PRISM

Création et Maintenance de mappings pour l'intégration de données XML

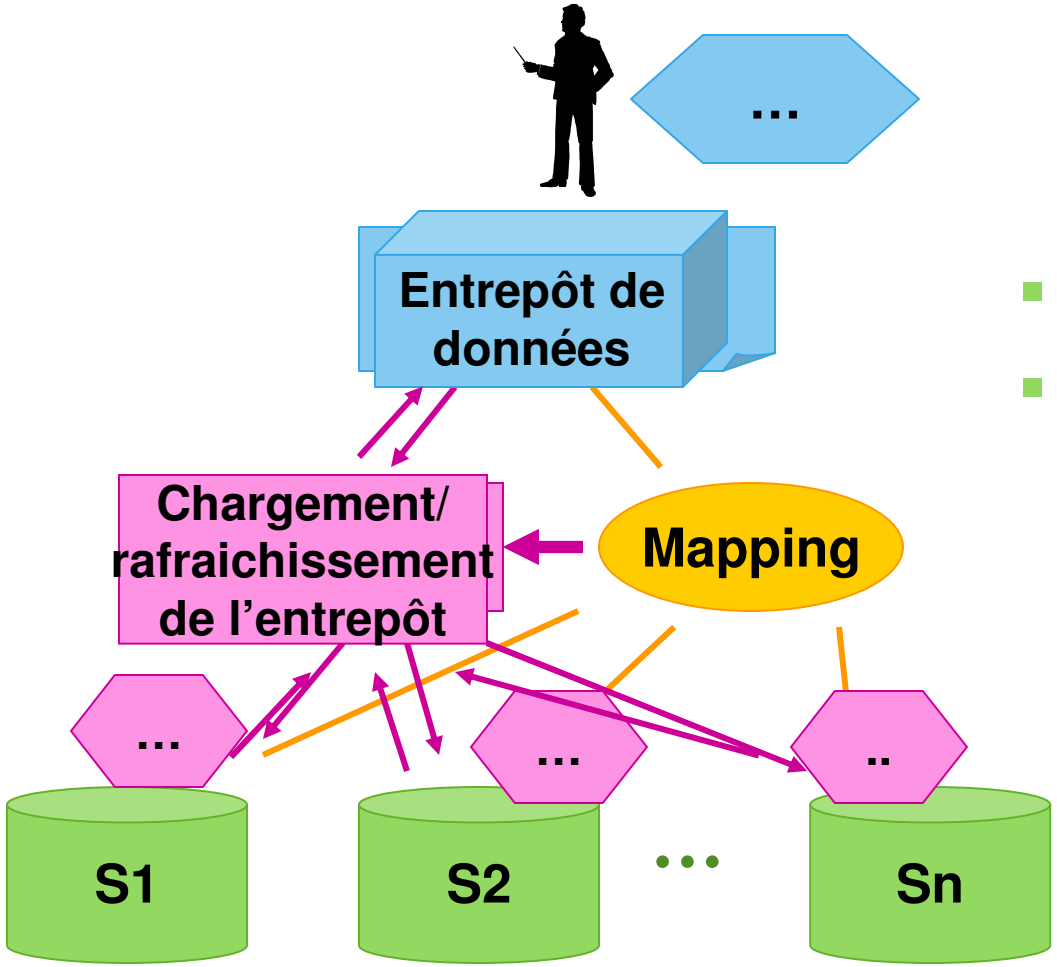
Xiaohui Xue, Zoubida Kedad

Laboratoire PRiSM

Université de Versailles

Contexte

- **Intégration de sources de données hétérogènes**



- Systèmes de médiation
- Entrepôts de données

Mapping

- Une expression spécifiant la façon dont les instances d'un schéma cible sont dérivées à partir des instances des schémas sources

Schéma cible

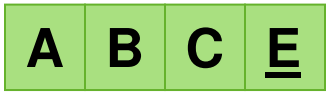
Mapping

$$\pi_{(A, B, C, D)}((R1 \cup R3) \bowtie_{E} R2)$$

Schémas sources

R1

R2



R3

Problématique

- **Création de mappings**
  - Quelles sont les sources contributives permettant de dériver un mapping pour le schéma cible ?
  - Si plusieurs sources peuvent être utilisées, comment les combiner ?
  - S'il existe plusieurs solutions, quelle est la « bonne solution » ?
- **Maintenance de mappings**
  - Comment détecter les changements dans les sources ?
  - Comment redéfinir le mapping : relancer la création ou adapter le mapping existant ?

