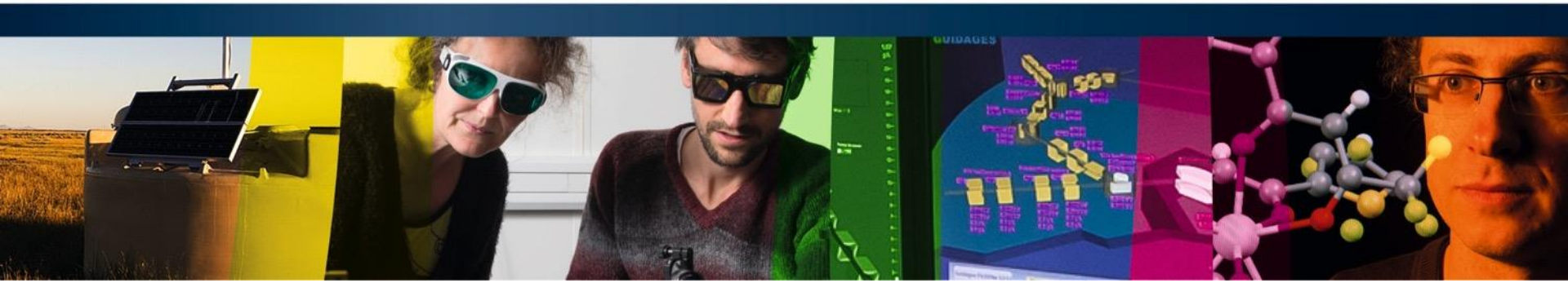
www.cnrs.fr

NoSQL

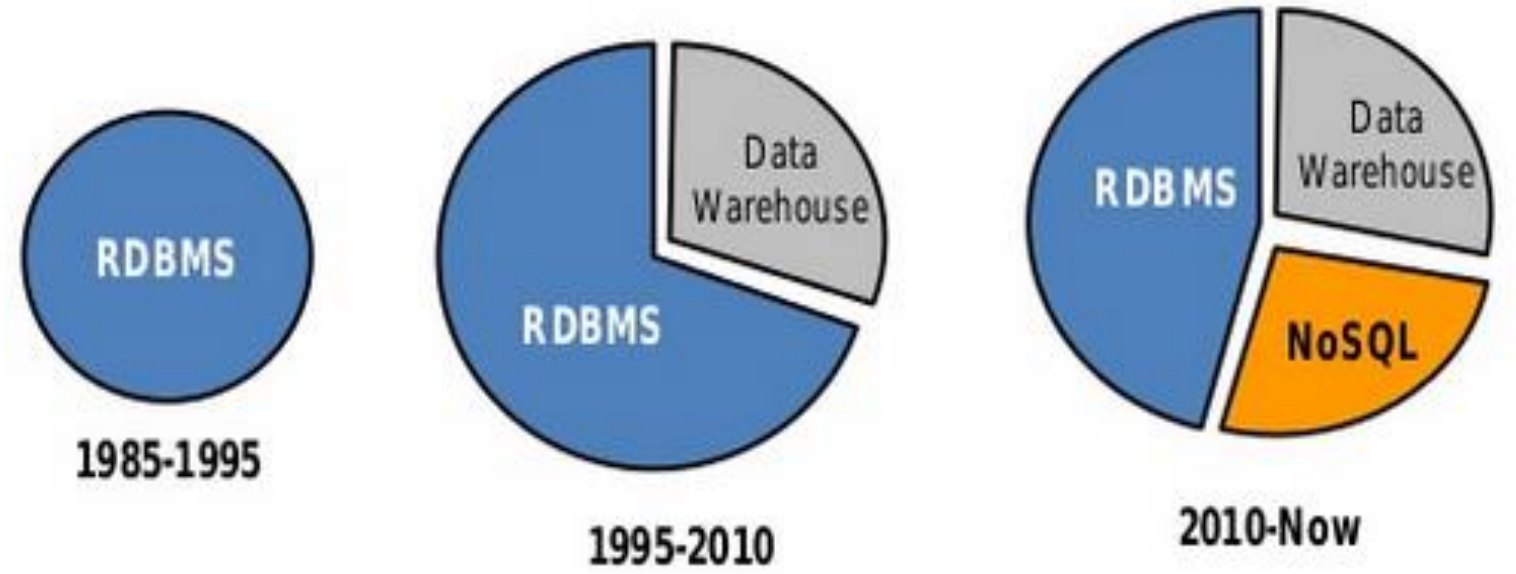
Que faut-il retenir ?



Guillaume HARRY | Olivier PORTE

Que faut-il retenir ?

- Les bases NoSQL ont une part importante dans le paysage des SGBD



- NoSQL est né pour répondre à des besoins du Web 2.0

Aller ou non vers le NoSQL ?

Magic Quadrant for Operational Database Management Systems

ATTENTION aux préjugés !

1. **NoSql, c'est plus facile pour démarrer → FAUX**

   Il n'existe pas à l'heure actuelle de base NoSQL embarquée qui arrive à la cheville de SQLite : ça marche partout, dans tous les langages, sans rien à avoir installer ou configurer pour la plupart des langages.

2. **Avec NoSql, pas besoin de réfléchir au modèle de données → FAUX**

   Une base NoSQL ne vous oblige pas à formaliser votre schéma, mais ça ne veut CERTAINEMENT PAS dire qu'il ne faut pas le faire. L'auteur de Redis a très bien expliqué le problème

   > *Redis is not the kind of system where you can insert data and then argue about how to fetch those data in creative ways. Not at all, the whole idea of its data model, and part of the fact that it will be so fast to retrieve your data, is that **you need to think in terms of organising your data for fetching**. You need to design with the query patterns in mind.*

3. **NoSql, c'est plus performant → FAUX**

   Les performances, ça dépend toujours du contexte.
   Par exemple, Redis est plus performant à la lecture et l'écriture, mais les données doivent tenir en RAM

4. **NoSQL remplace le SQL → FAUX**

Que faut-il retenir ?

