.	19.	SAP Adaptive Server	Relational, Multi-model	51.67	-0.51	-0.96
20.	18.	21.	Neo4j	Graph	51.04	-1.28	+0.24
21.	21.	17.	Solr	Search engine, Multi-model	50.60	+0.39	-2.99
22.	22.	20.	FileMaker	Relational	46.40	+0.84	-5.68
23.	23.	22.	HBase	Wide column	44.16	-0.53	-5.98
24.	24.	26.	Google BigQuery	Relational	35.58	-0.68	+7.61
25.	25.	24.	Microsoft Azure Cosmos DB	Multi-model	33.52	+1.11	+1.47
26.	26.	25.	Couchbase	Document, Multi-model	30.76	-0.24	+0.35
27.	27.	27.	PostGIS	Spatial DBMS, Multi-model	28.45		
28.	27.	29.	InfluxDB	Time Series, Multi-model	26.55	-0.31	+4.93
29.	30.	100.	Snowflake	Relational	26.46	+3.27	+24.06
30.	28.	27.	Memcached	Key-value	25.02	-0.38	-0.31

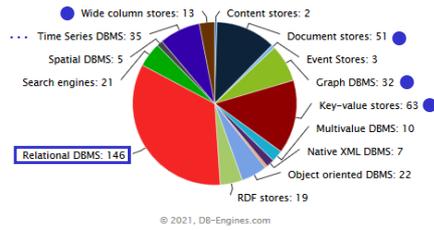
Select a ranking

- Complete ranking
- **Relational DBMS**
- Key-value stores
- Document stores
- Time Series DBMS
- Graph DBMS
- Object oriented DBMS
- Search engines
- RDF stores
- Wide column stores
- Multivalued DBMS
- Native XML DBMS
- Spatial DBMS
- Event Stores
- Content stores
- Navigational DBMS

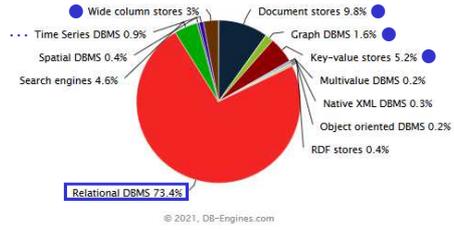
Cf. db-engines.com/en/ranking

Classement des SGBD par modèle (1/2)

Number of systems per category, April 2021

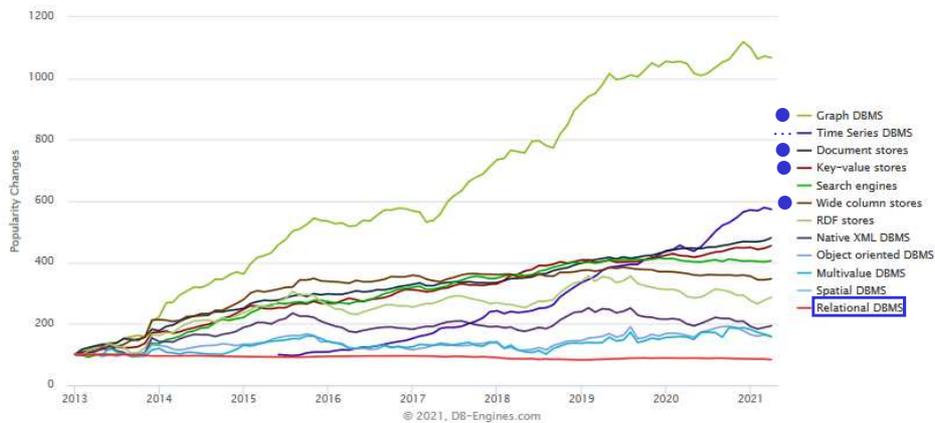


Ranking scores per category in percent, April 2021



Cf. db-engines.com/en/ranking_categories

Classement des SGBD par modèle (2/2)



