
Département de Génie Informatique et
Mathématiques Appliquées
Ecole Nationale d'Ingénieurs de Sfax

# BD NoSQL

# Chapitre1: Concepts de base

# Transactions ACID

**Atomique (Atomic)**

- Pas de modification partielle : Une transaction se fait au complet ou pas du tout

**Cohérente (Consistant)**

- Après une transaction, les données doivent être cohérentes.
- La validation des données est assurée par les contraintes d'intégrités.

**Isolées**

- Chaque transaction doit s'exécuter en isolation totale
- Aucune dépendance possible entre les transactions

**Durable**

- Après une validation, les données doivent même à la suite d'une panne d'électricité

**Les Bases de données Relationnelles sont basées sur les transactions ACID**

# SGBDR

- Données structurées, jointures faciles
- Forte consistance
- Transactionnels
- Très matures, stables, documentation  riche
- Duplications réduites, modification en une seule place

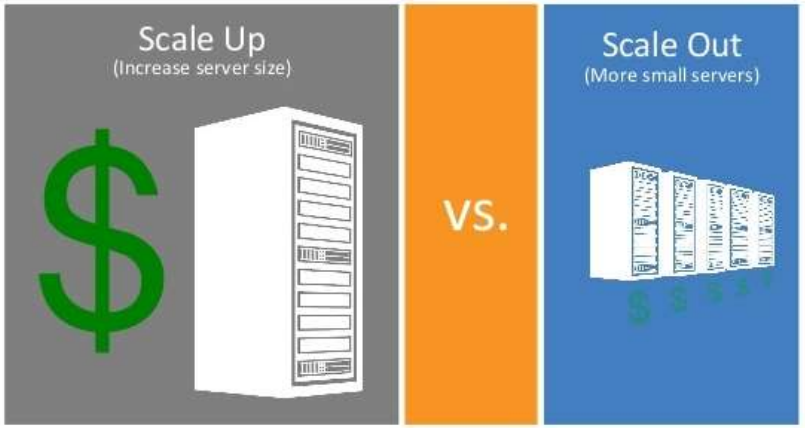
MAIS…

Avec l'avènement du Big Data, non scalables!

**Scalabilité:** capacité d'un système à maintenir ses fonctionnalités et ses performances en cas de forte demande

# SGBDR

- Scalabilité difficile:
  - Les règles d'intégrité compliquent la montée horizontale
  - Montée en charge verticale
    - Coût non linéaire
    - Atteint une limite
    - Point unique de défaillance
- Cout des transactions ACID
  - La lecture est éparpillée
  - L'écriture est lente
  - La durée d'une requête est difficile à prévoir



# NoSQL

- Pourquoi NoSQL ?
  - Licence des SGBDR très chère (Oracle, ...).
  - Le SQL a un schéma fermé.
  - Performances faibles de SQL, sur de gros volumes de données, comparées au NoSQL.
- Le NoSQL vise :
  - Gestion d'énormes quantités de données
  - Structuration faible du modèle
  - Montée en charge (scalabilité)

# Théorème CAP

| | |
|---|---|
| **Consistency (Cohérence)** | • Après la modification d'une donnée, tous les clients lisent la nouvelle valeur. |
| **Avalibility (Disponibilité)** | • Le système répond toujours aux requêtes dans un temps borné (timeout) |
| **Partition Tolerance (Tolérance aux pannes)** | • Le système continue à fonctionner si le réseau tombe en panne |

