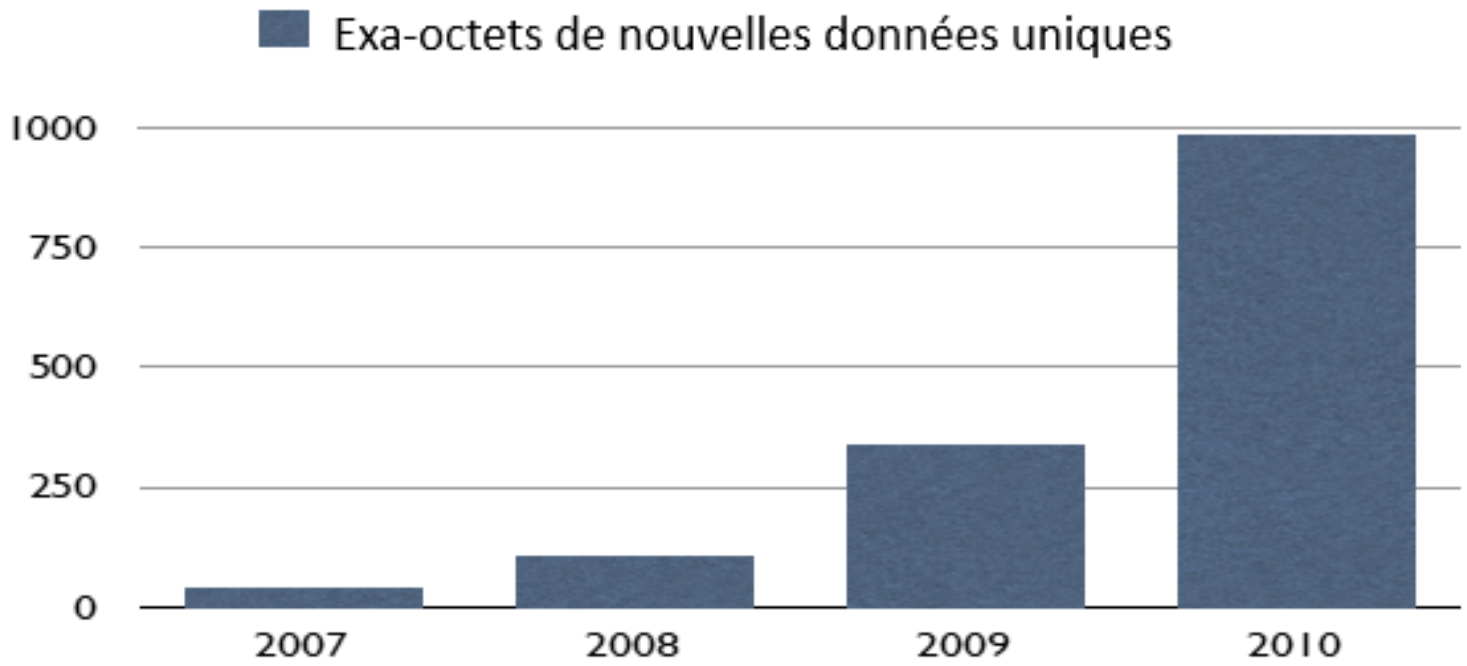
