

# Les Fondamentaux du NoSQL

Évolution des architectures de données et nouveaux paradigmes de stockage



# Objectifs de la Formation



## Limites des SGBDR

Appréhender les limites des systèmes relationnels dans les contextes Big Data et comprendre les défis d'évolutivité.



## Architectures Distribuées

Maîtriser les concepts fondamentaux d'architectures distribuées, incluant les théorèmes CAP et BASE.



## Typologies NoSQL

Identifier les 4 typologies NoSQL majeures et leurs cas d'usage spécifiques en production.



## Critères Décisionnels

Acquérir les critères techniques et métiers pour le choix optimal d'une technologie de stockage.

# Agenda

01

## Contexte d'Émergence et Big Data

Comprendre les facteurs historiques et technologiques qui ont conduit à l'émergence du NoSQL.

02

## Architectures Distribuées et Cloud

Explorer les paradigmes de distribution, réPLICATION et partitionnement dans le cloud.

03

## Analyse Comparative : SGBDR vs NoSQL

Étudier les différences fondamentales entre les approches relationnelles et NoSQL.

04

## Théorèmes Fondamentaux et Modèles de Cohérence

Maîtriser le théorème CAP, les propriétés BASE et les stratégies de cohérence.

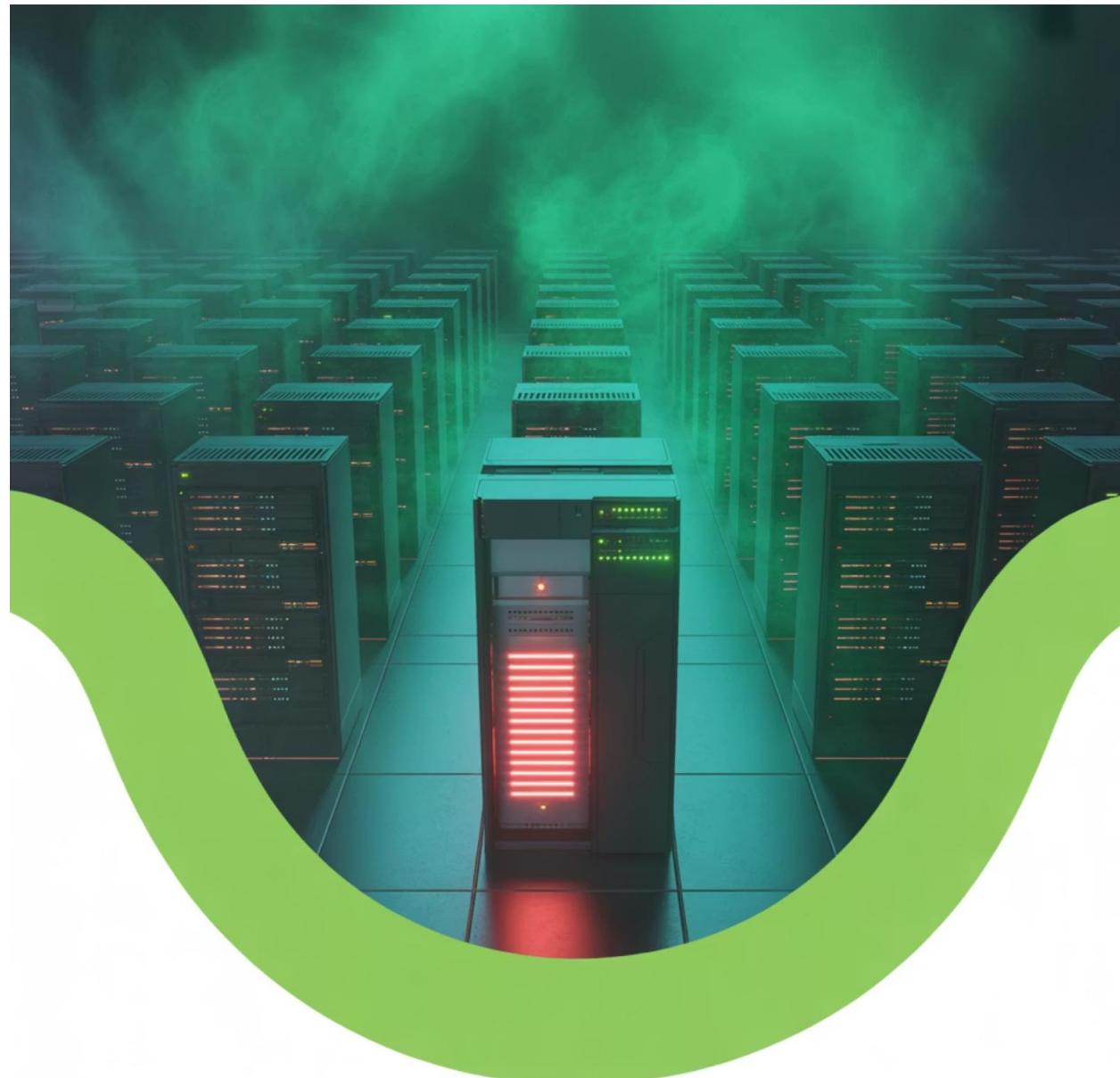


# Évolution Historique du Stockage

- 1 — 1970-2000 : Hégémonie du Modèle Relationnel  
Dominance absolue des SGBDR avec Oracle, SQL Server et DB2 comme standards industriels.
- 2 — Années 2000 : Avènement du Web 2.0  
Rupture des usages avec l'émergence des réseaux sociaux, du e-commerce mondial et du contenu généré par les utilisateurs.
- 3 — Constat Critique  
Inadéquation du modèle unique "One size fits all" face aux nouvelles exigences de volumétrie, vitesse et variété.

# Limitations des Architectures Traditionnelles

## Facteurs de Rupture



### → Passage à l'Échelle

De quelques utilisateurs internes à des millions d'utilisateurs simultanés à l'échelle mondiale.

### → Haute Disponibilité

Exigence de disponibilité 24/7 avec des SLA dépassant 99,99% pour les services critiques.

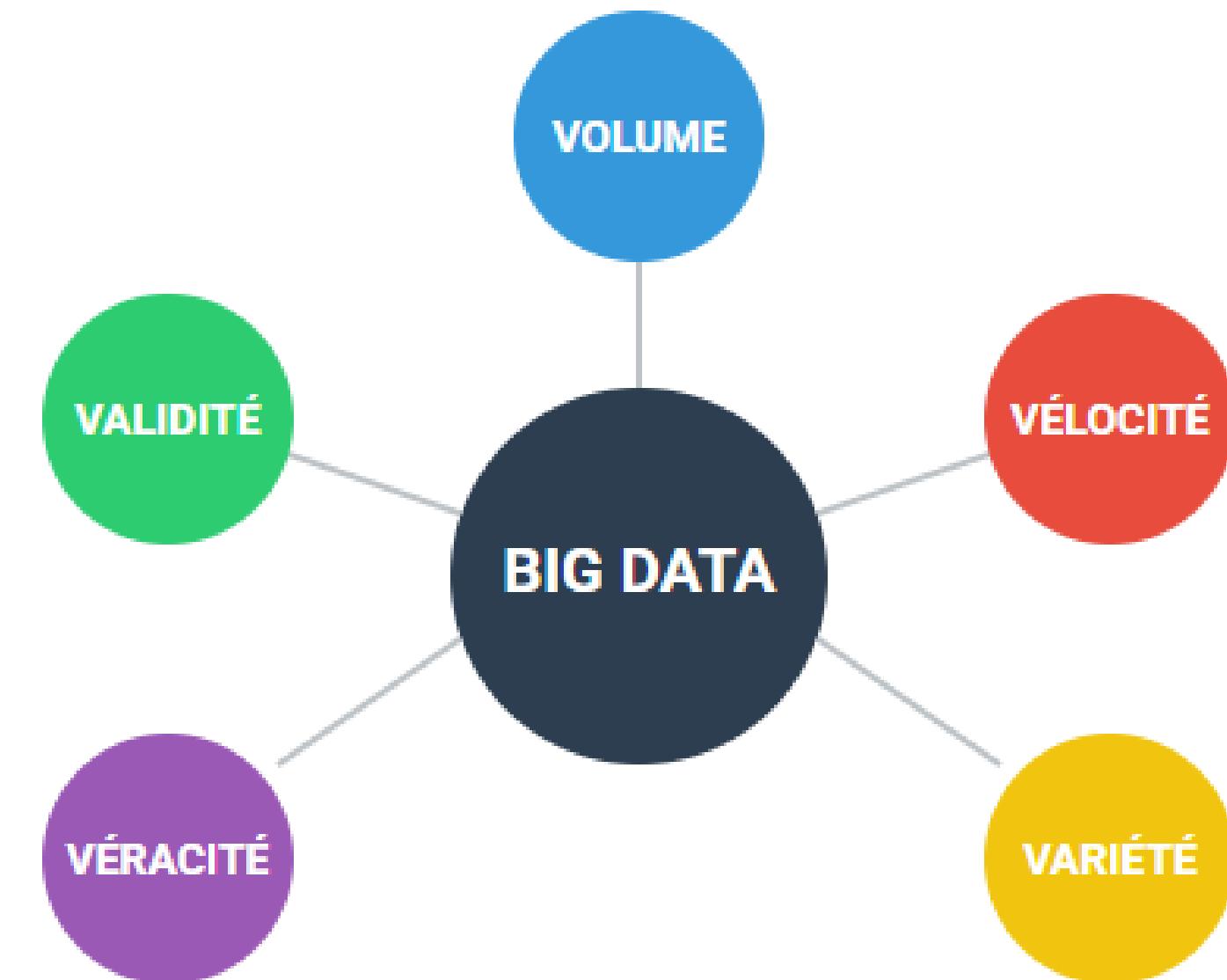
### → Saturation Verticale

Limites physiques et coûts prohibitifs des architectures monolithiques traditionnelles.

