

bases de données
avancées
SASUGG2

Les bases non relationnelles: NoSQL

Présenté par:

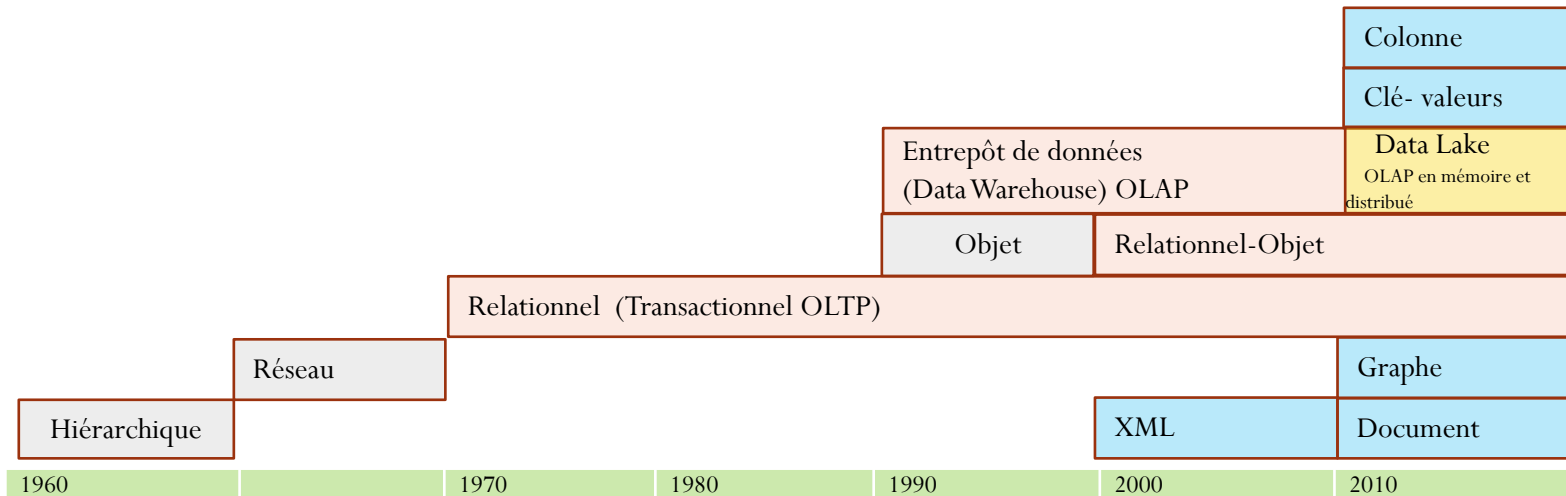
Amal HALFAOUI (Epse GHERNAOUT)

Amal.halfaoui@univ-tlemcen.dz

amal.halfaoui@gmail.com

Historique des bases de données

- Evolution des Types SGBD

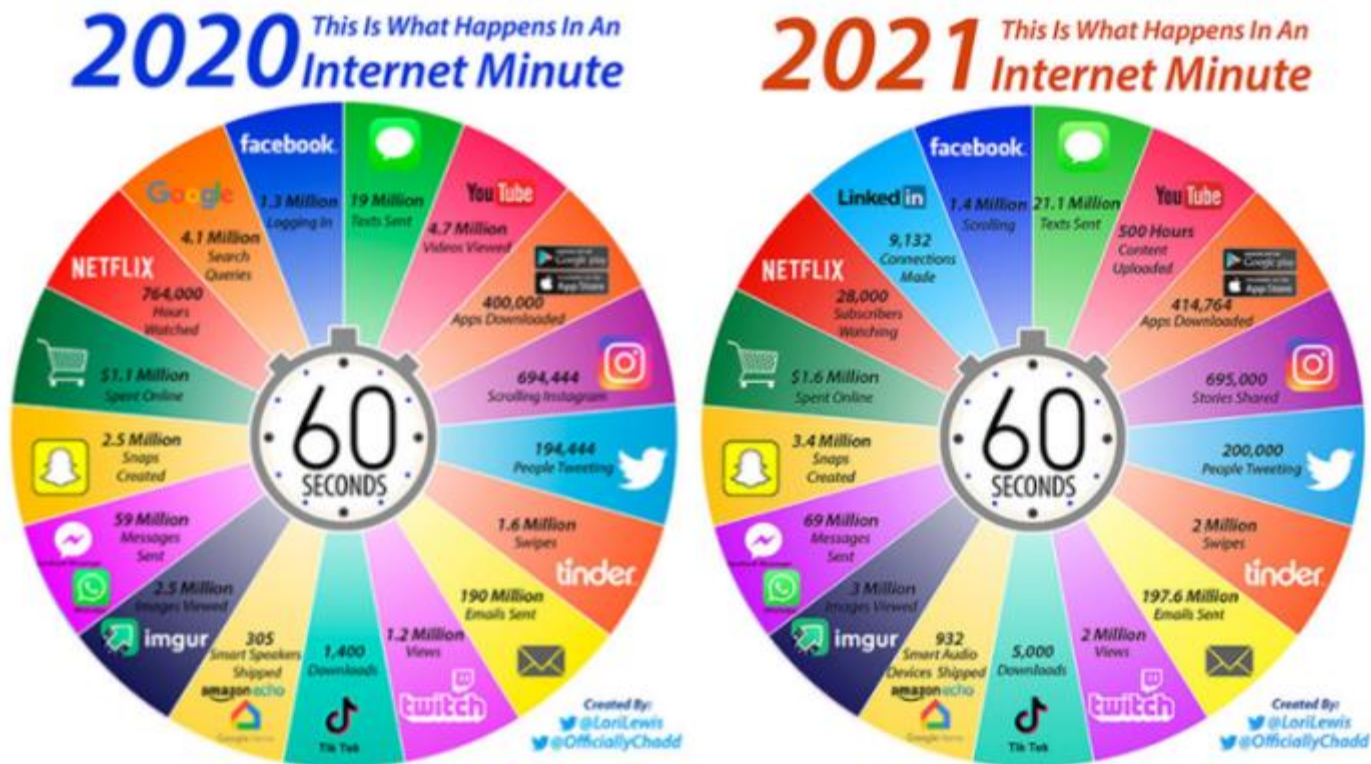


- Certains modèles n'ont pas réussi à s'imposer : Hiérarchique, réseaux, objets
- Dominance forte du modèle relationnel grâce à un modèle théorique puissant et simple: **schéma**, **normalisation** et **transaction**
- Nouveaux modèles (**NoSQL**) qui ne reposent plus sur le modèle relationnel (retour aux modèles hiérarchique et réseau)

Au delà des bases de données relationnelles: faire évoluer ou changer le modèle relationnel !