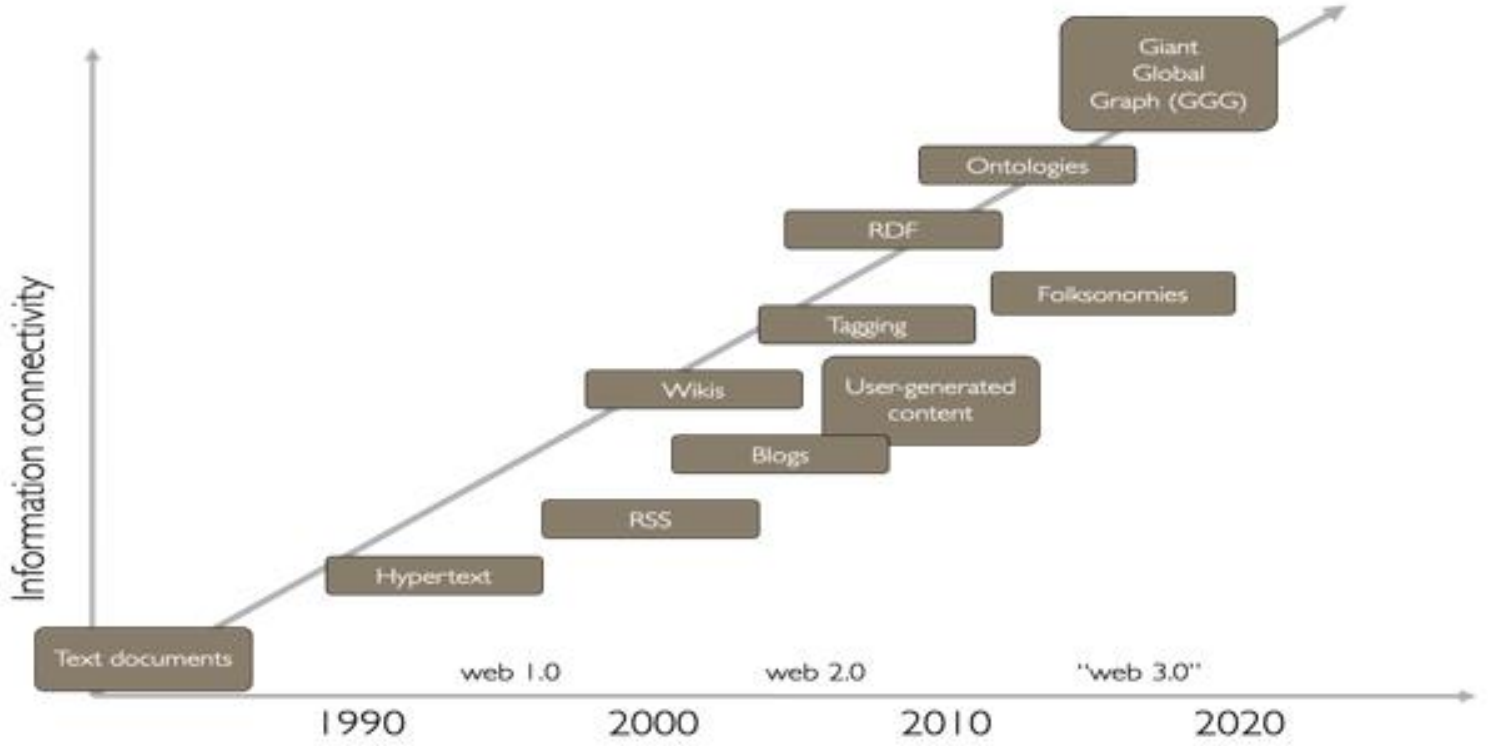
BASE DE DONNÉES ORIENTÉE GRAPHE

