



UNIVERSITE DE CORSE
Master informatique 1^{ère}
année
Parcours DFS et DE
2025-2026

Bases de données NoSQL
CH2 – Panorama SGBD noSQL



Evelyne VITTORI
vittori_e@univ-corse.fr

Plan du cours

CH1 – BD
Relationnelles
(Compléments)

- Anomalies de mise à jour
- Normalisation-dénormalisation

CH2 – BD NoSQL

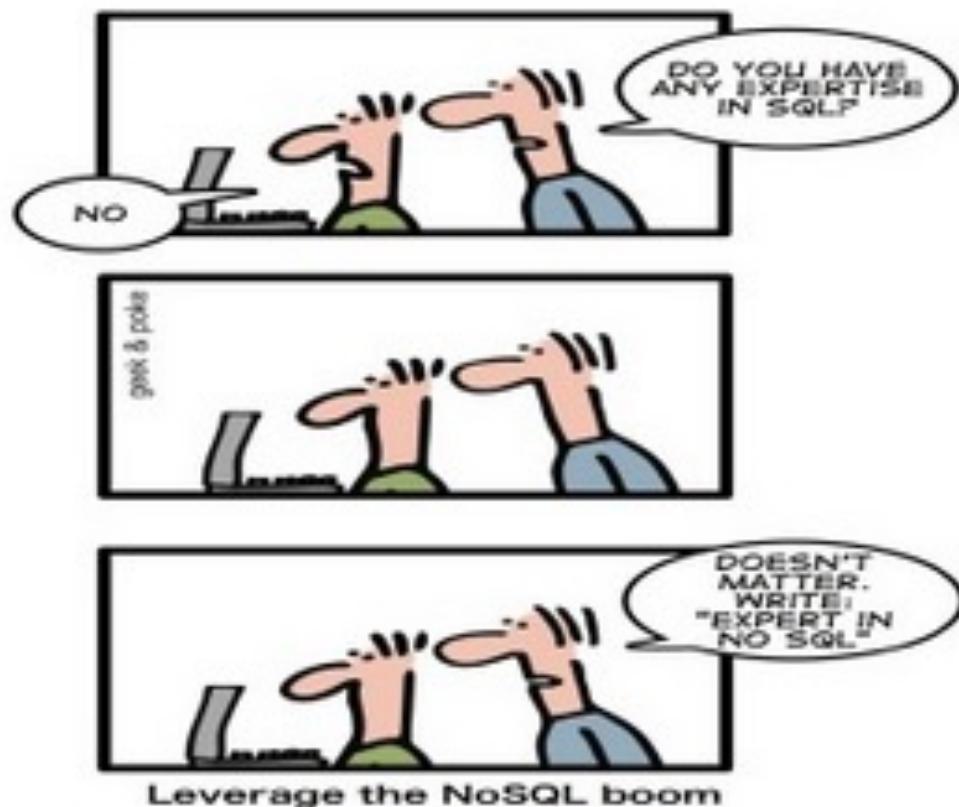
- Principes des BD non relationnelles
- Tour d'horizon des SGBD NoSQL

CH3 - MongoDB

- Initiation à MongoDB

Qu'est ce que le NoSQL?

HOW TO WRITE A CV

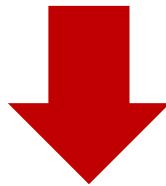


NOT Only SQL

It's about recognizing that for some problems other storage solutions are better suited!

Atouts du paradigme relationnel (SQL)

- La technologie SQL est « bien installée »
 - plus de 20 ans d'expérience : logiciels « matures » et fiables
 - un standard du stockage et de la manipulation des données

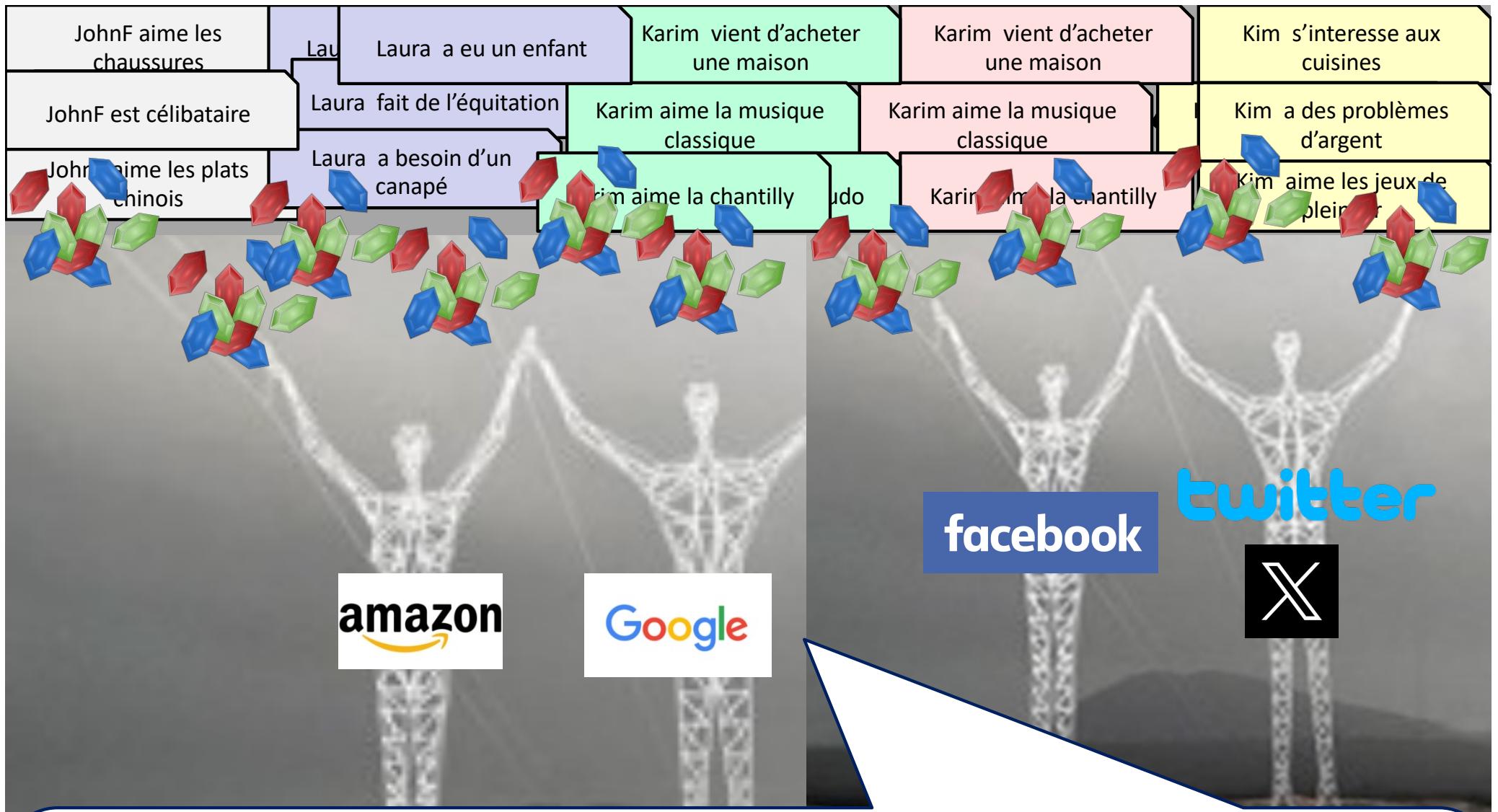


- Coûts de formation maîtrisés
- Outils adaptés aux données critiques
 - gestion de l'intégrité des données
 - notion de transaction

Des limites cependant...

- Solutions mal adaptées aux « Big Data »





Les SGBDr sont trop lents!!
Comment stocker **de plus en plus de données très différentes et pouvoir les traiter de plus en plus vite, vite ?**
L'intégrité, la cohérence, ce n'est pas notre priorité!!!

Pourquoi le relationnel n'est pas adapté?

- Des opérations trop couteuses dans un contexte distribué (ex:**Jointures**)
- Les principes ACID du transactionnel pénalisent les performances :
 - **Atomicity**: transactions « atomiques »: le tout ou rien!
 - **Consistency**: maintien de l'intégrité référentielle
 - **Isolation**: exécution séquentielle des transactions
 - **Durability**: journal des transactions

Performances insuffisantes

Pourquoi le relationnel n'est pas adapté?

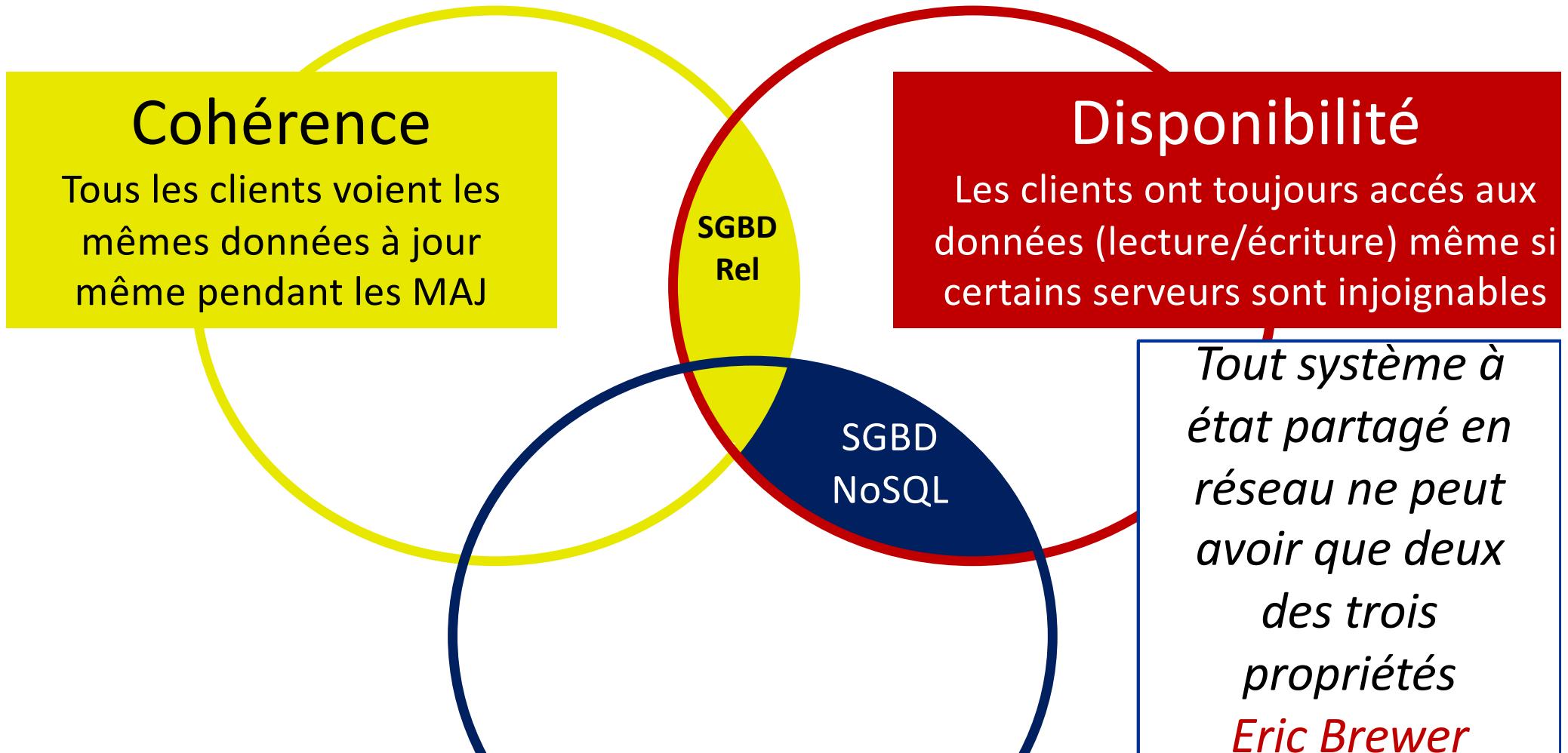
- La structure des données (tables, colonnes) doit être **déclarée préalablement** à son utilisation
- Une table ne peut pas rassembler des **données hétérogènes** (structures différentes)

Manque de souplesse

- Les extensions proposées (ex: extension orientées objets) ne sont pas efficaces en termes de performance

Le Théorème CAP

Consistency Availability Partition tolerance



Résistance au morcellement (*Partition tolerance*)

Le système fonctionne malgré la répartition physique des données

Les besoins liés aux big data

- Manipuler **simplement** et **rapidement** des **énormes volumes** de données **très hétérogènes**
 - Répartition du stockage
 - Distribution des traitements
- Les points clés pour les données:
 - assurer la **disponibilité** des données (continuité des services en cas de panne ou d'indisponibilité)
 - gérer les **montées en charge**



Data-centers

Les 3 V du Big Data:
Volume, Vélocité, Variété

Scalabilité: au cœur du Big Data

Extensibilité, capacité à évoluer en cas de montée en charge

- **Scalabilité horizontale** (« externe ») :

- possibilité d'ajouter des serveurs supplémentaires
- Une priorité pour les SGBD NSQL

- **Scalabilité verticale** (« interne »):

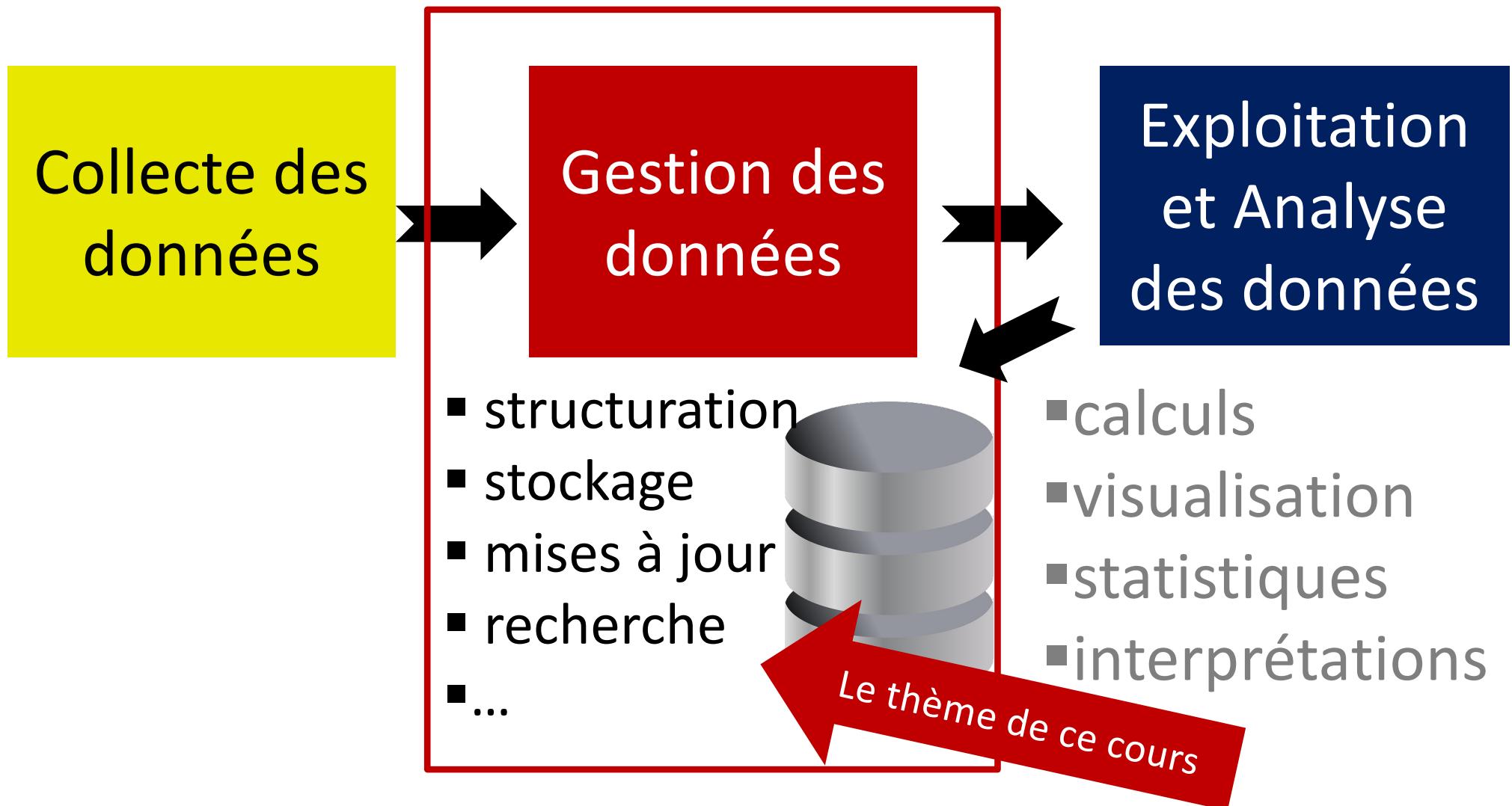
- possibilité d'ajouter des ressources supplémentaires à un serveur (ajout de processeurs, RAM, disques...).