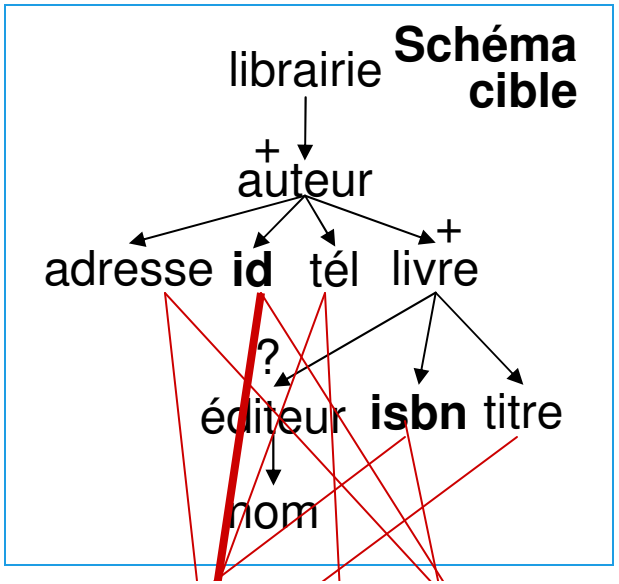
Notre objectif

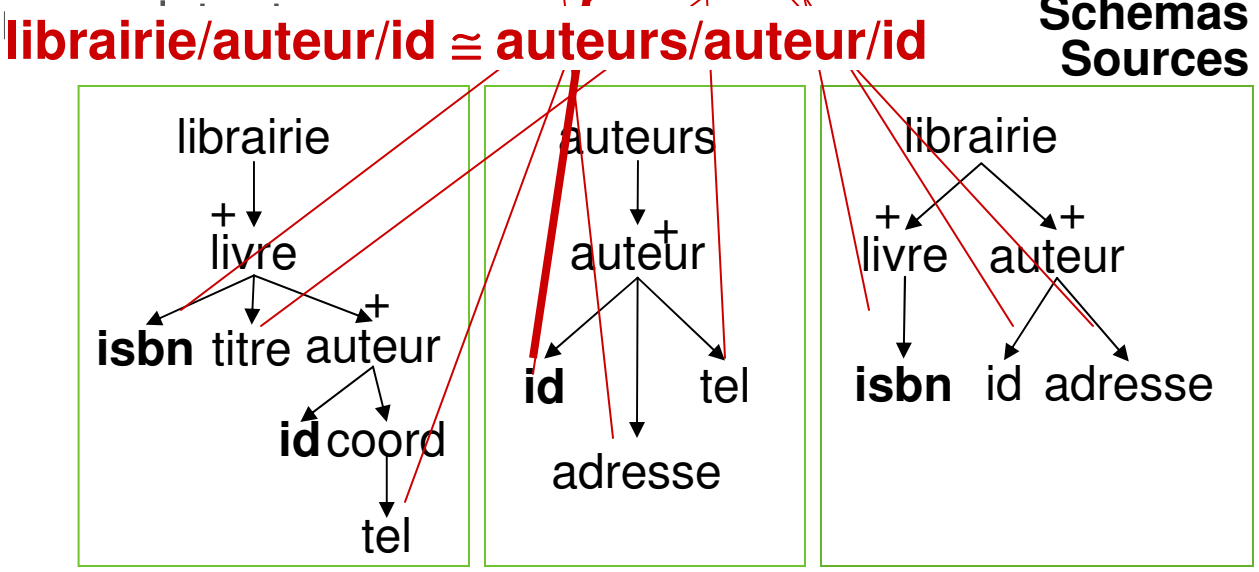
Étant donné :

- un schéma cible
- Plusieurs schémas sources
- Des correspondances sémantiques

Objectif : Produire des mappings XQuery qui instancient le schéma cible

- Création from-scratch
- Maintenance de mappi

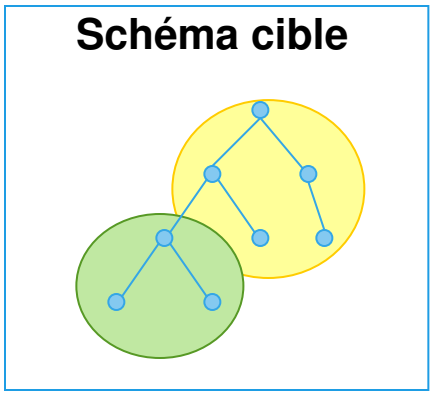


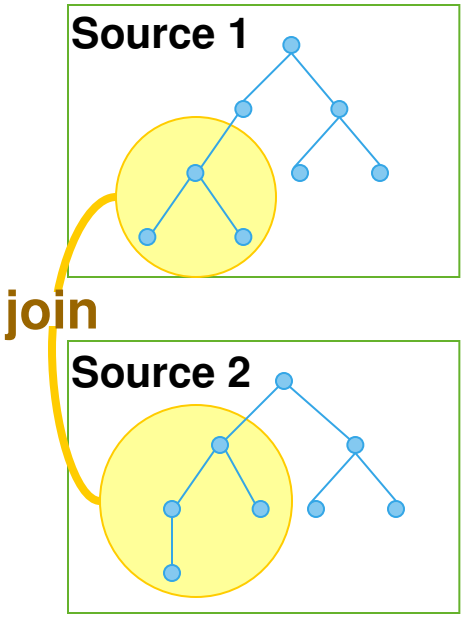


Création de mappings

Décomposition du problème

- Décomposition du schéma cible en sous arbres
- Recherche de mappings permettant d'instancier chacun des sous arbres (mappings partiels)
- Génération de mappings à partir des mappings partiels