1. Problème de l'agrégat: le relationnel est peu performant pour les agrégats → Problème posé par le décisionnel (naissance des [entrepôts de données](#)).
2. Mal adaptés pour les structures de données complexes: données géographiques et multimédias (naissance des [bases Relationnel objet et XML](#)).
3. Problème de la distribution et big data: Passage du serveur central (*main frame*) à *des grappes de machines modestes* :
 - pour gérer l'explosion des données :un ensemble immense de données hétérogènes: images, données enregistrées par des capteurs, documents Json, réseaux sociaux (flux Twitter et autres données) .
 - à cause de l'évolution du *hardware*
naissance des [bases NoSQL](#)

Croissance de données en exponentiel



Source: <https://ediscoverytoday.com/2021/04/16/here-is-your-2021-internet-minute-infographic-ediscovery-trends/>

3V: Volume, variété et vélocité des données

Volume

Stocker de
l'information en
grande quantité



Vélocité

Accéder rapidement
aux données, en
temps réel

Variété

Enregistrer des
types de données
différentes

Contexte dans lequel les grands systèmes ont été construits

- **Google (big table)**

données en tables 2D, mais les lignes ont des colonnes différentes Service de base de données NoSQL Big Data de Google. Cette base de données est utilisée par beaucoup de services Google, tels que la recherche, Analytics, Maps et Gmail. → schéma orienté colonnes

- **Amazon : (DynamoDB) e-commerce** → schéma orienté clé valeur

- **Facebook : (Cassandra puis HBase)** fonction de recherche dans la boîte de réception → schéma orienté colonne

- **Yahoo : (Pnuts)** construit pour stocker les données des utilisateurs qui peuvent être lues ou écrites sur toutes les page Web, stocker les listes de données pour les pages de shopping de Yahoo, stocker les données pour servir ses applications de réseau social → schéma orienté clé valeur

- **LinkedIn : (Voldemort)** gérer les mises à jour en ligne à partir de diverses caractéristiques d'écritures intensives sur le site internet → schéma orienté clé valeur

No SQL : définition

- **2009 : NoSQL** été inventé en 2009 lors d'un événement sur les bases de données distribuées . Il est choisi pour intituler tous les SGBD de type non-relationnel.
- NoSQL veut dire **Not Only SQL**

Avant de commencer !

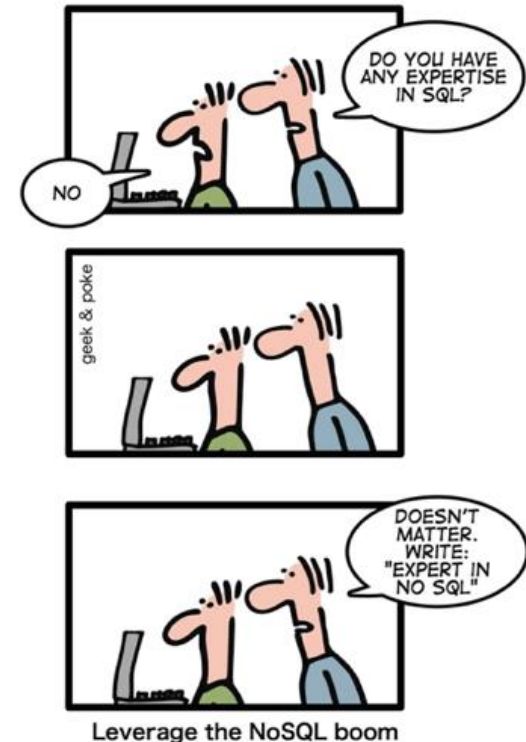
- **NoSQL ne remplace pas les SGBDR**
 - il s'agit d'un **besoin différent** ou (actuellement) **de compléments** aux SGBDR pour des besoins spécifiques et non de solutions de remplacement.
 - Il faudra impérativement comprendre
 - 1- les **besoins de votre application**
 - 2- la différences entre **les différents schémas** et **types d'usage** des bases NoSQL et SGBDR

- **Attention :**

« Si le seul outil que vous avez est un marteau, vous tendez à voir tout problème comme un clou »

Abraham Maslow (*The Psychology of Science*, 1966).

HOW TO WRITE A CV



SGBDR vs Distribution

- **Forces des SGBDR**

- Jointures entre les tables
- Langage d'interrogation normalisé et riche
- Contraintes d'intégrité solides

- **Limites dans le contexte distribué :**

Comment distribuer/partitionner les données ?