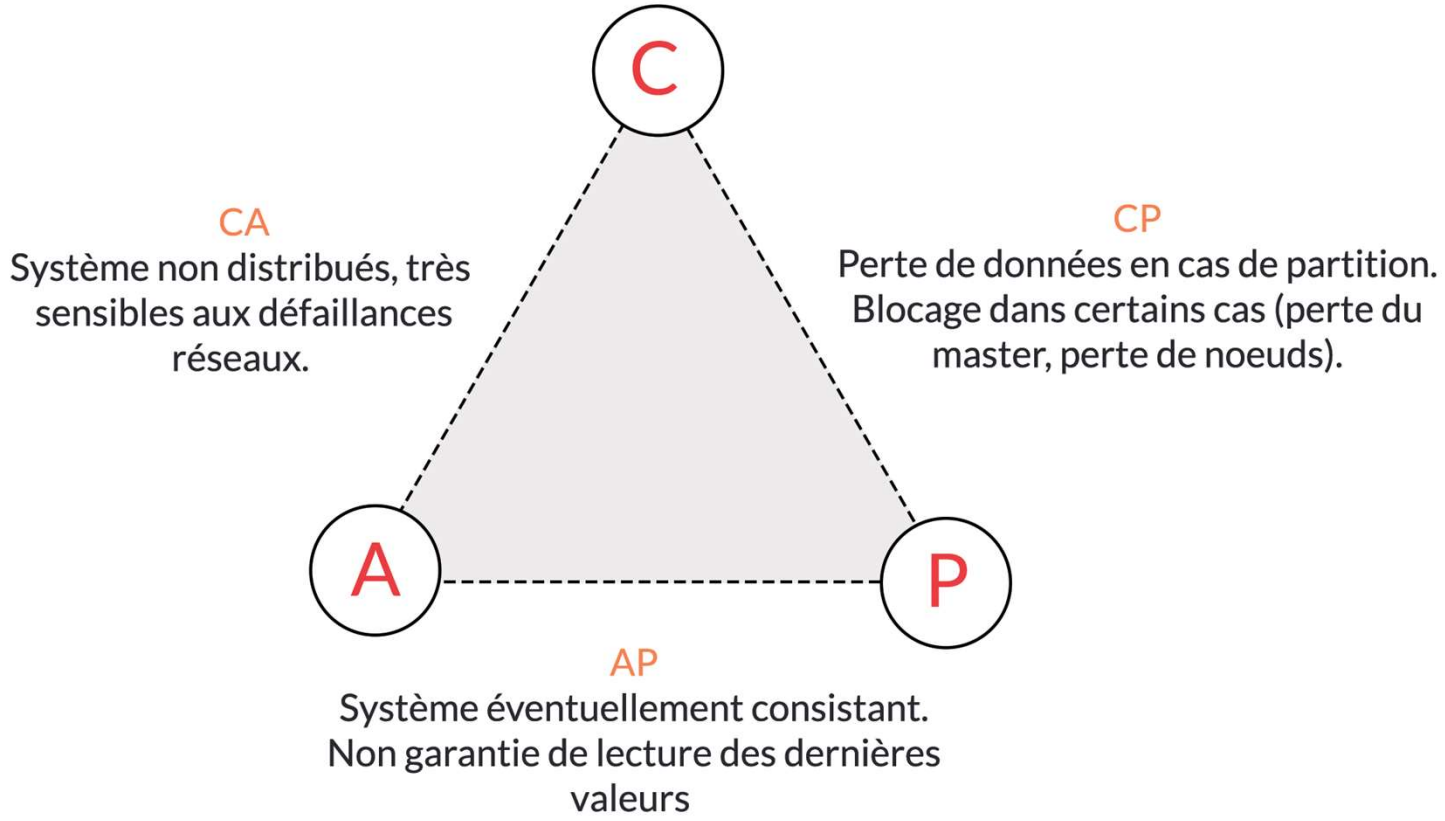
**Théorème CAP**

**« You can have at most two of these properties for any sharded-data system. » Eric A. Brewer — 19 juillet 2000**

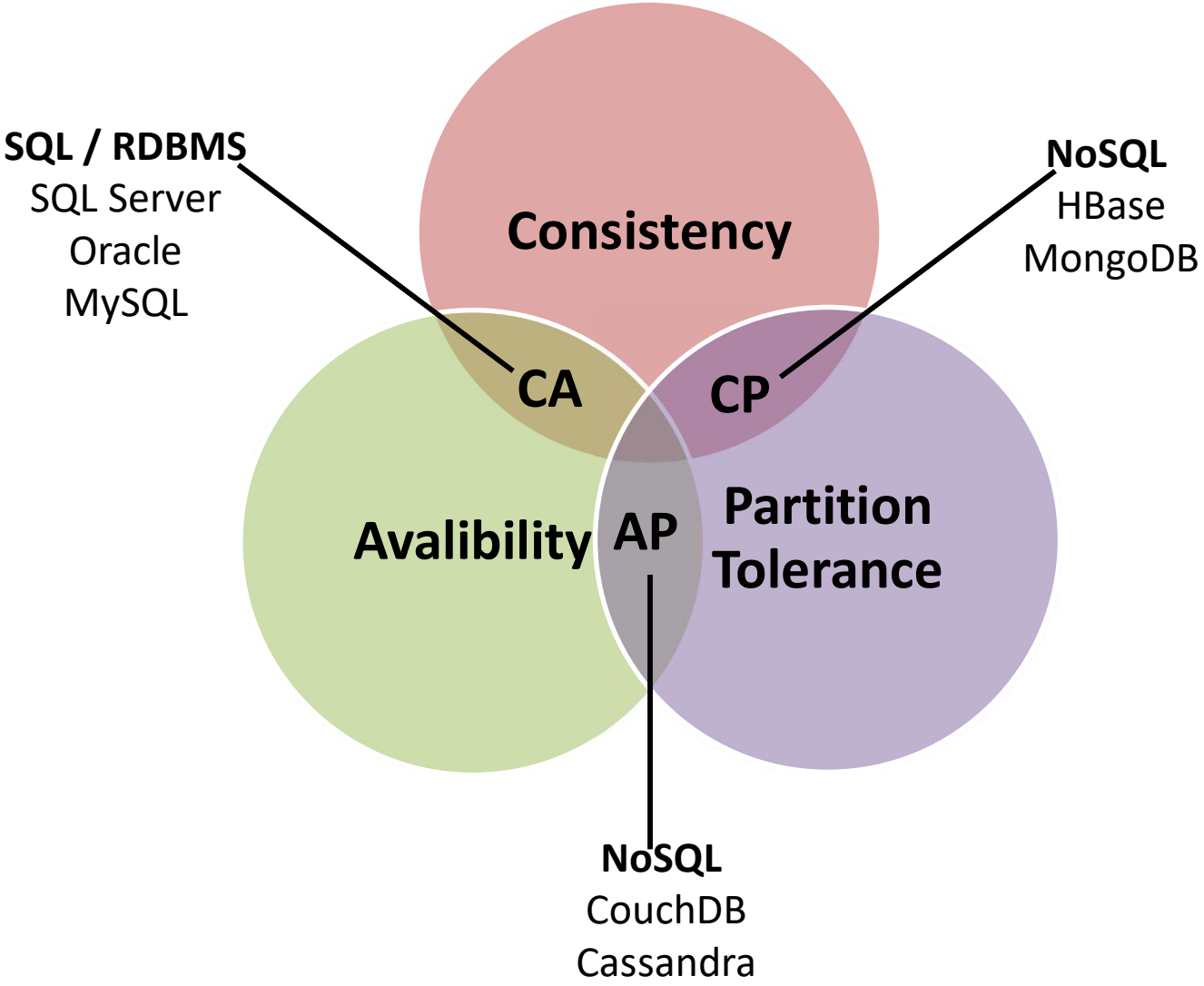
# Théorème CAP

- CAP oblige un système distribué de suivre 2 des 3 exigences.

→Toutes les bases de données NoSQL suivent les différentes combinaisons de C, A, P du théorème CAP.



# Théorème CAP



# La norme BASE

## Basically Available

- **Simplement disponible:** Le système garantit bien la disponibilité dans le même sens que celle du théorème de CAP.

## Soft state

- **Etat Souple**: Indique que l'état du système peut changer à mesure que le temps passe, et c'est sans action utilisateur.

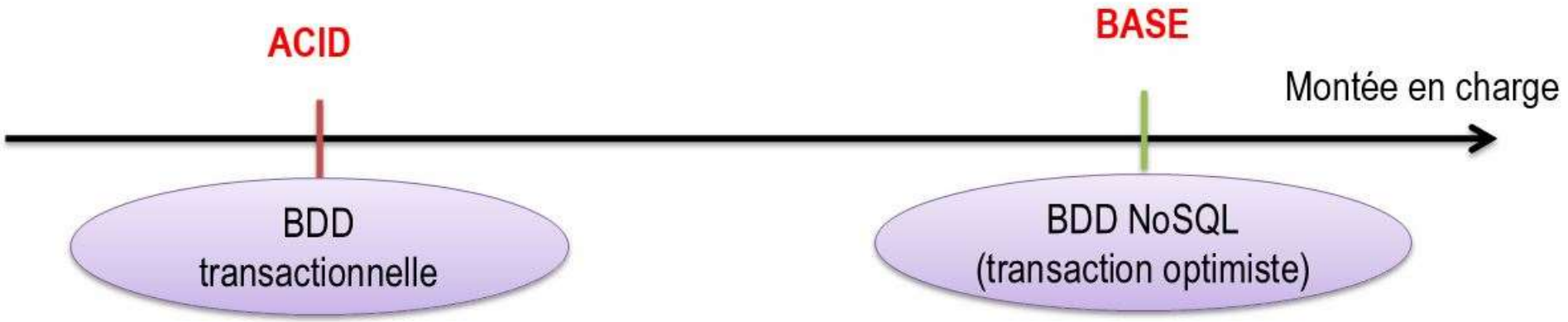
## Eventually consistent

- **Finalement consistent:** spécifie que le système sera consistent à mesure que le temps passe, à condition qu'il ne reçoive pas une action utilisateur entre temps

# ACID vs BASE

| ACID | BASE |
| --- | --- |
| • Atomique<br>• Cohérent<br>• Isolé<br>• Durable | • Basiquement disponible<br>• Souple état<br>• Eventuelle consistence |
| • Cohérence forte<br>• Transactions<br>• Schéma<br>• Évolutions difficiles | • Cohérence faible<br>• Procédure de harmonie<br>• Pas de schéma<br>• Évolutions faciles<br>• Rapide<br>• Favorise la disponibilité |

# ACID vs BASE




- **A**tomicity : Une transaction est une instruction appliquée totalement ou rollbacké totalement
  **C**onsistency : Toutes transaction effectuée doit garder des données cohérentes et respecter les contraintes de la base (index, typage…)
- **I**solation : Les transactions ne peuvent pas interférer les une avec les autres. Elles sont isolées.
- **D**urability : après l'exécution d'une transaction, ces effets seront permanents sur la base de données.