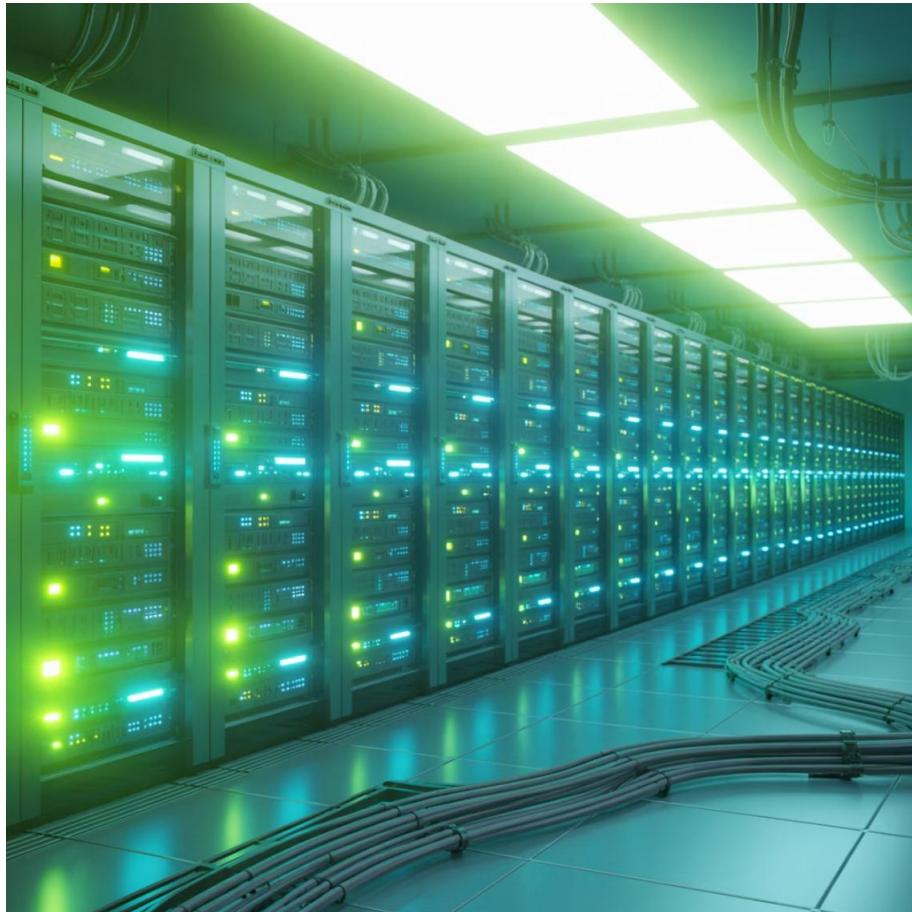
## Caractérisation du Big Data

### LES 5 V DU BIG DATA

---



# Le Défi de la Volumétrie



## Dimension Volume

La croissance exponentielle des données générées quotidiennement dépasse les capacités des systèmes traditionnels.

2.5EB

Données quotidiennes

175ZB

Projection 2025

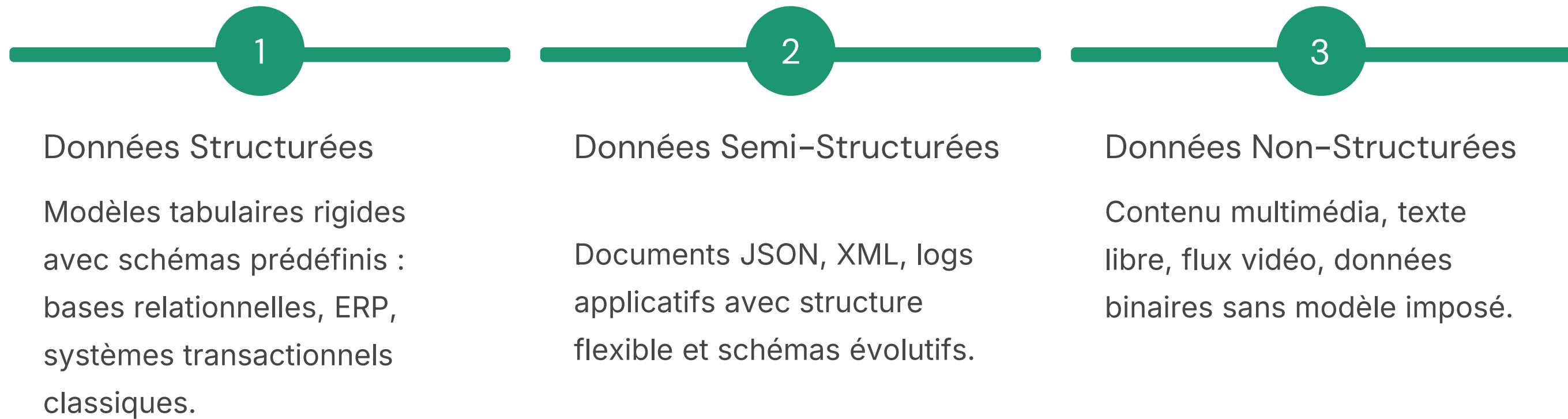
Volume généré chaque jour à  
l'échelle mondiale

Volume total estimé de la  
datasphère globale

**Conséquence critique :** Obsolescence totale du stockage centralisé unique et nécessité d'architectures distribuées.

# Hétérogénéité des Données

## Dimension Variété



**Enjeu architectural :** Nécessité de flexibilité du modèle de stockage pour accommoder le polymorphisme des données.

# Contraintes de Vélocité et Temps Réel

## Dimension Vélocité

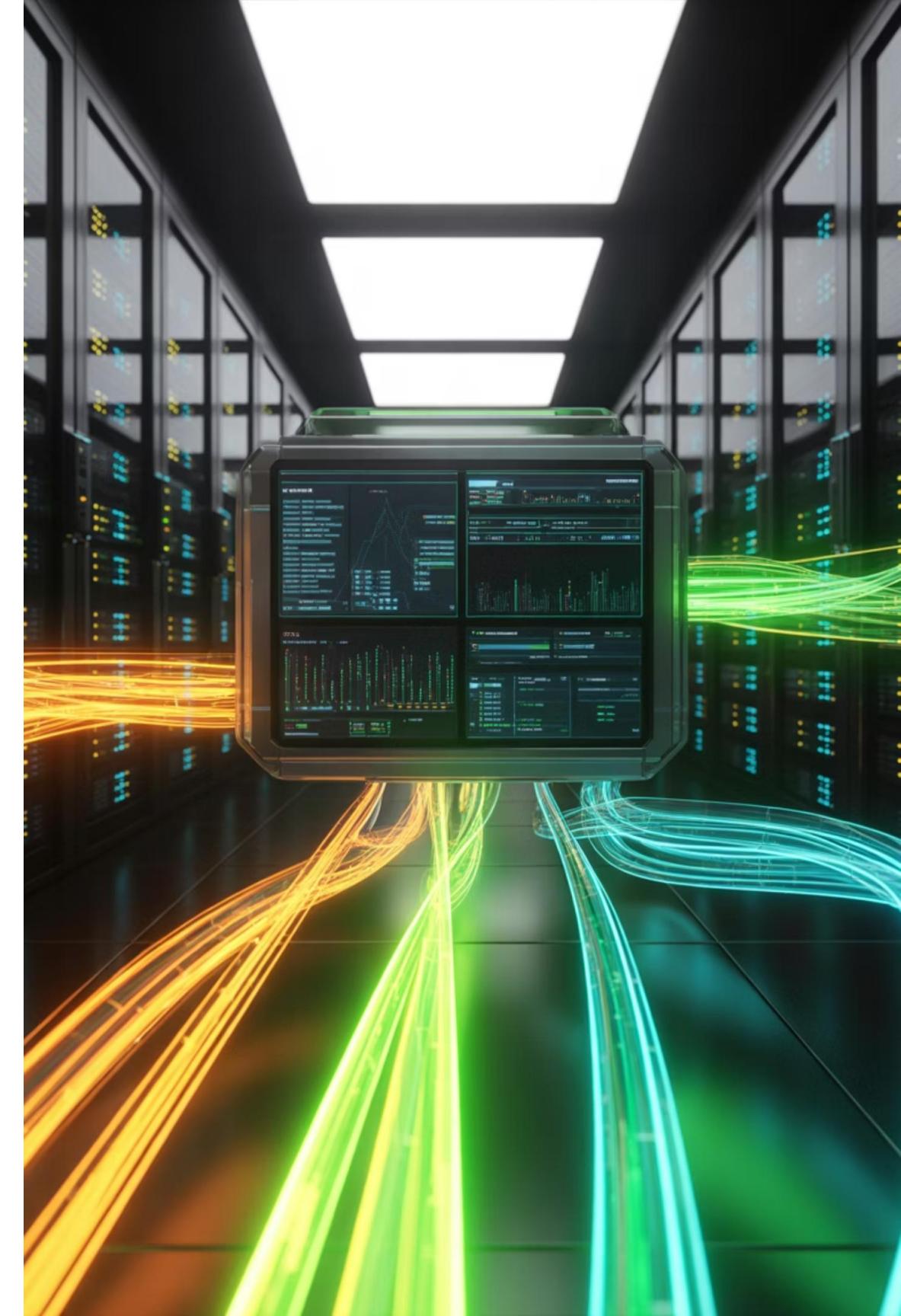
- Traitement des flux continus en mode streaming avec latences sub-seconde
- Ingestion massive de métriques IoT (millions d'événements/seconde)
- Nécessité de latences d'écriture minimales pour les applications temps réel