- Liens entre entités → Même serveur
- Mais plus on a de liens, plus le placement des données est complexe

SGBDR vs Distribution

Propriétés **ACID** pour les transactions

- **Atomicité** : une transaction s'effectue entièrement ou pas du tout
- **Cohérence** : le contenu d'une base doit être cohérent au début et à la fin d'une transaction
- **Isolation** : les modifications d'une transaction ne sont visibles/modifiables que quand celle-ci a validé
- **Durabilité** : une fois la transaction validée, l'état de la base est permanent (non affecté par les pannes ou autre)

Systèmes distribués : modèle **BASE**

- **Basically Available** : le système doit toujours être accessible quelle que soit la charge de la base de données (données/requêtes), le système garantit un taux de disponibilité de la donnée
- **Soft-state** : l'état de la base de données n'est pas garanti à un instant t.
- **Eventually consistent** : la cohérence des données à un instant t n'est pas primordiale. À terme, la base atteindra un état cohérent, et le système devient à un moment consistant,

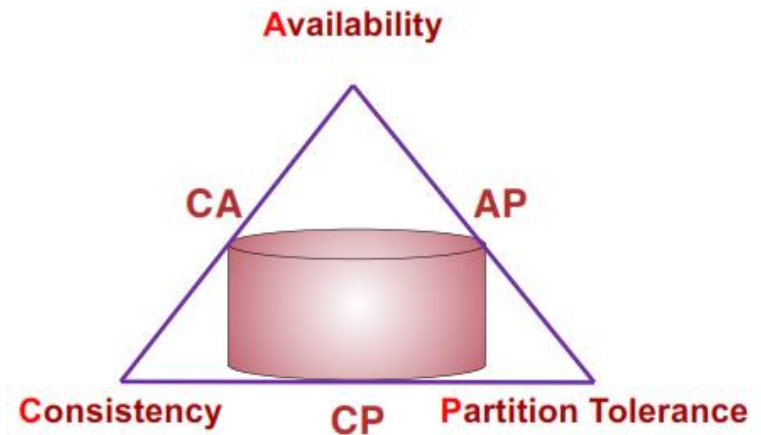
Théorème CAP (Brewer (2000))

- **3 propriétés** fondamentales pour les systèmes distribués

1. Consistency: *Tous les serveurs voient la même donnée (valeur) en même temps (ou Cohérence)*

2. Availability: *Si un serveur tombe en panne, les données restent disponibles (disponibilité)*

3. Partition Tolerance: *Le système même partitionné doit répondre correctement à toute requête (sauf en cas de panne réseau)*



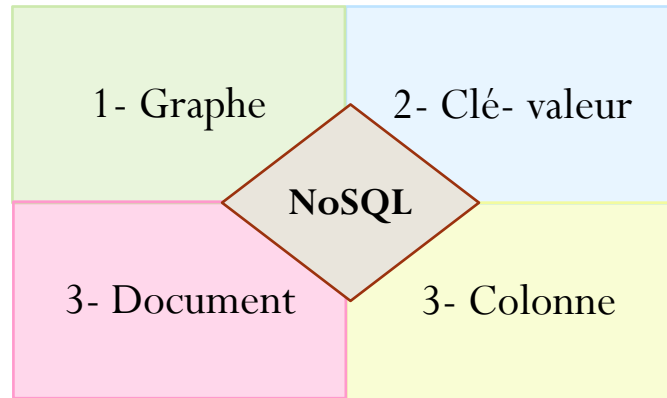
- **Théorème** : “Dans un système distribué, il est impossible que ces 3 propriétés co-existent, vous devez choisir 2 d’entre elles”.

SGBDR vs NoSQL

Relationnel	NoSQL
Schéma défini au départ	Schémaless
Structure rigide	Structure dynamique
Gestion des valeurs Null	Pas de valeurs Null
Transactions ACID	Transactions BASE
Scalabilité Verticale	Scalabilité Horizontale sans limites
Insiste sur la Cohérence des données	Insiste sur la Disponibilité et la Performance
Mauvaise gestion de gros volumes de données	Conçu pour les gros volumes de données
Langage SQL	Différents Langages selon le modèle utilisé

Familles des bases NoSQL

1-Classement par schéma de données



2- Classement par Usage

- Performance
- Structures spécifique
- Structure souples
- volumétrie

Exemple: base livres (relationnelle)

Livre



ISBN	TITRE	AUTEUR
2081399253	l'Alchimiste	3
2212320434	SQL pour oracle	2
208134761X	Maktub	3
2212141556	Les bases de données NoSQL et le big data	1

Relation écrire (1,n)

LIVRES.LIVRE		
P *	ISBN	VARCHAR2 (20 BYTE)
	TITRE	VARCHAR2 (20 BYTE)
F	AUTEUR	NUMBER
	PK_LIVRE (ISBN)	
	FK_AUTEUR (AUTEUR)	
	PK_LIVRE (ISBN)	

Auteur

ID	NOM	PRENOM	FONCTION	INTERRET
1	Bruchez	Rudi	Consultant	Bases de données
2	Soutou	Christian	Maitre de conférences	Modélisation, Bases de données
3	Coelho	Paulo	(null)	(null)

LIVRES.AUTEUR	
P * ID	NUMBER
NOM	VARCHAR2 (20 BYTE)
PRENOM	VARCHAR2 (20 BYTE)
FONCTION	VARCHAR2 (20 BYTE)
INTERRET	VARCHAR2 (20 BYTE)
 PK_AUTEUR (ID)	
 PK_AUTEUR (ID)	