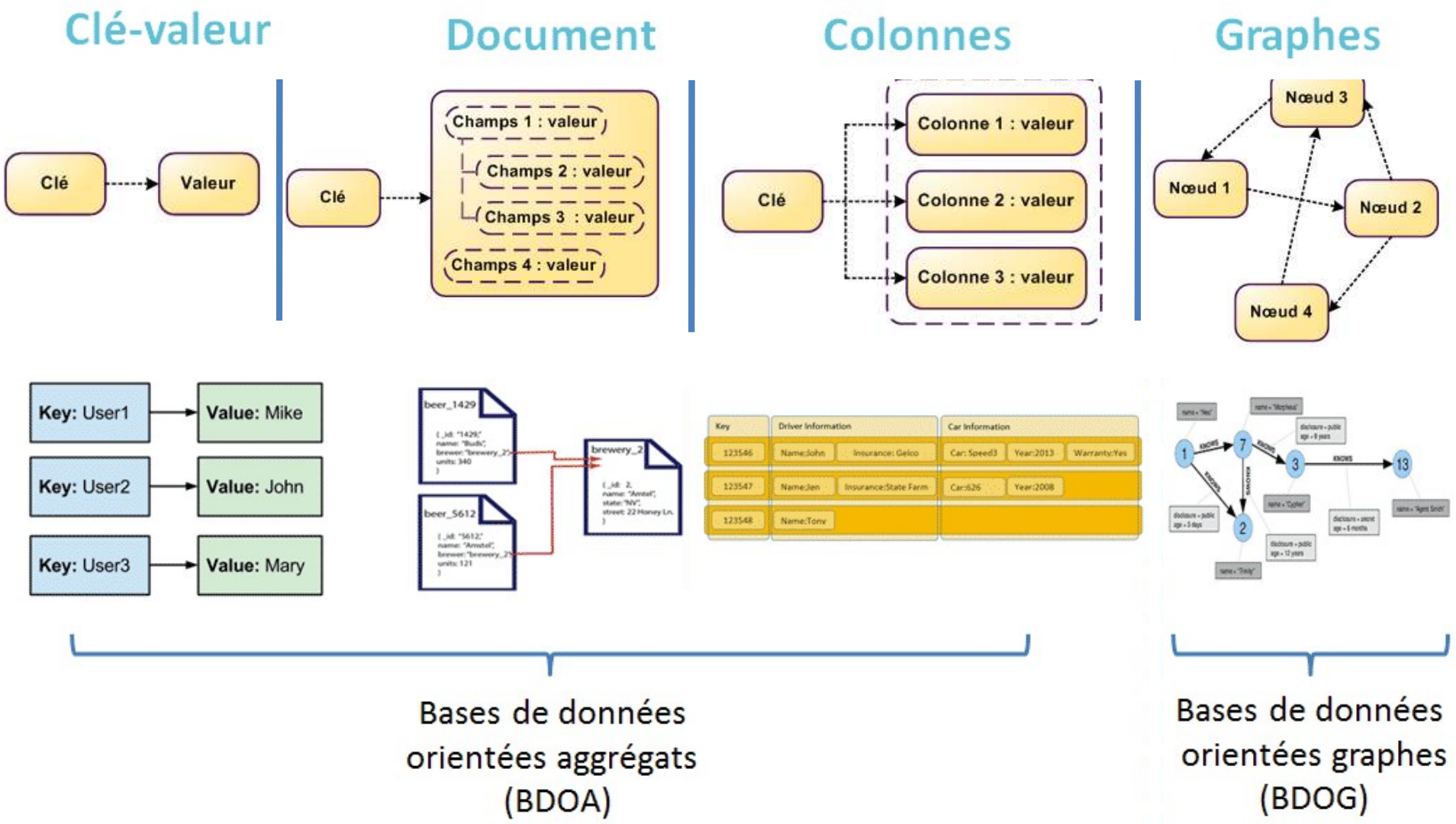
- **B**asically **A**vailable : les données sont disponibles selon le théorème CAP (haute disponibilité). La réponse ne sera pas forcement juste à 100% mais disponible.
- **S**oft state : la cohérence des données n'est pas géré par la base de données mais par les développeurs.
- **E**ventual consistency : La cohérence des données n'est pas garantie à un instant t mais les données convergeront plus tard et seront cohérentes.

# Qui utilise NoSQL?

# Chapitre2: Catégories de NoSQL


**A lire** https://www.journaldunet.com/solutions/dsi/1194284-base-nosql-laquelle-choisir-pour-quels-besoins/
https://www.lemagit.fr/article/NoSQL-le-choix-difficile-de-la-bonne-technologie

