Réécriture tolérante de requêtes en termes de vues

Hélène Jaudoin, O Colomb, O Pivert

Réécriture tolérante de requêtes en termes de vues

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

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

email: jaudoin,pivert@enssat.fr

ENSSAT, 6 rue Kerampont, 22305 Lannion

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

email: colomb@isima.fr

LIMOS, 24 Avenue des Landais, 63177 Aubieres

Résumé: Ce papier s'intéresse au problème de réécrire les requêtes en termes de vues de façon tolérante. L'approche proposée est basée sur un appariement approximatif entre les contraintes issues de la requête et celles issues des vues, dans le cas où la requête et les vues contiennent des contraintes arithmétiques, exprimées sous la forme d'intervalles. Dans un tel contexte, les réponses obtenues ne sont plus certaines mais plus ou moins probables. Un algorithme qui calcule les k meilleures réécritures d'une requête donnée est fourni. Des expérimentations ont été conduites. Elles montrent que le coût supplémentaire induit par la nature tolérante du processus de réécriture est tout à fait acceptable.

Mots clés: Système d'intégration, Flexibilité, Réécriture de requête en termes de vues.

1 Introduction

Un système d'intégration d'information fournit une interface d'interrogation uniforme à un ensemble de sources de données distribuées sur un réseau. Le problème de répondre aux requêtes dans les systèmes d'intégration a été intensivement étudié durant cette dernière décennie [11]. Il a été démontré que la sémantique des requêtes dans un tel contexte peut être formalisée en termes de réponses certaines [1]. De manière intuitive, une réponse certaine à une requête Q exprimée en termes d'un schéma global — i.e., l'interface uniforme du système d'intégration— conformément à l'ensemble des instances de sources de données est une réponse à Q pour n'importe quelle base de données définie sur ce schéma et consistante avec les instances des sources. Le problème de répondre aux requêtes dans les systèmes d'intégration d'information consiste alors à calculer toutes leurs réponses certaines. Une technique pour calculer les réponses certaines à une requête dans un système d'intégration suivant une approche LAV (Local-As-View), i.e., où les sources de données sont définies comme des vues sur le schéma global, est de réduire ce problème à celui de la *réécriture de requêtes en termes de vues*. Etant donnée une requête utilisateur exprimée sur le schéma global, les sources de données pertinentes à la résolution de la requête sont sélectionnées au moyen d'un algorithme de réécriture qui permet de reformuler la requête de l'utilisateur Q en une requête équivalente ou en une requête maximalement contenue dans Q dont la définition ne fait référence qu'aux *vues*. Toute réécriture de ce type doit satisfaire toutes les contraintes imposées par la requête Q afin de ne retourner que des réponses correctes à Q.

Cependant, dans le cadre de systèmes d'intégration pour des environnements ouverts comme le web où les sources de données sont autonomes, il devient difficile de trouver des vues qui satisfassent totalement les contraintes de domaine imposées par la requête. Se pose alors la question de relaxer le processus d'appariement entre les domaines de valeurs des vues et ceux des requêtes. Considérons par exemple une requête Q qui demande les *noms* des *personnes* dont l'*âge* est dans l'intervalle $[28, 38]$ et deux vues V_1 et V_2 telles que:
V_1 fournit les *noms* des *personnes* dont l'*âge* est dans $[25, 35]$ et
V_2 fournit les *noms* des *personnes* dont l'*âge* est dans $[36, 46]$.
Les vues V_1 et V_2 ont toutes les deux une contrainte d'intervalle sur l'attribut *âge*. Les deux intervalles ne sont pas inclus dans celui de la requête et donc, l'appariement entre les deux intervalles et celui de la requête sont seulement partiels (imparfaits). De plus, comme V_1 et V_2 fournissent uniquement les noms des personnes, la sélection sur l'attribut âge est impossible, et ni V_1 ni V_2 ne peut être utilisée pour obtenir les réponses certaines à Q. Dans un tel contexte, les algorithmes de réécriture fondés sur la sémantique des réponses certaines ne peuvent proposer de réécriture à la requête, et ne peuvent donc fournir aucune réponse à l'utilisateur.