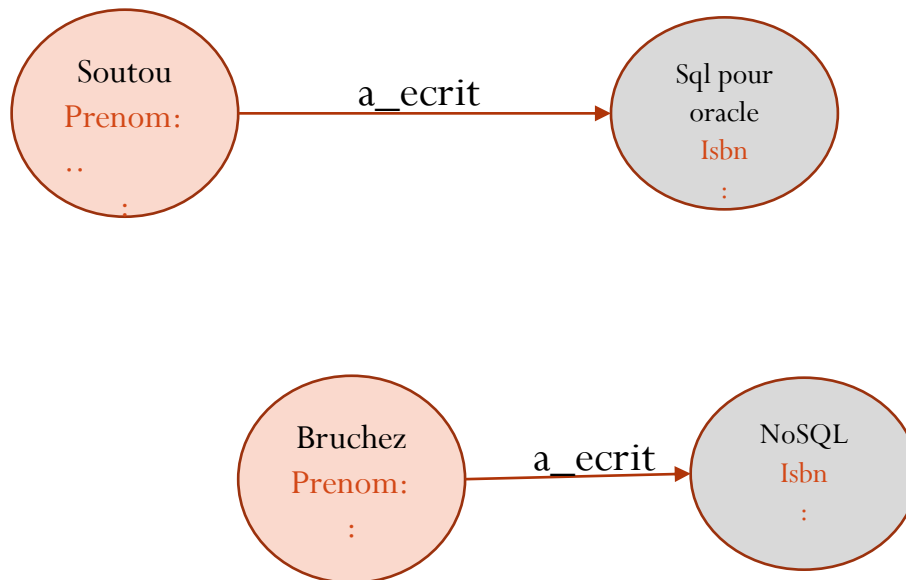
1 - Schéma Graphe

Schémas NoSQL : Orienté graphe

Principe

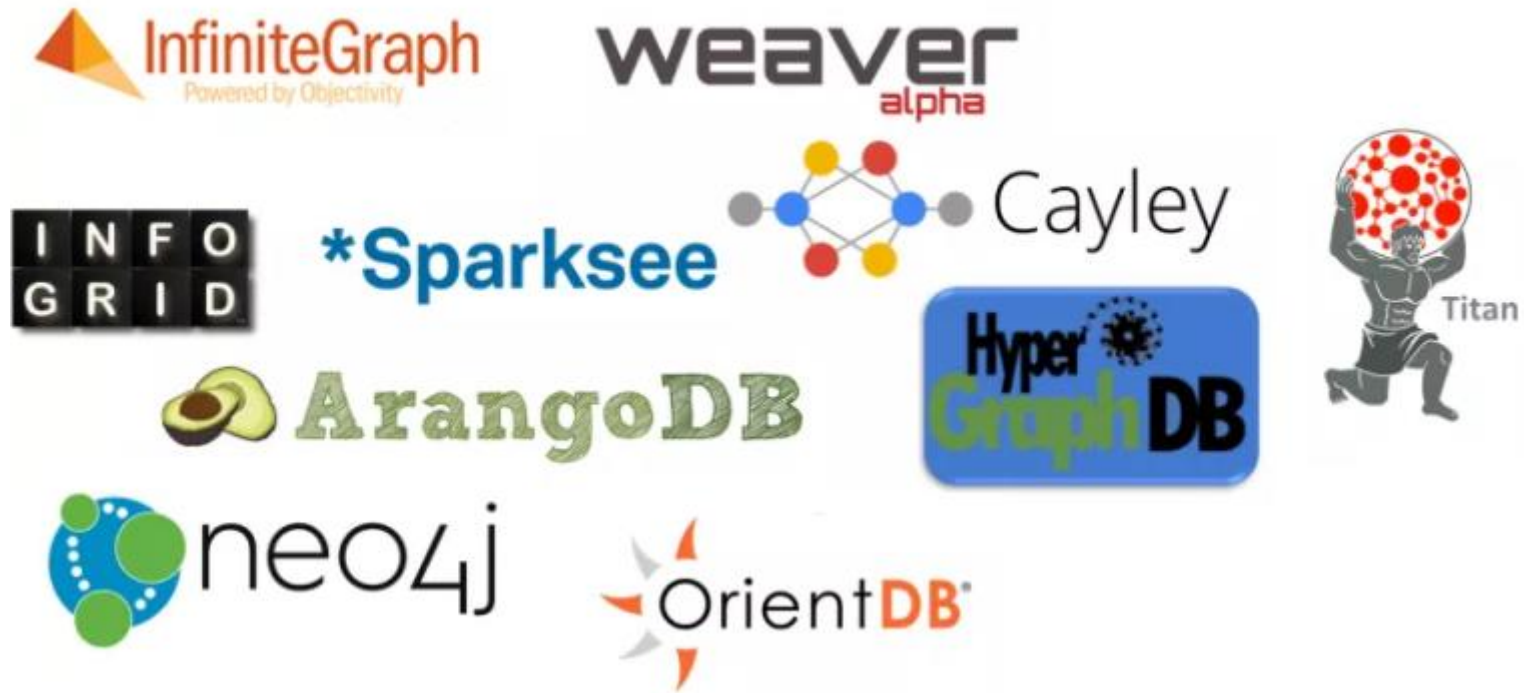
se basant sur la théorie des graphes:

- Des **nœuds** qui ont chacun leur propre structure
- Des **relations** entre les nœuds
- Des **propriétés** (de nœuds ou de relations)



Schémas NoSQL : Orienté graphe

- Quelques solutions



Schémas NoSQL : Orienté graphe

Exemple de Neo4J

- **Téléchargement:**

<https://neo4j.com/download/other-releases/#releases>, la version community actuelle est 4.X

Dézipper le fichier et lancer dans le shell .. \neo4j\bin>neo4j console

- **Lancement :** <http://localhost:7474>
- **Quelques commandes :** Le langage de requête utilisé par Neo4j est: **Cypher**

Création des nœud auteurs

```
create (Soutou:personne {nom:"Soutou",prenom:"Christian"})
create (SQL_Oracle:livre {titre:"Sql pour oracle", isbn:"2212320434"})
```

Création de la relation écrire

```
create (Soutou)-[:ecrit]->(SQL_Oracle)
return Soutou,SQL_Oracle
```

Retourner un nœud avec une propriété

```
match (n {isbn:'2212320434'}) return n
```

Schémas NoSQL : Orienté graphe

- **Forces**

- Permet d'appliquer les algorithmes de théorie des graphes et la mise en place de visualisation de graphes nativement
- Beaucoup plus rapides que les autres systèmes de stockage pour manipuler les données fortement connectées → Adaptées aux objets complexes organisés en réseaux, aux données présentant des dépendances fortes

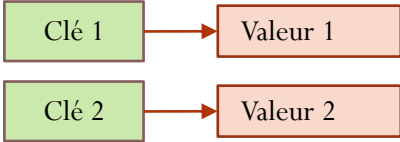
- **Quand utiliser une base NoSQL orientée graphe?**