- **SGBDR ne peut pas être remplacé**
  - Traitement transactionnels (propriété ACID)
  - Facilité d'apprentissage du langage SQL
  - Recherche de donnée avec de multiples conditions
  - Connecteurs standards

- **Mais**
  - Propriétés ACID
  - Problèmes de performances
  - Problèmes de gestion des gros volumes de données

Que faut-il retenir ?

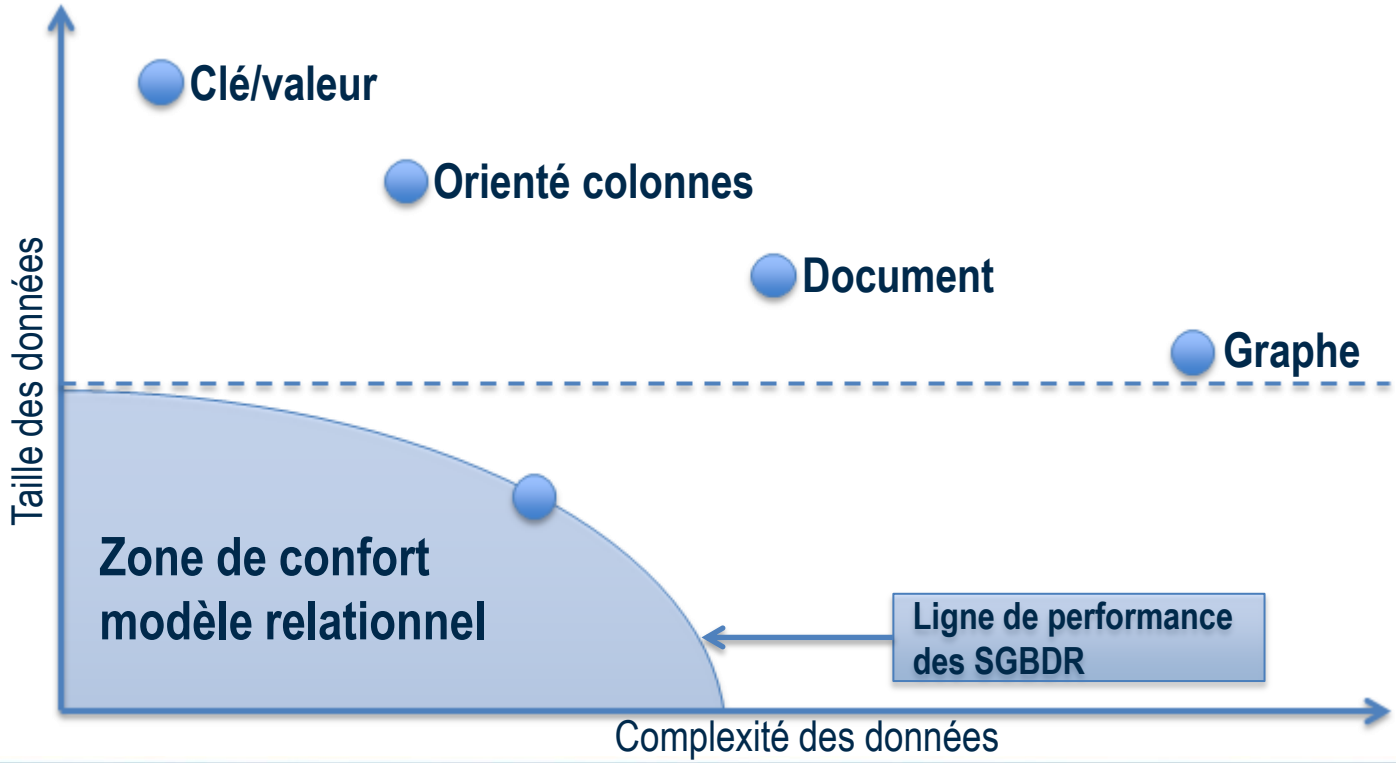
- **NoSQL ne peut pas être remplacé**
  - Données volatiles
  - Débit de données en écriture/lecture
  - Moindre investissement dans le design du modèle (flexibilité, données semi structurées)
  - Distribution, réplication des données nativement
  - Distribution des traitements
  - Très haute disponibilité
  - Faible coût des architectures

- **Mais**
  - Pas de langage standardisé
  - Interrogation des données nécessite expertise
  - Modèle de données est détenue par l'application
  - Choix à faire entre consistance, disponibilité et tolérance
  - Peu d'outils offrent des connecteurs vers des bases NoSQL

Quand aller vers NoSQL ?

- Faire cohabiter différents paradigmes (bases de données relationnelles et NoSQL) est la seule bonne réponse !

- Ce qui importe, c'est de choisir le paradigme et le moteur qui permet de répondre au besoin



Se poser les bonnes questions !

- Comment modéliser ses données ?
- Quel est le pattern d'accès aux données dont a besoin l'application ?
- Quel est le besoin d'intégrite, de consistance ?
  Que se passe-t'il si 2 applications accèdent à la même donnée en lecture/écriture ?
- Quelles sont les performances, la scalabité et la complexité de la solution choisie en répondant à ses questions ?

Vous ne pourrez commencer à créer votre base de données et votre application que quand vous aurez répondu à ces questions !!!