Dans ce papier, nous supposons un système d'intégration suivant une approche LAV. Nous considérons que les vues et les requêtes sont exprimées dans le cadre formel des requêtes conjonctives et qu'elles comportent des contraintes d'intervalle. Notre proposition est d'utiliser des appariements même approximatifs entre les contraintes définies dans les vues et celles définies dans la requête afin de calculer des réécritures de requête *tolérantes*. Toute réécriture Q' de ce type est associée à un score compris entre 0 et 1 qui reflète la probabilité pour un tuple retourné par Q' de satisfaire la requête initiale Q. Par conséquent, le mécan-

isme de réécriture n'est plus basé sur la notion de réponse certaine mais plutôt sur celle de *réponse probable*. Comme les appariements approximatifs entre les vues et les requêtes sont plus fréquents que les appariements classiques (stricts), le nombre de réécritures tolérantes possibles peut être élevé. Pour résoudre ce problème, nous proposons un algorithme qui génère uniquement les k meilleures réécritures d'une requête donnée. Cet algorithme a été implémenté et des expérimentations sur des données réelles ont été menées. Elles montrent que le coût supplémentaire induit par la nature tolérante de notre approche par rapport à des algorithmes classiques de réécriture est parfaitement acceptable.

Bien que la notion de tolérance soit à la base de l'approche proposée ici, il est important de souligner que le problème abordé est très différent de celui traité dans [7], qui consiste à trouver les k meilleures réponses à une requête en utilisant un mécanisme de réécriture en termes de vues. Dans ce papier, seules des requêtes usuelles sont considérées et l'objectif n'est pas de calculer un ensemble de tuples pondérés de scores, comme dans [7], mais plutôt de calculer les ensembles de réponses correspondant aux *meilleures réécritures* de la requête. Autrement dit, nous ne nous intéressons pas au problème de réécrire des requêtes de type "top-k" mais plutôt au problème de calculer les k meilleures réécritures de requêtes usuelles. Le problème de réécrire les requêtes conjonctives en présence de contraintes d'intervalles en utilisant des vues est bien connu et il a été démontré que le langage de réécriture défini comme l'union des requêtes conjonctives avec des contraintes d'intervalle permet d'obtenir toutes les réponses certaines d'une requête donnée [15, 2]. Cependant, notre contribution réside dans le traitement de ce problème selon une méthode *tolérante*. L'objectif de notre approche diffère également de la référence [8] qui traite le problème de réécrire les requêtes en utilisant des descriptions de vues imprécises. Nos travaux se distinguent également de ceux traités dans [6] où les auteurs s'intéressent au problème de répondre aux requêtes en utilisant les vues lorsque celles-ci contiennent des tuples pondérés d'une probabilité (celle de l'appartenance du tuple à la vue) et que le schéma global est associé à des statistiques portant sur le domaine général sous-jacent. Dans notre approche, l'imprécision résulte de la nature approximative des appariements considérés, elle ne concerne pas la description des données.

Le reste de l'article est organisé comme suit. La section 2 introduit le principe général de notre approche et définit la notion de réponse probable puis de réécriture tolérante et montre quelle est la forme des réécritures à calculer. La section 3 décrit l'algorithme qui calcule les k meilleures réécritures tolérantes d'une requête utilisateur. La section 4 présente des expérimentations et s'intéresse notamment aux performances du prototype comparées à une implémentation d'un algorithme classique de réécriture. La section 5 conclut l'article et discute les expérimentations et l'approche proposée.

2 Réécriture tolérante en utilisant des vues

2.1 Objectif général

Nous étudions le problème de classer les réécritures de requête en termes de vues dans le cadre formel des requêtes conjonctives, en présence de contraintes d'intervalle simples $X \in [a, b]$, où X est une variable et a et b sont des constantes. Une telle requête est une expression de

la forme [1, 15]:

$$Q(Y) : -r_1(Y_1), \ldots, r_n(Y_n), C_1, \ldots, C_m$$

où Q, r_1, ..., r_n sont des noms de prédicats et Y, Y_1,..., Y_n sont des ensembles de variables telles que $Y \subseteq \bigcup_{j=1}^{n} Y_j$. Les atomes $r_1(Y_1), \ldots, r_n(Y_n)$ sont les sous-buts de la requête où $r_1, \ldots, r_n$ sont des noms de relations. L'atome $Q(Y)$ est appelé l'entête de la requête et spécifie le schéma de relation du résultat de la requête. Les variables X qui apparaissent dans l'entête de la requête, i.e., $X \in Y$, sont dites *distinguées* tandis que celles qui n'y apparaissent pas sont dites *existentielles*. Chaque C_i, $i \in [1, m]$, est une contrainte d'intervalle représentée dans la suite par $X \in I_X$ où X appartient à $\bigcup_{j=1}^{n} Y_j$ et I_X dénote un intervalle $[a, b]$. Nous supposons dans cet article que chaque variable est contrainte *au plus* une fois. De plus, nous ne traitons pas le cas particulier des requêtes booléennes [1].