BigData : données ou Informations ?

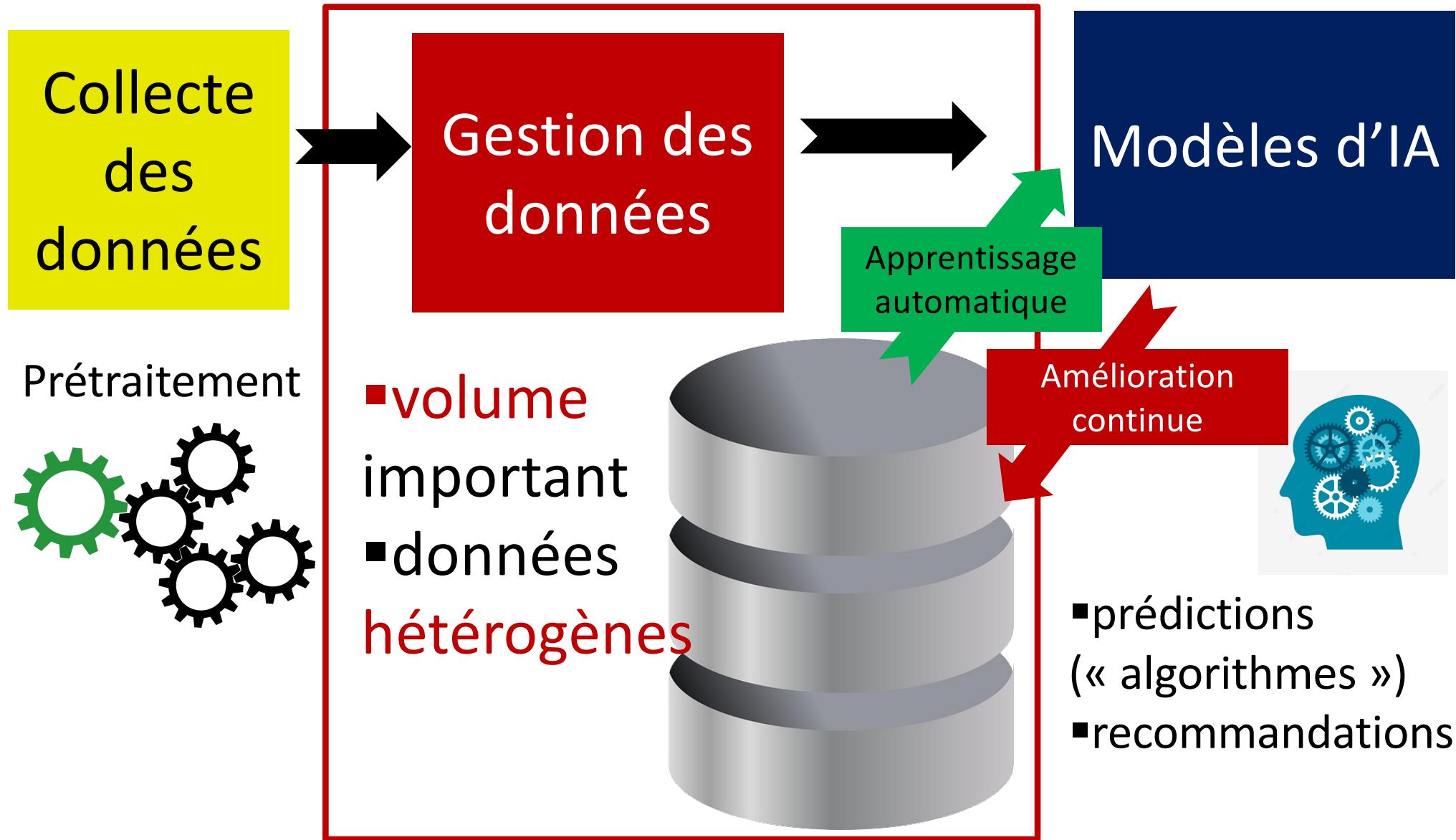
- **Donnée**= enregistrement d'un fait du monde réel
- **Information** = ce que l'on peut déduire d'un ensemble de données



Les données jouent toujours un rôle central !



Spécificités des données dans un contexte IA



Principes des SGBD NoSQL

NoSQL= Not Only SQL

- Structure **non relationnelle**
- Structure **évolutive**
- **Simplicité** d'utilisation
- Langage de **requête spécifique**
 - souvent limité
 - Et surtout: pas de jointure!!
- En général **Open-Source**
- **Scalabilité horizontale**

Les SGBD NoSQL



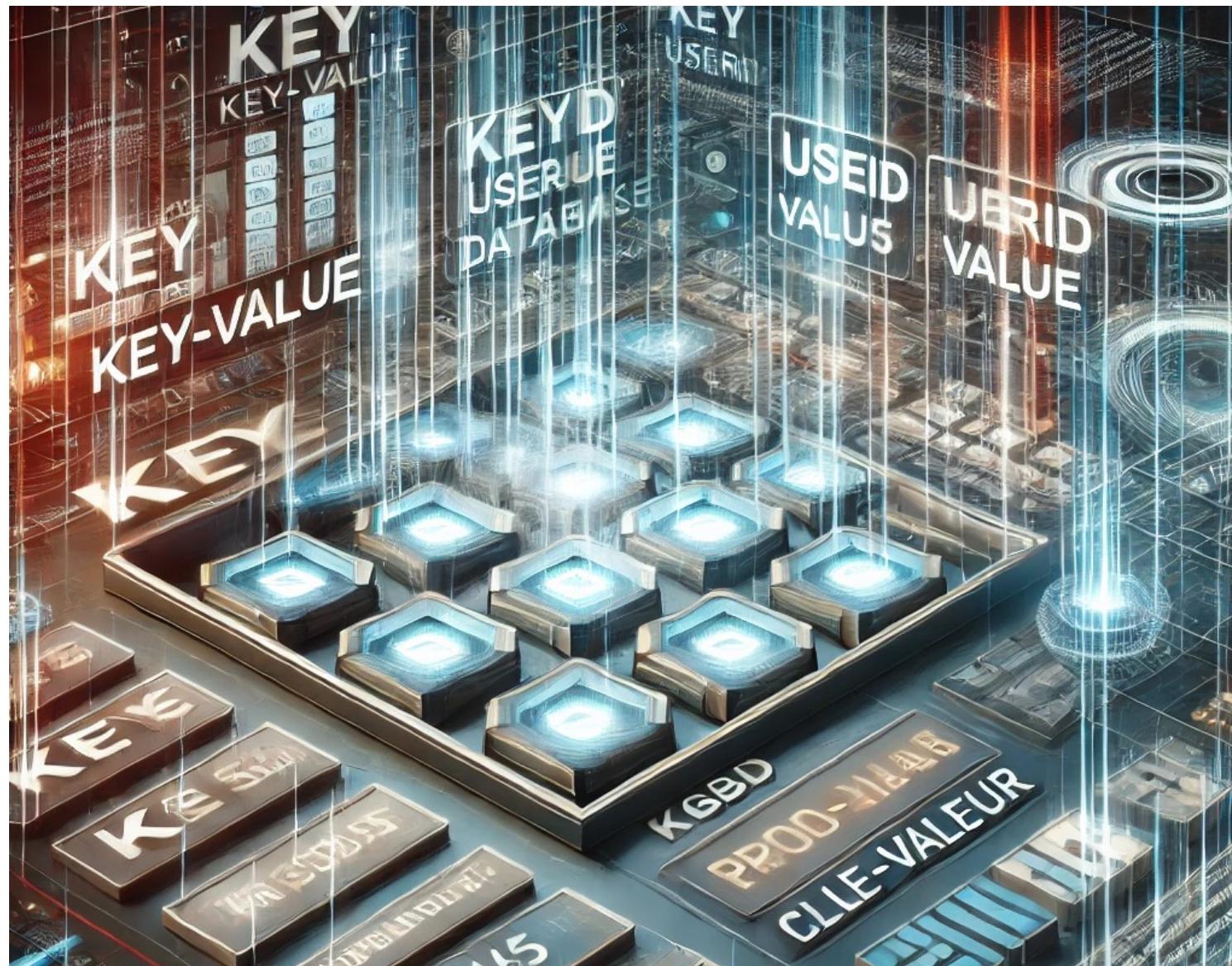
SGBDs NoSQL:

Un bref tour d'horizon

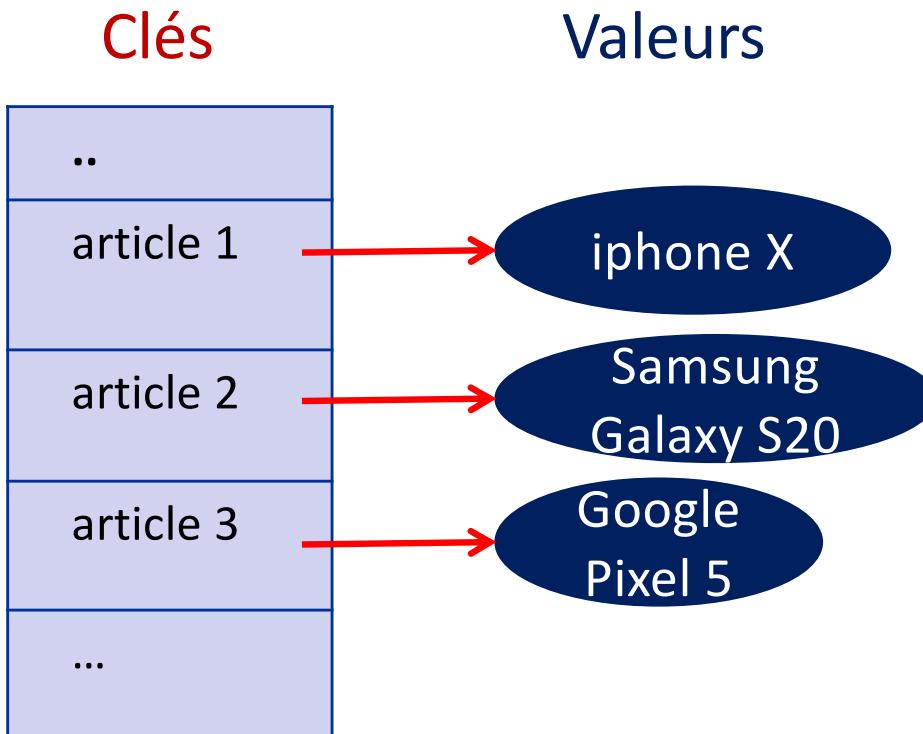
- 5 approches de représentation des données
 - Clé/valeur  redis
 - Colonnes  Apache HBase  Cassandra
 - Graphe  neo4j
 - Documents  mongoDB
 - Vectorielles

De plus en plus de SGBD proposent des approches multi-modèles : ex cle-valeur et document

1- SGBD clé-valeur



BD orientées clé/valeur



BD= gigantesque HashMap!!

