# Bases de données relationnelles et bases de données NoSQL

Camille Pradel – camillepradel@gmail.com

*Synapse Développement*

# Plan

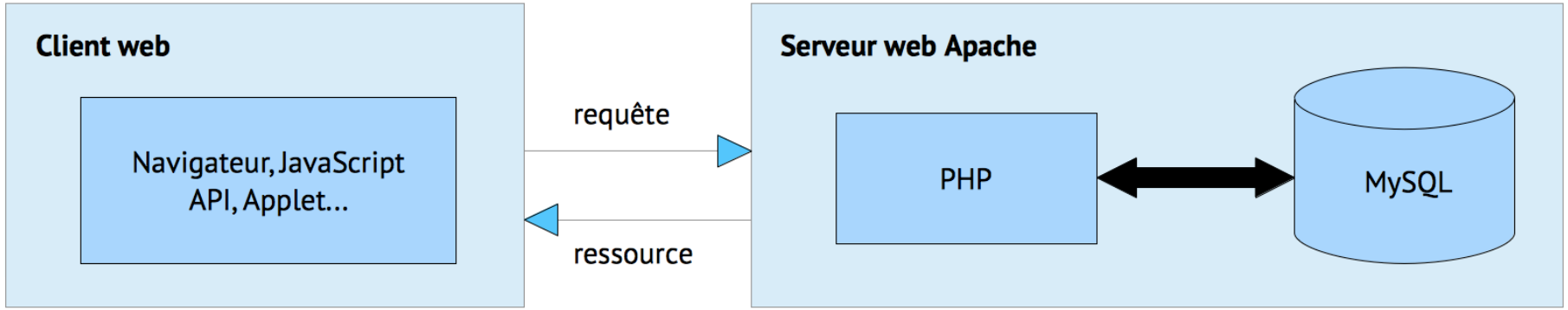
- Architecture web et relationnel
  - Architecture classique
  - Bilan architecture classique
  - Les ORM
- Au delà du SQL: la BD NoSQL
  - Pourquoi changer du relationnel?
  - Relâchement des contraintes de transactions
  - À quoi ressemble une BD NoSQL?
- Les types de bases de données NoSQL
  - Les bases de données clés-valeurs
  - Les bases de données orientées documents
  - Les bases de données orientées graphes
- Conclusion
  - Bonnes pratiques
  - NoSQL ou relationnelles?

# Plan

- Architecture web et relationnel
  - Architecture classique
  - Bilan architecture classique
  - Les ORM
- Au delà du SQL: la BD NoSQL
  - Pourquoi changer du relationnel?
  - Relachement des contraintes de transactions
  - À quoi ressemble une BD NoSQL?
- Les types de bases de données NoSQL
  - Les bases de données clés-valeurs
  - Les bases de données orientées documents
  - Les bases de données orientées graphes
- Conclusion
  - Bonnes pratiques
  - NoSQL ou relationnelles?

# Architecture classique

- Stacks habituelles
  - LAMP: Linux, Apache, MySQL, PHP – la plus commune
  - LEMP: Linux, Nginx, MySQL, PHP-FPM – monte en popularité



# Bilan architecture classique

- Avantages
  - Rapide à mettre en place
  - Optimisation possible (surtout verticalement)

- Inconvénients
  - Il faut une expertise base de données (schema, indexes, contraintes d’intégrité…)

# ACID

- **Atomicité** - La propriété d'atomicité assure qu'une transaction se fait au complet ou pas du tout : si une partie d'une transaction ne peut être faite, il faut effacer toute trace de la transaction et remettre les données dans l'état où elles étaient avant la transaction. L'atomicité doit être respectée dans toutes situations, comme une panne d'électricité, une défaillance de l'ordinateur, ou une panne d'un disque magnétique.
- **Cohérence** - La propriété de cohérence assure que chaque transaction amènera le système d'un état valide à un autre état valide. Tout changement à la base de données doit être valide selon toutes les règles définies, incluant mais non limitées aux contraintes d'intégrité, auxrollbacks en cascade, aux déclencheurs de base de données, et à toutes combinaisons d'évènements.
- **Isolation** - Toute transaction doit s'exécuter comme si elle était la seule sur le système. Aucune dépendance possible entre les transactions. La propriété d'isolation assure que l'exécution simultanée de transactions produit le même état que celui qui serait obtenu par l'exécution en série des transactions. Chaque transaction doit s'exécuter en isolation totale : si T1 et T2 s'exécutent simultanément, alors chacune doit demeurer indépendante de l'autre.
- **Durabilité** - La propriété de durabilité assure que lorsqu'une transaction a été confirmée, elle demeure enregistrée même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre problème. Par exemple, dans une base de données relationnelle, lorsqu'un groupe d'énoncés SQL ont été exécutés, les résultats doivent être enregistrés de façon permanente, même dans le cas d'une panne immédiatement après l'exécution des énoncés.