En guise d'exemple, considérons un schéma global $\mathcal{S}$ formé des relations $Person(ssn, name, firstname)$, $Child(ssn, Cfirstname, Cage)$ et $Emp(ssn, job, sal)$, et une requête Q sur $\mathcal{S}$ qui demande le numéro de sécurité sociale et le nom de toutes les personnes qui ont un enfant dont l'âge est compris entre 22 et 35 ans et un salaire compris entre 1200 et 2300 euros. Q peut être spécifiée comme suit:
$Q(nss, name)$: $-Person(nss, name, firstname), Child(nss, Cfirstname, Cage),$
$Emp(nss, job, sal),\ Cage \in A_Q,\ sal \in S_Q$
avec $A_Q = [22, 35]$ et $S_Q = [1200, 2300]$.

Une *vue* est une requête nommée. Plus précisément, il s'agit d'une requête conjonctive définie à l'aide des relations du schéma global $\mathcal{S}$. On suppose que les vues possèdent des contraintes d'intervalle qui sont soit définies par l'administrateur qui a fourni la spécification de la vue, soit attribuées de façon automatique par le médiateur sur la base du domaine de valeur de l'attribut.

L'idée que nous suggérons dans cet article est de calculer les *réponses probables* à une requête donnée Q dans un système suivant une approche de médiation de type LAV. Nous supposerons que les sources de données sont incomplètes et, par conséquent, la définition des vues reposera sur l'hypothèse du monde ouvert.

Définition 1 (Réponse probable) *Soit Q une requête en termes d'un schéma global $\mathcal{S}$. Soit $\mathcal{V}$ un ensemble de définitions de vues en termes de $\mathcal{S}$. Soit V une vue de $\mathcal{V}$ et V^{ext} son extension. Soit $\mathcal{C}$ l'ensemble des contraintes d'intervalle de la requête Q. Soit $\overline{Q}$ la requête Q privée de ses contraintes d'intervalle.*
Un tuple t est une ***réponse probable*** *sous l'hypothèse du monde ouvert si*

i) $t \in \overline{Q}(\mathcal{D})$ *pour toute base de données* $\mathcal{D}$, *telle que* $\forall\, V \in \mathcal{V},\ V^{ext} \subseteq V(\mathcal{D})$

ii) $Proba(t \models \wedge_{i=1}^{n} c_i \in \mathcal{C}) > 0)$

Autrement dit, une réponse probable à Q est une réponse certaine à $\overline{Q}$ telle que la probabilité qu'elle satisfasse toutes les contraintes de Q soit strictement positive. Le problème qui se pose est alors de calculer toutes les réponses probables d'une requête donnée en utilisant

[1] Les requêtes booléennes sont des requêtes conjonctives sans variables distinguées et qui retourne par conséquent comme résultat vrai ou faux

uniquement les vues de $\mathcal{V}$. Plus précisément, le problème est de définir la forme des réécritures de requêtes capables de fournir toutes les réponses probables d'une requête. Nous appelons par la suite ces réécritures des *réécritures tolérantes*. Nous introduisons tout d'abord ce concept de façon informelle.

Considérons t, une réponse probable à une requête donnée Q. D'après la définition 1, t est une réponse certaine à $\overline{Q}$. Il a été démontré que l'évaluation de l'union de toutes les conjonctions de vues *contenues* dans une requête donnée permet d'obtenir toutes ses réponses certaines [15, 2]. L'inclusion entre une conjonction de vues et une requête est définie comme suit.

