NoSQL : Jointures, dénormalisation & JSON

Nicolas.travers@devinci.fr

plan

I. JSon & Nosql orienté documents
- a) Structures
- b) Listes
- c) Identifiants
- d) Mise en pratique

II. Le problème des jointures
- a) Jointures et NoSQL
- b) Modélisation de documents
- c) Mise en pratique

Nicolas.travers@devinci.fr

DONNÉES SEMI-STRUCTURÉES AVEC JSON

Nicolas.travers@devinci.fr

JSon : Javascript Object Notation

- **JSON** (JavaScript Object Notation)
  - Créé pour les échanges entre navigateur et serveur Web
  - Léger, orienté texte, indépendant d'un langage d'interrogation
  - Utilisé par certains Web Services (Google API, Twitter API) ou web dynamique (Ajax)

- Versus

- **XML** utilisé initialement pour les échanges de communications Machine/Machine
  - Trop verbeux et coûteux en place
  - Dédié aux Services Web

Structures & Types

- Concept : Clé + Valeur
  - “nom” : “Travers”
  - Clés avec des guillemets, ne pas utiliser ”.-,”
- Objet : collection de paires “clé + valeur”
  - Encapsulé dans des accolades
  - { “nom” : “Travers”,
    “prenom” : “Nicolas”,
    “genre” : 1}
- Liste ordonnée de valeurs
  - { “liste” : [“SQL”, “XML”, “NoSQL”, “Optimisation BD”, “RI”] }
- Types de données
  - Atomique : String, Integer, flottants, booléens…
  - Liste : tableaux
  - Documents : objets {…}

Liste de valeurs : précisions

- Pas de contraintes sur le contenu de la liste
  - “cours” : [“SQL”, 1, 4.2, null, “Recherche d’Information”]
- Peut contenir des objets hétérogènes
  - “doc” : [{“test” : 1},
    {“test” : {“imbrication” : 1.0}},
    {“test” : “texte”, “valeur” : null}
    ]

Identifiant

- La clé « _id » est communément utilisée pour identifier un objet
  - Dans une base, l'objet de même identifiant écrase la version précédente
  - Peut être défini automatiquement
    - Exemple MongoDB : "_id" : ObjectId(1234567890)

Exemple complet

```
{
  “_id” : 1234,
  “nom” : “Travers”, “prenom” : “Nicolas”,
  “travail” : [
       {“etablissement” : “Cnam”,
        “localisation” : {
            “rue” : “2 rue conté”,
            “ville” : “Paris”,
            “CP” : 75141
       }
  ],
  “themes” : [ “BD” , “Optimisation BD”, “XML”, “NoSQL”, “RI” ]
}
```

Conclusion

- Concept **clé / valeur**
- Format utile pour les **documents**
- Format simple, léger et flexible
  - “cours” : [“SQL”, 1, 4.2, null, “Recherche d’Information”]
- Imbrication de données (dénormalisation)
- Non standardisé
  - “cours” : [“SQL”, 1, 4.2, null, “Recherche d’Information”]

Nicolas.travers@devinci.fr

LE PROBLÈME DES JOINTURES

Nicolas.travers@devinci.fr

NoSQL vs Jointures

- Contexte distribué incompatible avec les liens entre les collections de données
  - Coût basé sur les communications réseaux
  - Jointure multi-serveurs

- Si l'on considère deux collections de données
  - Pour une jointure entre les deux collections
    - Sur quel serveur se fait la jointure ?
    - Distribution des données à joindre ?
    - Problème de coût réseau (plus cher qu'en local)

Jointures : Solutions (1/2)

1. Effectuer la **jointure au niveau de l'application**
   - Récupérer les éléments de coll1
   - Pour chaque élément de coll1 (X), interroger coll2 (Y)
   - Peut être très couteux
   - Equivalent relationnel d'une boucle imbriquée avec index non optimisée et coup réseau supplémentaire
   - ➡ X + X * Y
2. Réunir le tout dans **une même collection**
   - Collection groupant coll1 (X) et coll2 (Z)
   - MapReduce de « jointure » dans le *Reduce (X'+Z')*
   - Coût du *« shuffle »* augmenté, ainsi que toute requête sur une des deux collections
   - Equivalent relationnel d'un index de type *Cluster*
   - ➡ (X+Z) + (X'+Z')