# Les ORM

- Définition : ORM
  - L'Object-Relationnal Mapping est une technique qui simule une base de données orientée objet à partir d'une base de données relationnelle.
- Fait la liaison entre le monde relationnel dans la couche stockage et le monde objet dans l'application ;
- Facilité de développement : pas besoin d'une connaissance poussée du SQL ;
- Facilite les interactions avec la base de données pour les développeurs.
- Les limites des ORM
  - Toujours **beaucoup** moins performant que des requêtes SQL optimisées.

# ORM: exemple de requêtes

```php
<?php
// Création d'un utilisateur
$data = [
      'prenom' => 'Antoine',
      'age' => 42
];
$user = User::create($data);

// Sélection des utilisateurs majeurs et articles qu'ils ont écrits
$users = User::with('articles')
      ->where('age', '>=', 18)
      ->get();

// Suppression des utilisateurs vivant à Paris
User::where('ville', 'Paris')->delete();

// Les derniers articles d'un utilisateur (3ème page)
$articles = Article::whereUserId($user->id)
      ->latest()
      ->take(10)
      ->skip(20)
      ->get();
```

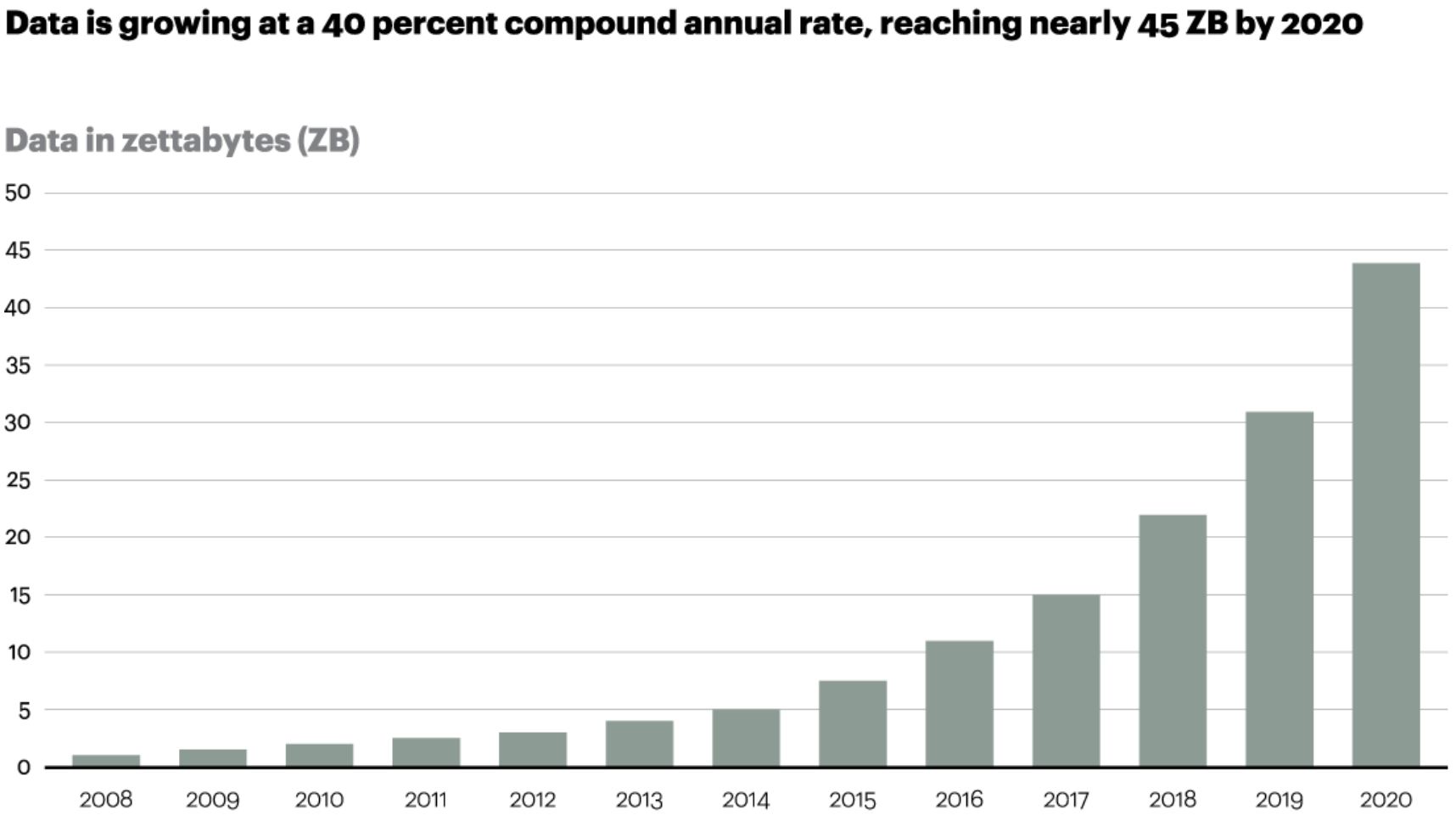
Listing 1: Quelques requêtes basiques avec l'ORM Eloquent (PHP).