Définition 2 *Considérons une conjonction de vues Q' et une requête Q. Q' est contenue dans Q, $Q' \sqsubseteq Q$, si les réponses à Q' sont contenues dans celles de Q pour toute instance de base de données $\mathcal{D}$ telle que $\forall\, V \in \mathcal{V}$, $V^{ext} \subseteq\ V(\mathcal{D})$.*

De plus, une réponse probable t doit satisfaire les contraintes de Q avec une probabilité strictement positive. Ainsi, les réécritures tolérantes d'une requête donnée Q sont des conjonctions de vues dont les contraintes d'intervalle satisfont *aussi bien que possible* les contraintes d'intervalle associées à Q. Chacune de ces réécritures tolérantes est associée à un degré $\alpha \in]0, 1]$ qui exprime la *probabilité* pour un tuple qu'elle retourne d'être une réponse certaine à Q.

Définition 3 (Réécriture tolérante) *Soit Q une requête, $\overline{Q}$ la requête Q privée de ses contraintes d'intervalle, et $\mathcal{V}$ un ensemble de définitions de vues. La requête Q' est une réécriture tolérante de Q en termes de $\mathcal{V}$ si:*

- *Q' est une conjonction de vues issues de $\mathcal{V}$, et*
- *$Q' \sqsubseteq \overline{Q}$*
- *$Q' \sqsubseteq_\alpha Q$, avec $\alpha \in]0, 1]$.*

Dans cette définition, l'expression $Q' \sqsubseteq_\alpha Q$ signifie que Q' fournit des tuples dont la probabilité qu'ils satisfassent les contraintes de Q est α.

En plus des réponses certaines, les réécritures tolérantes fournissent des réponses susceptibles de satisfaire les contraintes d'intervalle d'une requête Q. Comme un degré est associé à chacune des réécritures tolérantes de Q, il est possible de les classer. Plutôt que de calculer toutes les réécritures tolérantes de Q et donc toutes les réponses probables à Q — ce qui peut être très coûteux et peu utile pour l'utilisateur —, nous focaliserons par la suite sur le problème de calculer uniquement un sous-ensemble significatif d'entre-elles : soit les k meilleures (où k est un entier), soit celles dont le degré associé atteint un certain seuil $\alpha_{min} \in]0, 1]$. Le calcul du degré α est défini plus formellement dans la sous-section qui suit.

2.2 Calcul du degré d'une réécriture

Concrètement, le calcul du degré d'une réécriture tolérante est fondé sur la *proportion* de réponses qu'elle retourne qui satisfont les contraintes de la requête de l'utilisateur. Le degré est déduit des contraintes d'intervalles définies dans les réécritures. Considérons la requête

Q d'un utilisateur exprimée en termes d'un schéma global $\mathcal{S}$ et une réécriture Q' utilisant uniquement les vues de $\mathcal{V}$, telle que $Q' \sqsubseteq \overline{Q}$. L'objet de cette section est de montrer comment obtenir le degré associé à Q', réécriture tolérante de Q. Dans un premier temps, nous considérons le cas de base suivant: Q ne possède qu'une seule contrainte d'intervalle. Puis nous nous plaçons dans la situation où Q possède plusieurs contraintes d'intervalle.

Nous définissons tout d'abord la notion d'*inclusion tolérante* entre deux intervalles qui est à la base de la définition du degré de toute réécriture tolérante d'une requête donnée. L'inclusion tolérante est associée à un degré compris entre 0 et 1 qui détermine à quel point un intervalle est inclus dans un autre. Soit $I_{Q'}$ et I_Q deux intervalles. Le degré de l'inclusion tolérante de $I_{Q'}$ dans I_Q est définie de la façon suivante:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \alpha = \frac{\mid I_{Q'} \cap I_Q \mid}{\mid I_{Q'} \mid} \tag{1}$$

où $|I|$ dénote la cardinalité de I. Soit $I_Q = [a,\ b]$, $I_{Q'} = [c,\ d]$, et $I_Q \cap I_{Q'} = [e,\ f]$, et supposons que la variable considérée est encodée par des nombres avec n décimales ($n \geq 0$), on obtient alors:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \frac{(f-e) \times 10^n + 1}{(d-c) \times 10^n + 1}.$$

Dans l'hypothèse de la distribution continue et uniforme des valeurs sur le domaine, ce degré correspond évidemment à la *proportion* des éléments de $I_{Q'}$ qui sont dans $I_{Q'} \cap I_Q$. On peut noter que ce degré est équivalent au degré de sélectivité, bien connu dans le contexte de l'optimisation de requête [17, 14], qui est utilisé pour estimer combien de tuples dans une relation sont susceptibles de satisfaire la condition d'une requête.