Mme Ines Slimene
Ines_slimene@yahoo.fr

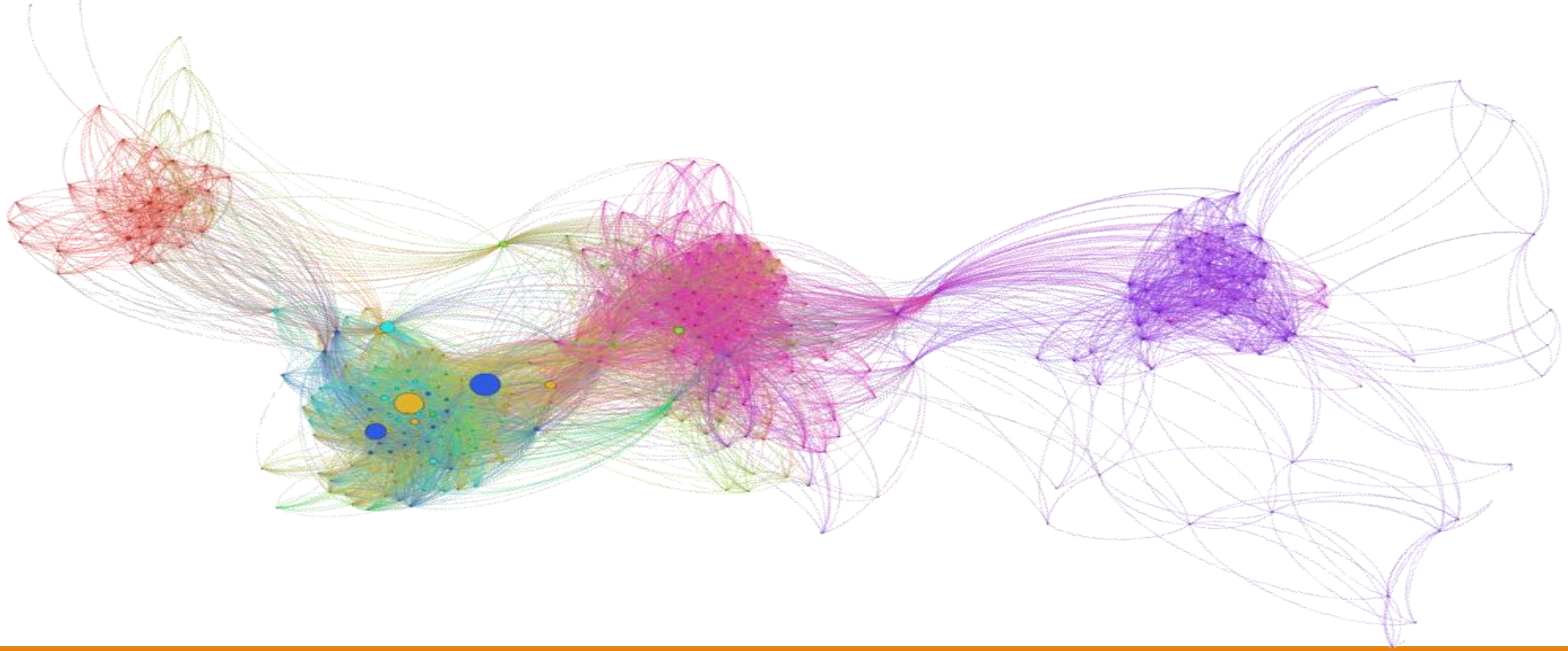
Problème 1 : Croissance exponentielle du volume de données



Problème 2: Explosion de la connectivité des données

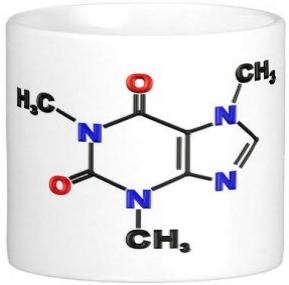


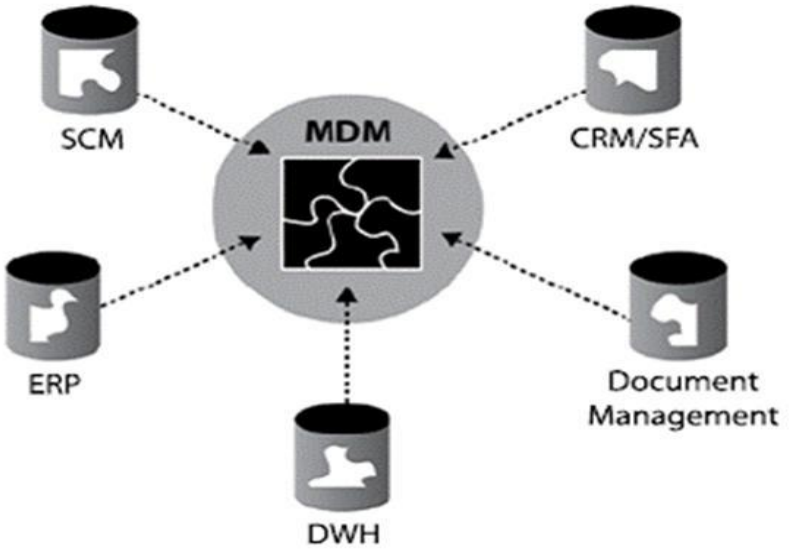
Volume x Connectivité = **Complexité**



Cas d’utilisation

- Détection de fraude
- Recommandation
- Réseaux sociaux
- Gestion de configuration
- Géo-Spatial
- Interactions moléculaires (Biologie)
- Gestion de ligne produit
- Collaboration