# Catégories NoSQL

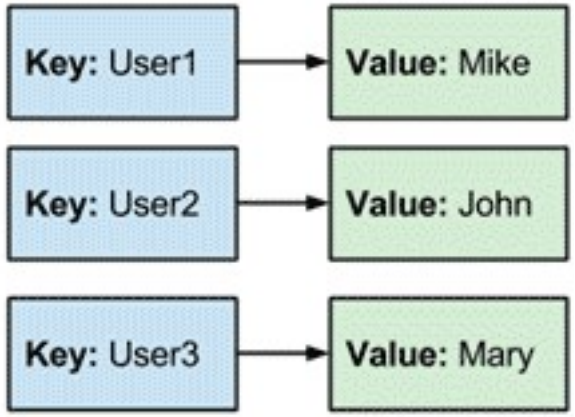


# NoSQL Clé-valeur

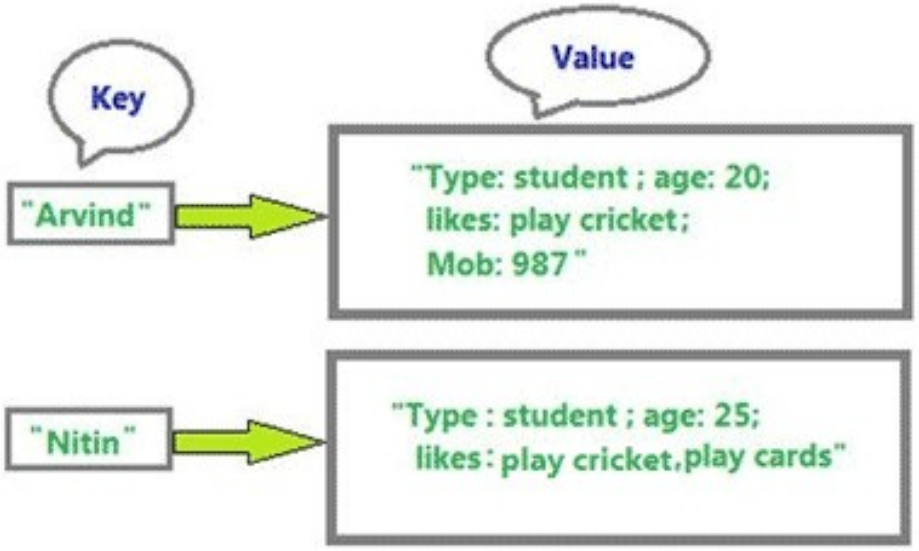
- Les BBD NoSQL les plus simples
- Chaque élément est une paire (clé, valeur)
- Conçu pour traiter les **énormes quantités** de données.
- Stockage de données avec **moins de schéma**.
- Aspects de AP du théorème **CAP.**
- Les données sont stockées comme **table de hachage** : chaque clef est unique et la valeur peut être **String, Objet sérialisé, BLOB (binary large object) etc.**