Dans le cas où on dispose, en plus des contraintes d'intervalle, d'histogrammes ou de fonctions de distribution décrivant les distribution des valeurs sur le domaine [2], comme par exemple une fonction gaussienne, la formule 1 peut être facilement adaptée de la façon suivante. Soit F la fonction de distribution. Supposons que $I_Q = [a,\ b]$, $I_{Q'} = [c,\ d]$, et $I_Q \cap I_{Q'} = [e,\ f]$. Le degré de l'inclusion tolérante de $I_{Q'}$ dans I_Q est défini comme suit:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \frac{\sum_{x \in [e,f]} F(x)}{\sum_{x \in [c,d]} F(x)} \tag{2}$$

Cas d'une seule contrainte d'intervalle dans Q

Dans le cas où Q ne fait intervenir qu'une seule contrainte sur un attribut X, deux sous-cas doivent être considérés: a) la contrainte restreint une variable qui est *distinguée* dans Q', i.e., X apparaît dans l'entête d'au moins une des vues de Q', et b) la contrainte restreint une variable qui est *existentielle* dans Q', i.e., X n'apparaît dans aucune entête des vues de Q'. La proposition suivante établit comment le degré de Q' est déterminé dans chacun des cas.

Proposition 1 *Soit Q une requête et Q' une conjonction de vues telle que $Q' \sqsubseteq \overline{Q}$. Soit X, l'attribut sur lequel Q a une contrainte d'intervalle. Soit I_Q et $I_{Q'}$ les intervalles des contraintes portant sur X dans Q et Q' respectivement. Si $I_{Q'}$ n'est pas explicitement indiqué*

[2] Une telle fonction peut être générée de façon automatique ou fournie par l'administrateur de la source de donnée

dans Q', le domaine actif de l'attribut correspondant X est utilisé. Soit α le degré d'inclusion de Q' dans Q, i.e., le degré de $Q' \sqsubseteq_\alpha Q$.

a) si X est une variable distinguée alors la contrainte d'intervalle I_Q est ajoutée à Q' et

- *si $I_Q \cap I_{Q'} = \emptyset$ alors $\alpha = 0$*
- *sinon $\alpha = 1$.*

b) si X est une variable existentielle alors $\alpha = deg(I_{Q'} \subseteq_{tol} I_Q)$.

Par conséquent, le degré associé à une réécriture tolérante est 0, 1 ou le degré d'inclusion tolérante entre $I_{Q'}$ et I_Q. Dans le cas où α vaut 0, Q' ne fournit aucune réponse probable et n'est plus considérée comme une réécriture tolérante.

Le principe de la preuve de cette proposition est le suivant:

- Cas a): Comme X est une variable distinguée d'une vue de Q', il se peut qu'il existe une jointure sur X entre plusieurs vues de Q'. Ainsi, une ou plusieurs vues de Q' peuvent avoir une contrainte d'intervalle sur X. Soit $\{I_1, \ldots, I_k\}$ l'ensemble de ces intervalles et soit $t \in Q'^{ext}$ une réponse certaine à $\overline{Q}$. Alors $t[X]$, la projection de t sur X appartient à l'intervalle $I_{Q'} = I_1 \cap \ldots \cap I_k$.

  X étant une variable distinguée, on peut appliquer une condition sur X et notamment, la contrainte d'intervalle I_Q sur X en plus de la contrainte d'intervalle $I_{Q'} = I_1 \cap \ldots \cap I_k$ (comme effectué dans [15]). Dès lors, deux cas peuvent se présenter. Soit $I_{Q'} \cap I_Q$ est vide et Q' ne peut fournir aucune réponse probable. Dans ce cas, le degré 0 est associé à Q'. Soit $I_{Q'} \cap I_Q$ n'est pas vide et Q' ne fournit que des réponses certaines à Q. Le degré associé à Q' vaut alors 1.

- Cas b): X est une variable existentielle dans l'expression Q'. Comme ni la sélection ni la jointure n'est possible sur ce type de variable [15], une seule vue de Q' couvre tous les sous-buts de Q contenant X. Ainsi seule cette vue possède une contrainte d'intervalle $I_{Q'}$ sur X.