D'autres ORM : Hibernate (Java), SQLAlchemy (Python)…

# Plan

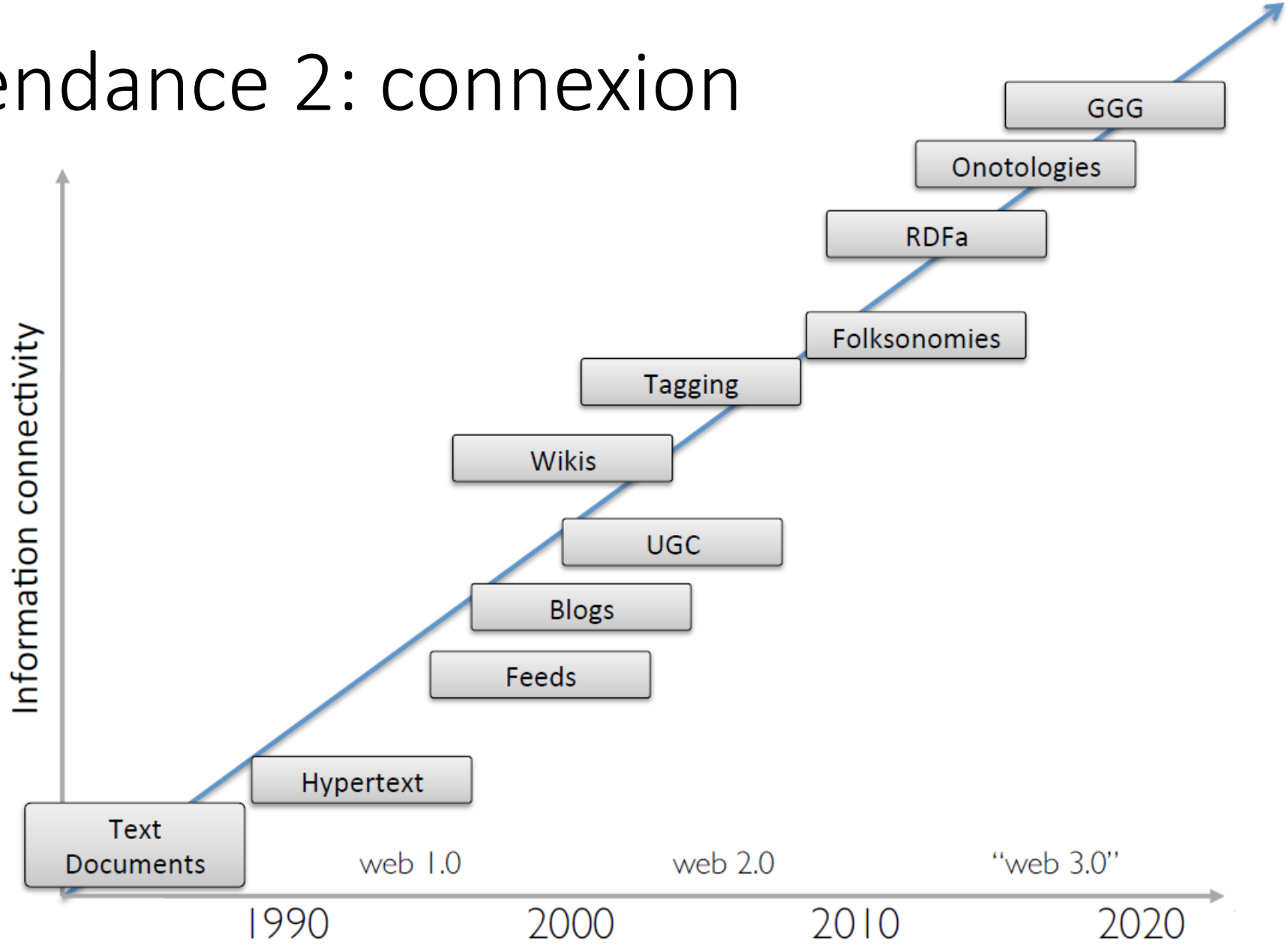
- Architecture web et relationnel
  - Architecture classique
  - Bilan architecture classique
  - Les ORM
- Au delà du SQL: la BD NoSQL
  - Pourquoi changer du relationnel?
  - Relachement des contraintes de transactions
  - À quoi ressemble une BD NoSQL?
- Les types de bases de données NoSQL
  - Les bases de données clés-valeurs
  - Les bases de données orientées documents
  - Les bases de données orientées graphes
- Conclusion
  - Bonnes pratiques
  - NoSQL ou relationnelles?

