Génération automatique des requêtes de médiation dans un environnement hétérogène

Soukane Assia

UNIVERSITE de VERSAILLES SAINT-QUENTIN EN YVELYNES

THESE

Pour obtenir le diplôme de

DOCTEUR DE L'UNIVERSITE DE VERSAILLES SAINT-QUENTIN EN YVELYNES

Spécialité : Informatique

Présentée et soutenue par

Assia SOUKANE

Le 8 décembre 2005

Sujet de la thèse

GENERATION AUTOMATIQUE DES REQUETES DE MEDIATION DANS UN ENVIRONNEMENT HETEROGENE

Membres du jury

Anne DOUCET Professeur à l'Université de Paris VI - Président
Omar BOUCELMA Professeur LISIS à l'Université d'Aix-Marseille 3 - Rapporteur
Ana Carolina SALGADO Professeur à l'Université Fédérale du Brésil - Rapporteur
Isabelle COMYN-WATTIAW Professeur au CNAM Paris - Examinateur
Mokrane BOUZEGHOUB Professeur à l'Université de Versailles - Directeur
Zoubida KEDAD Maître de Conférences à l'Université de Versailles - Examinateur

Remerciements

Je tiens à présenter mes plus vifs remerciements

A Mokrane Bouzeghoub,

Il a inspiré le sujet de cette thèse, il a veillé à sa réalisation ne ménageant ni son temps ni ses précieux conseils, qu'il trouve ici le témoignage de ma respectueuse considération et de ma profonde gratitude.

A Daniela Grigori, Zoubida Kedad, Stephane Lopes et Elisabeth Métais,

J'ai été marquée par le vif intérêt qu'ils ont porté à ce travail, et par leurs encouragements et précieux conseils.

Aux membres du jury,

Ils nous font honneur d'accepter de juger ce travail, qu'ils trouvent ici l'expression de notre reconnaissance.

A Dimitre Kostadinov, Xiaohui Xue,

Pour leur contribution à cette thèse, et leur aide précieuse et amicale.

A Pengfei Liu,

Pour sa précieuse contribution aux évaluations du prototype, je lui exprime mes vifs remerciements,

A tout le personnel du laboratoire PRiSM,

Je lui exprime mes respects les plus profonds.

A mon époux, *Malik*,

Pour toute son attention et son soutien précieux et constant, qui m'ont aidé à mener au mieux ce travail, je lui exprime mes sentiments les plus sincères et profonds.

A mon père et à ma mère,

Avec l'expression de ma haute gratitude et de mon profond respect, en témoignage de leur soutien constant et à tous les principes qu'ils m'ont inculqués, que je n'oublierai jamais.

A mes sœurs, *Radia*, *Latifa* et *Hind*.

Aux familles, Bentounes, Kadi et Soukane.

A tous mes amis, pour leur aide précieuse et amicale.

Table des matières

Résumé

Les systèmes de médiation sont aujourd'hui largement développés et connus. Cependant, leur mise en œuvre pose un certain nombre de problèmes, en particulier la définition de requêtes de médiation en présence d'un grand nombre de sources de données, et d'un volume important de méta-données les décrivant. Ce problème est d'autant plus complexe que les sources sont hétérogènes.

Face a cette problématique, nous proposons dans cette thèse pour le contexte relationnel, une approche de génération automatique de requêtes de médiation. A partir de la description d'un ensemble de sources de données distribuées et hétérogènes et de méta-données, notre algorithme produit un ensemble de requêtes de médiation possibles. Nous avons développé un outil qui permet de générer automatiquement des requêtes de médiation dans un environnement hétérogène. Notre objectif principal étant de fournir à l'utilisateur un outil adapté aux petits et grands systèmes, nous avons réalisé une série de tests d'évaluation des performances pour mesurer son passage à l'échelle. Ces tests montrent la faisabilité de notre approche.

Abstract

Nowadays, mediation systems are widely used. However, their implementation raises several problems, especially the definition of mediation queries when there is a high number of sources, and an important amount of metadata. The heterogeneity of the data sources makes this problem even more complex.

We propose in this thesis an approach to automatically generate the mediation queries in a relational context. Our algorithm produces a set of possible mediation queries given the description of a set of heterogeneous and distributed data sources. We have developed a tool to automatically generate mediation queries in a heterogeneous context. We provide some performance evaluations to test the scalability of our tool, and to show its usability for systems of different size.

Chapitre1- Introduction

1-1 - Contexte

De nos jours, les systèmes multi-source sont de plus en plus développés. Ils sont définis comme l'intégration de plusieurs sources hétérogènes et distribuées. Parmi ces systèmes d'informations, nous distinguons les entrepôts de données, les systèmes d'informations basés sur le web, les systèmes de bases de données fédérées, ou encore les systèmes de médiation. Notre étude s'inscrit principalement dans le contexte des systèmes de médiation.

Un système de médiation est un système qui permet d'interopérer sur un ensemble de sources hétérogènes et distribuées. Ses composants essentiels sont : le schéma global appelé schéma de médiation, les mappings du schéma global avec les sources, les fonctions de réécriture de requêtes et les fonctions de composition des résultats. Les mappings du schéma global avec les sources sont des requêtes, appelées requêtes de médiation, dont l'expression varie selon l'approche choisie : 1) approche descendante (Global As View ou GAV) où chaque objet du schéma global est défini par une requête sur les sources, 2) approche ascendante (Local As View ou LAV) où chaque objet d'une source de données est défini par une requête sur le schéma global. La figure suivante illustre l'architecture d'un système de médiation :

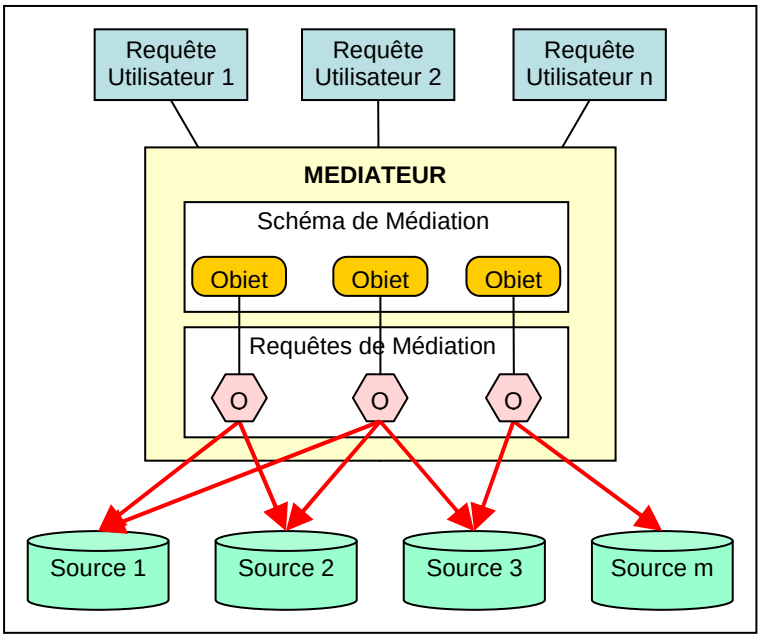


Figure 1 : Architecture d'un système de médiation

1-2- Problématique

Plusieurs problèmes de conception émergent lors de l'utilisation de ces médiateurs. L'une des principales difficultés rencontrée dans un système de médiation est la définition du schéma global et la définition des mappings (requêtes de médiation) qui relient le schéma global aux sources de données.

L'écriture manuelle des requêtes de médiation donne sans doute le résultat le plus pertinent au regard des besoins des utilisateurs. Cependant il est difficile de l'entreprendre en raison du grand nombre de sources de données qui peuvent être impliquées (des centaines ou des milliers) et du volume important de méta-données les décrivant (description des schémas des sources et du schéma global, assertions de correspondance linguistique, assertions intra-source et inter-source, etc). La question principale est de savoir comment automatiser la génération de requêtes de médiation ?

Par ailleurs, comme les données sont hétérogènes, il est important de garantir leur conformité mutuelle et leur conformité avec le schéma global. Deux types de conflits liés à l'hétérogénéité des sources sont distingués: 1) les conflits sémantiques liés au schéma dus à l'utilisation d'une terminologie différente pour décrire un même concept (ex. *Nom* et *Nom_Pers* ; et 2) les conflits sémantiques liés aux données dus à la provenance de données de diverses origines, leur saisie à des moments distincts par des personnes différentes qui n'ont pas la même perception du réel, et qui utilisent des conventions différentes. Par exemple, la différence de monnaie (ex. euros et francs) entre la valeur de l'attribut *prix* appartenant à une relation et la valeur de l'attribut *prix* d'une autre relation. La non prise en compte de ces conflits peut fausser la sémantique des requêtes de médiation et retourner à l'utilisateur des résultats non conformes à ceux définis dans son schéma de médiation. Les opérations qui composent une requête de médiation ne sont valides que si les conflits liés à l'hétérogénéité des données sont détectés et résolus. Par exemple lorsque les valeurs dans une source de données n'ont pas la même précision que celle définie dans le schéma de médiation, ou encore une jointure de deux relations source sur l'attribut prix peut retourner un résultat incorrect quand les prix sont exprimés dans des monnaies différentes. La transformation préalable des monnaies est une solution possible au problème ; encore faut–il savoir détecter les conflits liés à l'hétérogénéité des données et identifier les fonctions de transformations appropriées. La seconde question que l'on se pose est comment détecter et résoudre les conflits liés à l'hétérogénéité des sources de données ?

Par définition un schéma de médiation (respectivement un schéma source) est décrit par sa structure mais aussi par un ensemble de contraintes d'intégrité telles que : les dépendances fonctionnelles, les contraintes de valeurs, etc. Lorsque les contraintes définies sur le schéma global ne sont pas préservées par les requêtes de médiation, la sémantique des requêtes peut être aussi remise en question. Par exemple supposons que la dépendance fonctionnelle $num\_veh \rightarrow marque$ définie sur la relation de médiation $R_m(\underline{num_veh}, marque, puissance)$soit violée par la requête produite lors du processus de génération de requête de médiation, c'est-à-dire la même clé identifie deux objets différents ce qui entraîne un problème d'identification d'objets. La requête produite ne peut donc être considérée comme une requête valide. Après le problème de génération automatique de requête de médiation face au grand nombre de sources de données hétérogènes, le troisième problème est de savoir comment

tester la validité d'une requête par rapport aux contraintes définies sur le schéma de médiation ?

1-3- Objectifs et contributions

Au vu des articles de recherche étudiés, deux catégories de travaux se distinguent. Celle qui vise à interroger les sources de données distribuées et hétérogènes et à la définition de requêtes de médiation, et celle qui vise à intégrer les données et à construire le schéma global. Très peu de travaux se sont intéressées à la génération de requêtes de médiation et à la prise en compte des conflits liés à l'hétérogénéité des sources durant le processus de génération, la plupart se sont focalisés sur la détection et (ou) sur la résolution d'un conflit particulier dans le contexte de l'intégration de données. Nous présentons un état de l'art dans le chapitre suivant qui illustre le manque de travaux dans le contexte de l'interrogation.

Les systèmes de médiation étant de nos jours des systèmes de plus en plus développés et connus, méritent une exploration plus approfondie dans le domaine de la recherche. Le peu de travaux a motivé notre étude en vu d'explorer plus en détails la problématique liée à ces systèmes et d'apporter une solution originale, adaptée aux attentes des utilisateurs.

Face à la problématique de définition de requêtes dans un système de médiation, nous avons opté pour une approche de génération automatique qui permet de décharger l'utilisateur de l'exploration d'un volume important de méta-données.

Nous proposons dans cette thèse une approche de génération automatique de requêtes de médiation dans un contexte GAV (Global As View) qui, à partir du schéma d'une relation de médiation, de dépendances fonctionnelles définies sur cette relation, d'assertions de correspondance linguistique existant entre les sources et le schéma de médiation, et d'assertions intra-source et inter-source reliant les relations sources entre elles, produit un ensemble de requêtes potentiel calculant cette relation. Notre proposition pour la génération de requêtes de médiation se décompose en quatre étapes :

1) Identification des relations sources pertinentes pour la définition d'une requête de médiation du schéma de médiation. Nous introduisons le concept *de relations de mapping* qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation de médiation.

2) Identification des opérations relationnelles possibles entre les relations de mapping, et génération du graphe d'opérations : nous avons développé un algorithme permettant d'identifier les opérations de jointures candidates qui combinent les relations de mapping entre elles.

3) Recherche des chemins de calcul à partir du graphe d'opérations pour calculer la relation de médiation : notre algorithme permet de trouver tous les chemins de jointures possibles.

4) Génération de requêtes de médiation déduites à partir des chemins de calcul de la relation de médiation : notre algorithme génère le texte de la requête SQL permettant son exécution immédiate sur un SGBD quelconque.

L'approche de génération automatique tente de trouver toutes les solutions possibles au calcul de la relation de médiation; ce qui rend le processus automatique très complexe. Mais sous certaines conditions simplificatrices que nous préciserons, notre algorithme permet de générer un ensemble de solutions dans des temps raisonnables, comme le montrent nos évaluations.

La complexité du processus de génération de requêtes est accrue lorsqu'on tient compte de l'hétérogénéité des données. Pour résoudre la problématique liée à la définition de requêtes de médiation dans un environnement hétérogène, notre algorithme de génération de requêtes de médiation tient compte des conflits liés à l'hétérogénéité des sources. Il ajoute automatiquement dans la requête SQL générée la fonction de transformation appropriée.

Notre algorithme permet aussi de détecter pendant l'exploration les requêtes invalides et de ne pas les considérer comme des requêtes candidates au calcul d'une instance du schéma global. L'identification des requêtes invalides est guidée par des règles d'invalidation spécifiées sur un chemin de calcul. Ces règles d'invalidation permettent, pour une relation de médiation donnée de détecter les chemins de calcul qui ne préservent pas les dépendances fonctionnelles définies sur cette relation.

Nos algorithmes s'appuient sur un ensemble de connaissances que nous avons regroupé dans une base de méta-connaissances. Cette base est constituée de :

- La description du schéma de médiation comportant les schéma de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations, et pour chaque attribut d'une relation son type étendu,
- La description du schéma local à chaque source de données comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre les relations d'une même source, et pour chaque attribut d'une relation son type étendu.
- Les assertions de correspondances linguistiques entre les attributs sources et les attributs de médiation (synonymie, abréviations, équivalences terminologiques des noms des attributs).
- Les assertions de correspondances linguistiques entre les attributs de sources différentes.
- Une librairie de fonctions de transformations de données.
- Un dictionnaire linguistique.

Nous avons développé un prototype qui permet de générer automatiquement des requêtes SQL et qui tient compte aussi de l'hétérogénéité des sources de données. Il compte essentiellement six modules : 1) *interface graphique* ; 2) *recherche des relations de mapping* ; 3) *recherche des relations de transitions* ; 4) *recherche des opérations de jointures* ; 5) *recherche des chemins de calcul* ; 6) *génération de requêtes de médiation*.

Notre objectif principal étant de fournir à l'utilisateur un outil adapté aux petits et grands systèmes, nous réalisons une série de tests d'évaluation des performances pour mesurer son passage à l'échelle. Ces tests montrent la faisabilité de notre approche.

1-4- Plan de la thèse

La présentation de cette thèse est organisée de la façon suivante :

Le chapitre 1 situe le contexte de cette étude et expose la problématique abordée et les contributions.

Le chapitre 2 présente un état de l'art sur les travaux de recherche existants dans les contextes de la génération de requêtes de médiation et de l'intégration de données. Il présente aussi un état de l'art sur les outils commerciaux présents sur le marché industriel.

Le chapitre 3 est centré sur notre approche de génération automatique de requêtes de médiation dans un environnement « semi-hétérogéne », où seuls les conflits sémantiques liés au schéma sont considérés. L'objectif de notre algorithme est de produire un ensemble de requêtes potentiel calculant une instance du schéma global, à partir de sources de données distribuées et hétérogènes.

Le chapitre 4 décrit une extension de notre approche de génération de requêtes de médiation dans un environnent hétérogène et qui tient compte des conflits liés à l'hétérogénéité des sources. Bien que l'algorithme décrit dans le chapitre 3 pour la génération de requêtes exploite des correspondances linguistiques prédéfinies pour remédier au problème sémantiques liés au schéma, il ne prend pas en compte les conflits sémantiques liés aux données, il les suppose résolus. Cette nouvelle version de l'algorithme prend en compte ce second type de conflits lors du processus de génération.

Le chapitre 5 illustre l'architecture et le fonctionnement de notre prototype de génération de requêtes de médiation développé durant cette étude. Son objectif est de générer automatiquement des requêtes SQL calculant une instance du schéma global à partir de sources distribuées et hétérogènes.

Le chapitre 6 justifie par des tests la conformité des résultats produits et le passage à l'échelle de notre outil. Nous distinguons deux catégories de tests : 1) des tests sémantiques qui permettent de mesurer la conformité et la cohérence des requêtes produites par rapport aux attentes des utilisateurs; 2) des tests d'évaluation des performances pour mesurer le passage à l'échelle de notre outil.

Le chapitre 7 donne le bilan des résultats de cette thèse, et des perspectives possibles sur de futurs travaux de recherche et de développement.

Chapitre 2 - Etat de l'art

2-1- Introduction

Un système d'informations multi-source est défini comme l'intégration de plusieurs sources de données distribuées et hétérogènes. Parmi ces systèmes d'informations multi-source, nous distinguons les entrepôts de données qui matérialisent totalement ou partiellement les données acquises depuis des sources, les systèmes d'informations basés sur le web qui organisent l'information selon un domaine d'intérêt, les systèmes de bases de données fédérées, ou encore les systèmes de médiation qui permettent d'interroger un ensemble de sources de données distribuées et hétérogènes. Cette intégration s'exprime principalement à l'aide d'un schéma global, appelé schéma de médiation, et d'un ensemble de mappings associant le schéma global aux sources. Parmi ces mappings, on distingue les mappings de schémas qui expriment les correspondances terminologiques et structurelles entre les schémas des sources et le schéma de médiation, et les mappings de données qui expriment la façon dont les instances du schéma global sont calculées à partir des données sources. Ces derniers mappings sont des requêtes, souvent appelées requêtes de médiation, dont l'expression varie selon l'approche choisie : 1) approche descendante (Global As View ou GAV) où chaque objet du schéma global est défini par une requête sur les sources, 2) approche ascendante (Local As View ou LAV) où chaque objet d'une source de données est défini par une requête sur le schéma global.

Plusieurs problèmes de conception émergent lors de l'utilisation de ces systèmes. L'une des principales difficultés rencontrée est la définition de requêtes de médiation calculant une instance du schéma global.

L'écriture manuelle des requêtes de médiation donne sans doute le résultat le plus pertinent au regard des besoins des utilisateurs. Cependant il est difficile de l'envisager pour la définition de requêtes de médiation en raison du grand nombre de sources de données qui peuvent être impliquées (des centaines ou des milliers) et du volume important de méta-données les décrivant (description des schémas des sources et du schéma global, assertions de correspondance linguistique, assertions intra-source et inter-source, etc). La question principale est de savoir comment automatiser la génération de requêtes de médiation ?

Il est aussi difficile d'envisager l'intégration de données en général et la génération de requêtes en particulier sans tenir compte de l'hétérogénéité des sources. Deux types de conflits liés à l'hétérogénéité des sources sont distingués à savoir : 1) les conflits sémantiques liés au schéma : dues à l'utilisation d'une terminologie différente pour décrire le même concept. Par exemple dans la relation de médiation **produit** (Num-produit, désignation, ***prix***) et dans la relation source **produit** (Num-produit, désignation, ***prix-produit***), les attributs prix et prix-produit ont deux terminologies différentes mais une sémantique identique. 2) les conflits sémantiques liés aux données sont présents lorsque les sources contiennent des erreurs et des incohérences (erreurs de saisie, redondances, violation de contraintes d'intégrité, valeurs contradictoires,…). Ces conflits présents dans une source se multiplient lorsqu'il s'agit d'intégrer des données depuis plusieurs sources dans un système d'informations global. En plus des problèmes que peut contenir chaque source indépendamment des autres, chaque donnée peut être représentée différemment dans les sources. La cause principale de cette hétérogénéité entre les

données est la provenance des données de diverses origines, leurs saisies à des moments distincts par plusieurs personnes qui utilisant des conventions différentes. Par exemple la valeur de l'attribut ***prix*** dans la relation de médiation ***produit*** est exprimée en ***euros*** et la valeur de l'attribut ***prix-produit*** dans la relation source ***produit*** est exprimée en ***francs***. La seconde question est de savoir quelles sont les données qui correspondent à la même entité du monde réel ?

Dans le contexte de la génération de requêtes de médiation, la non prise en compte des conflits liés à l'hétérogénéité des sources peut fausser la sémantique des requêtes de médiation et retourner à l'utilisateur des résultats non conformes à ceux définis dans son schéma de médiation. Les opérations qui composent une requête de médiation ne sont valides que si les conflits liés à l'hétérogénéité des sources sont détectés et résolus. Par exemple lorsque les valeurs dans une source de données n'ont pas la même précision que celle définie dans le schéma de médiation, ou encore une jointure de deux relations source sur l'attribut prix peut retourner un résultat incorrect quand les prix sont exprimés dans des monnaies différentes (ex. euros et francs). La transformation préalable des monnaies est une solution possible au problème ; encore faut–il savoir détecter les conflits liés à l'hétérogénéité des données et identifier les fonctions de transformations appropriées.

Au vu des articles de recherche étudiés, la prise en compte des conflits liés à l'hétérogénéité des sources dans le contexte de l'intégration de données en général comprend essentiellement deux phases. Une phase d'analyse de données qui permet de détecter le type de conflit dans une ou plusieurs sources. Une seconde phase de résolution qui permet de résoudre les conflits identifiés.

Très peu d'approches se sont intéressées à la génération de requêtes de médiation et à la prise en compte des conflits liés à l'hétérogénéité des sources durant le processus de génération, la plupart se sont focalisées sur la détection et (ou) sur la résolution d'un conflit particulier dans le contexte de l'intégration de données en général.

Ce chapitre est organisé de la façon suivante : nous développons dans la section 2 les quelques approches qui se focalisent sur la génération automatique des requêtes de médiation dans un environnement hétérogène, dans la section 3 nous présentons les approches qui visent à intégrer des données dans un système global, et qui se focalisent sur la détection des conflits et la transformation des structures et des données hétérogènes pour garantir leur conformité mutuelle et leur conformité par rapport au schéma global, la section 4 présente quelques outils commerciaux disponibles sur le marché industriel dans le contexte de l'intégration de données, et dans la dernière section nous présentons un bilan sur les travaux existants.

2-2- Etat de l'art sur la génération de requêtes de médiation

Comme nous l'avons dit en introduction, la génération de requêtes consiste à générer des requêtes de médiation qui calculent une instance du schéma global à partir d'un ensemble de sources distribuées et hétérogènes. Très peu d'approches se focalisent sur l'interrogation des sources (bases de données réparties, bases de données fédérées,…etc.), et à la génération des requêtes de médiation qui traduisent les mappings entre le schéma global et les sources de données. Cette section présente ces approches :

Dans le cadre d'une collaboration entre le centre de recherche de la société « IBM Almaden » et l'université de «Toronto», un outil appelé CLIO est développé. Il permet de générer automatiquement des requêtes calculant une instance du schéma global à partir de sources de données structurées ou semi structurées. Cet outil a fait l'objet de plusieurs travaux de recherche, parmi ces travaux nous citons dans ce qui suit les trois articles les plus pertinents :

[Miller et al. 00] proposent dans le contexte relationnel une approche de génération de requêtes de médiation qui permet de calculer automatiquement des requêtes SQL. La figure suivante résume leur approche :

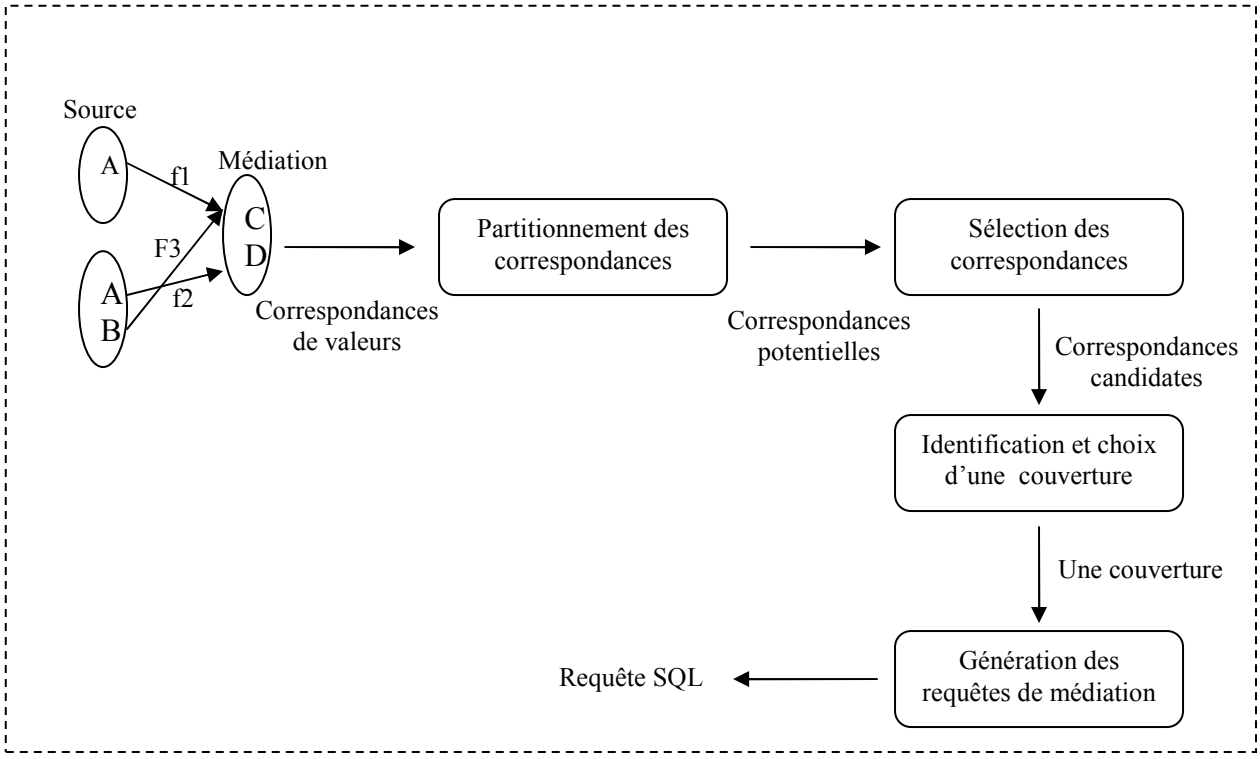


Figure 1 : Approche de génération de requêtes dans le contexte relationnel

Cette approche est basée sur des correspondances de valeurs définies entre les attributs sources et les attributs du schéma global. Par exemple dans la figure suivante l'attribut *caller* de la relation *calls* et l'attribut *artifact* dans la relation du schéma global *references* sont liés par la correspondance de valeur *f1*.

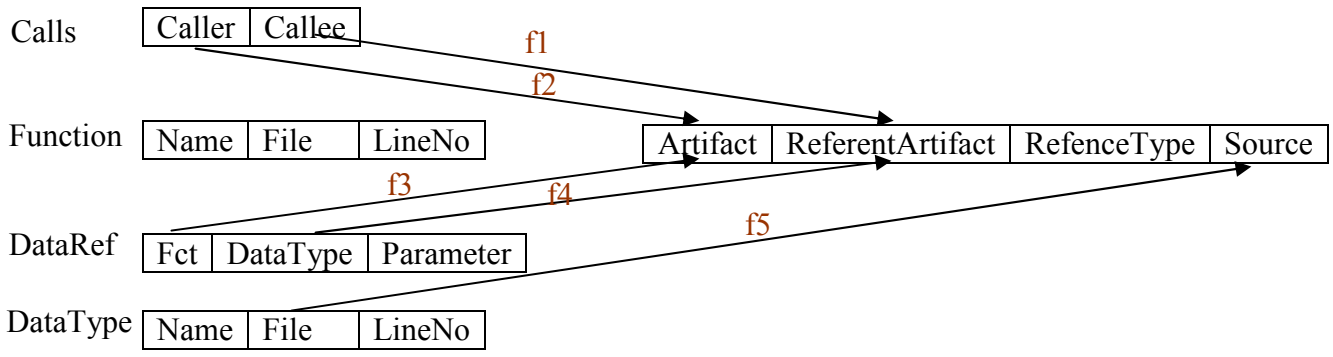


Figure 2 : Utilisation des correspondances de valeurs pour lier deux attributs

Les relations sources et la relation du schéma global représentent les nœuds du graphe, et les liens entre un attribut source et un attribut du schéma global représentent les arcs du graphe. La correspondance de valeur f_i assignée à chaque arc est une correspondance de valeur qui permet de lier deux attributs. Elle est soit déterminée par une technique de matching particulière (d'apprentissage, statistiques, ontologie,…etc), soit définie au préalable manuellement par le concepteur du système. Lorsque les attributs ont des valeurs, ces correspondances sont recherchées par des algorithmes basés sur des techniques de matching permettant de détecter les conflits sémantiques liés aux données. Par exemple si l'attribut A1 a pour valeur V1 et l'attribut A2 a pour valeur V2 une technique de matching est utilisée pour identifier un lien entre la valeur V1 et la valeur V2. Dans le cas où aucune valeur n'est associée à un attribut ces liens sont recherchés en se référant à des algorithmes basés sur une ontologie ou sur des dictionnaires linguistiques qui permettent d'identifier les conflits sémantiques liés au schéma. Ces techniques de détection et de résolution de conflits seront développées plus en détails dans la deuxième partie de notre état de l'art.

L'approche de génération de requêtes de médiation proposée dans cet article compte essentiellement quatre phases :

1- partitionnement des correspondances : cette phase prend en entrée l'ensemble de correspondances de valeurs noté V, il partitionne cet ensemble et il génère un autre ensemble P ={C1,C2, Ci,…,Cn} où chaque ci contient une seule correspondance possible pour un attribut de la relation du schéma global. En d'autres termes si pour le même attribut du schéma global il existe plus d'une correspondance avec les attributs sources, une seule correspondance est prise en

compte. Le résultat de cette phase est un ensemble de correspondances potentiel au calcul d'une relation du schéma global.
2- Sélection des correspondances : cette seconde phase prend en entrée l'ensemble de correspondances potentiel et cherche des combinaisons candidates entre les relations sources pour le calcul de la relation du schéma global. Elle est basée sur un algorithme qui utilise les clés étrangères. Ces clés sont définies au préalable ou recherchés par un algorithme de data mining. Le résultat de cette phase est un ensemble de correspondances candidates noté $C \subseteq P$.
3- Identification et choix d'une couverture : cette troisième phase prend en entrée l'ensemble C de combinaisons candidates et cherche un sous ensemble de C dit *couverture* où toutes les correspondances de valeurs de l'ensemble V apparaissent une seule fois. Si plusieurs couvertures sont identifiées la couverture qui contient le moins de combinaisons candidates est choisie.
4- Génération de requêtes : cette dernière phase prend en entrée la couverture sélectionnée. Pour chaque combinaison candidate une requête SQL est construite. Les requêtes produites par le système CLIO et qui permettent de calculer la relation *references* sont les suivantes :

```
Q1: Select C.Caller, C.Callee, relname (c), F.File
    From Function F,left outer join Calls C
    Where C.Caller = F.Name

Q2: Select C.Caller, C.Callee, relname (c), F.File
    From Function F,left outer join Calls C
    Where C.Callee = F.Name
```

Dans **[Popa et al. 02]** est proposé une généralisation de l'approche de génération de requêtes dans le contexte XML. Elle permet de générer des requêtes mais à partir d'un seul schéma source et dans un format particulier ad-hoc. Etant donné un schéma global et un schéma source ils définissent des requêtes en utilisant des contraintes référentielles. Leur algorithme prend en entrée les correspondances de valeurs et produit en sortie des requêtes dites (mappings logiques).

Dans **[Popa et Yu 04]** les auteurs proposent deux algorithmes de réécriture de requêtes basés sur des contraintes définies sur le schéma global. Etant donné des schémas sources et le schéma global ils identifient tout d'abord les mapping entre les attributs du schéma global et les attributs sources, et ensuite ils génèrent un ensemble de requêtes calculant une instance du schéma global. Ils défissent une instance canonique sur chaque requête produite à partir des instances des schémas sources et du schéma global. Sur cette instance ils définissent des contraintes et réécrivent la requête.

[Chen et al. 03] se focalisent sur la construction du schéma global et sur l'interrogation des sources de données semi-structurées. Ils proposent une approche basée sur un méta-modèle appelé ORA-SS qui compte trois composants essentiels : les classes d'objets, les attributs de chaque objet, et les relations entre les objets.

Ils prennent en entrée le méta-modéle qui représente l'ensemble des schémas sources et l'ensemble des liens définis au préalable entre les attributs (par exemple les clés étrangères). Ils combinent les schémas sources à l'aide d'opérateurs pour construire une instance du schéma global. Quatre opérateurs sont proposés à savoir : *Join, Drop, Swap,* et *Select. Join* réalise la jointure de deux objets, *Drop* permet de supprimer les objets et leurs attributs, *Swap* change de position aux objets dans le graphe, et Select permet de sélectionner des objets dans le graphe en

imposant des contraintes. Ils génèrent en sortie un autre méta-modéle qui est une combinaison des schémas sources et qui représente une instance du schéma global.

Exemple

La figure 3 illustre le méta-modéle ORA-SS représentant deux fichiers XML S1 et S2. Les classes de chaque fichier sont représentées par des rectangles. Les attributs de chaque objet sont représentés par des cercles. Les attributs clés sont décrits par des cercles noirs. Les relations entre les classes d'objets sont représentées par le nom de la relation entre deux classes, le nombre d'objets voisins d'un objet, et les cardinalités entre les objets, par exemple (SP, 2, 1:n,1:n). La figure montre aussi la relation entre les deux fichiers S1 et S2 qui est une clé étrangère définie entre la classe *project* et la clases *project*', elle est représentée par une flèche en pointillés.

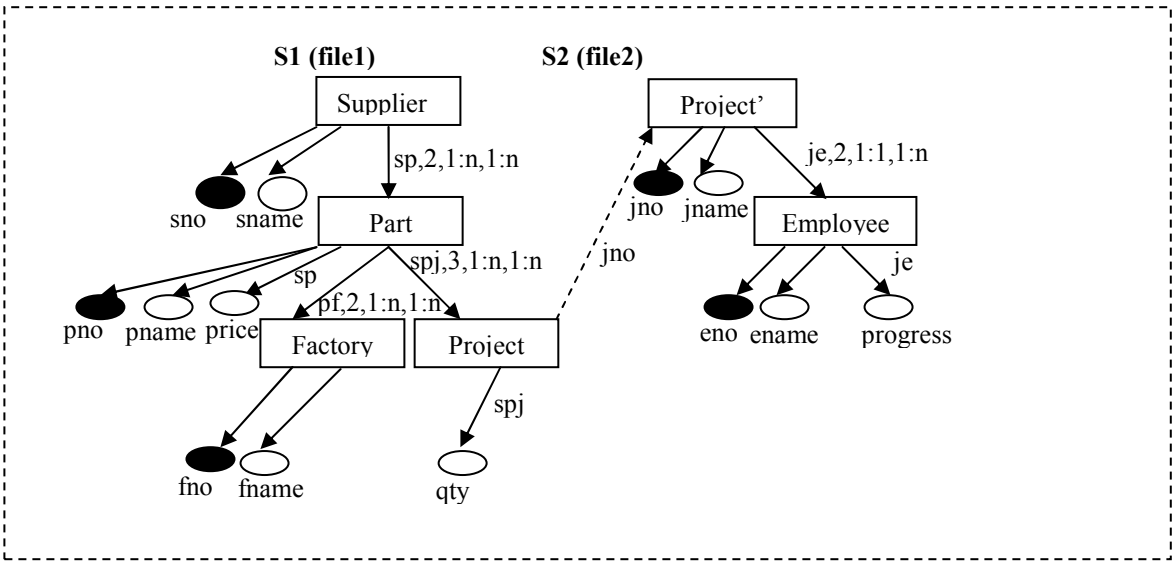


Figure 3 : représentation des schémas sources dans le modèle ORA-SS

La figure 4 illustre le méta-modéle ORA-SS qui cette fois décrit une instance du schéma global obtenue par la combinaison des deux fichiers S1 et S2, à l'aide des quatre opérateurs décrits précédemment.

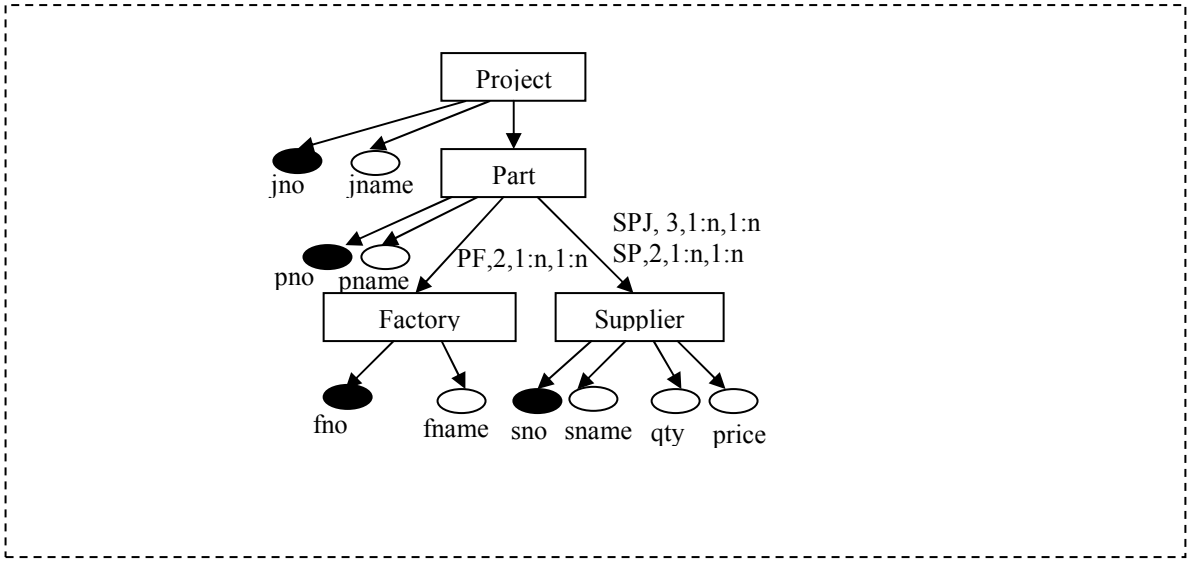


Figure 4 : Représentation d'une instance du schéma global

Une fois le schéma global défini, ils appliquent un algorithme qui permet de générer des requêtes XQuery qui calculent une instance du schéma global.

[Zamboulis 04] s'intéresse à la transformation et la restructuration du schéma de documents XML dans le système Automed. Ce système est décrit comme l'intégration de sources données hétérogènes qui supporte la transformation de schéma. Chaque source de données est décrite par son schéma noté Si sur lequel est appliqué une série de transformations primitives telle que : insert, rename, remove. Un schéma intermédiaire Ri associé à chaque Si est obtenu. L'union des schémas est aussi généré automatiquement notée USi. Automed supporte les deux approches *top-dow* et *bottom-up*. Pour l'approche *top-dowm* où le schéma global est défini, le système restructure les schémas des sources jusqu'à ce qu'ils matchent avec le schéma global. Dans l'approche *bottom-up* où le schéma global n'est pas défini, le système permet de le construire automatiquement. Pour cette restructuration, il propose un algorithme dont le principe est basé sur un graphe de restructuration. Il suppose que les correspondances entre les attributs sources et les attributs du schéma global sont détectés par des techniques de data matching évoquées dans [Rahm et Bernstein 01]. Son algorithme compte deux phases :1) *phase d'insertion* où chaque élément du graphe absent dans une source est inséré dans cette dernière; 2) *phase de suppression* où chaque élément d'une source absent dans le graphe est supprimé de la source.

Au vu des articles de recherche étudiés, nous avons présenté dans cette section un état de l'art sur les techniques utilisées par les approches qui visent à interroger des sources distribuées et hétérogènes, et à générer automatiquement des requêtes de médiation. Certaines approches se focalisent sur l'identification des correspondances entre les attributs, d'autres supposent que ces correspondances sont déjà définies ou appellent des techniques existantes pour les déterminer, et se focalisent sur la génération des requêtes et aussi sur la sémantique de ces dernières pour garantir un retour correct à l'utilisateur.

Parmi ces travaux de recherche, nous distinguons les travaux réalisés dans le cadre d'une collaboration entre le centre de recherche de la société IBM Almaden et l'université de Toronto pour le développement de l'outil CLIO. Ils proposent pour le contexte relationnel une approche de génération de requêtes de médiation dont le principe est similaire à celui de notre approche développée dans le chapitre suivant. Le tableau suivant résume les similarités et les différences entre ces deux approches :

	CLIO	**Notre approche**
Contexte	Système mono-source -un schéma global -un schéma source	Système de médiation -un schéma global -plusieurs schémas sources
Problématique adressée	- Définition de requêtes de médiation	- Définition des requêtes de médiation - Prise en compte de l'hétérogénéité des sources - Prise en compte des besoins des utilisateurs en termes de contraintes d'intégrité
Approche	- Partitionnement des correspondances - Sélection des correspondances - Identification et choix d'une couverture - Génération des requêtes de médiation	- Recherche des relations de mapping - Recherche des opérations candidates - Recherche des chemins de calcul - Génération des requêtes de médiation
Points forts	- Génération automatique des requêtes - Prise en compte des correspondances 1-n entre le schéma global et le schéma source - Prise en compte des conflits sémantiques liés au schéma - Génération des requêtes imbriquées	- Génération automatique des requêtes - Implication de plusieurs schémas sources - Prise en compte des conflits sémantiques liés aux données et au schéma - Détection et résolution automatique des correspondances - Prise en compte des besoins des utilisateurs en termes de contraintes d'intégrité - Passage à l'échelle
Points faibles	- Détection manuelle des correspondances linguistique - La non prise en compte des conflits sémantiques liés aux données	- La non prise en compte des correspondances 1-n, seules les correspondances 1-1 sont considérées - La non prise en compte des requêtes imbriquées

Comme nous l'avons déjà dit en introduction, il est difficile d'envisager la génération de requêtes de médiation sans tenir compte des conflits liés à l'hétérogénéité des sources. De nombreux travaux de recherche ont adressé la problématique liée à l'hétérogénéité des sources dans le contexte de l'intégration de données en général. La plupart des approches existantes sur la génération de requêtes de médiation référencent ces travaux pour prendre en compte les conflits

durant le processus d'interrogation des sources distribuées et hétérogènes. Dans la section suivante nous présentons un état de l'art sur les techniques utilisées pour la détection et la résolution des conflits liés à l'hétérogénéité des sources dans le contexte de l'intégration de données.

2-3- Etat de l'art sur la prise en compte des conflits liés à l'hétérogénéité des sources

Garantir la conformité mutuelle des données hétérogènes et leur conformité par rapport au schéma global est cruciale dans un système multi-source en général. Il est difficile d'envisager l'intégration de données sans tenir compte des conflits liés à l'hétérogénéité des sources. Les conflits sont présents lorsque les sources contiennent des erreurs et des incohérences (erreurs de saisie, redondances, violation de contraintes d'intégrité, valeurs contradictoires,…). Les problèmes se multiplient lorsqu'il s'agit d'intégrer des données depuis plusieurs sources dans un système d'informations global basé sur le web, un système de bases de données fédérées ou encore dans un entrepôt de données. En effet, en plus des données erronées que peut contenir chaque source individuellement, le problème de correspondance entre objets doit être pris en compte. En d'autres termes lorsque des données proviennent d'origines diverses, et saisies à des moments distincts par plusieurs personnes qui utilisent des conventions différentes, la question capitale est de savoir quelles sont les données qui correspondent à la même entité du monde réel ? Par exemple une société peut avoir des informations sur ses clients stockées dans des tables différentes, car chaque client peut acheter des produits dans plusieurs départements. Une fois la décision de centraliser l'information sur tous les clients dans une même source (ex : dans un entrepôt de données), un même client peut être représenté différemment dans les sources : « John Smith », « Smith John », « J.Smith »

La constitution d'un système multi-source est une réponse au problème de l'intégration d'une grande quantité de données variées, relatives à un certain domaine d'application, et stockées physiquement dans différentes sources de données. Un système multi-source est régulièrement alimenté et mis à jour via des extractions depuis des sources et la saisie de données quotidiennement, ce qui augmente la probabilité de présence de données erronées.

Un système global étant une solution pour faire face aux besoins des entreprises en termes de capitalisation de connaissances et d'aide à la décision, il est primordial que les données stockées soient correctes et fiables. Ces systèmes ont alors besoin d'un support de nettoyage de données très avancé qui permettra de détecter et de résoudre les conflits présents dans les sources et de mettre en conformité les données sources avant de les intégrer et de les délivrer à l'utilisateur.

Durant cette dernière décennie, la plupart des approches existantes sur l'intégration de données se sont focalisées sur les conflits liés à l'hétérogénéité des données appelés *conflits sémantiques liés aux données.* Dans la littérature existante, le terme « *nettoyage de données* » est utilisé pour désigner la prise en compte de ces conflits.

Les premières approches traditionnelles se sont focalisées sur les conflits d'intégration de schéma appelés *conflits sémantiques liés au schéma* qui découlent de l'utilisation d'une terminologie différente pour décrire le même concept. Par exemple dans la relation de médiation **produit** (Num-produit, désignation, ***prix***) et dans la relation source **produit** (Num-produit, désignation, ***prix-produit***), les attributs prix et prix-produit ont deux terminologies différentes mais une sémantique identique.

Dans le contexte de la génération de requêtes, il est important de prendre en compte les conflits sémantiques liés aux données ainsi que les conflits sémantiques liés au schéma afin de retourner à l'utilisateur un résultat sémantiquement correcte et conforme à celui défini dans son schéma global. Ce point sera développé plus en détails dans les chapitres suivants.

Suite à cette introduction, nous présentons tout d'abord les conflits sémantiques liés aux données que l'on peut rencontrer lors du nettoyage de données. Nous présentons ensuite la méthodologie générale de nettoyage de données adoptée par les approches existantes, et enfin les techniques proposées pour la détection et la résolution des conflits. Les conflits sémantiques liés au schéma ne sont pas développés dans cette thèse.

2-3-1- Les conflits sémantiques liés aux données

Nous présentons dans ce qui suit une classification possible des conflits sémantiques liés aux données dans les sources proposée dans [Rahm et Hau Do 99], différenciant les problèmes structurels et sémantiques concernant une ou plusieurs sources de données. Les conflits sémantiques liés aux données sont à deux niveaux :

A) Problèmes mono-source : une source de données est confrontée en général à deux types de problèmes : les problèmes d'ordre structurels, et les problèmes d'ordre sémantique.

Problèmes structurels : Les problèmes structurels dans une source sont relatifs à la structure du schéma de la base, aux contraintes d'intégrité imposées et qui contrôlent les valeurs de données permises. Ces problèmes influent de façon systématique sur la sémantique des données. Les exemples suivants illustrent les problèmes structurels présents au niveau d'une source :

Exemples

- Violation d'une contrainte dans une plage de valeurs : date = 30/13/70 (le mois 13 n'est pas inclus dans l'intervalle [01- -12]).

- Violation d'une contrainte de dépendance entre attributs : âge =22 date de naissance = 12/02/70.

Problèmes sémantiques : Les problèmes sémantiques dans une source sont plutôt relatifs aux données invalides (erreurs d'orthographe, erreurs de frappe, etc). Ces problèmes n'influent pas sur la structure de la base car elles respectent les valeurs permises mais elles ne sont pas correctes. Les exemples suivants illustrent les problèmes sémantiques présents au niveau d'une source.

Exemples :

- Erreur d'orthographe : Ville = Pariis

- Violation d'une relation sémantique entre attributs : Ville = Versailles Code Postal = 79000 (le code postal existe mais ne correspond pas à la ville de Versailles).

B) Problèmes multi-souce : Les problèmes présents dans une source s'aggravent lorsqu'il s'agit d'intégrer des données depuis plusieurs bases dans un entrepôt de données. En plus des données erronées que peut contenir chaque source indépendamment des autres, chaque donnée peut être représentée différemment dans les sources. La cause principale de cette hétérogénéité entre les

données est la provenance des données de diverses origines, leur saisie à des moments distincts par des personnes différentes qui n'ont pas la même perception du réel et qui utilisent des conventions différentes. Les incompatibilités suivantes surviennent :

Problèmes structurels : L'incompatibilité structurelle est due à des définitions différentes des attributs. En général, on distingue les conflits suivants :

Type : Le même attribut peut avoir une définition de type incompatible dans les sources. Le numéro de sécurité sociale peut être de type numérique dans une source, et être de type caractère dans une autre.