- Cas d'utilisation typique : Les **réseaux sociaux** où l'aspect graphe prend tout son sens à des fins de calcul sur les graphes (recommandations, plus courts chemins, atteignabilité,...)
- Calculs sur les réseaux des SIG: données géographiques cartes, réseaux routiers, électricité, ...
- Web social (linked data)

2- Schéma Clé valeurs

Schémas NoSQL : Orienté clé-valeur

Principe:

- Une grosse HashMap :
un simple tableau associatif unidimensionnel avec des millions voire des milliards d'entrées
- Une entrée est un couple Clé+Valeur:
 - Pas de schéma pour la valeur (chaîne, objet, entier, binaires...)
 - Pas d'expressivité d'interrogation (pré/post traitement pour manipuler concrètement les données): les opérations sont atomiques :
 - Création de la paire (cle,valeur) :
 - Lecture: lire un objet en connaissant sa clé
 - Modification: mettre à jour l'objet associé à la clé
 - Suppression

Schémas NoSQL : Orienté clé-valeur

- **Quelques Solutions**

Redis, Memcached, Amazon DynamoDB, Riak, Ehcache Hazelcast (utilisé dans le site LinkedIn), OrientDB, Oracle NoSQL, Berkeley DB, etc.



- **Téléchargements pour la solution Redis**

- Le serveur Redis pour windows à partir : <https://redis.io/download>
- <https://github.com/uglide/RedisDesktopManager/releases/tag/0.9.0-3>
- GUI client: Redis Desktop à partir: <https://redisdesktop.com/download>
- Tutoriel complet à partir : http://fc.isima.fr/~lacomme/NoSQL/chapitre_gratuit/chapitre3_apres_fusion.pdf

Schémas NoSQL : Orienté clé-valeur

- **Exemple de la base Redis**

Les commandes selon les différents types de stockage:

- **String:**

La valeur peut-être un string de tout type (json, valeur d'une image jpeg), mais aussi un entier.

Les principales commandes associées sont **SET, GET, INCR, DECR, et GETSET**.

SET cle valeur
GET cle

Exp:

SET nom halfaoui
SET compteur 10
INCR compteur
GET compteur
DEL compteur

- **Hash**

permet de stocker dans un même enregistrement plusieurs couples de clef/valeurs.

Les principales commandes sont **HSET, HGET, HLEN, HGETALL** pour obtenir tous les couples clef-valeur, **HINCRBY** pour incrémenter un compteur dans la hash, **HKEYS** et **HVALS** pour obtenir toutes les clefs ou valeurs et **HDEL** pour la suppression.

Schémas NoSQL : Orienté clé-valeur

- **Exemple de la base Redis**

Les commandes selon les différents types de stockage:

- **Sets:**

des collections d'objets non ordonnées. Les commandes commencent toutes avec un **S** comme Set,

Les commandes

SADD pour ajouter une valeur à un set,

SCARD pour obtenir la taille (cardinalité) d'un set,

SINTER, SUNION, SDIFF qui permettent respectivement d'obtenir l'intersection, l'union et la différences entre 2 sets. Ces commandes existent en version "**STORE**" ; ainsi **SINTERSTORE** permet de stocker dans un nouveau set l'intersection de 2 autres.

DEL suppression de l'objet

EXPIRE

- **Autre types et commandes :**

Pour voir tous les autres commandes et types de stockage allez sur <http://redis.io/commands>

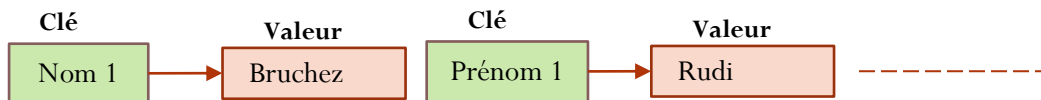
Schémas NoSQL : Orienté clé-valeur

- Exemple de modélisation

Attention : plusieurs manières de modélisation:

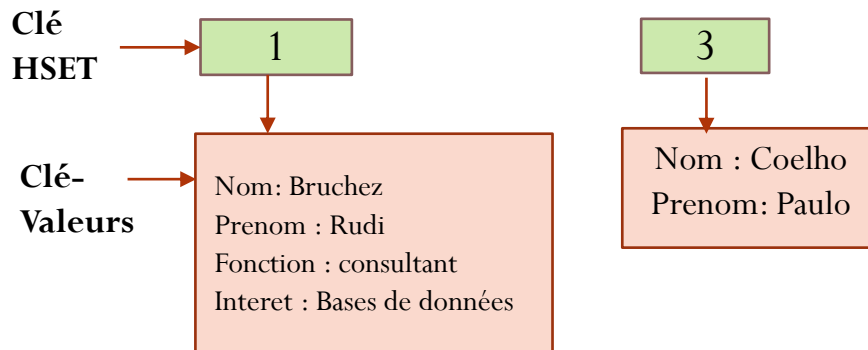


1- ensemble de clé valeurs uniques par l'association de clé aux attributs: **Set** Nom1 Bruchez, **Set** prénom1 Rudi, Set fonction1 consultant, Set interert1 bdd (**Pas intéressante**)



2- HSET associé aux clés: HSET 1 Nom Bruchez, HSET 1 Prenom Rudi, HSET 1 Fonction: Consultant

Attention: précéder la clé par le nom de la table HSET auteur1 Nom Bruchez,.. Car la clé de la HSET est unique est ne sera pas utilisable pour la clé 1 de la table Livre.