Cf. db-engines.com/en/ranking_categories

Tableau 1-1. Caractéristiques des différentes solutions

Type (modèle)	Nom (logiciel)	Langage	Date création	Langage d'interrogation natif	Web Services	Bibliothèque (API)
● Colonne	Hbase	Java	2007	Non	/	Java
	Hypertable	C++	2007	HQL	/	/
	Cloudata	Java	2011	CQL	REST	JAVA
● Clé-Valeur	Membase	C et C++	2009	/	/	/
	Kyoto	C++	/	/	/	C, C++, Java, C#, Python, Ruby, Perl...
	Redis	C	2009	/	/	C, C++, Java, Python, Ruby, ...
● Clé-Valeur/CI (contrainte d'intégrité)	Oracle	/	2012	/	/	Java, C
	Cassandra	Java	2008	/	/	Java, Python, PHP, Ruby...
	Voldemort	Java	2008	/	/	/
● Document	Riak	Erlang, C, Javascript	2008	/	/	Java, Ruby, Python, PHP, Javascript
	CouchDB	C, Javascript	2005	/	REST	/
● Graphe	MongoDB	C++	/	OUI	/	C, C++, Java, C#, Ruby, Javascript...
	Neo4j	Java	2003	SPARQL	REST	Java, Python, Ruby, PHP
	FlockDB	Scala	2010	/	/	/

Cf. editions-ellipses.fr/PDF/9782340002616_extrait.pdf,

p. 8 de l'extrait du chapitre 1 du livre de P. Lacomme, S. Aridhi, R. Phan

17

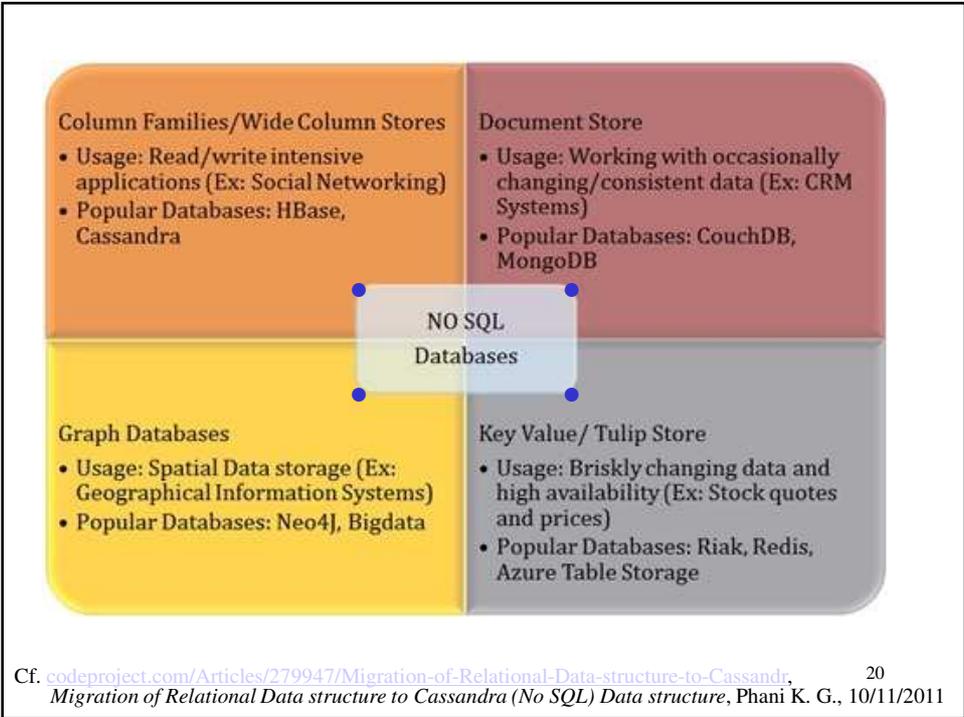
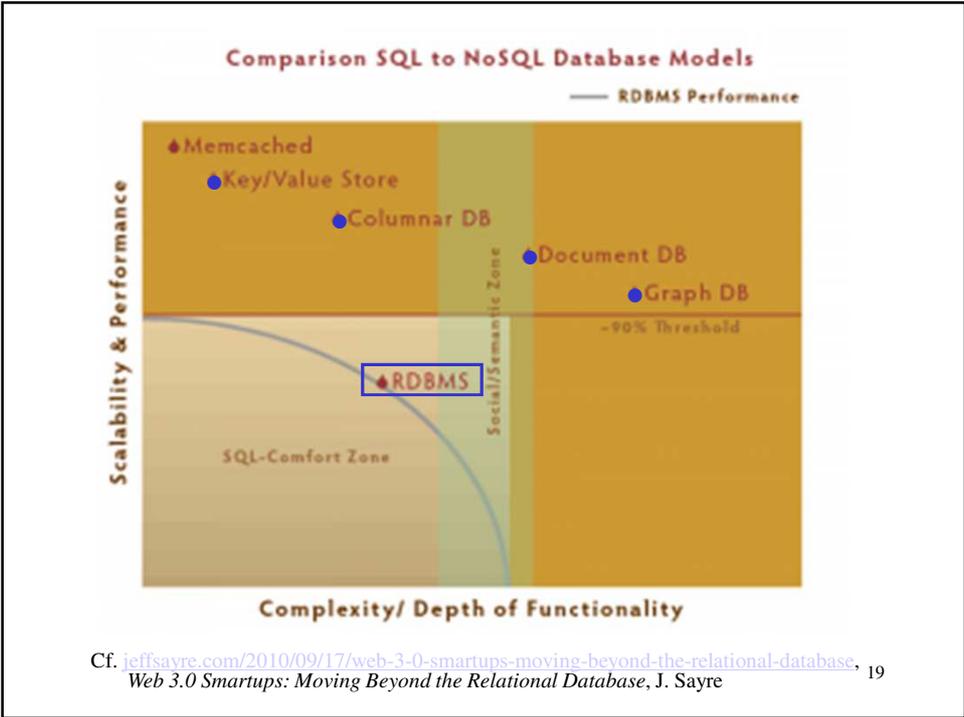
Comparaison entre SGBDR et modèles de SGBD *NoSQL*