# Tendance 1: quantités de données




http://www.atkearney.fr/documents/10192/698536/FG-Big-Data-and-the-Creative-Destruction-of-Todays-Business-Models-1.png/0c519468-de82-45cb-afde-5d7b431b00b5?t=1358296806758

# Tendance 2: connexion




http://2011.geecon.org/materials/presentations/2011_webber-a_programmatic_introduction_to_neo4j.pdf

# Tendance 3: contenus moins structurés

- Individualisation des données
  - Historiques, likes, amis…
- Monde plus complexe
  - Statistiques emplois et salaires: 1970’s ≠ 2010’s
- Besoin de croiser des données hétérogènes
  - Recommandation, détection de fraudes
- On veut stocker plus de données sur chaque entitée, mais on ne sait pas tout
- Décentralisation et génération de contenus
  - Web 2.0
- Contraintes de respect de la vie privée
  - RGPD

# Problèmes liés au relationnel

- Service réparti sur plusieurs continents ?
- Accès concurrent de plusieurs centaines de milliers de personnes ?
- Stockage d’objets avec prise en compte de l’héritage ?
- Reconnaissance automatique d’images ou traitement automatique de textes ?

# Pourquoi changer du relationnel ?

Besoin d'une alternative vers les années 2004 avec l'arrivée du Big Data.

- Des volumes de données important (plusieurs gigas, voire téraoctets),
- Un nombre de transactions très important, une forte demande de disponibilité et de temps de réponse,
- Des bases de données réparties sur plusieurs centres de données ou continents,
- Préférence pour l'ajout de petites machines plutôt qu'une configuration poussée.

# Relâchement des contraintes de transactions

Impossible de garantir les propriétés ACID des BDs relationnelles avec les nouvelles contraintes.

De nouvelles propriétés BASE:

- Basic Availability : système disponible dans son ensemble bien que certaines machines soient indisponibles,
- Soft state : l'état du système distribué peut changer, même sans nouvelles transactions,
- Eventual Consistency : En l'absence de nouvelles transactions, le système sera cohérent au bout d'un temps.

# À quoi ressemble une BD NoSQL

- Un SGBD qui n'est pas structuré en tables et dont l'élément de base n'est pas un tuple appartient au type de BD NoSQL,
- Un langage de requête non uniformisé, propre à chaque BD. API REST généralement disponible,
- Une dénormalisation des données où certains enregistrements sont en partie ou entièrement dupliqués,
- Type de base de données NoSQL à choisir en fonction de l'usage souhaité,
- Types de base de données NoSQL existants : clé-valeur, colonnes, documents, graphe, big table…

# Plan

- Architecture web et relationnel
  - Architecture classique
  - Bilan architecture classique
  - Les ORM
- Au delà du SQL: la BD NoSQL
  - Pourquoi changer du relationnel?
  - Relachement des contraintes de transactions
  - À quoi ressemble une BD NoSQL?
- Les types de bases de données NoSQL
  - Les bases de données clés-valeurs
  - Les bases de données orientées documents
  - Les bases de données orientées graphes
- Conclusion
  - Bonnes pratiques
  - NoSQL ou relationnelles?

# Modélisation BD clé-valeur

- Modélisation : la plus simple. À une clé, on associe une valeur. La valeur peut être de n'importe quel type (chaîne de caractères, entier, structure, objet sérialisé…).

Des exemples de paires clé-valeur avec des types différents.

| Clé | Valeur |
|---|---|
| pays.id-42 | {"id" :42,"name" :"Chad"} |
| statistiques.nombre-visiteurs | 1337 |
| configuration.periode-gratuite | false |
| articles.categories-sport.latest | [22, 45, 67, 200, 87] |

# BD clé-valeur en détail

- Opérations : création d'une paire clé-valeur, suppression, accès à une valeur à l'aide de la clé, incrémentation et décrémentation d'une valeur,
- On peut définir la durée de vie d'une paire clé-valeur ou adopter une politique least recently used,
- Stockage en RAM pour accélérer les temps de lecture. Mécanisme de reprise en cas de crash,
- Cas d'utilisation : cache d'une autre BD, comptage d'éléments, gestion de files d'attente, opérations ensemblistes…
- Principaux acteurs : Redis, Memcached, Riak.