Neo Technology (Neo4j)

Editeur de la base de données de graphes Neo4j depuis 2000

Présence en France, Allemagne, Angleterre, Suède, USA, Grèce et Malaisie

1 000 000+ téléchargements

27 000 membres dans la communauté dans 25 pays (7 villes en France)

Top 500 clients tels que Adobe, eBay, Wallmart, UBS, ABC Bank,Cisco, Deutsch Telecom, Deutsch Post, Telenor, SFR, Lockheed Martin, Airbus…

Partenaires locaux ou globaux tels que Accenture : +150

Partenaires technologiques tels que VMware, Informatica et Microsoft

Neo4j est leader mondial des bases de données Graph

Inconvénients autres types de BDNR

Pas de structure de données pour modéliser ou stocker les relations

Pas de requêtes développées pour supporter les relations entre les données

Les relations entre les données exigent une jointure logique dans l'application

Pas de support ACID pour les transactions

… les bases de données NoSQL ne sont pas adaptées pour les process exigeants des relations entre les données en temps réel

Neo4j

Neo4j est une base de données de graphe de type entreprise permettant:

- Modéliser et stocker les données comme un graphe
- Requêter les relations entre les données facilement et en temps réel
- Evolution simple des applications pour supporter les nouveaux besoins et ajouter les nouveaux types de données et relations

Caractéristiques

Stockage graphe natif qui assure la consistance des données et la performance

Native Graph Processing
Millions t'étapes, de "sauts" par seconde, en temps réel.

"Whiteboard Friendly" Data Modeling
Modélisation implicites des données

Haute intégrité des données

Transactions 100% ACID

Requête simple et performante

Requiert 10 à 100 fois moins de code que SQL

Scalabilité et haute disponibilité
Scalabilité verticale et horizontale

Built-in ETL
Import simple des bases de données et fichiers

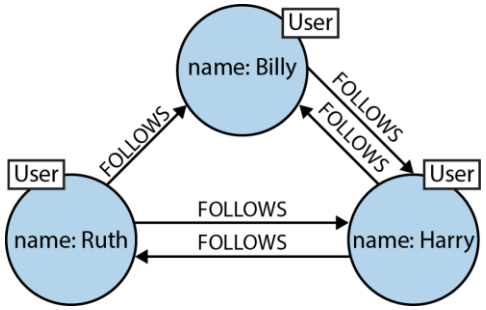
Integration
Drivers et APIs pour les langages standards

Base de données orientée « Graphe »

Une base dont la structure est représentée sous forme de « graphe » (théorie des graphes)

Ce graphe est basé sur des Nœuds et des Arcs

Par exemple, les relations entre les utilisateurs de Twitter peuvent être représentées de manière simplifiée par le graphe suivant :



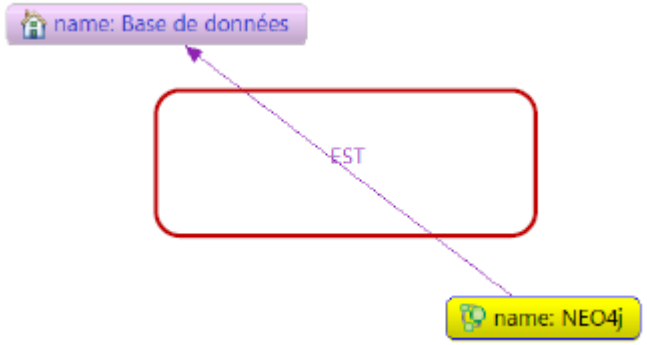
Base de données orientée « Graphe »

Dans cette structure de données, on considère que la relation est aussi importante que l'entité

La relation est identifiée par un nom (sémantique)

Exemple :

- NEO4j (Java) : Neo Technology
- OrientDB (Java) : Nuvela Base
- DEX (C++) : Sparsity Technologies
- TinkerGraph (Java) : TinkerPop



Base de données orientée « Graphe »

Un graphe peut représenter n'importe quel type de données.