Format : Un même élément de donnée peut avoir des formats distincts dans les sources. Dans une source, la date peut être de la forme suivante : jour/mois/année et dans une autre sous cette forme mois/jour/année

Unité : Un même élément de donnée peut être exprimé en plusieurs unités. Une quantité est exprimée en grammes dans une source et en kilogrammes dans une autre.

Granularité : Un même élément de donnée peut avoir des niveaux de granularité différents. Le nombre de naissances par mois ou les naissances par année.

Problèmes sémantiques : L'incompatibilité sémantique est due aux valeurs d'attributs différentes que peut avoir une même instance d'entité dans deux sources différentes. En général, on distingue les conflits sémantiques suivants :

Synonymes : Une même instance d'entité peut avoir des valeurs d'attributs différentes. L'attribut fonction concernant une même personne peut avoir la valeur « médecin » dans une source et « docteur » dans l'autre.

Homonymes : Des instances d'entités différentes peuvent avoir la même valeur d'attribut. Le nom John Smith peut designer plusieurs personnes.

Codes : Une même instance d'attribut peut avoir des codes différents. Le sexe d'une personne peut être codé « M » et « F » ou « 1 » et « 2 ».

Différence de précision sémantique: Une même instance d'entité peut avoir des niveaux de dénominations différents. "Vermillon" dans une source et "Rouge" dans une autre.

Différence de précision de valeurs numériques: Une même instance d'entité peut avoir des valeurs numériques imprécises. "4.8 kg" dans une source et "5.0 kg" dans une autre.

Valeurs contradictoires : Une même instance d'entité peut être modifiée à des moments différents. L'âge ou le poids d'une personne (problème de fraîcheur).

Erreurs de saisie : Une même instance d'entité peut avoir des variations orthographiques. Le nom Smith dans une source, et Schmidt ou Smythe dans une autre (Erreurs d'orthographe, erreurs phonétiques, etc.).

De ces problèmes mono-source et multi-source découle *le problème de correspondance entre objets* où il s'agit d'identifier les enregistrements qui correspondent à la même entité du monde réel. Par exemple une société peut avoir des informations sur ses clients stockées dans des tables différentes, car chaque client peut acheter des produits dans plusieurs départements. Une fois la décision de centraliser l'information sur tous les clients dans une même source (ex : dans un entrepôt de données), un même client peut être représenté différemment dans les sources : « John Smith », « Smith John », « J.Smith ». Ce problème est connu sous le nom d'identification d'objets dans [Galhardas et al.00] et sous le nom d'élimination de doublons/Fusion dans [Hernandez, J.Stolfo95].

Après avoir présenté les différents conflits sémantiques liés aux données que l'on peut rencontrer lors de l'intégration de données, la section suivante illustre la méthodologie générale de prise en compte des conflits adoptée par les approches existantes dite méthodologie de nettoyage de données.

2-3-2- La méthodologie de nettoyage de données

Au vu des articles de recherche étudiés, la prise en compte des conflits liés à l'hétérogénéité des sources comprend essentiellement deux phases. Une phase d'analyse de données qui permet de détecter la nature du conflit dans une ou plusieurs sources. Une seconde phase de résolution qui permet de résoudre les conflits identifiés et de transformer les données pour les mettre en conformité. Très peu d'approches se sont intéressées à la détection et à la résolution d'un ensemble de conflits, la plupart se sont focalisées sur la détection et (ou) sur la résolution d'un conflit particulier. Le tableau suivant illustre ces travaux de recherches.

Phases / Approches	*Analyse de données*	*Résolution*
[Chatterjee et Segev 91]		**X**
[Siegel et Madnick 91]	**X**	
[Sciore et al. 94]	**X**	
[Li et Clifton 94]	**X**	
[Hernandez et Stolfo 95]		**X**
[Kedad et Métais 99]		**X**
[Rahm et hai Do 99]	**X**	
[Sapia et al. 99]	**X**	
[Calvanese et al.99]	**X**	**X**
[Tejada et al. 00]	**X**	**X**
[Fan et al. 00]	**X**	**X**
[Galhardas et al. 00]	**X**	**X**
[Rahm et Bernstein 01]	**X**	
[Hellerstein et al. 01]	**X**	**X**
[Karayannidis et al.01]	**X**	**X**
[Moulton et al. 02]	**X**	**X**
[Claypool et Rundensteiner 03]		**X**
[Zamboulis 04]		**X**

Tableau récapitulatif des approches d'analyse et de résolution existantes.

Analyse des données

Une analyse de données approfondie est très utile pour identifier la nature du conflit présent dans les sources de données à savoir : **(1)** les problèmes mono-source au niveau structurel et sémantique, **(2)** les problèmes multi-souces au niveau structurel et sémantique, et **(3)** le problème de correspondance entre objets qui découle des problèmes mono-source et multi-source.

Hormis l'analyse manuelle des sources, différentes techniques de détection automatique de conflits sont utilisées dans la recherche notamment :

1. **Le data profiling :** Elle analyse les instances de chaque attribut individuellement dans une source, et détecte un conflit lorsqu'une instance ne correspond pas au profil prédéfini [Rahm et hai Do 99].
2. **Le data mining :** Elle analyse un ensemble d'attributs, et identifie les relations entre ces attributs et leurs valeurs [Sapia et al. 99]
3. **Les techniques statistiques :** Elles détectent des conflits entre les valeurs d'attributs numériques. [Fan et al. 00] utilisent la corrélation pour la détection.
4. **Les techniques d'apprentissage :** Elles détectent les similarités entre les attributs. [Li et Clifton 94] utilisent les réseaux de neurones pour reconnaître les attributs similaires.
5. **La dépendance entre attributs** : Les attributs dépendants tels que : l'âge et la date de naissance sont utilisés pour détecter les conflits entre les instances d'attributs [Rahm et hai Do 99].
6. **Les dictionnaires électroniques et les correcteurs :** Ils sont utilisés pour faciliter la détection automatique des erreurs. L'utilisation des dictionnaires linguistiques tels que WORDNET et EDR permettent de détecter les problèmes de synonymie, d'homonymie, de différents niveaux de dénominations, etc.
7. **Le domaine sémantique d'un attribut :** il est utilisé dans [Siegel et Madnick 91] et [Sciore et al. 94] pour détecter par exemple un conflit lié à la différence d'unités (Francs, Pesetas) entre deux valeurs d'attributs.

Résolution des conflits

Afin de garantir la conformité de données hétérogènes dans les sources et d'entreprendre une intégration dans un entrepôt de données, une phase de résolution des conflits détectés est nécessaire.

En général les techniques utilisées dans la recherche pour la résolution des conflits sont les suivantes :

1. **Les fonctions de transformations :** Elles sont utilisées pour transformer la valeur d'un attribut en une autre. Ces fonctions peuvent définies dans une librairie (ex la librairie de fonctions disponible dans Oracle), comme elles peuvent être des procédures génériques.

2. **Les règles de conversion :** Elles sont utilisées pour résoudre les incompatibilités de formats, d'unités…etc. [Fan et al.00] utilisent la régression pour les générer.
3. **Les correcteurs et les dictionnaires :** Ils sont utilisés pour faciliter la coorection automatique des erreurs.
4. **Les ontologies :** Elles sont utilisées dans [Zweigenbaum 98] pour éviter l'emploi des polysémes, supprimer les synonymes, et pour résoudre le problème de différence de niveau de dénomination. Elles sont utilisées aussi dans [Moulton et al. 02].
5. **Les opérateurs sémantiques :** Ils sont utilisées dans [Kedad et Métais 99] pour la résolution des problèmes sémantiques entre plusieurs sources à savoir le problème de différence de précision sémantique.
6. **Les règles de probabilité :** elles sont utilisées dans [Chatterjee et Segev 91] pour résoudre le problème de présence ou d'absence de clés communes entre les sources.
7. **Les règles heuristiques :** Elles sont utilisées dans [Wang et Madnick 89] pour résoudre le problème de présence ou d'absence de clés communes entre les sources. Elles permettent de déduire des informations supplémentaires sur les instances d'entités en utilisant une approche basée sur les connaissances.
8. **Les opérateurs logiques :** ils sont utilisés dans [Calvanese et al.99], [Galhardas et al.00] et [Claypool et Rundensteiner 03] pour mettre en conformité les données sources par rapport au schéma global.
9. **Le graphe de restructuration :** ils est utilisé dans [Zamboulis 04] pour transformer les données.

La plupart des techniques utilisées pour la détection et la résolution des conflits sémantiques liés aux données sont utilisées par les approches d'intégration de schéma pour prendre en compte les conflits sémantiques liés au schéma. Par exemple un dictionnaire linguistique peut être à la fois utilisé pour détecter un conflit sémantique lié au schéma entre l'attribut *prix* d'une relation *produit* et l'attribut *prix _produit* d'une autre relation, et il peut être aussi utilisé pour détecter un conflit sémantique lié au données entre la valeur *docteur* de l'attribut *fonction* appartenant à une relation *R* et la valeur *médecin* de l'attribut *fonction* d'une autre relation.

Ces techniques de détection et de résolution de conflits appelées *techniques de nettoyage de données* sont développées dans la section suivante.

2-3-3- Les techniques de nettoyage de données

Etant donné la multiplicité des techniques utilisées pour détecter et résoudre les conflits liés à l'hétérogénéité des sources, nous développons dans cette partie les techniques les plus référencées dans les travaux de recherche sur l'intégration de données en général et particulièrement dans les travaux de recherche sur l'interrogation des sources hétérogènes et distribuées par un système de médiation.

1- Les techniques d'apprentissage

Ces techniques utilisent l'apprentissage pour reconnaître les attributs similaires entre des sources différentes. Par exemple l'utilisation des réseaux de neurones dans [Li et Clifton 94] pour apprendre à détecter les similarités entre les attributs.

[Li et Clifton 94] s'intéressent au problème de correspondance entre objets évoqué précédemment, leur objectif est de détecter les attributs similaires qui correspondent à la même entité du monde réel. La figure suivante illustre leur approche :

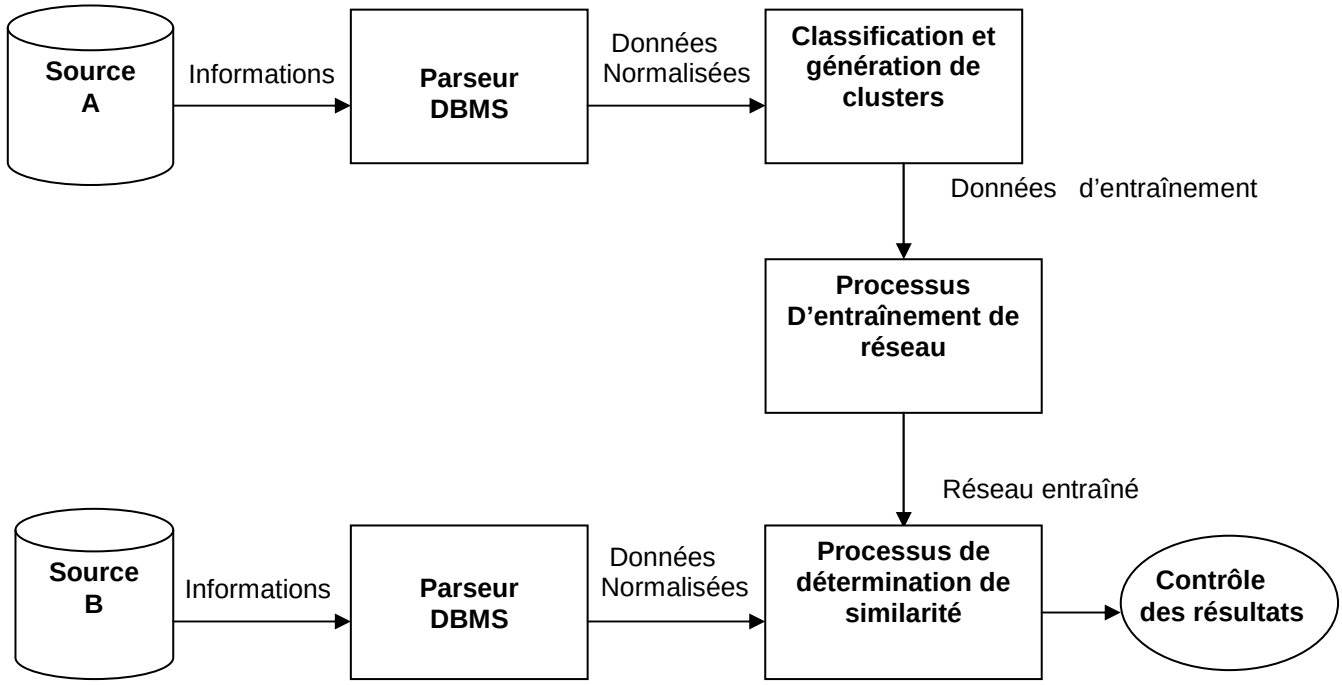


Figure 5 : Procédure d'apprentissage

Le principe de leur approche est basé sur l'hypothèse suivante : ils supposent que deux attributs correspondent à la même entité lorsqu'ils possèdent des similarités au niveau structurel et au niveau de leurs valeurs. Leur approche est constituée des étapes suivantes :

- **Extraction d'informations :** Dans un premier temps, ils utilisent des parseurs pour extraire depuis les sources les informations concernant la structure et les valeurs d'attributs. A la suite de cette opération le système transforme ces informations extraites sous un même format.
- **Classification et génération de clusters :** Un classificateur ou un réseau auto-organisateur est ensuite utilisé pour apprendre à distinguer les attributs dans une seule source (Source A) en se basant sur des discriminateurs ou des méta-données. Ces méta-données (nœuds du réseau) sont les informations collectées par les parseurs et qui décrivent les attributs telles que : la longueur, le type ou d'autres informations sur le format, la présence de contraintes (clés primaires, clés étrangères, clés candidates, valeurs nulles refusées,…).

Dans le cas ou deux attributs sont de même type et ont la même longueur mais des valeurs différentes (ex le nom et le prénom d'une personne) le classificateur se réfère à d'autres discriminateurs.

Pour les attributs de type caractères les discriminateurs suivants sont utilisés :

- Le ratio du nombre de caractères numériques sur le nombres total de caractères. Par exemple pour num-etudiant de type caractère (999-99-9999) le ratio est de 9/11. Alors que le ratio du nom ou du prénom est égal à zéro.
- Le ratio du nombre de caractères blancs sur le nombre total de caractères. Par exemple le nom et le prénom d'une personne contiennent très peu de caractères blancs contrairement à l'adresse.

Pour les attributs de type numériques ils utilisent comme discriminateurs des méthodes statistiques à savoir :

- La moyenne : qui mesure la valeur moyenne de l'ensemble des valeurs d'un attribut.
- La variance : qui mesure la variabilité d'un ensemble de valeurs d'un attribut particulier.
- Le coefficient de variation : qui mesure la déviation ou la variabilité d'un ensemble de valeurs d'un attribut particulier par rapport à la valeur moyenne.

Le résultat de ce classificateur est un ensemble de clusters qui regroupent les attributs de mêmes caractéristiques. En sachant que le nombre de clusters résultants peut être égal au [nombre d'attributs – les clés étrangères] dans une source.
A chaque cluster ou catégorie est assigné un poids diffèrent qui va permettre de les différencier. Ces données résultantes de la classification sont appelées des données d'entraînement.

- **Processus d'entraînement de réseau :** Un processus d'entraînement de réseau utilise en entrée les données d'entraînement pour s'entraîner à reconnaître les clusters générés par la classification. Durant cette étape d'autres poids sont assignés par le réseau aux différents clusters. Le résultat est un réseau entraîné, qui a pour entrée des informations sur un attribut provenant d'une autre source (source B). Ce réseau calcule les degrés de similitude de l'attribut de la source B avec chaque cluster de la source A, et génère des listes d'attributs similaires.
- **Contrôle des résultats :** Durant cette dernière étape l'utilisateur contrôle les résultats produits par le réseau en se basant sur des listes d'attributs similaires existantes.

2- Les techniques statistiques : Ces techniques utilisent l'analyse statistique pour détecter et résoudre les conflits structurels et sémantiques entre les attributs numériques dans plusieurs sources.

[Fan et al. 00] proposent une approche destinée à intégrer des données commerciales et financières. Elle permet de détecter les conflits entre les valeurs d'attributs numériques en utilisant la corrélation.

Exemple : Soient les deux tables suivantes Stock1 et Stock2 qui contiennent des informations concernant les stocks d'une même entreprise :

Stock	Monnaie	Volume	Prix-max	Prix-min	Prix-cloture
100	1	438	100.50	91.60	93.11
101	3	87	92.74	78.21	91.35
102	4	338	6.22	5.22	5.48
104	1	71	99.94	97.67	99.04
111	0	311	85.99	70.22	77.47
115	2	489	47.02	41.25	41.28
120	3	370	23.89	21.09	22.14

Stock	Prix	Volume	Valeur
100	65.18	438000	28548840.00
101	164.43	87000	14305410.00
102	31.78	338000	10741640.00
104	69.33	71000	4922430.00
111	30.99	311000	9637890.00
115	41.28	489000	20185920.00
120	39.85	370000	14744500.00

Deux types de problèmes sont pris en compte par l'approche. Le premier problème est la différence des valeurs de l'attribut « volume » dans les tables Stock1 et Stock2 (problème structurel entre deux sources). Le second problème est au niveau de la structure du schéma des deux tables qui est du à la présence d'attributs différents mais sémantiquement liés (problème de correspondance entre attributs).

[Fan et al. 00] utilisent la corrélation pour détecter d'une part le premier conflit entre les valeurs de l'attribut volume, et d'autre part les ensembles d'attributs sémantiquement liés entre les deux tables. Le calcul de la corrélation entre les valeurs d'attributs est effectué en utilisant la méthode PCA (Partial Corrélation Analysis), basée sur la formule suivante :

$$r_{x,y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sqrt{\sum (y_i - \bar{y})^2}}}$$

Un conflit entre deux attributs x et y est détecté dans le cas où la valeur de $r = \pm 1$, si la valeur de $r = 0$ alors absence de conflits.

3- Les règles de conversion : Les règles de conversion ou de transformation sont des règles qui convertissent les données sous un format compatible. (Ex : convertir les grammes en kilogrammes, les Francs Français en Dollars, les abréviations, …etc). Elles permettent de

résoudre les conflits structurels entre plusieurs sources. Les approches existantes se basent sur un ensemble d'outils pour effectuer ces conversions tels que : les dictionnaires de synonymes pour convertir les abréviations (MC, Maître de Conférences), des règles mathématiques (les litres en mètres cube), etc.

[Fan et Lu 00] utilisent une méthode de régression efficace LTS (Least Trimmed squares) basée sur la méthode des moindres carrés (LS), qui permet de détecter les « outliers » de les supprimer et ainsi de générer des fonctions de régression. Ces fonctions seront ensuite évaluées grâce à un support qui est un critère d'acceptation. Une fonction sera retenue, dans le cas de figure, ou son support est supérieur à un seuil prédéfini.

A partir de l'ensemble de fonctions de conversion retenu, et après un certain nombre de transformations syntaxique, un ensemble de règles de conversion est généré.

4- Les techniques de data profiling : La technique data profiling associe à chaque attribut un profil en utilisant les méta-données suivantes: le type, la longueur, des valeurs min et max, une cardinalité, et une unicité.

Evoke Software (http://www.evokesoft.com/ .) est un outil commercial sur le data profiling qui utilise ces méta-données pour analyser les instances de chaque attribut et détecte les problèmes structurels relatifs à une source (violation de contrainte d'intégrité) lorsqu'une instance ne correspond pas au profil prédéfini

Exemple

Méta-donnée min et max : 1<age<100

Une violation de la contrainte d'intégrité est détectée dans une source si l'âge d'une personne n'est pas compris entre 1 et 100.

5- Les techniques de data mining : Ces techniques sont utilisées pour analyser les sources et faciliter l'identification des relations entre les attributs dans une ou plusieurs sources. L'objectif de cette analyse est d'une part la détection de la sémantique dans les sources et leur compréhension, et d'autre part la détection et la résolution des conflits structurels et sémantiques mono-source ou multi-source. Un exemple de conflit résolu par les algorithmes de data mining : soient les trois attributs total, quantité et prix unitaire, les techniques de data mining permettent de générer la relation mathématique [total = quantité * prix unitaire] qui associe ces trois attributs.

[Sapia et al. 99] utilisent les techniques de data mining, les étapes suivantes illustrent leur approche :

A - Extraction de la sémantique dans les sources

a) La détection de la sémantique des attributs : Par exemple lorsque le sens exact d'un attribut ne peut pas être déduit directement de son nom ou de son type (ex : les attributs CTY et ST des tables A et B qui contiennent des informations sur la réparation de voitures dans le monde) :

VID	VTP	ST	CTY	POW	WGT	MIL	DAT
102009433	C240	1	D	170	1345	2319	99/02
102009433	C240	1	D	170	1301	25	99/03
102009565	C180	1	D	122	1205	19	99/04
102340339	E430	1	D	229	1456	24	99 :05
103035689	E290	1	D	129	1459	1578	99/06
…	…	…	…	…	…	…	…

VID	VTP	ST	CTY	POW	WGT	MIL	DAT
102340337	E430	2	GB	170	1345	5065	99/03
103012002	C250	2	GB	170	1301	5	99/06
103013212	C250	2	GB	122	1205	3	99/07
…	…	…	…	…	…	…	…

Dans ce cas, les techniques de data mining se réfèrent à des connaissances pour trouver des les règles d'associations entre les attributs qui vont apporter un plus d'informations sur leur sémantique. Exemple : 90% des voitures réparées en GB (Grande Bretagne) sont de type 2 (conduite à droite) et 20% de toutes les voitures réparées sont de GB et de type 2, où le 90% représente la confiance et le 20% le support de la règle. Cette information est utilisée en entrée par l'algorithme de data mining qui va générer comme résultat toutes les règles d'associations de la forme $X \rightarrow Y$ et qui ont un support supérieur au support minimum donné et une confiance supérieure au minimum de confiance donné. Par application de l'exemple, la règle d'association générée est la suivante : (GTY = GB) $\rightarrow$ (ST = 2). L'identification de la relation entre les attributs « CTY » et « ST » permet de comprendre le sens des attributs.

b) L'identification du plan de codification : Une fois que la sémantique des attributs connue, les techniques de data mining peuvent aussi identifier le plan de codification associé aux valeurs d'attributs si celui ci n'est pas spécifié dans la structure du schéma de la base. Elles utilisent les dépendances inter-attributs pour poser des hypothèses sur le plan de codification. Par exemple le fait que l'attribut ST (type de conduite) prend toujours la valeur 2 lorsque le pays est égal à GB (Grande Bretagne) elles déduisent que le code 2 correspond à la conduite à droite.

c) La détection des contraintes d'intégrité : Elle facilite également la compréhension des sources. Des méthodes de visualisation sont utilisées pour analyser les distributions de valeurs et déterminer les domaines d'attributs. Par l'exemple pour la réparation de voitures une méthode de data mining peut trouver que les attributs type de véhicule, puissance et poids sont corrélés et déduire des contraintes d'intégrité et des domaines de valeurs d'attributs.
D'autre méthodes peuvent trouver que dans 99% des cas WGT$\in$[1000,2000] et ainsi déduire le domaine de l'attribut WGT. Grâce à ces méthodes des problèmes structurels au niveau d'une source peuvent être détectés. Par exemple si l'algorithme de data mining trouve que dans 100% des cas WGT $\in$[1000,2000] cependant une des valeurs de l'attribut WGT = 999 un problème de violation d'intégrité référentielle est alors détecté.

B- La détection et résolution des conflits dans les sources

En ce qui concerne les conflits structurels et sémantiques entre plusieurs sources, les méthodes de data mining sont utilisées pour les détecter et les résoudre. Par exemple si le kilométrage d'une même voiture est exprimée dans une source en kilomètres et dans une autre en en mètres les algorithmes utilisent la régression pour trouver les facteurs de conversion.Pour la résolution des problèmes sémantiques dans une source les algorithmes de data mining utilisent les règles d'associations entre les attributs. Par exemple si le kilométrage d'une voiture vieille de dix ans est égal à 123 mètres. Ce problème sémantique dans une source du à une erreur de saisie peut être résolu grâce à l'utilisation de la règle d'association suivante : (Age>10) $\rightarrow$(Mil>100000).

6- Les ontologies : Une ontologie est une hiérarchie de concepts, de relations et de contraintes imposées pour modéliser un domaine particulier. Elle permet de résoudre les problèmes sémantiques entre plusieurs sources. En effet l'adhésion à une terminologie de référence (ontologie) est une réponse pour éviter d'une part l'emploi des mots polysémiques, en veillant à ce que chaque terme désigne un concept unique du domaine. Par exemple, dans le domaine médical le sens de « sinus » sera différencié dans « sinus paranasal » et « sinus pinonidal ». Ceci va également permettre de supprimer les synonymes, du fait que chaque concept sera désigné par un terme unique et normalisé. D'autre part, la possibilité de hiérarchiser va permettre de résoudre le problème de différence de niveau de dénomination, en effet chaque terme générique (exemple « infarctus) sera relier explicitement aux termes plus spécifiques (« infarctus de myocarde », « infarctus pulmonaire »).

[Moulton et al. 02] utilise une ontologie pour détecter et résoudre les conflits liés à l'hétérogénéité des sources. Les étapes suivantes résument leur approche :

- **Construction de l'ontologie**: ils représentent toutes les connaissances d'un domaine spécifique dans un modèle conceptuel. Ce modèle décrit en plus des classes, des attributs et des liens entre les classes. Il décrit : 1) les types sémantiques qui ajoutent de la connaissance sémantique à un attribut en plus de son nom et de son type par ex : le format de l'attribut date est dit type sémantique et sa valeur (JJ/MM/AA) est dite paramètre (modifier) ; 2) les catégories conceptuelles qui représentent des codes utilisés seulement par le concepteur du système pour positionner les données les unes par rapport aux autres ; 3) Les contraintes qui ajoutent de l'information sémantique entre les d'attributs ; 4) les règles qui décrivent des relations d'équivalences entre les valeurs d'attributs par ex : prix total = prix HT + TVA
- **Conception de modèles conceptuels de contextes:** ils définissent un modèle conceptuel à chaque relation source et à la relation du schéma global en utilisant les contraintes et les règles spécifiées dans l'ontologie. Chaque modèle est un héritage du modèle conceptuel représentant l'ontologie.
- **Définition des mappings :** Ils définissent tout d'abord des correspondances entre les attributs de la relation du schéma global et les attributs de l'ontologie et ensuite des correspondances entre les attributs sources et les attributs de l'ontologie, en se basant sur les contraintes définies entre les attributs. Par exemple l'attribut date dans le schéma global et l'attribut maturityDate dans l'ontologie, ces correspondances sont dites *Mappings d'attributs*.
- **Assignation du type sémantique :** ils assignent à chaque attribut source et à chaque attribut du schéma global un type sémantique, par exemple pour l'attribut *date* ils lui assignent le type sémantique *format*. A chaque type sémantique d'un attribut ils associent une valeur dite

paramètre, par exemple la valeur du type sémantique *format* de l'attribut *Date* est : *jj/mm/aaaa*. Ils se réfèrent aux types sémantiques définies dans l'ontologie.

- **Détection et résolution des conflits :** Une fois les contextes définis, ils résolvent les conflits sémantiques liés au schéma par l'utilisation des *mappings d'attributs* spécifiés au préalable. Ils détectent les conflits sémantiques liés aux données par la comparaison des types sémantiques et de leurs valeurs, pour la résolution de ce type de conflits ils appellent des fonctions externes.

7- Les correcteurs et les dictionnaires linguistiques : L'utilisation de correcteurs basés sur des dictionnaires électroniques facilite la détection et la correction automatique des problèmes sémantiques dans une source.

Exemples

- Détecter et corriger des erreurs d'orthographes.

- Détecter et corriger des erreurs dans les adresses en utilisant un dictionnaire géographique (nom ville, code postal, etc.)

- Détecter et corriger les problèmes de synonymie, d'homonymie, de différence de niveaux de dénominations en utilisant des dictionnaires linguistiques susceptibles d'apporter des stratégies de désambiguïsation sémantique tels que :

WORDNET : qui est une base de données lexicographique de l'anglais, développée depuis une quinzaine d'années à l'université de Princeton (http://www.cogsi.princeton.edu/~wn/) sous la direction du Professeur George A.Miller .

Cette base contient les catégories grammaticales suivantes : noms, verbes, adjectifs et adverbes à l'exclusion des mots grammaticaux. Elles sont organisées selon leur sens et non leur forme, ce qui importe n'est pas la définition du mot mais de situer celui-ci dans un réseau sémantique ou lexical (hiérarchique) de mots, de ce point de vue WordNet ressemble plus à un thesaurus qu'à un dictionnaire.

La notion de sens dans WordNet est représentée par un ensemble de synonymes (synset), accompagné d'une courte définition du mot.

Exemple1 : pour le nom « leg » la base lexicale contient neuf sens (synsets) :

1- **leg** –(a human limb, commonly used to refer to a whole limb but technically only the part between the knee and ankle)
2- **leg** –(a structure in animals that is similar to a human leg and used for locomotion)
3- **leg** –(one of the supports for a piece of furniture)
4- branch, fork, **leg** – (a part of a forked or branching shape, « he broke off one of the branches », « they took the south fork »)
5- **leg** –(the limb of an animal used for food)
6- Peg, wooden leg**, leg**, pegleg – (a prothesis that replaces a missing leg)
7- **leg** – (the part of garment that covers the leg)
8- **leg**—((nautical) the distance traveled by a sailing vessel on a single tack)
9- stage, **leg**—(a section or portion of a journey or course, « then we embarked on the second stage of our carribean cruise »)

Cette notion de sens est complétée par les liens qu'entretiennent les synsets avec d'autre synsets du lexique. Ils sont reliés entre eux par des relations de : synonymie, hyponymie, hyperonymie, antonymie, meronymie

Exemple 2

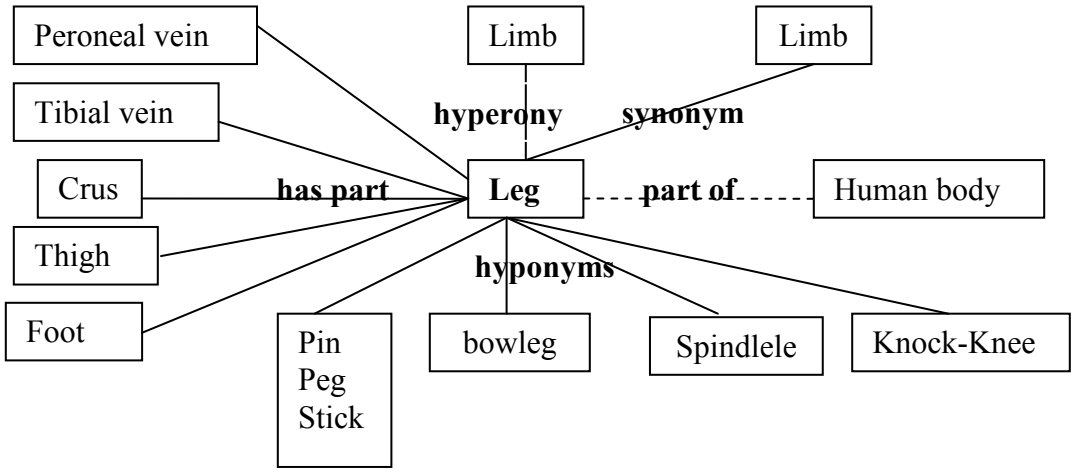


Figure 6 : Exemple de réseau pour le premier sens de leg

Le projet Européen EuroWordnet développé ces dernières années, a permis d'étendre le réseau lexical WordNet en définissant une base commune de concepts, dans plusieurs langues y compris le Français et ainsi construire une base de données multilingues qui enregistre des relations sémantiques entre les mots d'une langue à une autre, tout en respectant le standard de formalisation introduit par WordNet.

8- Les fonctions de transformations : Elles sont utilisées pour transformer la valeur d'un attribut en une autre. Ces fonctions peuvent être définies dans une librairie telle que la librairie de fonctions offerte par Oracle, comme elles peuvent être des procédures génériques.

La plupart des approches utilisent des fonctions de transformations existantes pour résoudre les conflits liés à l'hétérogénéité des sources et garantir la conformité mutuelle des données et leur conformité par rapport au schéma global.

[Galhardas et al.00] utilise la fonction « lowerCase » disponible dans la librairie offerte par le SGBD Oracle pour convertir les caractères majuscules en caractères minuscules. [Moulton et al.02] utilise des fonctions de conversion définies comme des méthodes dans [Bressan et al.99] tels que la fonction « scaleFactor » et la fonction moneyAmt qui transforment la valeur du revenu en une autre.

Le tableau suivant illustre quelques fonctions génériques disponibles dans le SGBD Oracle 9i :

Fonction	Principe	Syntaxe	Exemple
`Lower`	Convertit les caractères minuscules en majuscules	`Lower(Chaine)`	`Lower(ville)`
`Upper`	Convertit les caractères en majuscules	`Upper(Chaine)`	`Lower(ville)`
`Substr`	Extrait une sous-chaîne d'une chaîne	`Substr(chaîne,début[,compte])`	`Substr(Adr,5,8)`
`Soundex`	Compare la sonorité des mots et détecte les correspondances	`Soundex(chaîne)`	`Select * where Soundex(ville) = Soundex(Paris)`
`Decode`	Effectue une substitution val par val	`Decode(valeur,if1,then1,else1,if2,then2,…,else)`	`Decode(ville, 'Alm','Alemagne', 'Frc', 'France')`
`Translate`	Effectue une substitution caractère par caractère	`Decode(valeur,if1,then1,else1,if2,then2,…,else)`	`Translate(7671234,23456789 0, 'bcdefghij ')`

Ces fonctions sont disponibles dans :
http://downloadwest.oracle.com/otndoc/oracle9i/901doc/server.901/a90125/functions2.htm

9 - Les opérateurs logiques : ils sont utilisés pour détecter et résoudre les conflits sémantiques liés au schéma et les conflits sémantiques liés aux données. Ces opérateurs peuvent être composés pour réaliser des transformations de données plus complexes.

[Calvanese et al.99] proposent d'intégrer les transformations de données comme des ornements de requêtes de médiation (adorned queries) exprimés en logique de description dans un contexte LAV. Chaque objet d'une source est défini par une requête agrémentée d'annotations servant à résoudre les conflits liés à l'hétérogénéité des sources. Trois opérateurs sont proposés pour supporter ces annotations : *Convert, Match et Reconcile*. *Convert* réalise les conversions mono-source, *Match* permet de comparer deux objets de deux sources différentes, et *Reconcile* est un programme de nettoyage qui s'exprime à l'aide d'un ensemble d'opérations *Convert* et *Match*. La sémantique de chacune de ces opérations est définie par un programme externe utilisé lors de l'évaluation de requêtes.

[Galhardas et al.00] traitent la détection et la résolution de plusieurs conflits liés à l'hétérogénéité des sources dans une application particulière. Ils proposent le système « AJAX» ou sont définis les opérateurs logiques (opérateurs de nettoyage) : *Mapping*, *Matching*, *Clustering* et *Merging*. *Mapping* est un opérateur qui permet d'unifier et d'homogénéiser les données dans une relation par exemple : la conversion de tous les caractères majuscules en minuscules, changer le format de la date, l'extraction d'une partie de la valeur d'un attribut, corriger les erreurs de

saisie…etc. Il prend en entrée une relation et produit en sortie une ou plusieurs relations. Ce prétraitement des données permet de faciliter le processus de nettoyage des données. *Matching* prend en entrée deux relations et produit en sortie une seule relation. Il compare et calcule la distance entre deux valeurs d'attributs (Ex : comparer les noms Smith et Smich). Le résultât du matching est un degré de similitude dont la valeur est comprise entre 0 et 1. Ce degré de similitude est calculé par une fonction externe de similarité « nameSimF() » qui calcule la proximité entre deux noms. *Clustering* regroupe le résultat du matching dans un cluster (tuples qui représentent le même objet). *Merging* fusionne chaque cluster dans un seul tuple, il permet d'éliminer les valeurs doubles en faisant appel à une fonction « GetLongest() » qui choisit le nom le plus long.

Chaque opérateur logique peut faire appel à des fonctions externes lors des transformations conceptuelles (différentes selon les applications) telles que : la normalisation, l'extraction, le calcul de la distance entre deux valeurs, etc. Un langage déclaratif est proposé (SQL étendu) pour programmer les six opérateurs.

[Claypool et Rundensteiner 03] s'intéressent à la transformation de schéma et de données dans une source. Leur approche est basée sur un graphe « *Sangam* » où sont représentées toutes les attributs sources et leurs liens. Ils proposent quatre opérateurs pour effectuer des transformations simples sur le graphe : *Cross, Connect, Smoth et Subdivide. Cross* prend en entrée un nœud du graphe et produit en sortie un autre nœud. *Connect* prend en entrée un arc entre deux noeuds et produit en sortie un autre arc. *Smoth* prend en entrée deux arcs qui représentent deux liens entre deux paires d'attributs, il produit en sortie un lien qui est une combinaison des deux liens en entrée. *Subdivide* réalise la transformation inverse effectuée par l'opérateur *Smoth*, il prend en entrée un arc et produit en sortie deux arcs. Pour effectuer des transformations plus complexes sur le graphe, ils proposent deux techniques de composition : *Context dependency* et *Dérivation. Context dependency* permet de joindre les opérateurs entre eux et de produire en sortie un graphe. *Dérivation est une technique* qui permet de fusionner les opérateurs, les sorties de un ou plusieurs opérateurs sont les entrées d'un autre opérateur.

10- Le type sémantique d'un attribut : il est utilisé dans [Siegel et Madnick 91], [Sciore et al. 94], [Bressan et al. 99] et [Moulton et al 02] pour détecter par exemple un conflit sémantique lié à la différence d'unités (Francs, Pesetas) entre deux valeurs d'attributs.

Ces approches associent un type sémantique à un attribut, par exemple « unité » est un type sémantique associé à l'attribut « prix ». A chaque type est assigné une valeur sémantique, par exemple la valeur « euros » est assignée au type sémantique unité. Cette valeur est dite paramètre dans [Moulton et al.02] et contexte de valeur dans [Sciore et al. 94].

Les techniques de nettoyage de données utilisées pour la détection des conflits liés à l'hétérogénéité des sources sont dites *techniques de matching* dans [Rahm et Bernstein 01] où ils proposent une classification possible des techniques existantes. Ils référencent aussi des prototypes qui utilisent ces techniques tels que : Clio (IBM Almaden et Universités de Toronto) [Miller et al 01], Similarity flooding (Stanford Univ et Univ de Leipzig) [Melnik et al.02], Delta (MITRE), Tess(Uni de Massachusetts, Amherst) [Lerner 00], et Tree Matching (NYU) [ZSW00].

Compte tenu de la littérature étudiée sur l'intégration de données et sur la prise en compte des conflits liés à l'hétérogénéité des sources, très peu d'approches se sont intéressées à la détection et à la résolution d'un ensemble de conflits, la plupart se sont focalisées sur la détection et (ou) sur la résolution d'un conflit particulier. Parmi les approches qui ont défini des outils intégrant un ensemble de transformations nous citons les travaux : [Hellerstein et al.01] proposent un système interactif, nommé Potter's Wheel, pour la détection et la résolution des conflits basé sur un ensemble de transformations. Ce système permet de définir des domaines et de détecter les conflits lorsque les valeurs d'attributs ne sont pas conformes au domaine. [Karayannidis et al. 01] proposent un autre outil, ARKTOS, où des primitives de nettoyage permettent de détecter la violation de contraintes d'intégrité. Cet outil permet de modéliser et d'exécuter différents scénarios ETL (Extraction, Transformation and Loading). Le système AJAX [Galhardas et al. 02], développé à l'INRIA, a apporté une solution plus générique en proposant un ensemble d'opérateurs servant de briques de base à la réalisation manuelle de programmes de nettoyage de données. Ces opérateurs constituent une sorte de « patrons » (patterns) de nettoyage qu'il faut instancier en fonction des applications et des types de conflits rencontrés.

Après cette Etat de l'art sur les techniques existantes dans le domaine de la recherche, nous présentons dans la section suivante les outils commerciaux disponibles sur le marché industriel dans le contexte de l'intégration de données.

2-4- Les outils commerciaux

Une grande variété d'outils commerciaux est disponible sur le marché industriel pour garantir la conformité de données dans les sources hétérogènes. Trois catégories d'outils sont distinguées à savoir : les outils d'analyse de données, les outils de nettoyage de données et les outils ETL (Extraction Transformation Loading).

Les outils d'analyse de données sont des outils d'analyse qui permettent d'identifier les conflits liés à l'hétérogénéité des sources. *Validata* de DataMentors, et *MigrationArchitect de* Evoke Software sont des outils de diagnostic qui associent à chaque attribut un profil en se basant sur des méta-données telles que le type, la longueur, les valeurs min et max, la cardinalité, l'unicité, etc. Ils analysent les instances de chaque attribut par rapport aux méta-données, et ils détectent les problèmes relatifs à une source lorsqu'une une instance ne correspond pas au profil prédéfini. Par ex si le profil de l'attribut âge est $1<Age<100$, et si dans une source donnée la valeur de l'attribut âge = 120, un problème de violation de contrainte d'intégrité est détecté.

WizRule de WizSoft et *Data Mining Suite* de Information Discovery sont aussi des outils d'analyse qui utilisent des techniques de data mining. Ces outils analysent un ensemble d'attributs pour identifier les relations entre eux. Ils permettent de générer des relations mathématiques entre les attributs d'une ou plusieurs sources de données. Par exemple total = quantité∗prix unitaire est une relation mathématique qui associe les trois attributs total, quantité et prix unitaire.

Les outils de nettoyage de données sont des outils spécialisés dans le nettoyage d'attributs particuliers tels que l'attribut nom, adresse ou tout autre attribut dont les valeurs peuvent être comparé à une table de correspondance ou à un fichier existant. *IDCentric* de FirstLogic, *PureIntegrate* d'Oracle, *QuickAddress* de Qas Systems, *Reunion* de PitneyBowes, et *Trillium* de

Trillium Software, sont des outils commerciaux qui se focalisent sur le nettoyage de ce type de données. Ils utilisent des fonctions d'extraction, de standardisation, de conversion et de matching. D'autres outils tels que : *DataCleanser* de EDD, *Merge/Purge Library* de Sagent/QM Software, *MatchIT* de HelpIT Systems et *MasterMerge* de Pitney-Bowes s'intéressent à l'identification et à l'élimination de valeurs doubles. Cependant ils requièrent des sources de données déjà nettoyées pour faciliter l'identification des valeurs doubles.

Les outils ETL (Extraction Tranformation and Loading) *CopyManager de* Information Builders*, DataStage* d'Informix/Ardent*, Extract* d'ETI*, PowerMart* d'Informatica*, Data Transfomation Service* de Microsoft, *MetaSuite* de Minerva/Carleton, *Sagent Solution* de Plateform (Sagent), et *WarhouseAdministartor* de SAS sont des outils commerciaux chargés d'extraire des données depuis des sources, de les transformer et de les intégrer dans un système d'information global. Plusieurs fonctions de transformations y compris les fonctions citées dans les deux précédentes catégories peuvent être appelées pour transformer le format des données d'une plate-forme à une autre par exemple des fonctions de conversion du type de donnée (type date), des fonctions d'extraction, de concaténation, split, replace, sub-string, etc.

Ascential de IBM a développé un outil complet qui garantit la conformité des données dans les sources hétérogènes. Il intègre trois puissants outils à savoir : *MetaRecon* un outil de data profiling, *Integrity* un outil de nettoyage de données, et *DataStage* un outil ETL. Nous concluons cet état de l'art sur quelques liens vers ces outils commerciaux.

Vality: Ascential : http://www.ascential.com/products/iicontentedition.html
DataMentors: Validata http://www.datamentors.com
Evoke Software: MigrationArchitect http://www.evokesoft.com
WizSoft : WizRule http://www.wizsoft.com
Trillium Software : Trillium- Converter, Parser, Geocoder, Matcher http://www.trilliumsoft.com
Sagent/QM Software: Merge/Purge Library http://www.qmsoft.com
Information Builders (IBI): CopyManager http://www.ibi.com
Evolutionary technologies Inc (Eti): Extract http://www.eti.com
Informatica: PowerMart http://www.informatica.com
Minerva/Carleton MetaSuite : http://www.metasuite.com
SAS: WarhouseAdministartor http://www.sas.com

2-5- Synthèse

Au vu des articles de recherche étudiés, deux catégories de travaux se distinguent. Celle qui vise à interroger les sources de données distribuées et hétérogènes et à la définition de requêtes de médiation, et celle qui vise à intégrer les données et à construire le schéma global. Très peu de travaux se sont intéressées à la génération de requêtes de médiation et à la prise en compte des conflits liés à l'hétérogénéité des sources durant le processus de génération, la plupart se sont focalisés sur la détection et (ou) sur la résolution d'un conflit particulier dans le contexte de l'intégration de données.

Dans une première partie, nous avons présenté un état de l'art sur les quelques approches existantes qui visent la génération de requêtes de médiation, et qui s'inscrivent dans le contexte des systèmes multi-source en général.

Dans une seconde partie, nous avons présenté un état de l'art sur les nombreuses approches existantes sur l'intégration de données, et qui se focalisent sur les conflits liés à l'hétérogénéité des sources. Dans ce même contexte nous avons présenté les outils commerciaux existants sur le marché industriel.

La plupart des travaux cités dans la première partie référencent les travaux évoqués dans la seconde partie pour détecter et résoudre les conflits liés aux sources. Il est difficile d'interroger des sources hétérogènes et distribuées sans tenir compte des conflits liés à l'hétérogénéité. Lorsque la conformité mutuelle des données et leur conformité par rapport au schéma global n'est pas assurée, les requêtes peuvent ne pas s'exécuter ou encore être sémantiquement fausses. L'exploitation des techniques utilisées par les approches d'intégration de schéma et les approches d'intégration de données est donc une réponse qui permet de détecter et de résoudre les conflits durant le processus de génération de requêtes.

Les systèmes de médiation étant de nos jours des systèmes de plus en plus développés et connus, méritent une exploration plus approfondie dans le domaine de la recherche. Cette restriction de travaux a motivé notre étude à explorer plus en détails la problématique liée à ces systèmes et d'apporter une solution originale intégrant des fonctionnalités de nettoyage de données, adaptée aux attentes des utilisateurs.

Nous présentons dans les chapitres suivants notre approche qui s'inscrit principalement dans le contexte des systèmes de médiation, et qui consiste à générer des requêtes calculant une instance du schéma global à partir de sources de données distribuées et hétérogènes.

Chapitre 3- La génération de requêtes de médiation

3-1- Introduction

Un système d'informations multi-source est défini comme l'intégration de plusieurs sources de données distribuées et hétérogènes. Parmi ces systèmes d'informations multi-source, nous distinguons les entrepôts de données, les systèmes d'informations basés sur le web, les systèmes de bases de données fédérées ou encore les systèmes de médiation. Notre étude présentée ici s'inscrit principalement dans le contexte des systèmes de médiation, mais certains résultats peuvent être utilisés ailleurs comme les entrepôts de données par exemple.

Un système de médiation est défini comme un système qui permet d'interroger un ensemble de sources de données distribuées et hétérogènes. Ses composants essentiels sont : le schéma global appelé schéma de médiation, les mapping du schéma global avec les sources, les fonctions de réécriture de requêtes et les fonctions de composition des résultats. L'expression des requêtes de médiation varie selon l'approche choisie : 1) approche descendante (Global As View ou GAV) où chaque objet du schéma global est défini par une requête sur les sources, 2) approche ascendante (Local As View ou LAV) où chaque objet d'une source de données est défini par une requête sur le schéma global.

Plusieurs problèmes de conception émergent lors de l'utilisation de ces médiateurs. L'une des principales difficultés rencontrée dans un système de médiation est la définition de requêtes de médiation calculant une instance du schéma de médiation.

L'écriture manuelle des requêtes de médiation donne sans doute le résultat le plus pertinent au regard des besoins des utilisateurs. Cependant il est difficile de l'envisager pour la définition de requêtes de médiation en raison du grand nombre de sources de données qui peuvent être impliquées (des centaines ou des milliers) et du volume important de méta-données les décrivant (description des schémas des sources et du schéma global, assertions linguistique, assertions intra-source et inter-source, etc). La question principale est de savoir comment automatiser la génération de requêtes de médiation ?

Face à cette problématique, nous proposons dans ce chapitre pour un contexte relationnel une démarche de génération automatique de requêtes de médiation dans une approche GAV (Global As View) qui, à partir du schéma d'une relation de médiation, de dépendances fonctionnelles définies sur cette relation, d'assertions linguistiques déterminées entre les sources et le schéma de médiation, et d'assertions intra-source et inter-source reliant structurellement les relations sources entre elles, produit un ensemble de requêtes potentiel calculant cette relation.