- Structure très simple
 - Clé= chaîne de caractère
 - Valeur= n'importe quoi!!
- Clé unique: seul moyen de requêter

opérations CRUD:

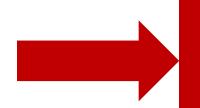
- Create : create(key, value)
- Read : read(key)
- Update : update(key, value)
- Delete: delete(key)

Exemple de requête d'interrogation (SGBD Redis)

La requête « GET article :2 » a pour résultat : «Samsung Galaxy S20 »

Bases orientées clé/valeur

- Performances exceptionnelles en lecture/écriture
- Approche de requêtage très limitée (CRUD uniquement)

 fort besoin de programmation

- Scalabilité horizontale très élevée

Systèmes adaptés aux applications limitées à des requêtes très simples et pour lesquelles l'intégrité des données n'est pas significative

Principaux SGBD clés-valeur

■ Redis

- Données en RAM!! Très efficace
- Un petit tutoriel très simple pour se faire une idée

<http://www.touilleur-express.fr/2013/04/03/redis/>



■ Amazon DynamoDB



- SGBD cloud (DbaaS « DataBase As A Service »)
- Charges élevées: « trillions » de requêtes par jour.
- Supporte également les documents.
- Un tutoriel pour être opérationnel en 10mn

<https://aws.amazon.com/fr/getting-started/hands-on/create-nosql-table/>

Principaux SGBD clés-valeur



■ **Azure Cosmos DB (Microsoft)**

- SGBD cloud (DbaaS « DataBase As A Service »)
- Plusieurs API disponibles

<https://azure.microsoft.com/fr-fr/services/cosmos-db/#overview>

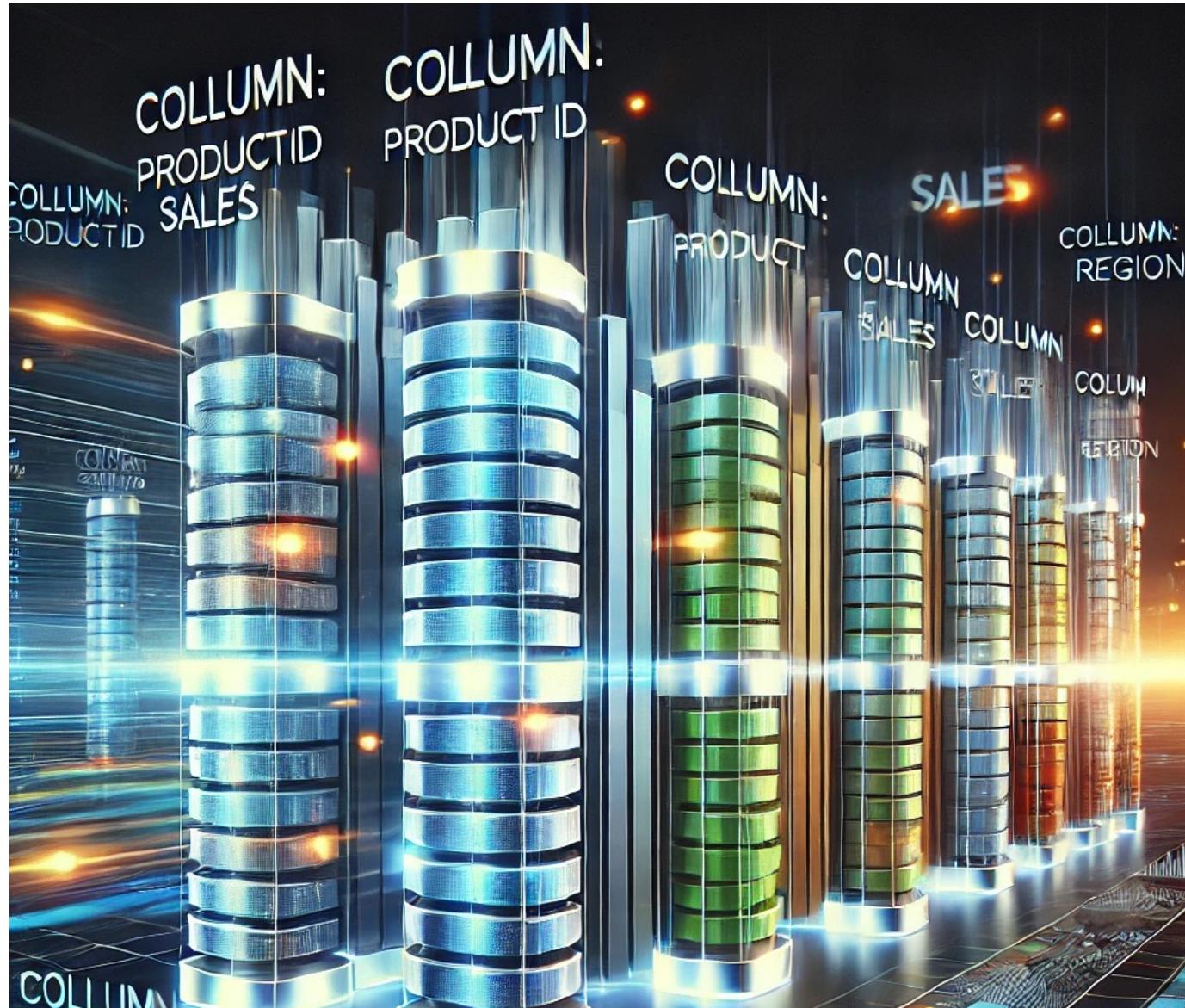
■ **Voldemort**

- Distribuée, tolérante aux pannes
- développée et utilisée par LinkedIn

<https://www.project-voldemort.com/voldemort/>



2 - SGBD colonnes



BD orientées colonnes

- **Colonne**= couple nomColonne-valeur
- Table = **familles de colonnes (column family)**
 - ligne= clé (**rowkey**) + couple colonne-valeur
 - nombre de colonnes dynamique et pouvant varier d'un enregistrement à l'autre
- Structure **optimisée pour l'accès par colonne**
 - les données de chaque colonne sont consécutives dans le système de stockage
 - Accés rapide get (*nomfamille, nomcolonne*)
 - MAJ facilitées

On parle aussi de Big tables

Exemple de table orientée colonnes

table relationnelle
ETUDIANT

ID	NOM	TELFIXE	TELMOBILE
1	Jean	0495354477	0654221130
2	Paul	0495321075	NULL
3	Marie	NULL	NULL



Column family ETUDIANT



rowkeys

FAMILY	ROW	COLUMN	VALUE
ETUDIANT	1	NOM	Jean
ETUDIANT	1	TELFIXE	0495354477
ETUDIANT	1	TELMOBILE	0654221130
ETUDIANT	2	NOM	Paul
ETUDIANT	2	TELFIXE	0495321075
ETUDIANT	3	NOM	Marie

Bases orientées colonnes

- Adaptées aux traitements **d'analyse de données** type **Map-Reduce**
- Couplage avec des logiciels spécifiques
 - Google analytics, ...

utilisés pour les Modèles prédictifs

Principaux SGBD

- Le précurseur : **BigTable** de Google
- HBASE
- CASSANDRA



3- SGBD graphes

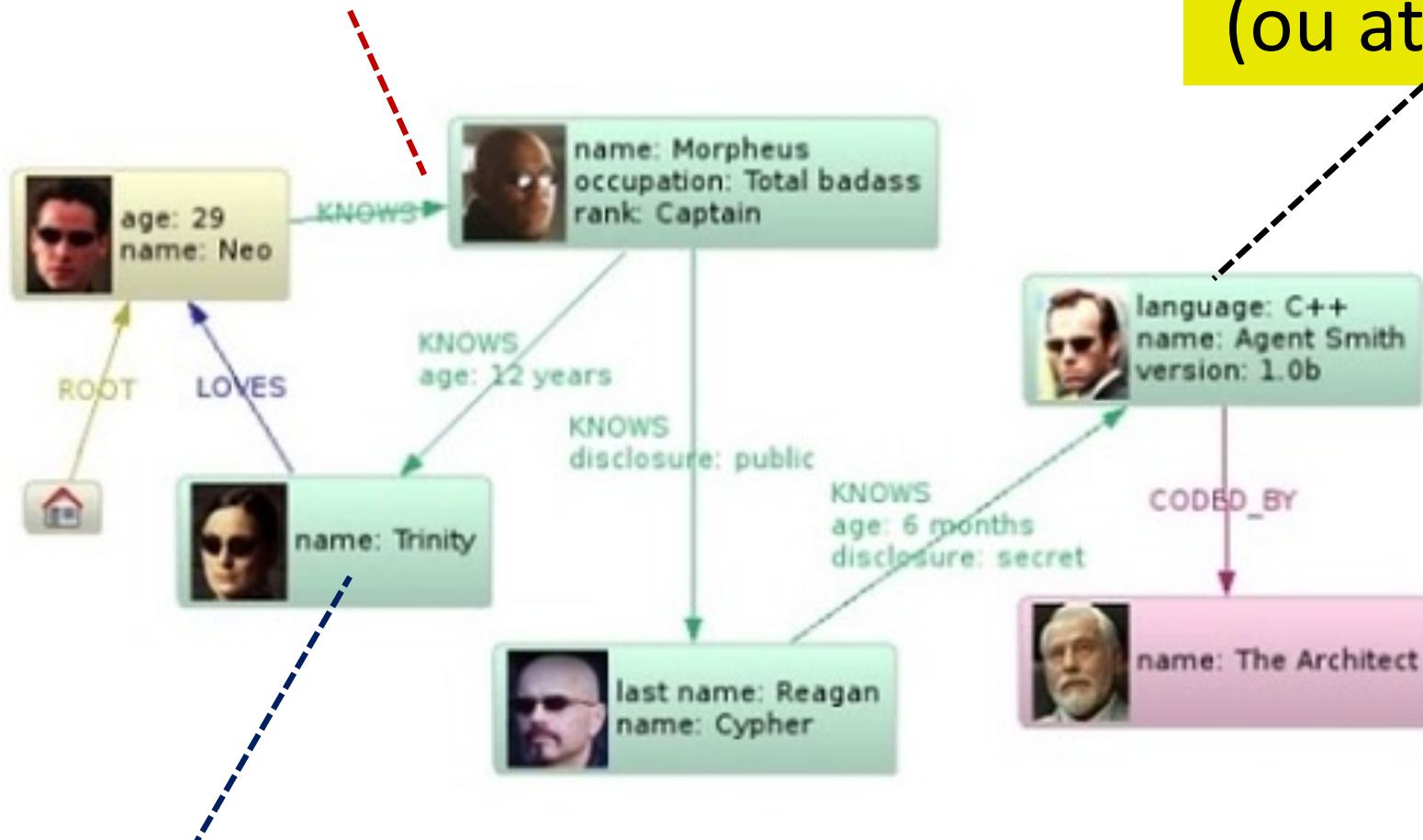


BD orientées graphes

- Structures de données basées sur la **théorie des graphes**
- Eléments de modélisation
 - **nœuds + arcs + propriétés**
- Principe de requêtage:
 - Parcours des nœuds le long des arcs
 - Algorithmes classiques de parcours de graphe
 - ex: plus-court-chemin
- Fonctionnalités transactionnelles

Bases orientées graphes

relation (ou arc ou arête)



propriété
(ou attribut)

nœud (ou sommet)

source image: <http://www.infoq.com/fr/articles/graph-nosql-neo4j>

Bases orientées graphes

- Adaptées aux problématiques liées à la modélisation et au parcours d'un réseau
 - ex: réseaux sociaux, cartographie
- Performances bien supérieures à un SGBDR qui nécessite des opérations complexes et couteuses pour manipuler ce type d'objets

Principaux SGBD

■ Neo4j <http://neo4j.com/>



■ OrientDB



■ Titan

aussi utilisés pour le
calcul prédictif

4 - SGBD documents



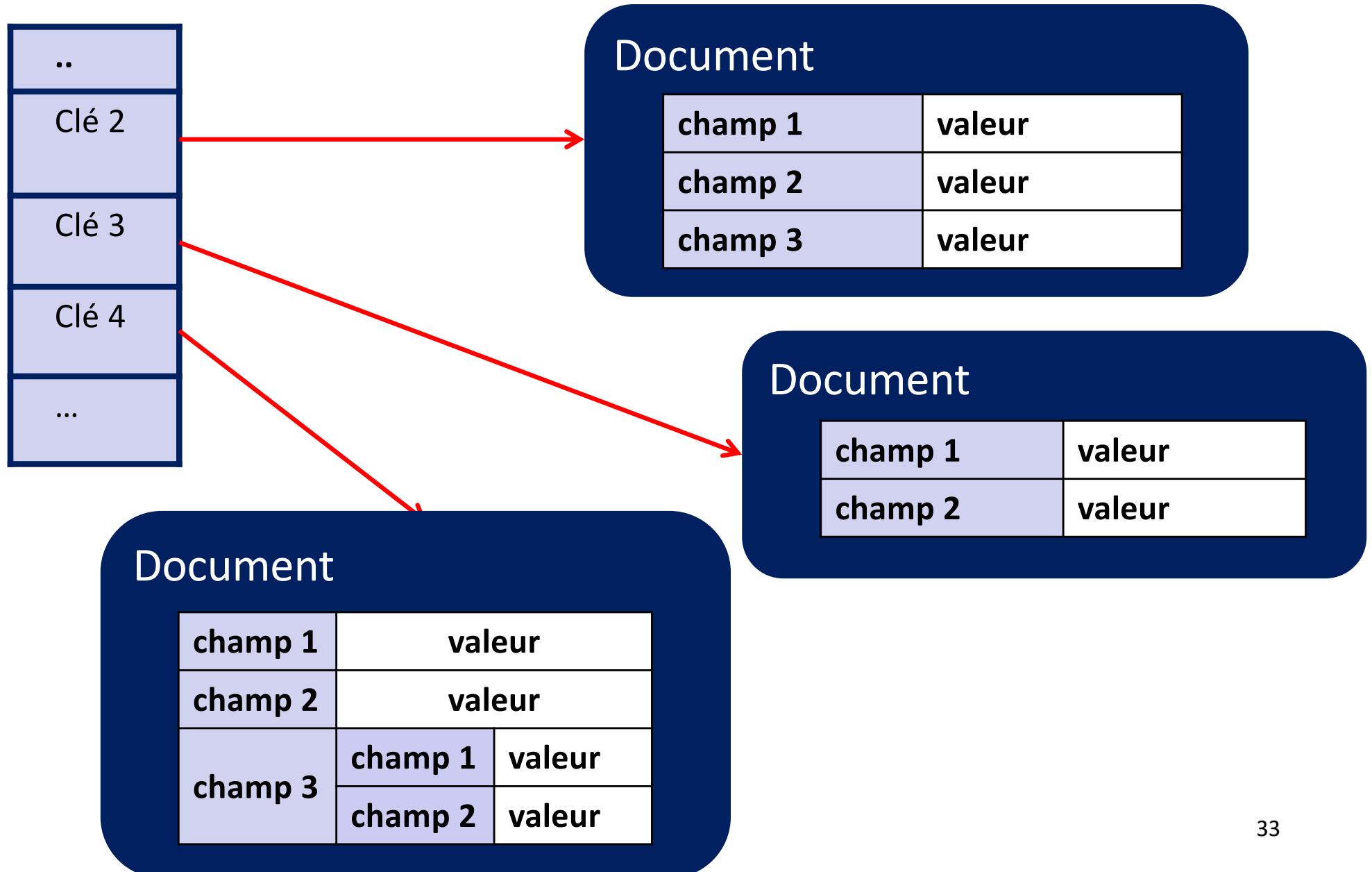
BD orientées documents

- Collections de documents
- Un **document** = ensemble (champ, valeur)
 - valeur= type simple ou couple (champ, valeur)
- Base « schemaless »
 - La structure des documents n'a pas à être déclarée
- Stockage au format JSON ou xml
- Requêtage sur les valeurs des champs
- Adapté aux besoins des CMS

version plus élaborée du paradigme clé/valeur

MongoDB,
CouchDB,
Terrastore

Bases orientées documents



Bases orientées documents

- Le leader:



mongoDB

- CouchDB (Apache) :

- Collections de documents au format JSON
- Requetage avec Map/Reduce



Cas particulier de SGBD document :

Moteurs de recherche

(search engines)

- SGBD orienté sur la recherche d'informations par **indexation** du contenu textuel
- Principales spécificités:
 - Recherche dans du texte intégral (**fullText** ou **PleinTexte**)
 - Outils complémentaires: auto-complétion, suggestions, corrections
 - Analyse et agrégation des résultats de recherche
 - Recherche distribuée