  Soit $t \in Q'^{ext}$ une réponse certaine à $\overline{Q}$. Alors $t[X]$, la projection de t sur X appartient à l'intervalle $I_{Q'}$. Or $I_{Q'}$ n'est pas forcément inclus dans I_Q. Cependant, on peut calculer la probabilité pour un tuple t issu de Q'^{ext} de valider la contrainte I_Q sur la base des connaissances disponibles concernant la distribution des tuples sur $I_{Q'}$.

  Si aucune information n'est disponible, on suppose une distribution continue et uniforme des tuples sur $I_{Q'}$. Dans ce cas, la probabilité que $t[X]$, pour tout $t \in Q'^{ext}$, appartienne à I_Q est définie par

  $$\frac{\mid I_{Q'} \cap I_Q \mid}{\mid I_{Q'} \mid}.$$

  C'est le degré d'inclusion tolérante défini par l'équation 1 qui utilisé dans la proposition. Si on dispose d'une fonction de distribution F sur le domaine de X, alors la probabilité que $t[X]$, pour tout $t \in Q'^{ext}$, appartienne à I_Q est définie plus finement par :

  $$\frac{\sum_{x \in [e,f]} F(x)}{\sum_{x \in [c,d]} F(x)}$$

où $I_{Q'} = [c, d]$ et $I_{Q'} = [e, f]$. On retrouve la définition de l'équation 2.

L'exemple suivant illustre le cas a) de la proposition 1.

Exemple 1 *Soit la requête suivante:*
$Q(N, A, S) : -Man(N, A), Salary(A, S),\ A \in I_{A_Q} = [28, 38]$
et deux vues V_6 *et* V_7 *ainsi définies:*
$V_6(N_1, A_1) : -Man(N_1, A_1),\ A_1 \in I_{A_1} = [25, 40]$
$V_7(A_2, S_2) : -Salary(A_2, S_2),\ A_2 \in I_{A_2} = [27, 45].$
Une réécriture possible de $Q(N, A, S)$ *est* $Q'(N, A, S) : -V_6(N, A), V_7(A, S)$. *La contrainte finale s'appliquant à* Q' *sur l'attribut* A *porterait alors sur l'intervalle* $I_{Q'} = [25, 40] \cap [27, 45] = [27, 40]$. *Cependant, comme* A *est une variable distinguée dans* V_6 *et* V_7, *il peut être restreint par sélection et la seule réécriture possible est :*
$Q''(N, A, S) : -\ V_6(N, A),\ V_7(A, S),\ A \in [28, 38]$. *La contrainte ajoutée à* Q' *est celle de* Q. *Le degré associé à* $Q''(N, A, S)$ *est alors* 1 *puisque l'intersection entre* $[27, 40]$ *et* $[28, 38]$ *n'est pas vide:* $Q'' \sqsubseteq_1 Q$.$\diamond$

L'exemple suivant illustre le second cas de la proposition 1.

Exemple 2 *Soit la requête suivante:*
$Q(ssn, name) : -\ Person(ssn, name, fname),\ Child(ssn, Cfirstname, Cage),$
$Cage\ \in\ [22, 35]$
et Q' *la réécriture candidate de* Q*:*
$Q'(ssn, name) : -V_8(ssn, name)$ *avec*
$V_8(ssn, name) : -\ Person(ssn, name, fname),\ Child(ssn, Cfirstname, Cage),$
$Emp(ssn, job, sal),\ Cage\ \in\ [20, 40].$
Si on ignore les contraintes d'intervalle portant sur Q, Q' *est contenue dans* Q *(*$Q' \sqsubseteq \overline{Q}$*). D'autre part, la présence des contraintes d'intervalle nous pousserait à écarter* Q' *du processus de réécriture si un appariement booléen était réalisé entre les contraintes d'intervalle. En utilisant une appariement approximatif basé sur les probabilités, on peut définir le "degré d'inclusion" de* Q' *dans* Q *sur la base de l'équation 1. Il donne la probabilité pour* $t[Cage]$ *issu de* Q'^{ext} *d'appartenir à l'intervalle* $[22, 35]$ *sachant qu'il appartient à l'intervalle* $[20, 40]$. *En supposant un encodage de* $Cage$ *par des entiers, le degré vaut* $\alpha = \frac{|[22,35]|}{|[20,40]|} = \frac{(35-22)*10^0+1}{(40-20)*10^0+1} = \frac{14}{21} = 2/3$. *Ainsi nous obtenons:* $Q' \sqsubseteq_{0.67} Q$.$\diamond$