L'approche intuitive de cet algorithme est la suivante : définir une requête de médiation pour une relation de médiation R_m consiste en une décomposition de la relation R_m en relations $T_1, T_2, \ldots . T_n$ telles que $R_m=Q(T_1, T_2, \ldots . , T_n)$ où Q est une expression de l'algèbre relationnelle définie sur les T_i. Les relations T_i sont obtenues par projection des relations sources sur leurs attributs communs respectifs avec la relation de médiation R_m. Parfois, il peut être utile d'inclure dans ces projections les clés des relations sources pour transposer dans les relations T_i les liens existants entre les relations sources. Il peut être aussi nécessaire de passer par des relations

sources intermédiaires pour établir ces liens. Les projections de ces relations sur leurs clés primaires et étrangères sont ajoutées à l'ensemble des relations T_i pour calculer la relation de médiation R_m. Les opérations de la requête de médiation Q sont déterminées par l'application d'un ensemble de règles de composition des T_i. Les étapes suivantes résument le processus de génération automatique de requêtes de médiation calculant une relation R_m du schéma de médiation:

1) Identification des relations sources pertinentes pour la définition d'une requête de médiation Q du schéma de médiation et génération de relations de mapping T_i qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation R_m.

2) Identification des opérations relationnelles possibles entre les relations de mapping T_i en fonction de leur schéma et de leurs clés et génération du graphe d'opérations.

3) Recherche des chemins de calcul à partir du graphe d'opérations pour calculer la relation de médiation R_m.

4) Génération de requêtes de médiation déduites à partir des chemins de calcul de la relation de médiation R_m.

Notre approche est fondée sur un ensemble d'hypothèses contenues dans une base de méta-connaissances ***A***. Cette base est constituée de :

- La description du schéma de médiation comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations,
- La description du schéma local à chaque source de données comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations d'une même source,
- Les assertions inter-source entre les relations de sources différentes et les assertions linguistiques entre les concepts des sources et les concepts du schéma de médiation (synonymie, abréviations, inclusions, et équivalences linguistiques des noms des attributs). Ces assertions ne sont pas définies au préalable par le concepteur du système, elles sont recherchées automatiquement et sont ajoutées dans la base de méta-connaissances au fur et à mesure de leur découverte au cours du processus de génération de requêtes de médiation.

Nous présentons ci-dessous un exemple simple pour faciliter la compréhension de notre approche.

Exemple

Pour notre exemple nous supposons la relation source `Véhicule` de la source S_1 et la relation source `Personne` de la source S_2 et une relation de médiation `vehicule_personne` de schéma `(num,marque,puissance,couleur,d_achat,prix_veh,nom_cond,prénom,adresse, date_nais)`

Véhicule

Num	marque	puissance	couleur	d_achat	prix_veh	nom-cond
902FEE75	Renault	7	Rouge	2000	1200	Dupond
434FR92	Renault	5	Noir	1998	15000	Martin
5647GT49	Peugeot	5	Vert	1996	8500	Martin

Personne

nom_pers	prénom	adresse	date_nais
Dupond	Paul	Paris	12/02/78
Dupont	Jacques	Paris	04/04/72
Martin	Georges	Lyon	31/01/72
Durand	Olivier	Dijon	01/04/67
Toule	Ange	Marseille	16/01/70

Soit la requête `Q` suivante qui calcule la relation de médiation `vehicule_personne` à partir des relations sources personne et véhicule :

```
Select *
From véhicule, personne
Where nom-cond = nom-pers
And adresse = paris
And prix < 1500
```

Pour cet exemple il est facile à l'utilisateur de définir la requête `Q` manuellement car le volume de méta-données à explorer n'est pas très important. Cependant La définition de requêtes de médiation est un problème complexe lorsque les sources sont multiples (des centaines ou des milliers) et le volume de méta-données qui les décrivent est important.

Pour résoudre la problématique liée à la définition de requêtes de médiation en regard d'un grand nombre de sources hétérogènes et du volume important de méta-données qui les décrivent, le but de notre approche est d'automatiser la génération de requêtes de médiation calculant une relation de médiation R_m du schéma de médiation. L'approche de génération automatique tente de trouver toutes les solutions possibles au calcul de la relation de médiation; ce qui rend le processus automatique aussi complexe. Mais sous certaines conditions simplificatrices que nous justifions, il est possible de générer dans des temps raisonnables un ensemble de solutions.

La complexité du processus de génération de requêtes est accrue lorsqu'on tient compte de l'hétérogénéité des données. Dans ce chapitre nous nous intéressons exclusivement aux conflits sémantiques liés au schéma qui proviennent du vocabulaire différent utilisé pour décrire le même concept (ex. prix et prix-produit). On suppose les conflits sémantiques liés aux données résolus (ex. différence d'unité de mesure euros et francs). Nous adressons la problématique liée à un environnement hétérogène dans le prochain chapitre.

Suite à cette introduction, nous présentons en section 2 un ensemble de définitions des concepts de base pour faciliter la compréhension de notre approche, en section 3 l'ensemble de méta-données utilisés pour la génération de requêtes, en section 4 la description détaillée des étapes de la génération de requêtes de médiation, et en section 5 nous présentons une approche de

génération de requêtes de médiation plus approfondie qui permet d'éviter d'énumérer toutes les requêtes parmi lesquelles l'utilisateur doit sélectionner celles qui correspondent le mieux à ses besoins.

3-2- Notations et définitions préliminaires

Avant d'aborder le principe et la description détaillée des algorithmes de génération de requêtes de médiation, nous introduisons dans cette section les concepts de base utilisés dans notre approche, suivis d'exemples simples pour faciliter la compréhension des algorithmes de génération.

Comme nous l'avons dit en introduction, la génération de requêtes de médiation consiste à générer un ensemble de requêtes de médiation $\boldsymbol{Q} = \{Q_1, Q_2, \ldots Q_n\}$ qui calculent une relation de médiation particulière à partir d'un ensemble de sources de données. Soit une relation de médiation R_m décrite par un ensemble d'attributs $\boldsymbol{X} = \{A_1, A_2, \ldots An\}$, et soit $\boldsymbol{S} = \{S_1, S_2, \ldots S_n\}$ l'ensemble de sources, chaque source S_i peut contenir un ensemble de relations sources $\boldsymbol{R} = \{R_1, R_2, \ldots R_n\}$ et chaque relation source R_i est décrite par un ensemble d'attributs sources $\boldsymbol{Y} = \{B_1, B_2, \ldots B_n\}$.

Définition d'une relation de mapping : Soit $R_m(X)$ une relation du schéma de médiation et $R_i(Y)$ une relation source, $T_i(Z)$ est une *relation de mapping* dérivée à partir de la relation $R_i(Y)$ si cette dernière possède des attributs communs avec la relation de médiation $R_m(X)$. Plus formellement si $Z = X \cap Y$, Z constitue l'ensemble d'attributs communs dits attributs de mapping.

Une relation de mapping $T_i(Z)$ est calculée par la projection de la relation source $R_i(Y)$ sur ses attributs communs avec la relation de médiation $R_m(X)$, $T_i(Z) = \prod_Z R_i(Y)$

La relation source $R_i(Y)$ est dite dans ce cas une relation source contributive au calcul de la relation de médiation.

Exemple

Soit la relation de médiation vehicule_personne de schéma (marque,puissance, prix_veh,nom_cond,prénom,adresse,date_nais), et les relations sources Véhicule et Personne suivantes :

Véhicule = (num,marque,puissance,couleur,d_achat,prix_veh,nom_cond)
Personne = (nom_pers,prénom,adresse,date_nais)

Les relations de mapping obtenues par la projection des relations sources Véhicule et Personne sur leurs attributs équivalents avec la relation de médiation sont :

T_1 = (marque,puissance,prix_veh,nom_cond)

T_2 = (nom_pers,prénom,adresse,date_nais)

Parfois il peut arriver que les clés primaires et étrangères des relations sources ne soient pas incluses dans les relations de mappping, dans ce cas de figure il peut être utile de les ajouter dans les projections pour transposer au niveau des relations de mapping les liens existant entre les relations sources. Nous désignons ses projections par des *relations de mapping étendu*.

Définition d'une relation de mapping étendue : Soit $R_m(X)$ une relation du schéma de médiation, $R_i(Y)$ une relation source et $T_i(Z)$ est une relation de mapping de base, $T_i(Z')$ est une *relation de mapping étendu* si elle comporte les attributs communs, ainsi que les clés primaires et étrangères de la relation $R_i(Y)$.

Une relation de mapping $T_i(Z')$ est calculée par la projection de la relation source $R_i(Y)$ sur les attributs équivalents avec $R_m(X)$ ainsi que sur l'ensemble des clés**,**
$T_i(Z') = \prod_{Z'} R_i(Y)$, Z' constitue l'ensemble des attributs communs et des clés ajoutées.

L'ensemble des relations de mapping étendu associées à une relation de médiation sur l'ensemble des sources S est noté M_e.

Exemple

Soit la relation de médiation vehicule_personne de schéma (marque,puissance, prix_veh,nom_cond,prénom,adresse,date_nais), et la relation source Véhicule suivante :

Véhicule = (num,marque,puissance,couleur,d_achat,prix_veh,nom_cond)

La relation de mapping étendue obtenue par la projection de la relation source Véhicule sur ses attributs communs avec vehicule_personne et sur sa clé primaire num est :

T_1 = (num,marque,puissance,prix_veh,nom_cond)

Il peut être aussi utile de passer par d'autres relations sources pour pouvoir établir des liens entre les relations. Nous désignons ces relations par des *relations de transition*.

- Nous distinguons deux types d'assertions sémantiques; 1) les assertions intra-source (contrainte référentielle) sont définies entre deux relations qui appartiennent à la même source ; 2) les assertions inter-source sont des liens sémantiques qui permettent de lier deux relations qui appartiennent à des sources différentes. Ces liens sont établis automatiquement entre deux attributs de sources différentes si les deux attributs sont « terminologiquement » équivalents, et au moins un des deux attributs est clé dans sa relation.

Définition d'une relation de transition : Soit R_m une relation du schéma de médiation, T_i et T_j deux relations de mapping dérivées à partir des relations sources R_i et R_j respectivement et R une relation source. Une relation $T(k_1,k_2)$ telle que $T = \prod_{k1,\ k2} R$ est une relation de transition entre les deux relations de mapping T_i et T_j si :

- T_i et T_j n'ont aucun attribut commun (aucun attribut de T_i n'est terminologiquement équivalent à un attribut de T_j),

- la relation R est non contributive au calcul de la relation de médiation R_m,
- il existe une assertion définie entre R et R_i et une assertion définie entre R et R_j; chaque assertion est soit une contrainte référentielle (assertion intra-source) soit une assertion inter-source.

L'ensemble des relations de mapping de transition associé à une relation de médiation sur l'ensemble des sources S est noté M_t

Exemple

Considérons la relation de médiation R_m de schéma (num_veh,marque,puissance, nom_pers,prénom,adresse), où num_veh est l'attribut clé de la relation.

La recherche des relations de mapping a conduit à identifier deux relations sources

R_1=(num_veh,marque,puissance,couleur,prix_veh)
R_2=(num_pers,nom_pers,prénom,adresse,date_nais)

Nous supposons que ces deux relations appartiennent à la même source de données. Les deux relations de mapping étendu correspondantes sont :

T_1 = (num_veh,marque,puissance,nom_cond)
T_2 = (num_pers,nom_pers,prénom,adresse,date_nais)

Les deux relations T_1 et T_2 n'ayant aucun attribut commun, aucun opérateur ne peut leur être appliqué. Pour trouver une requête de médiation calculant R_m nous sommes conduit à rechercher une ou plusieurs relations dans les sources pouvant être utilisés pour combiner les deux relations de mapping. Supposons qu'il existe dans la même source de données une relation :R_3(num_cond,num_pers,profession)telle que : num_cond est la clé primaire et telles que les assertions intra-source suivantes sont définies : R_1.num_cond $\subseteq$ R_3.num_cond et R_3.num_pers $\subseteq$ R_2.num_pers. La relation de schéma T_3(num_cond, num_pers)calculée par T_3(num_cond, num_pers)= $\prod$ num_cond, num_pers R_3 est utile car bien que n'ayant aucun attribut équivalent avec la relation de médiation elle permet de générer une requête de médiation (deux jointures entre T_1,T_2 et T_3). C'est ce type de relation que nous appelons relation de transition.

Au vu de cet exemple, plusieurs remarques peuvent être faites :

- Dans l'exemple, une seule relation de transition suffit pour permettre de combiner les relations de mapping étendu. Mais il est possible que plusieurs relations doivent être utilisées pour combiner les deux relations de mapping. Dans ce cas de figure plusieurs assertions successives sont impliquées pour lier deux relations de mapping représentées dans un graphe d'assertions. On dit alors qu'il existe une séquence d'assertions entre R_i et R_j. Cette séquence peut inclure à la fois des assertions inter-source et intra-soure.

- Entre deux relations d'une même source il peut exister des contraintes référentielles, s'il existe une telle contrainte entre deux attributs A et B, une jointure peut être effectuée entre les deux relations sur le critère A=B. Mais entre deux relations de sources différentes, ce type de contrainte ne peut exister explicitement. Nous considérons qu'une jointure avec le critère A=B entre deux relations appartenant à des sources distinctes peut être définie si les deux

attributs A et B sont « terminologiquement » équivalents, et au moins un des deux attributs est clé dans sa relation.

Graphe d'opérations : un graphe d'opérations G_{Rm} est constitué de l'ensemble des opérateurs relationnels qui combinent les relations de mapping étendu et de transitions. Un graphe G_{Rm} est défini par un couple (M,O) de deux ensembles finis et disjoints. Les éléments de l'ensemble $M=\{m_1, m_2, \ldots m_n\}$ appelés les sommets de G_{Rm} représentent l'ensemble des relations de mapping et de transitions. L'ensemble O est l'ensemble des éléments $\{o_1, o_2, \ldots o_n\}$ appelés arêtes de G_{Rm} représentent les opérations relationnelles qui lient les relations de mapping entre elles. Une opération o_i peut combinée soit une relation de mapping étendue avec une relation de mapping étendue, soit une relation de mapping étendue avec une relation de mapping de transition, ou encore deux relations de transition.

Exemple

Soit la relation de médiation vehicule_personne de schéma(num_veh,marque, puissance,couleur,date_location,nom,prénom,adresse,date_nais),

Soient les relations sources Véhicule et Personne suivantes :

Véhicule = (num_veh,marque,puissance,couleur,prix_veh,num_pers)
Personne = (num_pers,nom,prénom,adresse,date_nais,num_location)
Location = (num_location,date_location,prix_location,date_retour)

Les relations de mapping obtenues par la projection des relations sources Véhicule,Personne et location sur leurs attributs équivalents avec la relation de médiation et sur les clés sont :

T_1 = (num_veh,marque,puissance,couleur,num_pers)

T_2 = (num_pers, nom, prénom, adresse,date_nais, num_location)

T_3 = (num_location, date_location)

Supposons que les trois relations appartiennent à la même source et qu'elles sont liées par des contraintes référentielles suivantes Vehicule.num_pers ⊆ Personne.num_pers et Personne.num_location ⊆ Location.num_location

La figure1 illustre le graphe d'opérations représentant les deux opérations de jointure qui combinent les relations de mapping étendu entre elles.

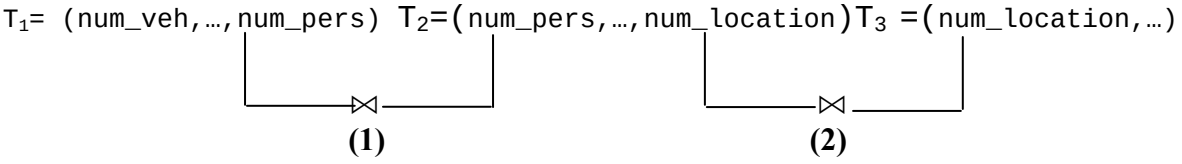


Figure 1 : le graphe d'opérations

Définition d'un chemin de calcul : La génération de requêtes de médiation se fait en recherchant des chemins de calcul dans le graphe d'opérations défini précédemment.

Un chemin C_{Rm} est un chemin de calcul dans le graphe G_{Rm} associé à une relation de médiation R_m dans les cas de figure suivants :

- C_{Rm} est un sous graphe connexe acyclique du graphe G_{Rm} telle que chaque attribut de la relation de médiation soit équivalent à un attribut figurant dans le sous graphe. En d'autres termes, chaque attribut de la relation de médiation figure au moins dans une des relations de mapping du sous graphe représenté par C_{Rm}.
- C_{Rm} est constitué d'un seul nœud représentant une relation de mapping T telle que chaque attribut de la relation de médiation soit équivalent à un attribut dans T. En d'autres termes, la relation T contient tous les attributs de médiation.

  Plus formellement un chemin de calcul est défini par :

  $C_{Rm}=\{(T_1,T_2,o_{12}),(T_2,T_3,o_{23}),\ldots(T_{n-1},T_n,o_{n-1,n})\}$, où $((T_i,T_{i+1},o_{i,i+1})$ est un arc reliant deux relations de mapping.

Exemple

Considérons le graphe d'opérations précèdent, le chemin de calcul qui permet de calculer la relation de médiation est le suivant :

- $C_{Rm} = \{(T_1, T_2, \bowtie),(T_2, T_3, \bowtie)\}$

Il peut exister plusieurs chemins de calcul qui permettent de calculer la même relation de médiation.
La génération de requêtes de médiation consiste donc à identifier pour chaque chemin de calcul tous les ordonnancements possibles d'opérateurs. La requête de médiation dérivée à partir du chemin C_{Rm} est la suivante :

- $Q_{Rm} = \Pi_{\text{num\_veh, marque, puissance, couleur, date\_location, nom, prénom, adresse, date\_nais}} (T_1 \bowtie_{T_2.\text{num\_pers}=T2.\text{num\_pers}} T_2 \bowtie_{T2.\text{num\_location}=T3.\text{num\_location}} T_3)$

Nous avons présenté dans cette section les définitions et les notations des concepts de base utilisées dans la génération de requêtes de médiation. Avant d'aborder le principe des algorithmes de génération en détail nous présentons dans la section suivante les connaissances exploitées par les algorithmes.

3-3- Méta-données utilisées

Avoir une bonne connaissance du schéma global et de chaque schéma local est nécessaire dans la définition de requêtes de médiation pour répondre au mieux aux besoins des utilisateurs.

Dans cette section nous présentons l'ensemble de méta-données exploitées par le processus de génération de requêtes de médiation. Certaines de ces connaissances sont prédéfinies par le concepteur du système de médiation telles que : la description des schémas de relations, les clés des relations, les dépendances fonctionnelles, les contraintes référentielles entre relations, et d'autres sont ajoutées dans la base de connaissances au fur et à mesure de leur découverte automatique au cours du processus de génération de requêtes de médiation telles que : les correspondances linguistiques entre les concepts des sources et les concepts du schéma de médiation, et les correspondances linguistiques entre les relations de sources différentes.

La base de connaissances notée ***A*** est constituée de trois catégories de méta-données à savoir :
- Les méta-données au niveau des sources,
- Les méta-données au niveau de la médiation,
- Les méta-données entre la médiation et les sources.

Nous présentons tout d'abord les méta-données définies au niveau des sources de données, ensuite les méta-données définies au niveau de la médiation, et enfin celles définies entre le niveau de la médiation et le niveau des sources.

3-3-1- Méta-données au niveau des sources

Les méta-données définies au niveau des sources décrivent le schéma de chaque source de données, l'ensembles des relations sources appartenant à chaque schéma source, les clés des relations, les dépendances fonctionnelles éventuelles, les attributs de chaque relation, les assertions intra-source et inter-source entre les relations.

Description d'un schéma source

- Un schéma source est constitué d'un ensemble de relations sources reliées entre elles par des assertions intra-source.

- Une assertion intra-source est une contrainte référentielle définie entre deux relations R_1 et R_2 appartenant à la même source de données. Elle est notée :
  `a = lhs.rel[lhs.attr]` $\subseteq$ `rhs.rel[rhs.attr]` tel que:

  `a.lhs.rel` : est la relation à gauche de l'assertion
  `a.rhs.rel` : est la relation à droite de l'assertion
  `a.lhs.attr`: est l'attribut à gauche de l'assertion, clé étrangère dans `a.lhs.rel`
  `a.rhs.attr`: est l'attribut à droite de l'assertion clé primaire dans `a.rhs.rel`

- Les attributs clés des relations sources sont soulignés. Par exemple pour la relation R_1 ($\underline{B_1}, B_2, B_3$) l'attribut $\underline{B_1}$ est clé primaire de la relation.

- Le symbole R_i désigne une relation source et le symbole $\underline{R_i}$ désigne le schéma de cette relation.

- Une dépendance fonctionnelle notée $B_1 \rightarrow B_2$ est dite valide dans une relation $R_1(\underline{B_1}, B_2, B_3)$ si pour une valeur de B_1 correspond une et une seule valeur de B_2.

- Dans ce chapitre nous supposons que nous sommes dans un environnement « semi-hétérogène », où seuls les conflits sémantiques liés au schéma sont considérés, les conflits sémantiques liés à l'hétérogénéité des données sont supposés résolus. Ainsi en l'absence d'hétérogénéité, les attributs sources sont caractérisés par leur nom et leur type de base. Dans le cas d'un environnement hétérogène nous introduisons la notion de type étendu à chaque attribut source pour permettre de résoudre la problématique liée à l'hétérogénéité des données. Cette notion est développée dans le chapitre 4.

Description de plusieurs schémas sources

- Plusieurs schémas sources sont un ensemble de sources de données reliées entre elles par des assertions inter-source. En effet lorsqu'il s'agit de deux relations sources appartenant à deux sources de données différentes elle sont reliées entre elles par des assertions inter-source et non pas par des contraintes référentielles comme pour le cas d'un seul schéma source.

- Une assertion inter-source est donc une correspondance linguistique qui lie un attribut B d'une relation R_1 de la source S_1 à un attribut B'd'une autre relation R_2 de la source R_2. Cette assertion est notée `a = lhs.rel[lhs.attr] ≅ rhs.rel[rhs.attr]` où `lhs.attr` et `rhs.attr` sont deux attributs équivalents liés par une correspondance linguistique (synonymie, abréviations, équivalence linguistique des noms).

- Ces assertions inter-source n'existent pas au préalable dans la base de connaissances elles sont recherchées automatiquement et ajoutées au fur et à mesure de leur découverte au cours du processus de génération de requêtes de médiation.

3-3-2- Méta-données au niveau du schéma de médiation

Les méta-données définies au niveau de la médiation caractérisent le schéma de médiation, l'ensemble des relations de médiation appartenant à ce schéma de médiation, les clés des relations, les dépendances fonctionnelles éventuelles, et les attributs de chaque relation.

- Un schéma de médiation est constitué d'un ensemble de relations de médiation.

- Le symbole R_m désigne la relation de médiation et le symbole $\underline{R_m}$ désigne son schéma.

- Les attributs clés des relations de médiation sont soulignés. Par exemple pour la relation R_m ($\underline{A_1}, A_2, A_3$) l'attribut A est clé primaire de la relation.

- L'ensemble d'assertions définies sur une relation de médiation est essentiellement des dépendances fonctionnelles qui relient l'attribut clé aux attributs non clés.

3-3-3- Méta-données entre la médiation et les sources

Les méta-données entre la médiation et les sources sont des correspondances linguistiques reliant un attribut d'une relation de médiation à un attribut d'une relation source. En d'autres termes un attribut A d'une relation de médiation R_m est relié par une correspondance linguistique à un attribut B d'une relation source R_i. Cette assertion est notée $a = R_m.A \cong R_i.B$ où A et B sont deux attributs équivalents liés par une correspondance linguistique (synonymie, abréviations, équivalence linguistique des noms).

Ces méta-données permettent de résoudre les conflits sémantiques au niveau du schéma liés à l'utilisation d'une terminologie différente pour designer deux concepts identiques. Par exemple l'attribut *prix* dans la relation de médiation R_m et l'attribut *prix-produit* dans la relation source R_i sont reliés par une correspondance linguistique (équivalence des noms) sous la forme de $a = R_m.prix \cong R_i.prix\text{-}veh$ pour designer qu'il s'agit bien du même concept.

Ces méta-données n'existent pas au préalable dans la base de connaissances, elles sont aussi recherchées automatiquement et ajoutées au fur et à mesure de leur découverte au cours du processus de génération de requêtes de médiation.

Dans cette section nous avons introduit les méta-données utilisées par le processus de génération de requêtes dans un environnement homogène, où on suppose les conflits sémantiques liés à l'hétérogénéité des données résolus. Dans le chapitre 4 nous présentons les méta-données supplémentaires dont a besoin le processus de génération pour générer des requêtes tout en tenant compte des conflits liés à un environnement hétérogène.

En résume la base de connaissances ***A*** est constituée de :

- la description du schéma de médiation comportant les schémas de relations, les clés des relations, dépendances fonctionnelles, les contraintes référentielles entre relations,

- la description du schéma local à chaque source de données comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les assertions intra-source (contraintes référentielles) entre les relations d'une même source et les assertions inter-source (correspondances linguistiques) entre les relations de sources différentes,

- des correspondances linguistiques entre les concepts des sources et les concepts du schéma de médiation (synonymie, abréviations, inclusions, et équivalences linguistiques des noms des attributs).

Nous présentons dans le chapitre 5 la description du méta-modéle exploité par le processus de génération de requêtes (figure 2).

3-4- La génération de requêtes de médiation

Un des principaux problèmes rencontrés dans la conception d'un système de médiation est le problème de définition de requêtes calculant une relation du schéma de médiation. En raison du grand nombre de sources de données qui peuvent être impliquées (des centaines ou des milliers) et du volume important de méta-données les décrivant (description des schémas des sources et du schéma global, assertions de correspondance linguistique, assertions intra-source,…etc.), il est difficile d'envisager une écriture manuelle des requêtes de médiation. La question principale est de savoir comment automatiser la génération de requêtes de médiation ?. En d'autres termes, comment, à partir du schéma d'une relation de médiation, et d'un ensemble de sources de données, produire automatiquement un ensemble potentiel de requêtes calculant cette relation ?

En réponse à cette problématique nous proposons dans cette section pour le contexte relationnel une approche de génération automatique de requêtes de médiation. Nous supposons le schéma de médiation déjà défini ainsi que l'ensemble de méta-données décrites dans la section précédente. Nous nous plaçons dans une approche GAV (Global As View) où chaque objet du schéma global est défini par une requête sur les sources de données.

Approche intuitive

L'approche intuitive de cet algorithme est la suivante : définir une requête de médiation pour une relation de médiation R_m consiste en une décomposition de la relation R_m en relations T_1, T_2,….T_n telles que $R_m=Q(T_1, T_2,….,T_n)$ où Q est une expression de l'algèbre relationnelle définie sur les T_i. Les relations T_i sont obtenues par projection des relations sources sur leurs attributs communs respectifs avec la relation de médiation R_m. Parfois, il peut être utile d'inclure dans ces projections les clés des relations sources pour transposer dans les relations T_i les liens existants entre les relations sources. Il peut être aussi nécessaire de passer par des relations sources intermédiaires pour établir ces liens. Les projections de ces relations sur leurs clés primaires et étrangères sont ajoutées à l'ensemble des relations T_i pour calculer la relation de médiation R_m. Les opérations de la requête de médiation Q sont déterminées par l'application d'un ensemble de règles de composition des T_i. Les étapes suivantes résument le processus de génération automatique de requêtes de médiation calculant une relation R_m du schéma de médiation:

1) Identification des relations sources pertinentes pour la définition d'une requête de médiation Q du schéma de médiation, et génération de relations de mapping T_i qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation R_m.

2) Identification des opérations relationnelles possibles entre les relations de mapping $T_i$en fonction de leur schéma et de leurs clés, et génération du graphe d'opérations.

3) Recherche des chemins de calcul à partir du graphe d'opérations pour calculer la relation de médiation R_m.

4) Génération de requêtes de médiation déduites à partir des chemins de calcul de la relation de médiation R_m.

Comme nous l'avons dit en introduction notre approche présentée ici se limite à prendre en compte l'hétérogénéité des schémas dans la génération automatique des requêtes de médiation, elle exploite des correspondances linguistiques définies entre les concepts. Elle ne tient pas compte des conflits liés à l'hétérogénéité des données, elle les suppose résolus. Nous reviendrons dans le chapitre 4 sur la génération de requêtes de médiation dans un environnement hétérogène.

Suite à cette introduction nous présentons ci-après le principe et la description de chaque étape du processus de génération de requêtes, les spécifications détaillées des algorithmes sont présentées en annexe.

3-4-1. Recherche des relations mapping

La première étape de la génération de requêtes de médiation consiste à identifier les relations sources pertinentes au calcul de la relation de médiation R_m, et à générer des relations de mapping T_i qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation R_m.

Pour une relation de médiation donnée R_m, la recherche des relations de mapping s'effectue en considérant successivement les relations de chaque source de données. Pour chaque relation source R_i, chaque attribut B de R_i est comparé aux attributs de la relation de médiation en se basant sur les correspondances linguistiques définies entre un attribut d'une relation source et un attribut d'une relation de médiation.

Lorsque l'ensemble d'attributs communs noté E entre la relation de médiation et la relation source est différent de l'ensemble vide, les clés primaires et étrangères sont recherchées en se basant sur les dépendances fonctionnelles, les contraintes référentielles et sur les assertions inter-source. Les relations obtenues sont alors appelées relations de mapping étendu.

Une relation de mapping étendu est donc obtenue en effectuant une projection de la relation source sur les attributs équivalents avec R_m ainsi que sur l'ensemble des clés déterminées contenu dans l'ensemble E. L'ensemble des relations de mapping étendu associées à une relation de médiation sur l'ensemble des sources $\boldsymbol{S}$ est noté M_e. L'algorithme suivant illustre le principe de la recherche des relations de mapping. Sa spécification détaillée est présentée en annexe 1.

```
Recherche de MappingEtendu (Rm (X), S, Me)
Entrée : Rm (X): la relation de médiation
         S : l'ensemble de sources de données
Sortie : Me : l'ensemble de relations de mapping étendu

1.  Me : = ∅
2.  Pour chaque source S dans S
3.   Pour chaque relation source R dans S
4.      Tester l'équivalence entre chaque attribut de R et les attributs de Rm
         et ajout des attributs équivalents dans l'ensemble E
5.      RechercheCléP (R, K) // recherche de la clé primaire K de R dans A
6.      RechercheCléE (R, B) // recherche des clés étrangères de R dans A
7.      Ajout des clés déterminées dans E
8.      Me = Me ∪{Ti = ΠE R (Y)}// Création d'une relation de mapping étendu Ti
9.   FinPour
10. FinPour
Fin Recherche de MappingEtendu
```

Algorithme1 : algorithme de recherche de mapping étendu

Exemple

Considérons la relation de médiation R_m de schéma (num_veh,marque,puissance, nom_pers,prénom,adresse), et les deux relations sources R_1, R_2 suivantes :

R_1 = (num_veh,marque,puissance,couleur,prix_veh,nom_cond), où num_veh est la clé primaire et nom_cond est la clé étrangère
R_2 = (num_pers,nom_pers,prénom,adresse,date_nais), où num_pers est la clé primaire

Nous supposons que les deux relations sources appartiennent à la même source de données S_1. Compte tenu des correspondances linguistiques définies entre un attribut d'une relation source et un attribut d'une relation de médiation l'algorithme de recherche des relations de mapping prend en entrée la relation de médiation et les relations sources, il compare chaque attribut source avec chaque attribut de médiation. La recherche des relations de mapping identifie deux ensembles d'attributs équivalents à savoir :

E_1 = (num_veh,marque,puissance)
E_2 = (nom_pers,prénom,adresse)

Comme nous l'avons dit précédemment lorsque l'ensemble d'attributs équivalents entre la relation de médiation et la relation source est différent de l'ensemble vide, les clés primaires et étrangères sont recherchées. La projection de chaque relation source sur ses attributs équivalents avec R_m ainsi que sur l'ensemble des clés déterminées produit les deux relations de mapping étendu suivantes :

T_1 = (num_veh,marque,puissance,nom_cond)
T_2 = (num_pers,nom_pers,prénom,adresse,date_nais)

Une fois les relations de mapping étendu T_i générées le but est de trouver des opérations relationnelles pour les combiner, principe de l'étape 2 de l'algorithme cité en introduction. Cependant lorsqu' il n'y pas d'attributs communs entre deux relations de mapping étendu T_i et T_j aucun opérateur relationnel ne peut leur être appliqué. Cela revient donc à chercher en plus des relations de mapping étendu, une ou plusieurs relations dans les sources pouvant être utilisés pour combiner les deux relations de mapping. Nous désignons ces relations par des relations de transition.

La recherche des relations de mapping de transition pour une relation de médiation particulière revient à chercher pour chaque paire de relations de mapping étendu (T_i,T_j) entre lesquelles aucune assertion n'est définie, s'il existe parmi les assertions définies dans la base de méta-connaissances ***A*** une séquence d'assertions permettant de lier la relation R_i(respectivement R_j) à d'autres relations sources hormis les relations contributives. R_i et R_j sont les relations sources qui ont conduit à dériver T_i et T_j.

La recherche des séquences d'assertions est effectuée par une procédure qui est appelée par l'algorithme de recherche des transitions. Cette procédure prend en entrée les relations source R_i et R_j et la base de connaissances ***A***. Elle cherche pour toute assertion `a` contenue dans la base ***A*** s'il existe un lien entre la relation d'origine R_i avec d'autres relations. S'il existe, la procédure concatène l'assertion `a` identifiée à la séquence d'assertions courante notée `SeqCourante`, elle continue à chercher des assertions jusqu'à ce qu'elle identifie une séquence d'assertions pertinente permettant de lier la relation origine R_i à la relation cible R_j.`RechercheSéquence` est une procédure complexe et récursive, elle cherche toutes les séquences possibles entre R_i et R_j.

Une fois les séquences d'assertions calculées, l'algorithme de recherche de transitions prend en entrée les séquences d'assertions pertinentes contenues dans l'ensemble `SeqTrouvée`, il déduit pour chaque séquence pertinente la (les) relation de transition. Ces relations sont obtenues par la projection des relations sources intermédiaires sur leurs attributs clés.
L'ensemble des relations de mapping de transition associées à une relation de médiation sur l'ensemble des sources ***S*** est noté M_t.

Nous présentons ci-dessous le principe de l'algorithme de recherche de mapping de transition et le principe de la procédure de recherche de séquences. Les spécifications détaillées des algorithmes sont présentées en annexe 2 et 3.

```
Recherche de MappingTransition (Me, SeqTrouvée, Mt)
Entrée : SeqTrouvée : l'ensemble de séquences d'assertions
         Me: l'ensemble de relations de mapping étendus
Sortie : Mt: l'ensemble de relations de mapping de transition

1. Mt : = ∅
2. Pour  chaque paire de relation de mapping étendu (Ti,Tj) dans Me tel que
     Ti ∩ Tj = ∅
3.   RechercheSéquence (Ri, Rj, A, SeqCourante, SeqTrouvée)
     //Rechercher toutes les séquences d'assertions dans A qui lient Ri et Rj
4.     Mt : = Mt ∪{Π B, B' (R'(Y))} // Création d'une relation de transition qui
   est une projection de la relation pertinente R' sur les attributs B et B'
5. FinPour
Fin Recherche de MappingTransition
```

Algorithme 2 : Algorithme de recherche de mapping de transition

```
RechercheSéquence (Ri, Rj, A, SeqCourante, SeqTrouvée)
Entrée : Ri : la relation source
         Rj : la relation cible
         A : l'ensemble d'assertions
         SeqCourante : la séquence d'assertions courante
Sortie : SeqTrouvée : l'ensemble de séquences d'assertions trouvées

1. Tantque Ri ≠ Rj
2.  Pour toute assertion a dans A
3.     RechercheSequence (Ri, Rj, SeqCourante ‖ a)
       // Procédure récursive qui cherche toutes les séquences  d'assertions
       dans A entre Ri et Rj
4.  FinPour
5. FinTantque
6.   SeqTrouvée : = SeqTrouvée  ∪ SeqCourante
     // Création d'une séquence d'assertions pertinente
Fin RechercheSéquence
```

Algorithme 3 : procédure de recherche de séquences d'assertions

Exemple

Soient la relation de médiation R_m de schéma(num_veh,marque,puissance,nom_pers, prénom,adresse), et les deux relations de mapping étendu obtenues par la recherche des relations de mapping :

T_1 = (num_veh,marque,puissance,nom_cond)
T_2 = (num_pers,nom_pers,prénom,adresse,date_nais)

Les deux relations T_1 et T_2 n'ayant aucun attribut commun, aucun opérateur ne peut leur être appliqué. L'algorithme de recherche des relations de transitions tente de trouver une ou plusieurs relations dans les sources pouvant être utilisés pour combiner les deux relations de mapping. Supposons qu'il existe dans la même source de données une relation
R_3 (num_cond,num_pers,profession) tel que num_cond est la clé primaire et telles sur les assertions intra-source suivantes sont définies: R_1.num_cond $\subseteq$ R_3.num_cond et
R_3.num_pers $\subseteq$ R_2.num_pers. La recherche de séquences d'assertions identifie une seule séquence contenant les deux assertions successives qui permet de lier la relation d'origine R_1 à la relation cible R_2.

SeqTrouvé = {((R_1. num_cond $\subseteq$ R_3. num_cond),
(R_3. num_pers $\subseteq$ R_2. num_pers))}

A partir de cette séquence d'assertions la recherche des relations de transitions déduit la relation de transition de schéma T_3 = (num_cond, num_pers) calculée par la projection de la relation intermédiaire R_3 sur ses attributs clé (primaire et étrangère) num_cond et num_pers
T_3(num_cond,num_pers)=$\prod_{num_cond,num_pers}$$R_3$(num_cond,num_pers,profession).

Dans l'exemple, une seule relation de transition suffit pour permettre de combiner les relations de mapping étendu. Mais il est possible que plusieurs relations doivent être utilisées pour combiner les deux relations de mapping ; Ce qui peut rendre la recherche des relations de transition complexe. Cette complexité est accrue par la procédure *RechercheSequence* qui tente de trouver toutes les séquences d'assertions possibles pour combiner deux relations.

La complexité de la recherche des relations de transition est de l'ordre de $O(|Me|^2[\Delta^n + nlogn+|S|^2)$, où Δ est le nombre maximum de sommets voisins d'un sommet et n le nombre de sommets. Le plus coûteux dans cet algorithme est la recherche des séquences d'assertions dont la complexité est de $O(\Delta^n)$. Elle est exponentielle en fonction de la dimension du graphe du fait de la recherche de toutes les séquences d'assertions possibles entre deux relations de mapping étendu T_i et T_j.

Réduire la complexité de l'algorithme de recherche de mapping de transition revient donc à réduire la complexité de la procédure récursive ***RechercheSequence***.

Au vu des articles de recherche étudiés sur la théorie des graphes, [Martins et Al. 98] proposent une généralisation de l'algorithme de Dijkstra. Le principe de cet algorithme consiste à chercher les *K* plus courts chemins entre deux relations sources R_i et R_j. Il permet de passer d'une complexité exponentielle à une complexité polynomiale de l'ordre de $O(K.n)$ (où K est le nombre de séquences d'assertions et n le nombre de sommets).

Pour résoudre cette problématique nous avons adapté le principe de l'algorithme de Dijkstra généralisé à la recherche des séquences d'assertions. Nous présentons ici une optimisation de la recherche des séquences d'assertions et de l'algorithme de mapping de transition. Les spécifications détaillées de ces deux algorithmes sont présentées en annexe 4 et 5.

```
RechercheSéquenceOptimisé (Ri, Rj, A, K, SeqTrouvée)
Entrée : Ri : la relation source
         Rj : la relation cible
         A : l'ensemble d'assertions
         K : le nombre de plus courts chemins entre Ri et Rj
Sortie : SeqTrouvée : l'ensemble de séquences d'assertions trouvées

1. Count_Rj := 0 // Le nombre de fois où une relation cible est traitée
2. SeqTrouvée := ∅
3. Tantque (count_Rj < K) et Ri ≠ Rj
4.  Pour toute assertion a dans A
5.     RechercheSequenceOptimisée (Ri, Rj, A, K, SeqTrouvée)
       // Procédure qui cherche les K plus courtes séquences
       d'assertions dans A entre Ri et Rj
6.  FinPour
7. FinTantque
8.   SeqTrouvée : = SeqTrouvée  ∪ SeqCourante
     // Création d'une séquence d'assertions pertinente
FinRechercheSéquenceOptimisé
```

Algorithme 4: Optimisation de la recherche des séquences

Le principe de l'algorithme de recherche de mapping de transition ne change pas. L'algorithme prend cette fois en entrée l'ensemble `Seqtrouvée` qui contient les `K` plus courtes séquences d'assertions entre les relations sources R_i et R_j produites par la procédure `RechercheSéquenceOptimisée` et non pas toutes les séquences d'assertions produites par la procédure `RechercheSéquence`.

```
Recherche de MappingTransition (Me, SeqTrouvée, Mt)
Entrée : SeqTrouvée : l'ensemble de séquences d'assertions
         Me: l'ensemble de relations de mapping étendus
Sortie : Mt: l'ensemble de relations de mapping de transition

1. Mt : = ∅
2. Pour  chaque paire de relation de mapping étendu (Ti,Tj) dans Me tel que
     Ti ∩ Tj = ∅
3  RechercheSéquenceOptimisée (Ri, Rj, A, K, SeqTrouvée)
     //Rechercher des K meilleures plus courtes séquences d'assertions dans
     A qui lient Ri et Rj
4.   Mt : = Mt ∪{Π B, B' (R'(Y))}
     // Création d'une relation de transition qui  est une projection de la
   relation pertinente R' sur les attributs B et B'
5. FinPour
Fin Recherche de MappingTransition
```

Algorithme 5: Algorithme de recherche de mapping de transition

3-4-2- Recherche du graphe d'opérations

Etant donné l'ensemble de relations de mapping étendu M_e, et l'ensemble de relations de mapping de transition M_t générés par la recherche des relations de mapping, le but de cette étape est de trouver des opérations relationnelles susceptibles de combiner chaque paire de relations en tenant compte des méta-données décrites en section 3. Dans cette étude on se limite à la recherche des opérations de jointures.

La recherche de ces opérations est guidée par des règles d'intégration spécifiées sur les connaissances. Ces règles d'intégration permettent, pour une relation de médiation particulière de déterminer l'ensemble des jointures candidates, et ce pour chaque paire de relations de mapping. Une opération de jointure déterminée entre deux relations de mapping peut combiner soit une relation de mapping étendu avec une autre relation de mapping étendu, soit une relation de mapping étendu avec une relation de transition, ou encore une relation de transition avec une autre relation de transition. L'ensemble de ces opérateurs est représenté par un graphe d'opérations noté G_{RM} où chaque nœud correspond à une relation de mapping, et chaque arc entre deux nœuds correspond à une jointure candidate déterminée à l'aide d'une règle d'intégration.

Considérons la règle d'intégration suivante : Si les deux relations appartiennent à la même source et que leurs schémas ne sont pas disjoints, et où l'une des relations référence l'autre alors l'opération candidate est une jointure naturelle déterminée par la règle suivante :

Si $\underline{T_i} \cap \underline{T_j} \neq \emptyset$ et $T_i.B \subseteq T_j.B$

Alors $T_i.B \bowtie T_j.B$

Si les deux relations n'appartiennent pas à la même source de données, on ne pourra pas disposer de contraintes référentielles auquel cas on utilisera la correspondance linguistique entre deux attributs, où l'un des deux attributs est clé dans l'une ou l'autre des relations. Dans ce cas la règle d'intégration est la suivante :

Si $\underline{T_i} \cap \underline{T_j} \neq \emptyset$ et $T_i.B \cong T_j.B'$

Alors $T_i.B \bowtie T_j.B'$

Compte tenu de ces règles d'intégration nous présentons ci-dessous le principe de l'algorithme de recherche du graphe d'opérations. La spécification détaillée de cet algorithme est présentée en annexe 6.

```
Recherche de GrapheOpérations (Me, Mt,J)
Entrée : Me : l'ensemble de relations de mapping étendus
         Mt : l'ensemble de relations de mapping de transition
Sortie : J : l'ensemble de Jointures

1. J : = ∅
2. Pour  chaque paire (Ti,Tj) dans Me et Mt  tel que Ti ∩ Tj ≠ ∅
3.    Si Ti et Tj appartiennent à la même source
4.    Alors J := J ∪ {j = Ti.B  ⋈  Tj.B}
      // Création d'une opération de jointure entre Ti et Tj sur le
     critère de jointure Ti.B = Tj.B
5.   Sinon J := J ∪ {j = Ti.B  ⋈  Tj.B'}
6.   FinSi
7. FinPour
Fin Recherche de GrapheOpérations
```

Algorithme 6 : Algorithme de recherche du graphe d'opérations

Exemple

Soit la relation R_m de schéma (num_veh,marque,puissance,nom_pers,prénom,adresse), soient les deux relations de mapping étendu T_1,T_2 et la relation de transition T_3:

T_1 = (num_veh, marque, puissance, num_cond)
T_2 = (num_pers, nom_pers, prénom, adresse,date_nais)
T_3 = (num_cond, num_pers)

Nous supposons que les trois relations appartiennent à la même source. Pour cet exemple la recherche des opérations de jointure candidates concerne chaque paire de relations de mapping. En d'autres termes L'algorithme cherche s'il existe une opération de jointure susceptible de combiner :

- T_1 avec T_2,
- T_1 avec T_3
- T_2 avec T_3

Supposons qu'il est défini dans la base de connaissances $\mathcal{A}$ les contraintes référentielles suivantes sources :

R_1.num_cond $\subseteq$ R_3.num_cond, R_3.num_pers $\subseteq$ R_2.num_pers

Au sens de la première règle d'intégration deux opérations de jointures sont produites à savoir :

J_1 = T_1.num_cond $\bowtie$ T_3.num_cond

$J_2 = T_3.num\_pers \bowtie T_2.num\_pers$

Le graphe d'opérations G_{RM} représentant les deux jointures candidates, où chaque nœud correspond à une relation de mapping, et chaque arc entre deux nœuds correspond à une jointure candidate déterminée à l'aide de la règle d'intégration est le suivant :

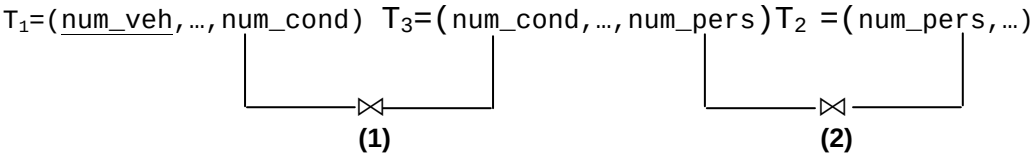


Figure 2 : exemple de graphe d'opérations

3-4-3. Recherche des chemins de calcul et définition des requêtes de médiation

Les règles d'intégration permettent, pour une relation de médiation et pour l'ensemble de relations de mapping associées, de déterminer l'ensemble des opérations de jointures candidates, et ce pour chaque paire de relations de mapping. L'ensemble de ces opérations est représenté dans le graphe d'opérations décrit précédemment, où chaque nœud correspond à une relation de mapping, et chaque arc entre deux nœuds correspond à un opérateur candidat déterminé à l'aide d'une règle d'intégration.

La génération de requêtes de médiation se fait en recherchant des chemins de calcul dans le graphe d'opérations G_{RM}.

Comme nous l'avons décrit dans les définitions (section2), un chemin de calcul C_{RM} associé à la relation de médiation R_m est un sous-graphe connexe et acyclique du graphe G_{RM} où chaque attribut de la relation de médiation est équivalent à un attribut figurant dans le sous-graphe. En d'autres termes, chaque attribut de la relation de médiation figure dans au moins une des relations de mapping du sous graphe. Il peut arriver que tous les attributs de R_m figurent tous dans une seule relation de mapping, dans ce cas le chemin de calcul est constitué d'un seul noeud représentant une relation de mapping.