Principe des moteurs de recherche web appliqués à des BD

ElasticSearch



elasticsearch

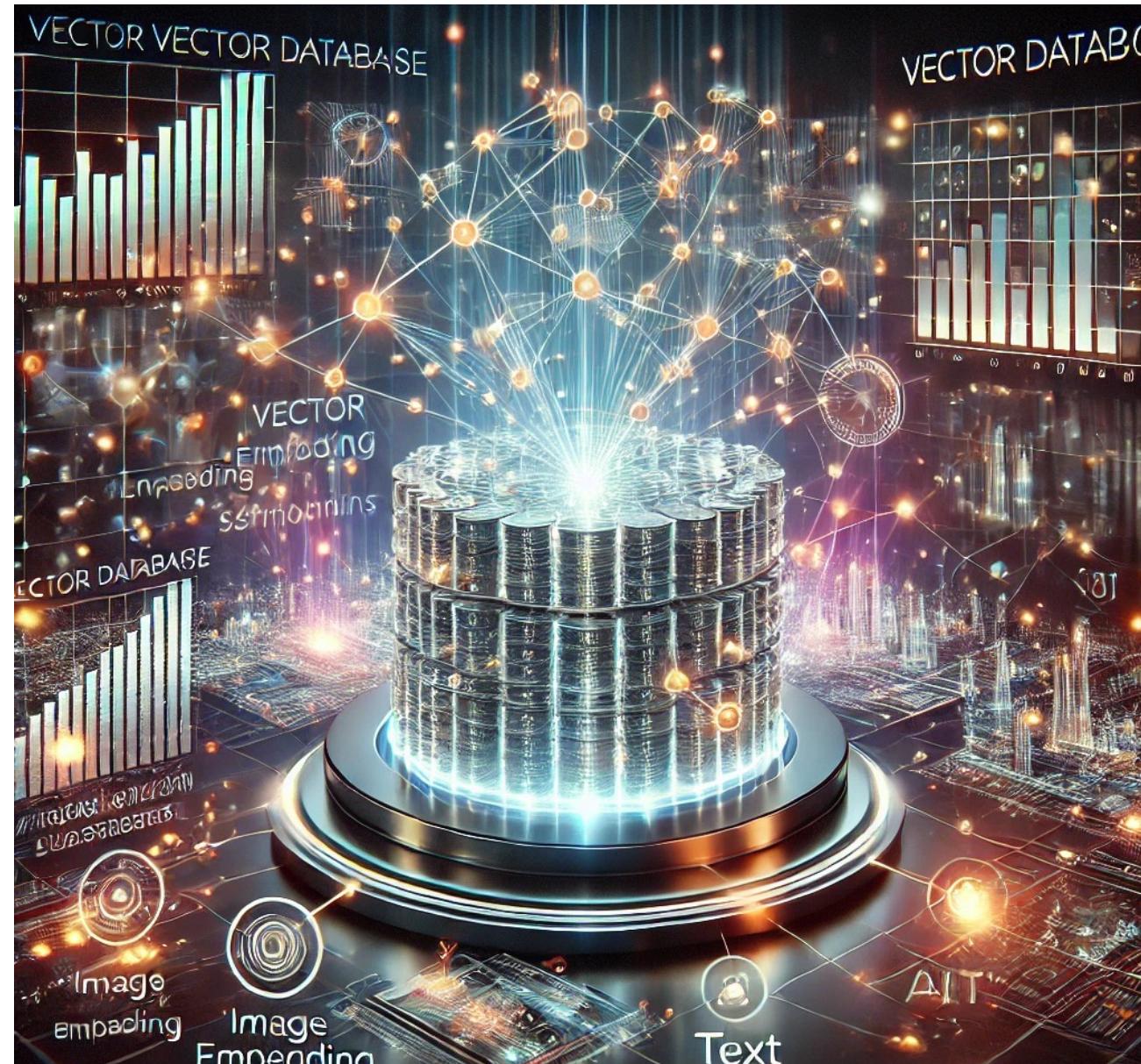
- SGBD orienté document basé sur le Moteur de recherche Lucene (bibliotheque Java permettant l'indexation et la recherche textuelle)

- Recherches très rapides (**index inversés**)
 - Index du contenu :
Un mot est associé à la liste des documents dans lequel il apparaît avec ses positions dans le document
- Interaction par le biais d'une API REST

Diaporama intéressant sur la Comparaison entre MongoDB et ElasticSearch

<https://fr.slideshare.net/sebprunier/mongodb-et-elasticsearch-meilleurs-enemis>

5 - SGBD Vectoriels



SGBD vectoriels (Vector DBMS)

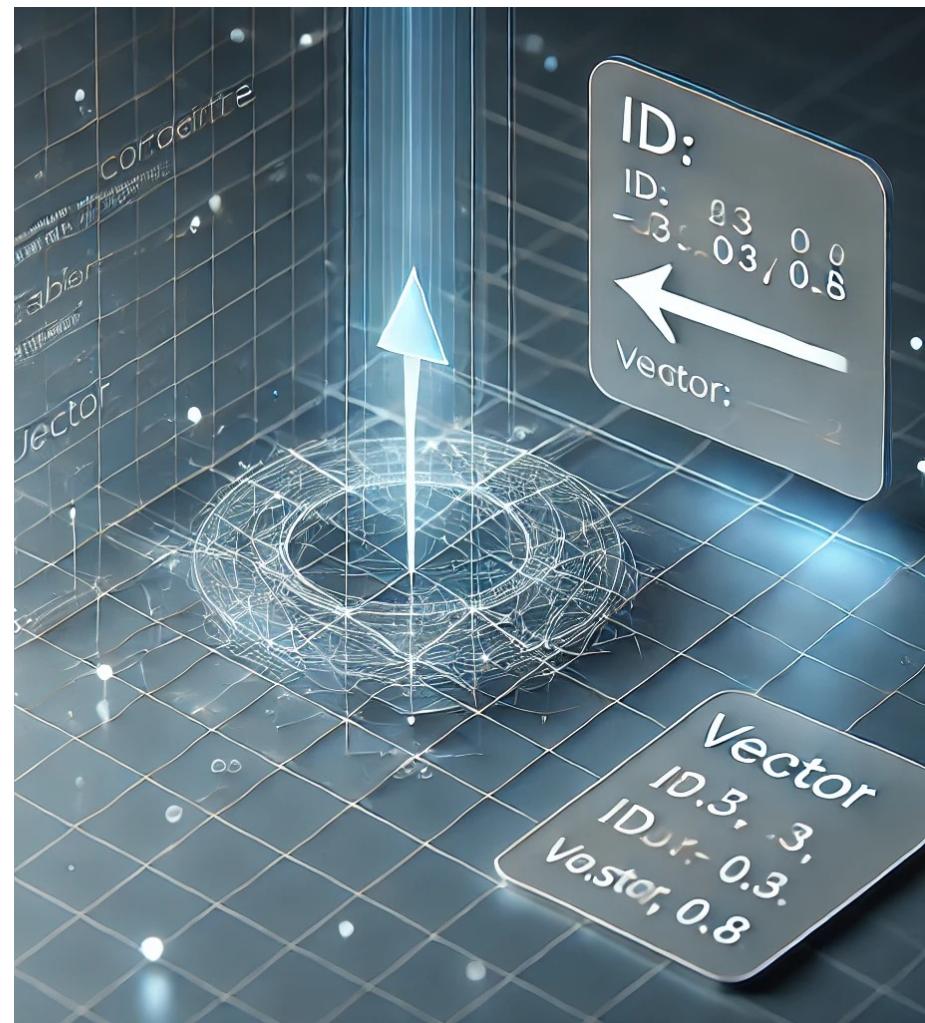
- Spécialisés pour stocker, indexer et rechercher des **vecteurs** dans un espace multidimensionnel
- Optimisés pour des opérations sémantiques
 - Recherche **par similarité**
 - **Classification** par catégories
 - **Clustering** : identification de regroupements
- Applications courantes
 - Support aux systèmes d'intelligence artificielle

Indexation (KD-Tree) et algorithmes spécifiques

Stockage d'embeddings

Notion de vecteur

- Un **vecteur** est une liste de nombres
- Représentation des caractéristiques ou relations d'une donnée dans un espace mathématique multidimensionnel
- Exemple : Un vecteur $[1.2, -0.5, 3.0]$



Qu'est-ce qu'un embedding?

- Vecteur particulier
généré par un modèle
d'IA
- Représente un objet
- *Exemples*

- Texte "chien"

→ Word Embedding : [0.12, -0.34, 0.56, ...]

- Image d'un chat

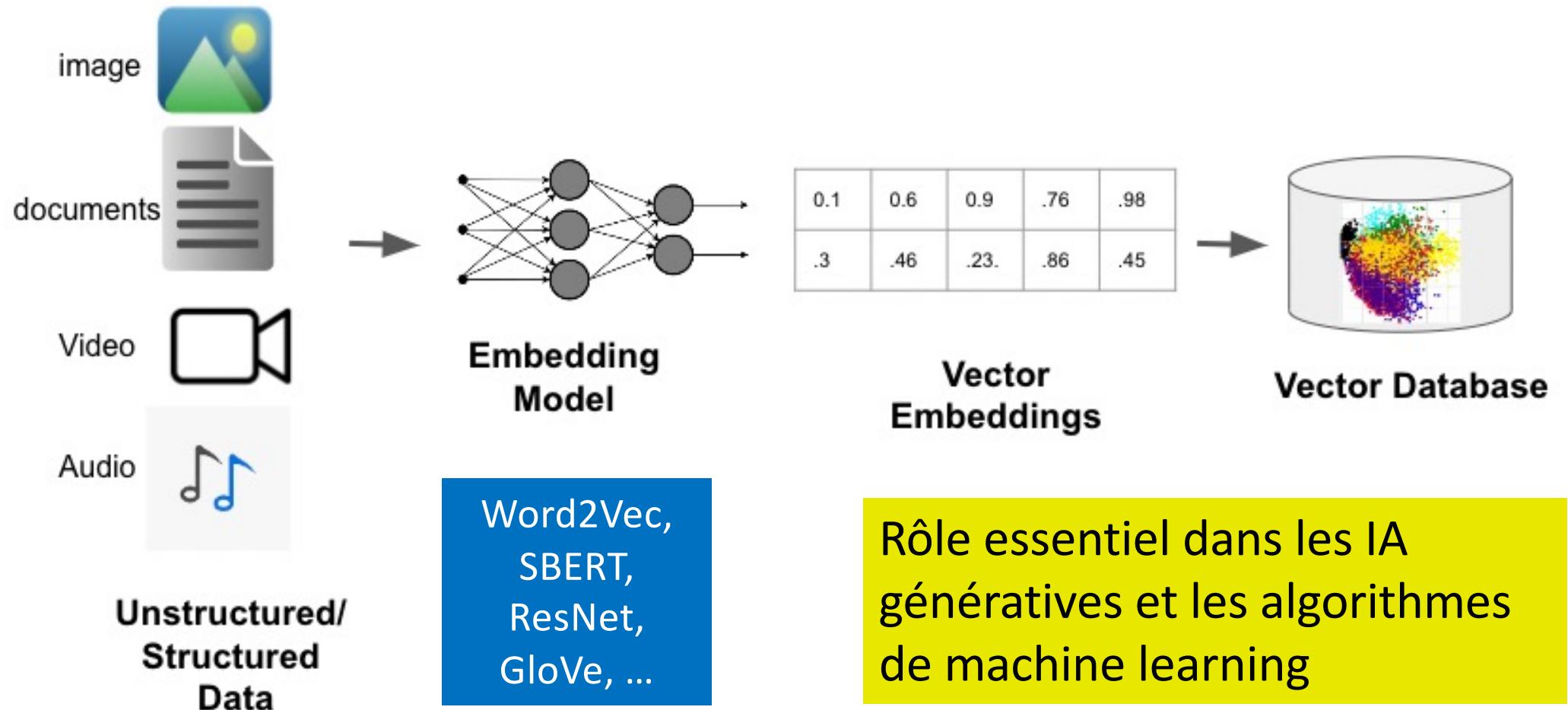


→ Image Embedding : vecteur unique
représentant ses traits visuels



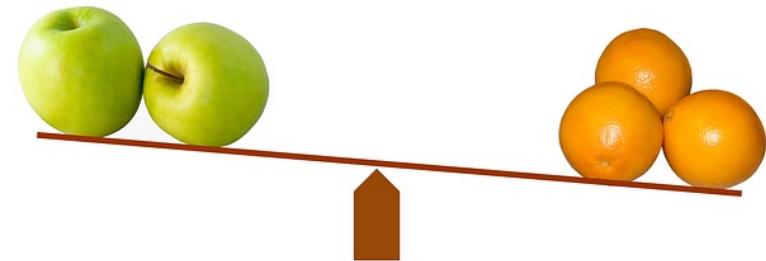
Source : <https://generationia.flint.media/p/c-est-quoi-un-embedding/>

BD Vectorielles et embeddings



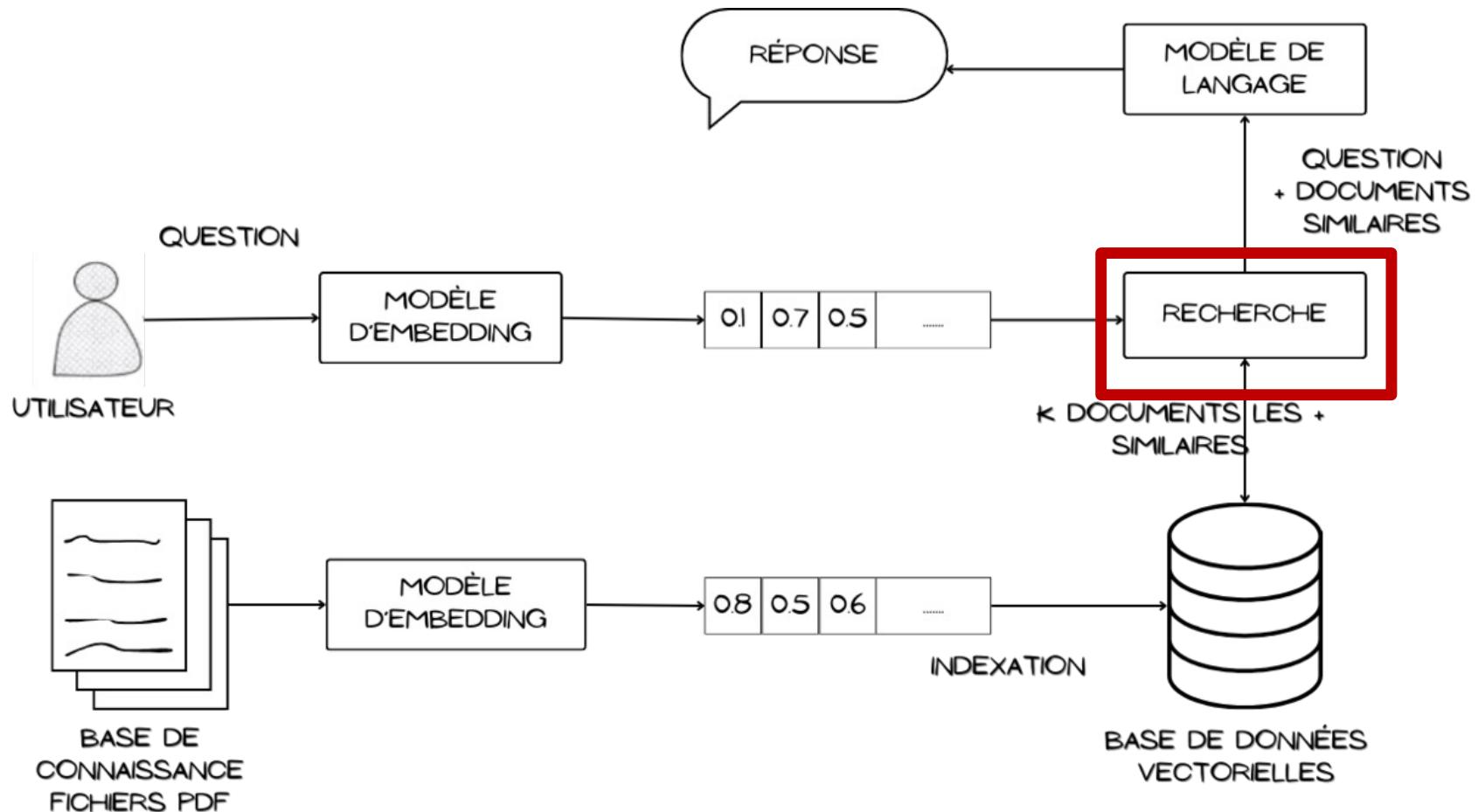
Recherche par similarité

- Principe
 - retrouver les éléments les plus proches d'un vecteur dans un espace vectoriel.
- Mesure de la proximité entre deux vecteurs par calcul de distance
 - cosinus, euclidienne, etc..
- Avantages
 - Approche rapide et efficace pour des recherches complexes
 - Idéal pour des données non structurées comme des images ou du texte