## Cas d'Usage Critiques

**IoT et Capteurs :** Télémetrie industrielle, véhicules connectés, smart cities

**Finance :** Trading algorithmique haute fréquence, détection de fraude en temps réel

**E-commerce :** Recommandations personnalisées instantanées, analyse comportementale



# Qualité et Pertinence de la Donnée

## Véracité

Gestion de l'incertitude et de la fiabilité des sources dans un écosystème multi-origines.

- Validation de la provenance des données
- Détection des anomalies et incohérences
- Gestion des données contradictoires ou obsolètes

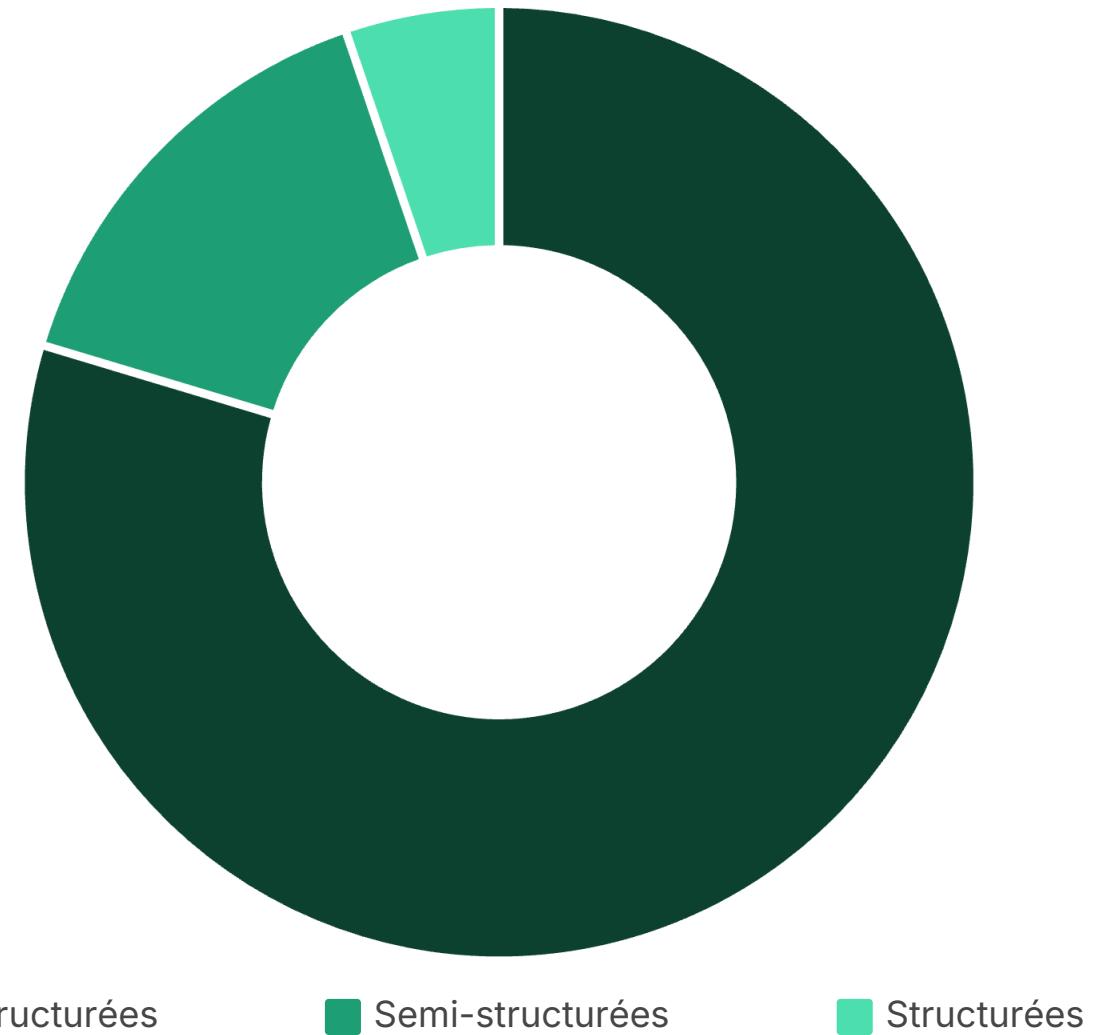
## Validité

Extraction de la valeur métier et transformation en Smart Data exploitable.

- Alignement avec les objectifs business
- Pertinence contextuelle pour la prise de décision
- ROI mesurable de l'exploitation des données

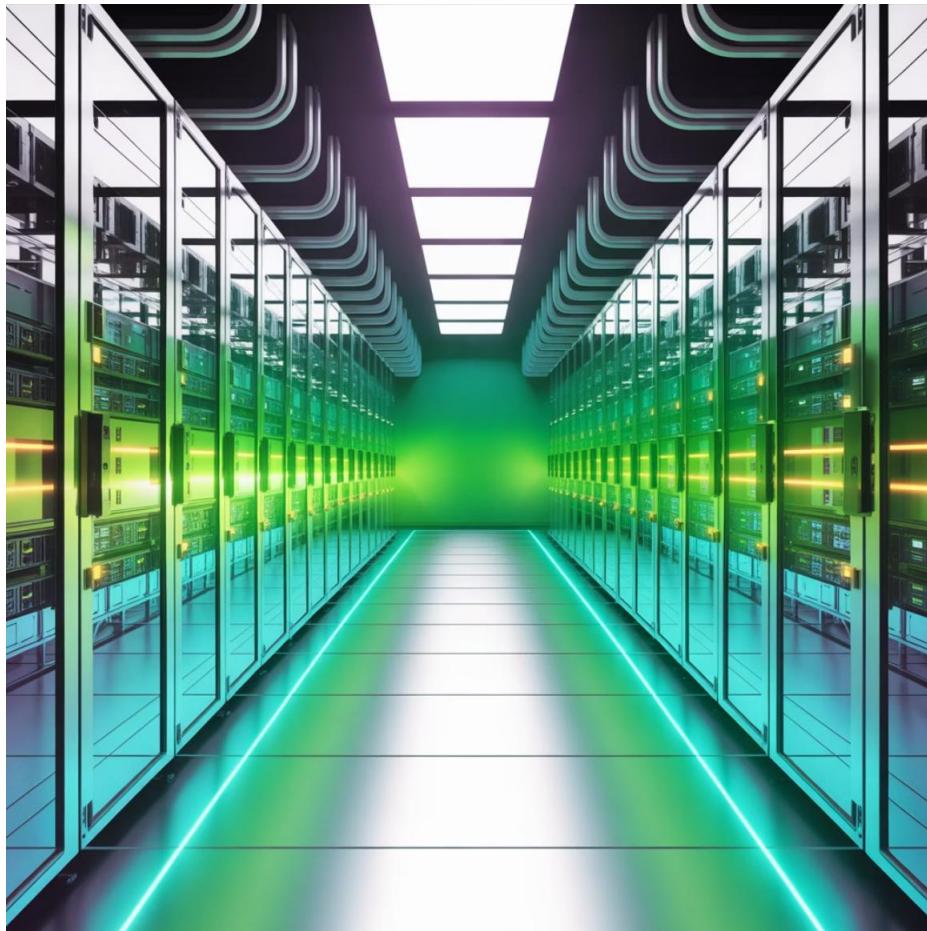
# Répartition Structurée vs Non-Structurée

Typologie des Données à l'Échelle Mondiale



Prédominance massive des données non-structurées (environ 80% du volume mondial), avec une stagnation relative des données transactionnelles classiques issues des systèmes ERP et CRM traditionnels.

# L'Apport Technologique de Google



## Rôle des GAFAM : Google



### Google File System (GFS)

Système de fichiers distribué révolutionnaire pour le stockage à grande échelle.



### BigTable (2006)

Introduction du modèle orienté colonnes pour indexation massive sur infrastructure standard.



### Commodity Hardware

Utilisation de serveurs standards pour réduire les coûts et augmenter la résilience.



# L'Apport Technologique d'Amazon

## Rôle des GAFAM : Amazon

### Dynamo (2007)

Introduction du Key-Value Store distribué haute disponibilité, fondement de DynamoDB.