# Commandes sous Redis

```
# Assignation de chaîne "bonjour" à la clé "cleTexte"
redis> SET cleTexte bonjour
OK
# Récupération de la valeur de la clé "cleTexte"
redis> GET cleTexte
"bonjour"

# Incrémentation d’un compteur
redis> INCR compteur
(integer) 42
# Suppression du compteur
redis> DEL compteur
(integer) 1

# Création d’une liste avec insertion en fin de liste
redis> RPUSH liste "Hello"
(integer) 1
# Insertion en fin de liste
redis> RPUSH liste "World"
(integer) 2
```

- Listing 2: Quelques commandes Redis en console.

# Bases de données orientées documents

- Modélisation : aucun schéma fixe, un document peut contenir n'importe quel type d'information ;
- Optimisation horizontale ;
- Opérations : utilise le JSON pour les requêtes et l'améliore pour le stockage (BSON) ;
- Facile d'accès (API REST, shell…) ;
- Cas d'utilisation : base de données principalement utilisées pour du stockage ;
- Principaux acteurs : MongoDB, CouchDB, CouchBase.

# Exemple d’un document

```
{
    "id":1500,
    "nom":"Thibaud",
    "prénom":"Dauce",
    "professeur_préférée":{
        "numero":42,
        "prenom":"Géraldine",
        "nom":"Del Mondo",
        "matières": ["BD", "Réseaux"]
    },
    "commentaires":[
        {
            "id":646,
            "contenu":"J’adore la BD."
        },
        {
            "id":647,
            "contenu":"Boule et Bill aussi."
        },
        {
            "id":648,
            "contenu":"Et pleins d’autres trucs."
        }
    ]
}
```

Listing 3: Exemple d’un document JSON.

# Exemples de requêtes MongoDB

```
db.inventory.find( { type: "snacks" } )
```

Listing 4: Exemple de requête find sur MongoDB.

# Exemples de requêtes MongoDB

```
db.inventory.find(
    {
        type: 'food',
        $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]
    }
)
```

Listing 5: Exemple de requête find sur MongoDB avec des opérateurs spéciaux.

# Exemples de requêtes MongoDB

```
db.inventory.insert(
        {
                item: "ABC1",
                details: {
                        model: "14Q3",
                        manufacturer: "XYZ Company"
                },
                stock: [
                        { size: "S", qty: 25 },
                        { size: "M", qty: 50 }
                ],
                category: "clothing"
        }
)
```

Listing 6: Exemple de requête insert sur MongoDB.

# Exemples de requêtes MongoDB

```
db.inventory.update(
	{ item: "MNO2" },
	{
		$set: {
			category: "apparel",
			details.model: "14Q2"
		},
		$currentDate: { lastModified: true }
	}
)
```

Listing 7: Exemple de requête update sur MongoDB.