Embeddings et RAG

(Retrieval Augmented Generation)



Principaux SGBD vectoriels

include secondary database models 22 systems in ranking, September 2025

Rank			DBMS	Database Model	Score		
Sep 2025	Aug 2025	Sep 2024			Sep 2025	Aug 2025	Sep 2024
1.	1.	1.	Elasticsearch	Multi-model 	118.26	+3.99	-10.53
2.	2.	↑3.	OpenSearch	Multi-model 	19.93	+1.45	+3.62
3.	3.	↓2.	Couchbase	Multi-model 	12.58	-0.09	-4.16
4.	4.	4.	Kdb	Multi-model 	8.18	+1.21	+0.45
5.	5.	5.	Pinecone	Vector	5.98	+1.03	+2.95
6.	6.	6.	Milvus	Vector	4.52	+0.69	+1.51
7.	7.	↑10.	Qdrant	Vector	3.44	+0.51	+1.91
8.	8.	8.	OceanBase	Multi-model 	3.04	+0.22	+1.07
9.	9.	↑11.	Weaviate	Vector	2.91	+0.63	+1.43
10.	10.	↓9.	Chroma	Vector	2.37	+0.15	+0.48



open source

<https://www.trychroma.com/>



Milvus

apparu en 2019

source : <https://db-engines.com/en/ranking/vector+dbms>



Weaviate



```

# Initialiser la fonction d'embedding
custom_embedding_function = CustomEmbeddingFunction()

# Créer le client Persistent
import chromadb
client = chromadb.PersistentClient(path="/Users//.../chromadb")

# Créer une collection avec la fonction d'embedding
collection = client.create_collection(name="livres", embedding_function=custom_embedding_function)

# Ajouter des descriptions de livres/films
collection.add(
    documents=[
        "Un jeune sorcier découvre ses pouvoirs et affronte un sorcier maléfique.",
        "Un hobbit et ses amis partent en quête pour détruire un anneau maléfique.",
        "Une enquête sur un meurtre mystérieux dans une ville futuriste.",
        "Des astronautes découvrent des créatures dangereuses sur une planète inconnue.",
        "Des super-héros s'unissent pour sauver la Terre d'une menace cosmique."
    ],
    ids=["id1", "id2", "id3", "id4", "id5"],
    metadatas=[
        {"titre": "Harry Potter", "genre": "Fantastique", "auteur/realisateur": "J.K. Rowling"},
        {"titre": "Le Seigneur des Anneaux", "genre": "Fantasy", "auteur/realisateur": "J.R.R. Tolkien"},
        {"titre": "Blade Runner", "genre": "Science-fiction", "auteur/realisateur": "Ridley Scott"},
        {"titre": "Alien", "genre": "Science-fiction", "auteur/realisateur": "Ridley Scott"},
        {"titre": "Avengers", "genre": "Action", "auteur/realisateur": "Marvel Studios"}
    ]
)

# Requête
query_text = "Une équipe de héros lutte contre une menace qui pourrait détruire la planète"
results = collection.query(
    query_texts=[query_text],
    n_results=3,
    include=["documents", "distances", "metadatas"]
)

# Afficher les résultats
for doc, metadata, distance in zip(results["documents"][0], results["metadatas"][0], results["distances"][0]):
    print(f"Titre : {metadata['titre']}")
    print(f"Genre : {metadata['genre']}")
    print(f"Auteur/Réalisateur : {metadata['auteur/realisateur']}")
    print(f"Description : {doc}")
    print(f"Distance de similarité : {distance:.3f}")
    print("-" * 50)

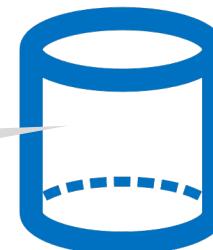
```

Exemple



Exemple de résultat de requête

"Une équipe de héros lutte contre une menace qui pourrait détruire la planète"



 Chroma

collection
Livres

Titre : Avengers

Genre : Action

Auteur/Réalisateur : Marvel Studios

Description : Des super-héros s'unissent pour sauver la Terre d'une menace cosmique.

Distance de similarité : 0.517

Titre : Alien

Genre : Science-fiction

Auteur/Réalisateur : Ridley Scott

Description : Des astronautes découvrent des créatures dangereuses sur une planète inconnue.

Distance de similarité : 0.775

Titre : Blade Runner

Genre : Science-fiction

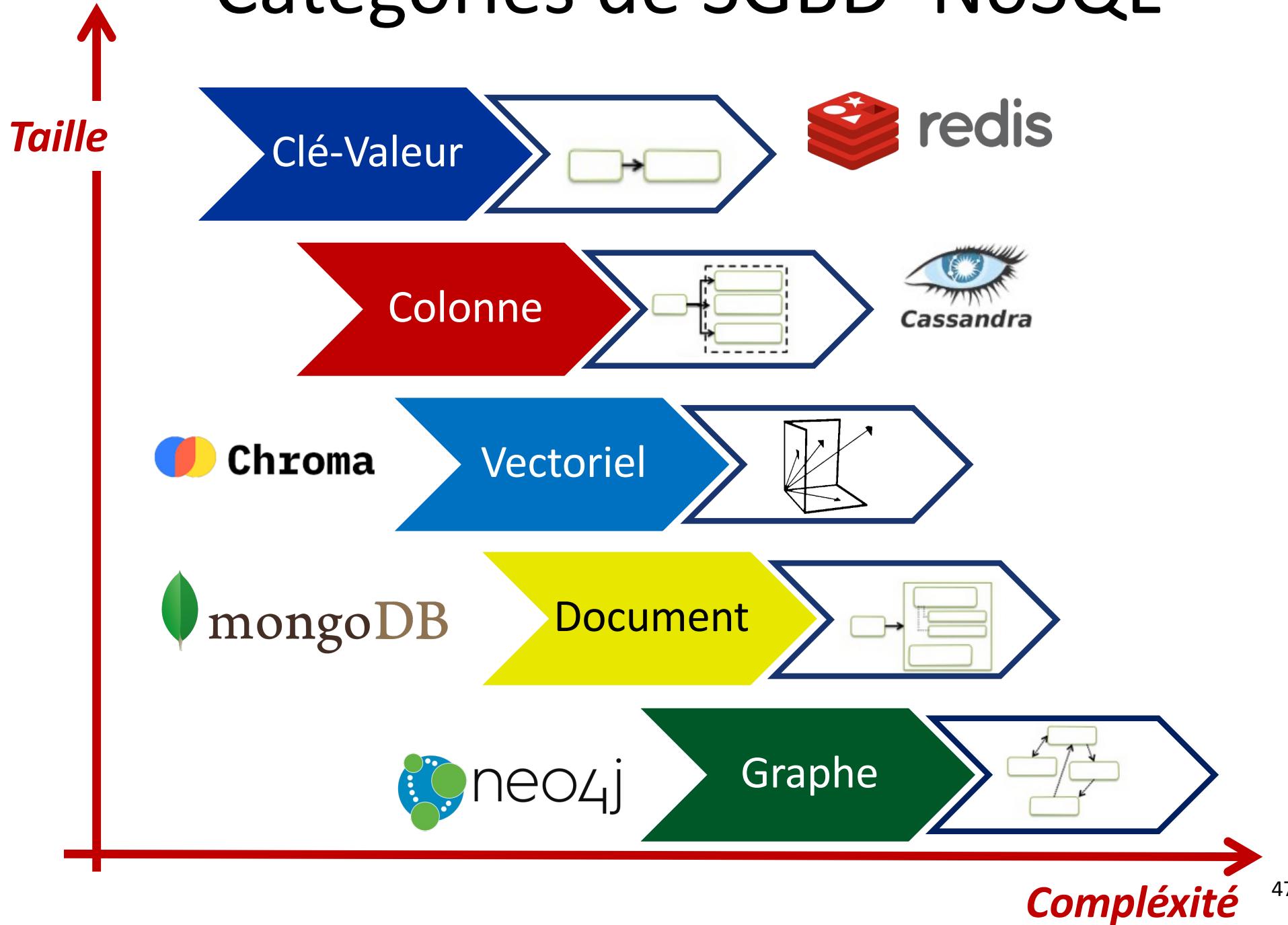
Auteur/Réalisateur : Ridley Scott

Description : Une enquête sur un meurtre mystérieux dans une ville futuriste.

Distance de similarité : 1.048

similarité entre
0 (égalité) et 2 (aucune)

Catégories de SGBD NoSQL

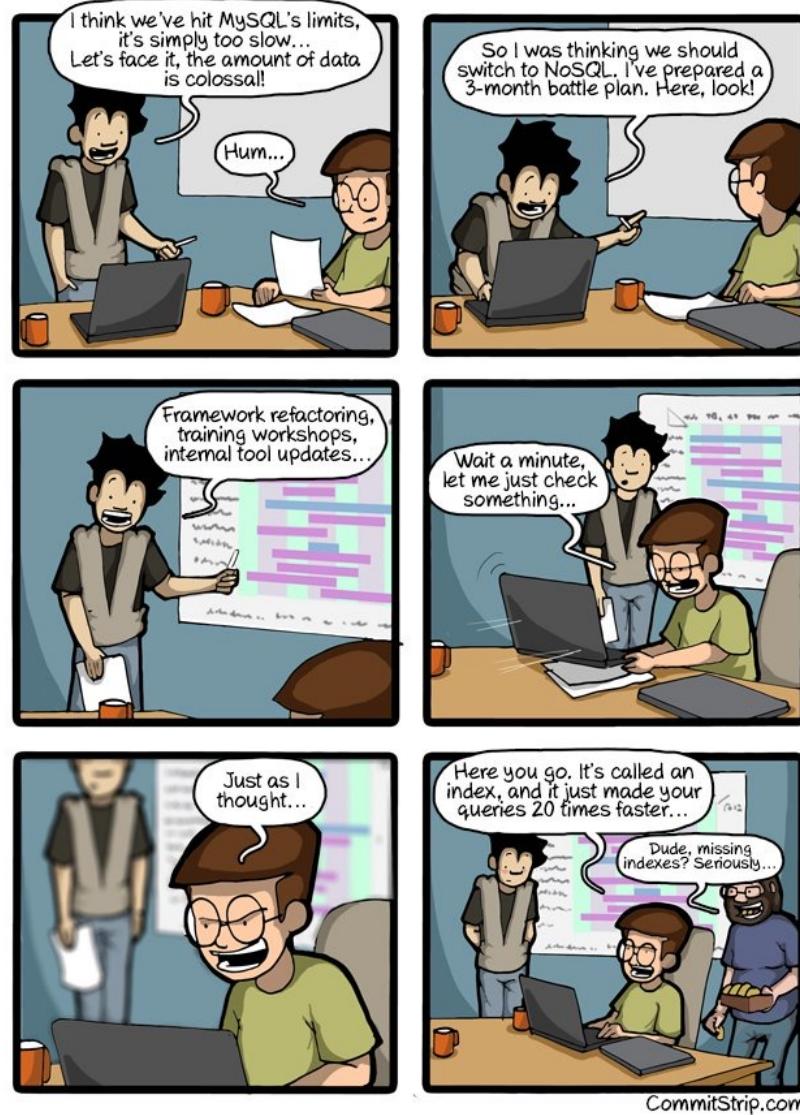


Il ne faut pas s'emballer trop vite!

4 idées fausses sur le NoSQL

- **NoSQL est plus facile pour démarrer**
 - Pas toujours! il faut se méfier de l'apparente facilité
- **Avec NoSQL plus besoin de réfléchir au modèle de données**
 - Attention: catastrophe assurée!
- **NoSQL est toujours plus performant**
 - Cela dépend du contexte
- **NoSQL remplace SQL**

Parfois, il suffit d'ajouter un index dans une BD relationnelle !!



