FlexIS: vers un système d'intégration d'information flexible

P Colomb, H Jaudoin

FlexIS: vers un système d'intégration d'information flexible

P. Colomb[1], et H. Jaudoin[2]

[1]LIMOS - CNRS UMR 6158, Université Blaise Pascal, France

email: colomb@isima.fr

LIMOS, 24 Avenue des Landais, 63177 Aubieres

[2]IRISA-ENSSAT, Université de Rennes 1, France

email: jaudoin@enssat.fr

ENSSAT, 6 rue Kerampont, 22305 Lannion

June 11, 2008

Résumé: Cette démonstration présente FLEXIS un système de médiation flexible de type LAV. FLEXIS permet de réécrire les requêtes des utilisateurs en utilisant les vues, de façon flexible, sur la base des contraintes d'intervalles présentes dans la requête et les vues. Les réécritures obtenues sont pondérées d'un degré qui reflète la probabilité que les tuples issues de celles-ci respectent les contraintes d'intervalles de la requête. FLEXIS présente à l'utilisateur les k meilleures réécritures de la requête, ou celles dépassant un certain seuil, dans l'ordre décroissant des degrés.

Mots clés: Système d'intégration, réécriture de requêtes en termes de vues, flexibilité.

1 Introduction

Les systèmes d'intégration d'information fournissent une interface d'interrogation uniforme à un ensemble de sources de données distribuées [4]. Ces systèmes font l'objet de nombreuses études dans la communauté de recherche en bases de données du fait du besoin croissant du partage d'information sur le web, mais aussi, du fait de leur aide à la définition des stratégies commerciales des entreprises [3].

Une approche possible d'intégration d'information est celle de la médiation dans laquelle les données restent stockées au niveau des sources. Les systèmes de médiation reposent sur le paradigme *médiateur/wrapper* dans lequel les requêtes réalisables sur les sources de données sont transformées à l'aide des wrappers en *vues* logiques stockées au sein du médiateur. En plus des vues, le médiateur dispose d'un *schéma global* qui donne une interface uniforme sur l'ensemble des vues et donc des sources de données, et sur lequel les requêtes sont exprimées. Il existe deux approches de médiation qui impactent la façon de résoudre les requêtes. Dans l'approche Global As View (GAV), le schéma global est défini en termes des vues logiques tandis que dans l'approche Local As View (LAV), ce sont les vues qui sont décrites en termes des

relations du schéma global. L'approche GAV est peu extensible dans le sens où l'ajout de toute nouvelle source conduit à une redéfinition du schéma global et est donc peu adaptée aux environnements d'intégration dynamiques comme le web. L'approche LAV est plus extensible mais la résolution de requêtes dans ce contexte est connue pour être en général plus délicate.

Le problème de répondre à des requêtes dans les systèmes de médiation a été largement étudié durant cette dernière décennie [4]. L'investigation de ce problème dans le cadre de l'approche LAV (Local As View) a conduit à de nombreux résultats théoriques [1, 4]. Ces travaux ont notamment montré que le traitement des requêtes dans ce cadre est lié au problème général de répondre aux requêtes en utilisant des vues. Dans un tel contexte, la sémantique des requêtes peut être formalisée en termes de *réponses certaines* [1]. Une réponse certaine à une requête Q exprimée sur le schéma global d'un système de médiation est une réponse à Q sur une instance de base de données définie sur le schéma global, compatible avec le contenu des sources de données. Le traitement d'une requête Q dans un système de médiation suivant une approche LAV peut alors être formulé comme le problème de calculer toutes les réponses certaines à Q.

Une technique pour calculer les réponses certaines à une requête posée à un système de médiation est celle de *la réécriture de requête en termes de vues* qui peut se définir comme suit. Soit une requête Q exprimée en termes d'un schéma global, les sources de données pertinentes à la construction des réponses sont sélectionnées via un algorithme de réécriture qui reformule la requête en une expression de requête équivalente ou contenue dans Q, et qui utilise uniquement les vues. Ces réécritures doivent capturer toutes les contraintes imposées par la requête Q afin de fournir uniquement des réponses correctes à Q.