# NoSQL Clé-valeur

**Exemple 1 :**




**Exemple 2 :**



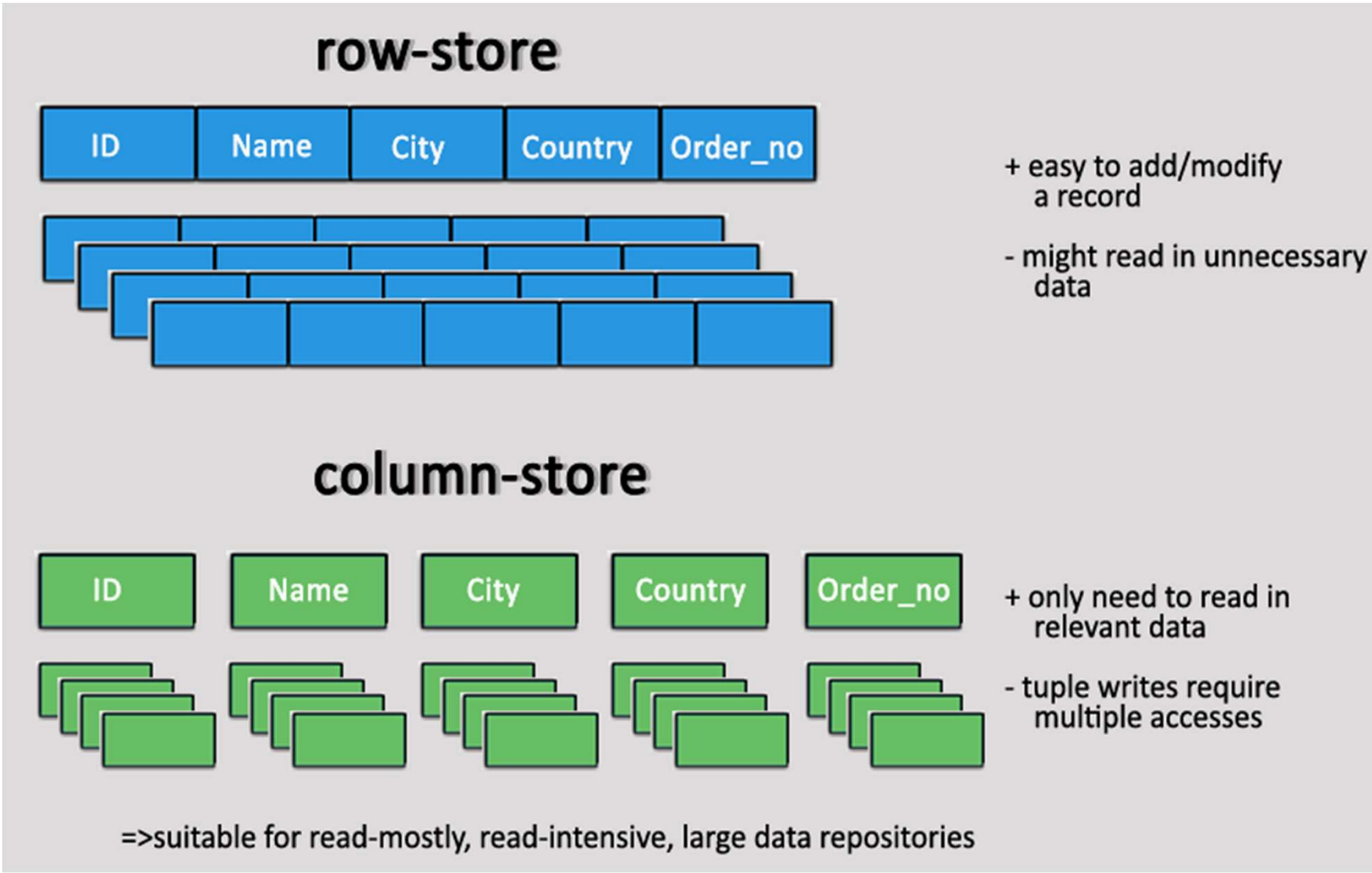
# NoSQL Orientée colonnes

- Les données sont stockées dans des **fichiers spécifiques** «column datafile».

- **Haute performance sur les requêtes d'agrégation:** COUNT, SUM, AVG, MIN, MAX.

- Les stockages orientés colonnes sont utilisés dans : **Data Warehouses** et **Business Intelligence**, **etc.**