# Bases de données orientées graphes

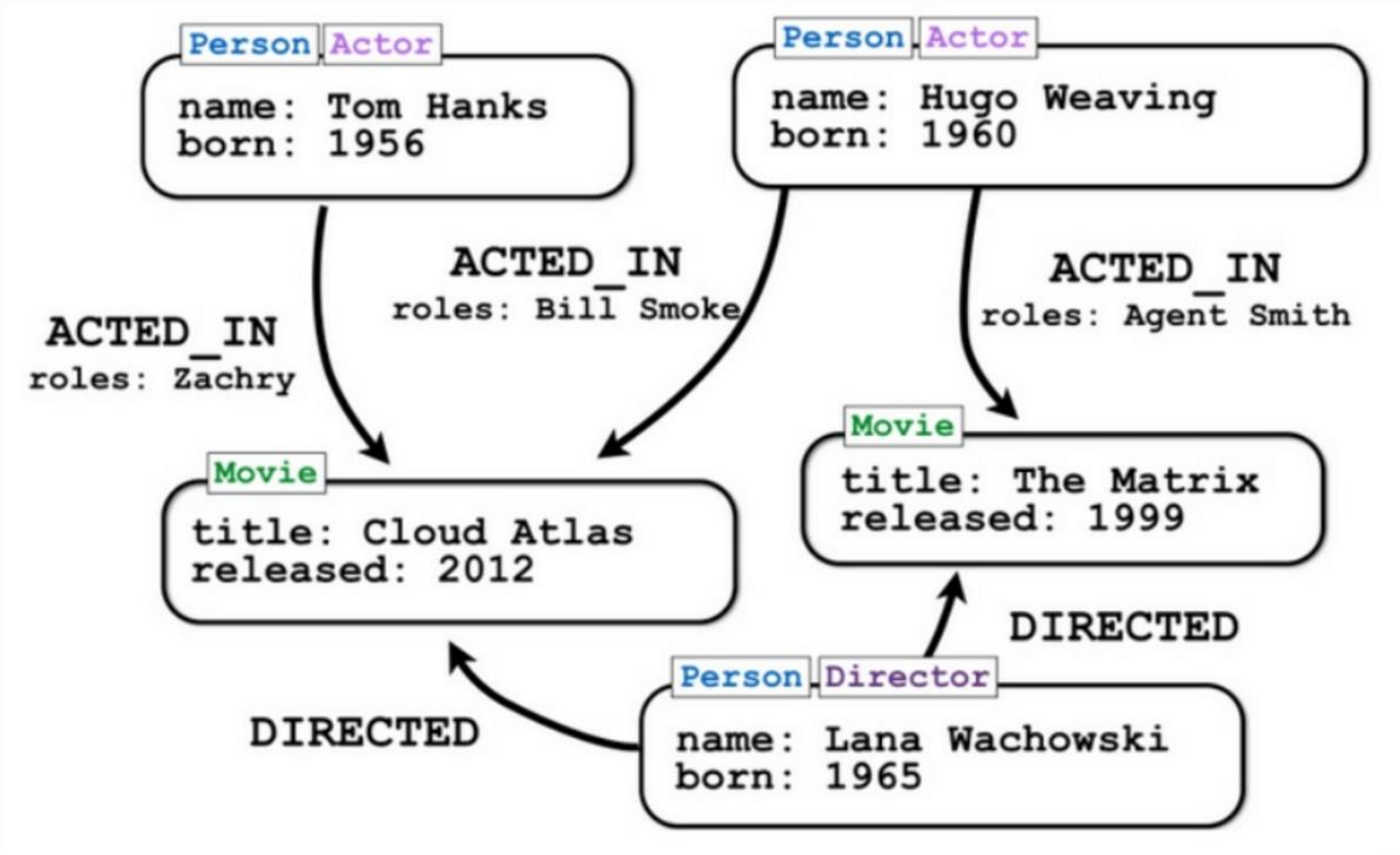



Figure : Exemple de graphe.

# Bases de données orientées graphes

- Implications importantes sur la modélisation
- Cas d’utilisation : utilisé principalement pour les réseaux
- Base de données rarement utilisée pour du stockage
- Principaux acteurs : Neo4j, Titan, OrientDB.