L'algorithme de recherche des chemins de calcul proposé ici est un processus récursif complexe qui consiste à chercher dans le graphe de jointures G_{RM} tous les chemins possibles permettant de calculer la relation de médiation R_m. Il prend en entrée le graphe d'opérations et la relation de médiation. Il teste tout d'abord pour une jointure donnée reliant deux relations de mapping si tous les attributs de la relation de médiation figurent dans les deux relations, si oui un chemin de calcul est déjà identifié, sinon il ajoute la jointure `J` au chemin de jointures courant `ChemCourant` et il continu à chercher dans le graphe une jointure ayant un lien avec le chemin de jointures courant jusqu'à ce qu'il trouve un chemin de calcul pertinent où tous les attributs de R_m figurent. Un lien entre une jointure donnée et un chemin courant est établi si l'extrémité droite ou gauche de la jointure est égale à l'une des extrémités du `chemcourant.`

Exemple

Soit la jointure (1) entre la relation de mapping T_1 et T_3 dans le graphe d'opérations précèdent,

les attributs de la relation de médiation ne figurant pas tous dans les relations de mapping, la jointure (1) est ajoutée au chemin courant, le processus prend ensuite en entrée la deuxième jointure entre les relations T_3 et T_2 et teste si ses extrémités sont égales à l'une des extrémités de la jointure (1), Etant donné que l'extrémité gauche (T_3)de la jointure(2)est égale à l'extrémité droite (T_3) de la jointure (1) et que tous les attributs de Rm figurent dans les relations T_1, T_2 et T_3 alors le chemin de jointures $C_{Rm} = \{(T_1, T_3, \bowtie), (T_3, T_2, \bowtie)\}$est identifiée. Le chemin de calcul candidat est alors ajouté à l'ensemble de chemins trouvés ChemTrouvé.

Pour notre exemple un seul chemin est candidat, mais il peut arriver que plusieurs chemins soient candidats au calcul d'une relation de médiation.

Exemple

Soit le graphe d'opérations suivant associé à la relation de médiation R_m (K, A, B, C) :

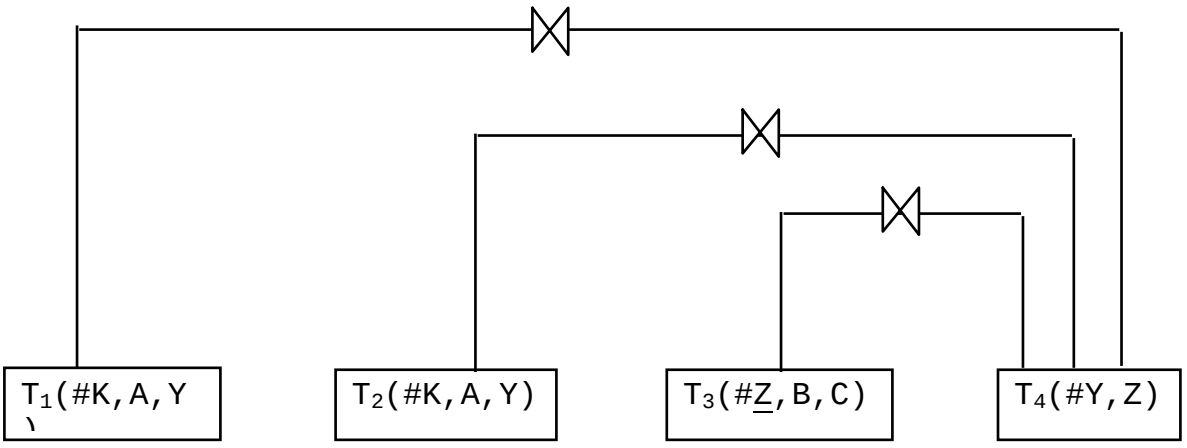
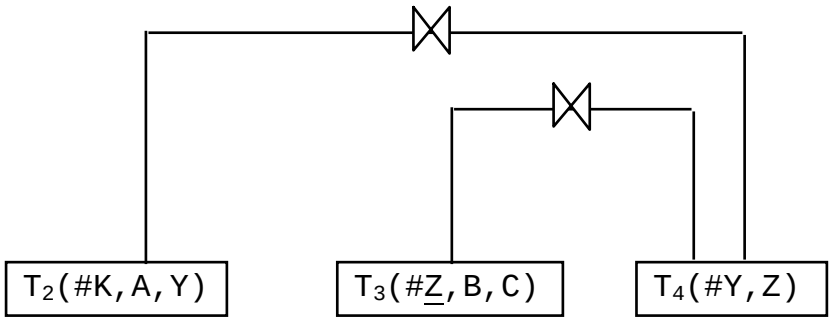
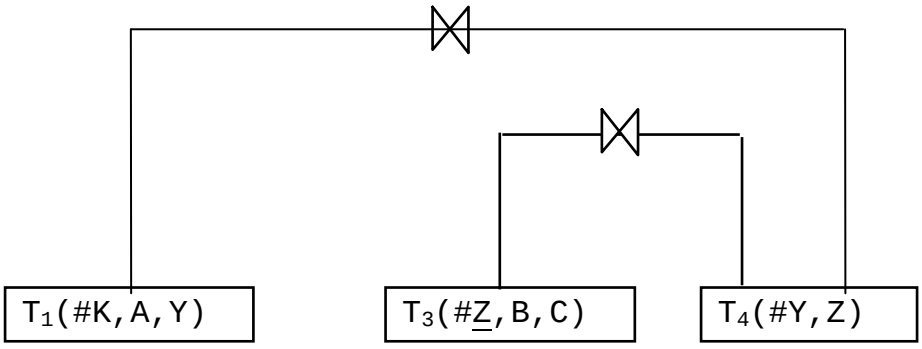


Figure 3 : Exemple de graphe d'opérations

Les chemins de calculs obtenus à partir du graphe d'opérations sont illustrés dans la figure 4 :



$C_{Rm} = \{(T_2, T_4, \bowtie), (T_4, T_3, \bowtie)\}$]



$C_{Rm} = \{(T_1, T_4, \bowtie), (T_4, T_3, \bowtie)\}$]

Figure 4 : Exemple de chemins de calcul dans un graphe d'opérations

Le processus de recherche des chemins de calcul est réitéré jusqu'à ce tous les chemins possibles soient identifiés. La spécification détaillée de cet algorithme est présentée en annexe 7.

```
RechercheChemin (XCourant, X, G_RM , ChemCourant, ChemTrouvé)
Entrée : XCourant : l'ensemble des attributs du chemin courant
         X : l'ensemble des attributs de la relation R_m.
         G_RM : le graphe d'opérations
         ChemCourant : le chemin de jointures courant
Sortie : ChemTrouvé : l'ensemble de chemins de jointures trouvées

1. TantQue le chemin courant ne contient pas tous les attributs de R_m
2.   Pour toute jointure j ∈ G_RM
3.     RechercheChemin (XCourant, X, ChemCourant ‖ j)
       // Procédure récursive qui cherche toutes les chemins de jointures
       dans G_RM qui calculent R_m
4.   FinPour
5. FinTantQue
6.     ChemTrouvé : = ChemTrouvé ∪ ChemCourant
       //création d'un chemin de calcul
FinRechercheChemin
```

Algorithme 7 : Algorithme de Recherche des chemins de calcul

Lors de la recherche des chemins de calcul un prétraitement est effectué pour rechercher s'il existe une relation de mapping ou de transition où figurent tous les attributs de la relation de médiation. Dans le cas où cette relation existe un chemin de calcul est identifié et est ajouté à l'ensemble de chemins trouvés.

La complexité de cet algorithme est de l'ordre de $O(\Delta^n)$, où Δ est le nombre maximum de sommets voisins d'un sommet et n le nombre de sommets. Elle est exponentielle en fonction de la dimension du graphe du fait de la recherche de tous les chemins de jointures possibles au calcul de la relation de médiation. Nous justifions dans le chapitre 6 par des tests d'évaluation réalisés sur notre outil (décrit dans le chapitre 5) qu'il est possible sous certaines conditions de générer dans des temps raisonnables des chemins de calcul en regard d'un grand nombre de sources et de méta-données.

Une fois tous les chemins de calcul identifiés, notre objectif final est de générer des requêtes de médiation pour calculer la relation du schéma de médiation.

La requête de médiation dérivée à partir du chemin $C_{Rm} = \{(T_1, T_3, \bowtie), (T_3, T_2, \bowtie)\}$ est la suivante :

$Q_{Rm} = \Pi_{\text{num\_veh, marque, puissance, nom\_pers, prénom, adresse}}\{(T_1 \bowtie_{\text{T1.num\_cond = T3.num\_cond}} T_3 \bowtie_{\text{T3.num\_cond >= T2.num\_cond}} T_2)$.

A l'issue de la phase de détermination des relations sources pertinentes, les relations de mapping étendu et de transition sont déterminées. Puis les opérateurs candidats entre ces relations sont définis en utilisant les règles d'intégration. Sur le graphe d'opérations représentant l'ensemble des opérateurs, les chemins de calcul sont ensuite recherchés. Enfin, les requêtes de médiation sont dérivées des chemins identifiés. La figure suivante récapitule la démarche de génération de requêtes de médiation.

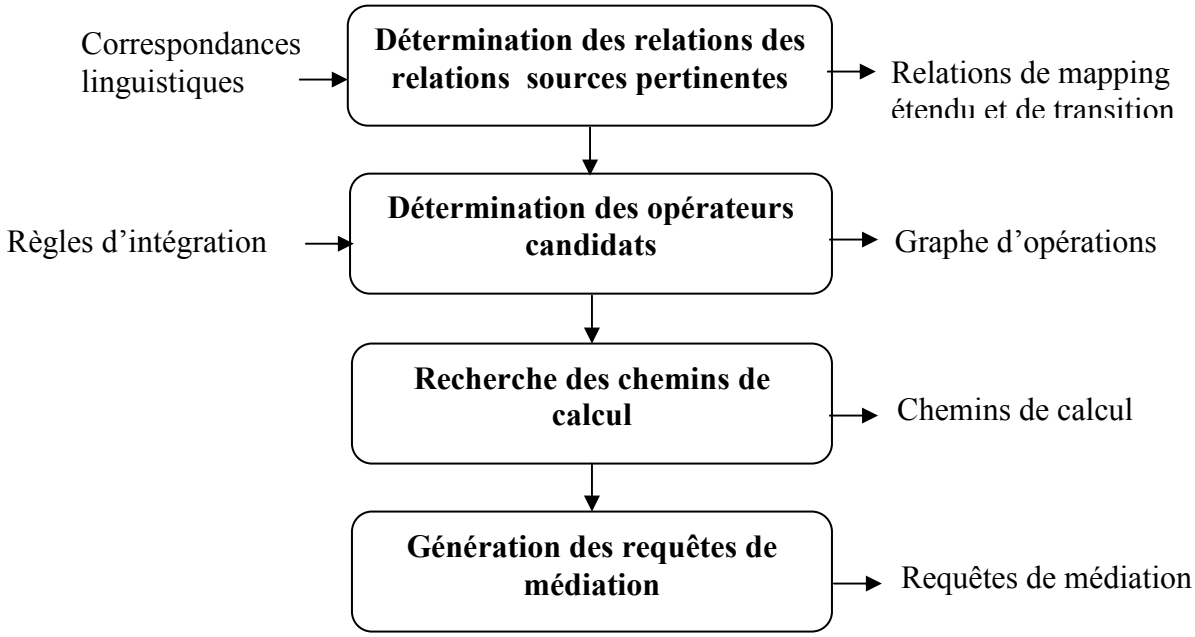


Figure 5 : Etapes de la génération de requêtes de médiation

Parmi les requêtes de médiation générées par le processus de génération, il peut exister des requêtes qui ne répondent pas aux besoins des utilisateurs en termes de contraintes d'intégrité. La question que l'on se pose est de savoir comment éviter de générer des requêtes de médiation non valides au cours du processus de génération de requêtes de médiation, et ainsi décharger l'utilisateur de faire un choix sur les requêtes produites ?

Face à cette problématique nous présentons dans la section suivante une approche de génération plus approfondie qui tient compte des besoins des utilisateurs en termes de contraintes d'intégrité. Notre approche se résume à la prise en compte des dépendances fonctionnelles.

3-5- Amélioration de la génération de requêtes de médiation

Face à la problématique de définition de requêtes de médiation dans un système multi-source en regard d'un grand nombre de sources de données distribuées et hétérogènes, nous avons proposé

une approche de génération automatique de requêtes. A partir du schéma d'une relation de médiation, de contraintes d'intégrité définies sur cette relation et de correspondances linguistiques existant entre les sources et le schéma de médiation, l'algorithme de génération produit un ensemble potentiel de requêtes calculant cette relation.

Par définition un schéma de médiation (respectivement un schéma source) est décrit par sa structure mais aussi par un ensemble de contraintes d'intégrité telles que : les dépendances fonctionnelles, les contraintes de valeurs, etc.

La non prise en compte de ces contraintes d'intégrité lors du processus de génération de requêtes peut fausser la sémantique des requêtes produites. En d'autres termes lorsque les contraintes d'intégrité définies sur le schéma global ne sont pas préservées par les requêtes de médiation, la validité de ces dernières est remise en cause. Nous présentons ci-dessous un exemple simple pour faciliter la compréhension de cette problématique.

Exemple :

Considérons la relation de médiation R_m de schéma (num_veh, marque, puissance, nom_pers, prénom, adresse), et les deux relations sources R_1, R_2 suivantes :

Véhicule = (num_veh, marque, puissance, couleur, prix_veh, nom_cond), où num_veh est la clé primaire

Personne = (num_pers, nom_pers, prénom, adresse, date_nais), où num_pers est la clé primaire

Soit la requête SQL générée par le processus de génération

```
Select *
From véhicule, personne
Where nom-cond = nom-pers
```

Supposons que la dépendance fonctionnelle définie sur la relation de médiation : num_veh → marque soit violée c'est-à-dire la même clé identifie deux objets différents ce qui entraîne un problème d'identification d'objets. La requête produite n'ayant pas préservée la DF ne peut être considérée comme une requête valide.

Après le problème de la génération automatique des requêtes de médiation face à un grand nombre de sources de données, le second problème est le test de validité d'une requête par rapport aux dépendances fonctionnelles définies sur le schéma de médiation.

En réponse à cette problématique, nous proposons une extension du processus de génération de requêtes par la prise en compte des contraintes d'intégrité comme les dépendances fonctionnelles.

On peut aborder ce problème de deux façons différentes, selon que l'on élimine les requêtes ne validant pas les dépendances fonctionnelles définies sur le schéma de médiation, ou de détecter le plus tôt possible la non validité de ces dépendances fonctionnelles pour éviter d'énumérer les requêtes correspondantes. Dans le premier cas, il s'agit d'éliminer à la fin du processus de

génération les requêtes qui ne peuvent pas être inférées de la fermeture transitive des dépendances fonctionnelles des sources. Par définition l'ensemble de dépendances fonctionnelles définies sur la relation de médiation R_m est préservé par la requête de médiation Q si la fermeture transitive des dépendances fonctionnelles notées DF définies sur la relation de médiation R_m est inclue ou égale à la fermeture transitive de l'union des DF définies sur l'ensemble de relations sources ***R***. Plus formellement :

$$D_{Rm}^{+} \subseteq (D_{R1} \cup D_{R2} \cup \ldots . \cup D_{Rn})^{+}$$

Dans le second cas, les dépendances fonctionnelles sont utilisées pendant le processus de génération afin de détecter le plutôt possible la non validité des requêtes et d'éviter l'énumération de toutes les requêtes correspondants à la relation de médiation. Notre démarche présentée ici s'inscrit dans le second cas de figure.

Détecter des requêtes de médiation non valides lors du processus de génération revient à revisiter l'algorithme de recherche des chemins afin d'identifier les chemins de calcul invalides dans le graphe d'opérations. Nous rappelons brièvement le principe de la génération de requêtes.

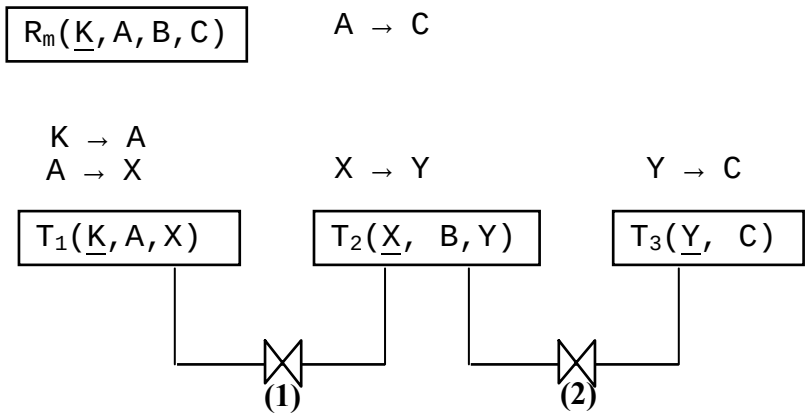
A l'issue de la phase de détermination des relations sources pertinentes, les relations de mapping étendu et de transition sont déterminées. Puis les opérateurs candidats entre ces relations sont définis. Sur le graphe représentant l'ensemble des opérateurs, les chemins de calcul sont ensuite recherchés. Enfin, les requêtes de médiation sont dérivées des chemins de calcul identifiés.

L'identification des chemins non valides au cours du processus de génération est guidée par des règles d'invalidation spécifiées sur un chemin de calcul. Ces règles d'invalidation permettent, pour une relation de médiation donnée de détecter les chemins de calcul qui ne préservent pas les dépendances fonctionnelles définies sur cette relation.

Un chemin de calcul est dit valide par rapport à une dépendance fonctionnelle $A \rightarrow B$ définie sur la relation de médiation R_m s'il existe une séquence de dépendances fonctionnelles allant de A vers B qui suit la séquence des opérations relationnelles. Si la séquence de dépendances fonctionnelles est interrompue, le chemin est alors invalide.

Exemple de chemin valide

Soit le graphe de jointures suivant correspondant à la relation R_m et les DF définies sur la relation de médiation et sur les relations sources:



La recherche des chemins dans le graphe d'opérations identifie un chemin candidat C_{Rm} :

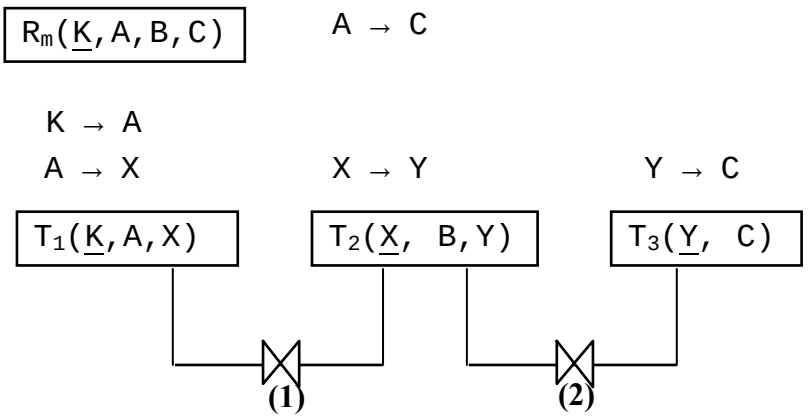
$C_{Rm} = \{(T_1, T_2, \bowtie), (T_2, T_3, \bowtie)\}$ au calcul de la relation R_m. Selon la définition d'un chemin valide, C_{Rm} préserve la DF définie sur la relation de médiation $A \rightarrow C$ car il existe une séquence de dépendances fonctionnelles allant de A vers $C (A \rightarrow X, X \rightarrow Y, Y \rightarrow C)$ qui suit la séquence des opérations relationnelles.

Considérons les règles d'invalidation suivantes qui permettent d'identifier les chemins de calcul invalides au cours du processus de génération :

Notation	Explication
- R_m	La relation de médiation
- $A \rightarrow B$	La dépendance fonctionnelle définie sur R_m
- $Q_{Rm} = T_1 \bowtie \ldots \bowtie T_i \bowtie \ldots T_j \ldots \bowtie \ldots T_n$	L'expression de jointures
- T_i, T_j	Les relations de mapping
- K_i, K_j	Les attributs de jointure dans T_i et T_j
- A	Un attribut de T_i
- B	Un attribut de T_j

Règle d'invalidation 1

Soit la dépendance fonctionnelle $A \rightarrow C$ définie sur la relation de médiation R_m et soient les dépendances fonctionnelles $A \rightarrow X$, $X \rightarrow Y$ et $Y \rightarrow C$ définies respectivement sur les relations sources T_1, T_2 et T_3



Supposons que la DF $A \rightarrow X$ définie sur la relation T_1 soit violée c'est-à-dire que pour une occurrence de A il existe plusieurs occurrences de X. L'attribut A ne déterminant pas l'attribut X qui est l'attribut de jointure des deux relations T_1 et T_2 il n'y a donc pas de séquence de DF allant de A vers C qui suit la séquence de jointures dans le graphe.

Le chemin $C_{Rm} = \{(T_1, T_2, \bowtie), (T_2, T_3, \bowtie)\}$ est dit un chemin non valide car la DF $A \rightarrow C$ définie sur la relation R_m n'est pas préservée.

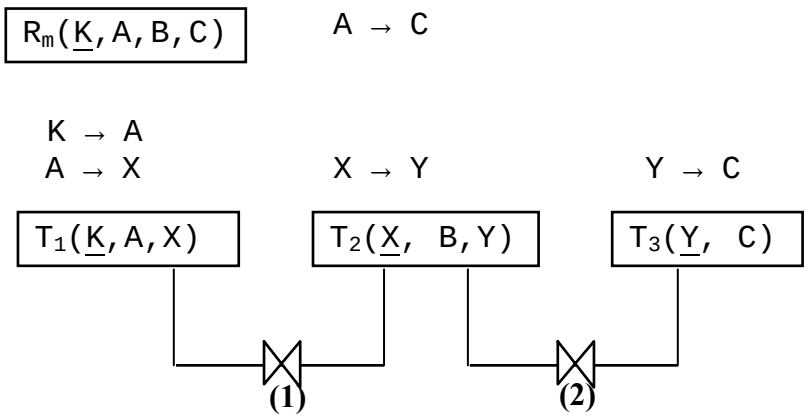
Plus formellement : Si la partie gauche de la dépendance fonctionnelle $A \rightarrow C$ définie sur la relation de médiation R_m ne détermine pas l'attribut de jointure K_i dans la relation de mapping T_i alors la dépendance fonctionnelle $A \rightarrow C$ n'est pas préservée par l'expression de jointures Q_{Rm} et ainsi le chemin est dit invalide.

Règle d'invalidation 1 :

Si $A \not\rightarrow K_i$ dans T_i
Alors $A \not\rightarrow C$

Règle d'invalidation 2

Soit la dépendance fonctionnelle $A \rightarrow C$ définie sur la relation de médiation R_m :



De même si la DF $Y \rightarrow C$ est violée il n'y a donc pas de séquence de DF allant de A vers C qui suit la séquence de jointures. Le chemin $C_{Rm} = \{(T_1, T_2, \bowtie), (T_2, T_3, \bowtie)\}$ est dans ce cas de figure également invalide.

Plus formellement : si l'attribut de jointure K_j dans la relation de mapping T_j ne détermine pas la partie droite de la dépendance fonctionnelle $A \rightarrow C$ définie sur la relation de médiation R_m alors la dépendance fonctionnelle $A \rightarrow C$ n'est pas préservée par l'expression de jointures Q_{Rm} et ainsi le chemin est dit invalide.

Règle d'invalidation 2 :

Si $K_j \not\rightarrow C$ dans T_j
Alors $A \not\rightarrow C$

Règle d'invalidation 3

Dans le cas où la partie gauche de la dépendance fonctionnelle $A \rightarrow C$ définie sur la relation de médiation R_m est trouvée pour chaque jointure entre T_i et T_{i+1} sur le critère $T_i.k_i = T_{i+1}.K_i$ si l'attribut de jointure K_i n'est pas clé primaire dans T_{i+1} ou si K_i dans T_{i+1} ne détermine pas l'attribut de jointure K_{i+1} entre T_{i+1} et T_{i+2} alors la dépendance fonctionnelle $A \rightarrow C$ définie sur la relation de médiation R_m n'est pas préservée par l'expression de jointures Q_{Rm} et ainsi le chemin est dit invalide.

Règle d'invalidation 3 :

Si $A \rightarrow K_i$ dans T_i
Et K_i n'est pas clé primaire dans T_{i+1}
Ou $K_i \not\rightarrow K_{i+1}$ dans T_{i+1}
Alors $A \not\rightarrow C$

Pour le cas où la partie droite de la dépendance fonctionnelle $A \rightarrow B$ définie sur la relation de médiation R_m est trouvée pour chaque jointure entre T_i et T_{i+1} sur le critère $T_i.k_i = T_{i+1}.K_i$ si

l'attribut de jointure K_i n'est pas clé primaire dans T_i ou si K_i dans T_i ne détermine pas l'attribut de jointure K_{i-1} entre T_{i-1} et T_i alors la dépendance fonctionnelle $A \rightarrow B$ définie sur la relation de médiation R_m n'est pas préservée par l'expression de jointures Q_{Rm} et ainsi le chemin est dit invalide.

```
Règle d'invalidation 4 :

Si Kj→ B dans Tj
Et Ki n'est pas clé primaire dans Ti
Ou Ki ↛ Ki-1 dans Ti
Alors A ↛ B
```

Compte tenu de ces règles d'invalidation d'un chemin de calcul nous présentons ci-dessous la nouvelle version de l'algorithme de recherche des chemins revisité intégrant une procédure qui consiste à valider un chemin de calcul en cours d'énumération. La spécification détaillée de cet algorithme est décrite en annexe 8.

```
RechercheCheminValide (XCourant, X, GRM, DF, ChemCourant, ChemTrouvé)

   Entrée : XCourant : l'ensemble des attributs du chemin courant
            X : l'ensemble des attributs de la relation Rm.
            GRM : le graphe d'opérations
            DF : ensemble des DF définies sur Rm et sur les sources
            ChemCourant : le chemin de jointures courant
   Sortie : ChemTrouvé : l'ensemble de chemins de jointures trouvées

   1. TantQue le chemin courant ne contient pas tous les attributs de Rm
   2.   Pour toute jointure j ∈ GRM
   3.     RechercheCheminValide (XCourant, X, ChemCourant ‖ j)
          // Procédure récursive qui cherche toutes les séquences de jointures
          dans J validant l'ensemble des DF
   4.  FinPour
   5. FinTantQue
   6.    ChemTrouvé : = ChemTrouvé  ∪ ChemCourant
          //création d'un chemin de calcul valide
   FinRechercheChemin
```

Algorithme 8 : Algorithme de Recherche des chemins de calcul

Dans cette section nous avons pris en compte les contraintes d'intégrité définies sur le schéma global et sur chaque schéma local lors de la définition des requêtes de médiation.

Par rapport à l'algorithme initial sur la recherche des chemins présenté en section 4 cette nouvelle version de l'algorithme permet de détecter pendant l'exploration les chemins invalides et de ne pas les considérer comme des chemins candidats au calcul de la relation de médiation.

3-6- Conclusion

Le problème de définition de requêtes de médiation dans un système de médiation est un problème complexe en raison du grand nombre de sources de données qui peuvent intervenir mais aussi des méta-données qui les décrivent. La génération automatique de requêtes de médiation est une réponse à ce problème, elle est très utile dans des systèmes de médiation à grande échelle où la connaissance des sources de données est très difficile à maîtriser et à exploiter manuellement. Par ailleurs, dans de tels systèmes où les sources de données sont autonomes, le concepteur du système de médiation doit nécessairement explorer toutes les sources potentielles et procéder à un choix judicieux.

Dans ce chapitre nous avons proposé dans une approche GAV(Global As View) pour le contexte relationnel un algorithme qui, à partir du schéma d'une relation de médiation, de contraintes d'intégrité définies sur cette relation et de correspondances linguistiques existant entre les sources et le schéma de médiation, produit un ensemble potentiel de requêtes calculant cette relation.

La prise en compte des contraintes d'intégrité pendant le processus de génération permet d'améliorer la qualité des résultats de la génération de requêtes. Nous avons proposé une version de l'algorithme de recherche de chemins qui permet de détecter le plutôt possible la non validité des requêtes et d'éviter l'énumération de toutes les requêtes parmi lesquelles l'utilisateur doit sélectionner les requêtes qui préservent les contraintes d'intégrité définies sur le schéma global.

La complexité du processus de génération de requêtes est accrue lorsqu'on tient compte de l'hétérogénéité des données. Dans ce chapitre nous nous sommes intéressé exclusivement aux conflits d'hétérogénéité liés au schéma qui proviennent du vocabulaire différent utilisé pour décrire le même concept (ex. prix et prix-produit). On a supposé les conflits d'hétérogénéité liés aux données résolus (ex. différence d'unité de mesure euros et francs).

Nous adressons dans le prochain chapitre la problématique liée à un environnement hétérogène. Notre objectif est d'étendre le processus de génération de requêtes de médiation avec prise en compte de l'hétérogénéité des données.

Chapitre 4- Prise en compte de l'hétérogénéité dans la génération de requêtes de médiation

4-1- Introduction

Un système de médiation est un système multi-source qui permet d'interroger un ensemble de sources de données distribuées et hétérogènes. L'une des principales difficultés rencontrées dans un système de médiation est la définition de requêtes de médiation calculant une instance du schéma global.

L'écriture manuelle des requêtes de médiation donne sans doute le résultat le plus pertinent au regard des besoins des utilisateurs. Cependant il est difficile de l'envisager pour la définition de requêtes de médiation en raison du grand nombre de sources de données qui peuvent être impliquées (des centaines ou des milliers) et du volume important de méta-données les décrivant (description des schémas des sources et du schéma global, assertions de correspondance linguistique, assertions intra-source et inter-source, etc.).

Pour résoudre ce problème nous avons présenté dans le chapitre 3 un algorithme pour la génération automatique de requêtes de médiation qui, à partir du schéma d'une relation de médiation, de dépendances fonctionnelles définies sur cette relation, d'assertions de correspondance linguistique existant entre les sources et le schéma de médiation, et d'assertions intra-source et inter-source reliant le relations sources entre elles, produit un ensemble potentiel de requêtes calculant cette relation. Les étapes suivantes résument le principe du processus de la génération de requêtes de médiation :

1) Identification des relations sources pertinentes pour la définition d'une requête de médiation Q du schéma de médiation, et génération de relations de mapping T_i qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation R_m.

2) Identification des opérations relationnelles possibles entre les relations de mapping T_i en fonction de leur schéma et de leurs clés, et génération du graphe d'opérations.

3) Recherche des chemins de calcul à partir du graphe d'opérations pour calculer la relation de médiation R_m.

4) Génération de requêtes de médiation déduites à partir des chemins de calcul de la relation de médiation R_m.

Cependant dans un environnement hétérogène, cet algorithme a des limites car il ne tient pas compte des conflits liés à l'hétérogénéité des données. Il les suppose résolus.

Les conflits liés à l'hétérogénéité des données sont présents lorsque les sources contiennent des erreurs et des incohérences (erreurs de saisie, redondances, violation de contraintes d'intégrité, valeurs contradictoires,…). Ces conflits présents dans une source se multiplient lorsqu'il s'agit d'intégrer des données depuis plusieurs sources dans un système d'informations global. En plus

des problèmes que peut contenir chaque source indépendamment des autres, chaque donnée peut être représentée différemment dans les sources. La cause principale de cette hétérogénéité entre les données est la provenance des données de diverses origines, leurs saisies à des moments distincts par plusieurs personnes qui utilisant des conventions différentes. La question capitale est de savoir quelles sont les données qui correspondent à la même entité du monde réel ?
La non prise en compte des conflits liés à l'hétérogénéité des données peut fausser la sémantique des requêtes de médiation et retourner à l'utilisateur des résultats non conformes à ceux définis dans son schéma de médiation, ou encore empêcher leur exécution. Les opérations qui composent une requête de médiation ne sont valides que si les conflits liés à l'hétérogénéité des données sont détectés et résolus. Par exemple lorsque les valeurs dans une source de données n'ont pas la même précision que celle définie dans le schéma de médiation, ou encore une jointure de deux relations source sur l'attribut prix peut retourner un résultat incorrect quand les prix sont exprimés dans des monnaies différentes (ex. euros et francs). La transformation préalable des monnaies est une solution possible au problème ; encore faut–il savoir détecter les conflits liés à l'hétérogénéité des données et identifier les fonctions de transformations appropriées.

Lors du processus de génération de requêtes de médiation deux types de conflits liés à l'hétérogénéité des sources sont distingués à savoir :

- les conflits sémantiques liés au schéma,
- les conflits sémantiques liés aux données.

Les conflits sémantiques liés au schéma

L'utilisation d'une terminologie différente pour désigner deux concepts identiques entraîne la présence de conflits sémantiques liés au schéma. Par exemple dans la relation de médiation **produit**(Num-produit,désignation,***prix***)et dans la relation **produit** (Num-produit, désignation, ***prix-produit***), les attributs prix et prix-produit ont deux terminologies différentes mais une sémantique identique.

Dans le chapitre précèdent, ce type de conflits lié à l'utilisation d'un vocabulaire différent est pris en compte par l'algorithme de génération de requêtes de médiation. Il exploite des assertions linguistiques déterminées entre un attribut de médiation et un attribut source, et entre deux attributs sources dans le cas où ils appartiennent à des sources différentes.

Nous avons dit que ces assertions sont ajoutées automatiquement dans la base de connaissances, au fur et à mesure de leur découverte au cours du processus de génération de requêtes, sans préciser la technique utilisée pour découvrir ces assertions.

En réponse à cette question, nous présentons dans ce chapitre des techniques évoquées dans le contexte de l'intégration de schéma [Rahm et Bernstein 01] présentées dans le chapitre 2 (état de l'art), et qui peuvent être utilisées dans notre contexte d'interrogation de sources hétérogènes et distribuées.

L'utilisation d'une ontologie [Moulton et al. 02] ou de dictionnaires linguistiques tels que WordNet, EDR, etc est une réponse possible qui permet de détecter et de résoudre les conflits

sémantiques liés au schéma. Elles permettent d'apporter des stratégies de désambiguïsation aux problèmes de synonymie, d'homonymie, etc.

Les conflits sémantiques liés aux données

La provenance de données de diverses origines, leur saisie à des moments distincts par des personnes différentes qui n'ont pas la même perception du réel, et qui utilisent des conventions différentes entraîne ce type de conflits. Par exemple, différence d'unité de mesure, de précision, d'échelle, de format de date, etc.

Lorsqu'on ne tient pas compte de ces problèmes sémantiques au niveau des données il est possible que la sémantique des résultats des requêtes de médiation soit faussée. Les requêtes générées ne peuvent s'exécuter. Par exemple : Le résultat d'une jointure de deux relations sources R_1 et R_2 sur le critère d'égalité $R_1.prix = R_2.prix$ retourne une valeur incorrecte ou nulle quand les prix sont exprimés dans des monnaies différentes dans les relations R_1 et R_2.

Le but de notre approche est d'étendre l'algorithme de génération de requêtes de médiation pour qu'il prenne en compte ces conflits sémantiques liés aux données de façon à retourner à l'utilisateur des résultats conformes à ceux définis dans son schéma de médiation. Cela revient donc à revisiter chaque étape de l'algorithme de génération de requêtes de médiation.

L'idée de base de cette approche pour résoudre ces conflits sémantiques liés aux données est d'introduire la notion de type étendu d'un attribut. En plus de son nom et de son type de base un attribut est décrit par un ensemble de méta-données qui décrivent sa valeur.

Nous présentons ci-dessous un exemple simple pour faciliter la compréhension de notre approche.

Exemple

Pour notre exemple nous considérons la relation source `Véhicule` de la source `S1` et la relation source `Personne` de la source `S2`,et une relation de médiation `Vehicule_Personne` de schéma(`Num,Marque,Puissance,Couleur,D_Achat,Prix_Veh,Nom_Cond,Prénom,Adresse,Date_Nais`), nous supposons que l'utilisateur requière un résultat où les prix des véhicules sont exprimés en euros

Véhicule

Num	Marque	Puissance	Couleur	D_Achat	Prix	Nom-Cond
902FEE75	Renault	7	Rouge	2000	6500	Dupond
434FR92	Renault	5	Noir	1998	10000	Martin
5647GT49	Peugeot	5	Vert	1996	8500	Martin

Personne

Nom_Pers	Prénom	Adresse	Date_Nais
Dupond	Paul	Paris	12/02/78
Dupont	Jacques	Paris	04/04/72
Martin	Georges	Lyon	31/01/72
Durand	Olivier	Dijon	01/04/67
Toule	Ange	Marseille	16/01/70

Nous supposons aussi que les correspondances linguistiques suivantes sont déterminées au cours du processus de génération de requêtes de médiation :

- `Véhicule.Prix` ≅ `Vehicule_Personne.Prix-Veh`
- `Véhicule.Nom-Pers` ≅ `Personne.Nom-Cond`

Soit la requête Q suivante générée par l'algorithme de génération de requêtes de médiation qui calcule la relation de médiation `Vehicule_Personne` à partir des relations sources personne et véhicule :

```
Select *
From Véhicule V, Personne P
Where V.Nom-Cond = P.Nom-Pers
And Adresse = paris
And Prix_Veh > 1500
```

La requête générée retourne un seul tuple :

(902FEE75,Renault,7,Rouge,2000 ,**6500**,Dupond,Paul,Paris, 12/02/78)

Ce résultat est incorrect car les prix des véhicules dans la relation `véhicule` sont exprimés en francs alors que l'utilisateur requière une vue où les prix sont exprimés en euros. Cette information de monnaie n'étant spécifiée ni dans le schéma de la relation de médiation `vehicule_personne` ni dans celui de la relation source `Véhicule` entraîne des conflits sémantiques lié aux données.

Pour résoudre la problématique de la définition de requêtes de médiation au regard d'un grand nombre de sources hétérogènes et distribuées et du volume important de méta-données qui les décrivent, le but de notre approche est d'étendre l'algorithme de génération de requêtes de médiation pour qu'il prenne en compte les conflits liés à l'hétérogénéité des sources, et retourner à l'utilisateur des résultats conformes à ceux définis dans son schéma de médiation.

Suite à cette introduction, nous présentons dans ce chapitre tout d'abord les méta-données supplémentaires utilisées pour la génération de requêtes dans un environnement hétérogène, leur exploitation dans le processus de génération de requêtes, ensuite nous présentons le nouveau principe de l'algorithme de génération de requêtes de médiation qui tient compte des conflits liés à l'hétérogénéité des données, et enfin nous présentons quelques perspectives pour guider une approche de génération plus approfondie lorsque des conflits liés à l'hétérogénéité des données ont besoin de traitements plus complexes .

4-2- Les méta-données utilisées

Etendre le processus de génération de requêtes pour qu'il prenne en compte les conflits liés à l'hétérogénéité des sources revient tout d'abord à étendre la base de connaissances en termes de méta-données. En effet avoir une bonne connaissance du schéma global et de chaque schéma local est l'idée de base de notre approche pour gérer l'hétérogénéité des sources.

Nous avons distingué deux types de conflits liés à l'hétérogénéité lors du processus de génération de requêtes de médiation : 1) les conflits sémantiques liés au schéma dus à l'utilisation d'une terminologie différente pour décrire un même concept (ex. `Nom_Cond` et `Nom_Pers` ; et 2) les conflits sémantiques liés aux données dus à la provenance de données de diverses origines, leur saisie à des moments distincts par des personnes différentes qui n'ont pas la même perception du réel, et qui utilisent des conventions différentes. Par exemple, différence d'unité de mesure pour l'attribut `prix_veh` (euros et francs). Pour remédier à cette problématique le processus de génération de requêtes de médiation a besoin de méta-données supplémentaires dans la base de connaissances initiale.

La base de connaissances exploitée par l'algorithme de génération automatique de requêtes proposé dans le chapitre 3 est constituée de : 1) la description du schéma de médiation comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations ; 2) la description du schéma local à chaque source de données comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations d'une même source; 3) des correspondances linguistiques entre les concepts de sources différentes, et des correspondances linguistiques entre les concepts sources et les concepts du schéma de médiation (synonymie, abréviations, inclusions, et équivalences linguistiques des noms des attributs). Cette dernière catégorie de méta-données est utilisée par l'algorithme pour gérer les conflits sémantiques liés au schéma qui peuvent exister entre un attribut de médiation et un attribut source, ou entre deux attributs de sources différentes.

Notre idée de base pour étendre le processus de génération automatique de requêtes de médiation est d'une part utiliser une des techniques évoquées dans le contexte d'intégration de schéma [Rahm et Bernstein 01] telle que : une ontologie ou un dictionnaire linguistique pour identifier les correspondances linguistiques entre deux concepts et ainsi remédier aux conflits sémantiques liés au schéma, et d'autre part introduire la notion du type étendu d'un attribut pour détecter les conflits sémantiques liés aux données et d'utiliser les fonctions de transformations pour résoudre ces conflits.

La nouvelle approche que nous proposons pour la génération de requêtes de médiation dans un environnement hétérogène est fondée sur les hypothèses suivantes :

En plus des méta-données décrites dans la base de connaissances initiale il existe :

- Un dictionnaire linguistique automatique pour détecter et résoudre automatiquement les conflits liés au schéma,
- Un type étendu d'un attribut pour détecter les conflits sémantiques liés aux données,
- Une librairie de fonctions de transformations pour transformer les données hétérogènes et garantir leur conformité mutuelle et leur conformité par rapport au schéma global.

Le dictionnaire linguistique

Dans le contexte de la génération de requêtes de médiation, l'utilisation d'un dictionnaire linguistique ou d'une ontologie est une solution possible aux problèmes de synonymie, d'homonymie,…etc. Ils permettent de détecter automatiquement les assertions linguistiques et remédier aux conflits sémantiques liés au schéma qui existent entre deux attributs particuliers.Par exemple l'utilisation d'une ontologie qui est une hiérarchie de concepts modélisant un domaine particulier, ou encore l'utilisation d'un dictionnaire linguistique spécialisé permet d'identifier la correspondance linguistique qui existe entre l'attribut `Prix` de la relation `Véhicule` et l'attribut `Prix_Veh` dans la relation de médiation. Parmi les techniques utilisées dans la littérature pour gérer les conflits sémantiques liés au schéma dans le contexte de l'intégration en général, nous distinguons le dictionnaire automatique linguistique WordNet. Dans ce qui suit nous rappelons sa définition suivie d'un exemple.

Définition de WORDNET : WORDNET est une base de données lexicographique de l'anglais, développée depuis une quinzaine d'années à l'université de Princeton, sous la direction du Professeur George A.Miller. Cette base contient les catégories grammaticales suivantes : noms, verbes, adjectifs et adverbes à l'exclusion des mots grammaticaux. Elles sont organisées selon leur sens et non leur forme, ce qui importe n'est pas la définition du mot mais de situer celui-ci dans un réseau sémantique ou lexical (hiérarchique) de mots, de ce point de vue WordNet ressemble plus à un thesaurus qu'à un dictionnaire [Fellbaum 98].
La notion de sens dans WordNet est représentée par un ensemble de synonymes (synset), accompagné d'une courte définition du mot.

Exemple1 : pour le nom « leg » la base lexicale contient neuf sens (synsets) :

1- **leg** –(a human limb, commonly used to refer to a whole limb but technically only the part between the knee and ankle)
2- **leg** –(a structure in animals that is similar to a human leg and used for locomotion)
3- **leg** –(one of the supports for a piece of furniture)
4- branch, fork, **leg** – (a part of a forked or branching shape, « he broke off one of the branches », « they took the south fork »)
5- **leg** –(the limb of an animal used for food)
6- Peg, wooden leg**, leg**, pegleg – (a prothesis that replaces a missing leg)
7- **leg** – (the part of garment that covers the leg)
8- **leg**—((nautical) the distance traveled by a sailing vessel on a single tack)
9- stage, **leg**—(a section or portion of a journey or course, « then we embarked on the second stage of our carribean cruise »)

Cette notion de sens est complétée par les liens qu'entretiennent les synsets avec d'autres synsets du lexique. Ils sont reliés entre eux par des relations de : synonymie, hyponymie, hyperonymie, antonymie, meronymie.

Exemple 2 :

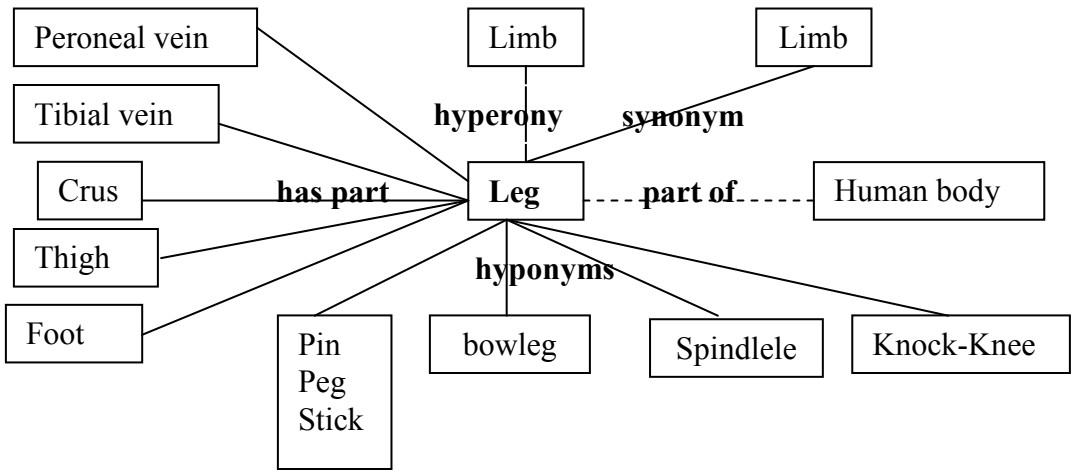


Figure 1 : Exemple de réseau pour le premier sens de Leg

Le projet Européen EuroWordnet développé ces dernières années, a permis d'étendre le réseau lexical WordNet en définissant une base commune de concepts, dans plusieurs langues y compris le Français et ainsi construire une base de données multilingues qui enregistre des relations sémantiques entre les mots d'une langue à une autre, tout en respectant le standard de formalisation introduit par WordNet.

Cette technique est une solution possible parmi d'autres qui peut être intégrée dans notre outil de génération de requêtes de médiation pour identifier automatiquement les correspondances linguistique entre un attribut source et un attribut de médiation, ou encore une correspondance entre deux attributs de sources différentes. La plupart des approches existantes dans le contexte de l'interrogation de sources hétérogènes et distribuées définissent ces assertions manuellement [Miller et al. 00], ce qui est souvent dérisoire en regard d'un grand nombre de sources et du volume important de méta-données qui les décrivent.

Le type étendu

L'idée de base de notre approche pour résoudre la problématique liée à la sémantique des données est d'introduire la notion de type étendu d'un attribut [M.Bouzeghoub et al.02]. En plus de son nom et de son type de base un attribut est décrit par un ensemble de méta-données qui décrivent sa valeur.

Définition du type Etendu : Le type étendu d'un attribut A d'une relation R est défini comme un tableau associatif d'éléments à deux colonnes, où la première colonne décrit le nom de l'élément et la deuxième décrit sa valeur. Il est noté $TE_{(R.A)}$.

Description du type Etendu **:** Le type étendu d'un attribut est décrit de la façon suivante :

<nom-attribut> : [Type = type de base,
Element = { $elem_1$, $elem_2$,……,$elem_n$}]

- Type : est le type de base de l'attribut
- Eléments : un ensemble de méta-données qui décrivent la valeur de l'attribut.

Dans [M.Bouzeghoub et al.02] nous avons présenté un exemple de type étendu d'un attribut :

<nom-attribut> : [Type = <type reconnu>,
{Format = <structure-valeur>,
Unité = <unité-mesure>,
Echelle = <échelle-valeur>,
Précision = <seuil>,
Confiance = <degré> }]

- Type : est l'un des types de base reconnus par les systèmes : entier, réel, booléen, etc.
- Format : représente la structure d'un attribut par exemple le format d'une date, d'une adresse.
- Unité : représente l'unité dans laquelle est exprimée la valeur d'un attribut. Par exemple euros, francs, etc.
- Echelle : représente le niveau de granularité dans lequel sont représentées les valeurs d'un attribut. Par exemple kilo, méga.
- Précision : indique la marge possible d'erreur dans la valeur d'un attribut
- Confiance : définit le degré de confiance que l'on a dans les valeurs d'un attribut, représentant ainsi la possibilité d'erreurs de saisie ou de crédibilité de l'information donnée. Ce degré est défini dans l'intervalle [0,1].

A l'exception du type de base, les autres éléments peuvent prendre des valeurs nulles. Et selon le type d'un attribut certains paramètres peuvent ne pas être renseignés. Par exemple si un attribut est de type caractère les paramètres unité et précision prennent la valeur null.

Exemples

- $TE_{(Rm.prix)}$ [unité] = « euros »
- $TE_{(R.prix)}$ [échelle] = « KF»
- $TE_{(Rm.prix)}$ [précision] = « 0.05% »

Pour remédier à la problématique de l'exemple présenté en introduction et représenter les informations absentes, à savoir la monnaie, nous proposons un type étendu pour l'attribut prix_veh de la relation de médiation R_m noté $TE_{(Rm.prix\_veh)}$ [unité] = euros et un type étendu pour l'attribut prix de la relation véhicule noté $TE_{(vehicule.prix)}$ [unité] = francs, ou l'élément unité décrit l'unité de l'attribut prix.