Un **nœud** est l'équivalent d'un enregistrement

Une **relation** permet de relier des nœuds, et peut être typée

Les nœuds et les relations peuvent avoir des **propriétés,** représentant des attributs nommés

Un **label** est un nom permettant d'organiser les nœuds en groupes

Stockage et traitement graphe

Stockage Graphe

- Objectif: performance et *scalabilité*.
- Stockage des données représentées sous forme d'un graphe, **avec des nœuds et des relations**.
- Utilisant des **structures de stockage dédiées** aux nœuds et relations.

Traitement Graphe

- Parcours des relations grâce à des pointeurs physiques.
- Système de stockage capable de fournir une adjacence entre éléments voisins : chaque voisin d'une entité est accessible grâce à un pointeur physique.
- Lecture et parcours des données **sans recours à un index**, en utilisant les arcs pour passer d'un nœud à l'autre.

Scalabilité: sharding

•S'il faut dépasser la capacité d'un cluster, nous pouvons partitionner le graphe sur plusieurs instances de la base via la construction d'une **logique de sharding dans l'application**.

•Le sharding implique l'utilisation d'un identifiant synthétique pour la jointure des données sur plusieurs instances de la base au niveau applicatif.

•Comment cela s'effectuera dépend beaucoup de la forme du graphe. Certains graphes se prêtent très bien à ce cas avec des «frontières pratiques». Bien sûr, pas tous les graphes ont des «frontières» si évidentes.

•L'objectif futur de la plupart des bases de données graphe est d'être capable de partitionner un graphe sur plusieurs machines sans intervention au niveau de l'application, de sorte que l'accès en lecture et en écriture sur le graphe puisse être *scalable* horizontalement.

•Dans le cas général, cela est connu pour être un problème NP-dur, et donc impossible à résoudre.

La base de données Neo4j

Neo4j ne gère pas les bases de données multiples.

Le cas d'utilisation d'une base de données orientée graphes est spécifique, et il n'y a pas vraiment de sens de créer plusieurs bases de données.

Neo4j stocke toutes ses données dans un répertoire.

Pour supprimer la base de données entière, il convient d'arrêter le service, de supprimer tous les

fichiers du répertoire (neo4j/data/graph.db) et de redémarrer le service.

Les langages de Neo4j

Mis à part les pilotes spécifiques pour les langages clients, qui permettent de manipuler les

nœuds et les relations dans un mode procédural, Neo4j offre deux langages spécifiques :

Cypher et Gremlin.

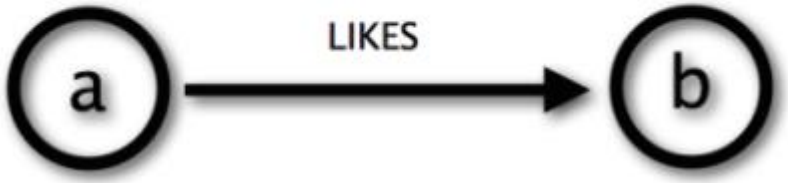
Cypher est un langage déclaratif inspiré du SQL, qui permet d'exprimer une requête complexe de façon élégante et compacte.

Gremlin, quant à lui, est un langage de script basé sur Groovy (un langage pour la plate-forme Java dont la syntaxe s'inspire de langages de script comme Python et Ruby). Il permet d'envoyer des scripts qui seront exécutés du côté serveur à travers l'interface REST de Neo4j.

Nous allons utiliser ici des commandes Cypher.

Language Cypher

Cypher using relationship 'likes'



Cypher

```
(a) -[:LIKES]-> (b)
```