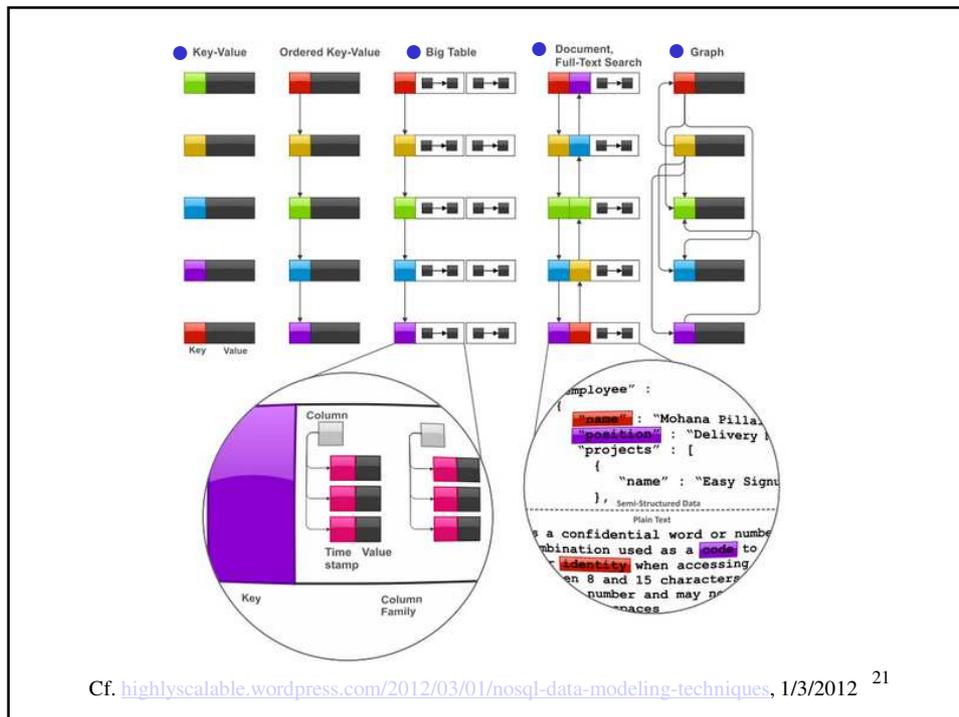
Data Model	Performance	Scalability	Flexibility	Complexity	Functionality
● Key-Value Store	high	high	high	none	variable (none)
● Column-Oriented Store	high	high	moderate	low	minimal
● Document-Oriented Store	high	variable (high)	high	low	variable (low)
● Graph Database	variable	variable	high	high	graph theory
● Relational Database	variable	variable	low	moderate	relational algebra

Cf. fr.slideshare.net/bcofield/nosql-codemash-2010,

B. Scofield, *NoSQL - Death to Relational Databases(?)*, 14/1/2010

18

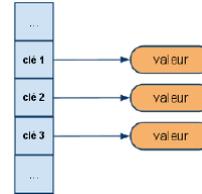




Présentation de 4 modèles de SGBD *NoSQL*

- Clé-valeur (*Key-value (KV)*)
Collection de couples (clé,valeur) (*key-value pairs*) = tableau associatif (*associative array, map, symbol table, dictionary*)
- Colonne
Modèle clé-valeur permettant de disposer de plusieurs valeurs (et donc des associations *one-to-many*)
- Document
Modèle clé-valeur permettant de disposer d'une valeur complexe (un document), chaque document étant composé de champs et de valeurs associées
- Graphe
Gestion de relations multiples entre les objets