Les fonctions de transformations

Nous supposons l'existence d'une librairie de fonctions de transformations pour la résolution des conflits sémantiques au niveau des données.

Définition d'une fonction de transformation : Une fonction de transformation transforme la valeur d'un élément du type étendu d'un attribut en une autre valeur.

Description d'une fonction de transformation :

```
<nom-fonction> : [élément  = < élément à traiter >
                  valeur_in = < valeur de l'élément en entrée >
                  valeur_out = <valeur en sortie de l'élément> ]
```

- Elément : l'élément du type étendu d'un attribut à transformer.
- Valeur_in : la valeur de l'élément en entrée de la fonction de transformation.
- Valeur_out : la valeur de l'élément en sortie de la fonction de transformation.

Exemples

- Convert_Unit : [unité, francs, euros]
- Convert_echelle : [échelle, KF, euros]
- Convert_precision : [précision, 0, 0.05%

Nous avons présenté dans cette section l'ensemble des méta-données dont a besoin le processus de génération de requêtes dans un environnement hétérogène.

En résumé la base de connaissances ***A*** est donc constituée de :

- la description du schéma de médiation comportant les schéma de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations, et pour chaque attribut d'une relation son type étendu,
- la description du schéma local à chaque source de données comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes

référentielles entre les relations d'une même source, et pour chaque attribut d'une relation son type étendu.
- Les assertions linguistiques entre les attributs sources et les attributs de médiation (synonymie, abréviations, équivalences terminologiques des noms des attributs), et des assertions linguistiques entre les attributs de sources différentes. Ces assertions sont ajoutées au fur et à mesure de leur identification par le dictionnaire.
- La librairie de fonctions de transformations de données.
- Le dictionnaire linguistique.

Dans ce qui suit nous présentons comment ces connaissances sont exploitées par le processus pour garantir à l'utilisateur les résultats conformes à ces besoins.

4-3- Exploitation des méta-données

Nous supposons le schéma de médiation déjà défini ainsi que l'ensemble des méta-données décrites dans la section précédente. Nous nous plaçons dans une approche descendante (Global As View ou GAV) où chacune des relations du schéma de médiation est définie comme une relation de médiation (une requête) sur les sources de données.

Notre approche présentée ici consiste à revisiter chaque étape principale de l'algorithme de génération de requêtes de médiation afin de détecter et de résoudre les conflits liés à l'hétérogénéité des sources au cours du processus de génération de requêtes de médiation.

Nous proposons trois procédures appelées procédures de nettoyage de données : `Compare`, `CheckType` et `Search` qui exploitent les méta-données précédentes à savoir le dictionnaire linguistique, le type étendu, et la librairie de fonctions de transformations pour l'identification et la résolution des conflits liés à l'hétérogénéité des données. Ces procédures sont appelées au cours de la génération de requêtes de médiation.

Nous présentons tout d'abord le principe et la spécification de chaque procédure, et ensuite nous présentons la nouvelle version des algorithmes revisités à savoir : l'algorithme de recherche de mapping étendu, l'algorithme de recherche de mapping de transition, et l'algorithme de recherche du graphe d'opérations intégrant ces procédures de nettoyage.

La procédure Compare

Principe : elle prend comme paramètres d'entrée un attribut A de la relation de médiation R_m et un attribut B d'une relation source R. Elle teste tout d'abord la correspondance linguistique des attributs A et B en utilisant le dictionnaire linguistique (ligne2). Si A et B sont sémantiquement équivalents ($A \cong B$) elle appelle la procédure `CheckType` (algorithme 2) sinon elle retourne faux (ligne 4).

```
Compare (A, B, CF)
Entrée : A : un attribut de la relation Rm
         B : un attribut de la relation source
Sortie : CF : ensemble de fonctions de transformations
Retour : un booléen

1. CF = ∅
2.    Si A ≅ B
3.    Alors return CheckType (TE (Rm.A), TE(R, B))
4.    Sinon return (faux)
5.    FinSi
FinCompare
```

Algorithme 1: La procédure Compare

Exemple

`Compare(Prix_Veh,Prix,CF)` prend en paramètres d'entrée l'attribut `Prix_Veh` de la relation `Vehicule_Personne` et l'attribut `Prix` de la relation source `Véhicule`, elle retourne l'ensemble `CF` qui contient la fonction `Convert_Unit`.

La procédure CheckType

Principe : elle est appelée pour détecter les conflits sémantiques liés aux données entre les attributs A et B. Elle exploite le type étendu des attributs, elle prend comme paramètres d'entrée le type étendu de l'attribut A et le type étendu de l'attribut B.
Elle compare terme à terme chaque élément e_i du $TE_{(Rm.A)}$ avec chaque élément e_j du $TE_{(R.B)}$ (lignes 4 et 5). Si l'appariement réussit sur l'ensemble des éléments elle retourne un booléen CT qui prend la valeur vraie (ligne 14), et dans ce cas la procédure `Compare` retourne aussi vraie (algorithme 1 ligne 3). Si une des valeurs des éléments du type étendu de A est différente de la valeur de l'élément du type étendu de B alors `CheckType` appelle une autre procédure nommée `Search` (algorithme3) pour résoudre ce conflit. Par exemple si $TE_{(Rm.prix)}$(unité)=euros et $TE_{(vehicule.prix)}$(unité)=francs.

```
CheckType (TE (Rm.A), TE(R,B), CF)
Entrée : TE (Rm.A) : le type étendu de A
         TE(R,B) : le type étendu de B
Sortie : CF : l'ensemble de fonctions de transformations
Retour : un booléen CT

1.   CT = vrai
2.   Pour chaque élément  ei de TE(Rm.A)
3.      Pour chaque élément ej de TE(R,B)
4.       Si ei = ej alors
5.        Si (TE (Rm.A) [ei] ≠ null et  TE(R,B) [ej]≠ null) et
```

Algorithme 2 : la procédure CheckType

Exemple

CheckType ($TE_{(vehicule\_personne.prix\_veh)}$, $TE_{(vehicule,.prix)}$, CF)

La procédure Search

Principe : elle exploite la librairie de fonctions, elle prend comme paramètres d'entrée l'élément e_i à traiter (ex : unité), sa valeur en entrée (ex : francs) et sa valeur en sortie (ex : euros), elle exploite la librairie de fonctions et cherche une fonction de transformation f qui a une valeur de paramètre d'entrée (valeur_in) égale à la valeur de l'élément e_j et qui a une valeur de paramètre de sortie (valeur_out) égale à la valeur de l'élément e_i (ligne 2). Dans notre contexte l'attribut à transformer est l'attribut source `B`. Si cette fonction existe la procédure `Search` retourne la fonction `f` (ligne 3), la procédure `CheckType` retourne vrai (algorithme 2 ligne 14) et l'ensemble de fonctions `CF` (algorithme 2 ligne 7), `Compare` retourne vrai (algorithme 1 ligne 3) et l'ensemble de fonctions de transformation `CF`, sinon `CheckType` et `Compare` retournent faux.

```
Search (e_i,TE_(R.B) [e_j], TE_(Rm.A)[e_i], Lib)
   Entrée :   e_i : l'élément du type étendu à traiter
              TE_(R.B) [e_j]: la valeur de l'élément e_j
              TE_(Rm.A) [e_i]: la valeur de l'élément e_i
              Lib : la librairie de fonctions
   Retour :   la fonction de transformation

   1.  Pour chaque fonction f ∈ Lib
   2.    Si (TE_(R.B) [e_j] = f.valeur_in)and TE_Rm.A) [e_i] = f.valeur_out)
   3.    alors return(f)
   4.    Finsi
   5.  FinPour
FinSearch
```

Algorithme 3 : La procédure Search

Exemple

Search (« unité », « francs », « euros », Lib) prend comme paramètres d'entrée : l'élément unité , la valeur de l'unité dans la relation R (francs), la valeur de l'unité dans la relation R_m, et la librairie de fonctions, elle retourne si elle existe la fonction ConvertUnit : [unité, francs, euros]

Nous avons présenté dans cette section le principe et la spécification des procédures `Compare`, `CheckType` et `Search` qui sont appelées au cours du processus de génération de requêtes pour mettre en conformité les donnés hétérogènes. Dans la section suivante nous proposons les algorithmes de génération de requêtes revisités intégrant ces procédures.

4-4- Le nouveau principe de l'algorithme de génération revisité

En présence d'hétérogénéité, il est difficile d'envisager la génération automatique des requêtes de médiation sans tenir compte des conflits liés au schéma et des conflits liés aux données.

Compte tenu de notre travail de recherche sur l'hétérogénéité des sources dans le contexte de la génération de requêtes de médiation, nous distinguons parmi les algorithmes de génération de requêtes de médiation proposés pour un environnement homogène trois algorithmes à revisiter. Ces algorithmes sont les suivants :

- la recherche des relations des mapping étendu,
- la recherche des relations de transitions,
- la recherche du graphe d'opérations.

Dans un environnement hétérogène, ces algorithmes peuvent retourner des résultats sémantiquement incorrects ou encore empêcher l'exécution des requêtes si les conflits ne sont pas détectés et résolus au cours du processus de génération de requêtes de médiation.

Tout d'abord pour la recherche des relations de mapping étendu, qui consiste à identifier les relations sources pertinentes, et à générer de relations de mapping T_i qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation de médiation R_m. Cet algorithme peut retourner en sortie des relations de mapping incorrects si les conflits liés au schéma et ceux liés aux données ne sont pas détectés et résolus.

Exemple

Soit la relation de médiation vehicule_personne (num, marque, puissance, couleur, d_achat,**prix_veh**,nom_cond,prénom,adresse,date_nais), et la relation vehicule (num, marque,puissance,couleur,d_achat,**prix**,nom_cond).

Si on ne tient pas compte des conflits liés au schéma et aux données le résultat de la recherche des relations de mapping étendu est le suivant :

T1 = (num,marque,puissance,couleur,d_achat,nom_cond).

Ce résultat est incorrect car les attributs prix_veh dans la relation de médiation et l'attribut prix dans la relation source vehicule sont sémantiquement équivalents même si leur terminologie est différente (prix_veh et prix) et leur type étendu sont aussi différents (euros, francs).

Lorsqu'on tient compte de ces conflits le résultat de T1 est le suivant :

T1 = (num, marque,puissance,couleur,d_achat,**F(prix)**,nom_cond), où **F** est la fonction appropriée qui transforme les francs en euros.

Ensuite pour la génération des relations de transition, qui consiste à trouver pour chaque paire de relations de mapping étendu entre lesquelles aucune contrainte référentielle ni assertion inter-source n'est définie, une ou plusieurs relations de transitions pour permettre de relier cette paire de relations de mapping. Dans cette étape nous supposons que les problèmes liés à l'hétérogénéité sont absents lorsque deux relations d'une même source sont reliées par une contrainte référentielle. Cependant lorsque deux relations appartiennent à deux sources différentes, et sont liées par une assertion inter-source les conflits liés au schéma et aux données sont présents. Dans ce cas de figure il est important de résoudre ces conflits afin de générer des relations de transition sémantiquement correctes.

Exemple

Soit la relation de médiation de schéma (X1,X2,X3,X4)et soient les deux relation de mapping T1(X1,X2) et T2(X3,X4). On suppose que ces deux relations sont déduites à partir de sources différentes. Le résultat de recherche des relations de transition est T3 (X2,X3). Les relations de mapping T1, T2, et T3 appartenant à des source différentes il est important de comparer le type étendu des attributs équivalents. Si par exemple le type étendu des attributs T1.X2 et le type étendu de l'attribut T3.X2 est différent le résultat de la recherche des relations de transition est T3(F(X2), X3) ou T1(X1,F(X2)) où F est la fonction de transformation appliquée à l'un ou l'autre des attributs sources. Le choix de l'attribut à transformer est fixé selon la fonction de transformation qui existe dans la librairie de fonctions.

Enfin pour la recherche du graphe d'opérations, qui consiste à identifier les opérations relationnelles possibles entre les relations de mapping T_i. Pour cet algorithme il faut aussi tenir compte de l'hétérogénéité car une opération de jointure n'est valide que si les conflits liés au schéma et aux données sont détectés et résolus. Comme nous l'avons dit pour la recherche des relations de transition seules les relations sources appartenant à des sources différentes sont concernées par ces conflits. Par exemple une jointure de deux relations sources sur l'attribut prix

peut retourner un résultat incorrect quand les prix sont exprimés dans des monnaies différentes (ex. euros et francs). La transformation des euros en francs ou inversement est une solution au conflit.

Dans le chapitre précèdent nous avons présenté les algorithmes de génération de requêtes de médiation sans tenir compte des conflits liés à l'hétérogénéité des données. Dans cette section nous présentons la nouvelle version des algorithmes de génération de requêtes de médiation revisités, intégrant les procédures Compare, CheckType et Search pour remédier aux conflits liés à l'hétérogénéité des données.

La recherche des relations de mapping étendu

Principe : Comme nous l'avons dit en introduction, la première étape de l'algorithme consiste à identifier les relations sources pertinentes au calcul de la relation de médiation R_m et à générer les relations de mapping T_i qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation R_m. A la différence de l'algorithme de recherche de mapping étendu proposé dans le chapitre 3 pour un environnement homogène, celui ci tient compte des conflits liés à l'hétérogénéité. Il appelle les trois procédures précédentes qui exploitent les méta-données supplémentaires à savoir : le dictionnaire, le type étendu des attributs, et les fonctions de transformations.

Pour une relation de médiation donnée R_m, la recherche des relations de mapping étendu s'effectue en considérant successivement les relations de chaque source de données. Pour chaque relation source R_i, chaque attribut B est comparé aux attributs de la relation de médiation en appellent la procédure Compare. La procédure Compare teste tout d'abord la correspondance linguistique des attributs A et B en utilisant le dictionnaire (Algorithme1 ligne2). Si A et B sont sémantiquement équivalents ($A \cong B$) elle ajoute cette correspondance automatiquement dans la base de connaissances et appelle la procédure CheckType (algorithme 2) pour comparer terme à terme chaque élément e_i du $TE_{(Rm.A)}$ avec chaque élément e_j du $TE_{(R.B)}$ (Algorithme2 lignes 4 et 5) sinon elle retourne faux (Algorithme1 ligne 4). Si Compare retourne vraie et que l'ensemble des fonctions de transformations CF est vide alors on ajoute l'attribut B à l'ensemble des attributs communs entre R_m et une relation source R noté E. Si CF n'est pas vide ce qui signifie qu'il existe une ou plusieurs fonctions de transformation (trouvées par la fonction Search) alors on ajoute à E l'attribut B et l'ensemble de fonctions CF qui lui sont appliqués, en sachant que chaque fonction f de CF est appliquée sur un des éléments du type étendu de l'attribut B. Si E est différent de l'ensemble vide alors les clés primaires et étrangères sont recherchées en se basant sur les assertions et sur les clés contenues dans la base de méta-connaissances notée ***A***. La relation de mapping étendu notée T_i est obtenue en effectuant une projection de la relation source sur les attributs communs avec R_m ainsi que l'ensemble des clés déterminées. L'ensemble des relations de mapping étendu associées à une relation de médiation sur l'ensemble des sources S est noté M_e.

Remarque : Lorsque plusieurs fonctions doivent être appliquées sur un attribut B pour que sa valeur soit conforme à celle de A, en d'autres termes lorsque le cardinal de l'ensemble CF est supérieur à un, la question est de savoir dans quel ordre appliquer ces fonctions sur les éléments

du type étendu de B. Dans le cadre de notre exemple si on suppose que l'attribut prix de la relation véhicule et de la relation de médiation est décrit par deux méta-données unité et échelle, et qu'il existe deux fonctions de transformations telle que : *ConvertUnit* et *ConvertEchelle* (retournées par la procédure), l'ordre d'application des fonctions sur les méta-données correspondantes est simple car on sait que la composition de ces fonctions est commutative. Tandis que lorsque la composition des fonctions de transformations n'est pas commutative on a besoin de connaissances supplémentaires prédéfinies par le concepteur du système de médiation pour guider l'application des fonctions. La spécification détaillée de l'algorithme suivant est décrite en annexe 9.

```
Recherche de MappingEtendu (Rm (X), S, Me)
Entrée : Rm (X): la relation de médiation
         S : l'ensemble de sources de données
Sortie : Me : l'ensemble de relations de mapping étendu

1.  Me : = ∅
2. Pour chaque source S dans S
3.  Pour chaque relation source R dans S
4.   //Tester l'équivalence entre chaque attribut de R et les attributs de Rm
      // Et ajout des attributs équivalents dans l'ensemble E
        Si Compare (A, B, CF) = vrai
         Alors E : = E ∪ {(B, CF )}
        FinSi
5.     RechercheCléP (R, K) // recherche de la clé primaire K de R dans A
6.     RechercheCléE (R, B) // recherche des clés étrangères de R dans A
7.      Me = Me ∪{Ti = ΠE R (Y)}// Création d'une relation de mapping étendu Ti
8.   FinPour
9. FinPour
Fin Recherche de MappingEtendu
```

Algorithme 4 : algorithme de recherche de mapping étendu

<u>La recherche des relations de transitions</u>

Principe : Une fois les relations de mapping étendu T_i générées le but est de trouver des opérations relationnelles pour les combiner, principe de l'étape 2 de l'algorithme cité en introduction. Cependant lorsque il n'y pas d'attributs communs entre deux relations de mapping étendu T_i et T_j aucun opérateur relationnel ne peut leur être appliqué. Cela revient donc à

chercher en plus des relations de mapping étendu, d'autres relations sources pertinentes appelées relations de transitions pour lier ces deux relations de mapping.

La recherche de ces relations de mapping de transition pour une relation de médiation particulière consiste à chercher pour chaque paire de relations de mapping étendu (T_i, T_j) entre lesquelles aucune assertion n'est définie, les ***K*** plus courtes séquences d'assertions entre les deux relations sources R_i et R_j (relations qui ont conduit à dériver T_i et T_j).

Nous distinguons deux types d'assertions pouvant lier deux relations sources : 1) les assertions intra-source (contrainte référentielle) définies entre deux relations sources qui appartiennent à la même source de données; 2) les assertions inter-source (correspondances linguistiques) déterminées entre deux relations sources qui appartiennent à deux sources différentes.

Comme nous l'avons dit en introduction lors de la recherche des relations de transition nous supposons que les problèmes liés à l'hétérogénéité sont absents lorsque deux relations d'une même source sont reliées par une assertion intra-source (une contrainte référentielle). Cependant lorsque deux relations appartiennent à deux sources différentes, et sont liées par une assertion inter-source (correspondances linguistiques) les conflits liés au schéma et aux données sont présents. Dans ce cas de figure il est important de tenir compte des problèmes liés au schéma et aux données lors de la recherche des relations de transition afin de générer des relations de transition sémantiquement corrects.

Ainsi pour chaque paire de relations de mapping étendu appartenant à l'ensemble M_e, si l'ensemble de séquences d'assertions trouvée n'est pas vide (ligne 4) alors pour chaque séquence d'assertions `S` appartenant à l'ensemble de séquences d'assertions trouvé, et pour chaque assertion inter-source la procédure `Compare` est appelée. La procédure compare teste tout d'abord la correspondance linguistique des attributs `a.rhs.attr` (attribut à droite de l'assertion et `a.lhs.attr` (attribut à gauche de l'assertion) en se basant sur les assertions de correspondances linguistiques identifiés automatiquement par le dictionnaire. Si `a.rhs.attr` et `a.lhs.attr` sont sémantiquement équivalents elle appelle la procédure `CheckType` (algorithme 2) pour comparer terme à terme chaque élément e_i du type étendu de `a.rhs.attr` avec chaque élément e_j du type étendu de `a.lhs.attr` (Algorithme 2 lignes 4 et 5), sinon elle retourne faux (Algorithme 1 ligne 4). Si Compare retourne vraie et que l'ensemble des fonctions de transformations `CF` est vide alors on passe à la création de la relation de transition. Si `CF` n'est pas vide ce qui signifie qu'il existe une ou plusieurs fonctions de transformation (trouvées par la fonction `Search`) alors on intègre à la relation de transition l'ensemble de fonctions `CF` qui sont appliqués à l'un ou l'autre de ses attributs. La spécification détaillée de l'algorithme suivant est décrite en annexe 10.

```
Recherche de MappingTransition (Me, SeqTrouvée, Mt)
Entrée : SeqTrouvée : l'ensemble de séquences d'assertions
         Me: l'ensemble de relations de mapping étendus
Sortie : Mt: l'ensemble de relations de mapping de transition

1. Mt : = ∅
2. Pour  chaque paire (Ti, Tj) ∈ Me / Ti ∩ Tj = ∅
3.  RechercheSéquenceOpt (Ri, Rj, A, K, SeqTrouvée)
4.  Si SeqTrouvée ≠ ∅
5.      Pour chaque séquence s ∈ SeqTrouvée
6.         Pour chaque assertion a ∈ s
7.          //tester l'équivalence entre les deux attributs de l'assertion a
            //Dans le cas d'une assertion inter-source
              Si Compare (a.rhs.attr, a.lhs.attr, CF)= vrai
```

Algorithme 5 : Algorithme de recherche de mapping de transition

La recherche du graphe d'opérations

Principe : le principe de cette étape est de trouver des opérations relationnelles susceptibles de combiner chaque paire de relations de mapping appartenant à l'ensemble M_e et à l'ensemble M_t.

La recherche du graphe d'opérations s'effectue pour chaque paire de relations de mapping(T_i, T_j) de l'ensemble M_e et M_t entre lesquelles une contrainte référentielle est définie ou une assertion inter-source est identifiée. Une opération relationnelle déterminée entre deux

relations de mapping peut être une opération qui combine une relation de mapping étendu avec une autre relation de mapping étendu, ou une relation de mapping étendu avec une relation de transition, ou encore une relation de transition avec une autre relation de transition.

A la différence de l'environnent homogène, dans un environnement hétérogène il est important de tenir compte des conflits liés à l'hétérogénéité des données lors de la recherche des opérations relationnelles candidates qui combinent chaque paire de relations de mapping. Lorsque deux relations sources d'une même source de données sont liées par une contrainte référentielle (ligne 3), il est évident que les conflits liés à l'hétérogénéité des données ne se présentent pas. Cependant lorsque deux relations sources appartenant à deux sources différentes (ligne5) sont liées par une correspondance linguistique (assertion inter-source) il est important de vérifier qu'il n'existe pas de conflits liés à la sémantique des données entre elles avant de générer une jointure (ligne6). Par exemple une jointure de deux relations sources sur l'attribut prix peut retourner un résultat incorrect quand les prix sont exprimés dans des monnaies différentes (ex. euros et francs). La transformation des euros en francs ou inversement est une solution au conflit.

Ainsi pour chaque paire de relations de mapping de l'ensemble M_e et M_t et appartenant à deux sources différentes la procédure `Compare` est appelée. La procédure `Compare` teste tout d'abord la correspondance linguistique des attributs sur lesquelles portent le critère de jointure (ligne6) en se basant sur le dictionnaire. Si ils sont sémantiquement équivalents elle appelle la procédure `CheckType` (algorithme 2) pour comparer terme à terme chaque élément de leur type étendu (Algorithme2 lignes 4 et 5), sinon elle retourne faux (Algorithme 1 ligne 4). Si `Compare` retourne vraie et que l'ensemble des fonctions de transformations `CF` est vide alors on ajoute la jointure `j` à l'ensemble de jointures `J`. Si `CF` n'est pas vide ce qui signifie qu'il existe une ou plusieurs fonctions de transformation (trouvées par la fonction `Search`) alors on ajoute à `J` la jointure `j` intégrant l'ensemble de fonctions `CF` qui lui sont appliqués.
Dans le cas où une paire de relations de mapping est reliée par une assertion intra-source l'algorithme créé directement une opération de jointure, il n'appelle pas la fonction `Compare` (ligne 3 et 4).

```
Recherche de GrapheOpérations (Me, Mt,  J)
Entrée : Me : l'ensemble de relations de mapping étendus
         Mt : l'ensemble de relations de mapping de transition
Sortie : J : l'ensemble de Jointures

1. J : = ∅
2. Pour  chaque paire (Ti,Tj) dans Me et Mt  tel que Ti ∩ Tj ≠ ∅
3.   Si Ti et Tj ∈ Si alors
4.    J := J ∪ {j = Ti.A ⋈ Tj.A}
      // Création d'une opération de jointure entre Ti et Tj sur le
      critère de jointure Ti.A = Tj.A
5.   Sinon
6.     Si Compare (A, B, CF)= vrai alors
7.       j = Ti.A ⋈ Tj.B
```

Algorithme 6 : Algorithme de recherche du graphe d'opérations

Notre but étant de prendre en compte les besoins des utilisateurs en terme de conformité des résultats avec le schéma global, nous avons proposé dans cette section le nouveau principe des algorithmes de génération de requêtes en présence de données hétérogènes. Les nouveaux algorithmes de génération de requêtes de médiation permettent d'identifier et de résoudre les conflits sémantiques liés au schéma et les conflits sémantiques liés aux données au cours de l'évaluation de requêtes de médiation.

En plus des conflits liés au schéma et les conflits liés aux données qui peuvent subvenir au cours du processus de génération de requêtes de médiation et que nous pouvons résoudre par l'intégration de fonctions de transformations dans les requêtes de médiation, il peut subsister d'autres conflits qui ont besoin de traitement plus complexes. La section suivante introduit ces traitements.

4-5- Autres formes d'hétérogénéité

En plus des conflits liés au schéma et les conflits liés aux données qui peuvent subvenir au cours du processus de génération de requêtes de médiation, et que nous pouvons résoudre par l'intégration de fonctions de transformations dans les requêtes de médiation, il peut subsister d'autres conflits qui ont besoin de traitement plus complexes.

Exemple 1

Soit la relation Véhicule suivante :

Num	Marque	Puissance	Couleur	D_Achat	Prix	Nom-cond
902FEE75	Renault	7	Rouge	2000	6500	Dupond
434FR92	Renault	5	Noir	1998	15000	Martin
5647GT49	Peugeot	5	Vert	1996	8500	Martain
1375FR93	Renault	9	Blanc	2002	4500	Dupand.p

213FR45	Peugeot	7	Jaune	2003	7000	Dupont.paul

On veut fusionner dans cette relation l'ensemble de tuples représentant le même objet, mais avec des variations orthographiques dans le nom ('Dupont', 'Dupand.p' et 'Dupont.paul'), en un seul tuple.

Exemple 2

Soit la relation Publication de schéma (`#Ref, Auteurs`)

Ref	Auteurs
01	Dupond, Martin, Georges
02	Jacques, Martin

A partir de l'attribut Auteurs qui comporte l'ensemble des auteurs d'un article, on veut générer autant de tuples qu'il y a d'auteurs.

Notre approche d'intégration de fonctions dans les requêtes de médiation ne suffit pas à exprimer ces transformations complexes. L'algèbre relationnelle elle même ne permet pas de faire tous les calculs nécessaires à certains schémas de médiation ; de nouveaux traitements sont donc nécessaires.

Nous introduisons ces traitements sous forme de nouveaux opérateurs relationnels fondés sur les contraintes suivantes :

- chaque nouvel opérateur doit être composable avec les opérateurs classiques de l'algèbre relationnel, ce qui implique qu'il ne prend en entrée qu'une ou deux relations (selon qu'il soit unaire ou binaire) et génère en sortie une et une seule relation.

- Chaque opérateur doit pouvoir être transformé en un programme impératif incluant une ou plusieurs requêtes classiques (c'est-à-dire ne mettant en œuvre que les opérateurs classiques de l'algèbre relationnelle).

Dans cette section nous introduisons deux nouveaux opérateurs *Fusion* et *Explode* pour initier une démarche de génération de requêtes de médiation plus approfondie.

Notre approche se limite pour l'instant à la génération de requêtes de médiation intégrant les opérateurs relationnels traditionnels de l'algèbre relationnelle.

L'opérateur de fusion

Définition de L'opérateur de fusion : L'opérateur de fusion de tuples, noté $\succ_{f(\{X\}),\ GB(Y/s(Y,Y)\leq d)}(R)$, est une généralisation de la notion d'agrégat utilisée en SQL. Il partitionne l'ensemble des tuples de R sur les valeurs de Y et applique ensuite à chaque partition la fonction d'agrégat f. Le partitionnement sur les valeurs de Y peut se faire soit sur l'égalité des valeurs (GroupBy classique de SQL), soit sur la similarité des valeurs en utilisant la fonction s et la distance de similarité d. La fonction d'agrégat f est une fonction de mapping N-1 qui transforme un ensemble de valeurs de X en une seule valeur résultat de la fonction. Le schéma de la relation résultat est le même que celui de la relation R à l'exception de l'attribut X qui est remplacé par f. Le nombre de tuples de la relation résultat est égal au nombre de partitions.

Exemple

L'opérateur de fusion peut fusionner dans la relation Véhicule un ensemble de tuples représentant le même objet, mais avec des variations orthographiques dans le nom (par exemple 'Dupont', 'Dupand.p' et 'Dupont.paul') en un seul tuple. Les partitions sont formées sur la similarité des valeurs de l'attribut Nom en utilisant une distance d. Une fois les partitions formées, il faut appliquer la fusion effective (agrégat) des tuples de chaque partition, le choix du tuple résultat étant donné par une fonction *freq* qui choisit le tuple correspondant à l'orthographe la plus fréquente dans la partition : $\succ_{freq(\{Nom\}),GB(Nom/s((Nom,Nom)\leq 0.2}$(Véhicule).

La fonction d'agrégat définit donc la modalité de choix du tuple résultat ; au lieu de la fréquence, on peut choisir le premier tuple dans l'ordre alphabétique ou le tuple le plus récent si des informations sur la date de mise à jour existent.

Le résultat obtenu par l'application de l'opérateur de Fusion sur la relation Véhicule suivant la fonction freq qui choisit le premier tuple dans l'ordre alphabétique est le suivant :

Num	Marque	Puissance	Couleur	D_Achat	Prix	Nom-cond
5647GT49	Peugeot	5	Vert	1996	8500	Martain
1375FR93	Renault	9	Blanc	2002	4500	Dupand.p

L'opérateur Explode

Définition de L'opérateur Explode : L'opérateur d'explosion de tuple, noté $\prec_{f(X)}(R)$, transforme une table R en une autre table en appliquant une fonction de mapping 1-N à chacune des valeurs de l'attribut X. Chaque tuple t_i de R est composé successivement avec les n valeurs produites par f pour former les tuples résultats. Le schéma de la relation résultat est le même que celui de R à l'exception de l'attribut X qui est remplacé par f. Le nombre de tuple de la relation résultat dépend des résultats successifs de la fonction f, mais il est au moins égale au nombre de tuples de la relation R.

Exemple

Dans la relation Publication(#Ref,Auteurs), à partir de l'attribut Auteurs qui comporte l'ensemble des auteurs d'un article, on peut générer autant de tuples qu'il y a d'auteurs:

$\prec_{Expand(Auteurs)}$(Publication).

Le résultat obtenu par l'application de l'opérateur Explode sur la relation Publication est le suivant :

Publication

Ref	Auteurs
01	Dupond
01	Martin
01	Georges
02	Jacques
02	Martin

Le nettoyage et la transformation des données peut être très complexe dans certaines applications ; il est difficile d'affirmer qu'une liste finie d'opérateurs et de fonctions est complète et permet de réaliser l'ensemble des traitements désirés par les applications. La seule façon de prendre en compte ce problème est d'admettre la définition de nouveaux opérateurs spécifiques aux applications. Les opérateurs de fusion et d'explosion de tuples sont des exemples de tels opérateurs. La seule contrainte imposée est la composabilité des nouveaux opérateurs avec les autres opérateurs pour permettre une définition algébrique des requêtes de médiation. Ce problème de complétude a été évoqué dans [Galhardas et al., 2001] ; une approche extensive a été proposée pour la définition de nouveaux programmes de nettoyage comme des workflows d'opérations mais pas comme des requêtes algébriques.

Une implémentation de ces opérateurs a été réalisée dans le cadre d'un projet d'ISTY3. Ces opérateurs ne sont pas encore intégrés par le processus de génération de requêtes de médiation actuel.

Notre approche se limitant pour l'instant à la génération de requêtes de médiation intégrant les opérateurs relationnels traditionnels de l'algèbre relationnelle. Il reste donc à explorer l'intégration de ces nouveaux opérateurs dans les requêtes de médiation.

4-6- Conclusion

Lors de l'utilisation d'un système de médiation, la définition de requêtes de médiation est l'une des taches les plus complexes à effectuer manuellement, surtout lorsque le nombre de sources et le volume de méta-données qui les décrivent sont importants. Cette complexité se multiplie en présence de données hétérogènes dans les sources.

Compte tenu de notre étude sur l'hétérogénéité des données dans le contexte de la génération de requêtes de médiation nous avons distingué deux types de conflits liés à l'hétérogénéité des données lors du processus de requêtes de médiation :1) les conflits sémantiques liés au schéma dus à l'utilisation d'une terminologie différente pour décrire un même concept (*prix* et *prix_veh* ; et 2) les conflits sémantiques liés aux données dus à la provenance de données de diverses origines, leur saisie à des moments distincts par des personnes différentes qui n'ont pas la même perception du réel, et qui utilisent des conventions différentes. Par exemple, différence d'unité de mesure pour les attributs *prix_veh* et *prix* (euros et francs respectivement).

La non prise en compte des conflits liés à l'hétérogénéité des données peut fausser la sémantique des requêtes de médiation et retourner à l'utilisateur des résultats non conformes à ceux définis dans son schéma de médiation. Les opérations qui composent une requête de médiation ne sont

valides que si les conflits liés à l'hétérogénéité des données sont détectés et résolus. Par exemple une jointure de deux relations source sur l'attribut prix peut retourner un résultat incorrect quand les prix sont exprimés dans des monnaies différentes (ex. euros et francs). La transformation préalable des monnaies est une solution possible au problème ; encore faut–il savoir détecter les conflits liés à l'hétérogénéité des données et identifier les fonctions de transformations appropriées.

L'intégration des fonctions de transformations dans les requêtes de médiation permet de résoudre certains conflits sémantiques liés aux données mains ne répond à tous les conflits. L'algèbre relationnelle elle même ne permet pas de faire tous les calculs nécessaires à certains schémas de médiation ; de nouveaux opérateurs sont donc nécessaires.

En réponse à cette problématique nous avons présenté dans ce chapitre une extension du processus de génération de requêtes qui tient compte des conflits liés à l'hétérogénéité des données. Il permet de détecter et de résoudre dans une approche GAV (Global As View) les conflits sémantiques liés au schéma et les conflits sémantiques liés aux données au cours du processus de génération de requêtes de médiation. Pour résoudre les conflits sémantiques liés au schéma, notre idée est d'utiliser un dictionnaire linguistique automatique tel que WordNet pour identifier les correspondances linguistiques entre les attributs de médiation et les attributs sources, et aussi entre les attributs de sources différentes. Ces correspondances sont ajoutées par le processus automatiquement dans la base de connaissances au fur et à mesure de leur découverte. Pour résoudre les conflits sémantiques liés aux données notre idée de base est l'introduction du type étendu d'un attribut. En plus de son type de base et de son nom un attribut est documenté par des éléments correspondants à des méta-données. Le processus compare le type étendu des attributs terme à terme, et cherche dans une librairie de fonctions de transformations une ou plusieurs fonctions de transformations pour mettre en conformité les données entre elles. Nous supposons que cette librairie est fournie au préalable par le concepteur du système de médiation.

Nous avons introduit dans ce chapitre deux nouveaux opérateurs *Explode* et *Fusion* pour initier une démarche de génération de requêtes de médiation plus approfondie mais ne sont pas encore pris en compte par le processus de génération de requêtes de médiation actuel. Il reste donc à explorer l'intégration de ces nouveaux opérateurs dans les requêtes de médiation.

Dans le chapitre suivant nous présentons la description et l'évaluation de notre prototype qui permet la génération de requêtes de médiation dans un contexte relationnel et hétérogène. Il compte deux modules : 1) interface de définition, 2) génération de requêtes de médiation. Toutes les méta-données sont stockées dans la base de méta-connaissances sous Oracle. Il a pour objectif de générer automatiquement des requêtes de médiation en tenant compte des conflits liés à l'hétérogénéité des données. Notre prototype supporte le passage à l'échelle, il est possible de générer dans des temps raisonnables un ensemble de solutions conformes aux besoins des utilisateurs.

Chapitre 5 - Description du prototype de génération de requêtes de médiation

5-1- Introduction

L'ensemble des algorithmes spécifiés dans les chapitres 3 et 4 sont implémentés et forment un outil original de définition de requêtes de médiation pour le contexte relationnel.

Dans [Kostadinov et al. 04] nous avons proposé un système adaptatif d'aide à la génération de requêtes de médiation qui a pour objectif d'une part de générer des requêtes de médiation en tenant compte des conflits liés à l'hétérogénéité des sources, et d'autre part d'adapter les requêtes aux besoins des utilisateurs en termes de qualité. Ce chapitre est centré sur la description et le fonctionnement du processus qui permet de générer automatiquement des requêtes SQL dans un environnement hétérogène.

La première section, décrit son architecture générale. Il compte essentiellement six modules : 1) *interface graphique* ; 2) *recherche des relations de mapping* ; 3) *recherche des relations de transitions* ; 4) *recherche des opérations de jointures* ; 5) *recherche des chemins de calcul* ; 6) *génération de requêtes de médiation*.

La deuxième section, illustre la méta-base exploitée par le processus de génération de requêtes. Elle est constituée de : 1) Méta-données au niveau de la médiation ; 2) Méta-données au niveau des sources ; 3) Méta-données au niveau intermédiaire (entre les sources et la médiation).

Les section 3, 4, 5, 6,7 et 8 présentent respectivement les modules : *recherche des relations de mapping* ; *recherche des relations de transitions* ; *recherche des opérations de jointures* ; *recherche des chemins de calcul* ; et *génération de requêtes de médiation*.

La section 9, montre le fonctionnement de notre outil sur un scénario.

La section 10, présente une conclusion et des extensions sur des travaux en cours de développement.

5-2 – Architecture générale

Dans cette section nous présentons l'architecture générale du système de génération automatique de requêtes de médiation. Il compte essentiellement six modules : 1) *interface graphique* ; 2) *recherche des relations de mapping* ; 3) *recherche des relations de transitions* ; 4) *recherche des opérations de jointures* ; 5) *recherche des chemins de calcul* ; 6) *génération de requêtes de médiation*. Les méta-données nécessaires à la génération sont stockées dans une méta-base. La figure 1 illustre l'architecture du système.

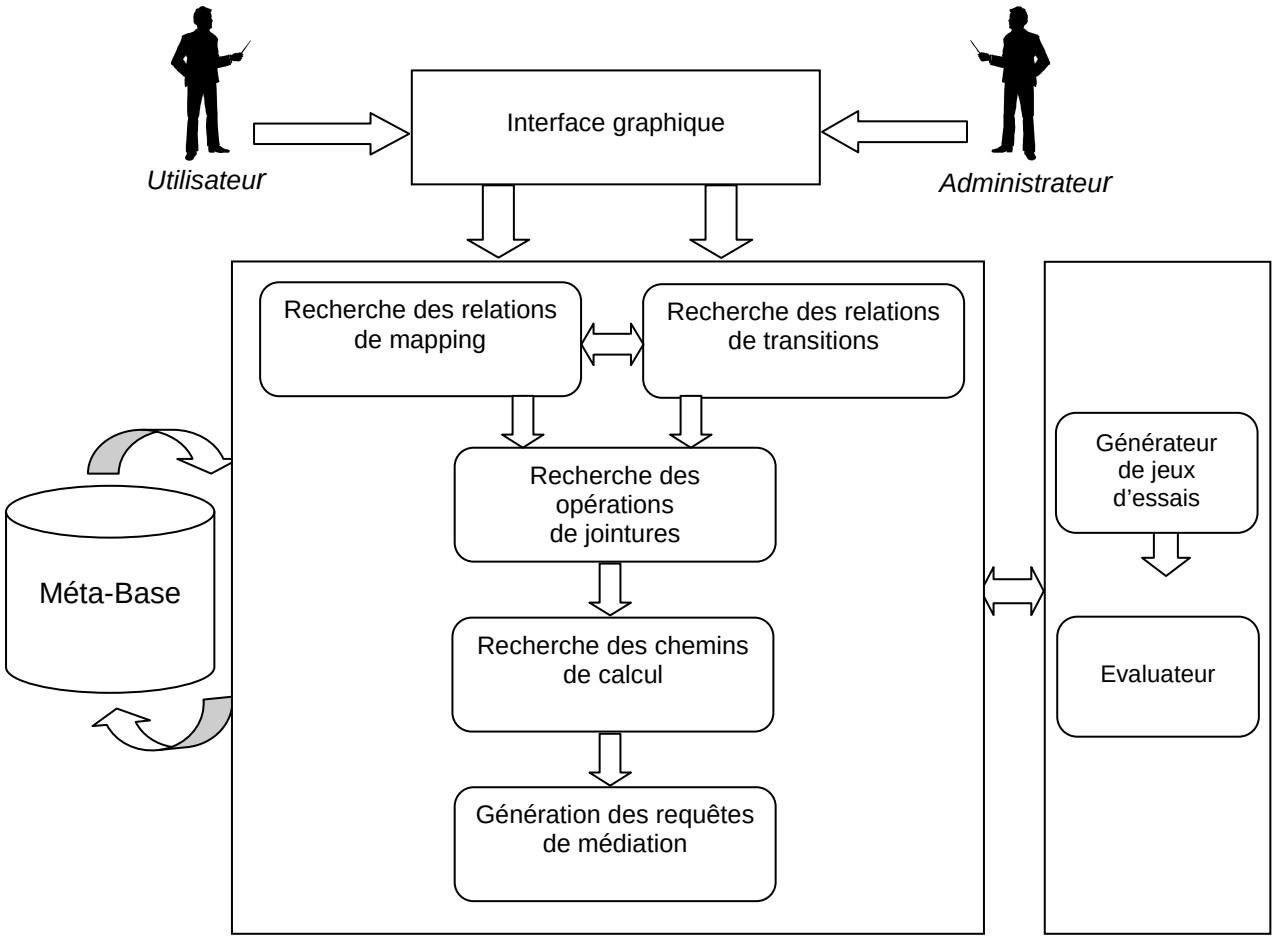


Figure 1 : Architecture du système de génération automatique de requêtes de médiation

L'interface graphique permet de mettre à jour (ajout, suppression et modification) toutes les méta-données utilisées pour la génération de requêtes de médiation, d'interagir avec les cinq autres modules, et de montrer au fur et à mesure les résultats de chaque module du processus de génération. Tous les modules communiquent via la méta-base. Les résultats produits sont également stockés dans cette dernière.

En plus de ces six modules notre outil dispose de deux modules supplémentaires : *générateur de jeux d'essais* et *tests et évaluation. générateur de jeux d'essais* a pour objectif de générer automatiquement des jeux d'essais ; *tests et évaluation* a pour but de tester la conformité et la cohérence des requêtes produites par rapport aux attentes des utilisateurs, et d'évaluer les performances de notre outil pour mesurer son passage à l'échelle. La description et le fonctionnement de cet outil sont développés dans le chapitre suivant.

L'implémentation du prototype de génération automatique de requêtes de médiation a été réalisée en Java (JDK 1.4) et la méta-base est stockée sous oracle.

Les fonctionnalités de chaque module se trouvent dans les classes Java suivantes :

- Pour le module recherche des relations de mapping ses fonctionnalités se trouvent dans la classe *MappingEtendu*,
- Pour le module recherche des relations de transition ses fonctionnalités se trouvent dans la classe *MappingTransition*,
- Pour le module recherche des opérations de jointures ses fonctionnalités se trouvent dans la classe *GrapheOperations*,
- Pour les modules recherche des chemins de calcul et génération des requêtes de médiation leur fonctionnalités se trouvent dans la classe *RechercheChemin*.
- Pour le module de génération de test ses fonctionnalités se trouvent dans la classe *GénérateurTest*.

5-3- Description de la méta-base

Dans cette section nous présentons la description de la méta-base exploitée par le processus de génération de requêtes. Elle est constituée de :

- Méta-données au niveau de la médiation : la description du schéma global (*Schéma_Médiation*) comporte les schéma des relations de médiation, les clés des relations, chaque relation (*Relation_Médiation*) est constituée d'un ensemble d'attributs de médiation (*Attribut_Médiation*), et pour chaque attribut d'une relation son type étendu (*Element_Médiation*).

- Méta-données au niveau des sources : la description du schéma local (*Schéma_Source*) à chaque sources de données comporte les schémas des relations sources, les clés des relations, chaque relation (*Relation_Source*) est constituée d'un ensemble d'attributs source (*Attribut_Source*), et pour chaque attribut d'une relation son type étendu (*Element_Source*).

- Méta-données au niveau intermédiaire : ce niveau décrit les assertions intra-source (*Contrainte_Ref*) définies entre deux attributs d'une même source, les assertion inter-source (*Corresp Ling (S-S)*) définies entre deux attributs de sources différentes, les correspondances linguistiques (*Corresp Ling (S-S)*) définies entre un attribut médiation et un attribut source, et les fonctions de transformations (*Fonction*) qui transforment la valeur d'un élément du type étendu d'un attribut source en une autre valeur.

  Par ailleurs, ce niveau décrit aussi les résultats obtenus au fur et à mesure de la découverte des requêtes de médiation.

  Etant donné une relation de médiation et un ensemble des sources de données, le processus génère tout d'abord une ou plusieurs relations de mapping (*Relation_Mapping*). Chaque relation de mapping est dérivée à partir d'une seule relation source. Une fois les relations de mapping identifiées le système cherche des relations de transition. Pour deux occurrences de relations de mapping il peut exister une ou plusieurs relations de transitions (*Relation_Transition)*. Une relation de transition peut concerner soit deux relations de mapping, deux relations de transitions, ou une relation de transition et une relation de mapping.
  Le processus cherche ensuite les opérations de jointures candidates pour chaque paire de relations. Une *Jointure_Mapping* combine deux relations de mapping, et une *Jointure_Transition* combine soit une relation de mapping avec une relation de transition, ou deux relations de transition.
  A partir du graphe d'opérations constitué des opérations de jointures, le processus cherche un ou plusieurs chemins (*Chemin_Calcul*) au calcul de la relation de médiation.

La figure 2 montre le méta-modéle qui décrit toutes le méta-données nécessaires à la génération de requêtes de médiation :

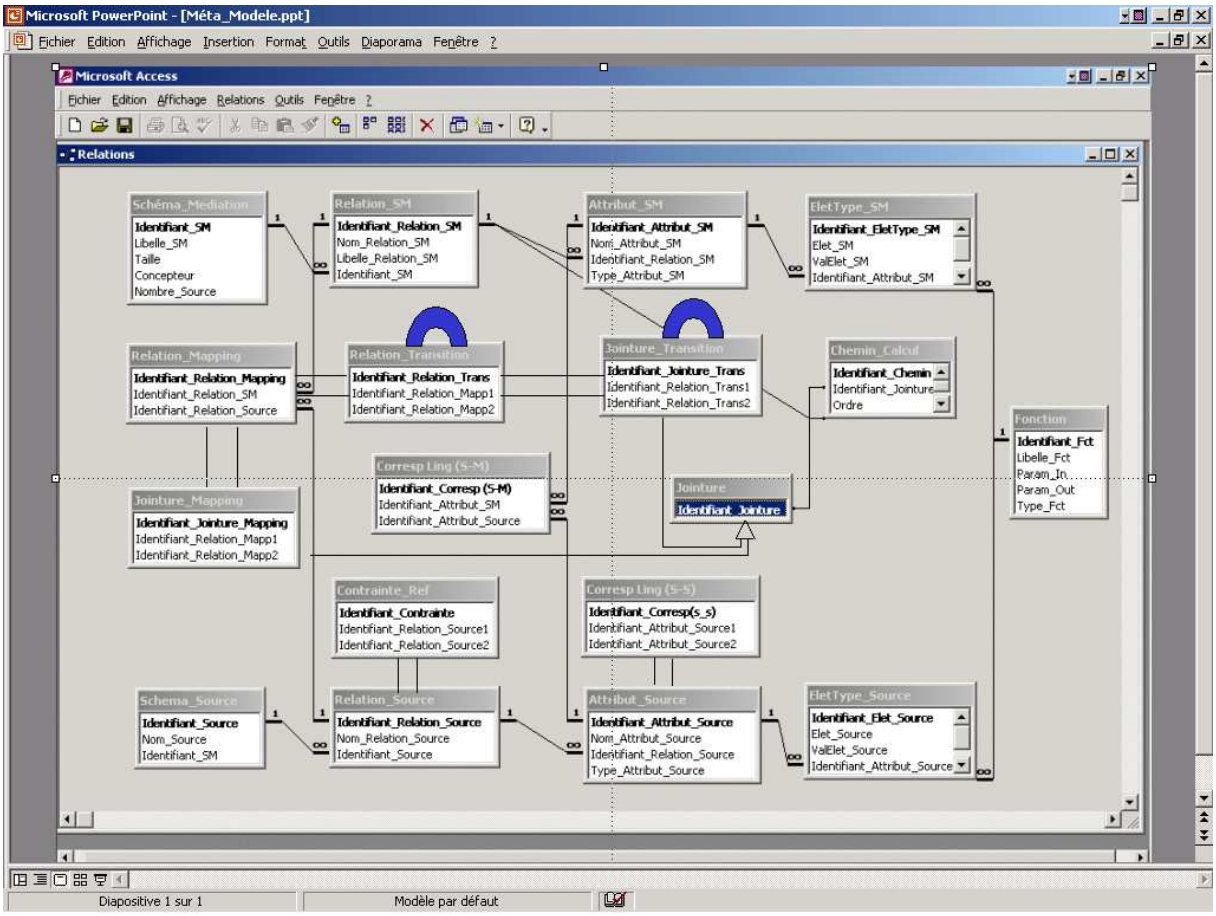


Figure 2 : la description du Méta-Modéle

5-4- La recherche des relations de mapping

Le module de recherche des relations de mapping a pour objectif d'identifier les relations sources pertinentes au calcul de la relation de médiation, et de générer en sortie les relations de mapping étendu.