### Problématique E-commerce

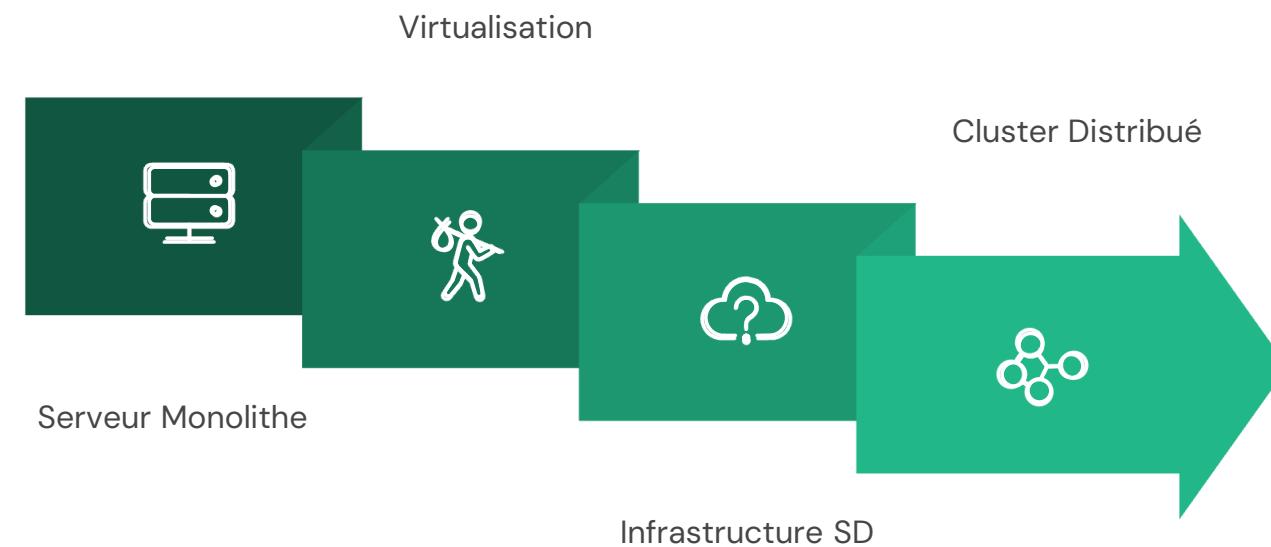
Élimination totale des temps d'arrêt (SPOF) pour maintenir la continuité des ventes en ligne.

### Priorisation Stratégique

Choix architectural de privilégier la disponibilité sur la cohérence stricte (modèle AP).

# Principes de l'Architecture Cloud

## Paradigme Cloud Computing



Du serveur dédié au cluster de nœuds

Transition vers une architecture élastique et horizontalement scalable.

Infrastructure Software-Defined

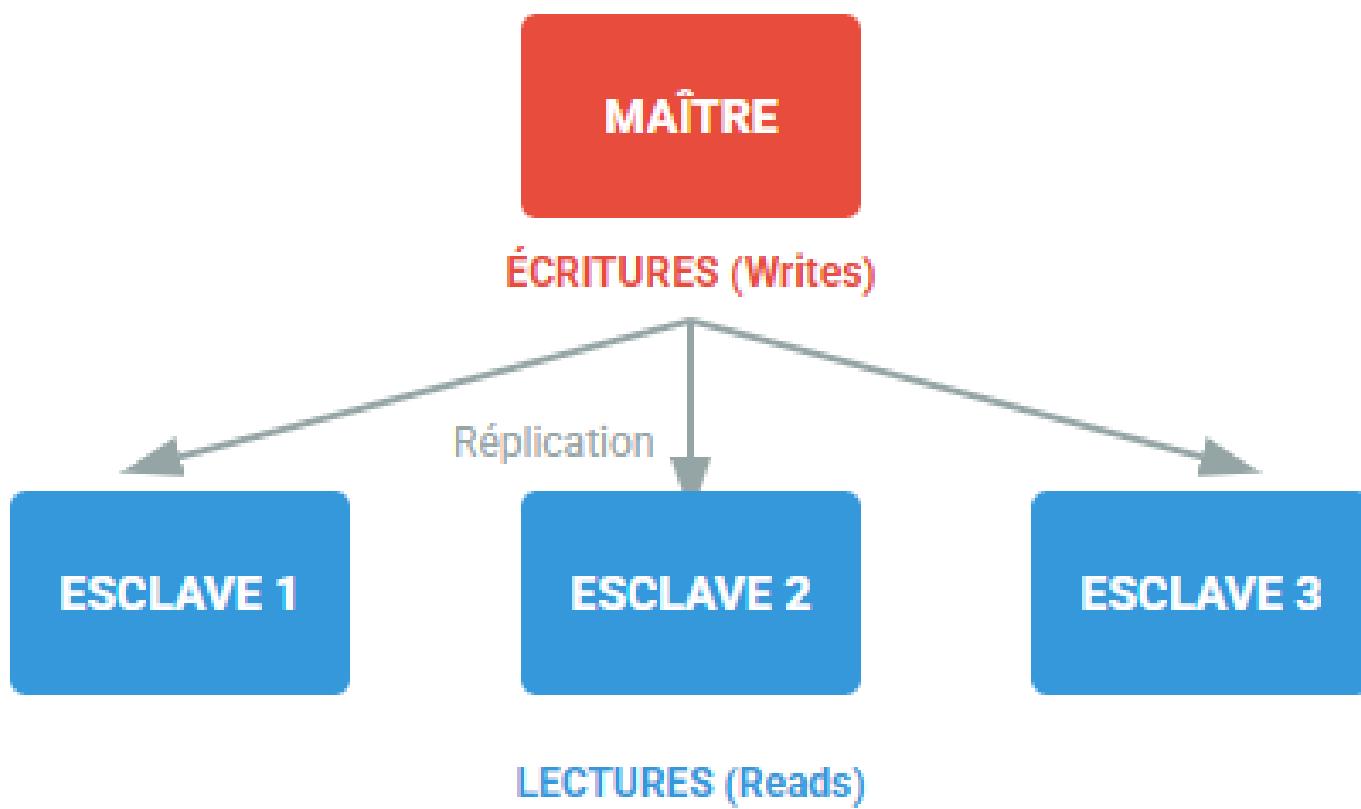
Abstraction matérielle et automatisation complète de la gestion des ressources.

Commodity Hardware

Utilisation stratégique de matériel banalisé pour optimiser le rapport coût/performance.

# Architecture Master-Slave

## ARCHITECTURE MAÎTRE / ESCLAVE



Nœud Maître (Primary)

1

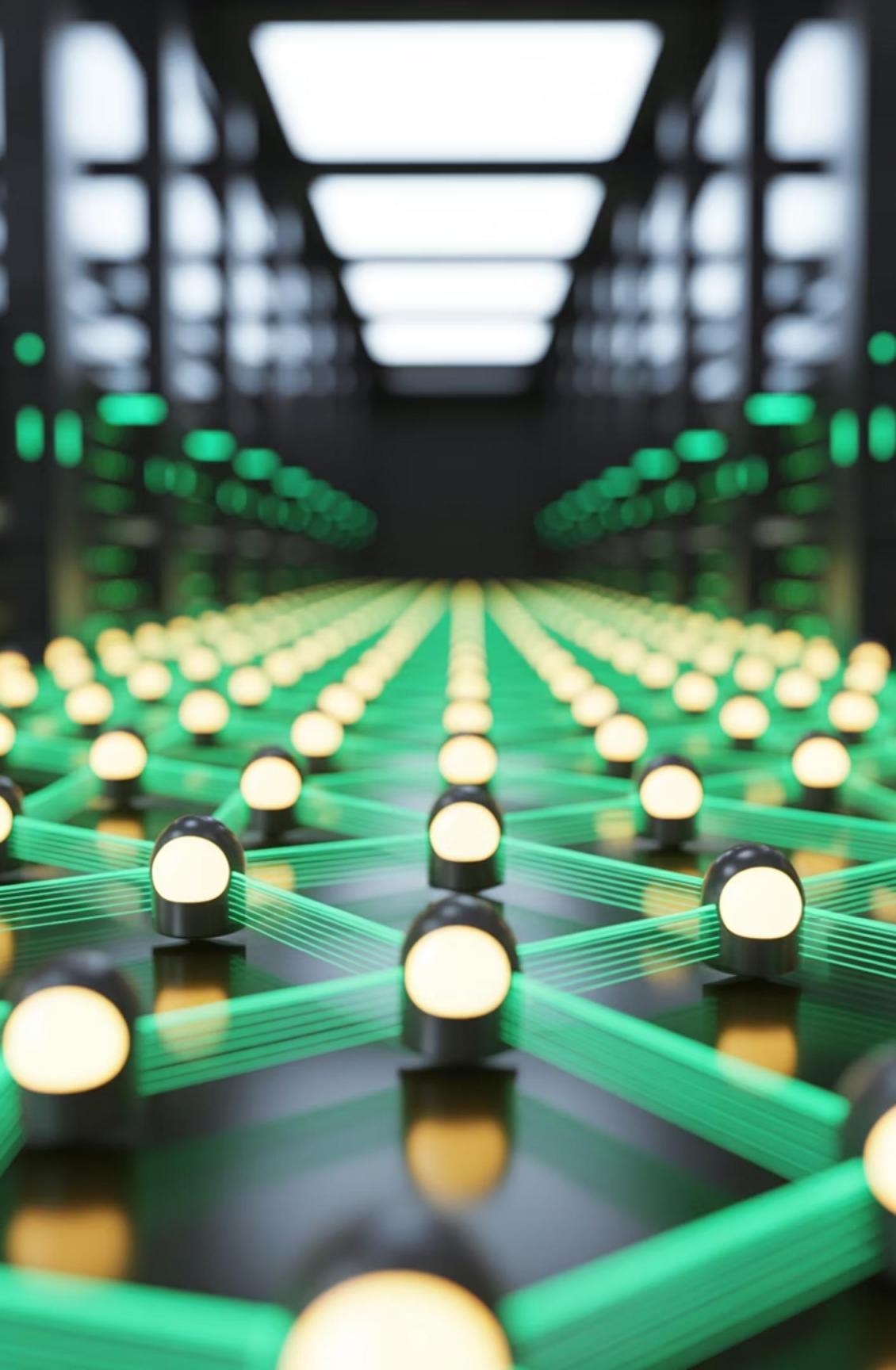
Point d'entrée unique pour toutes les opérations d'écriture et gestion de la cohérence.