Etat des lieux en septembre 2025

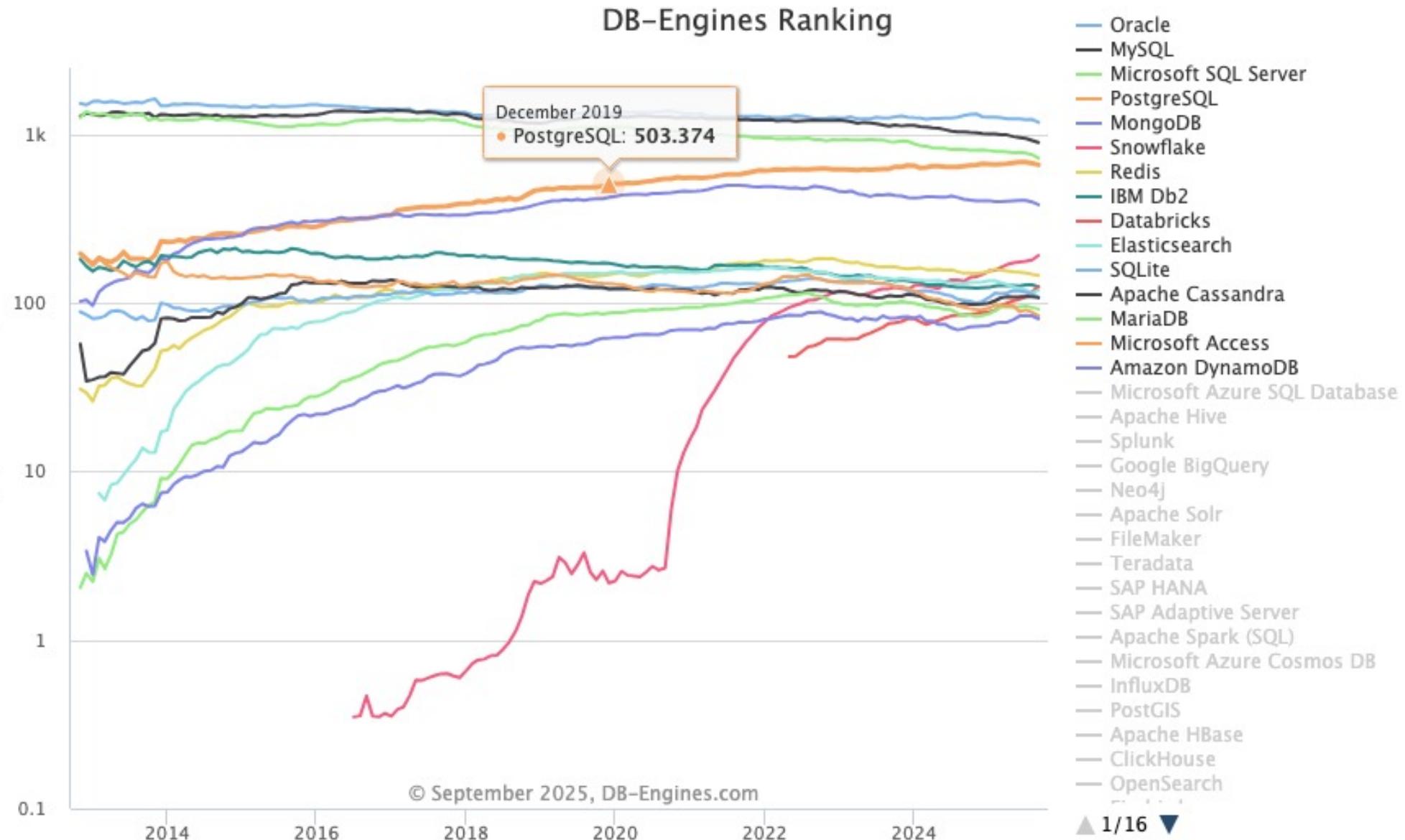
Comparaison de la popularité des SGDB du marché

Source: <http://db-engines.com/en/ranking>

424 systems in ranking, September 2025

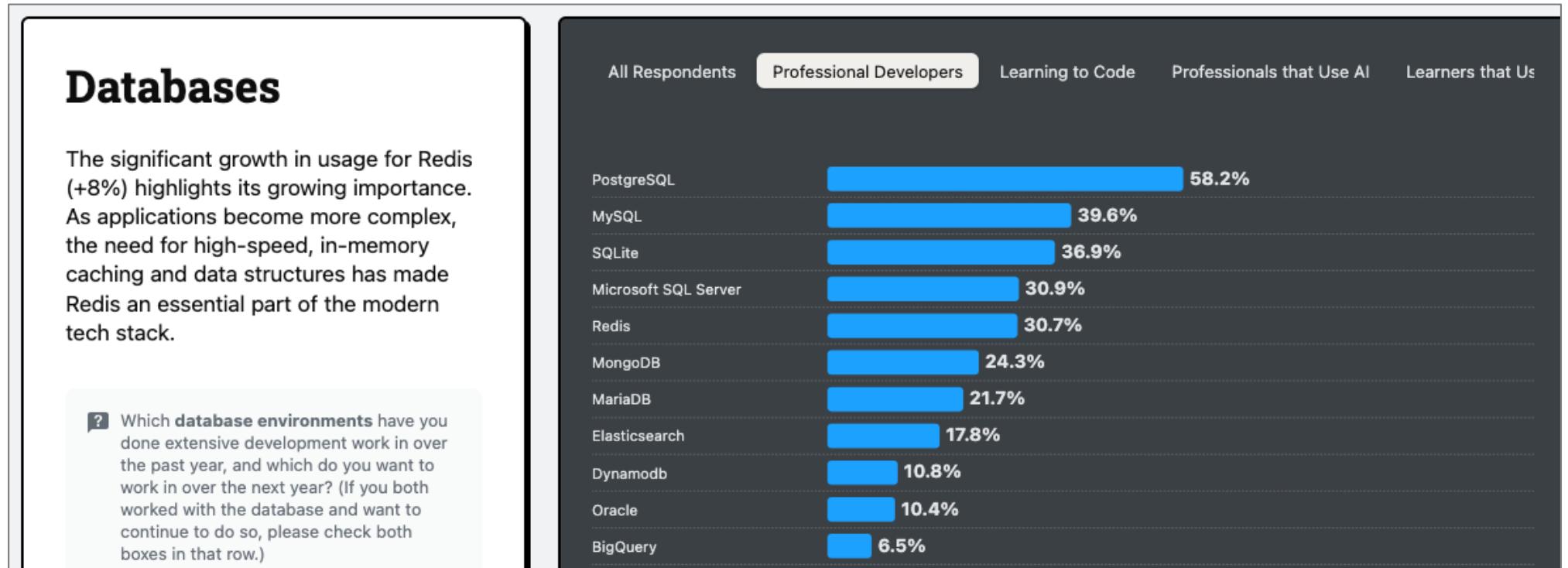
Rank				Database Model	Score		
	Sep 2025	Aug 2025	Sep 2024		Sep 2025	Aug 2025	Sep 2024
1.	1.	1.	Oracle	Relational, Multi-model 	1170.62	-50.08	-115.97
2.	2.	2.	MySQL	Relational, Multi-model 	891.77	-23.69	-137.72
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model 	717.32	-36.84	-90.45
4.	4.	4.	PostgreSQL	Relational, Multi-model 	657.17	-14.08	+12.81
5.	5.	5.	MongoDB 	Document, Multi-model 	380.50	-15.08	-29.74
6.	6.	↑7.	Snowflake	Relational	190.19	+11.29	+56.47
7.	7.	↓6.	Redis	Key-value, Multi-model 	145.17	-2.02	-4.25
8.	8.	↑9.	IBM Db2	Relational, Multi-model 	124.19	-3.12	+1.14
9.	9.	↑14.	Databricks	Multi-model 	124.06	+8.25	+39.82
10.	10.	↓8.	Elasticsearch	Multi-model 	118.26	+3.99	-10.53
11.	11.	↓10.	SQLite	Relational	107.88	-4.72	+4.53
12.	12.	↓11.	Apache Cassandra	Wide column, Multi-model 	106.98	-1.53	+8.04
13.	13.	↑15.	MariaDB 	Relational, Multi-model 	91.46	-2.13	+8.02
14.	14.	↓12.	Microsoft Access	Relational	83.61	-4.15	-10.15
15.	15.	↑17.	Amazon DynamoDB	Multi-model 	80.28	-3.20	+10.22
16.	16.	16.	Microsoft Azure SQL Database	Relational, Multi-model 	79.18	+3.34	+6.23
17.	17.	↑18.	Apache Hive	Relational	76.10	+5.06	+23.02
18.	18.	↓13.	Splunk	Search engine	75.77	+6.00	-17.26
19.	19.	19.	Google BigQuery	Relational	66.00	+0.82	+13.33
20.	20.	↑21.	Neo4j	Graph	53.78	-0.69	+11.10

Evolution de la popularité des SGDB du marché depuis 2014



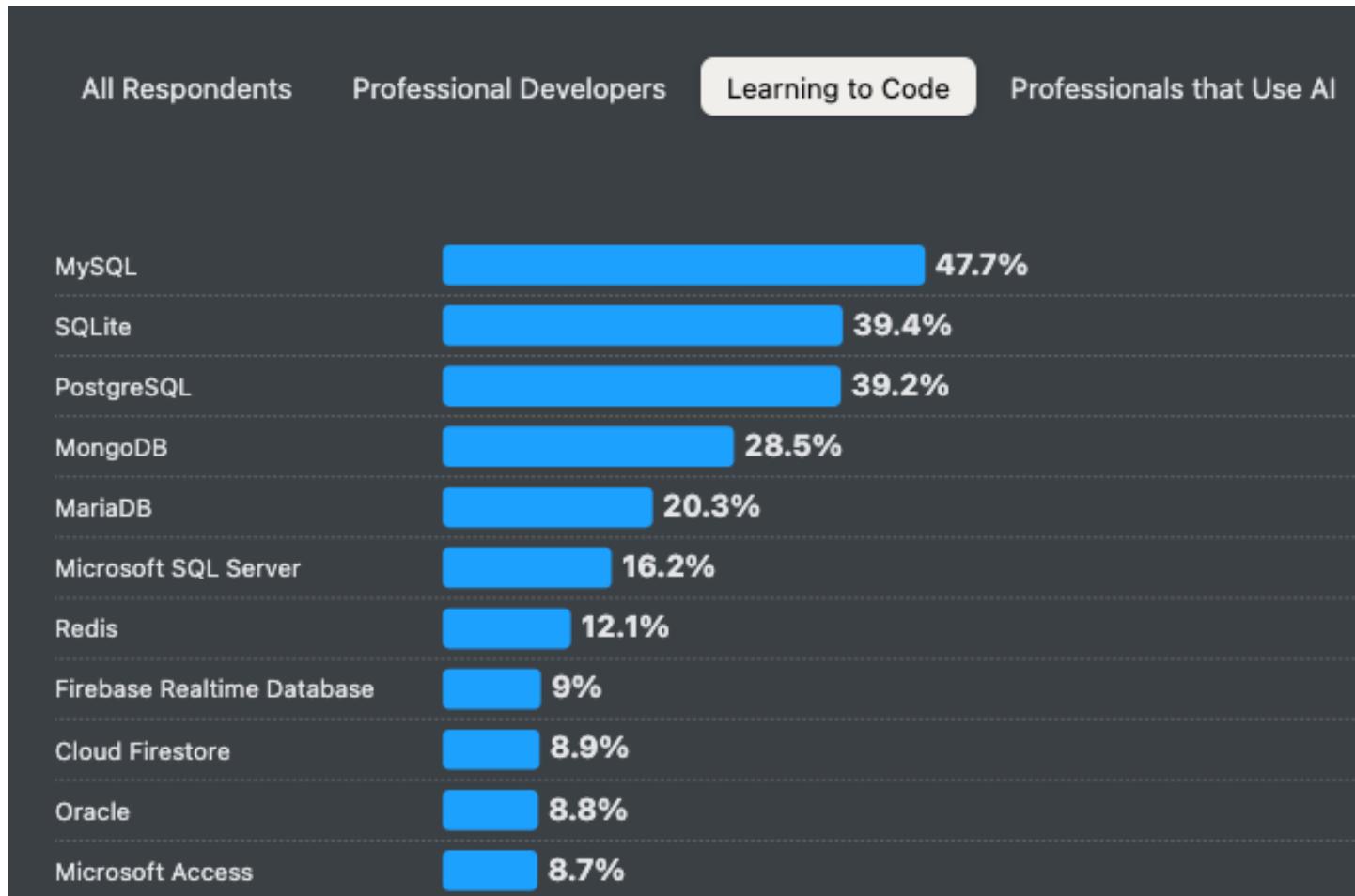
Les choix des développeurs en 2025

1^{er} : PostgreSQL – 2^{ème} : MySQL - ... 6^{ème} : MongoDB



Les choix des débutants en 2025

1^{er}: MySQL – 2^{ème}: SQLite - 3^{ème} : PostgreSQL - 4^{ème}: MongoDB



Quelles remarques sur l'évolution en 2025?



SGBDr et NoSQL: Complémentaires?

	SGBDr	NoSQL (document)
Structure	schéma normalisé	structure dynamique
Liaisons entre données	clés étrangères	Intégration des données corrélées dans l'objet lui-même
Requêtage	déclaratif (SQL) jointures	accès très rapide langages de requête limité programmation nécessaire
Transactions Mises à Jour	simplifiées et sécurisées (intégrité réf. assurée)	intégrité non assurée ou à gérer par programmation
Performances	pénalisées	très élevées

Les extensions NoSQL des SGBBDr



Les SGBDr font de la résistance !

- Mouvement NewSQL :SGBDr étendus par des fonctionnalités NoSQL



	Old SQL	NoSQL	NewSQL
Relational	Yes	No	Yes
SQL	Yes	No	Yes
ACID transactions	Yes	No	Yes
Horizontal scalability	No	Yes	Yes
Performance / big volume	No	Yes	Yes
Schema-less	No	Yes	No

Quelques exemples:

- Nouveaux SGBD :VoltDB, TiDB, AzureCosmosDB, MemSQL , CockroachDB,
- Extensions de sgbdr existants :Vitess (r MySQL), Citus (PostgreSQL), ...

Les extensions NoSQL des SGBDR

La plupart des SGBDr proposent aujourd’hui des outils de gestion de documents JSON de type document Store:

- Oracle Document Store et API SODA
 - <https://www.oracle.com/database/technologies/appdev/json.html>
- MySQL Document Store
 - <http://dasini.net/blog/2019/04/02/mysql-json-document-store/>
- Postgres Document Store
 - <https://community.sisense.com/t5/knowledge/postgres-vs-mongodb-for-storing-json-data-which-should-you/t-p/111>

Extension JSON de PostgreSQL

- Possibilité de définir des colonnes JSON
- Un type spécifique JSONB
 - stockage binaire optimisé
 - possibilité de création d'index

Exemple

```
CREATE TABLE utilisateurs (
    id SERIAL PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    profil JSONB
);
```

Intégration des Documents JSON dans PostgreSQL

Exemple de données

```
INSERT INTO utilisateurs (nom, email, profil) VALUES
('Alice Martin', 'alice.martin@example.com', '{
    "preferences": {
        "langue": "français",
        "notifications": true
    },
    "adresses": [
        {"type": "domicile", "ville": "Paris", "code_postal": "75001"},
        {"type": "travail", "ville": "Versailles", "code_postal": "78000"}
    ]
}'),
('Bob Dupont', 'bob.dupont@example.com', '{
    "preferences": {
        "langue": "anglais",
        "notifications": false
    },
    "reseaux_sociaux": {
        "twitter": "@bobdupont",
        "linkedin": "linkedin.com/in/bobdupont"
    }
}');
```

Intégration des Documents JSON dans PostgreSQL

Exemple de données

Query Query History

```
1 select nom, profil from utilisateurs
```

Data Output Messages Notifications

	nom	profil
1	Alice Martin	{"adresses": [{"type": "domicile", "ville": "Paris", "code_postal": "75001"}, {"type": "travail", "ville": "Versailles", "code_postal": "78000"}], "preferences": {"langue": "français", "notifications": true}}
2	Bob Dupont	{"preferences": {"langue": "anglais", "notifications": false}, "reseaux_sociaux": {"twitter": "@bobdupont", "linkedin": "linkedin.com/in/bobdupont"}}

Code ▾

```
1 {
2   "preferences": {
3     "langue": "anglais",
4     "notifications": false
5   },
6   "reseaux_sociaux": {
7     "twitter": "@bobdupont",
8     "linkedin": "linkedin.com/in/bobdupont"
9   }
10 }
```

Ln: 1 Col: 1

Code ▾

```
1 [
2   {
3     "adresses": [
4       {
5         "type": "domicile",
6         "ville": "Paris",
7         "code_postal": "75001"
8       },
9       {
10         "type": "travail",
11         "ville": "Versailles",
12         "code_postal": "78000"
13       }
14     ],
15     "preferences": {
16       "langue": "français",
17       "notifications": true
18     }
19   }
20 ]
```

Ln: 1 Col: 1

Documents JSON dans PostgreSQL

Comment accéder aux champs des colonnes JSON et JSONB en SQL?