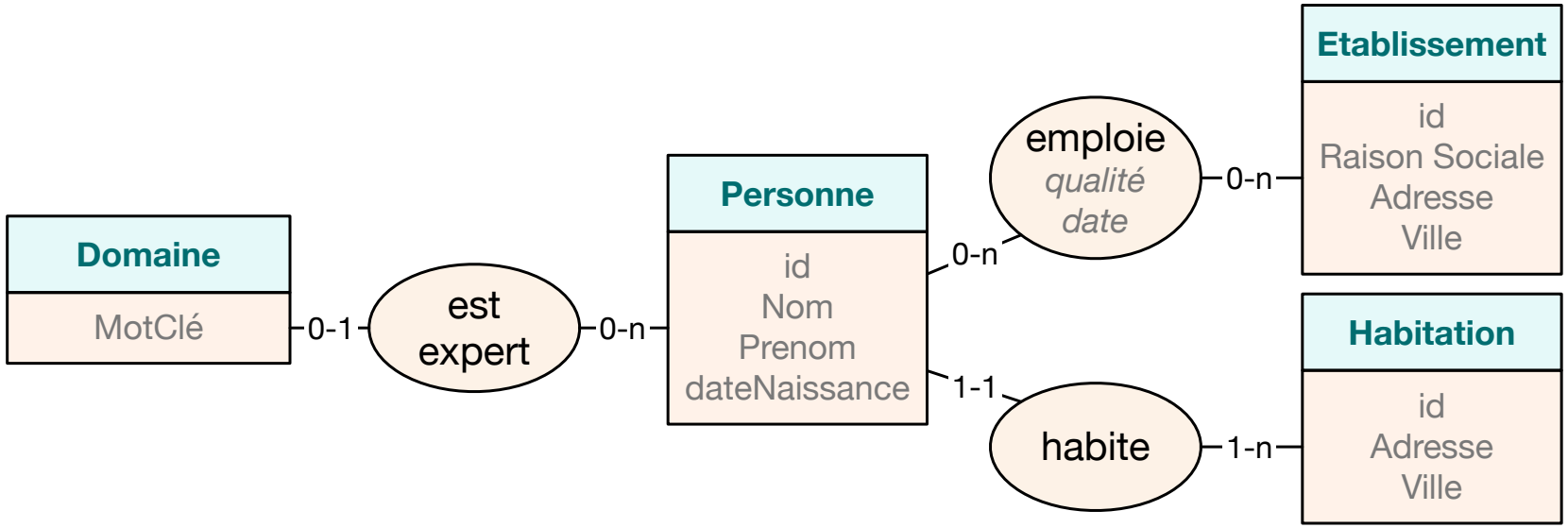
Nicolas.travers@devinci.fr

Jointures : Solutions (2/2)

3. **Dénormaliser** le schéma
   - Créer une collection (X) en regroupement les deux collections
   - Créer des sous-documents imbriqués contenant les valeurs jointes (cf JSon)
   - Problème de cohérence des données (répétitions)
   - Equivalent au résultat de la jointure
   - Demande un travail de réflexion et de modélisation
   - ➡ X
4. **Framework** d'exécution de jointure sur les serveurs
   - Les données de jointures sont distribuées sur les serveurs pour un calcul local
   - La base NoSQL doit permettre cette distribution (ex: *Hadoop/Pig, MongoDB/$lookup*)
   - Dépend de la taille de la collection à distribuer
     - Algorithmes de jointure basés sur le relationnel (hachage, tri-fusion)
   - ➡ dépend du tri…

Jointures & Modélisation

- **Comment passer du schéma vers le document ?**
  - Quelle entité centrale utilisée ?
  - Comment les fusionner ?
  - Quels sont les risques ?