Les fonctionnalités de ce module se trouvent dans la classe *MappingEtendu*. Via un accès JDBC à la méta-base, il récupère les méta-données relatives à un schéma de médiation des tables suivantes:
Schéma_Source, Relation_Source, Attribut_Source, Attribut_Médiation, Element_Source et Element_Médiation.

Pour prendre en compte l'hétérogénéité des sources, il appelle la classe *Compare* qui a pour objectif de mettre en conformité les données hétérogènes par rapport au schéma global. Cette classe prend en paramètres d'entrée un attribut de médiation et un attribut source, et pour chaque attribut son type étendu, elle compare tout d'abord l'équivalence linguistique entre les deux attributs, et ensuite elle compare terme à terme chaque élément du type étendu de l'attribut de médiation avec chaque élément du type étendu de l'attribut source. Elle accède à la table Fonction de la méta-base par la classe *connexion*, elle retourne en paramètre de sortie un ensemble de fonctions de transformations `CF`.

La classe *MappingEtendu* calcule les relations de mapping étendue et génère en paramètre de sortie l'ensemble de relations de mapping étendu `Me`, elle stocke les résultats produits dans la table temporaire Mapping_Etendu. Les résultats de la recherche des relations de mapping étendu sont affichés par le module d'interface graphique. Le diagramme de classe suivant montre les fonctionnalités de ce module.

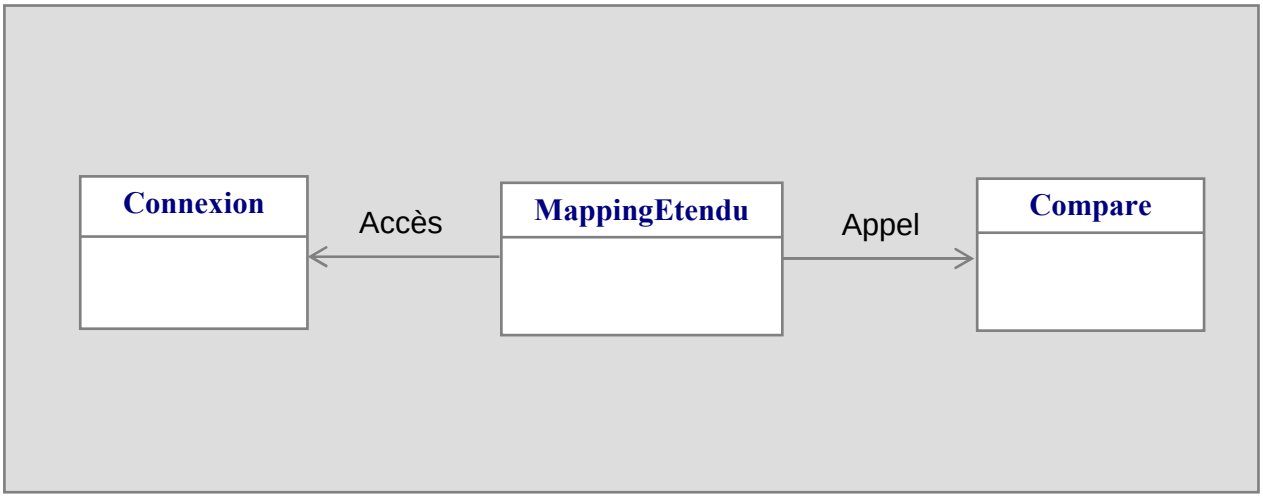


Diagramme 1 : La recherche des relations de mapping

5-5- La recherche des relations de transition

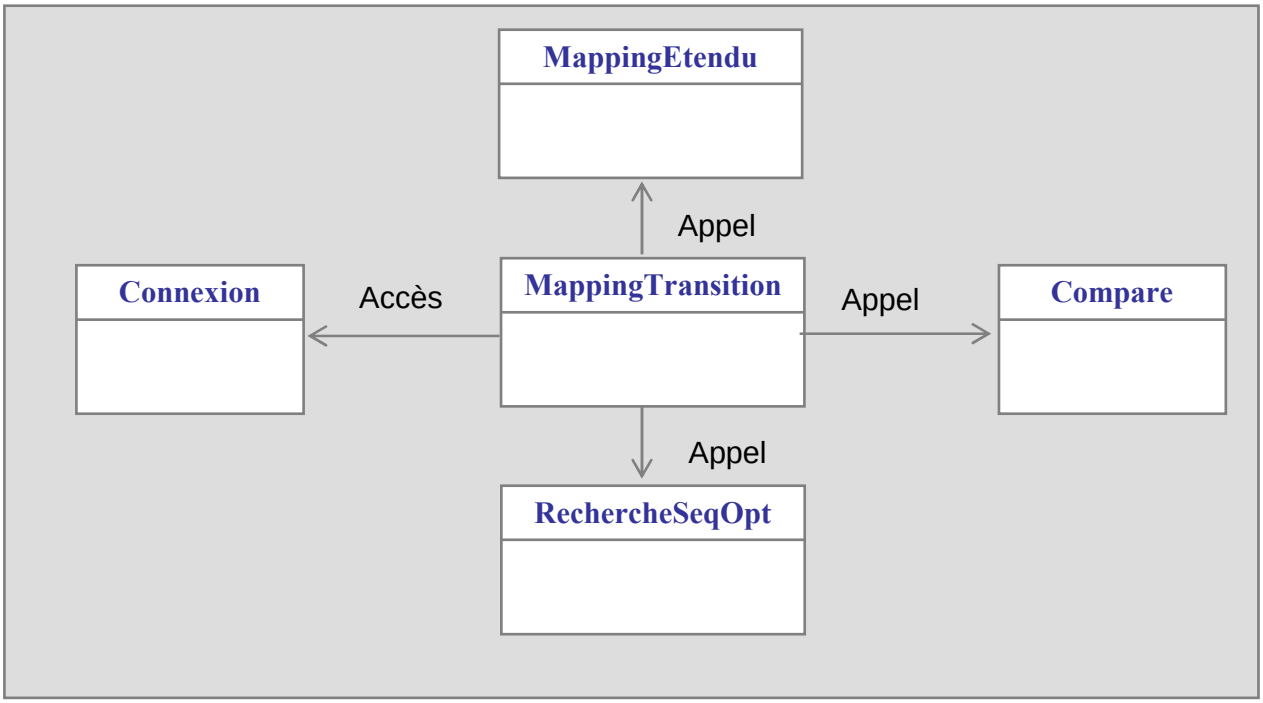
Le module de recherche des relations de mapping de transition a pour objectif d'identifier des relations sources intermédiaires pour établir des liens entre les relations de mapping étendu.

Les fonctionnalités de ce module se trouvent dans la classe *MappingTransition*. Il prend en paramètre d'entrée l'ensemble de relations de mapping étendu `Me` généré par la classe *MappingEtendu.*

Il appelle la classe *RechercheSeqOpt* qui a pour objectif de calculer les `K` plus courtes séquences d'assertions entre chaque paire de relations de mapping étendu. Cette dernière prend en entrée le paramètre `K` qui est une connaissance fixée au préalable par l'utilisateur. Elle récupère via un accès JDBC à la base de données les méta-données des tables *Contrainte_Ref* et *Corresp Ling (S-S)*, elle retourne en sortie un ensemble de séquences d'assertions pertinentes entre chaque relation de mapping étendu.

Lorsque deux relations de mapping étendu n'appartiennent pas à la même source, la classe *MappingTransition* appelle la classe *Compare* pour gérer les conflits liés à l'hétérogénéité des données qui génère en sortie un ensemble de fonctions de transformations `CF.`

La classe *MappingTransition* calcule les relations de transitions à partir des séquences d'assertions trouvées et génère en paramètre de sortie l'ensemble de relations de mapping de transition `Mt`, elle stocke les résultats produits dans la table temporaire Mapping_Transition. Les résultats de la recherche des relations de mapping transition sont affichés par le module d'interface graphique. Le diagramme de classe suivant montre les fonctionnalités de ce module :



5-6- La recherche des opérations de jointures

Diagramme 2 : La recherche des relations de transition

Le module de recherche des opérations de jointure a pour objectif d'identifier les jointures candidates pour combiner les relations de mapping. Il génère en sortie un graphe d'opérations de jointures.

Les fonctionnalités de ce module se trouvent dans la classe *GrapheOperations*. Il prend comme paramètres d'entrée l'ensemble de relations de mapping étendu `Me` et l'ensemble de relations de mapping de transition `Mt`, il récupère via un accès JDBC les tables Contrainte_Ref et Corresp Ling (S-S*)*.

Cette classe appelle aussi la classe *Compare* pour détecter et résoudre les conflits liés à l'hétérogénéité des données qui génère un ensemble de fonctions de transformations.

La classe *GrapheOperations* identifie les opérations de jointures et génère en sortie le graphe d'opérations représentant toutes les jointures. Elle stocke les résultats dans la table temporaire Jointure. Les résultats de la recherche des jointures sont affichés par le module d'interface graphique. Le diagramme de classe suivant montre les fonctionnalités de ce module :

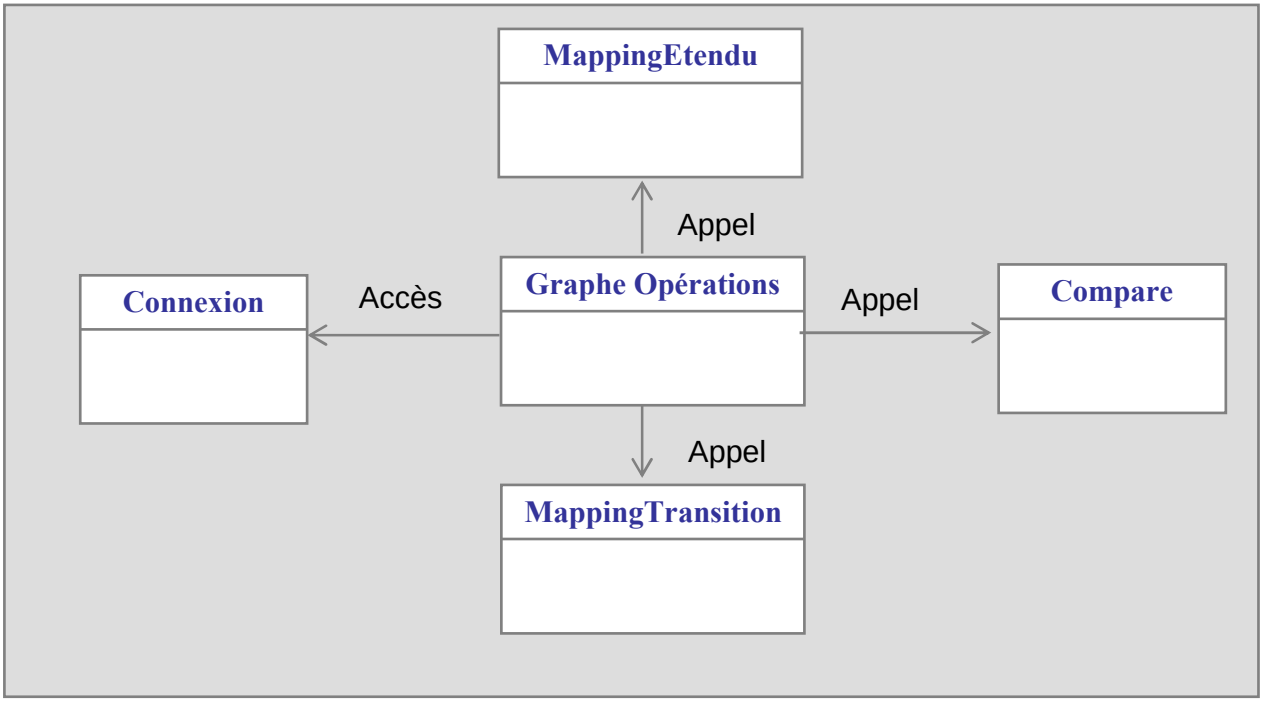


Diagramme 3 : La recherche des opérations de jointure

5-7- La recherche des chemins de calcul

Le module de recherche des chemins de calcul a pour objectif d'identifier les chemins de jointure au calcul d'une relation de médiation. Il génère en sortie un ensemble de chemins de calcul.

Les fonctionnalités de ce module se trouvent dans la classe *RechercheChemin*. Il prend comme paramètres d'entrée la relation de médiation et le graphe d'opérations généré par la classe *GrapheOperations*.

La classe *RechercheChemin* énumère tous les chemins de jointures possibles depuis le graphe de jointures et génère en sortie un ensemble de chemins. Elle stocke les résultats dans la table temporaire Chemin_Calcul. Les résultats de la recherche des chemins sont affichés par le module d'interface graphique. Le diagramme de classe suivant montre les fonctionnalités de ce module :

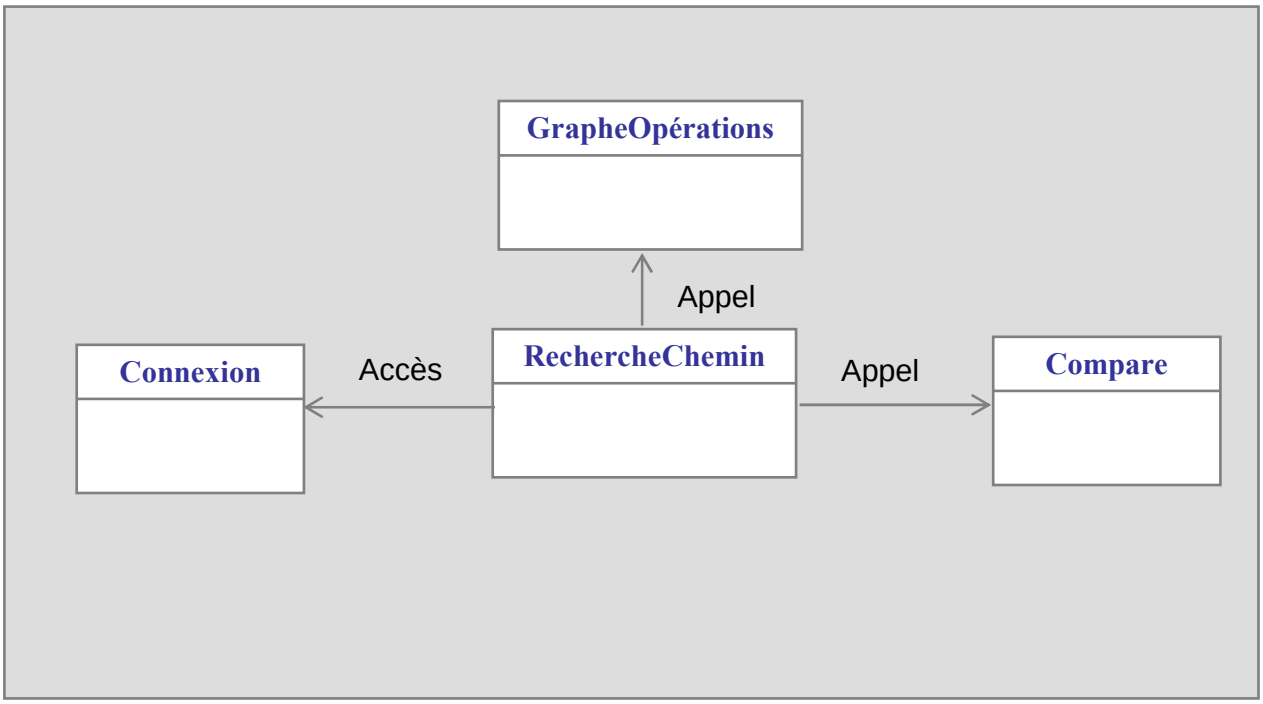


Diagramme 4 : La recherche des chemins de calcul

5-8- La recherche des requêtes de médiation

Le module de recherche des requêtes de médiation a pour objectif de générer les requêtes de médiation relatives à une relation de médiation particulière. Il génère en sortie des requêtes SQL.

Les fonctionnalités de ce module se trouvent aussi dans la classe *RechercheChemin*. Cette classe déduit des requêtes SQL à partir des chemins de calcul.

La classe *RechercheChemin* stocke les requêtes de médiation dans la table temporaire Requête. Les résultats de la recherche des requêtes de médiation sont également affichés par le module d'interface graphique. Le diagramme de classe suivant montre les fonctionnalités de ce module :

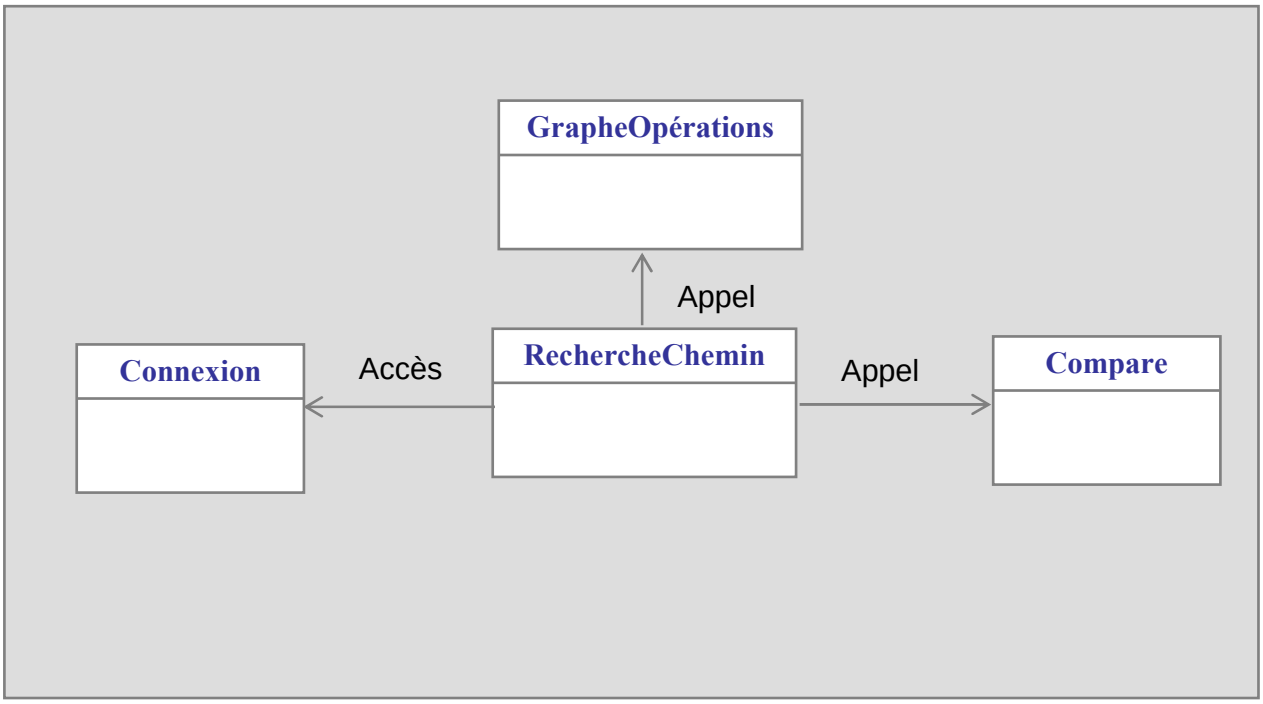


Diagramme 5 : La génération des requêtes de médiation

5-9- Scénario de fonctionnement

Afin d'illustrer le fonctionnement de notre prototype nous présentons dans cette section le scénario suivant :

Exemple

Pour notre exemple nous considérons la relation de médiation « Book » et les relations sources relatives aux sources 1 et 2 respectivement.

Schema_médiation

Book (ISBN, publisherName, publisherAddress)

Source 1

Chapter (chapterId, chapterNumber, chapterAbstract, *ISBN*)
Book (ISBN, bookTitle)
Author (authorId, authorName)
Author_Book (*ISBN*, *authorId*)
Address (*authorId*, authorAddress)

Source 2

Chapter (chapterId, chapterNumber, chapterTitle, *ISBN*)
Book (ISBN, publisherName, publisherAddress)
Author (authorId, authorName)

Nb :

Les clés primaires sont soulignées et les clés étrangères sont en caractère italique.

La figure 8 illustre la méta-base exploitée par le processus de génération automatique de requêtes de médiation. Le niveau supérieur et le niveau inférieur décrivent les méta-données respectivement au niveau de la médiation et au niveau des sources. Le niveau intermédiaire décrit les méta-données entre les schémas des sources et le schéma de médiation (Correspondances linguistiques, fonctions de transformations,…etc.).

Le menu en haut à gauche de l'écran montre les différents modules de la génération de requêtes de médiation.

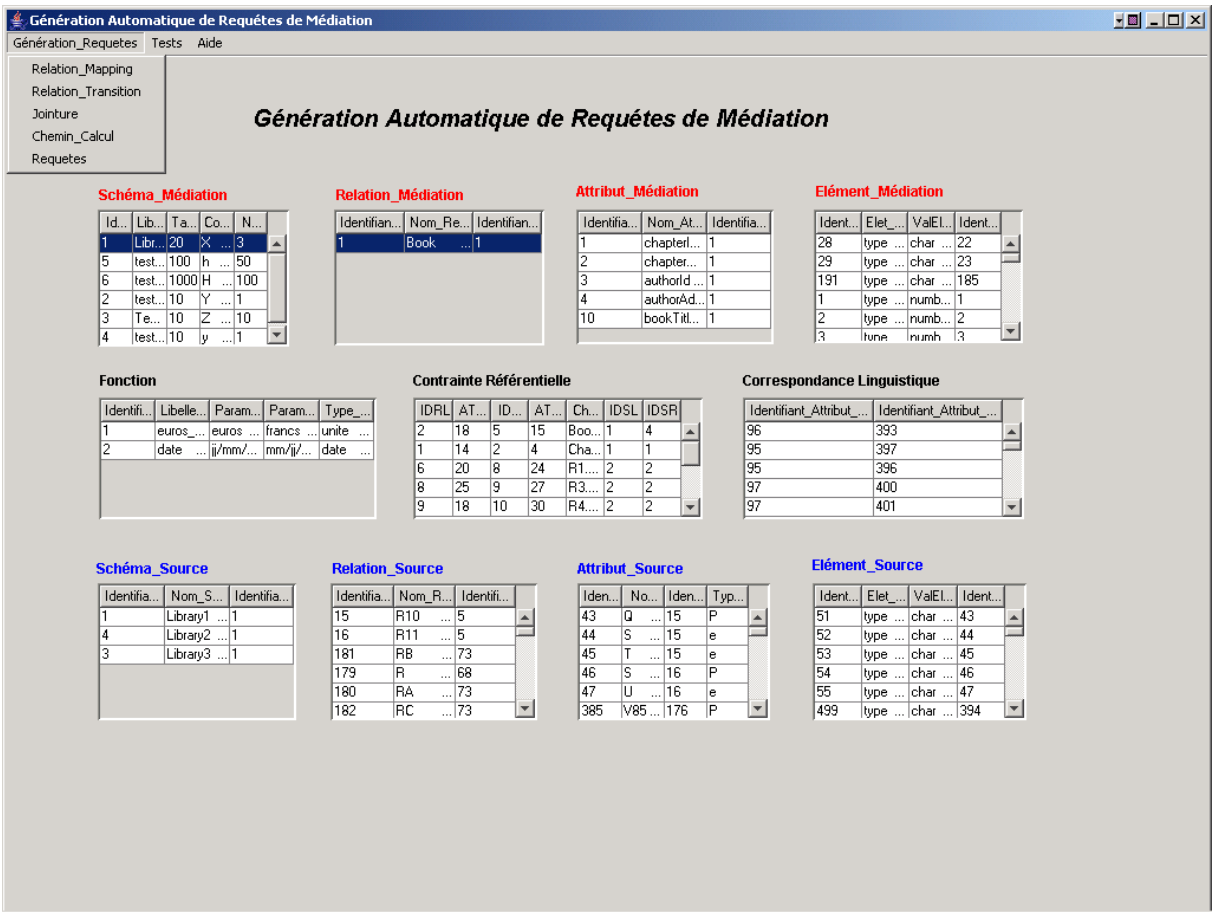


Figure 3 : Vue globale de la Méta-Base

L'interface de définition permet d'une part de donner à l'utilisateur une vue globale de la méta-base et la possibilité de mettre à jour (ajout, suppression et modification) des méta-données directement. Il est possible de mettre à jour les données de chaque table représentée sur l'interface avec un simple click sur le bouton droit de la souris qui active une fenêtre de mise à jour. Par exemple pour mettre à jour les données de la table Relation-Médiation il faut tout d'abord sélectionner un objet de la table précédente Schéma_Médiation avec le bouton gauche de la souris, et ensuite activer un menu textuel avec le bouton droit de la souris. En sélectionnant dans le menu textuel *mise à jour Relation_médition*, une fenêtre de mise à jour est alors activée. Via JDBC les données sont automatiquement stockées dans la table Relation-Médiation.
Il est possible aussi via l'interface de visualiser en cascade les données relatives à un objet particulier. Par exemple en cliquant avec le bouton gauche de la souris sur le premier schéma de médiation (Library) on visualise dans les tables Schéma_Source et Relation_Médiation respectivement les sources (Library1, library2, Library3) et la relation de médiation (Book) relatives à ce schéma de médiation. Et ainsi de suite en cliquant sur la relation de médiation Book il est possible de visualiser ses attributs dans la table Attribut_Médiation.

Cette interface permet aussi d'interagir avec le module de génération de requêtes de médiation et de montrer au fur et à mesure les résultats générés par chacune des étapes principales du module de génération de requêtes. Les résultats produits sont également stockés dans la méta-base.

Génération des relations de mapping étendu

Cette étape permet d'identifier les relations sources pertinentes au calcul de la relation de médiation à partir des sources de données. Elle génère en sortie des relations de mapping étendu qui sont obtenues par la projection des relations sources sur leurs attributs communs avec la relation de médiation.

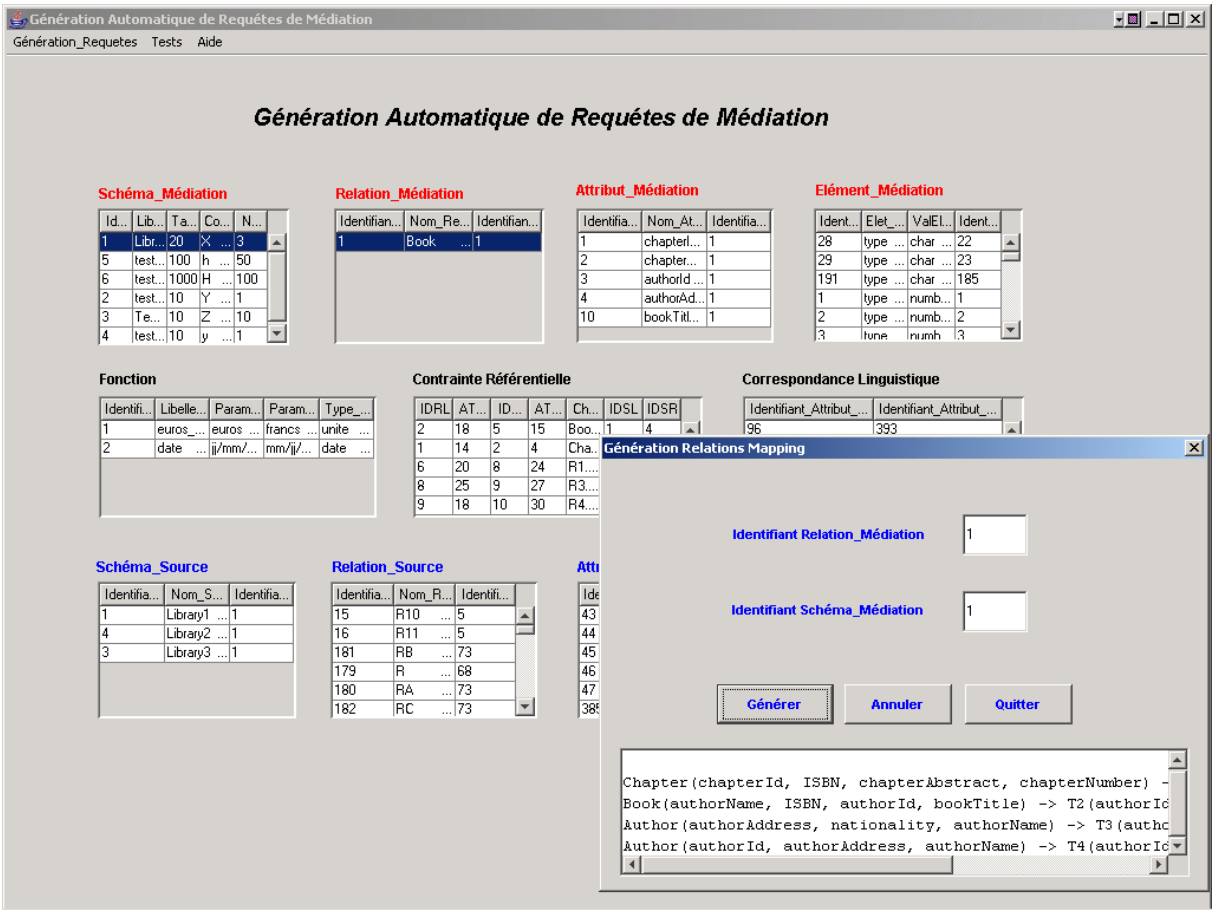


Figure 4 : Génération de relations de mapping

La fenêtre au premier plan de la figure 9 illustre les quatre relations de mapping pertinentes au calcul de la relation de médiation « Book » (id : 1) du schéma de médiation « lib » (id : 1) générées par le processus de recherche des relations de mapping étendu, première étape du processus de génération de requêtes de médiation.

Pour cet exemple nous supposons que seules les sources contributives sont impliquées dans le calcul de la relation de médiation. La recherche des relations de transition n'est donc pas nécessaire au calcul de cette relation.

Génération des jointures

Etant donné l'ensemble de relations de mapping généré précédemment, cette étape a pour objectif d'identifier les jointures possibles entre les relations de mapping en fonction de leur schéma et de leurs clés, et génération du graphe d'opérations.

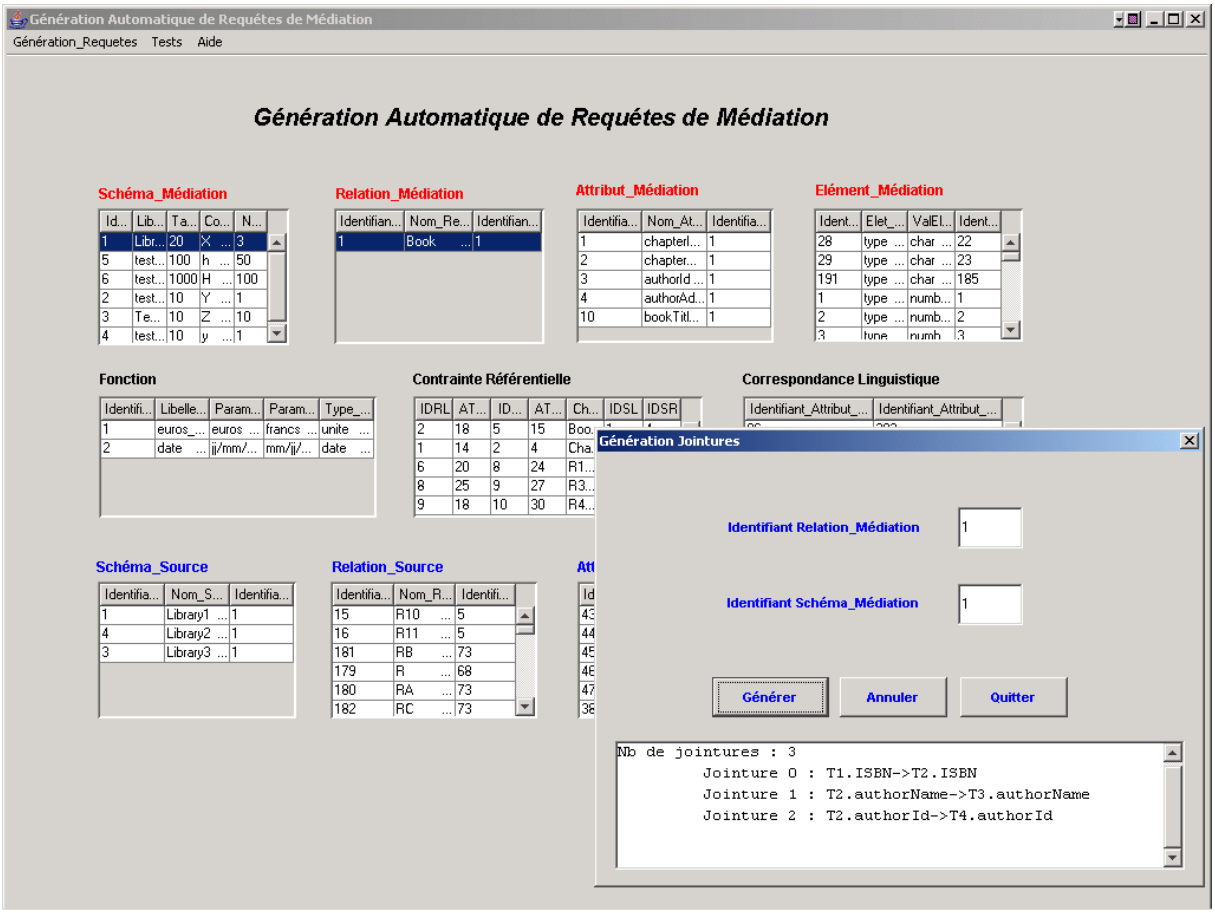


Figure 5 : Génération des opérations de jointure

La fenêtre au premier plan de la figure 10 illustre les trois opérations de jointures qui combinent les quatre relations de mapping calculées à l'étape précédente. Ces opérations de jointures sont générées par le module de recherche du graphe d'opérations, deuxième étape du processus de génération de requêtes de médiation.

Génération des chemins de calcul

L'objectif de cette étape est de rechercher des chemins de calcul à partir du graphe d'opérations pour calculer la relation de médiation.

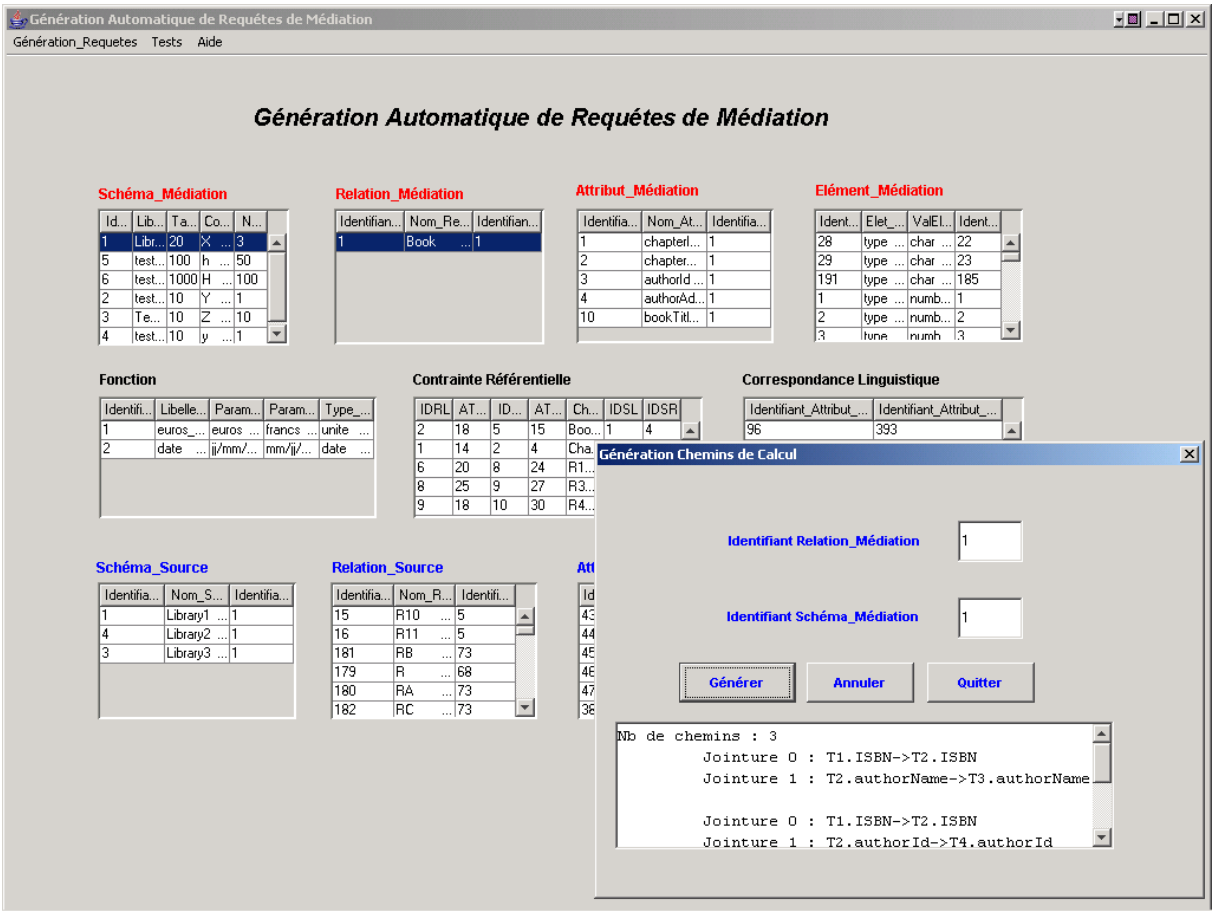


Figure 6 : Génération des chemins de calcul

La fenêtre au premier plan de la figure 11 illustre les trois chemins de jointures qui permettent de calculer la relation de médiation (id : 1) du schéma de médiation (id : 1). Ces chemins de jointures sont générés par le module de recherche des chemins de calcul.

Génération des requêtes de médiation

Une fois les chemins de calcul générés, le but est de déduire des requêtes de médiation qui calculent la relation de médiation. Chaque chemin de calcul peut dériver une requête SQL.

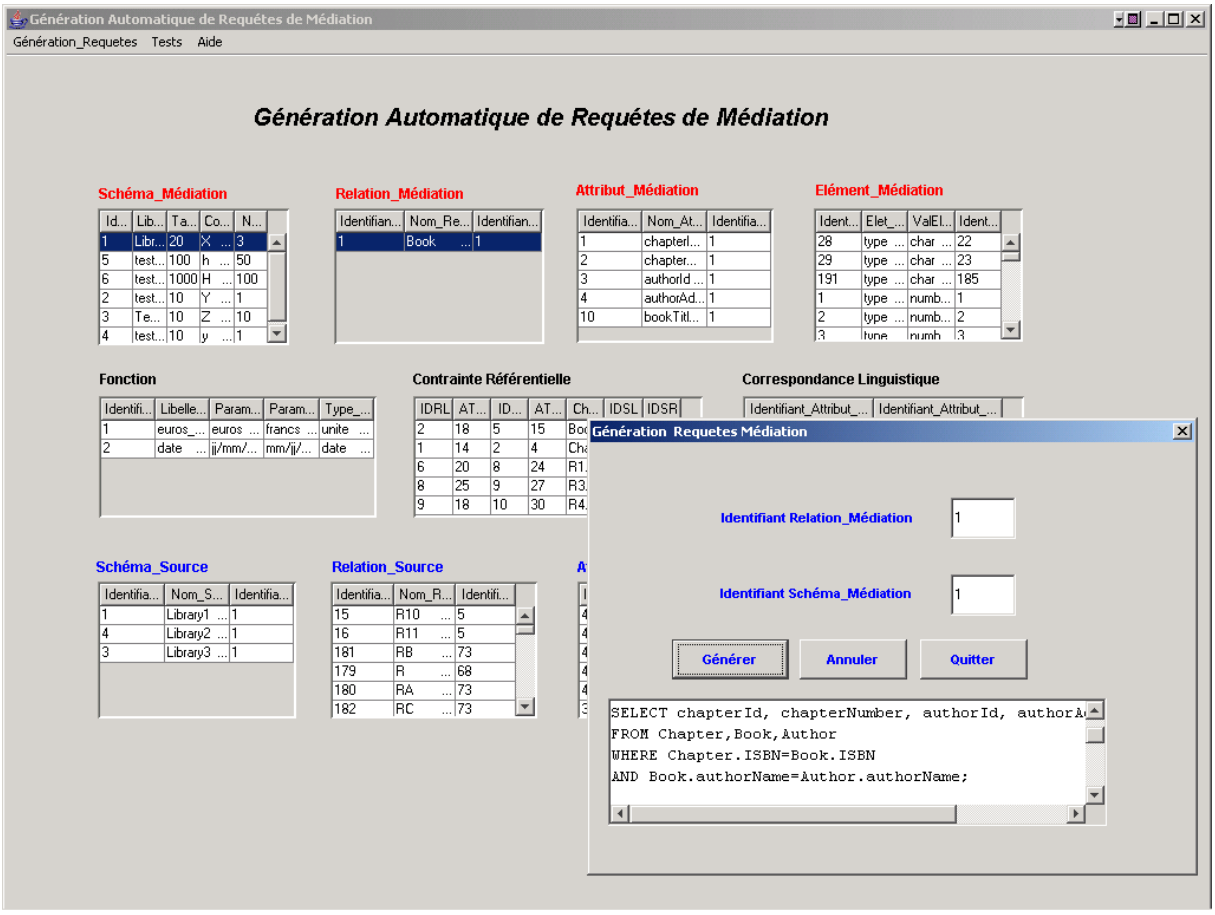


Figure 7 : Génération de requêtes de médiation

La fenêtre au premier plan de la figure 12 illustre une requête SQL dérivée à partir du premier chemin de calcul et qui permet de calculer la relation de médiation (id :1) du schéma de médiation (id :1).

Si nous supposons que dans la relation de médiation *Book*, l'attribut *price* est caractérisé par son type de base et par l'élément unité qui a pour valeur des *francs*, et dans la relation source *Book* l'attribut *price* est caractérisé par son type et par l'élément unité dont la valeur est *euros*. La conversion des euros en francs est nécessaire pour que la requête de médiation soit valide.

La fenêtre au premier plan de la figure 13 montre en gras la fonction de conversion *euros_francs* appliquée sur l'attribut *price* dans la relation de mapping T_2 générée lors de la génération des relations de mapping.

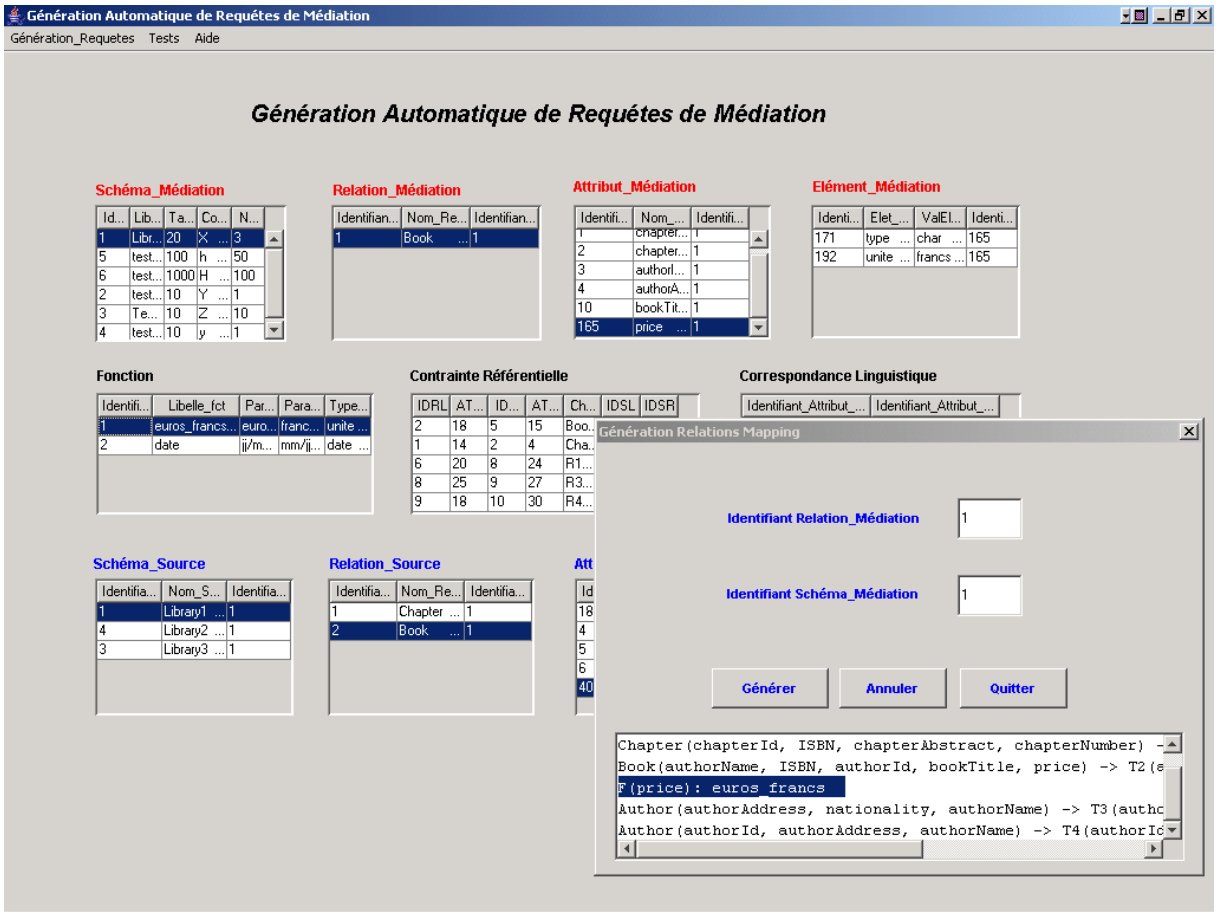


Figure 8 : Prise en compte des problèmes liés à l'hétérogénéité des données

5-10- Conclusion

Compte tenu de la problématique liée à la définition des requêtes de médiation en regard d'un grand nombre de sources et du volume important des méta-données les décrivant, nous avons présenté dans ce chapitre un outil qui permet de générer automatiquement des requêtes de médiation pour le contexte relationnel.

Cet outil tient compte des conflits liés à l'hétérogénéité des sources de données, il permet de transformer les données hétérogènes pour garantir leur conformité mutuelle et leur conformité par rapport au schéma global.

Dans [Kostadinov et al. 04] est présentée une extension de l'approche de génération de requêtes de médiation. Il est possible à partir de données semi structurées de générer des requêtes de médiation XQuery, d'évaluer la qualité des données retournées aux utilisateurs, et d'exprimer leurs préférences sous la forme de profils. Ce système compte trois modules : *interface de définition*, *génération de requêtes* et *adaptabilité*. Toutes les méta-données nécessaires sont stockées dans une méta-base. *L'interface de définition* permet de saisir des méta-données (schéma global, schéma sources, etc.) et d'interagir avec le module de génération de requêtes et le module d'adaptabilité. Le module de *génération de requêtes* est implémenté pour un contexte relationnel et XML. Le module *d'adaptabilité* comprend la gestion de profils qui vise à exprimer les préférences des utilisateurs, et à évaluer la qualité qui vise à évaluer la qualité des données afin de délivrer des résultats adaptés à leurs préférences. Ces deux modules communiquent via une méta-base commune. Les résultats produits sont aussi stockés dans la méta-base. La figure suivante illustre l'architecture du système adaptatif d'aide à la génération de requêtes de médiation.