Mappings

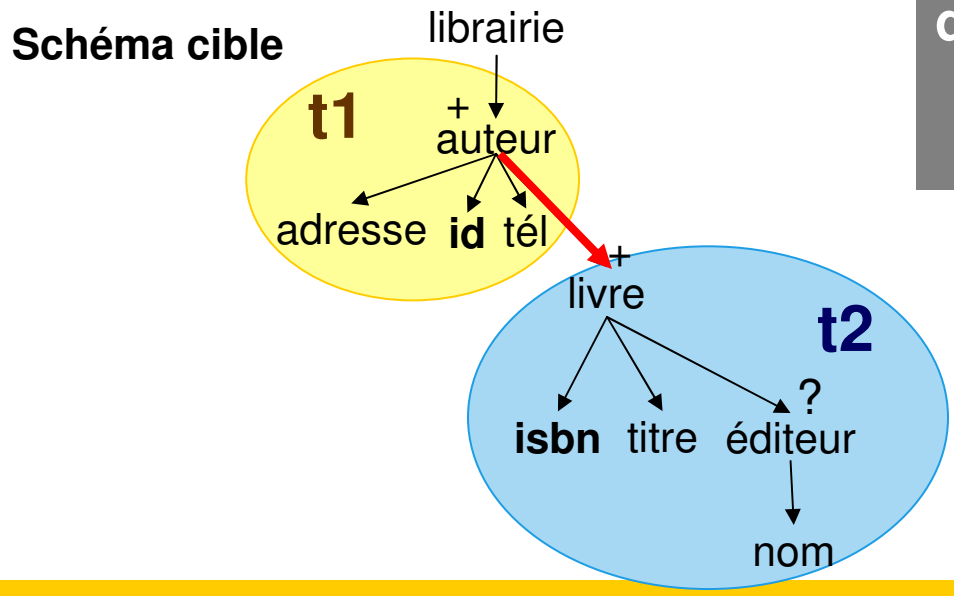
for ..
where ..
return ..

for ..
whe
return ..

Décomposition du schéma cible

- **Problématique**
  - Granularité des sous arbres issus de la décomposition ?
  - Comment identifier ces sous arbres ?
- **Définition d'un sous arbre (hiérarchie d'un ou plusieurs niveaux)**
  - Le nœud racine est un élément multivalué
  - Les autres éléments sont monovalués
  - Il y a au moins un élément texte

Procédure de décomposition :
Parcours d'arbre en partant de la racine

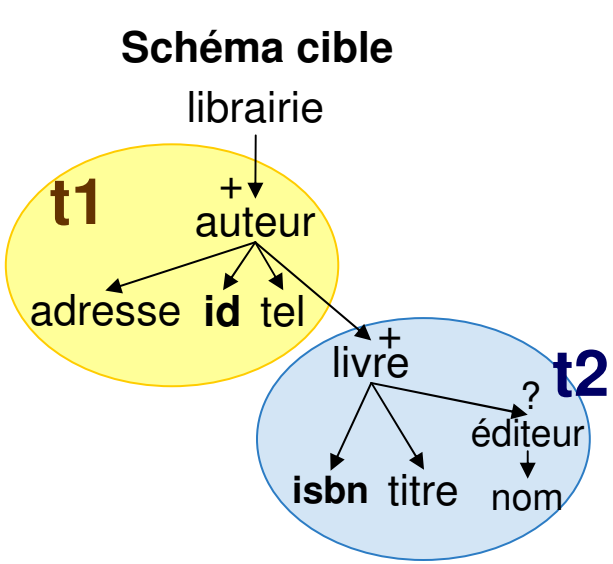


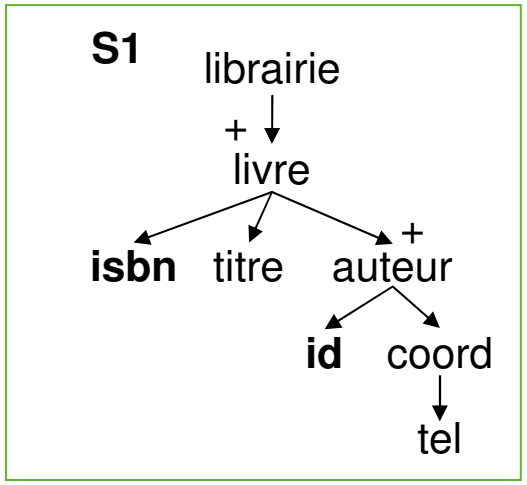
Définition de mappings partiels

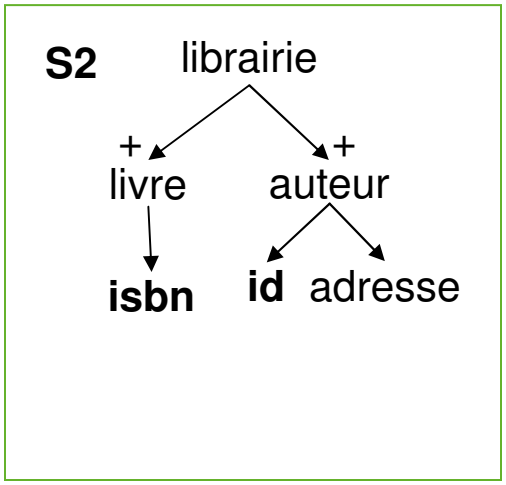
- **But : trouver des façons de dériver des instances de chaque sous arbre cible à partir des sources**
  - Quelle est la granularité d'une portion source contributive ? Comment les toutes identifier ?
  - Comment inférer les jointures permettant de les combiner ?
  - Comment construire des mappings partiels à partir des portions sources contributives et des jointures ?