Contexte et motivation du prototype

Une des tâches du projet FORUM du programme ARA - MDMSA - 2005 est d'étudier le problème de la réécriture de requêtes dans les environnements ouverts à large échelle, de façon flexible. Dans ce type d'environnement où les sources de données sont autonomes, il est difficile de trouver des vues qui valident les contraintes d'intervalles imposées par la requête. Une manière d'éviter l'élimination systématique des vues du processus de réécriture parce qu'elles ne vérifient pas exactement les contraintes alors qu'elles pourraient cependant retourner un ensemble conséquent de réponses correctes, est d'assouplir la notion de réponses certaines. Une sémantique facilement exploitable des réponses dans ce contexte est celle des *réponses probables*. Les réécritures capables de fournir de telles réponses peuvent être calculées sur la base des contraintes d'intervalle apparaissant dans les requêtes et les vues. Les réécritures sont alors associées à un degré qui reflète la probabilité que les tuples issus de celles-ci soient des réponses correctes à la requête. En d'autre termes, sous l'hypothèse d'une distribution uniforme des valeurs, une réécriture ayant pour degré $\alpha = 0.25$ a 25% de chance de retourner des réponses correctes à la requête. On peut également remarquer que les environnements à large échelle disposent d'un nombre important de sources de données et donc de vues. Le nombre de réécritures peut être par conséquent très élevé. Les probabilités associées aux réécritures permettent alors de les discriminer puis de n'évaluer que les meilleures d'entre elles.

Dans cette démonstration, nous présentons un système de médiation de type LAV, appelé FLEXIS (Flexible Integration System), qui calcule les réponses probables d'une requête Q.

Concrètement, le médiateur de FLEXIS étend l'algorithme de réécriture du MINICON [6] et calcule sur la base des contraintes d'intervalle de la requête et des vues, les réécritures de Q susceptibles de fournir des réponses correctes à Q. FLEXIS permet de plus de ne calculer que les k meilleures réécritures, où k est un entier, d'une requête ou celles dépassant un certain seuil et les présente dans l'ordre décroissant de leur probabilité.

Dans la section 2, nous présentons l'architecture de FLEXIS ainsi que son principe de fonctionnement. Dans la section 3, nous décrivons les scenarii que nous testerons lors de la démonstration.

2 Présentation de FlexIS

2.1 Principe général

FLEXIS est un système de médiation suivant une approche LAV. Ainsi, il est formé *(i)* d'un médiateur, *(ii)* de wrappers et, *(iii)* de sources de données. Le médiateur dispose d'un schéma global $\mathcal{S}$, de vues logiques $\mathcal{V}$ et d'un moteur d'évaluation de requêtes qui calcule les réécritures d'une requête à partir de $\mathcal{V}$ puis, qui évalue les réécritures sur les sources afin de générer l'ensemble des réponses à retourner. Le moteur d'évaluation de requête est basé sur un algorithme qui étend celui du MINICON. L'algorithme du MINICON repose sur une approche en deux phases. La première consiste à générer un ensemble de réécritures partielles de la requête utilisateur, appelées MCD. La seconde vise à combiner ces MCDs pour obtenir les réécritures de la requête utilisateur. Dans sa version de base, l'algorithme gère uniquement la réécriture des requêtes impliquant des sélections, des projections et des jointures. Une version étendue est proposée dans l'article original qui permet de prendre en compte des requêtes disposant de contraintes arithmétiques quand elles impliquent uniquement un attribut et une valeur constante. Une nouvelle extension a été proposée dans [2] permettant de résoudre le problème de réécriture contenant des prédicats de comparaison dans le cas général. Cependant, dans les deux cas la solution proposée est peu flexible. En effet, une vue dont les intervalles de valeur ne sont pas compatibles avec ceux de la requête sera éliminée lors de la première phase de l'algorithme.