Nœuds Esclaves (Secondaries)

2

Réplicas en lecture seule pour distribution de la charge des requêtes SELECT.

- **Limite critique :** Le nœud Maître reste un point unique de défaillance (SPOF) pour les opérations d'écriture, nécessitant des mécanismes de failover sophistiqués.



# Architecture Décentralisée (Masterless)

## Topologie Peer-to-Peer

Absence de  
Hiérarchie

Tous les nœuds  
sont égaux et  
peuvent accepter  
lectures et  
écritures sans  
coordination  
centralisée.

Haute  
Disponibilité  
Native

Aucun SPOF  
dans le système,  
résilience  
maximale aux  
pannes de  
nœuds  
individuels.

Gestion de  
Conflits  
  
Complexité  
accrue avec  
mécanismes de  
résolution (Vector  
Clocks, Last Write  
Wins).

# Le Sharding (Partitionnement)

Partitionnement Horizontal

Clé de shard

Attribut utilisé pour partitionner

Scalabilité linéaire

Ajout simple de nœuds

Sharding horizontal

Partitionnement par hash

Répartition uniforme des enregistrements

Shards distincts

Noeuds de stockage indépendants

Partitionnement par intervalle

Plages de valeurs séquentielles



# La RéPLICATION des Données

## Mécanismes de RéPLICATION



### Redondance

Copie synchrone ou asynchrone des données sur N nœuds définis par le facteur de réPLICATION (Replication Factor).



### Résilience

Tolérance aux pannes avec possibilité de perdre N-1 nœuds sans perte de données.



### Distribution de Charge

Répartition intelligente des requêtes de lecture sur l'ensemble des réplicas disponibles.

# Haute Disponibilité et Failover

## Gestion des Pannes



### Détection

Surveillance continue via mécanismes de heartbeat entre nœuds pour identifier les défaillances.



### Bascule Automatique

Failover transparent pour les clients avec élection automatique d'un nouveau leader si nécessaire.



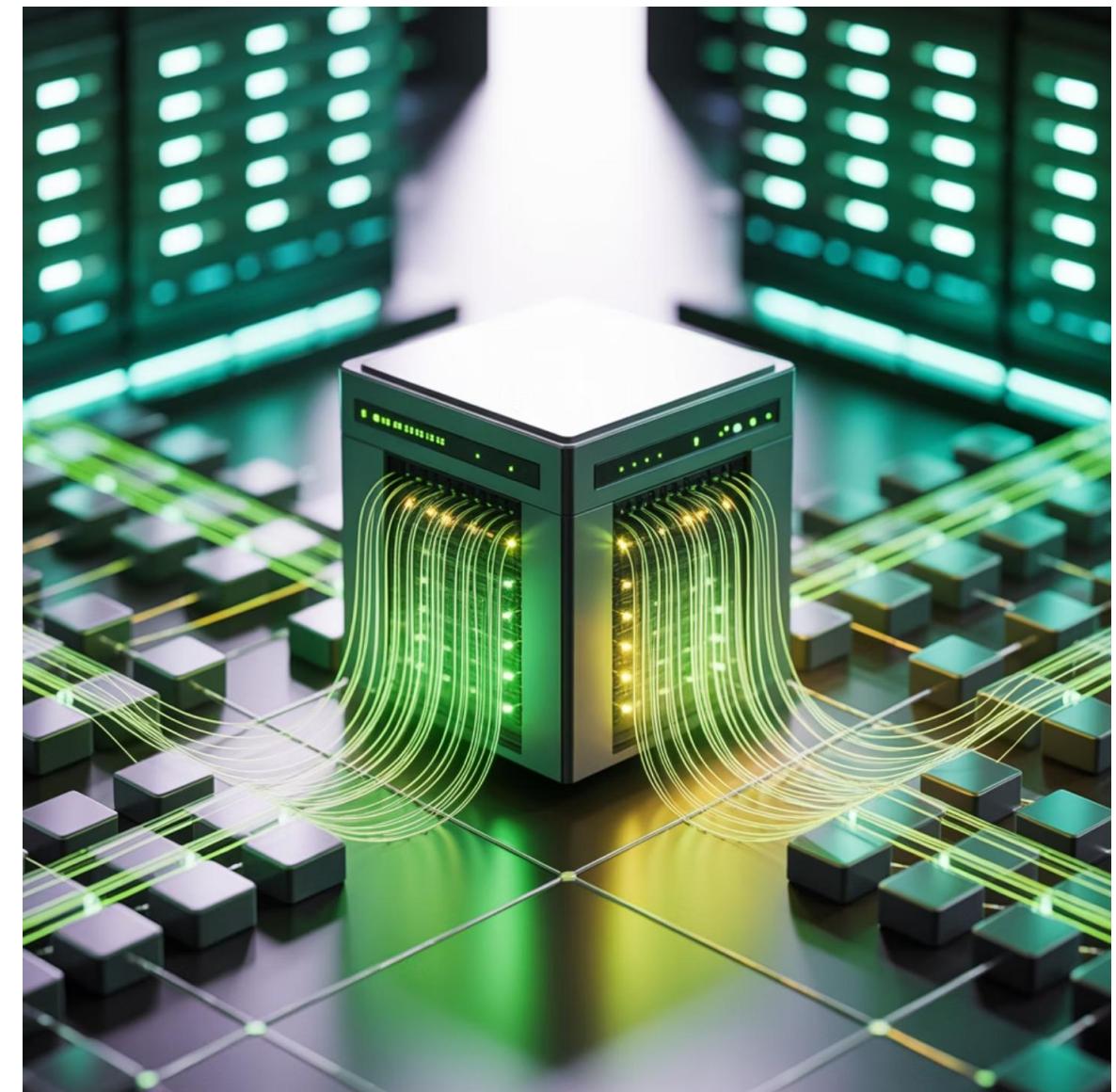
### Résilience Systémique

Capacité du cluster à maintenir les services malgré les pannes matérielles ou réseau.

# Load Balancing

## Répartition de Charge

- **Distribution algorithmique** des requêtes entrantes selon des stratégies variées (round-robin, least-connections, consistent hashing)
- **Prévention des hotspots** pour éviter la surcharge de nœuds spécifiques
- **Optimisation des ressources** avec utilisation homogène de la capacité du cluster
- **Latence minimisée** via routage intelligent vers les nœuds les plus proches ou les moins chargés



# De la Transaction à l'Interaction

## Évolution des Modèles d'Usage

Dimension	OLTP Classique	Big Data / Interactionnel
Nature des données	Transactionnelles, critiques	Comportementales, analytiques
Volume unitaire	Faible, structuré	Massif, hétérogène
Intégrité	Forte, ACID requis	Tolérance à l'approximation
Latence	Milliseconde garantie	Sub-seconde acceptable
Cas d'usage	Banking, ERP, inventaire	Recommandations, logs, IoT



# Caractéristiques du Modèle Relationnel

## Le Standard SGBDR

### Maturité Technologique

Plus de 40 années d'évolution, d'optimisation et de standardisation avec SQL comme langage universel.

### Normalisation des Données

Application rigoureuse des formes normales (1NF à 5NF) pour éliminer la redondance et garantir l'intégrité.

### Garanties ACID

Propriétés transactionnelles strictes avec intégrité référentielle et contraintes déclaratives.

# Contraintes du Modèle Relationnel

## Limitations SGBDR dans les Contextes Big Data

### Rigidité Structurelle

Schéma statique imposé avant insertion, migrations coûteuses pour évolutions.

### Coût des Jointures

Complexité algorithmique exponentielle  $O(n^2)$  sur tables volumineuses, goulot d'étranglement majeur.