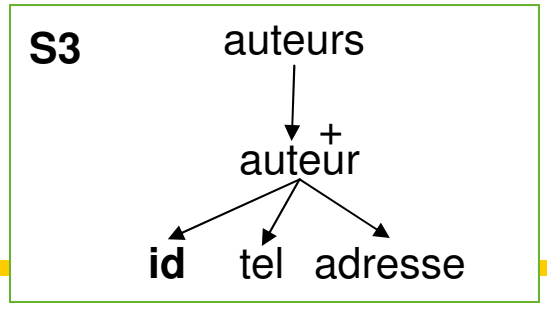
Identification de parties sources 1/2

- **Définition** – étant donné un sous arbre cible, une partie source est un ensemble d'éléments sources tel que :
  - Chacun élément est équivalent à un élément du sous arbre cible
  - Ils sont dans un sous arbre dans lequel tous les éléments sont monovalués à part la racine









Identification de parties sources 2/2

Principe

- Pour chaque sous arbre cible T et pour chaque source S
  - Déterminer l'ensemble E des éléments de S équivalents à des éléments de T
  - Vérifier si chaque sous-ensemble de E est une partie source

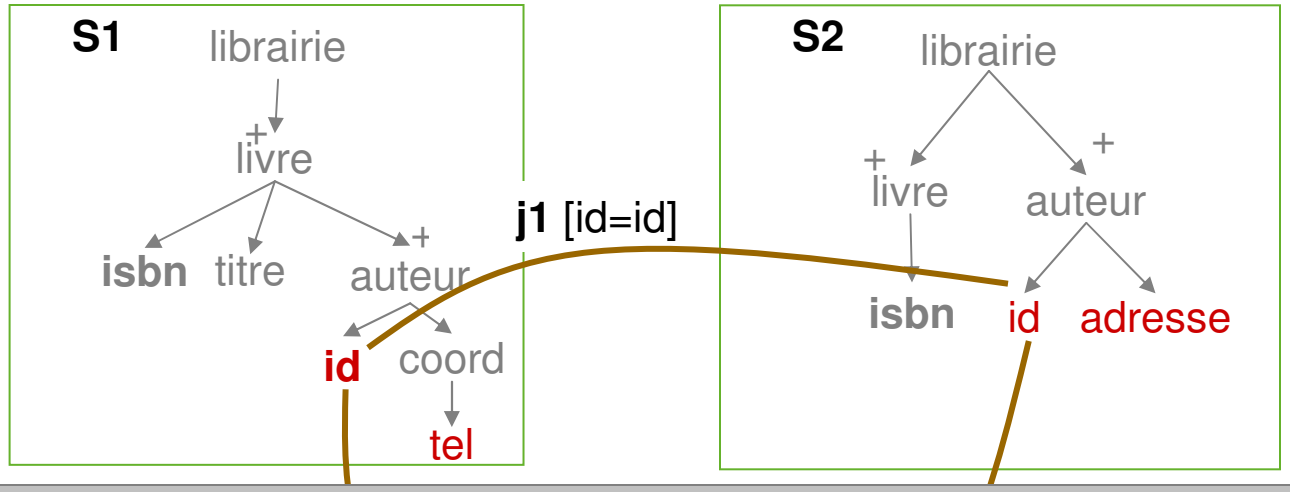
Complexité

- Pour un sous arbre cible et une source : $O(2^n)$

  n : nombre de correspondances sémantiques

Identification de jointures 1/2

- **But :** inférer des jointures pour combiner les différentes parties sources
- Les jointures sont identifiées en utilisant les contraintes définies dans les sources



Règle : il y a une jointure entre deux parties sources de schémas différents si:

Il y a une correspondance entre 2 éléments de ces parties sources, et l'un des deux éléments est défini comme une clé

Identification de jointures 2/2

Principe

- Pour chaque partie source
  - Pour chaque définition de clé sur un élément K
  - Rechercher dans tous les autres partie sources un élément K' équivalent à K
  - Proposer une jointure pour chaque K' trouvé