# Bases de données orientées graphes

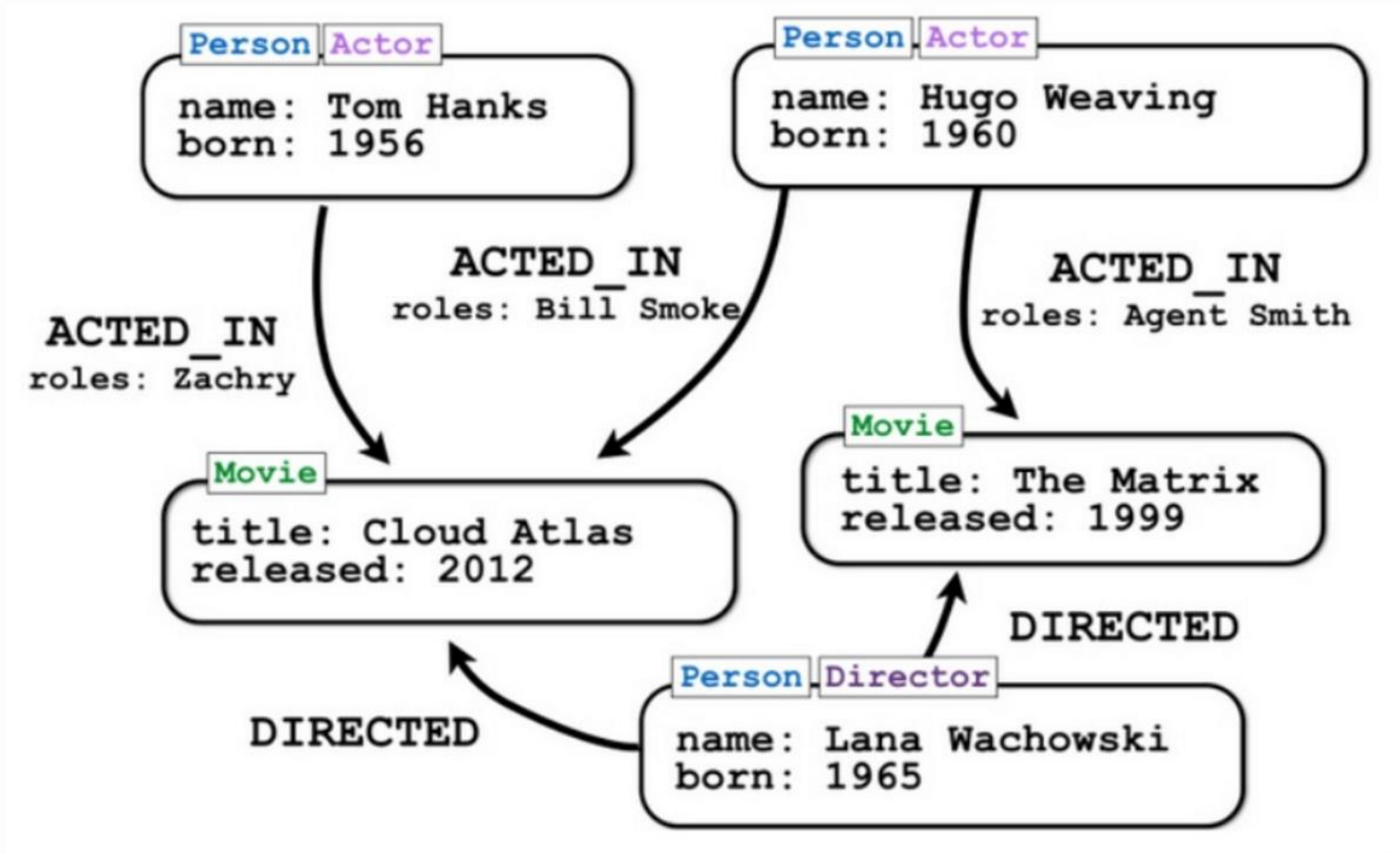



Figure : Exemple de graphe.

# Exemple de requête

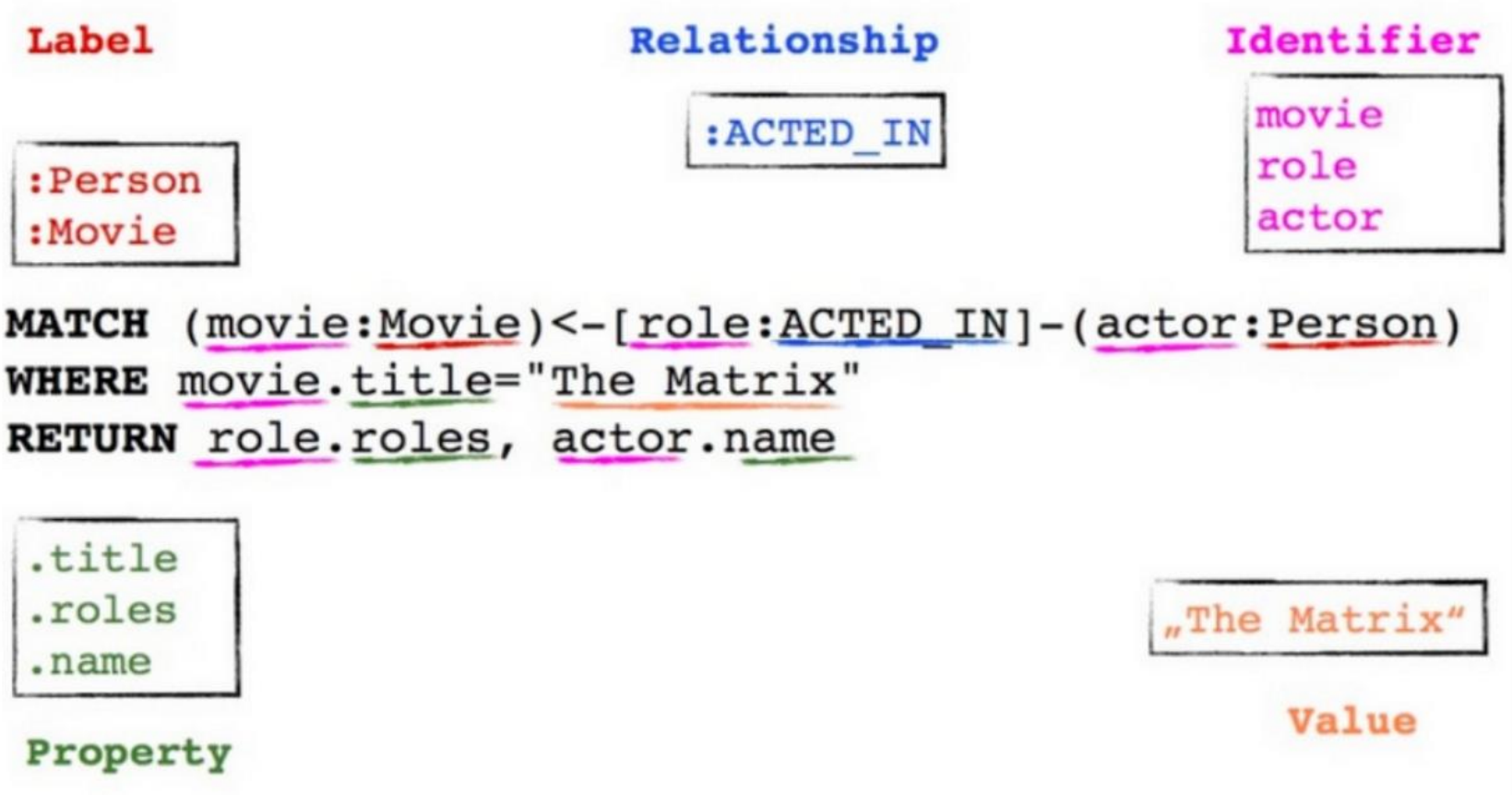



Figure : Exemple de requête Cypher pour Neo4j.