- **opérateur ->** : accède à un champ en conservant le type JSON
 - Ex: SELECT data->'nom' AS nom FROM nom est ici de type JSON
- **opérateur ->>** : pour extraire un champ JSON et le convertir en texte
 - Ex: SELECT profil->>'langue' AS langue_preferee FROM langue_preferee est de type texte

Documents JSON dans PostgreSQL

Comment exprimer des conditions sur des champ JSON et JSONB en SQL?

- Utiliser **->>** pour convertir les champs et exprimer des conditions
 - Ex: `SELECT nom FROM utilisateurs WHERE profil->>'langue' = 'français';`
- Vérifier si un champ existe avec **?** (JSONB uniquement)
 - `SELECT nom FROM utilisateurs WHERE profil ? 'preferences';`
- Accéder aux éléments d'un tableau en utilisant **jsonb_array_elements** pour extraire chaque élément du tableau
 - `SELECT nom, jsonb_array_elements(profil->'adresses')->>'ville' AS ville`
 - `FROM utilisateurs;`

Intégration des Documents JSON dans PostgreSQL

Exemples de requête

1 - Rechercher les préférences de langue

The screenshot shows a PostgreSQL query tool interface. At the top, there are tabs for "Query" (which is selected) and "Query History". Below the tabs is a code editor containing the following SQL query:

```
1 v SELECT nom, profil->'preferences'->>'langue' AS langue_preferee
2 FROM utilisateurs;
3
```

Below the code editor are three tabs: "Data Output" (selected), "Messages", and "Notifications". Under "Data Output", there is a toolbar with various icons for file operations like copy, paste, and save. The main area displays a table with two rows of data:

	nom character varying (100)	langue_preferee text
1	Alice Martin	français
2	Bob Dupont	anglais

Intégration des Documents JSON dans PostgreSQL

Exemples de requête

2 - Rechercher les utilisateurs recevant des notifications

The screenshot shows a PostgreSQL query editor interface. The top navigation bar has tabs for "Query" and "Query History", with "Query" currently selected. Below the tabs is a code editor area containing the following SQL query:

```
1 ✓ SELECT nom
2   FROM utilisateurs
3 WHERE profil->'preferences'->>'notifications' = 'true';
4
```

Below the code editor is a toolbar with several icons: a plus sign, a file icon, a dropdown arrow, a clipboard icon, another dropdown arrow, a trash bin, a database icon, a download icon, and a refresh icon.

Underneath the toolbar is a "Data Output" section. It displays a table with two columns: "nom" and "character varying (100)". The table contains one row with the value "Alice Martin".

	nom character varying (100)
1	Alice Martin

Intégration des Documents JSON dans PostgreSQL

Exemple de requête

Rechercher toutes les villes où habitent des utilisateurs

The screenshot shows a PostgreSQL query editor interface. The top navigation bar has tabs for "Query" (which is selected) and "Query History". Below the tabs is a code editor area with four numbered lines of SQL:

```
1 v  SELECT nom, jsonb_array_elements(profil->'adresses')->>'ville' AS ville
2   FROM utilisateurs;
3
4
```

Below the code editor is a toolbar with icons for file operations (New, Open, Save, etc.) and a refresh button.

The main content area displays the query results in a table format. The table has two columns: "nom" and "ville". There are two rows of data:

	nom	ville
1	Alice Martin	Paris
2	Alice Martin	Versailles

Colonnes JSON versus Texte en PostgreSQL

Avantages

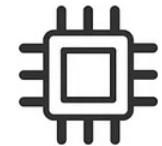
- **Flexibilité**
 - données semi-structurées ou non structurées sans schéma rigide
 - adaptation aux évolutions imprévues
- **Compatibilité**
 - données provenant d'API ou d'applications web.
- **Requetage en SQL**
- **Indexation possible** avec JSONB

Limites

- **Absence de validation de Structure**
- **Performances d'Écriture pénalisées**
- **Consommation Mémoire** importante
- **Requêtes SQL plus complexes**
 - surtout si jointures aussi
 - Optimisation difficile
 - Difficultés dans l'Estimation des Coûts

Conseil : Limiter les colonnes JSON aux données à structure variable

Extension vectorielle de PostgreSQL: pgVector



pgvector

- Pour stocker, indexer et rechercher des vecteurs (embeddings) directement dans une table PostgreSQL
- Intégration directe dans PostgreSQL : Pas besoin de maintenir une base externe
 - Calcul de distances
 - Compatible avec les frameworks d'IA (TensorFlow, PyTorch, OpenAI)
 - Performant pour des millions de vecteurs grâce à des index optimisés.

pgVector : exemple

■ Installation

```
CREATE EXTENSION pgvector;
```

■ Crédit d'une table

```
CREATE TABLE items (
    id serial PRIMARY KEY,
    vecteur_features vector(3) );
```

■ Insertion

```
INSERT INTO items (vecteur_features)
VALUES ('[0.1, 0.2, 0.3]');
```

■ Requêtes

Opérateur \leftrightarrow (distance)

```
SELECT id FROM items ORDER BY
vecteur_features  $\leftrightarrow$  '[0.1, 0.2, 0.3]' LIMIT 5;
```

- Calcule la distance entre vecteur_features et le vecteur donné
- Trie en fonction de la distance

NoSQL ou relationnel?

Vers des architectures hybrides

NoSQL

- Stockage de masse
- Données diverses
- Scalabilité horizontale



SQL

- Stockage fiable
- Données formatées
- Scalabilité verticale



Système
d'information
(Projet)

Une vidéo de synthèse



Source : <https://www.youtube.com/watch?v=W2Z7fbCLSTw>

SGBD et IA ?



TOUTE L'ACTUALITÉ / LOGICIEL / INTELLIGENCE ARTIFICIELLE

10 façons dont l'IA bouleverse les bases de données traditionnelles

Peter Wayner, InfoWorld (adapté par Dominique Filippone), publié le 30 Mai 2024



La GenAI n'est pas réservée aux chatbots. Voici 10 façons dont l'IA et l'apprentissage automatique transforment la façon dont nous stockons, structurons et interrogeons les données.



Bases de données et intelligence artificielle font bon ménage mais encore faut-il savoir en profiter en maîtrisant les atouts mais aussi défis de leur intégration technique et fonctionnelle. (crédit : Geralt / Pixabay)

LIVRES BLANCS

Accélérez vos processus de vente en intégrant un logiciel CPQ au sein de votre entreprise

Copilot Microsoft 365 : Libérez votre créativité, simplifiez votre quotidien !

VIA LE MONDE INFORMATIQUE

SUIVRE TOUTE L'ACTUALITÉ

Newsletter

Recevez notre newsletter comme plus de 50 000 professionnels de l'IT!

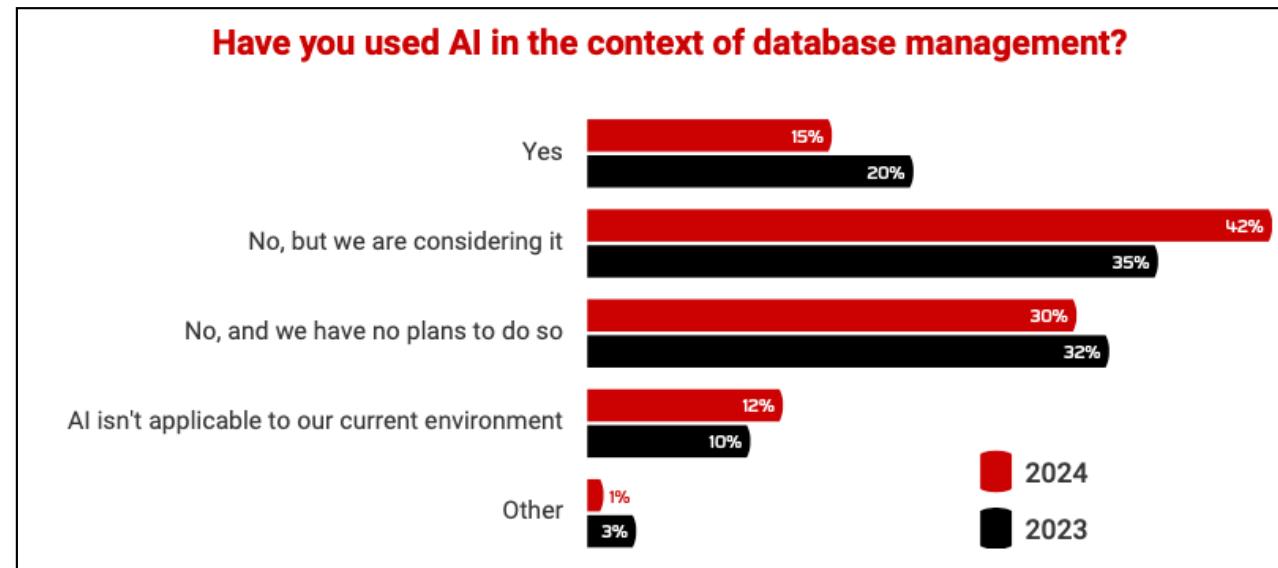
[JE M'ABONNE](#)

workday.
**ROCKSTARS
DU BUSINESS**

**INNOVEZ
JOUR**

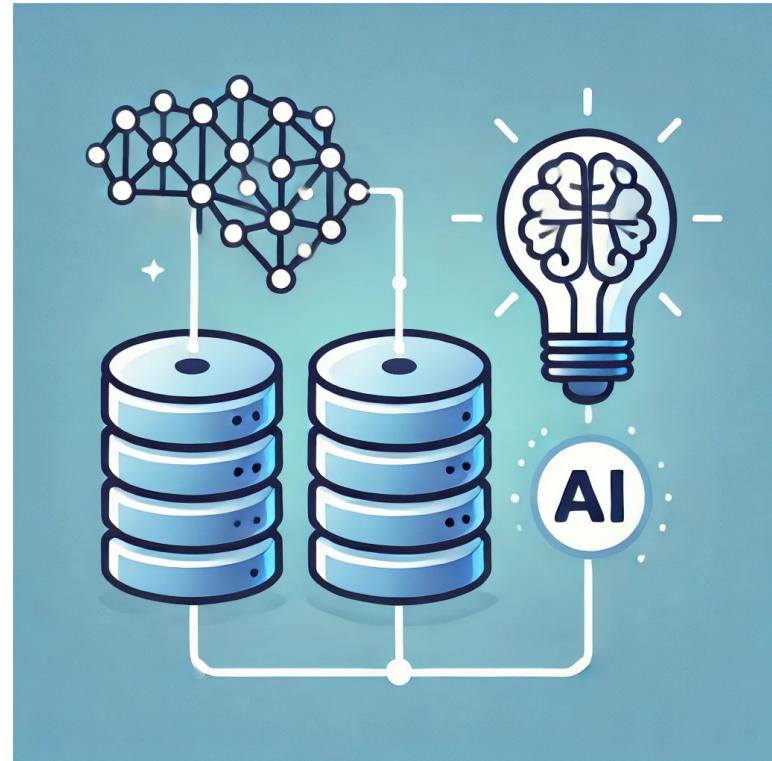
Utilisation des outils IA dans la gestion des données ?

The tasks AI is being used for



Vers des BD plus intelligentes "dopées" à l'IA

- Accessibilité accrue
 - Les non-experts pourront interagir avec les BD sans connaissances techniques
- Gain de temps
 - Moins de gestion manuelle, avec des tâches automatisées ou simplifiées
- Décisions plus rapides et précises
 - Des analyses enrichies et compréhensibles directement via les SGBD



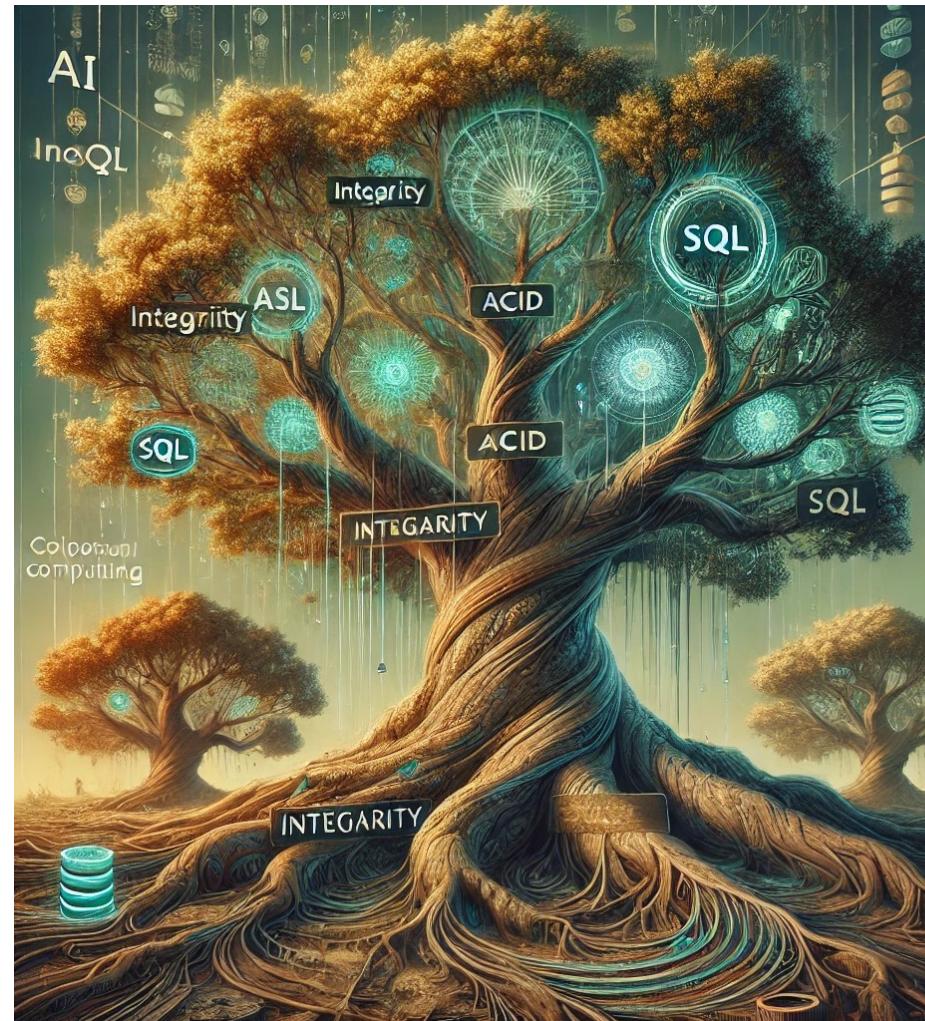
L'IA comble certaines lacunes des SGBD NoSQL

- Amélioration significative des capacités d'optimisation
- Recherche vectorielle et requêtes sémantiques
- Automatisation des pipelines de transformations de données (ETL)



Mais les BD relationnelles ne disparaîtront pas !