### Sharding Manuel

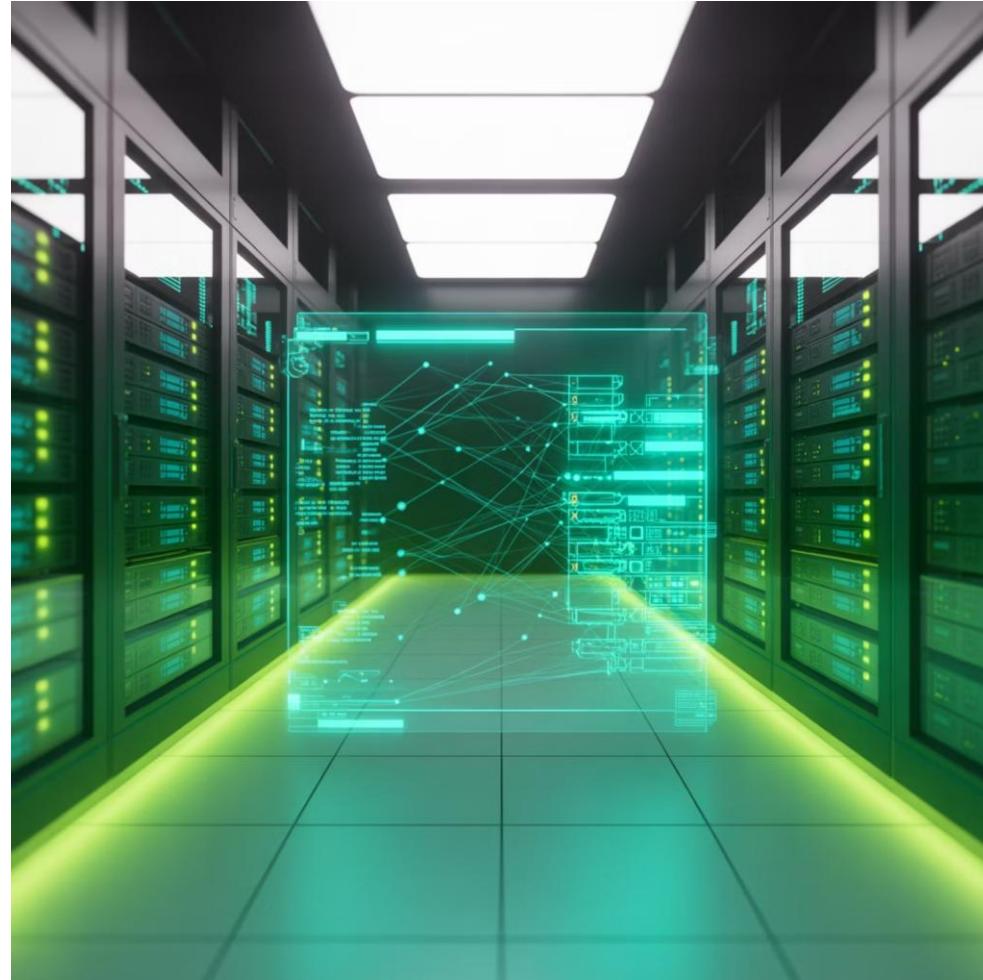
Partitionnement horizontal complexe, non natif, nécessitant une logique applicative lourde.

Ces limitations deviennent critiques au-delà de plusieurs téraoctets et millions de transactions par seconde.

# Gestion de la Structure de Données

Schema-on-Write

Approche SGBDR



Structure rigide imposée à l'écriture avec validation des contraintes et types de données avant persistance.

Schema-on-Read

Approche NoSQL



Structure interprétée à la lecture, permettant le polymorphisme et l'évolution continue du modèle.

# Alignement avec le Développement Agile

## Méthodologie Agile et NoSQL

Réduction Friction  
Diminution des barrières entre Dev et Ops grâce à la flexibilité du schéma.



Évolution Continue  
Capacité d'adaptation du modèle sans interruption de service ni migration complexe.

Mapping Naturel  
Stockage orienté  
Objet/Document (JSON) aligné avec les langages modernes.



# Garanties Transactionnelles : ACID

## Propriétés ACID



### Atomicity

Indivisibilité : une transaction est entièrement exécutée ou entièrement annulée (rollback).



### Consistency

Validité : respect des contraintes d'intégrité et règles métier après chaque transaction.



### Isolation

Sérialisation : exécution concurrente équivalente à une exécution séquentielle.



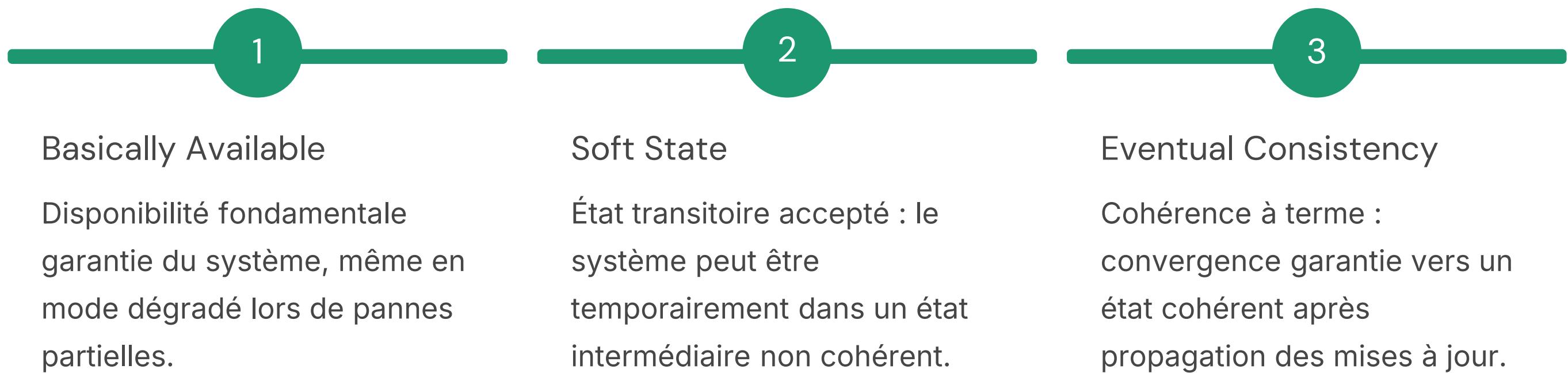
### Durability

Persistante : garantie de conservation des données validées même en cas de panne.

**Domaines d'application critiques :** Finance, comptabilité, gestion des stocks, systèmes de paiement, où l'exactitude est impérative.

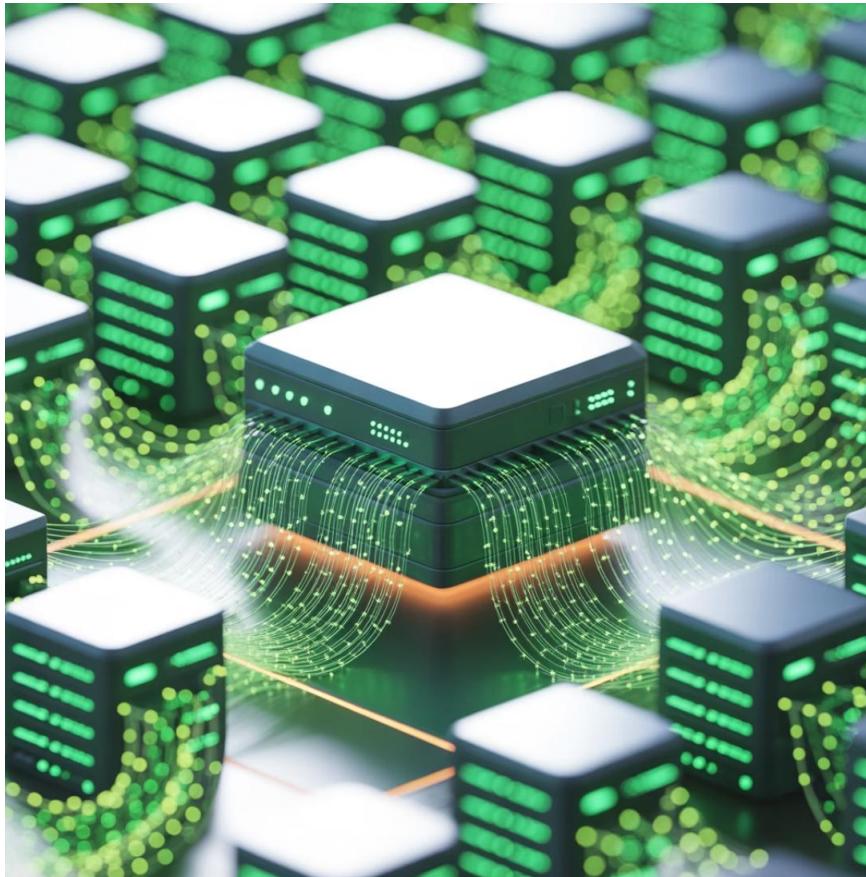
# Garanties Distribuées : BASE

## Propriétés BASE



**Philosophie architecturale :** Priorisation de la latence minimale et de la disponibilité maximale au détriment de la cohérence immédiate.

# Le Concept d'Eventual Consistency



## Cohérence à Terme

01

---

### Propagation Asynchrone

Diffusion progressive des mises à jour sur l'ensemble des réplicas sans blocage synchrone.

02

---

### Fenêtre d'Incohérence

Période temporaire (généralement < milliseconde) où différents nœuds peuvent retourner des valeurs différentes.

03

---

### Convergence Garantie

Compromis nécessaire pour obtenir performance et disponibilité en environnement distribué.