# Plan

- Architecture web et relationnel
  - Architecture classique
  - Bilan architecture classique
  - Les ORM
- Au delà du SQL: la BD NoSQL
  - Pourquoi changer du relationnel?
  - Relachement des contraintes de transactions
  - À quoi ressemble une BD NoSQL?
- Les types de bases de données NoSQL
  - Les bases de données clés-valeurs
  - Les bases de données orientées documents
  - Les bases de données orientées graphes
- Conclusion
  - Bonnes pratiques
  - NoSQL ou relationnelles?

# Les bonnes pratiques du NoSQL

- Bien connaître ses données ;
- Ne pas avoir peur de la redondance ;
- Ne pas trop redonder ;
- Et surtout, bien quantifier ses besoins.

- RTFM: la majorité des bases de données NoSQL connues sur le marché possèdent une très bonne documentation, souvent faite à destination de personnes venant du monde du relationnel.

# NoSQL ou relationnel ?

- Les deux! Les bases de données NoSQL ont été inventées afin de résoudre des problèmes insolubles par les bases de données relationnelle et non pas pour les remplacer.

| Relationnel | NoSQL |
| --- | --- |
| Stockage fiable | Stockage de masse |
| Données formatées | Données diverses |
| Scalabilité verticale | Scalabilité horizontale |

# Choix du type de BD NoSQL

| | Modélisation | Cas d'utilisation |
|---|---|---|
| Clé/valeur | Modélisation simple, permettant d'indexer des informations diverses via une clé | Mise en cache |
| Documents | Modélisation souple permettant de stocker des documents au format JSON dans des collections | Stockage de masse |
| Graphes | Modélisation optimisée | Stockage provisoire pour traiter les données |

# Baromètre de l'emploi freelance septembre 2017

| Rang | Base de données | Recherches | Tarif moyen | Paris | Marseille | Lyon | Toulouse | Nice |
|---|---|---|---|---|---|---|---|---|
| 1 | mysql | 25,6% | 322 €/jour | 383 €/jour | 322 €/jour | 358 €/jour | 357 €/jour | 332 €/jour |
| 2 | oracle | 22,4% | 429 €/jour | 496 €/jour | 458 €/jour | 422 €/jour | 366 €/jour | 370 €/jour |
| 3 | elasticsearch | 17,7% | 484 €/jour | 518 €/jour | 538 €/jour | 469 €/jour | 450 €/jour | 335 €/jour |
| 4 | postgresql | 16% | 392 €/jour | 449 €/jour | 336 €/jour | 391 €/jour | 408 €/jour | - |
| 5 | mongodb | 11,4% | 427 €/jour | 497 €/jour | 383 €/jour | 439 €/jour | 418 €/jour | 374 €/jour |
| 6 | cassandra | 3,4% | 610 €/jour | 676 €/jour | - | 443 €/jour | - | 650 €/jour |
| 7 | solr | 1,8% | 442 €/jour | 511 €/jour | 750 €/jour | - | 450 €/jour | - |
| 8 | neo4j | 1,4% | 515 €/jour | 579 €/jour | - | 500 €/jour | - | - |
| 9 | sqlserver | 0,5% | 359 €/jour | 439 €/jour | 374 €/jour | 361 €/jour | 296 €/jour | 296 €/jour |

Source: https://www.hopwork.fr/stats/barometer

# Sources

- Introduction aux bases de données NoSQL
  - Antoine Augusti et Thibaud Dauce
  - http://www.slideshare.net/antoineaugusti/introduction-aux-bases-de-donnes-nosql
  - https://www.youtube.com/watch?v=oIpjcqHyx2M
- Graph Databases
  - Ian Robinson, Jim Webber, et Emil Eifrém
  - O'Reilly Media

# Détails organisationnels

- Dates et programme des séances
  - P1
    - Mercredi 05/09 après-midi
    - Jeudi 06/09 après-midi
    - Jeudi 20/09 après-midi

    } Cours + TP Neo4j + TP MongoDB
  - Séance 4
  - Séance 5
  - Séance 6

    } Projet
  - Examen
- Ressources logicielles
  - Netbeans (ou Eclipse)
  - MySQL
  - Neo4j
  - MongoDB

# Evaluation

- TP + Projet = 50%
- Exam = 50%