Modèle clé-valeur (1/2)



- Définition
 - BD = 1 tableau associatif unidimensionnel
 - Chaque objet de la base représenté par un couple (clé,valeur) est identifié par une clé unique qui est le seul moyen d'accès à l'objet
 - Structure de l'objet libre (du ressort du programmeur)
Variante : clés triées en ordre lexicographique
- Opérations
 - Les 4 opérations *CRUD* :
 - `create (clé, valeur)` : crée un couple (clé,valeur)
 - `read (clé)` : lit une valeur à partir de sa clé
 - `update (clé, valeur)` : modifie une valeur à partir de sa clé
 - `delete (clé)` : supprime un couple à partir de sa clé
 - Souvent interface *HTTP REST (Representational State Transfer)* disponible depuis n'importe quel langage

Cf. smile.fr/Smile-france/Livres-blancs/Culture-du-web/Nosql

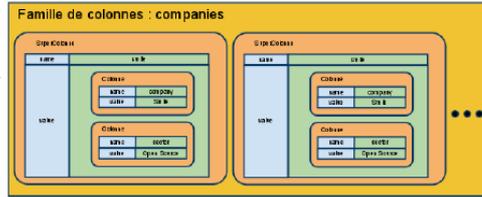
23

Modèle clé-valeur (2/2)

- Utilisation
 - Dépôt de masses de données avec des besoins de requêtage simple pour des analyses en temps-réel (sessions web et fichiers de *log*, profils utilisateurs, données de capteurs, ...), gestion de caches
- Logiciels
 - Riak (ou Amazon Dynamo), Redis, Voldemort, Oracle NoSQL Database
- Critique
 - ☺ simple, très performant, bonne mise à l'échelle, disponibilité, évolutivité des valeurs
 - ☹ interrogation seulement sur la clé, complexité des valeurs à gérer reportée dans les programmes

24

Modèle colonne (1/4)



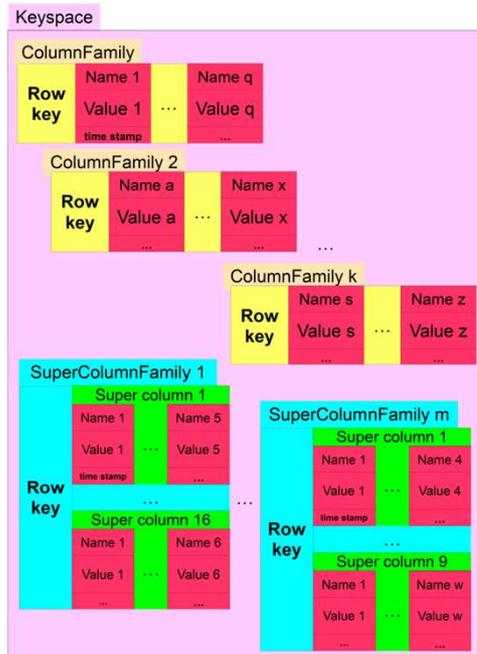
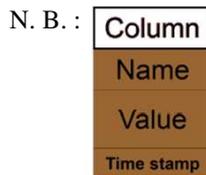
- Définition
 - Données stockées en colonnes
 - La colonne est l'entité de base représentant un champ de donnée, chaque colonne est définie par un couple (clé, valeur) avec une estampille (pour gérer les versions et les conflits)
 - Une super-colonne est une colonne contenant d'autres colonnes
 - Une famille de colonnes regroupe plusieurs colonnes ou super-colonnes où les colonnes sont regroupées par ligne et chaque ligne est identifiée par un identifiant unique et par un nom unique
- Opérations
 - Les requêtes doivent être prédéfinies en fonction de l'organisation en colonnes (et super-colonnes et familles de colonnes) choisie

Cf. smile.fr/Smile-france/Livres-blancs/Culture-du-web/Nosql

25

Modèle colonne (2/4)

Schéma des données (keyspace) d'une application



Cf. en.wikipedia.org/wiki/Keyspace (distributed data store)

Modèle colonne (3/4)