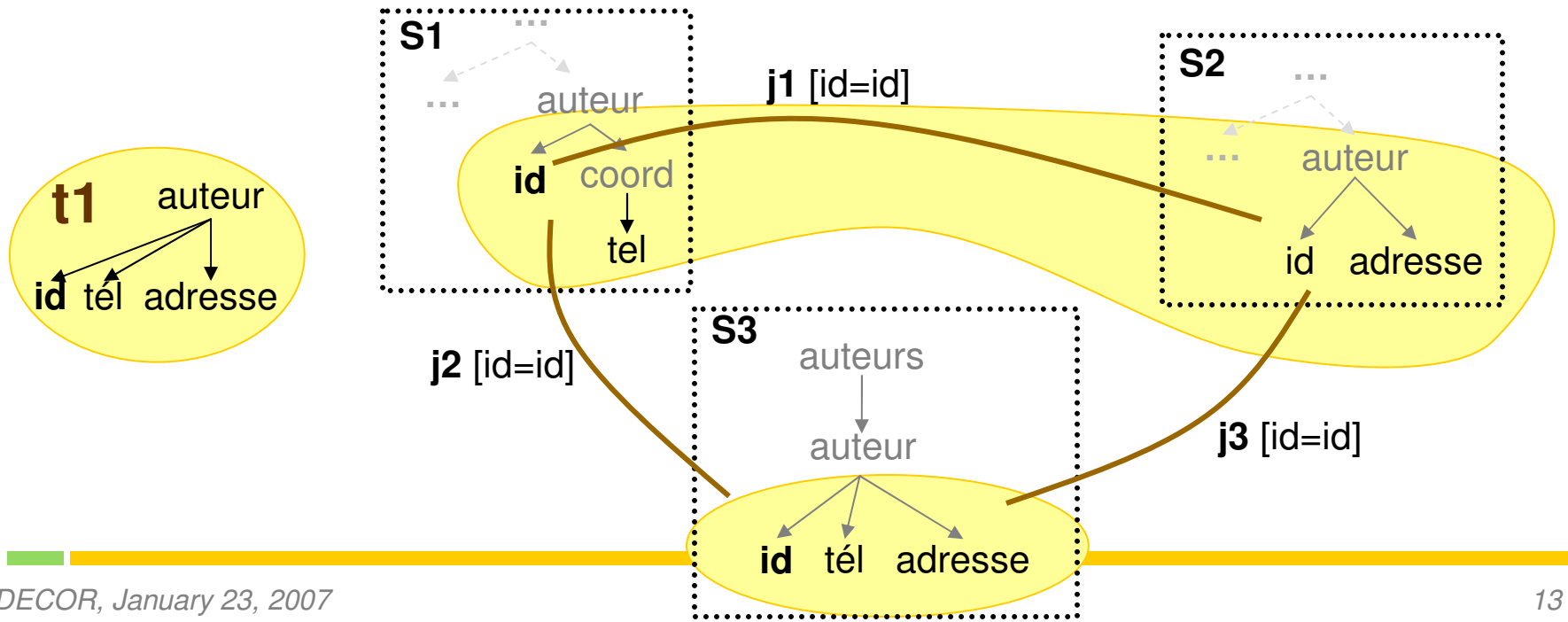
Complexité

- Pour un sous arbre cible : $O(n^2)$

  n : nombre de parties sources

Définition de mappings partiels 1/2

- **Définition - graphe de jointures :**
  - Le graphe formé par l'ensemble de jointures identifiées pour un sous arbre cible
  - Chaque nœud représente une partie source
  - Chaque arête représente une jointure
- **Chaque sous graphe connexe et acyclique du graphe de jointure est un mapping partiel**



Définition de mappings partiels 2/2

Principe :

- Recherche de l'ensemble des sous graphes connexes et sans cycle dans un graphe

- Complexité :
  - Pour un sous arbre cible : $O(n^p)$

    n : le nombre de nœuds; p : le nombre d'arêtes

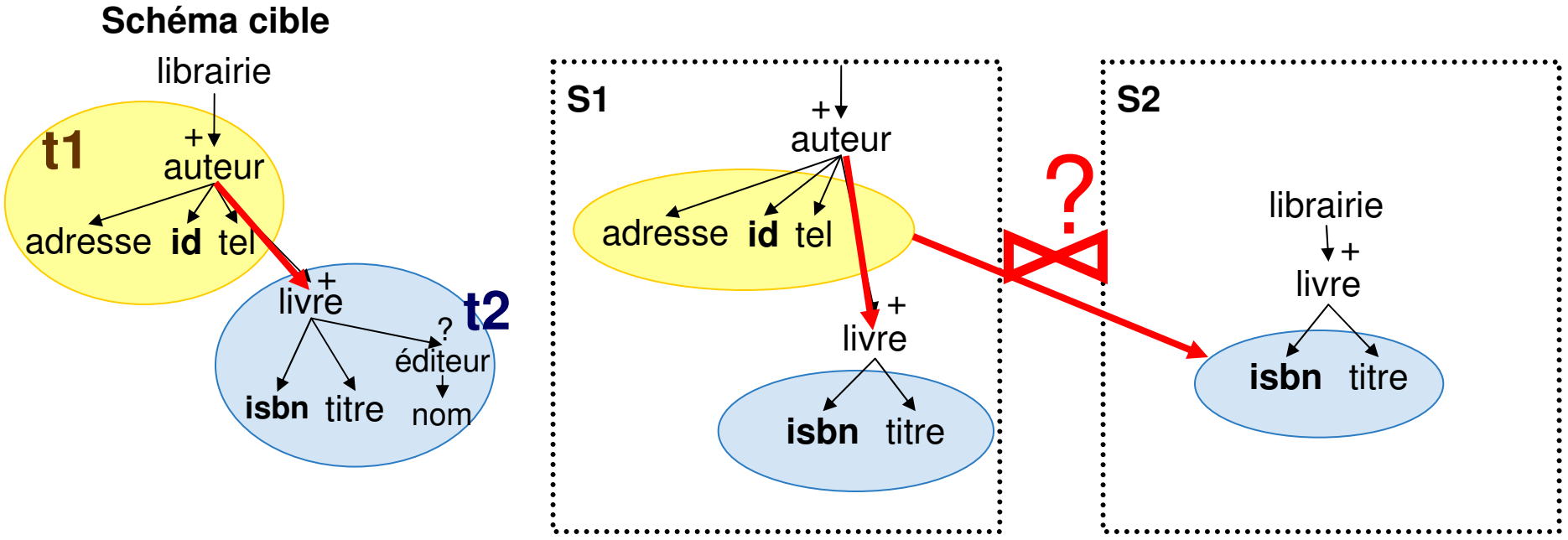
➔ Rechercher les K plus petits sous graphes