Availability

Partition tolerance

# Le Théorème CAP (Eric Brewer)

## Théorème CAP – Définition

Consistency (C)

Cohérence atomique : tous les nœuds voient les mêmes données au même instant, lecture reflétant toujours la dernière écriture.

Availability (A)

Disponibilité totale : chaque requête reçoit une réponse (succès ou échec) sans garantie que les données soient les plus récentes.

Partition Tolerance (P)

Tolérance au partitionnement réseau : le système continue de fonctionner malgré les coupures de communication entre nœuds.

# Impossibilité du "CAP" Simultané

Théorème CAP – Implications Architecturales

## LE THÉORÈME CAP

### Systèmes CP

Priorité à la cohérence : bloquent les opérations lors de partitions réseau pour garantir la cohérence.

Exemples : MongoDB, HBase, Redis Cluster

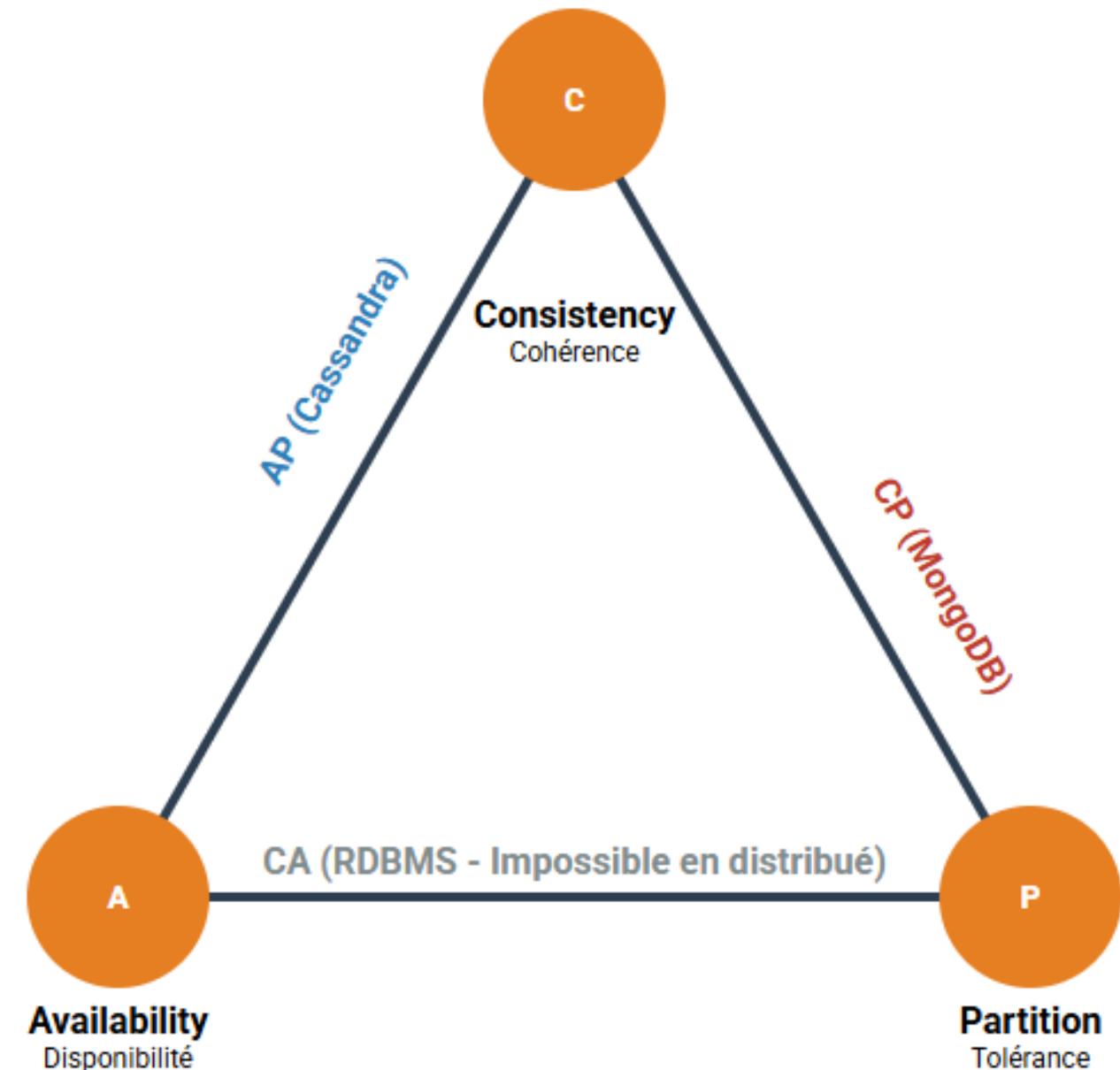
### Systèmes AP

Priorité à la disponibilité : acceptent les incohérences temporaires pour maintenir le service.

Exemples : Cassandra, DynamoDB, Riak

### Constat fondamental :

Dans un système distribué, P est obligatoire, imposant un choix binaire entre C et A.



# Scalabilité Verticale (Scale-Up)

## Stratégies de Scalabilité : Approche Verticale

Augmentation des ressources matérielles d'un nœud unique (CPU, RAM, disques) pour accroître les capacités de traitement.

### Contraintes Majeures

- **Coût exponentiel** : Les serveurs haute performance ont un prix non-linéaire
- **Limites physiques** : Plafonds technologiques des composants matériels
- **Temps d'arrêt** : Nécessité d'interruption de service pour les upgrades matériels
- **Single Point of Failure** : Dépendance totale à un seul serveur



# Scalabilité Horizontale (Scale-Out)

## Stratégies de Scalabilité : Approche Horizontale

100%                     $\infty$                      $\approx 1X$

Disponibilité Continue	Capacité Illimitée	Coût Linéaire
Ajout de nœuds sans interruption de service	Croissance théoriquement sans limite	Augmentation proportionnelle des coûts

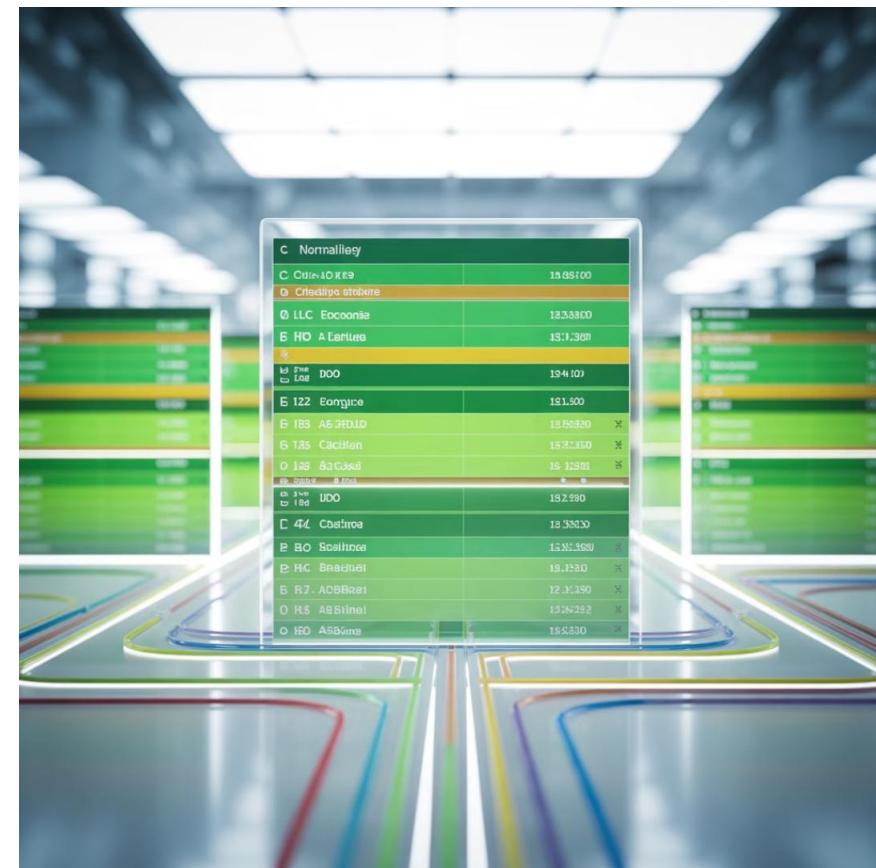
**Principe :** Ajout de serveurs standards supplémentaires au cluster pour distribution de la charge et des données. Approche privilégiée par les architectures NoSQL modernes.



# Normalisation vs Agrégation

## Modélisation des Données

### Approche Relationnelle



Normalisation : Dispersion de la donnée sur multiples tables pour éliminer la redondance.

- Intégrité maximale via contraintes
- Flexibilité des requêtes ad-hoc
- Coût élevé des JOINs en lecture

### Approche NoSQL



Dénormalisation : Agrégation de la donnée pour optimiser la Data Locality.

- Performance de lecture optimale
- Modèle aligné avec les use cases
- Redondance contrôlée acceptable

# Optimisation des Accès

## Patterns d'Accès en NoSQL



Accès par Clé

Remplacement des JOINS coûteuses par accès direct via clé primaire.



DHT

Utilisation de tables de hachage distribuées pour localisation rapide.