3- SET pour les valeurs de la table et un HSET pour les association clé auteur, nom auteur (pour recherche par nom et retrouver la clé) voire le pdf: http://fc.isima.fr/~lacomme/NoSQL/chapitre_gratuit/chapitre3_apres_fusion.pdf

Schémas NoSQL : Orienté clé-valeur

- **Forces**

- Leur simplicité, scalabilité, disponibilité
- Très bonnes performances dans la mesure où les lectures et écritures sont réduites à un accès disque simple

- **Faiblesses ?**

- Pas de requêtes sur le contenu des objets stockés
- Non-conservation des relations entre les objets (elles ne sont pas faites pour les contextes où la modélisation métier est complexe)

- **Quand utiliser une base NoSQL orientée clé valeur?**

Elles sont beaucoup utilisées en tant que cache , pour conserver les sessions d'un site web et plus généralement pour toutes les données que l'on ne souhaite conserver que pendant un certain laps de temps, pouvant aller de quelques secondes à quelques jours.

- **Exemple :**

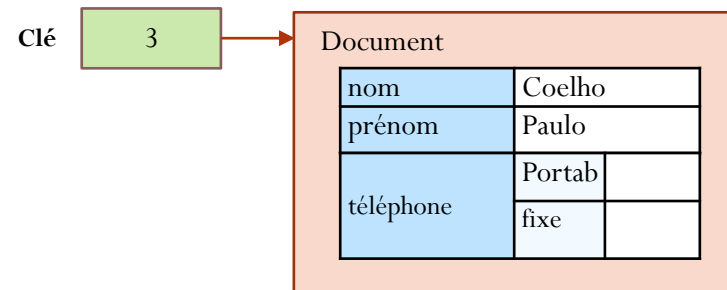
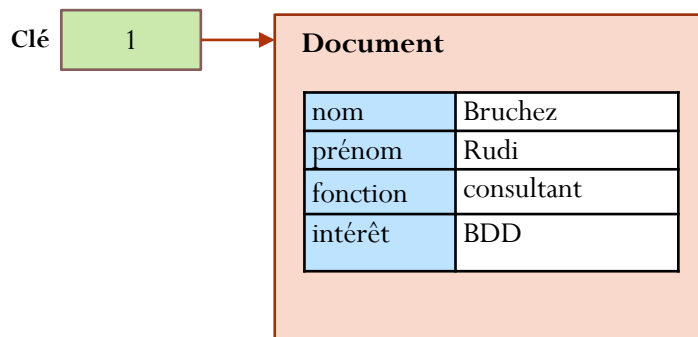
- gestion de panier d'achat (Amazon)
- collecte d'événements (jeu en ligne)

3- Schéma Document

Schémas NoSQL : Orienté document

Principe:

- Basé sur le modèle clé-valeurs → par contre, **valeur** = **document**
- Le document a une structure arborescente → les données sont semi structurées (**JSON** ou **XML**)
- Les documents sont structurés mais **aucune définition de structure préalable** n'est nécessaire.
 - Document composé de clés/valeurs
 - le types de donnée peut être simples (Int, String, Date) ou sous forme d'une liste de données
 - les données peuvent être imbriqués → schéma arborescent



Schémas NoSQL : Orienté document

Quelques Solutions:

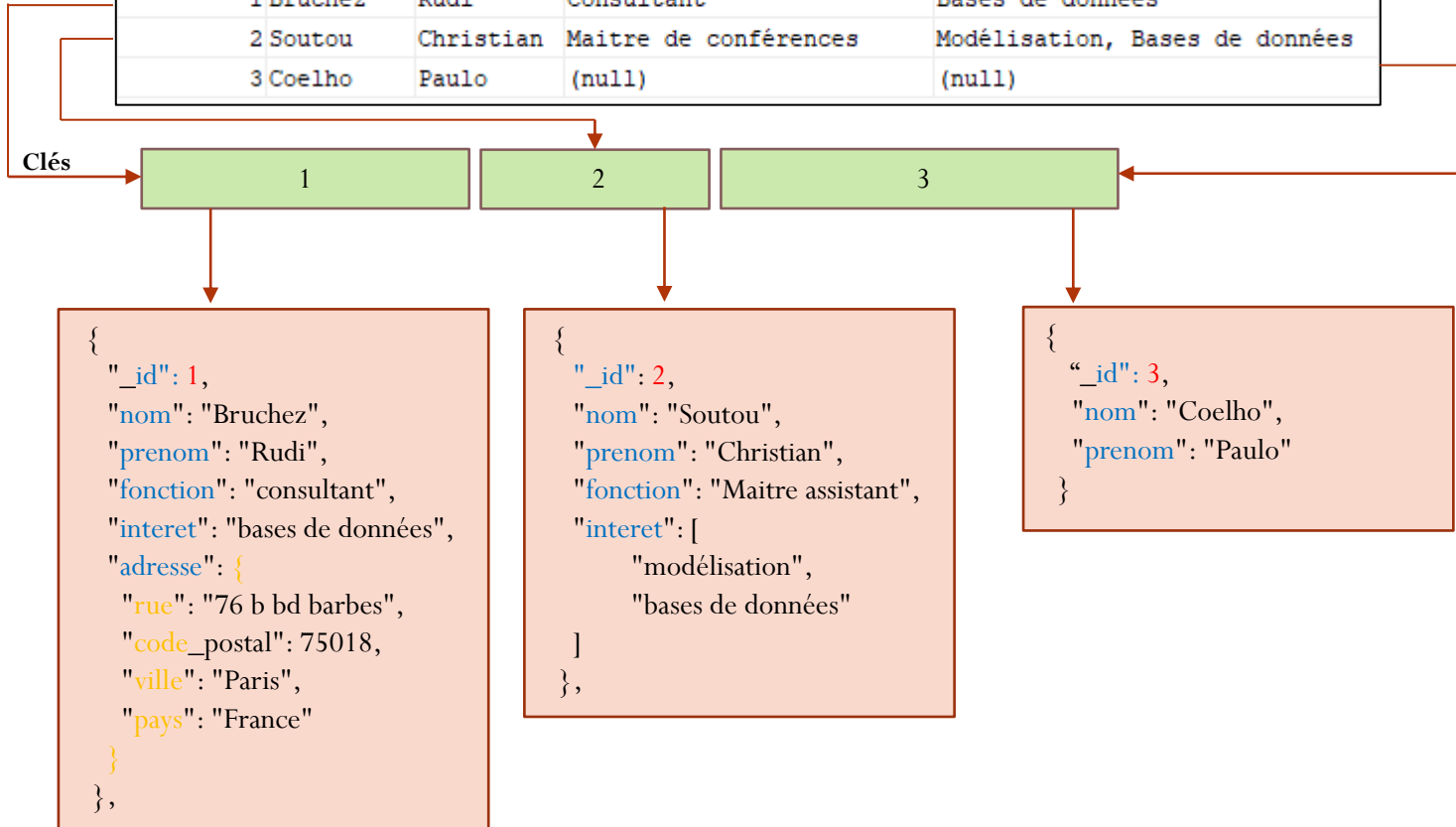


Schémas NoSQL : Orienté document

Exemple:

Auteur:

ID	NOM	PRENOM	FONCTION	INTERRET
1	Bruchez	Rudi	Consultant	Bases de données
2	Soutou	Christian	Maitre de conférences	Modélisation, Bases de données
3	Coelho	Paulo	(null)	(null)



Schémas NoSQL : Orienté document

Quelques commandes: Mongodb (sera détaillé dans le prochain cours)

- Afficher toutes les bases: `show dbs`
- Basculer à la base livre si cette dernière n'est pas créée elle le sera à la première création de collection: `use livre`
- Insérer un document dans la collection auteur:
`db.auteur.insert({"_id": 3, "nom": "Coelho", "prenom": "Paulo"})`
- Retourner tous les documents de la collection:
`db.auteur.find()`
- Retourer toutes les collections: `show collections`
- Restriction: `db.auteur.find({"nom": "coelo"})`

Schémas NoSQL : Orienté document

- **Forces**

- Les documents sont structurés mais aucune définition de structure préalable n'est nécessaire
- On peut récupérer , via une seule clé, un ensemble d'informations structurées de manière hiérarchique (tout le document). Dans un environnement relationnel, cette opération nécessite plusieurs jointures qui sont très coûteuse en ressources.

- **Faiblesses**

- Elles ne sont pas très adaptées pour les données interconnectées ni pour les données non-structurées

- **Quand utiliser une base NoSQL orientée document?**

parmi les bases NoSQL les plus utilisées notamment pour la:

- gestion de contenu de bibliothèques numériques,
- gestion catalogue de produits,
- collections multimédia, etc
- Gestion des historiques d'utilisateurs (Stockage de toutes les transactions et information du client au sein d'un même document (même clef)).

4- Schéma Colonne

Schémas NoSQL : Orienté Colonne

Principe

- Les informations sont stockées colonne par colonne.
- celles qui se rapprochent le plus des bases relationnelles (principe de tables) mais beaucoup plus souples
 - **Les colonnes sont dynamiques** : Les lignes peuvent avoir des colonnes différentes avec un nombre différent
 - **pas de champ «NULL»** contrairement à une table relationnelle (pas de colonne inutile dans une ligne)
 - **L'historisation des données se fait à la valeur et non pas à la ligne** comme dans les SGBDR

Schémas NoSQL : Orienté Colonne

Exemple de modélisation:

ID	NOM	PRENOM	FONCTION	INTERET
1	Bruchez	Rudi	Consultant	Bases de données
2	Soutou	Christian	Maitre de conférences	Modélisation, Bases de données
3	Coelho	Paulo	(null)	(null)

Les informations sont passées d'un état **horizontal** à une situation **verticale**

Id	NOM
1	Bruchez
2	Soutou
3	Coelho

id	FONCTION
1	Consultant
2	Maitre de conférence

Les informations sont regroupées par colonnes

Id	INTERET
1	Bases de données
2	modélisation
2	Bases de données

row	column	value
1	nom	Bruchez
1	prenom	Rudi
1	fonction	Consultant
1	interet	Bases de données
2	nom	Soutou
2	prenom	Christian
2	fonction	Maitre de conférence
2	interet	modélisation
2	interet	Bases de données
3	nom	Coelho
3	prenom	Paulo
2	adresse	75018, Paris France

Schémas NoSQL : Orienté Colonne

Quelques solutions



HYPERTABLE INC

- Hbase : <http://hbase.apache.org/>
- Google Big Table : <https://cloud.google.com/bigtable/>
- Hypertable: <http://www.hypertable.org/>
- Apache Parquet : <https://parquet.apache.org/>

Schémas NoSQL : Orienté colonne

- **Forces**

- Flexibilité
- Temps de traitement
- Non-stockage des valeurs null

- **Faiblesses ?**

- Elles ne sont pas très adaptées pour les données interconnectées ni pour les données non-structurées

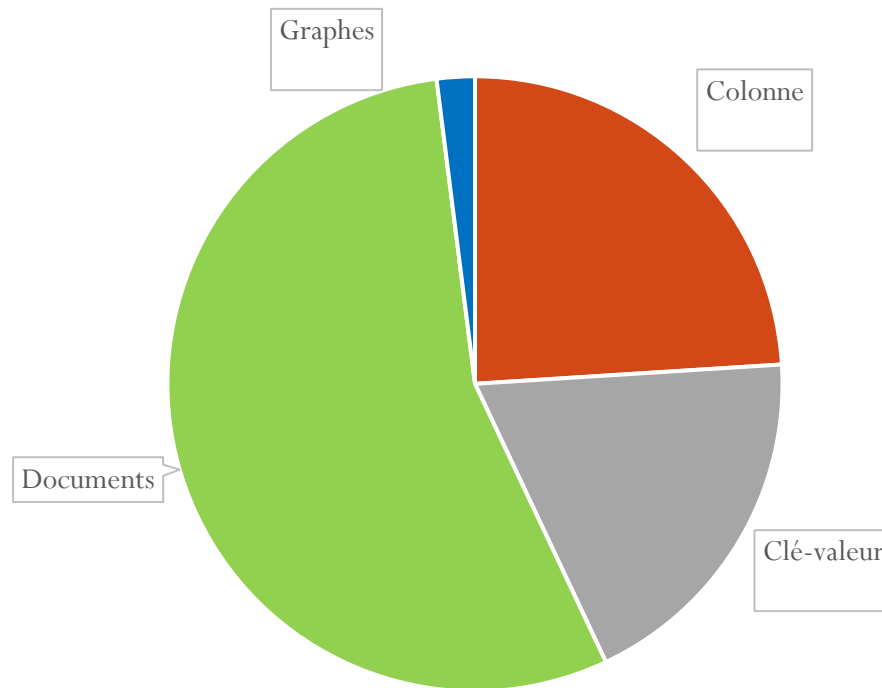
- **Quand utiliser une base NoSQL orientée colonne?**