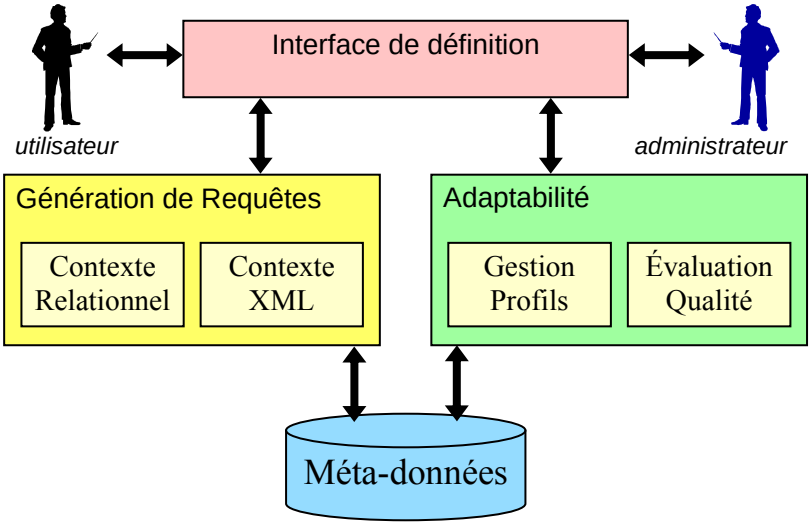


Figure 9 : Architecture du système

Le processus de génération de requêtes de médiation pour le contexte relationnel tente de trouver toutes les requêtes de médiation possibles au calcul d'une relation du schéma de médiation ; ce qui rend le processus de génération automatique complexe. Mais, sous certaines conditions simplificatrices que nous justifions, il est possible de générer dans des temps raisonnables des requêtes de médiation.

Nous présentons dans le prochain chapitre les tests réalisés pour évaluer les performances de notre outil de génération automatique de requêtes de médiation du point de vue sémantique et aussi du point de vue passage à l'échelle.

Chapitre 6 – Tests de performances

6-1- Introduction

Nous avons présenté dans le chapitre précèdent la description et le fonctionnement de notre outil de génération de requêtes de médiation. A partir du schéma d'une relation de médiation, de dépendances fonctionnelles définies sur cette relation, d'assertions linguistiques déterminées entre les sources et le schéma de médiation, et d'assertions intra-source et inter-source reliant structurellement les relations sources entre elles, notre outil produit automatiquement dans un environnement hétérogène, un ensemble de requêtes potentiel calculant cette relation.

La génération de requêtes de médiation est un problème complexe en raison de la grande diversité des sources hétérogènes et distribuées qui peuvent intervenir dans un système de médiation, mais aussi en raison du grand volume de méta-données qui les décrivent.

Au-delà des problèmes de génération automatique de requêtes de médiation, et de prise en compte des conflits liés à l'hétérogénéité des sources que nous savons résoudre, notre principal objectif est de réaliser un prototype qui supporte le passage à l'échelle.

Compte tenu de l'évaluation théorique de notre système, nous distingons parmi les phases du processus de génération de requêtes de médiation à savoir : *la recherche des relations de mapping étendu*, *la recherche des relation de transition*, *la recherche des opérations jointures* et *la recherche des chemins de calcu*l, deux phases critiques où la complexité est exponentielle. Ces phases sont les suivantes :

- La recherche des relations de transition : qui consiste à chercher pour chaque paire de relations de mapping étendu (T_i,T_j) entre lesquelles aucune assertion n'est définie, si il existe parmi les assertions définies dans la base de méta-connaissances ***A*** une séquence d'assertions permettant de lier les relations T_i et T_j.

- La recherche des chemins de calcul : qui consiste à chercher dans le graphe de jointures G_{RM} généré par le processus de génération de recherche des jointures tous les chemins de jointures possibles permettant de calculer la relation de médiation R_m.

Bien que nous avons adapté l'algorithme de *Dijkstra généralisé* proposé dans [Martins et Al. 98] qui cherche les k plus courts chemins entre deux sommets dans un graphe afin de réduire la complexité du processus de recherche des relations de transition, et ainsi passer d'une complexité exponentielle à une complexité polynomiale de l'ordre de $O(K.n)$, où k est le nombre des plus courts chemins et n le nombre de sommets, il peut subsister des cas de figure où la complexité reste exponentielle, et cela est en fonction de la distance des *K* plus courts chemins entre deux sommets.

Cependant la recherche des chemins de calcul reste dans tous les cas de figures toujours exponentielle, elle est de l'ordre de $O(\Delta^n)$, où Δ est le nombre maximum de sommets voisins d'un sommet et n le nombre de sommets. Cela revient au fait de chercher tous les chemins de jointures possibles au calcul de la relation de médiation.

Les questions principales que l'on se pose sont : 1) peut-on exploiter notre outil à grande échelle ? 2) quelles sont ses limites en regard d'un grand nombre de sources de données hétérogènes et distribuées et d'un volume important des méta-données les décrivant; 3) parmi les paramètres *relation source*, *attribut source*, *assertion (intra-source* et *inter-source)*, attribut de médiation, quels sont les paramètres influant sur le passage à l'échelle du processus de génération de requêtes de médiation ?

En réponse à ces questions, ce chapitre est centré sur des tests effectués pour évaluer sur des exemples concrets d'une part la conformité des résultats produits et d'autre part le passage à l'échelle de notre outil. Nous distinguons deux catégories de tests à savoir :

- *des tests sémantiques* : qui permettent de tester la conformité et la cohérence des requêtes produites par rapport aux attentes des utilisateurs;
- *des tests d'évaluation de performances* : qui permettent de mesurer les performances de notre outil à grande l'échelle.

Par ailleurs, nous avons développé un outil de tests qui compte essentiellement deux modules : *générateur de jeux d'essais* et *évaluateur de performances*. Le module *générateur de jeux d'essais* a pour objectif de générer automatiquement des jeux d'essais. Chaque jeu d'essai est représenté par un graphe, où les sommets du graphe sont les relations sources et les arcs représentent les assertions entre les relations sources (assertions intra-source, assertions inter-source). Nous distinguons deux catégories de jeux d'essais générés par le générateur :

- *Graphe connexe acyclique* est un graphe connexe sans cycles où chaque sommet est relié au minimum à un sommet voisin,
- *Graphe connexe cyclique* est un graphe connexe comportant des cycles où chaque sommet est relié au minimum à un sommet voisin.

Le module *évaluateur* a pour objectif de mesurer les performances du processus de génération de requêtes de médiation à grande échelle.

Ce chapitre est organisé de la façon suivante : la section 2 décrit le principe et le fonctionnement de notre outil de tests, la section 3 décrit les tests sémantiques et d'évaluation réalisés sur les jeux d'essai, et enfin la dernière section présente une synthèse.

6-2- Génération et évaluation automatique des jeux d'essais

Comme nous l'avons dit en introduction, notre objectif dans ce chapitre est de mesurer le passage à l'échelle de notre outil de génération automatique de requêtes de médiation, de définir ses limites, et d'énumérer les paramètres ayant un impact considérable lors de son passage des petits vers les grands systèmes.

Pour cette étude nous avons développé un outil de tests qui compte essentiellement deux modules : *générateur de jeux d'essais* et *évaluateur*. Le module *générateur de jeux d'essais* a pour objectif de générer automatiquement des jeux d'essais. Il prend en entrée un nombre de relations sources relatives à un schéma de médiation, et un nombre d'assertions (assertions intra-source et inter-source). Il génère en sortie un jeu d'essai représenté par un graphe, où les sommets du graphe sont les relations sources et les arcs représentent les assertions entre les relations sources. Deux types de graphe sont distingués :

- Graphe connexe acyclique : est un graphe connexe sans cycles (arbre) où chaque sommet peut être relié au minimum à un sommet voisin. La figure suivante illustre à gauche un exemple.
- Graphe connexe cyclique : est un graphe connexe comportant des cycles où chaque sommet est relié au minimum à un sommet voisin. La figure suivante illustre à droite un exemple.

Exemple

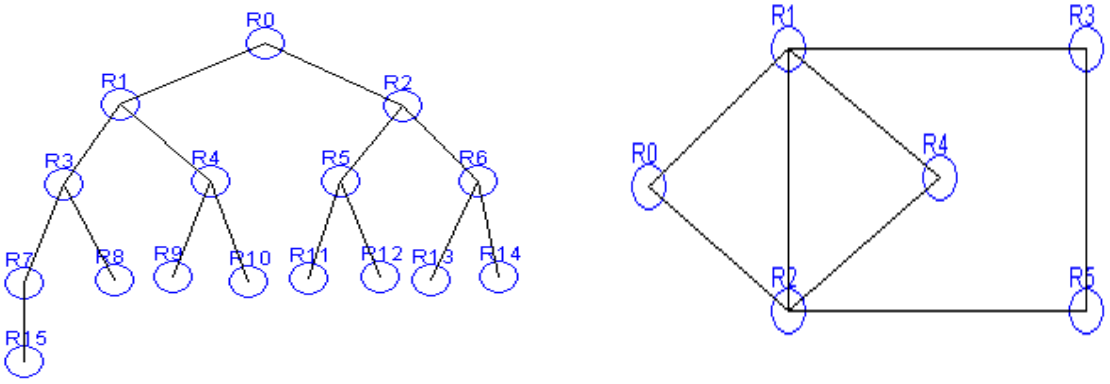


Figure 1 : Exemples de graphe connexe acyclique et cyclique

Pour le cas de figure d'un graphe connexe acyclique, nous notons que le nombre d'assertions est déduit automatiquement par le générateur à partir du nombre de relations source en raison de la linéarité du graphe. Nombre d'assertions = Nombre de relations sources - 1.

Le module *évaluateur* a pour objectif de mesurer les performances du processus de génération de requêtes de médiation. Il prend en entrée le graphe généré par le générateur, les paramètres fixes,

et un paramètre variable influent sur le passage à l'échelle à savoir : *attribut médiation*, *attribut source*, *relation source* et *assertion*. Il évalue pour chaque paramètre variable (par exemple nombre d'attribut de médiation = 10, 20,30,…) le temps d'exécution de chaque étape du processus de génération. Il génère en sortie des temps d'exécution calculés en milliseconde (par exemple pour 10 attribut de médiation le temps d'exécution de la recherche des chemins est de 8454 ms et le temps d'exécution de la génération de requêtes SQL est de 9016 ms). On remarque que le temps d'exécution de la génération de requêtes SQL est cumulé au temps d'exécution de la recherche des chemins de calcul. Ces résultats sont traduits sur des courbes par le logiciel *GNUplo*t. La figure suivante illustre l'architecture de cet outil de tests :

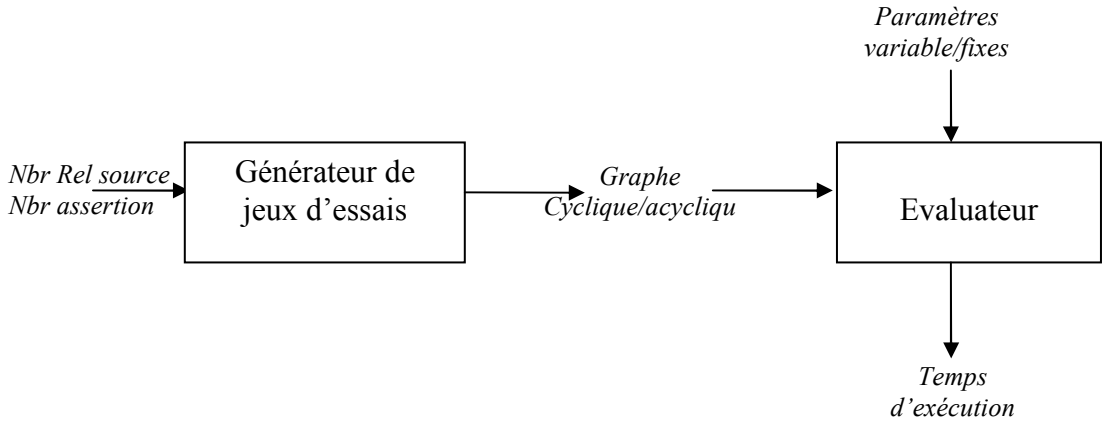


Figure2 : architecture de l'outil de tests

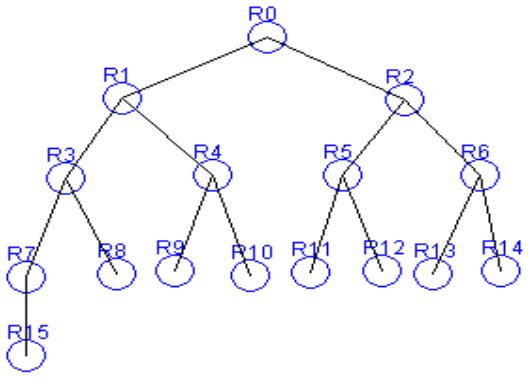
Suite à la description du fonctionnement de notre outil de tests, nous présentons dans la section suivante les résultats des tests obtenus, suivie d'une synthèse.

6-3- Résultats des tests

Avant de mesurer les performances de notre outil de génération à grande échelle, nous avons effectué des tests dits ***tests sémantiques*** pour tester dans un premier temps la conformité et la cohérence des requêtes produites par rapport aux attentes des utilisateurs. Nous justifions dans ce qui suit sur deux exemples assez complexes que notre outil de génération de requêtes de médiation peut produire des requêtes correctes à partir de n'importe quelle type de graphe acyclique ou cyclique.

Exemple 1

Considérons le jeu d'essai suivant qui contient 16 sommets et 15 assertions intra-source



Soit la relation de médiation de schéma R_M = (X158, X94, K5, X65, X59) et les relations sources appartenant à la même source de données S1 :

R0 = (K0,**K2**,**K1**,X1,X2,X3,X4,X5,X6,X7,X8,X9)
R1 = (K1,**K4**,**K3**,X11,X12,X13,X14,X15,X16,X17,X18,X19)
R2 = (K2,**K6**,**K5**,X21,X22,X23,X24,X25,X26,X27,X28,X29)
R3 = (K3,**K8**,**K7**,X31,X32,X33,X34,X35,X36,X37,X38,X39)
R4 = (K4,**K10**,**K9**,X41,X42,X43,X44,X45,X46,X47,X48,X49)
R5 = (K5,**K12**,**K11**,X51,X52,X53,X54,X55,X56,X57,X58,X59)
R6 = (K6,**K14**,**K13**,X61,X62,X63,X64,X65,X66,X67,X68,X69)
R7 = (K7,**K15**,X71,X72,X73,X74,X75,X76,X77,X78,X79)
R8 = (K8,X81,X82,X83,X84,X85,X86,X87,X88,X89)
R9 = (K9,X91,X92,X93,X94,X95,X96,X97,X98,X99)
R10 = (K10,X101,X102,X103,X104,X105,X106,X107,X108,X109)
R11 = (K11,X111,X112,X113,X114,X115,X116,X117,X118,X119)
R12 = (K12,X121,X122,X123,X124,X125,X126,X127,X128,X129)
R13 = (K13,X131,X132,X133,X134,X135,X136,X137,X138,X139)
R14 = (K14,X141,X142,X143,X144,X145,X146,X147,X148,X149)
R15 = (K15,X151,X152,X153,X154,X155,X156,X157,X158,X159)

Les clés étrangères sont représentées en caractère gras et les clés primaires sont soulignées

Soient les assertions intra-source suivantes:

R1.K1->R0.K1
R2.K2->R0.K2
R3.K3->R1.K3
R4.K4->R1.K4
R5.K5->R2.K5
R6.K6->R2.K6
R7.K7->R3.K7
R8.K8->R3.K8
R9.K9->R4.K9
R10.K10->R4.K10
R11.K11->R5.K11
R12.K12->R5.K12
R13.K13->R6.K13

R14.K14->R6.K14
R15.K15->R7.K15

Notre outil génère deux requêtes de médiation possibles pour le calcul de la relation de médiation :

Requête 1

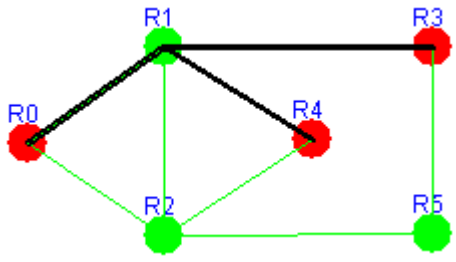
```
SELECT R15.X158 AS X158, R9.X94 AS X94, R2.K5 AS K5, R6.X65 AS X65,
R5.X59 AS X59
FROM R15, R7, R3, R1, R4, R9, R0, R2, R6, R5
WHERE R15.K15 = R7.K15
AND R7.K7 = R3.K7
AND R3.K3 = R1.K3
AND R4.K4 = R1.K4
AND R9.K9 = R4.K9
AND R1.K1 = R0.K1
AND R2.K2 = R0.K2
AND R6.K6 = R2.K6
AND R5.K5 = R2.K5
```

Requête 2

```
SELECT R15.X158 AS X158, R9.X94 AS X94, R5.K5 AS K5, R6.X65 AS X65,
R5.X59 AS X59
FROM R15, R7, R3, R1, R4, R9, R0, R2, R5, R6
WHERE R15.K15 = R7.K15
AND R7.K7 = R3.K7
AND R3.K3 = R1.K3
AND R4.K4 = R1.K4
AND R9.K9 = R4.K9
AND R1.K1 = R0.K1
AND R2.K2 = R0.K2
AND R5.K5 = R2.K5
AND R6.K6 = R2.K6
```

Exemple 2

Considérons le jeu d'essai suivant qui contient 6 sommets et 8 assertions intra-source



Considérons la relation de médiation de schéma R_M = (X17, X23, X3) et les relations sources appartenant à la même source de données S1 :

R0 = (K0,**K2**,**K1**,X1,X2,X3,X4)
R1 = (K1,**K2**,**K4**,**K3**,X6,X7,X8,X9)
R2 = (K2,**K4**,**K5**,X11,X12,X13,X14)
R3 = (K3,X16,X17,X18,X19)
R4 = (K4,X21,X22,X23,X24)
R5 = (K5,**K3**,X26,X27,X28,X29)

Soient les assertions intra-source suivantes :

R1.K1->R0.K1
R2.K2->R0.K2
R3.K3->R1.K3
R4.K4->R1.K4
R5.K5->R2.K5
R4.K4->R2.K4
R3.K3->R5.K3
R2.K2->R1.K2

Notre outil génère quatorze requêtes de médiation possibles au calcul de la relation de médiation à savoir :

Requête 1:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R0, R2, R4
WHERE R3.K3 = R1.K3
AND R1.K1 = R0.K1
AND R2.K2 = R0.K2
AND R4.K4 = R2.K4
```

Requête 2:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R4, R5, R2, R0
WHERE R3.K3 = R1.K3
AND R4.K4 = R1.K4
AND R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R2.K2 = R0.K2
```

Requête 3:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R4, R0
WHERE R3.K3 = R1.K3
AND R4.K4 = R1.K4
AND R1.K1 = R0.K1
```

Requête 4:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R4, R2, R0
WHERE R3.K3 = R1.K3
```

```
AND R4.K4 = R1.K4
AND R2.K2 = R1.K2
AND R2.K2 = R0.K2
```

Requête 5:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R4, R2, R0
WHERE R3.K3 = R1.K3
AND R4.K4 = R1.K4
AND R4.K4 = R2.K4
AND R2.K2 = R0.K2
```

Requête 6:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R2, R4, R0
WHERE R3.K3 = R1.K3
AND R2.K2 = R1.K2
AND R4.K4 = R2.K4
AND R1.K1 = R0.K1
```

Requête 7:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R1, R2, R4, R0
WHERE R3.K3 = R1.K3
AND R2.K2 = R1.K2
AND R4.K4 = R2.K4
AND R2.K2 = R0.K2
```

Requête 8:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R0, R1, R4
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R2.K2 = R0.K2
AND R1.K1 = R0.K1
AND R4.K4 = R1.K4
```

Requête 9:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R4, R1, R0
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R4.K4 = R2.K4
AND R3.K3 = R1.K3
AND R1.K1 = R0.K1
```

Requête 10:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R4, R0
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
```

```
AND R4.K4 = R2.K4
AND R2.K2 = R0.K2
```

Requête 11:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R4, R1, R0
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R4.K4 = R2.K4
AND R2.K2 = R1.K2
AND R1.K1 = R0.K1
```

Requête 12:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R4, R1, R0
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R4.K4 = R2.K4
AND R4.K4 = R1.K4
AND R1.K1 = R0.K1
```

Requête 13:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R1, R4, R0
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R2.K2 = R1.K2
AND R4.K4 = R1.K4
AND R2.K2 = R0.K2
```

Requête 14:

```
SELECT R3.X17 AS X17, R4.X23 AS X23, R0.X3 AS X3
FROM R3, R5, R2, R1, R4,R0
WHERE R3.K3 = R5.K3
AND R5.K5 = R2.K5
AND R2.K2 = R1.K2
AND R4.K4 = R1.K4
AND R1.K1 = R0.K1
```

A partir de ces deux exemples nous justifions que notre outil permet de générer pour un graphe connexe acyclique ou un graphe connexe cyclique, toutes les solutions pertinentes possibles au calcul d'une relation de médiation particulière tout en tenant compte des conflits liés à l'hétérogénéité des sources.

Après avoir montrer l'efficacité de notre outil d'un point de vue sémantique, nous justifions dans un second temps par des tests d'évaluation de performances que notre outil peut produire des résultats dans des temps raisonnables. Nous présentons tout d'abord les résultats des tests effectués sur un graphe connexe acyclique et ensuite sur un graphe connexe cyclique en fonction des paramètres variables suivants : *attribut médiation*, *attribut source*, *relation source* et *assertion*.

6-3-1- Tests d'évaluation de performances effectués sur un graphe acyclique

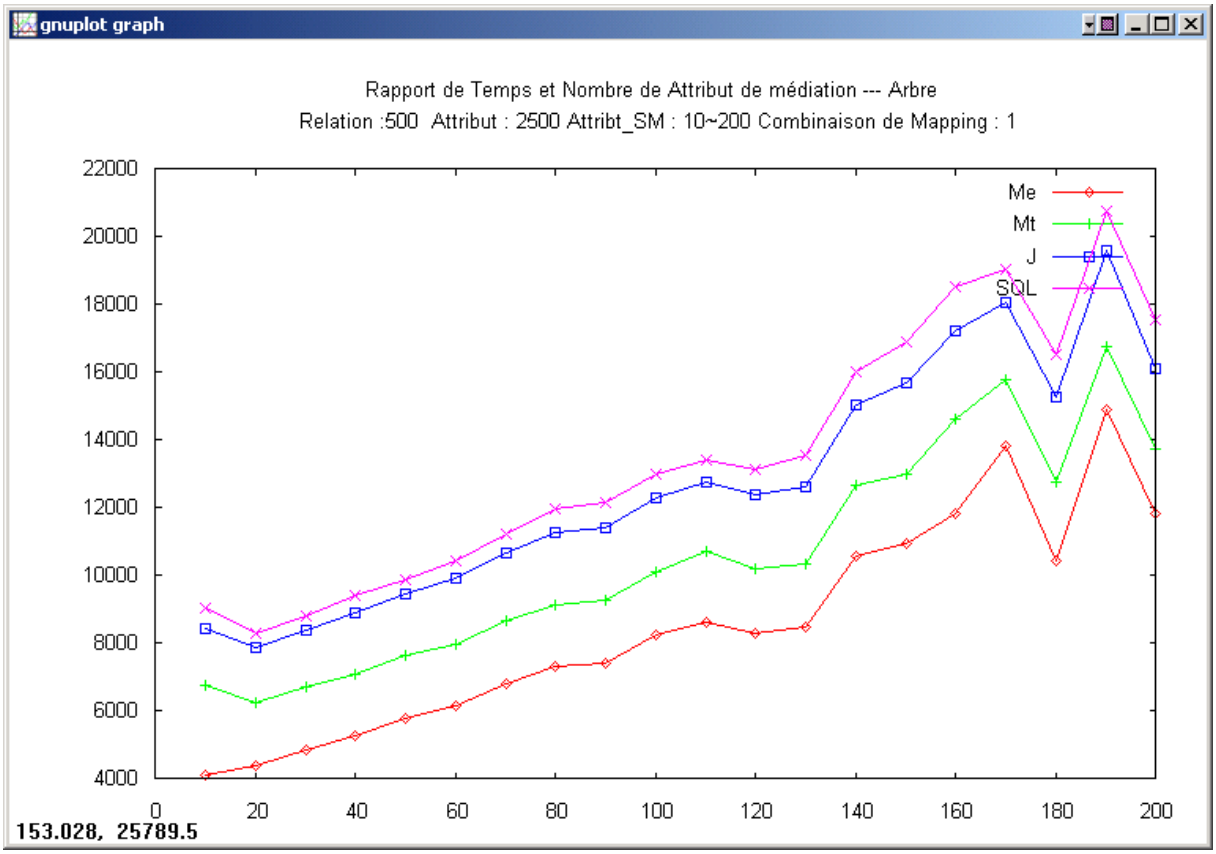
<u>Test 1</u> :

Etant donné les paramètres fixes suivants :

Nombre de relations source = 500,
Nombre d'attributs source = 5 pour chaque relation source
Nombre d'assertions = 499

Nous rappelons que pour le cas de figure d'un graphe connexe acyclique, le nombre d'assertions est déduit automatiquement par le générateur à partir du nombre de relations source en raison de la linéarité du graphe. Nombre d'assertions = Nombre de relations sources - 1.

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***attribut de médiation*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre attribut de médiation (10, 20, 30,…,200) et l'axe des Y décrit les temps d'exécution en milliseconde.

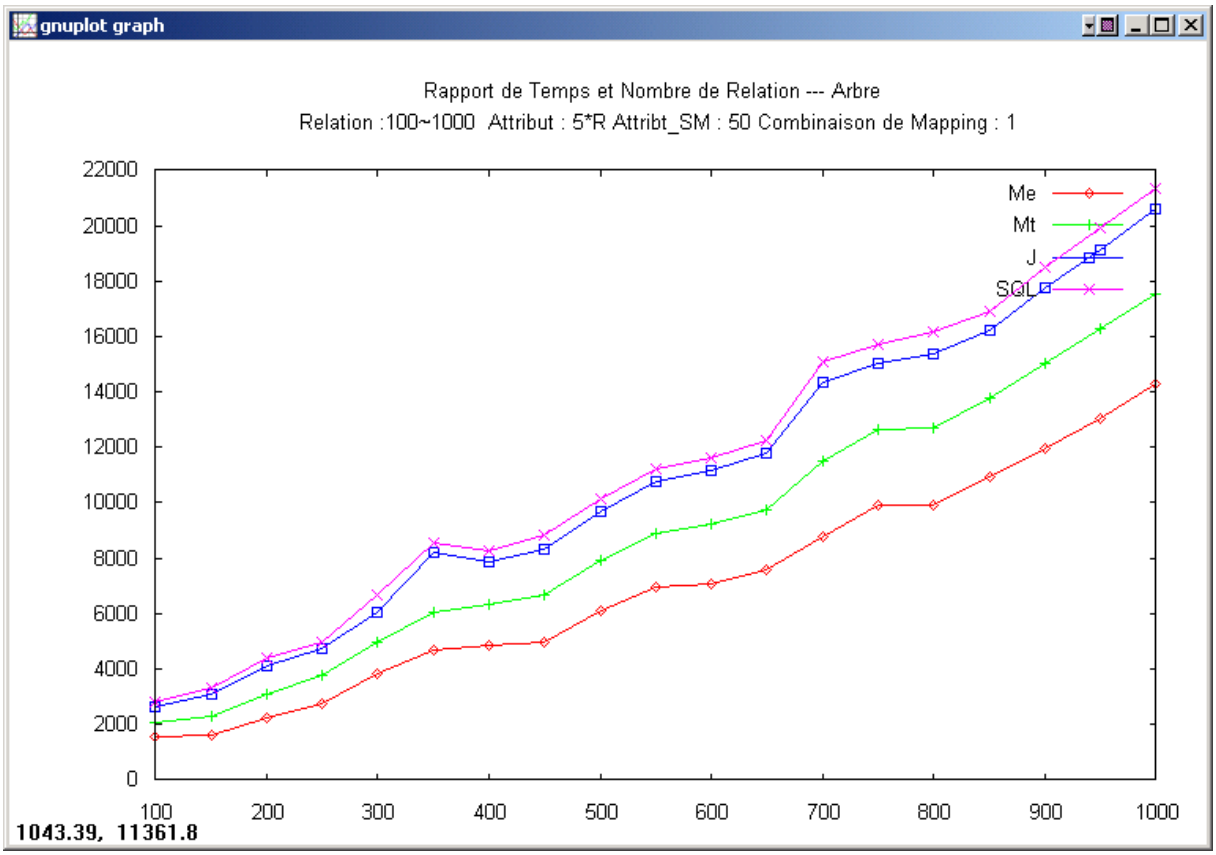
Cette courbe montre que le temps d'exécution du processus de génération augmente linéairement dans les quatre étapes du processus. La complexité du processus de génération pour ce cas de figure est toujours polynomiale quelque soit le nombre d'attributs de médiation.

<u>Test 2</u> :

Etant donné les paramètres fixes suivants :

Nombre d'attribut de médiation = 50,
Nombre attribut source = 5 pour chaque relation source
Nombre d'assertions = nombre de relation source -1

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***relation source*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre relation source (100, 150, 200,…,1000) et l'axe des Y décrit les temps d'exécution en milliseconde.

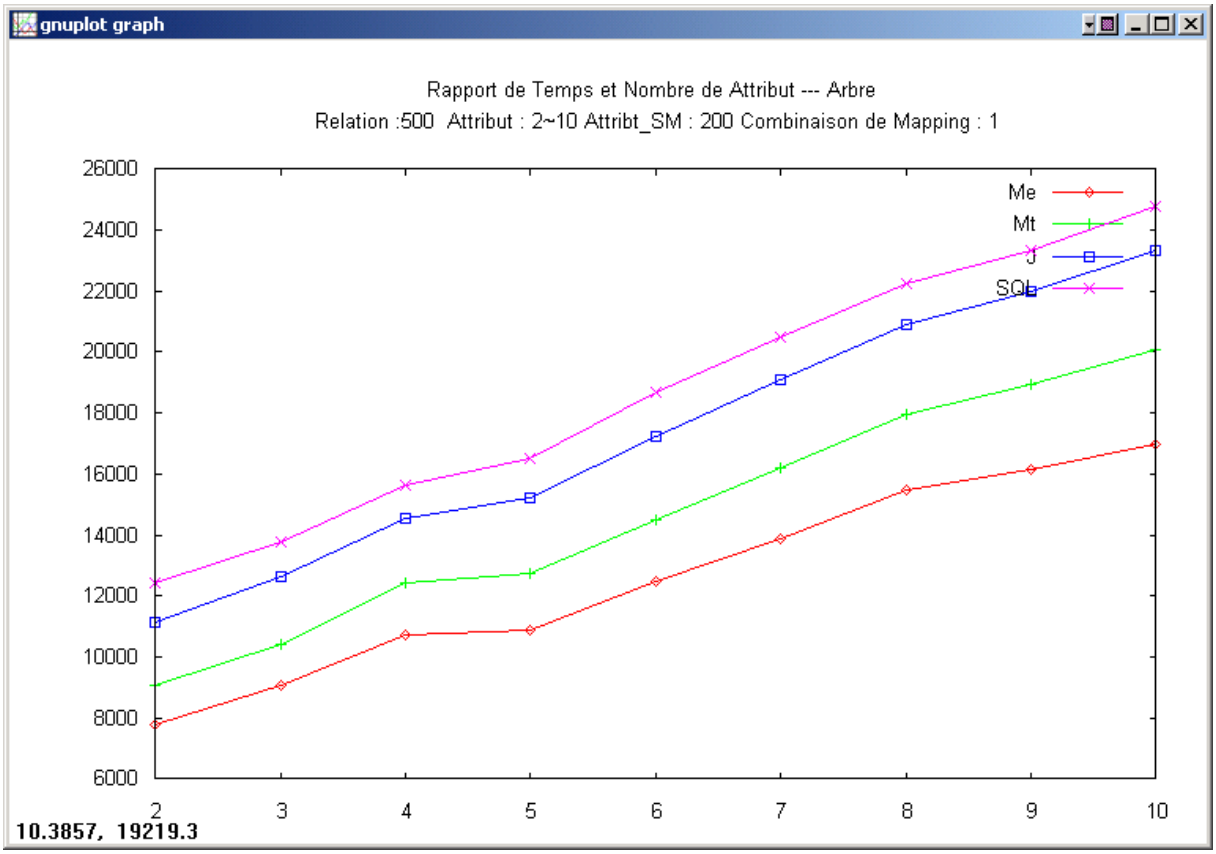
Cette courbe montre que le temps d'exécution du processus de génération augmente linéairement dans les quatre étapes du processus. La complexité du processus de génération pour ce cas de figure est toujours polynomiale quelque soit le nombre de relations source.

Test 3 :

Etant donné les paramètres fixes suivants :

Nombre d'attribut de médiation = 200,
Nombre relation source = 500
Nombre d'assertions = 499

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***attribut source*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre attribut source (2, 3, 4, 5,…,10) et l'axe des Y décrit les temps d'exécution en milliseconde. Le nombre d'attributs varie pour chaque relation source.

Cette courbe montre que le temps d'exécution du processus de génération augmente linéairement dans les quatre étapes du processus. La complexité du processus de génération pour ce cas de figure est toujours polynomiale quelque soit le nombre d'attribut source.

A partir de ces trois courbes nous justifions que le temps d'exécution du processus de génération augmente toujours linéairement dans les quatre étapes du processus. La complexité du processus de génération pour les graphes connexe acyclique est toujours polynomiale quelque soit la taille du graphe et les paramètres que l'on fait varier.

Pour le cas de figure des graphes connexes acycliques, on ne teste pas l'impact du paramètre assertion en raison d'absence de cycles. L'ajout d'une assertion dans un graphe connexe acyclique peut entraîner la création de cycles dans le graphe.

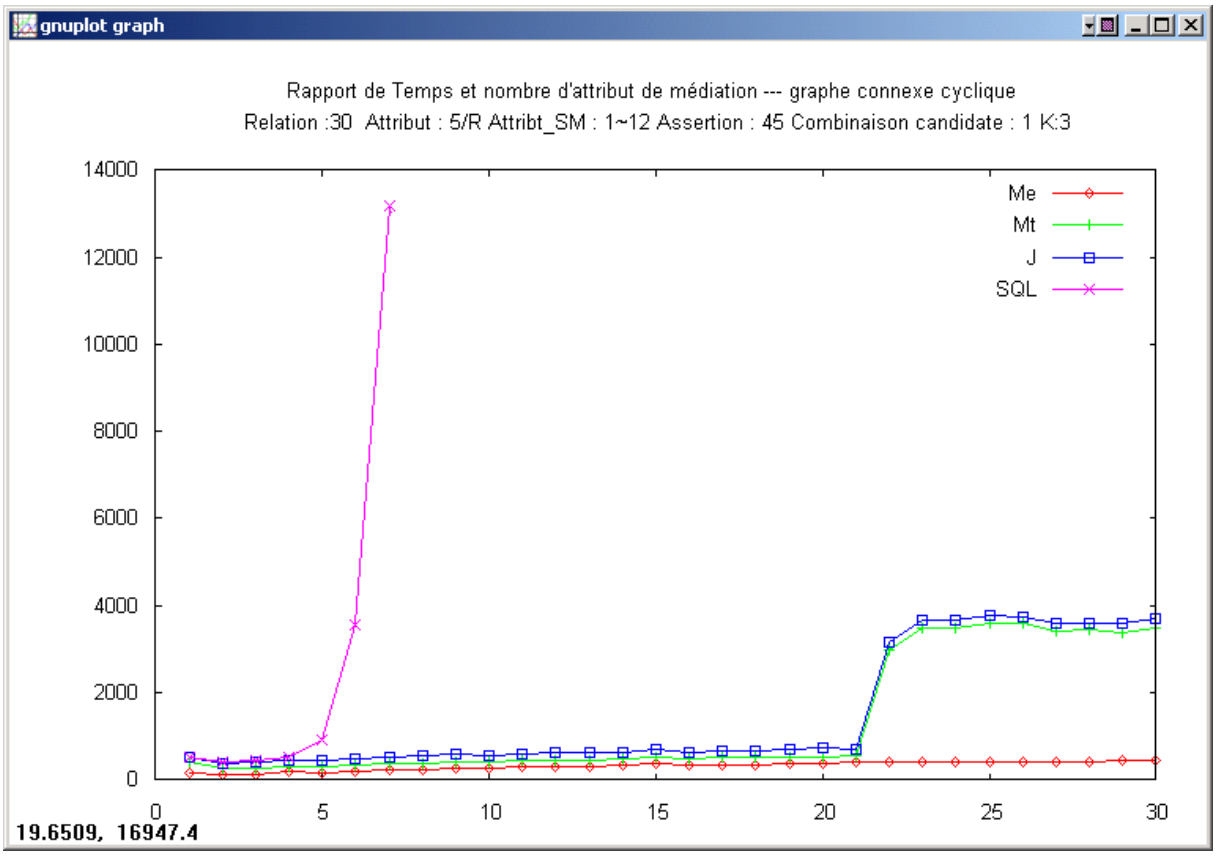
6-3-2-Tests d'évaluation effectués sur un graphe connexe cyclique

Test1 :

Etant donné les paramètres fixes suivants :

Nombre relation source = 30
Nombre attribut source = 5/relation source
Nombre assertion = 45

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***attribut médiation*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre attribut médiation (1, 2,…,30) et l'axe des Y décrit les temps d'exécution en milliseconde.

Cette courbe montre que le paramètre attribut médiation a un impact considérable sur le processus de génération de requêtes de médiation et en particulier sur l'algorithme de recherche des chemins de calcul. Plus le nombre d'attributs de médiation augmente, plus le temps de calcul augmente en raison de la complexité du graphe (cyclique).
Cependant ce paramètre n'a aucun impact sur la recherche des relations de mapping étendu, des relations de transitions et des jointures. La complexité de ces phases est en général linéaire.

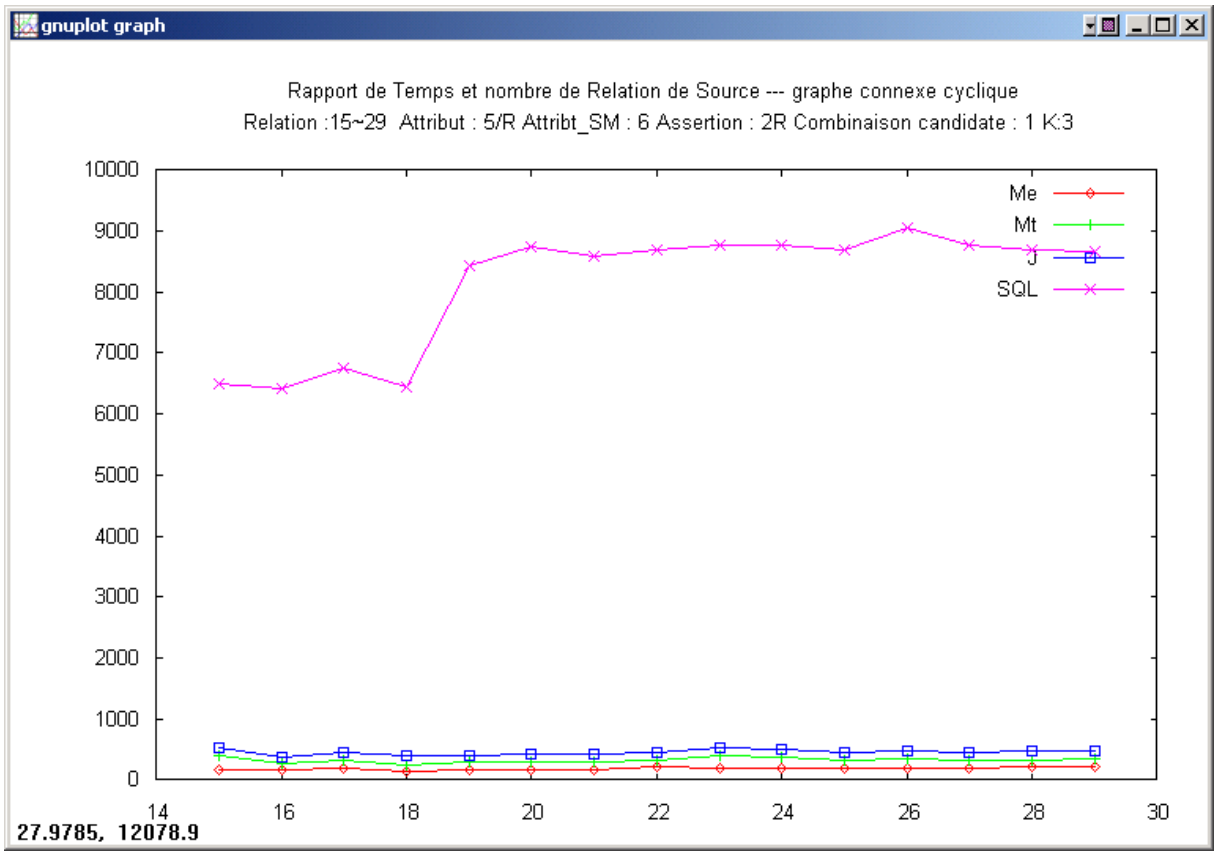
Test 2 :

Etant donné les paramètres fixes suivants :

Nombre d'attribut de médiation = 5,
Nombre attribut source = 5 pour chaque relation source

Etant donné que les tests sont effectués sur des graphes connexes, l'ajout d'une relation source entraîne l'ajout d'une assertion pour préserver la connexité du graphe.

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***relation source*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre relation source (15, 16,…,30) et l'axe des Y décrit les temps d'exécution en milliseconde.

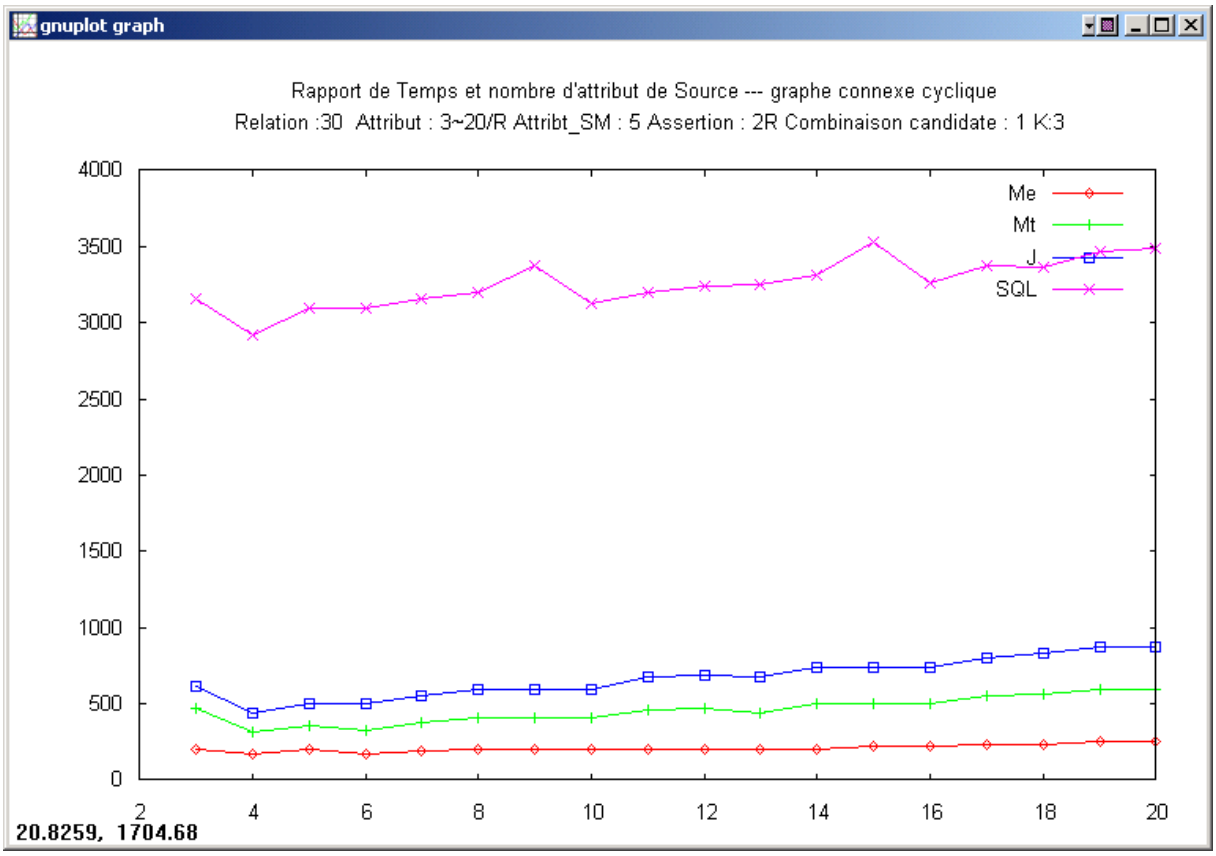
Cette courbe montre que le paramètre relation source n'a pas d'impact sur le processus de génération de requêtes de médiation. La complexité de la recherche des relations de mapping étendu, des relations de transitions, de la recherche des jointures et de la recherche des chemins de calcul est linéaire.

Test 3 :

Etant donné les paramètres fixes suivants :

Nombre d'attribut de médiation = 5
Nombre relation source = 30
Nombre d'assertions = 45

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***attribut source*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre attribut source (15, 16,…,20) et l'axe des Y décrit les temps d'exécution en milliseconde.

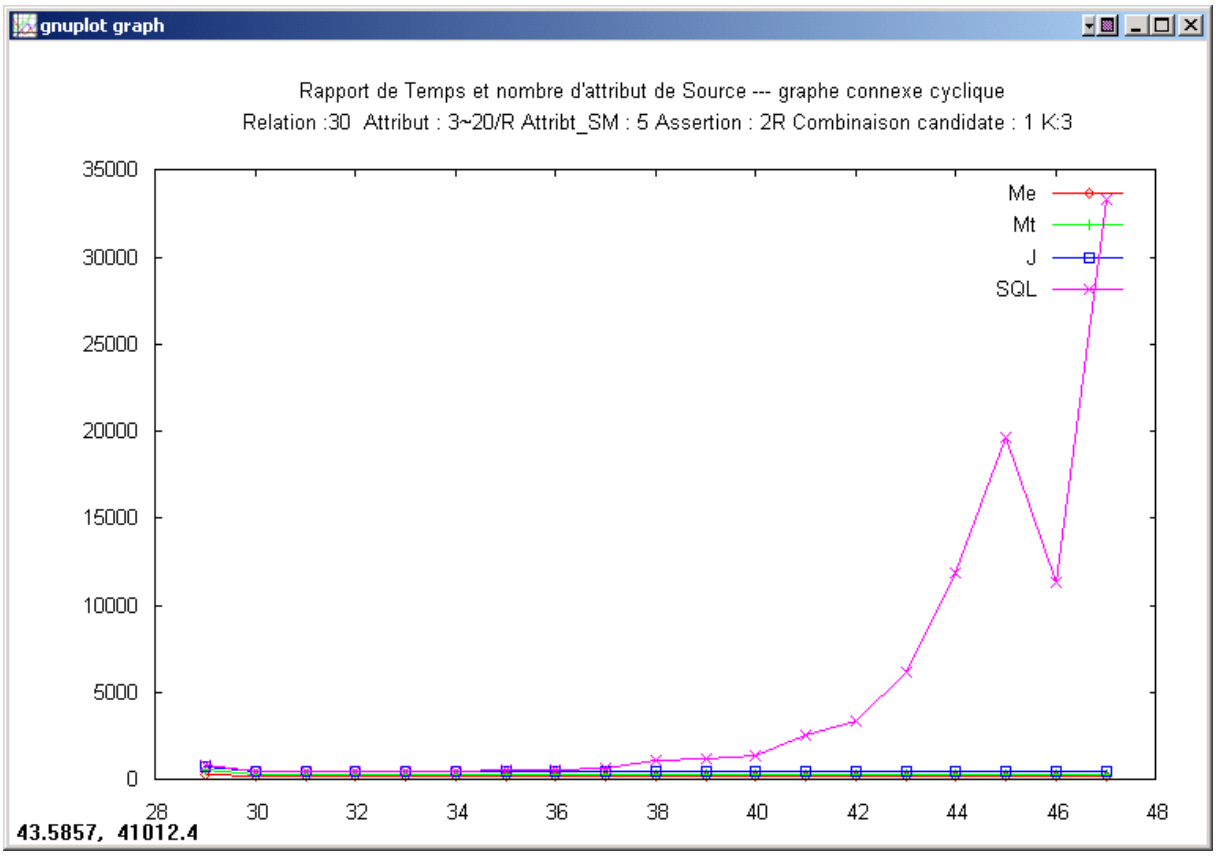
Cette courbe montre que le paramètre attribut source n'a pas d'impact sur le processus de génération de requêtes de médiation. La complexité de la recherche des relations de mapping étendu, des relations de transitions, de la recherche des jointures et de la recherche des chemins de calcul est toujours linéaire.

<u>Test 4</u> :

Etant donné les paramètres fixes suivants :

Nombre d'attribut de médiation = 5
Nombre relation source = 30
Nombre attribut source = 5/ relation source

L'évaluation des performances de la génération de requêtes de médiation en fonction du paramètre variable ***assertion*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre assertion (29,30,…,47) et l'axe des Y décrit les temps d'exécution en milliseconde.

Cette courbe montre que le paramètre assertion a un impact considérable sur le processus de génération de requêtes de médiation, en particulier sur la recherche des chemins de calcul. Plus le nombre d'assertions est augmenté, plus le temps d'exécution est long. Lorsqu'on augmente le nombre d'assertions dans un graphe connexe cyclique, ce graphe devient encore plus complexe. Nous notons que parfois lorsqu'on ajoute une assertion `a` particulière entre deux sommets il peut arriver que le temps d'exécution soit moins long que lors du rajout de l'assertion précédente `a-1` (cas de l'assertion 45 et 46 dans la courbe), cela est du au fait de la génération aléatoire des assertions par le générateur de tests. Si l'assertion `a` ajoutée entre deux sommets permet d'établir un chemin de calcul mois long que le précèdent, il est évident que le temps d'exécution diminue.
La courbe montre que le paramètre assertion n'a pas d'impact sur la complexité de la recherche des relations de mapping étendu, des relations de transitions, de la recherche des jointures. Elle est toujours linéaire.

6-4- Synthèse

Pour cette étude, nous distinguons deux catégories de jeux d'essais générés par le générateur à savoir :

- *Graphe connexe acyclique* est un graphe connexe sans cycles où chaque sommet est relié au minimum à un sommet voisin,
- *Graphe connexe cyclique* est un graphe connexe comportant des cycles où chaque sommet est relié au minimum à un sommet voisin.

Avant de mesurer les performances de notre outil de génération à grande échelle, nous avons effectué des tests dits ***tests sémantiques*** pour tester dans un premier temps la conformité et la cohérence des requêtes produites par rapport aux attentes des utilisateurs. Nous justifions sur deux exemples assez complexes que notre outil de génération de requêtes de médiation peut produire des requêtes correctes à partir de n'importe quelle type de graphe à savoir acyclique ou cyclique.

Les tests d'évaluation des performances effectués sur les graphes connexes acycliques ont montré que les paramètres variables à savoir *relation source*, *assertion*, *attribut source* et *attribut médiation* n'ont pas d'impact sur le processus de génération de requêtes de médiation. Il est possible de générer toutes les requêtes possibles dans des temps raisonnable et ce quelque soit la taille du graphe. Le temps de réponse moyen est d'environ 30 secondes.

Par ailleurs, les tests d'évaluation des performances effectués sur des graphes connexes cycliques ont montré que certains paramètres variables ont un impact considérable sur le processus de génération de requêtes de médiation et particulièrement sur la recherche des chemins de calcul. Ces paramètres influents sont : le paramètre *assertion* et le paramètre *attribut médiation.* Pour le paramètre *assertion*, plus le nombre d'assertions est augmenté le temps d'exécution est long. Lorsqu'on augmente le nombre d'assertions dans un graphe connexe cyclique, ce graphe devient encore plus complexe. Nous notons que parfois lorsqu'on ajoute une assertion `a` particulière entre deux sommets il peut arriver que le temps d'exécution soit moins long que lors du rajout de l'assertion précédente `a-1` (cas de l'assertion 45 et 46 dans le test 4), cela est du au fait de la génération aléatoire des assertions par le générateur de tests. Si l'assertion `a` ajoutée entre deux sommets permet d'établir un chemin de calcul mois long que le précèdent, il est évident que le temps d'exécution diminue. Pour le paramètre *attribut médiation,* plus où le nombre d'attribut de médiation augmente, plus le temps d'exécution de la recherche des chemins de calcul augmente aussi, en raison d'une part de la complexité du graphe (cyclique), et d'autre part pour la recherche de tous les chemins de jointures possibles permettant de calculer la relation de médiation.

Compte tenu de l'évaluation théorique et pratique de notre système, nous distinguons parmi les phases du processus de génération de requêtes de médiation à savoir : *la recherche des relations de mapping étendu*, *la recherche des relation de transition*, *la recherche des opérations jointures* et *la recherche des chemins de calcu*l, deux phases critiques où la complexité est exponentielle. Ces phases sont les suivantes :

- La recherche des relations de transition : qui consiste à chercher pour chaque paire de relations de mapping étendu (T_i,T_j) entre lesquelles aucune assertion n'est définie, si il existe parmi les assertions définies dans la base de méta-connaissances ***A*** une séquence d'assertions permettant de lier les relations T_i et T_j.

- La recherche des chemins de calcul : qui consiste à chercher dans le graphe de jointures G_{RM} généré par le processus de génération de recherche des jointures tous les chemins de jointures possibles permettant de calculer la relation de médiation R_m.

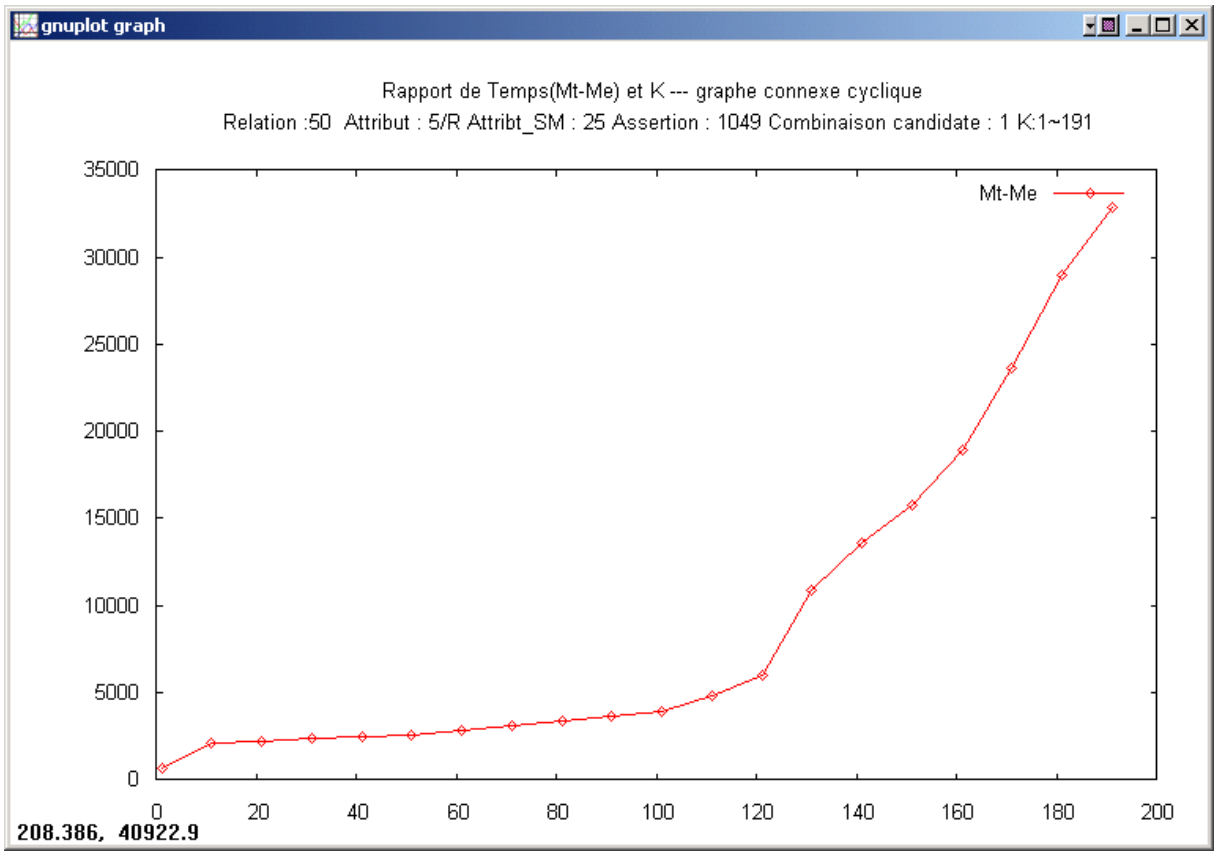
Pour la recherche des relations de transition, les tests effectués montrent tous la linéarité de cette phase. Cependant, il peut subsister des cas de figure où la complexité reste exponentielle, et cela est en fonction de la distance des *K* plus courts chemins entre deux sommets. Le test suivant illustre ce cas de figure :

Test :

Etant donné les paramètres fixes suivants :

Nombre d'attribut de médiation = 25
Nombre relation source = 50
Nombre attribut source = 5/ relation source
Nombre assertion = 1049

L'évaluation des performances de ***la génération des relations de transition*** en fonction du paramètre variable ***K*** est donnée dans la courbe ci-dessous :



L'axe des X décrit les variations du paramètre K (1,11,…,191) et l'axe des Y décrit les temps d'exécution en milli seconde.

Bien que nous avons adapté l'algorithme de *Dijkstra généralisé* proposé dans [Martins et Al. 98] qui cherche les k plus courts chemins entre deux sommets dans un graphe afin de réduire la complexité du processus de recherche des relations de transition, et ainsi passer d'une complexité exponentielle à une complexité polynomiale de l'ordre de $O(K.n)$, où k est le nombre des plus courts chemins et n le nombre de sommets, il peut subsister des cas de figure où la complexité reste exponentielle, et cela est en fonction de la distance des K plus courts chemins entre deux sommets. Elle peut être linéaire si la longueur des k plus courts chemins n'est pas très longue entre deux sommets, sinon elle est peut être exponentielle.

Cependant la recherche des chemins de calcul reste dans tous les cas de figures toujours exponentielle, elle est de l'ordre de $O(\Delta^n)$, où Δ est le nombre maximum de sommets voisins d'un sommet et n le nombre de sommets. Cela revient au fait de chercher tous les chemins de jointures possibles au calcul de la relation de médiation.

En résumé, pour un graphe connexe acyclique notre outil permet le passage à l'échelle, il peut générer des requêtes de médiation dans des temps raisonnables (en moyenne 30 secondes) pour des graphes de grandes tailles. Pour un graphe connexe cyclique notre outil peut générer des requêtes de médiation dans des temps raisonnables (en moyenne 30 secondes) pour des tailles de graphes limités. Au-delà du nombre d'attribut de médiation = 7 et du nombre d'assertion = 46, le

temps d’exécution du processus de génération de requêtes de médiation et particulièrement le temps d’exécution de la recherche des chemins de calcul sont très long.

Chapitre 7- Conclusion

Nous concluons cette thèse par un bilan résumant notre apport dans le contexte des systèmes de médiation, et des possibilités d'extensions et d'améliorations de notre outil de génération automatique des requêtes de médiation.

7-1- Bilan

Le problème de définition de requêtes de médiation est un problème complexe en raison de la grande diversité des sources hétérogènes et distribuées qui peuvent intervenir dans un système de médiation et du grand volume de méta-données qui les décrivent, mais aussi en raison des conflits liés à l'hétérogénéité des données qui peuvent exister entre deux sources de données.

Bien que les systèmes de médiation soient de nos jours des systèmes de plus en plus développés et connus, très peu de travaux se sont intéressés à la génération de requêtes de médiation et à la prise en compte des conflits liés à l'hétérogénéité des sources durant le processus de génération. La plupart se sont focalisés sur la détection et (ou) sur la résolution d'un conflit particulier dans le contexte de l'intégration de données. Ce manque de travaux méritant une exploration plus approfondie dans le domaine de la recherche, a motivé notre étude à explorer plus en détails la problématique liée à ces systèmes et d'apporter une solution originale, adaptée aux attentes des utilisateurs.

Face à cette problématique, nous avons proposé pour le contexte relationnel, une approche de génération automatique des requêtes de médiation. Etant donné le schéma d'une relation de médiation, de dépendances fonctionnelles définies sur cette relation, d'assertions de correspondance linguistique existant entre les sources et le schéma de médiation, et d'assertions intra-source et inter-source reliant les relations sources entre elles, notre algorithme peut produire un ensemble de requêtes potentiel calculant cette relation.

La complexité du processus de génération de requêtes est accrue lorsqu'on tient compte de l'hétérogénéité des données. Pour résoudre la problématique liée à la définition de requêtes de médiation dans un environnement hétérogène, notre algorithme de génération de requêtes de médiation tient compte des conflits liés à l'hétérogénéité des sources. Il détecte ces conflits et ajoute automatiquement dans la requête SQL générée la fonction de transformation appropriée.

Notre algorithme permet aussi de détecter pendant l'exploration les requêtes invalides et de ne pas les considérer comme des requêtes candidates au calcul d'une instance du schéma global. L'identification des requêtes invalides est guidée par des règles d'invalidation spécifiées sur un chemin de calcul. Ces règles d'invalidation permettent, pour une relation de médiation donnée d'identifier les chemins de calcul qui ne préservent pas les dépendances fonctionnelles définies sur cette relation.

Nos algorithmes s'appuient sur un ensemble de connaissances que nous avons regroupé dans une base de méta-connaissances. Cette base est constituée de :

- La description du schéma de médiation comportant les schéma de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre relations, et pour chaque attribut d'une relation son type étendu,
- La description du schéma local à chaque source de données comportant les schémas de relations, les clés des relations, les dépendances fonctionnelles éventuelles, les contraintes référentielles entre les relations d'une même source, et pour chaque attribut d'une relation son type étendu.
- Les assertions de correspondances linguistiques entre les attributs sources et les attributs de médiation (synonymie, abréviations, équivalences terminologiques des noms des attributs).
- Les assertions de correspondances linguistiques entre les attributs de sources différentes.
- Une librairie de fonctions de transformations de données.
- Un dictionnaire linguistique.

Nous avons développé un prototype qui permet de générer automatiquement des requêtes SQL et qui tient compte aussi de l'hétérogénéité des sources de données. Il compte essentiellement six modules : 1) *interface graphique* ; 2) *recherche des relations de mapping* ; 3) *recherche des relations de transitions* ; 4) *recherche des opérations de jointures* ; 5) *recherche des chemins de calcul* ; 6) *génération de requêtes de médiation*.

Au-delà des problèmes de génération automatique de requêtes de médiation, et de prise en compte des conflits liés à l'hétérogénéité des sources que nous savons résoudre, notre outil permet aussi le passage à l'échelle. Compte tenu d'une évaluation théorique et pratique, les résultats des tests montrent qu'il est possible de générer des requêtes de médiation dans des temps raisonnables (en moyenne 30 secondes) pour des graphes connexes acycliques de grandes tailles, et pour des graphes connexes cycliques de tailles moyennes.

7-2- Perspectives

a) Intégration de nouveaux opérateurs dans les requêtes de médiation.

L'intégration des fonctions de transformations dans les requêtes de médiation permet de résoudre certains conflits sémantiques liés aux données mais ne répond pas à tous les conflits. L'algèbre relationnelle elle même ne permet pas de faire tous les calculs nécessaires à certains schémas de médiation ; de nouveaux opérateurs sont donc nécessaires.
Nous avons introduit dans cette thèse deux nouveaux opérateurs *Explode* et *Fusion* pour initier une démarche de génération de requêtes de médiation plus approfondie mais ne sont pas encore pris en compte par le processus de génération de requêtes de médiation actuel. Une implémentation de ces opérateurs a été réalisée dans le cadre d'un projet d'étudiants en école d'ingénieurs. Ces opérateurs ne sont pas encore intégrés dans le processus de génération de requêtes de médiation actuel. Il reste donc à explorer l'intégration de ces nouveaux opérateurs dans les requêtes de médiation.

b) La prise en compte des liens réflexifs

Le processus de génération de requêtes actuel ne traite pas les liens réflexifs entre les relations. Dans la recherche des séquences et des chemins de calcul, il n'est pas possible de passer deux fois par le même sommet, et cela pour éliminer les cycles dans le graphe. La figure suivante illustre un exemple de lien réflexif :

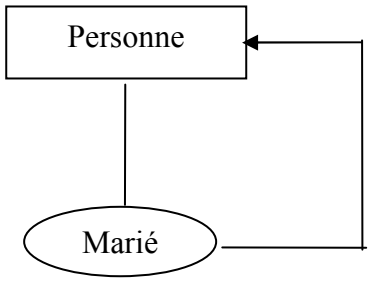
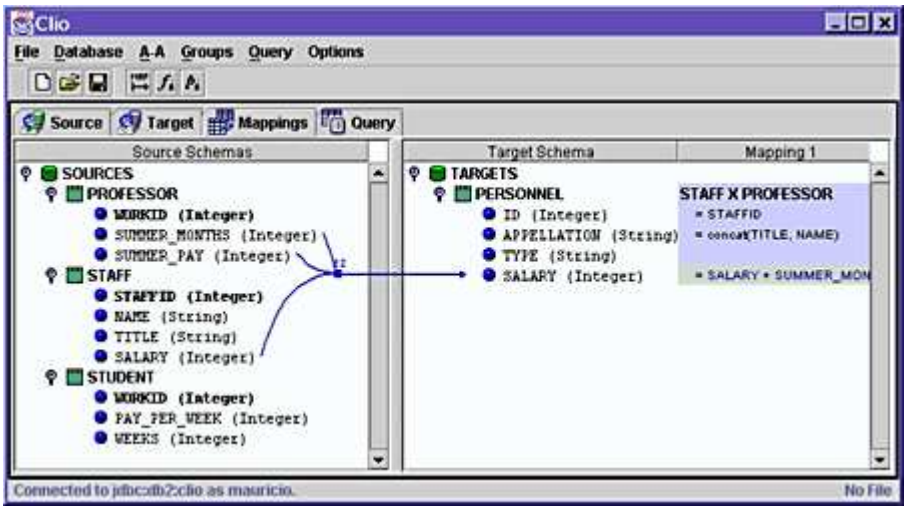


Figure 1 : Exemple de lien réflexif

La non prise en compte de ces liens empêche la génération de certaines requêtes de médiation, par exemple la requête qui calcule la relation de médiation R_m=(NomPrénom-épouse,NomPrénom-époux). Des détails de développement sont présentés dans [Liu 05], il propose une nouvelle version de l'algorithme de recherche des chemins où il est possible de passer plus d'une fois par le même sommet. Cependant le nombre de fois où on passe par le même sommet n'est pas connu au préalable. Il est intéressant d'explorer ce point pour que notre outil fonctionne aussi pour des cas particuliers.

c) La prise en compte des correspondances linguistiques de type 1-n

Seules les correspondances linguistiques de type 1-1 entre le schéma global et un schéma source, ou entre deux schémas sources sont détectées automatiquement par notre outil. Cependant les correspondances linguistiques de type 1-n ne sont pas recherchées. Dans [Dhamankar et al. 04] ces correspondances sont recherchées automatiquement. Dans [Miller et al. 00] elles sont prédéfinies manuellement, par exemple la correspondance `Salary = Salary + Summer_Pay`. La figure suivante illustre cet exemple:



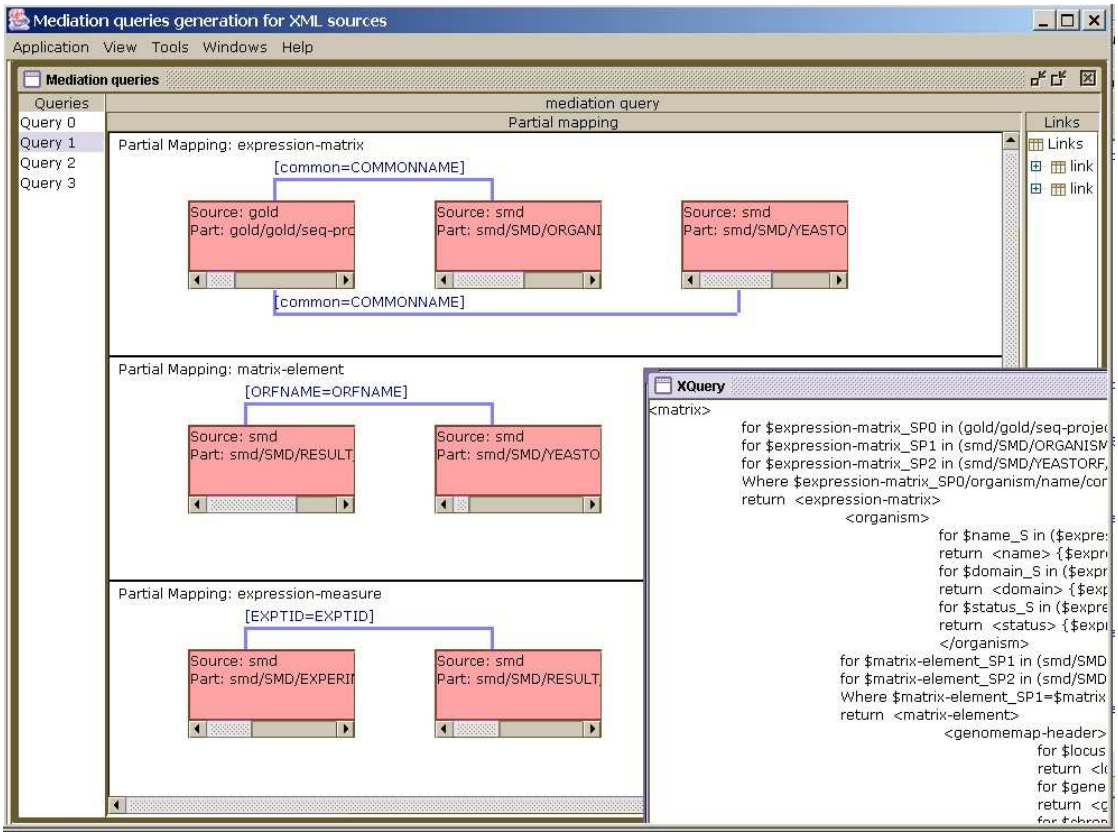
d) Adaptation du système de génération de requêtes de médiation par rapport aux attentes des utilisateurs

Actuellement une extension de l'approche de génération de requêtes de médiation est en cours. Il est possible à partir de données semi structurées de générer des requêtes de médiation XQuery, d'évaluer la qualité des données retournées aux utilisateurs, et d'exprimer leurs préférences sous la forme de profils. Ce système compte trois modules : *interface de définition*, *génération de requêtes* et *adaptabilité*. Toutes les méta-données nécessaires sont stockées dans une méta-base. *L'interface de définition* permet de saisir des méta-données (schéma global, schéma sources, etc.) et d'interagir avec le module de génération de requêtes et le module d'adaptabilité. Le module de *génération de requêtes* est implémenté pour un contexte relationnel et XML. Le module *d'adaptabilité* comprend la gestion de profiles qui vise à exprimer les préférences des utilisateurs, et d'évaluation de la qualité qui vise à évaluer la qualité des données afin de délivrer des résultats adaptés à leur préférences. Ces deux modules communiquent via une méta-base commune. Les résultats produits sont aussi stockés dans la méta-base. Cette extension a fait l'objet d'une démonstration dans [Kostadinov et al. 04], et d'un article de recherche dans [Kostadinov et al. 05].

e) Adaptation du système de génération de requêtes de médiation au contexte XML et Objet

Dans les contextes XML et objet, il émerge un problème supplémentaire qui vient de la structure hiérarchique de données. Pour le contexte XML, [Kedad et Xue 05] propose une extension de l'approche dans le contexte relationnel pour générer des requêtes de médiation XQuery/XSL. Cette approche compte trois étapes principales : (1) décomposition du schéma de médiation en plusieurs sous arbres, appelés *sous-arbres de médiation*. La racine de chaque sous-arbre est un nœud multivalué et tous les autres nœuds sont monovalués. (2) recherche des différentes façons de définir chaque sous-arbre de médiation à partir des schémas sources, appelés mappings partiels. La détermination des mappings partiels pour chaque sous-arbre de médiation est faite

indépendamment des autres. Elle partage les mêmes étapes que pour le contexte relationnel. (3) combinaison de mappings partiels des différents sous-arbres de médiation, pour générer les requêtes de médiation. La figure suivante illustre un exemple de résultat, à la fois graphique et textuel, obtenu par le générateur de requtes XQuery :



[Lui 05] introduit une approche similaire aux contextes XML et relationnel pour initier une démarche de génération automatique de requêtes de médiation pour le contexte objet. Cette approche compte trois phases : (1) décomposition de la relation source objet en plusieurs relations ; (2) transformation de la relation de médiation objet; (3) génération des requêtes de médiation, cette étape utilise le système existant de génération de requêtes de médiation dans le contexte relationnel. (4) transformation de la requête SQL en requête objet.

Références bibliographiques

[Bouzeghoub et Kedad 99]: Bouzeghoub M., Kedad Z., "Discovery View Expressions from a Multi-Source information System", Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (COOPIS), Edinburgh, Scotland, p.57-68, 1999.

[Bouzeghoub et al. 02]: Bouzeghoub M., Farias Loscio B., Kedad Z., Soukane A., "Heterogeneous Data Source Integration and Evolution", DEXA, p.751-757, 2002.

[Bouzeghoub et al. 02]: Bouzeghoub M., Kedad Z., Soukane A., "génération de requêtes de médiation intégrant le nettoyage de données", RSTI série ISI-NIS Ingénierie des Systèmes d'Information, volume 7, N°3, p.39-66, 2002.

[Bouzeghoub et al. 04]: Bouzeghoub M., Kedad Z., Soukane A., "Improving Médiation Query Generation using Constraints and métadata", Bases de Données Avancées (BDA), Montpellier, 2004.

[Bressan et al. 99]: Bressan S., Madnick S.E., Siegel M.D., "Context Interchange: New features and formalisms for the intelligent integration of information", ACM Transactions on Informations Systems, Volume 17, N°3, p.270-298.

[Calvanese et al. 99]: Calvanese D., De Giacomo G., Lenzerini M., Nardi D., Rosati R., "A Principled Approach To Data Integration and Reconciliation In Data Warehousing", Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW), Heidelberg, Germany, 1999.

[Chatterjee et Segev 91]: Chatterjee A., Segev A., " Data Manipulation In Heterogeneous Databases", SIGMOD Record, volume 20, N°4, p.64-68, 1991.

[Chen et al. 03]: Chen Y.B., Ling T.W., Lee L.M., "Automatic generation of XQuery View definitions from ORA-SS Views", International Conference on Conceptual Modeling (ER), p.158-171, 2003.

[Claypool et Rundensteiner 03]: Claypool K. T., Rundensteiner E. A, "Gangam: A Transformation Modeling Framework" Proceedings of Eighth Int Conf on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan, 47-54, 2003.

[Dhamankar et al. 04]: Dhamankar R., Lee Y., Doan A., "iMAP : Discovery Complex Semantic Matches between Databases Schemas", Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, p.383-394, 2004.

[Fan et al. 00]: Fan W., Lu H., Chueng D., Madnick S.E., "Discovering and Reconciling Value Conflicts for Numerical Data Integration", Information Systems Journal, Volume 26, N°8, p.635-656, 2001.

[Galhardas et al. 00]: Galhardas H., Florescu D., Shasha D., Simon E., Saita C., "Declarative Data Cleaning: Language, Model and Algorithms", INRIA Report N° 4149, 2001.

[Galhardas et al. 01]: Galhardas H., "Nettoyage de Données : Modèle, Langage Déclaratif et Algorithmes", Thèse de Doctorat, Université de Versailles St Quentin en Yvelines, 2001.

[Hellerstein et al. 01]: Hellerstein J.M., Raman V., "Potter's Wheel: An Interactive Data Cleaning System", Proceedings of the 27th International Conference on Very Large Databases (VLDB) , Roma, Italy, p. 381-390, 2001.

[Hernandez et Stolfo 95]: Hernandez M.A., Stolfo S.J., "The Merge/Purge Problem for Large Databases", Proceedings of the ACM SIGMOD International Conference on Data Management, San Jose, California, May 22-25, p. 127-138, 1995.

[Hass et al. 00]: Haas L. M., Hernández M. A, Miller R.J., "Schema Mapping as Query Discovery" Proc of the 26th Int Conf on Very Large Data Bases (VLDB'00), Cairo, Egypt, p.77-88, 2000.

[Karayannidis et al. 01]: Karayannidis N., Vassiliadis P., Vagena Z., Skiadopoulos S., Sellis T.,"ARKTOS: towards the modeling, design, control and execution of ETL processes", Information Systems Journal, Vol 26, N°8, p. 537-561, 2001.

[Kedad et Métais 99]: Kedad Z., Métais E., "Dealing with semantique heterogeneity during data integration", International Conference on Conceptual Modeling (ER), 1999.

[Kedad et Xue 05]: Kedad Z., Xue X., "Mapping Generation for XML Data Sources: a General Framework", Proc of the Int Workshop on Challenges in Web Information Retrieval and Integration (WIRI'05), in conjunction with the 21st Int. Conf. on Data Engineering (ICDE'05), Tokyo, Japan, 2005.

[Kostadinov et al. 04]: Kostadinov D., Peralta V., Soukane A., Xue X., " Système adaptatif d'aide à la génération de requêtes de médiation", Bases de Données Avancées (BDA), 2004.

[Kostadinov et al. 05]: Kostadinov D., Peralta V., Soukane A., Xue X., " Intégration de données hétérogènes basée sur la qualité", Actes du XXIIIème Congrès INFORSID, 2005.

[Lerner 00]: Lerner BS., "A model for compound type changes encountered in schema evolution, ACM Transactions on Database Systems (TODS) 25(1), p.83-127, 2000.

[Li et Clifton 94]: Li W., Clifton C., "Semantic integration in heterogeneous databases using neural networks", Proceeding of the 20th VLDB conference Santiago, Chile, 1994.

[Liu 05] : Liu P.; " Génération des requêtes de médiation : Optimisation et amélioration", rapport de stage de DEA, université de Versailles Saint-Quentin en Yvelines, laboratoire PRiSM, équipe SIAL, 2005.

[Martins et Al. 98] : Martins E. Q. V., Pascoal M. M. B., Santos J. L. E., "The K Shortest Paths Problem" **,** Research Report, CISUC, 1998.

[Melnik et al. 02]: Melnik S., Garcia-Molina H., Rahm E., "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching", ICDE, p.117-128, 2002.

[Miller et al. 00]: Miller R.J., Haas L., Hernandez M.A., "Schema Mapping as query discovery", In proc of the Int Conf on Very Large Data Bases (VLDB), Cairo, Egypt, p.77-88, ,2000.

[Miller et al. 01]: Miller R.J., Hernandez M.A., Haas L.Yan L., "The Clio Project: Managing Heterogeneity", SIGMOD Record 30, P.78-83, 2001.

[Moulton et al. 02]: Moulton A., Madnick S.E., Siegel M.D., "Semantic Interoperability in the fixed income securities industry: A knowledge representation architecture for dynamic integration of web-based information", Proc of the 36th Hawaii Int Conf on System Sciences (HICSS'03).

[Popa et al. 02]: Popa L., Velegrakis Y., Miller R.J., Hernandez M.A., Fagin R. "Translating web data", Proc of the 28th Int Conf on Very Large Data Bases (VLDB'02), Hong Kong, China, p. 598-609, 2002.

[Popa et Yu 04]: Popa L., Yu C., "Constraint-based XML query rewriting for data integration", Proc of Int Conf ACM SIGMOD (SIGMOD'04), Paris, France, p.371-382, 2004.

[Rahm et Hai Do 99]: Rahm E., Hai Do H., "Data cleaning: problems and current approaches", IEEE 1999.

[Rahm et Bernstein 01]: Rahm E., Bernstein P., "A survey of approaches to automatic schema matching, VLDB Journal, 10(4), p.334-350, 2001.

[Sciore et al. 94]: Sciore E., Siegel M., Rosenthal A., "Using semantic values to facilitate interoperability among heterogeneous information systems", ACM Trans Database Syst 19, p.254-290, 1994.

[Sapia et al. 99]: Sapia C., Hofling G., Muller M., Hausdort C., Stoyan H., Grimmer U.,"On supporting the data warehouse design by data mining techniques", GI-Workshop data mining an data warehousing 1999.

[Siegel et Madnick 91]: Siegel M., Madnick S.E., A Metadata approach to resolving semantic conflicts, In proceedings of the 17th Conference on Very Large Data Bases, Barcelone, Spain, p.133-145, 1991.

[Tejada et al. 00]: Tejada S., Knoblock C.A., Minton S., "Learning object identification rules for information integration", Information Systems Journal, volume 26, N°8, p. 607-633 , 2001.

[Zamboulis 04]: Zamboulis L., Poulovassilis A.: "XML data integration by Graph Restructuring", Proc of the 21st Annual British National Conf on Databases (BNCOD21), Edinburgh, p. 57-71, 2004.

[Zhan et al. 00]: Zhan K., Shasha D., Wang JTL:
Http://cs.nyu.edu/cs/faculty/shasha/papers/agm.html,
Http://cs.nyu.edu/cs/faculty/shasha/papers/tree.html,
Http://cs.nyu.edu/cs/faculty/shasha/papers/treesearch.html**,**

[Zweigenbaum 98]: Zweigenbaum P., "Automatic Acquisition of Morphological Knowledge for Medical Language Processing", AIMDM, p. 416-422, 1999.

Annexes

Annexe 1 : Algorithme de recherche de mapping étendu

```
Recherche de MappingEtendu (Rm, S, Me)
Entrée : Rm : la relation de médiation
         S  : l'ensemble de sources de données
Sortie : Me :  l'ensemble de relations de mapping étendu

1.  Me : = ∅
2. Pour chaque source S ∈ S
3.    Pour chaque relation source R ∈ S
4.      E : = ∅ // E : ensemble d'attributs communs entre Rm et R
5.       Pour  chaque B ∈ R
6.         Pour chaque A ∈ Rm
7.           Si A ≅ B
8.           Alors E : = E ∪ {B}
9.           FinSi
10.        FinPour
11.      FinPour
12.      Si E ≠∅
13.      alors
14.      RechercheClé (R, K)//recherche de la clé primaire K dans A
15.            Si K ∉E
16.            alors E  = E ∪ {(K,∅)}
17.            FinSi
18.            Pour chaque assertion a ∈ A /a = R.B ⊆ R'.B' /B ∈ R
19.               E : = E ∪ {(B,∅)}
20.            FinPour
21.            Me = Me ∪{ΠE R (Y)}
22.      FinSi
23.    FinPour
24.  FinPour
Fin Recherche de MappingEtendu
```

Algorithme1 : **algorithme de recherche de mapping étendu**

Annexe 2 : Algorithme de recherche des relations de transition

Notation	Explication
-(T_i,T_j)	une paire de relations de mapping étendu
-R_i, R_j	les relations sources pertinentes au calcul de T_i et T_j
-$S_{traitée}$	l'ensemble de relations sources traitées
-old.rhs.rel	la relation de droite de l'assertion old
-old.rhs.attr	l'attribut de droite de l'assertion old
-courant.lhs.attr	l'attribut de gauche de l'assertion courant

```
Recherche de MappingTransition (Me,SeqTrouvée, Mt)
Entrée : SeqTrouvée : l'ensemble de séquences d'assertions
         Me: l'ensemble de relations de mapping étendus
Sortie : Mt: l'ensemble de relations de mapping de transition

1. Mt : = ∅
2. Pour  chaque paire (Ti,Tj) ∈ Me / Ti ∩ Tj = ∅
3.  RechercheSéquence (Ri, Rj, A, SeqCourante, SeqTrouvée)
4.     Si SeqTrouvée ≠ ∅
5.         Pour chaque séquence s ∈ SeqTrouvée
6.             Pour chaque assertion a' ∈ s
7.                 Old : = 1ére assertion
8.                 Courant : = 2éme assertion
9.                 Tant que courant ≠  fin
10.                  Mt : = Mt ∪{Π old.rhs.attr,courant.lhs.attr(old.rhs.rel)}
11.                  Old : = courant
12.                  Courant : = assertion_suivante
13.                FinTanque
14.             FinPour
15.         FinPour
16.    FinSi
17.  FinPour
Fin Recherche de MappingTransition
```

Algorithme 2 : Algorithme de recherche de mapping de transition

Annexe 3 : Algorithme de recherche des séquences d'assertions

```
RechercheSéquence (Ri, Rj, A, SeqCourante, SeqTrouvée)
Entrée : Ri : la relation source
         Rj : la relation cible
         A : l'ensemble d'assertions
         SeqCourante : la séquence d'assertions courante
Sortie : SeqTrouvée : l'ensemble de séquences d'assertions trouvées

1. Si Ri ≠ Rj alors
2.      Straitée : = Straitée  ∪ {Ri}
3.           Si Ri ∈ Saut alors
4.              Pour tout assertion a ∈ A / a.lhs.rel = Ri
                                          ou a.rhs.rel = Ri
5.                   Si a.lhs.rel = Ri alors
6.                       Si a.rhs.rel ∉ Straitée  alors
7.                       RechercheSequence a. rhs.rel, Rj,
                                             SeqCourante‖a)
8.                       FinSi
9.                   Sinon
10.                  Si  a.rhs.rel = Ri alors
11.                    Si a.lhs.rel ∉ Straitée  alors
12.                      RechercheSequence(a.lhs.rel, Rj,
                                   SeqCourante‖a)
13.                    FinSi
14.                  FinSi
15.                FinSi
16.            FinPour
17.         FinSi
18. Sinon
19.   SeqTrouvée  : = SeqTrouvée  ∪ SeqCourante
20. FinSi
Fin RechercheSequence
```

Algorithme 3 : Algorithme de recherche de séquences d'assertions

Annexe 4 : Recherche des séquences d'assertions optimisée

Notation	Explication
-**Count$_{Ri}$**	Le nombre de fois ou une relation R_i est traitée
-elm	Le numéro de la relation traitée
-X	Un ensemble d'éléments correspondants aux numéros des nœuds traités
-h(elm)	La fonction h détermine la relation correspondante à elem
-h^{-1}(R)	La fonction h^{-1} détermine le numéro (elem) correspondant à la relation R
-ξ(elm)	La fonction ξ détermine le numéro qui précède elem

```
RechercheSéquenceOptimisé (Ri, Rj, A, SeqTrouvée)
Entrée : Ri : la relation source
         Rj : la relation cible
         A : l'ensemble d'assertions
Sortie : SeqTrouvée : l'ensemble de séquences d'assertions trouvées

1.Countrj := 0
2.elem := 1
3.h(elem) := Ri
4. X := {elm}
5. SeqTrouvée := ∅
6. Tantque (countrj < K ) et (X ≠ ∅)
7.       k := 1er élément de X
8.       X := X - {elm}
9.       Rs := h(k)
10.      Si Rs = Rj  alors
11.         countrj := countrj + 1
12.         //reconstruire le chemin de Rj à Ri
13.         Rx := h(k)
14.         seq := {Rx}
15.         Tantque Rx ≠ Ri
16.           k : = ξ(k)
17.           Rx := h(k)
18.           seq := seq ∪ { Rx}
19.         FinTanque
20.           SeqTrouvée := SeqTrouvée ∪ {seq}
21.      Sinon
22.           Straitée : = Straitée  ∪ {Rs}
23.           Pour chaque assertion a ∈ A /a.lhs.rel = Rs
                                          ou a.rhs.rel = Rs
24.              Si a.rhs.rel ∉ Straitée  alors
25.                  elm := elm + 1
26.                 ξ(elm) := k
27.                 h(elm) := a.rhs.rel
28.                 X := X ∪ {elm}
29.              FinSi
30.          FinPour
31.      FinSi
32. FinTantque
FinRechercheSéquenceOptimisé
```

Algorithme 4 : Algorithme de recherche de séquences d'assertions optimisée

Annexe 5 : Algorithme de Recherche de Mapping de Transition Optimisée

```
Recherche de MappingTransition (Me, SeqTrouvée, Mt)
Entrée : SeqTrouvée : l'ensemble de séquences d'assertions
         Me: l'ensemble de relations de mapping étendus
Sortie : Mt    : l'ensemble de relations de mapping de transition

1. Mt : = ∅
2. Pour  chaque paire (Ti,Tj) ∈ Me / Ti ∩ Tj = ∅
3.  RechercheSéquenceOptimisée (Ri, Rj, A, SeqCourante, SeqTrouvée)
4.  Si SeqTrouvée ≠ ∅
5.      Pour chaque séquence s ∈ SeqTrouvée
6.          Pour chaque assertion a' ∈ s
7.              Old : = 1ére assertion
8.              Courant : = 2éme assertion
9.              Tant que courant ≠  fin
10.                Mt : = Mt ∪{Π old.rhs.attr,courant.lhs.attr(old.rhs.rel)}
11.                Old : = courant
12.                Courant : = assertion_suivante
13.             FinTanque
14.         FinPour
15.     FinPour
16. FinSi
17.FinPour
Fin Recherche de MappingTransition
```

Algorithme 5: **Algorithme de recherche de mapping de transition optimisée**

Annexe 6 : Algorithme de Recherche du graphe d'opérations

```
Recherche de GrapheOpérations (Me, Mt,J)
Entrée : Me : l'ensemble de relations de mapping étendus
         Mt : l'ensemble de relations de mapping de transition
Sortie : J : l'ensemble de Jointures

1. J : = ∅
2. Pour  chaque paire (Ti,Tj) dans Me et Mt  tel que Ti ∩ Tj ≠ ∅
3.    Si Ti et Tj appartiennent à la même source
4.    Alors J := J ∪ {j = Ti.B  ⋈  Tj.B}
      // Création d'une opération de jointure entre Ti et Tj sur le
     critère de jointure Ti.B = Tj.B
5.   Sinon J := J ∪ {j = Ti.B  ⋈  Tj.B'}
6.   FinSi
7. FinPour
Fin Recherche de GrapheOpérations
```

Algorithme 6 : **Algorithme de recherche du graphe d'opérations**

Annexe 7 : Algorithme de recherche des chemins de calcul

```
RechercheChemin (XCourant, X, G_RM, ChemCourant, ChemTrouvé)

Entrée : XCourant : l'ensemble des attributs du chemin courant
         X : l'ensemble des attributs de la vue R_m.
         G_RM : le graphe d'opérations
         ChemCourant : le chemin de jointures courant
Sortie : ChemTrouvé : l'ensemble de chemins de jointures trouvées

1.    Si X ⊄ Xcourant alors
2.    S_traitée : = S_traitée  ∪ {c.lhs.rel,c.rhs.rel}
3.        Pour toute jointure j ∈ G_RM
4.           Si chemCourant ≠ ∅
5.              Si j.lhs.rel = c.lhs.rel ou  j.lhs.rel = c.rhs.rel alors
6.                Si j.rhs.rel ∉ S_traitée  alors
7.                   Xcourant := Xcourant ∪ X_j
8.                   RechercheChemin (XCourant, X, ChemCourant ‖ j)
9.                FinSi
10.             Sinon
11.               Si j.rhs.rel = c.lhs.rel ou  j.rhs.rel = c.rhs.rel alors
12.                 Si j.lhs.rel ∉ S_traitée  alors
13.                    Xcourant := Xcourant ∪ X_j
14.                    RechercheChemin (XCourant, X, ChemCourant ‖ j)
15.                FinSi
16.              FinSi
17.           FinSi
18.         Sinon
19.            Xcourant := Xcourant ∪ X_j
20.            RechercheChemin (XCourant, X, ChemCourant ‖ j)
21.        FinSi
22.     FinPour
23.   Sinon
24.     ChemTrouvé : = ChemTrouvé  ∪ ChemCourant
25.   FinSi
FinRechercheChemin
```

Algorithme 7 : Algorithme de recherche des chemins de calcul

Annexe 8 : Algorithme de recherche des chemins de calcul valides

```
RechercheChemin (XCourant, X, J, ChemCourant, ChemTrouvé)

Entrée : XCourant : l'ensemble des attributs du chemin courant
         X : l'ensemble des attributs de la vue R_m.
         J  : l'ensemble de jointures
         ChemCourant : le chemin de jointures courant
Sortie : ChemTrouvé : l'ensemble de chemins de jointures trouvées

1.   Si X ⊄ Xcourant alors
2.    S_traitée : = S_traitée  ∪ {c.lhs.rel,c.rhs.rel}
3.       Pour toute jointure j ∈ J
4.          Si chemCourant ≠ ∅
5.              Si j.lhs.rel = c.lhs.rel ou  j.lhs.rel = c.rhs.rel alors
6.                Si j.rhs.rel ∉ S_traitée  alors
7.                  RechercheDF
8.                  Xcourant := Xcourant ∪ Xj
9.                  RechercheChemin (XCourant, X, ChemCourant‖j)
10                FinSi
11.             Sinon
12.              Si j.rhs.rel = c.lhs.rel ou  j.rhs.rel = c.rhs.rel alors
13.              Si j.lhs.rel ∉ S_traitée  alors
14.                 RechercheDF
15.                 Xcourant := Xcourant ∪ Xj
16.                 RechercheChemin (XCourant, X, ChemCourant‖j)
17.              FinSi
18.             FinSi
19.           FinSi
20.        Sinon
21.           RechercheDF
22.           Xcourant := Xcourant ∪ Xj
23.           RechercheChemin (XCourant, X, ChemCourant‖j)
24.       FinSi
25.     FinPour
26.  Sinon
27.   ChemTrouvé  : = ChemTrouvé  ∪ ChemCourant
28.  FinSi
FinRechercheChemin
```

Algorithme 8 : Algorithme de recherche des chemins de calcul valides

Annexe 9 : Algorithme de recherche des relations de mapping étendu dans un environnement hétérogène

```
Recherche de MappingEtendu (Rm, S, Me)
Entrée : Rm  : la relation de médiation
         S   : l'ensemble de sources de données
Sortie : Me  :  l'ensemble de relations de mapping étendu

1.  Me : = ∅
2.  Pour chaque source S ∈ S
3.    Pour chaque relation source R ∈ S
4.     E : = ∅ // E : ensemble d'attributs communs entre Rm et R
5.       Pour  chaque B ∈ R
6.        Pour chaque A ∈ Rm
7.          Si Compare (A, B, CF) = vrai
8.          Alors E : = E ∪ {(B, CF )}
9.          FinSi
10.       FinPour
11.      FinPour
12.      Si E ≠∅
13.      alors
14.       RechercheClé (R, K)//recherche de la clé primaire K dans A
15.       Si K ∉E
16.       alors E  = E ∪ {(K,∅)}
17.       FinSi
18.       Pour chaque assertion a ∈ A /a = R.B ⊆ R'.B' /B ∈ R
19.         E : = E ∪ {(B,∅)}
20.       FinPour
21.       Me = Me ∪{ΠE R (Y)}
22.      FinSi
23.   FinPour
24. FinPour
Fin Recherche de MappingEtendu
```

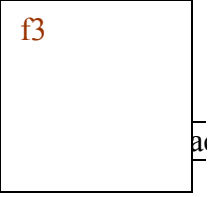
Algorithme 9 : Algorithme de recherche des mapping étendu

Annexe10 : Algorithme de recherche des relations de mapping de transition dans un environnement hétérogène

```
Recherche de MappingTransition (SeqTrouvée,Me, Mt)
Entrée : Rm (X): la relation de médiation
         S     : l'ensemble de sources de données
         Me    : l'ensemble de relations de mapping étendus
Sortie : Mt    : l'ensemble de relations de mapping de transition

1. Mt : = ∅
2. Pour  chaque paire (Ti,Tj) ∈ Me / Ti ∩ Tj = ∅
3.     Si Ti et Tj ∈ Si Alors
4.         Select * from Assertion where a.rhs.src = a.lhs.src
5.         RechercheSéquenceOpt (Ri, Rj, A, SeqCourante, SeqTrouvée)
6.          Si SeqTrouvée ≠ ∅
7.            Pour chaque séquence s ∈ SeqTrouvée
8.               Pour chaque assertion a' ∈ s
9.                   Old : = 1ére assertion
10.                  Courant : = 2éme assertion
11                   Tant que courant ≠  fin
12.                    Mt : = Mt ∪{Π old.rhs.attr,courant.lhs.attr(old.rhs.rel)}
13.                    Old : = courant
14.                    Courant : = assertion_suivante
15.                  FinTanque
16.              FinPour
17.            FinPour
18.          FinSi
19.    Sinon
20.       Select * from Assertion where a.rhs.src = a.lhs.src
21.       RechercheSéquenceOpt (Ri, Rj, A, SeqCourante, SeqTrouvée)
22.          Si SeqTrouvée ≠ ∅
23.            Pour chaque séquence s ∈ SeqTrouvée
24.              Pour chaque assertion a ∈ s
25.                 Compare (a.rhs.attr,a.lhs.attr, CF)
26.                   Old : = 1ére assertion
27.                   Courant : = 2éme assertion
28.                   Tant que courant ≠  fin
29.                     Mt : = Mt ∪{Π old.rhs.attr,courant.lhs.attr(old.rhs.rel)}
30.                     Old : = courant
31.                     Courant : = assertion_suivante
32.                   FinTanque
33.              FinPour
34.            FinPour
35.          FinSi
36.     FinSi
36. FinPour
Fin RechercheTransition
```

Algorithm e10 : Algorithme de recherche des mapping de transition


f3