- Utilisation

Analyse de données, traitement analytique en ligne (*OnLine Analytical Processing (OLAP)*), exploration de données (*data mining*), entrepôt de données (*data warehouse*), gestion de données semi-structurées, jeux de données scientifiques, génomique fonctionnelle, journalisation d'événements et de compteurs, analyses de clientèle et recommandation, stockage de listes (messages, *posts*, commentaires, ...), traitements massifs

Ex. : Netflix (*logging* et analyse de sa clientèle), eBay Inc. (optimisation de la recherche), Adobe Systems Incorporated (traitement de données structurées et d'informatique décisionnelle (*Business Intelligence (BI)*)), sociétés de TV (connaissance de leur audience et gestion du vote des spectateurs)

- Logiciels

HBase (ou BigTable), Cassandra, SimpleDB

27

Modèle colonne (4/4)

- Critique

- ☺ bonne mise à l'échelle horizontale, efficace avec l'indexation sur les colonnes et pour des requêtes temps-réel connues à l'avance, supporte des données tabulaires à schéma variable et des données semi-structurées (facile d'ajouter/fusionner des colonnes et d'ajouter une colonne/super-colonne à n'importe quelle ligne d'une colonne/super-colonne/super-colonne), nombre de colonnes dynamique (variable d'un enregistrement à un autre permettant d'éviter les indéterminations)
- ☹ ne supporte pas les données structurées complexes ou interconnectées, maintenance nécessaire pour la modification de structure en colonne, ajout de ligne coûteux, requêtes doivent être pré-écrites

28

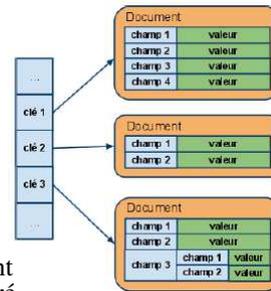
Modèle document (1/2)

- Définition

- BD = collection de documents
Modèle clé-valeur où la valeur est un document (lisible par un humain) au format semi-structuré hiérarchique (*XML*, *YAML*, *JSON* ou *BSON*, etc.)
- Document (structure arborescente) = collection de couples (clé,valeur)
Valeur de type simple ou composée de plusieurs couples (clé,valeur)

- Opérations

- Les opérations *CRUD* du modèle clé-valeur
- Souvent interface *HTTP REST* disponible
- Requêtage (API ou langage) possible sur les valeurs des documents



Cf. smile.fr/Smile-france/Livres-blancs/Culture-du-web/Nosql

29

Modèle document (2/2)

- Utilisation

Outils de gestion de contenu (*Content Management System (CMS)*), catalogues de produits, web analytique, analyse temps-réel, enregistrement d'événements, stockage de profils utilisateurs, systèmes d'exploitation, gestion de données semi-structurées

- Logiciels

CouchDB, RavenDB, MongoDB, Terrastore

- Critique

- ☺ performances élevées, bonne mise à l'échelle, modèle simple augmenté de la richesse des documents semi-structurés, expressivité des requêtes, schéma de BD évolutif, efficace pour les interrogations par clé
- ☹ peut être limité pour les interrogations par le contenu des documents, limité aux données hiérarchiques, inadapté pour les données interconnectées, baisse des performances pour de grandes requêtes

30

Resource Description Framework (RDF)

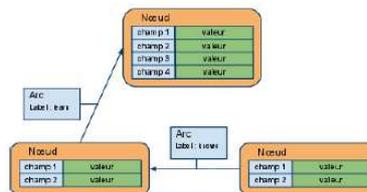
- Modèle de graphe destiné à décrire de façon formelle les ressources web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions
- Langage de base du web sémantique
- Document structuré en un ensemble de triplets (sujet,prédicat,obj)et où :
 - le sujet représente la ressource à décrire
 - le prédicat représente un type de propriété applicable à cette ressource
 - l'objet représente une donnée ou une autre ressource : c'est la valeur de la propriété

Multigraphe orienté étiqueté = ensemble de 

- *SPARQL (SPARQL Protocol and RDF Query Language)*
Langage de requête et protocole qui permet de rechercher/ajouter/modifier/supprimer des données RDF disponibles à travers Internet

31

Modèle graphe (1/2)



- Définition
Gestion d'un graphe (a priori orienté) c.-à-d. la modélisation, le stockage et la manipulation de données complexes liées par des relations non-triviales ou variables
- Mécanisme
 - Moteur de stockage pour les objets (qui se présentent sous la forme d'une base documentaire, chaque entité de cette base étant un nœud)
 - Description des arcs (relations entre les objets) disposant de propriétés (nom, date, ...)
- Opérations
 - *SPARQL* pour les SGBD *NoSQL* Graphe *RDF*
 - *API* et langages spécialisés de programmation et de requêtes sur les graphes