➔ Rechercher les K premiers

➔ …

Génération de mappings

- **Comment retrouver dans les sources les liens entre sous arbres cibles :**
  - Le lien est contenu dans une source : il y a un lien hiérarchique entre deux parties sources correspondants
  - Le lien n'est pas contenu dans une source : recherche de jointures comme précédemment



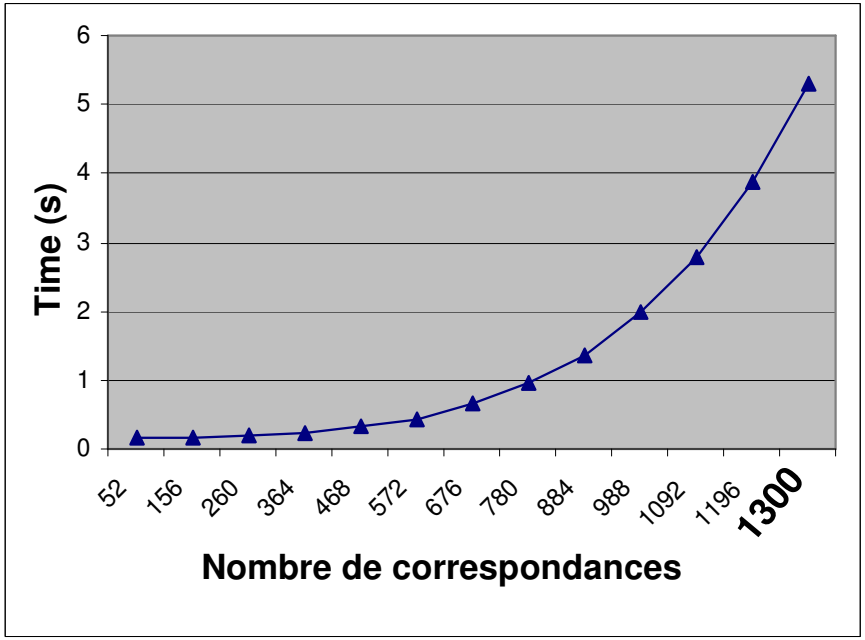
Evaluation de performance 1/2

- **Les paramètres considérés ayant une influence sur les performances de la génération de mappings**
  - Nombre d'éléments sources
  - Nombre d'éléments cibles
  - Nombre de correspondances sémantiques
  - Nombre de contraintes dans les sources

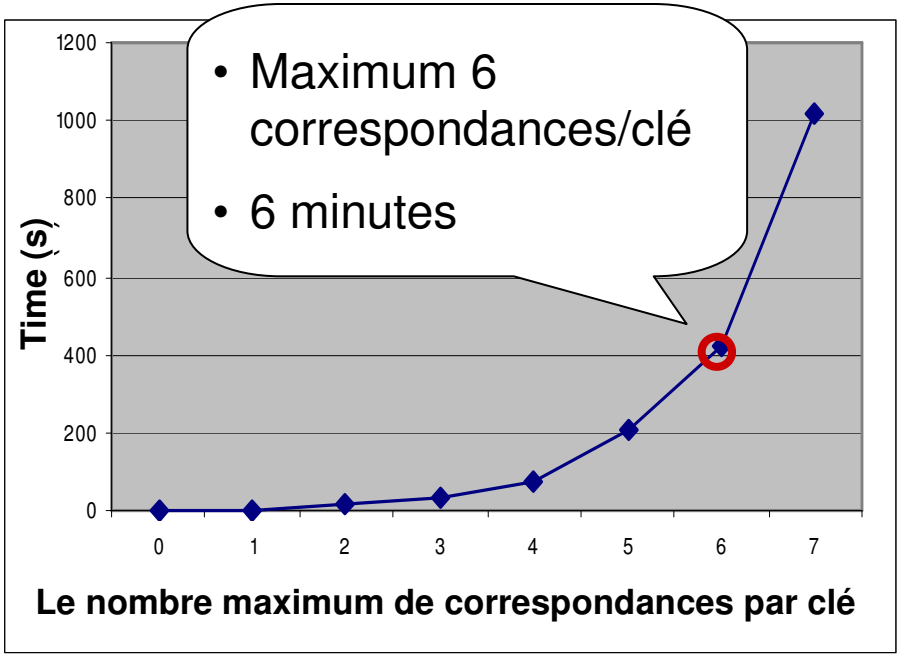
- **Résultats des tests d'évaluation de performance**
  - 18 éléments cibles – 56 éléments sources – 26 correspondances : 1 seconde
  - 47 éléments cibles – 1650 éléments sources – 1300 correspondances : 7 minutes