Jointures & modélisation : méthodologie

- **Méthodes de modélisation :**
  - Query-Driven [Chebotko15]
  - Document-oriented NoSQL modelling [Mason15]
  - Model-Driven-Architecture [Abdelhedi et al. 17]
  - But :
    - Eviter les requêtes de jointure couteuses
    - Rassembler les données peu mises à jour ou indépendantes

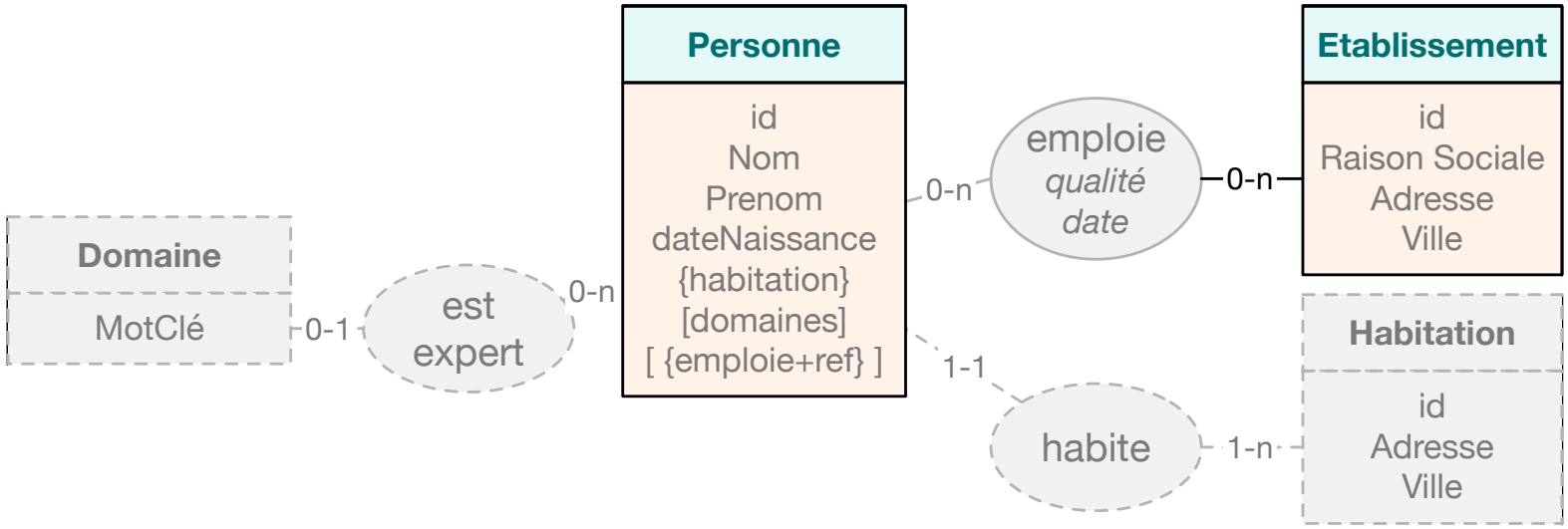
⚠ **Attention à l'incohérence liée aux répétitions !**

1 – [Chebotko15] https://pdfs.semanticscholar.org/22c6/%20740341ef13d3c5ee52044a4fbaad911f7322.pdf
2 – [Mason15] http://proceedings.informingscience.org/InSITE2015/InSITE15p259-268Mason1569.pdf
3 – [Abdelhedi et al. 17] https://cedric.cnam.fr/index.php/publis/article/AAA17

Nicolas.travers@devinci.fr

Jointures & modélisation : Etapes

- **Principales étapes de modélisation :**
  1. Fréquence d'interrogation *(jointure : personne et habitation)*
  2. Données indépendantes *(jointure : personne et domaine)*
  3. Relations 0-n des deux côtés : imbrication pour références *(emploie liste de références)*
  4. Même taux de mises à jour *(personne et emploie)*
  5. Relation 1-1 des deux côtés



Nicolas.travers@devinci.fr

Jointures & modélisation : Résultat

- {
- "_id" : 1,
- "nom" : "Travers",
- "prenom" : "Nicolas",
- “habitation” : {“adresse” : “292 rue Saint Martin”, “ville” : “Paris”},
- "domaines" : ["SGBD", "NoSQL", "RI", "XML"],
- "emplois" : [
- {"id_etablissement" : "100", "qualité" : "Maître de Conférences", "date" : "01/09/2007"},
- {"id_etablissement" : "101", "qualité" : "Vacataire", "date" : "01/09/2012"}
-]
- }

Nicolas.travers@devinci.fr

Conclusion

- Modélisation avec des Documents JSON
- Dénormalisation complexe
  - Attention aux mises à jour
  - Attention à l'incohérence