Elles sont adaptées au stockage (one to many)

- Reporting large échelle (agrégats calculés sur une colonne)
- Comptage (vote en ligne, compteur, etc)
- Recherche de produits dans une catégorie (Ebay)
- La récupération et l'analyse de données en temps réel issues de capteurs, IoT (Internet Of Things) etc.....

Taux d'utilisation des bases NoSQL

Site de classement des SGBD <https://db-engines.com/en/ranking>



■ Colonne ■ Clé-valeur
■ Documents ■ Graphes

Classement des bases NoSQL par Usage

- Les Types d'usage des Moteurs NoSQL

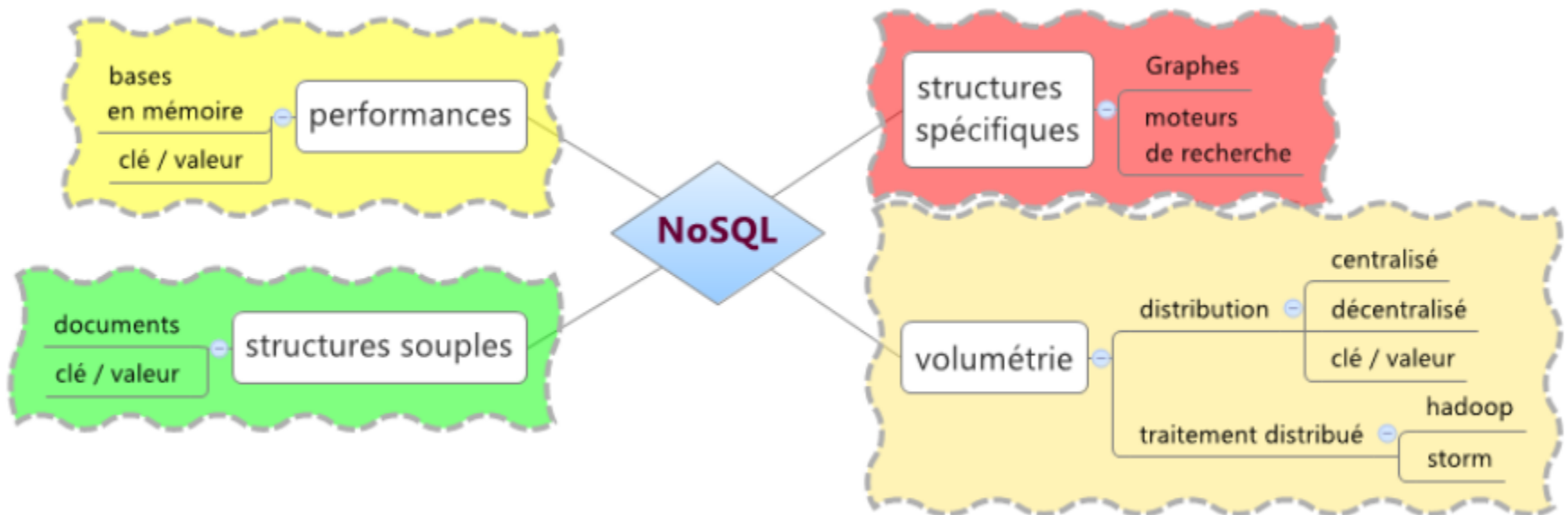


Image tirée du livre: Les bases de données NoSQL et le Big Data De l'auteur **Rudi Bruchez** (disponible dans la bibliothèque de l'université)

- Le premier chapitre du livre est téléchargeable à:

<https://www.eyrolles.com/Chapitres/9782212141559/9782212141559.pdf>

Pour résumer

- NoSQL : **Not Only SQL**

- Nouvelle approche de stockage et de gestion de données qui ne repose pas sur les tables relationnelles
- Permet le passage à l'échelle via un contexte hautement distribué
- Gestion de données complexes et hétérogènes
- Pas de schéma pour les objets (schemaless): **Attention**, actuellement le mouvement NoSQL essaye de réintégrer des fonctions de schématisation a priori, comme ce qui se fait en XML : le schéma est optionnel, mais conseillé en contexte de contrôle de cohérence.
- Pas de standards et un ensemble croissant de bases (actuellement > 255 base voir <http://nosql-database.org/>)

- **Ne remplace pas les SGBDR !!**

À utiliser dans le contexte:

- Quantité de données énorme (PétaBytes)
- Besoin de temps de réponse
- Cohérence de données faible
- Besoin de flexibilité de schéma

Pour terminer: insuffisances de NoSQL

- Non-existence de langages de requête normalisés tels que SQL
- Cohérence de données abandonnée relativement au profit de la haute disponibilité
- Sécurité des données et manque d'autres fonctionnalités supplémentaires insuffisamment développées : Les systèmes NoSQL se base sur l'application pour la protection des données
- Maturité et stabilité : Les bases relationnelles ont une longueur d'avance sur ce point. Les utilisateurs sont familiers avec leur fonctionnement et ont confiance en elles
- Manque d'architectures et d'interfaces normalisées
- Moins de documentation et d'outils disponibles