Cf. smile.fr/Smile-france/Livres-blancs/Culture-du-web/NoSQL

32

Modèle graphe (2/2)

- Utilisation

Moteurs de recommandation, informatique décisionnelle, web sémantique, internet des objets (*internet of things (IoT)*), sciences de la vie et calcul scientifique (bioinformatique, ...), données géospatiales, données liées, données hiérarchiques (catalogue des produits, généalogie, ...), réseaux sociaux, réseaux de transport, services de routage et d'expédition, services financiers (chaîne de financement, dépendances, gestion des risques, détection des fraudes, ...), données ouvertes (*open data*)

- Logiciels

Neo4J, OrientDB

- Critique

- ☺ modèle riche et évolutif bien adapté aux situations où il faut modéliser beaucoup de relations, nombreux langages et *API* bien établis et performants
- ☹ répartition des données peut être problématique pour de gros volumes de données, fragmentation (*sharding*)

33

Bibliographie

Webographie

Bibliographie



R. Bruchez, Les bases de données NoSQL et le big data : comprendre et mettre en œuvre, *Eyrolles*, 2015 (2^e édition)



V. Gaurav, Getting started with NoSQL your guide to the world and technology of NoSQL, *Packt Publishing*, 2013



P. Lacomme, S. Aridhi, R. Phan, Bases de données NoSQL et Big Data : Concevoir des bases de données pour le Big Data, Cours et travaux pratiques, *ellipses ~ Technosup*, 2014



A. Ploetz, D. Kandhare, S. Kadambi, X. (B.) Wu, Seven NoSQL Databases in a Week: get up and running with the fundamentals and functionalities of seven of the most popular NoSQL databases, *Birmingham: Packt Publishing*, 2018



P. Raj, G. C. Deka, A Deep Dive into NoSQL Databases: The Use Cases and Applications, *Academic Press*, 2018



S. Tiwari, Professional NoSQL, *John Wiley & Sons*, 2011

35

Webographie (1/2)

- isis.org/epinasseb/Supports/BD/BD_NOSQL-4p.pdf, /4/2013, 76 p., B. Espinasse
- smile.fr/Smile-france/Livres-blancs/Culture-du-web/Nosql, 17/10/2011, 55 p., A. Foucret
- lirmm.fr/~mougenot/Enseignement/GMIN332/Cours/gmin332_c6_14.pdf, 2014, 73 p., I. Mougenot
- perso.univ-rennes1.fr/virginie.sans/BDO/cours/BDO_CM4.pdf, 38 p., V. Sans
- www-lisic.univ-littoral.fr/~verel/TEACHING/15-16/data-science/cours06-nosql.pdf, 50 p., S. Verel

36

Webographie (2/2)

- db-engines.com/en/, Knowledge Base of Relational and NoSQL Database Management Systems
- fr.wikipedia.org/wiki/NoSQL
en.wikipedia.org/wiki/NoSQL
- nosql-database.org, Your Ultimate Guide to the Non-Relational Universe!
- newtech.about.com/od/databasemanagement/a/Nosql.htm, NoSQL: An Overview of NoSQL Databases, T. Perdue
- w3resource.com/mongodb/nosql.php, NoSQL, 7/2015
- nosql.developez.com

DB-ENGINES



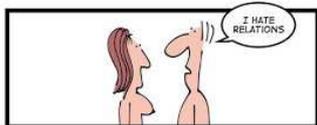
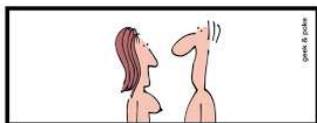
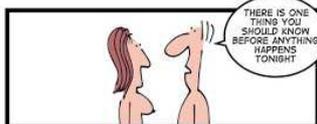
NO★SQL

about tech

w3resource

Developpez.com
Club des développeurs et IT pro

The Hard Life of a NoSQL Coder



Part 1: The Outing

HOW TO WRITE A CV



Leverage the NoSQL boom

joyreactor.com



Cf.

geekandpoke.typepad.com/geekandpoke/2011/12/the-hard-life-of-a-nosql-coder.html

joyreactor.com/tag/nosql

strozzi.it/cgi-bin/CSA/tw7/1/en_US/nosql

38