- Robustesse, maturité et fiabilité éprouvée
 - Large adoption dans les applications critiques
 - Essentielles dans des domaines comme la finance, la santé ou le commerce
- Standards et interopérabilité
 - Rôle fondamental de SQL : universellement utilisé et compris
- Évolutivité et innovation
 - Les SGBD relationnels s'adaptent, s'enrichissent et évoluent sans cesse



Relational + NoSQL : une coexistence durable

- Le **relationnel reste irremplaçable** dans des domaines critiques
- Les SGBD NoSQL et Big Data répondent à des **besoins spécifiques**, mais ne couvrent pas tous les cas d'usage
- Une **synergie croissante**

Les architectures modernes s'appuient sur la complémentarité des deux types de systèmes, souvent dans un contexte cloud ou distribué

L'avenir : Diversification des SGBD

Livestream

Navigating the database landscape in 2024: Shifting skills to match constant demands

What is the state of the database landscape in 2024?

We surveyed 3,849 database professionals from every sector and every company size, from developers and DBAs to IT leaders and CTOs across the globe to find out.

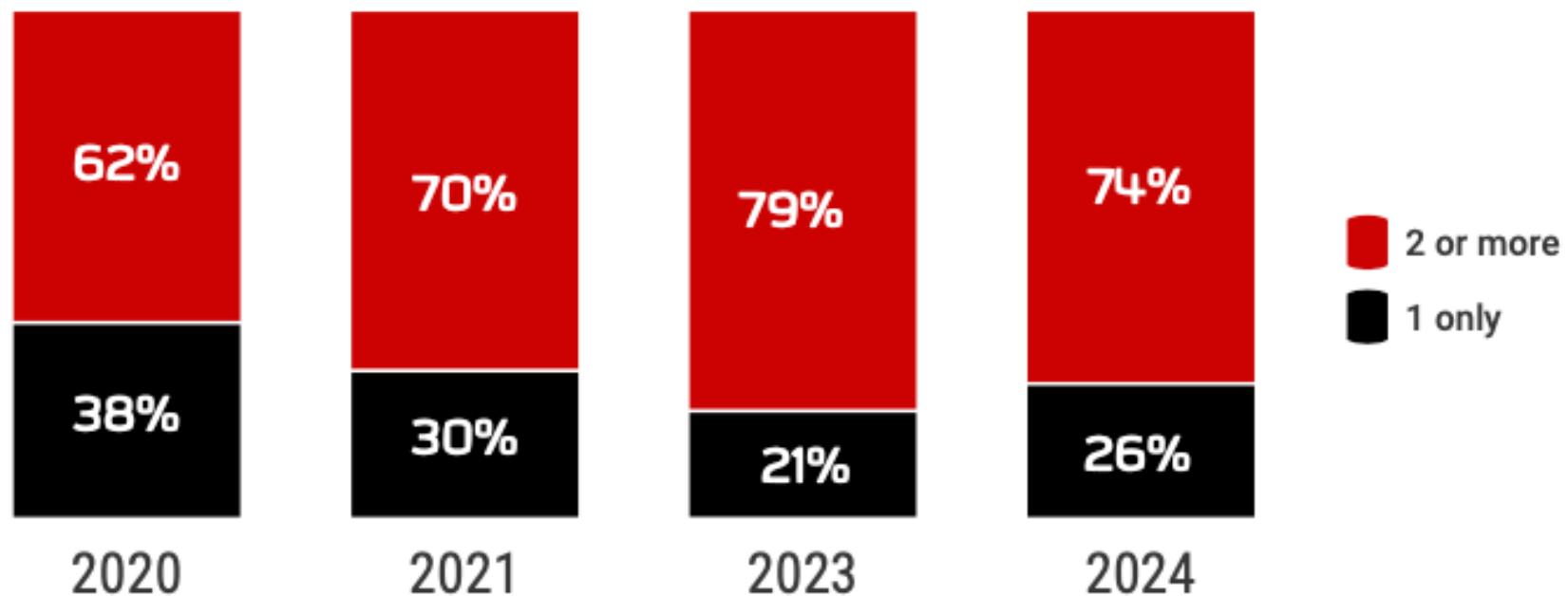
The key finding? That the pace of change in our industry is faster than ever. This is causing a recurring challenge to emerge: the rapid need for skill diversification from data professionals everywhere.

■ Augmentation du nombre de SGBD utilisés

- Cloud adoption is increasing; with 88% of respondents using the cloud in some capacity, though many are implementing sub-optimal migration strategies
- Organizations are struggling to keep up with the pace of AI; 20% of respondents are using it for database management and a further 35% would like to

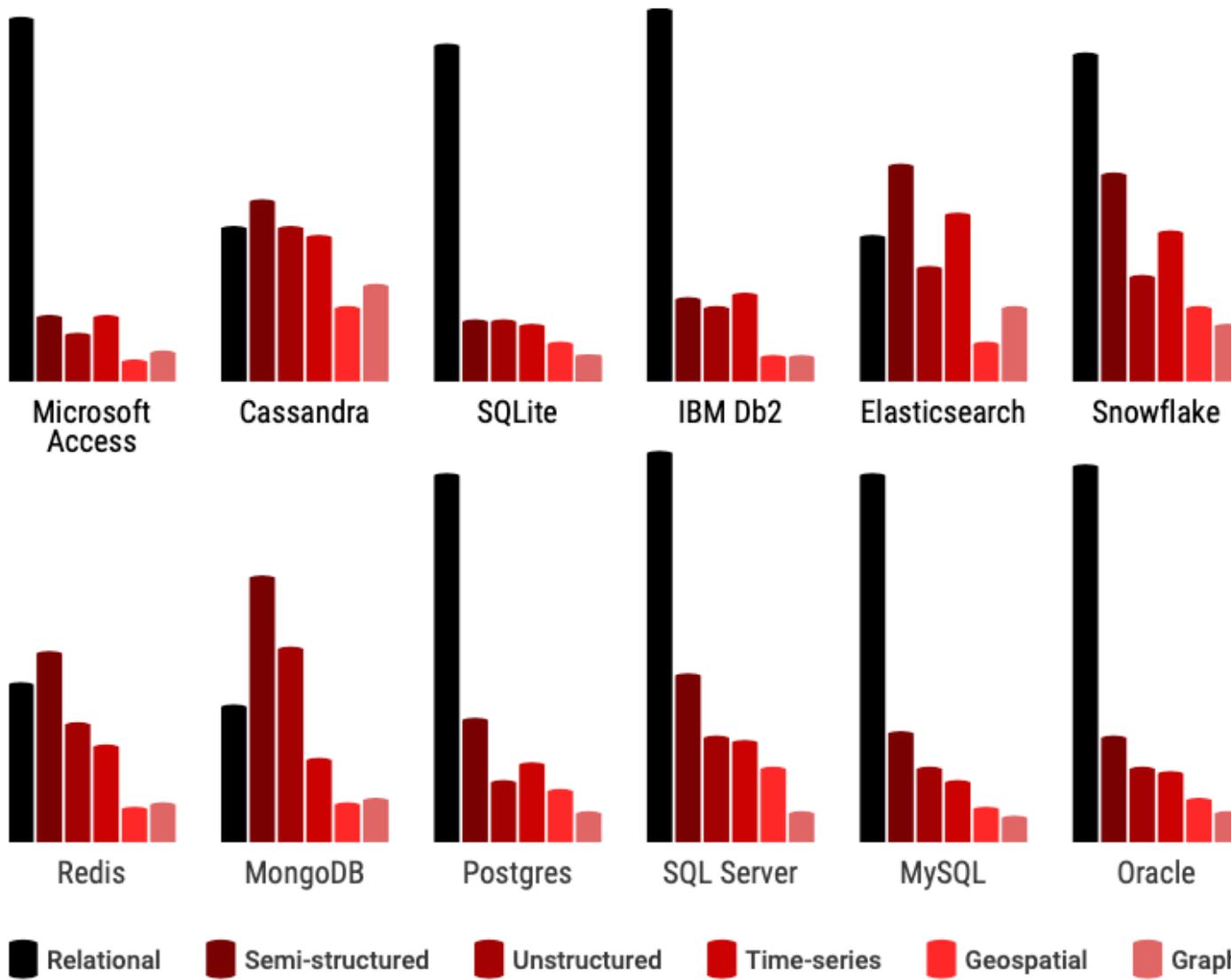
Diversification des SGBD

The number of different database platforms in use



Hétérogénéité des données

The kinds of data stored in different database platforms



- Données semi-structurées

Pas de schéma rigide mais une organisation

- Ex: JSON, XML, YAML, fichiers CSV ...

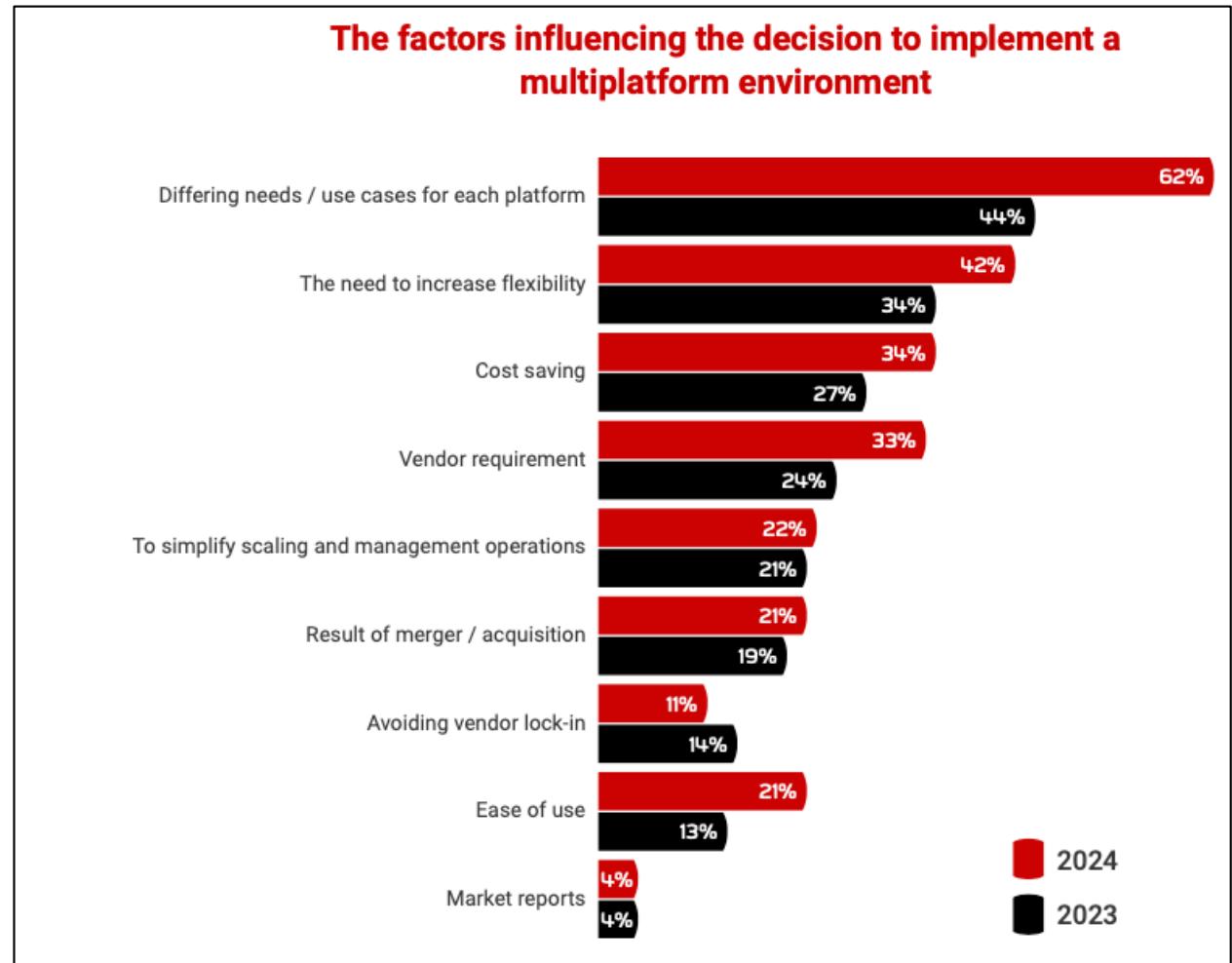
- Données non structurées

Ni structure, ni organisation

- Ex: texte brut, MP3, MP4, JPEG, PNG, PDF, ...

Complexité accrue

- Besoin de nouvelles compétences (DBA, développeurs, ...)

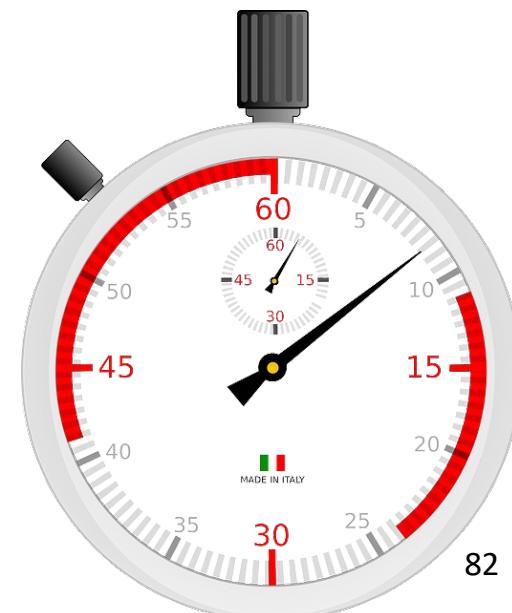


A vous de jouer ..

Travail à réaliser en trinôme



- Choisissez un SGBD NoSQL (*cité dans le cours ou autre*)
 - Saisissez votre choix dans le fichier Excel partagé sur Teams
- Téléchargez et installez une version d'essai
- Testez le SGBD sur un exemple
 - Vous pouvez utiliser l'exemple proposé dans le tutoriel officiel (s'il existe)
- Téléchargez et complétez la fiche d'évaluation de votre SGBD :
 - Fiche_Eval_SGBD_NoSQL XXXX.docx



A vous de jouer ..

Travail à réaliser en binôme



■ Dans 2h !!

- Déposez votre fiche complétée sur TEAMS
- Présentez votre retour d'expérience
(10-15 mn)
 - Facilité/difficulté d'installation : expliquez la procédure
 - Démonstration d'un exemple simple
 - Votre avis sur la prise en main du langage de définition et de requêtage