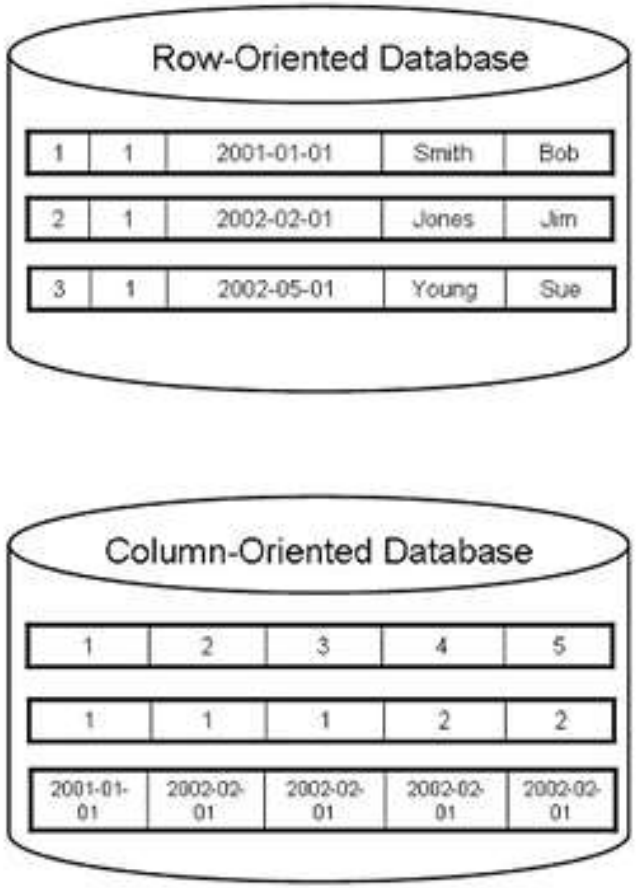
# NoSQL Orientée colonnes



# NoSQL Orientée colonnes

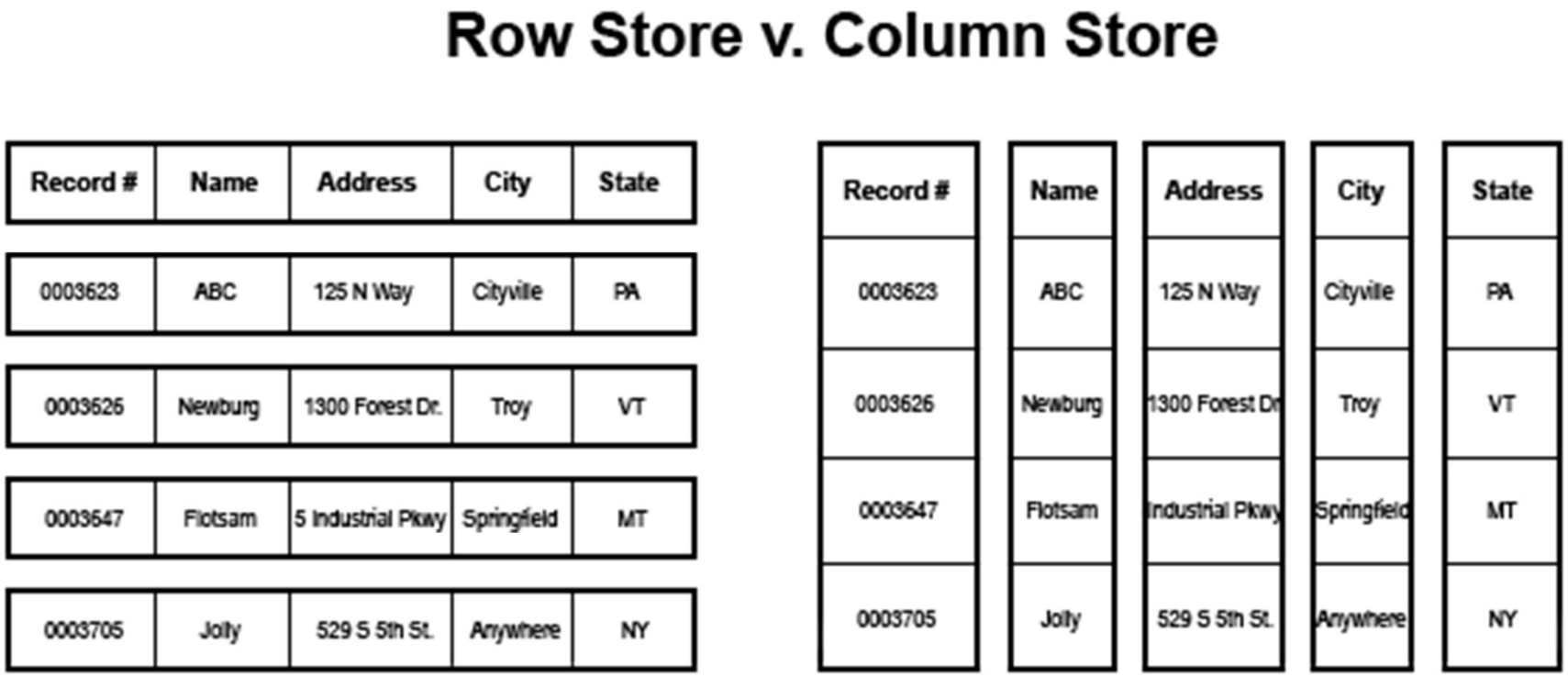
**Exemple 1 :**

| Emp_no | Dept_id | Hire_date | Emp_ln | Emp_fn |
|---|---|---|---|---|
| 1 | 1 | 2001-01-01 | Smith | Bob |
| 2 | 1 | 2002-02-01 | Jones | Jim |
| 3 | 1 | 2002-05-01 | Young | Sue |
| 4 | 2 | 2003-02-01 | Stemle | Bill |
| 5 | 2 | 1999-06-15 | Aurora | Jack |
| 6 | 3 | 2000-08-15 | Jung | Laura |



# NoSQL Orientée colonnes

**Exemple 2 :**



# NoSQL Orientée documents

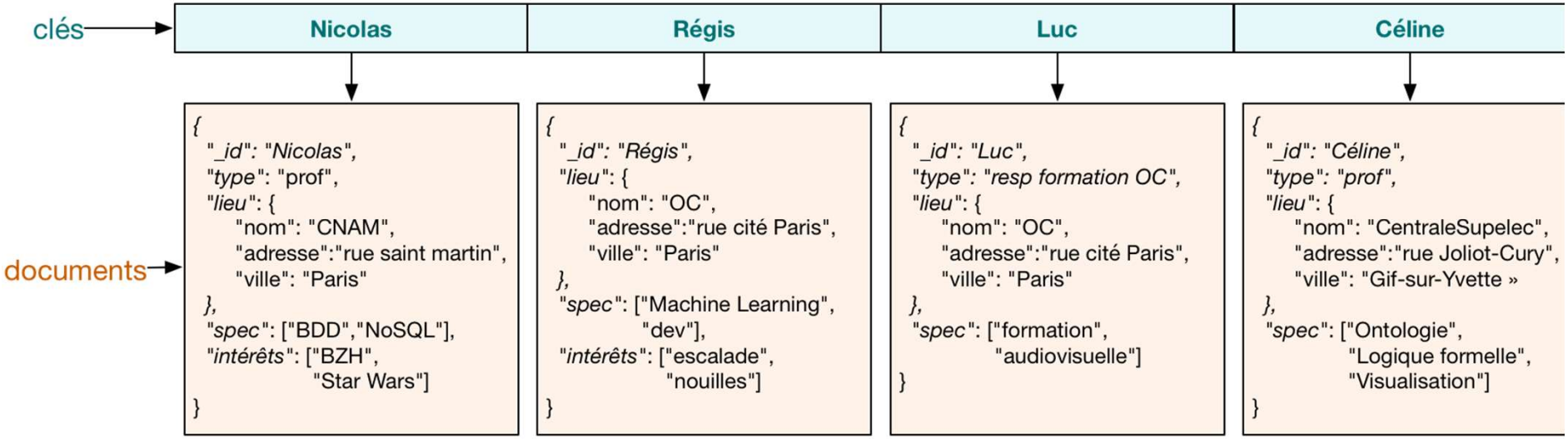
- C'est une **collection de documents**.
- Elle **étend** le paradigme **clef/valeur**, avec des « **documents** » plus complexes à la place des données simples, et une clef unique pour chacun d'eux.
- Les documents sont de type **JSON** ou **XML**
- Un **document** est un **ensemble de clef-valeur** où la clef permet d'accéder à sa valeur.