Dans FLEXIS, les deux phases principales du MINICON ont été étendues afin de sélectionner les vues et de calculer les réécritures de façon plus flexible. La première phase a été modifiée afin de pouvoir pondérer les MCDs à l'aide de probabilités, obtenues à partir des intersections entre les intervalles de valeur présents dans la requête et dans les vues. La seconde est étendue afin de générer uniquement les réécritures qui dépassent un certain seuil fixé par l'utilisateur dans l'ordre décroissant. Cette extension permet également de générer uniquement les k meilleures réécritures.

2.2 Fonctionnement et Implémentation

FLEXIS, décrit en figure 2.2 prend en entrée une requête SQL exprimée en termes du schéma global $\mathcal{S}$. Par exemple, le schéma global peut être formé de trois relations:

$$\mathcal{S} = \{man(name),\ salary(name, s),\ age(name, a)\}.$$

Une requête sur ce schéma pourrait demander les noms des hommes dont l'âge est compris entre 22 et 35 ans et le salaire entre 1200 et 2300 euros:

```
SELECT man.name
FROM man, age, salary
WHERE man.name = age.name AND
      salary.name = man.name AND
      A >= 22 AND A <= 35 AND
      S >= 1200 AND S <= 2300;
```

La requête est transférée au module de réécriture de requête flexible. Celui-ci s'appuie sur le cadre formel des requêtes conjonctives DataLog. Par conséquent, la requête de l'utilisateur doit être convertie en une expression DataLog. Dans le cadre de notre exemple, la requête sera transformée en l'expression suivante:

$Q(N)$:- $Man(N), Age(N, A), Salary(N, S)$,
$A \geq 22, A \leq 35, S \leq 2300, S \geq 1200$

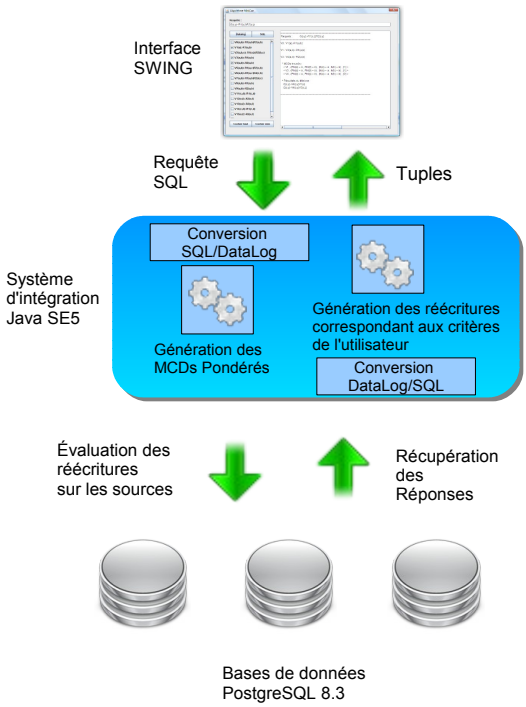
De même, les vues qui décrivent les sources de données vis à vis du schéma global, peuvent être saisies en SQL par l'administrateur, et converties en DataLog. Les vues sont des expressions en termes de $\mathcal{S}$. Par exemple, les vues V_1, V_2, V_3 et V_4 explicitées ci-dessous décrivent les requêtes réalisables sur des sources de données S_1, S_2, S_3 et S_4 respectivement:

$V_1(N_1)$:- $Man(N_1)$
$V_2(N_2)$:- $Age(N_2, A_2), A_2 \leq 40, A_2 \geq 20$
$V_3(N_3)$:- $Salary(N_3, S_3), S_3 \leq 3400, S_3 \geq 2100$
$V_4(N_4)$:- $Age(N_4, A_4), Salary(N_4, S_4), A_4 \leq 38,$
$A_4 \geq 18, S_4 \leq 2400, S_4 \geq 1000$

La vue V_1 indique qu'il est possible de récupérer le nom des hommes sur la source S_1 tandis que la source S_3 donne le nom des individus dont le salaire est compris entre 2100 et 3400 euros. Le module de réécriture flexible prend en entrée les vues et la requête puis retourne les réécritures approchées de la requête sous la forme d'un ensemble de requêtes conjonctives formées uniquement de vues de $\mathcal{V}$, chaque réécriture étant accompagnée d'une probabilité α. Si on considère la requête et les vues de l'exemple en précisant un seuil minimal de 0 et sans fixer le nombre maximum de réécriture, toutes les réécritures flexibles sont calculées et FLEXIS énumèrera les réécritures suivantes:

$Q(N)$:- $V_1(N), V_2(N), V_3(N)$ $\alpha = 0.1$
$Q(N)$:- $V_1(N), V_2(N), V_4(N)$ $\alpha = 0.5$
$Q(N)$:- $V_1(N), V_3(N), V_4(N)$ $\alpha = 0.1$
$Q(N)$:- $V_1(N), V_4(N)$ $\alpha = 0.5$

Ces requêtes DataLog sont ensuite converties en SQL afin d'être évaluées sur les sources de données. Pour terminer, l'ensemble des tuples résultant de cette évaluation est retourné à l'utilisateur.



Concrètement, l'interface graphique du prototype, les convertisseurs DataLog/SQL, le moteur de réécriture flexible et les connexions aux SGBD sont implémentés en Java SE5. FLEXIS se présente sous la forme d'une interface graphique réalisée en SWING. Cette interface permet d'afficher le schéma global et propose différentes fonctionnalités: **(i)** l'expression de requêtes SQL sur le système d'intégration, **(i)** la saisie du seuil minimal ainsi que du nombre maximal de réécritures désirées **(iii)** l'affichage des réponses. Les sources de données sont stockées dans des bases de données PostgreSQL 8.3. L'application permet également d'ajouter une nouvelle vue sur une source de données et d'importer les contraintes de valeurs associées aux vues.

3 Description des scenarii

La démonstration sera exécutée sur des sources de données issues du monde agricole, fournies par le Cemagref dans le cadre du projet FORUM. Nous évaluerons les scenarii suivants:

- La résolution d'une requête en utilisant la réécriture flexible. Nous exécuterons successivement cette requête à l'aide de FLEXIS puis à l'aide de l'implémentation de réécriture classique du MINICON et nous comparerons les résultats obtenus.
- L'ajout d'une nouvelle source de données, i.e., la création par l'administrateur de vues en termes du schéma global. Nous verrons alors comment le prototype met à jour de façon automatique les contraintes d'intervalles à associer aux vues.

Ces deux scenarii permettront de mettre en évidence plusieurs points. En particulier, nous distinguerons l'ensemble de réponses issus de la réécriture classique de celui fourni par la réécriture flexible. De plus, nous montrerons l'intérêt de générer uniquement les k meilleures réécritures d'une requête en présence d'un grand nombre de sources de données. Nous montrerons également la flexibilité de l'approche LAV en donnant la possibilité d'ajouter ou retirer facilement des sources de données, ainsi qu'en proposant un module de mise à jour automatique des contraintes d'intervalles présentent dans les vues.

4 Conclusion et Perspectives

Nous venons de présenter FLEXIS, un système d'intégration de données flexible. Il permet de traiter le problème de répondre à une requête en utilisant des vues avec une approche plus souple que celles proposées jusqu'ici. Ce travail va se poursuivre selon plusieurs axes. Tout d'abord, il serait intéressant de donner la possibilité aux utilisateurs d'utiliser des prédicats flous dans leurs requêtes et ainsi poser sur le système d'intégration des requêtes du type:

```
SELECT man.name
FROM man,age
WHERE man.name = age.name AND
      Young(man.age);
```

On peut également imaginer utiliser d'autre critères que les contraintes arithmétiques pour discriminer les sources, comme par exemple, la qualité des données, la cohérence des sources entre elles, etc.

References

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[2] F. N. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. In *PODS*, pages 209–220, 2002.

[3] T. Bell and R.E. Knox. Content Integration Will Become a Top Priority by 2006., February 2005. Gartner Group, ID Number: G00124957.

[4] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[5] M. Lenzerini. Data Integration : A Theoretical Perspective. In *Proceedings of PODS*, Madison, Wisconsin, 2002.

[6] R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *VLDB*, pages 484–495, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.