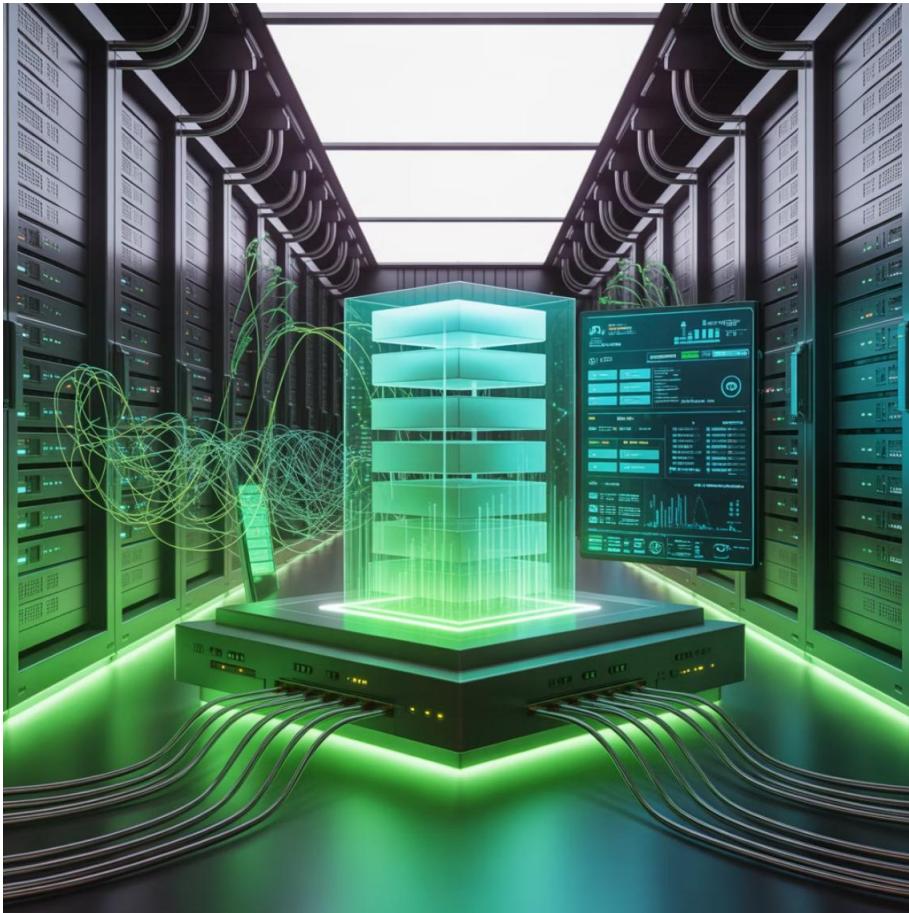


Performance  $O(1)$

Complexité algorithmique constante, déterministe et prévisible.

La modélisation NoSQL privilégie les patterns d'accès prédéfinis sur la flexibilité ad-hoc, optimisant pour les cas d'usage identifiés.

# L'Émergence du NewSQL



## Convergence des Paradigmes

Tentative de réconciliation entre les garanties ACID des SGBDR et la scalabilité horizontale du NoSQL.

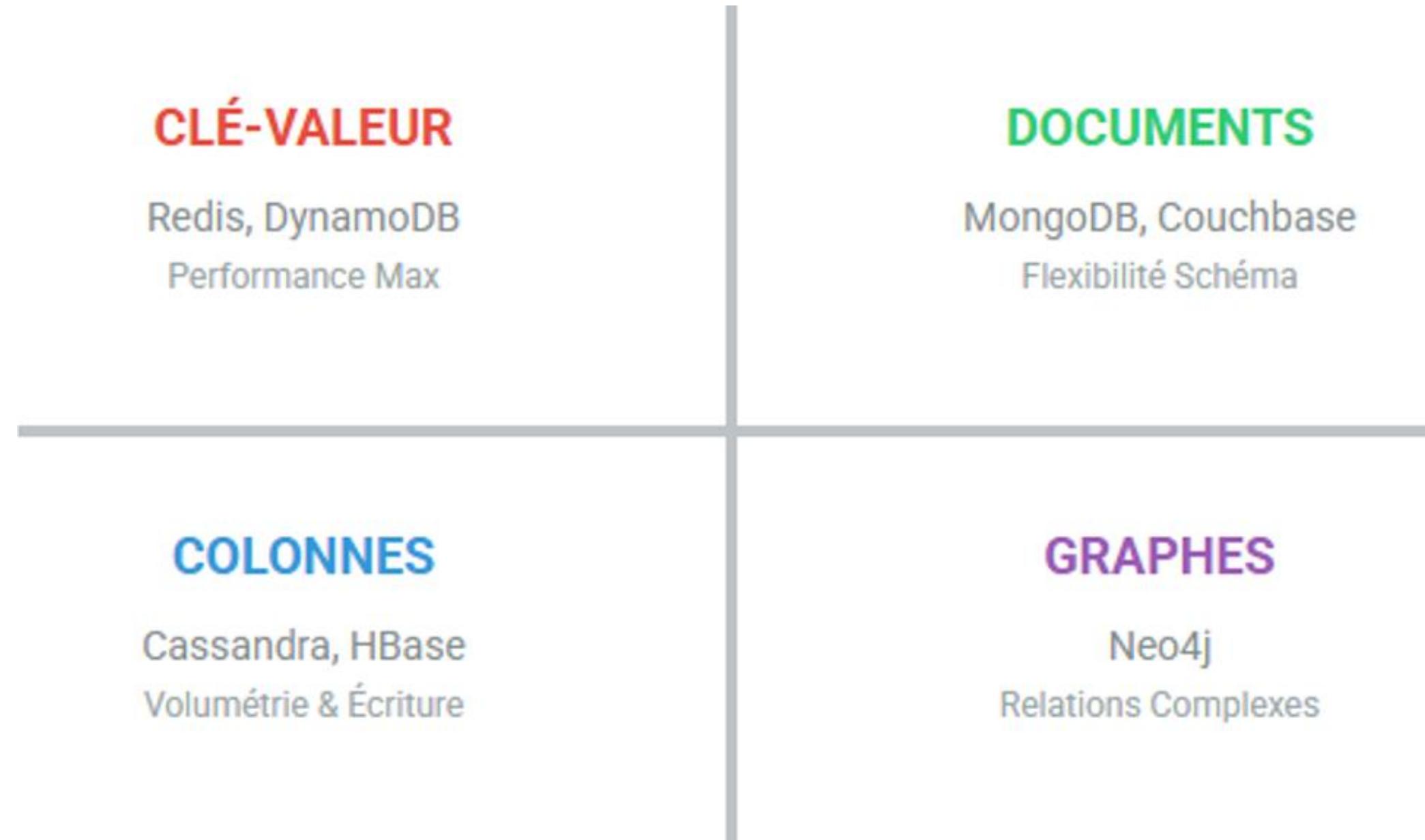
### Technologies Représentatives

- **Google Spanner** : Cohérence externe via horloges atomiques TrueTime
- **CockroachDB** : Distributed SQL avec protocole Raft
- **VoltDB** : OLTP in-memory distribué

❑ État de l'art actuel avec complexité architecturale accrue et compromis subtils entre performances et garanties.

# Taxonomie des Solutions NoSQL

Panorama Technologique



# Matrice de Décision – Architecture

## Critères de Sélection par Type de Données

### Relations Complexes et Traversées

1 Si votre cas d'usage implique des relations multiples, des traversées de graphes, des recommandations ou de l'analyse de réseau → **Graph Databases**

Exemples : Réseaux sociaux, détection de fraude, systèmes de recommandation

### Schéma Dynamique et Requêtes Flexibles

2 Si vous avez besoin d'un schéma évolutif, de requêtes ad-hoc riches et de stockage JSON natif → **Document Stores**

Exemples : CMS, catalogues produits, applications mobiles

### Écriture Massive et Séries Temporelles

3 Si votre priorité est l'ingestion à très haut débit avec compression et requêtes temporelles → **Column-Family**

Exemples : IoT, métriques, logs, télémétrie

# Matrice de Décision - Performance

## Critères de Sélection par Exigences Techniques

Besoin	Technologie	Justification
Latence sub-milliseconde	Key-Value (Redis)	Structure in-memory, accès O(1) direct
Requêtes complexes ad-hoc	Document (MongoDB)	Index secondaires riches, langage requête expressif
Scalabilité horizontale infinie	Column-Family (Cassandra)	Architecture masterless, partitionnement natif
Disponibilité maximale (AP)	Cassandra, DynamoDB	Eventual consistency, multi-datacenter
Cohérence forte (CP)	MongoDB, HBase	Primary writes, synchronisation stricte



# Synthèse et Perspectives

## Points Clés et Transition

### Complémentarité, Non Remplacement

Le NoSQL n'est pas un substitut universel mais une alternative spécialisée pour des besoins spécifiques que les SGBDR ne peuvent satisfaire efficacement.

### Analyse Métier Préalable

La compréhension approfondie des besoins métier, des volumes, de la criticité et des patterns d'accès est indispensable pour choisir l'architecture appropriée.

### Polyglot Persistence

Les architectures modernes combinent souvent plusieurs technologies de stockage, chacune optimisée pour une facette du système global.