  → **pouvoir récupérer, via une seule clef, un ensemble d'informations structurées de manière hiérarchique** (plusieurs jointures en BDR)



# NoSQL Orientée documents

**Exemple 1 :**



# NoSQL Orientée documents

**Exemple 2 :**


- Toutes les données stockées dans une ligne qui s'étend sur 20 tables d'une BDR seront regroupées dans un seul document/objet de type JSON.

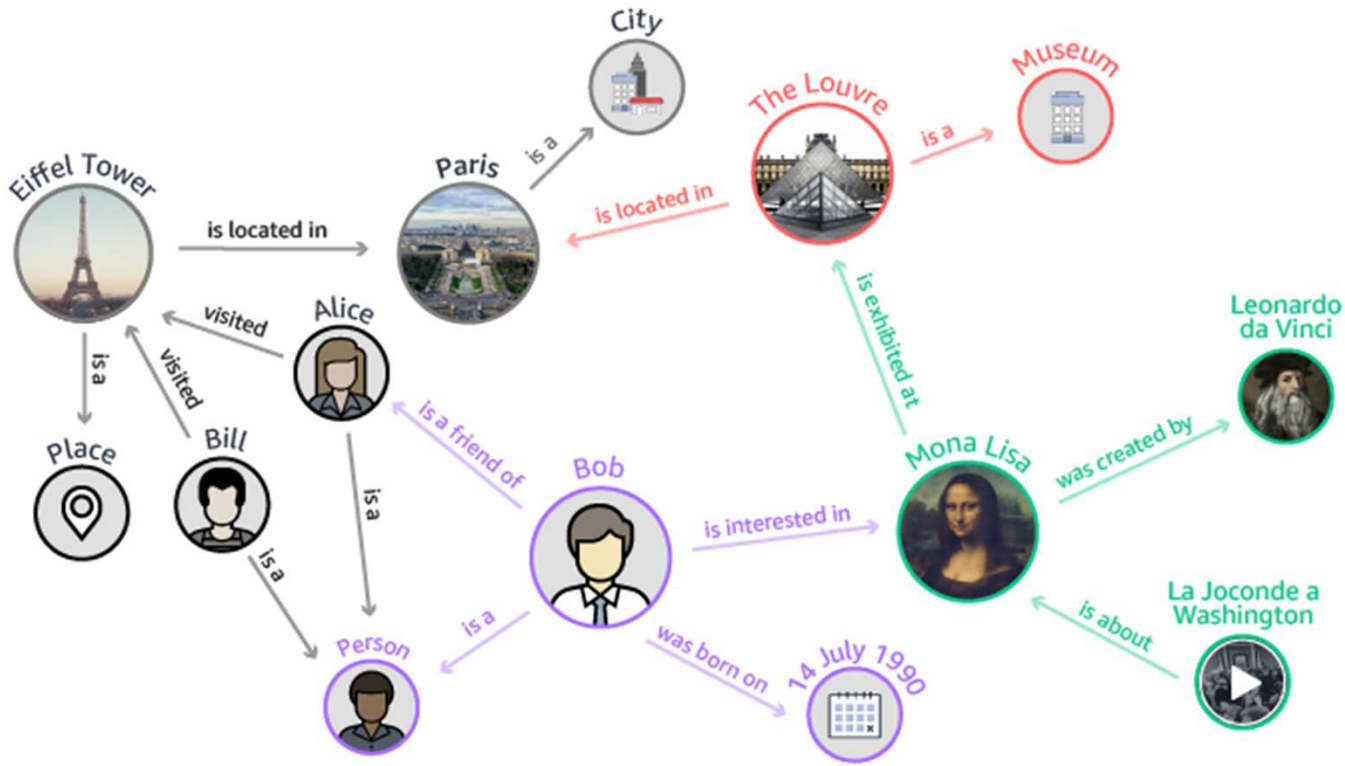
# NoSQL Orientée graphe

- Les données sont représentées par des graphes
  - Un élément : un nœud
  - Les relations : des arêtes orientées
  - Les deux peuvent avoir des attributs
- Adapté aux traitements des données des réseaux sociaux

| Modèle relationnel | Modèle de graphe |
|---|---|
| • Tables<br>• Lignes<br>• Colonnes<br>• Jointure | • Ensemble de sommets et des arêtes.<br>• Sommets<br>• Paires clef-valeur<br>• Arrêtes |

# NoSQL Orientée graphe

**Exemple 1**



# NoSQL Orientée graphe

**Exemple 2**