Evaluation de performance 2/2

- **Schéma cible : 47 éléments – 7 niveaux**
- **50 schémas sources – 30 éléments par source**

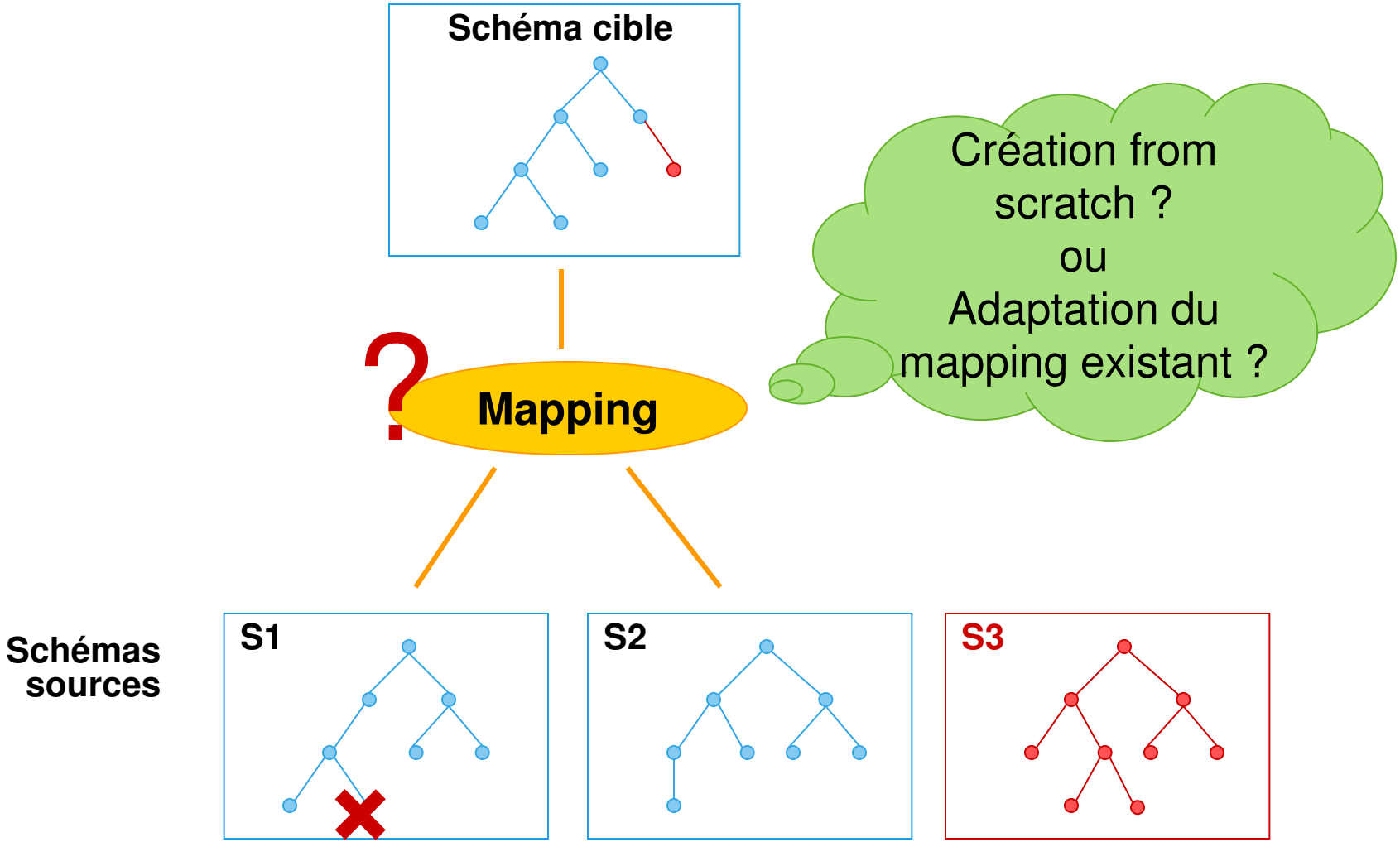


Identification de jointures



Recherche de mappings partiels

Maintenance de mappings



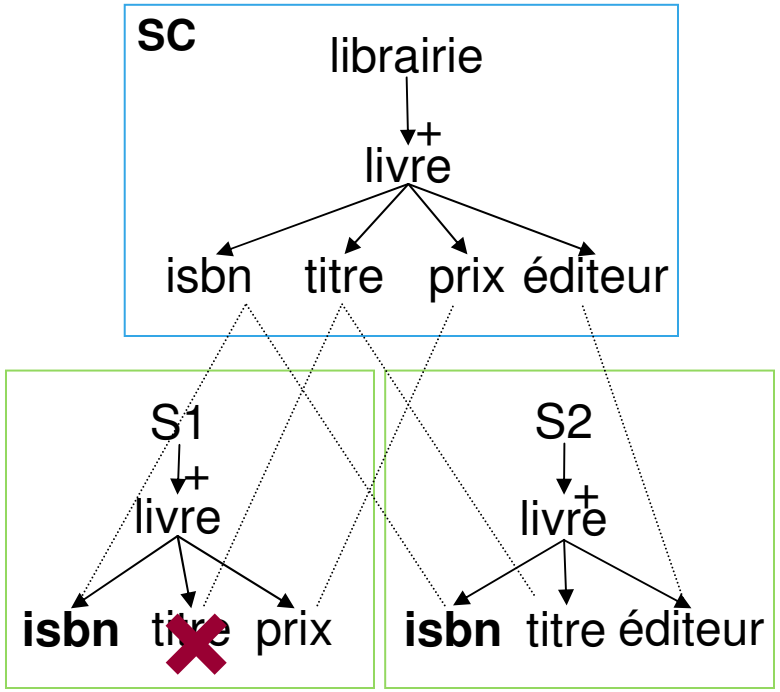
Hypothèse

- **Pattern de mappings**

```
[for      $variable_element_binding in path ..]
[where    [join_condition [and join_condition ..]]
          [grouping_condition]]
[return] target_assignments
```

Effets des changements sur les mappings

- **Trouver les mappings invalidés par un changement :**
  - Identifier les clauses du pattern concernées par les changements



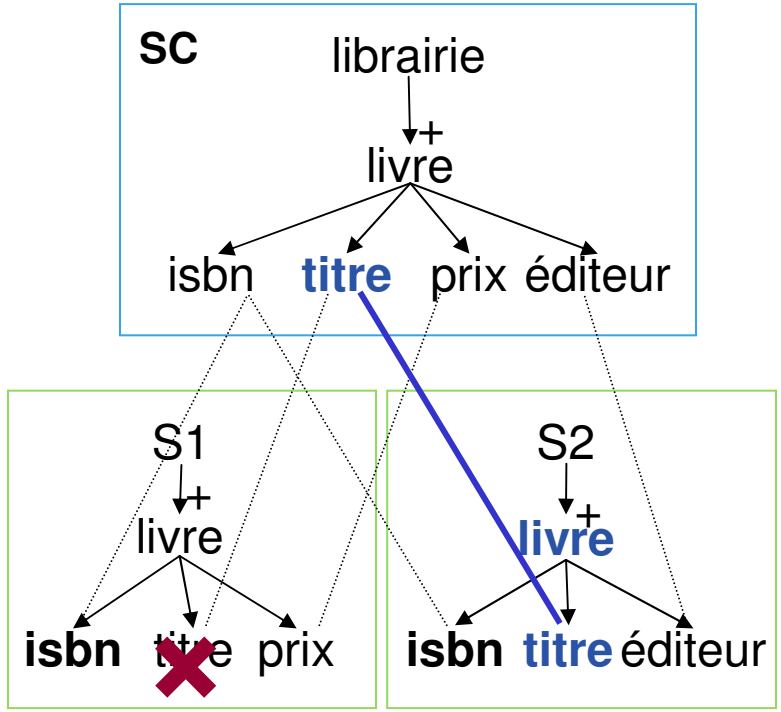

```
<librairie>{
   for $v1 in S1/livre
   for $v2 in S2/livre
   where $v1/isbn = $v2/isbn
   return <livre>{
     <isbn>{data($v1/isbn)}</isbn>,
     <titre>{data($v1/titre)}</titre>,
     <prix>{data($v1/prix)}</prix>,
     <éditeur>{data($v2/éditeur)}</éditeur>
   }</livre>
}</librairie>
```

Actions réparatrices

Règle de substitution : un élément source peut être utilisé pour instancier un élément cible si:

- Ces deux éléments sont équivalents
- Il existe un élément d'une clause **for** tel que l'élément source soit mono-valué

```
<librairie>{
    for $v1 in S1/livre
    for $v2 in S2/livre
    where $v1/isbn = $v2/isbn
    return <livre>{
      <isbn>{data($v1/isbn)}</isbn>,
      <titre>{data($v2/titre)}</titre>,
      <prix>{data($v1/prix)}</prix>,
      <éditeur>{data($v2/éditeur)}</éditeur>
    }</livre>
}</librairie>
```



Principe de l'algorithme de maintenance

```
Pour chaque changement C
début
    Pour chaque mapping M
    début
        identification_clauses_invalidées(M, C);
        pour chaque clause invalidée I
        début
            recherche_réparation(I, C);
            pour chaque solution de réparation trouvée S
                réparation_mappings(M, S);
        fin
    fin
fin
```

Conclusion et perspectives

- **Contributions**
  - Proposition d'une méthode de création de mappings et d'une approche de maintenance de mappings pour des schémas XML
  - Implémentation des algorithmes de création de mappings et évaluation des performances du système

- **Perspectives**
  - Prise en compte de facteurs de qualité et de préférences utilisateurs
  - Évaluation de performance de la maintenance mappings
  - Détection de changements sources