Cas général: Q possède plusieurs contraintes d'intervalle

Considérons maintenant le cas général où Q possède plusieurs contraintes d'intervalle. Dans ce cas, le problème est de calculer la probabilité pour une réponse à Q' d'être une réponse à Q, i.e., de satisfaire chacune de ses contraintes. La proposition suivante définit comment obtenir le degré d'une réécriture tolérante de Q dans ce contexte.

Proposition 2 *Soit* Q *une requête et* Q' *une expression en termes de vues telles que* $Q' \sqsubseteq \overline{Q}$. *Soit* $\mathcal{C}_Q$ *l'ensemble des contraintes d'intervalle — supposées indépendantes — dans* Q *avec* $|\mathcal{C}_Q| = n$. *Soit* α_i *le degré associé à* Q' *pour chacune des contraintes* $c_i \in \mathcal{C}_Q$, *obtenu selon*

les critères de la proposition 1. Le degré α de l'inclusion tolérante de Q' dans Q ($Q' \sqsubseteq_\alpha Q$) est défini par l'expression suivante :

$$\alpha = \prod_{i=1}^{n} \alpha_i$$

Preuve Soit Q' une réécriture de $\overline{Q}$. On sait que pour n'importe quelle contrainte $c_i \in \mathcal{C}_\mathcal{Q}$, une réponse t à Q' a la probabilité α_i de satisfaire c_i. La probabilité que t satisfasse toutes les contraintes d'intervalle de Q est donc le produit des α_i, les contraintes étant supposées indépendantes. •

L'exemple suivant illustre le calcul du degré global d'une réécriture donnée dans le contexte où la requête possède plusieurs contraintes indépendantes.

Exemple 3 *Soit les vues suivantes:*
$V_1(ssn_1, n_1)$: $-$ $person(ssn_1, n_1, f_1)$,
$V_2(ssn_2)$: $-$ $children(ssn_2, cf_2, a_2)$, $a_2 \in [20, 40]$,
$V_3(ssn_3)$: $-$ $employee(ssn_3, j_3, s_3)$, $s_3 \in [2100, 3400]$,
$V_5(ssn_5)$: $-$ $children(ssn_5, cf_5, a_5)$, $employee(ssn_5, j_5, s_5)$, $a_5 \in [18, 38]$, $s_5 \in [1000, 2400]$.
Aucune de ces vues ne possède de contraintes sur des variables distinguées. Considérons maintenant la requête:
$Q(ssn, name)$: $-Person(ssn, name, fname)$, $Child(ssn, Cfirstname, Cage)$,
$Emp(ssn, job, sal)$, $Cage \in [22, 35]$, $sal \in [1200, 2300]$
et les réécritures tolérantes:
$Q_1(nss, nom) : -V_1(nss, nom), V_2(nss), V_3(nss)$ *et*
$Q_2(nss, nom) : -V_1(nss, nom), V_5(nss)$.
En supposant un encodage des variables par des entiers, le degré associé à

- *la réécriture tolérante V_1 du premier sous-but de Q donne le degré $\alpha_1 = 1$ car il n'y a pas de contrainte d'intervalle sur ce sous-but;*

- *la réécriture tolérante V_2 du second sous-but de Q est calculée à partir des contraintes d'intervalle portant sur l'âge. Le degré déduit est $\alpha_2 = \frac{|[22,35]|}{|[20,40]|} = \frac{35-22+1}{40-20+1} = 14/21 = 0.67$;*

- *la réécriture tolérante V_3 du troisième sous-but est calculée à partir des contraintes d'intervalle portant sur le salaire. Le degré déduit est $\alpha_3 = \frac{|[2100,2300]|}{|[2100,3400]|} = \frac{2300-2100+1}{3400-2100+1} = 201/1301 = 0.15$.*

- *la réécriture tolérante V_5 du second et du troisième sous-buts de Q est calculée sur la base des contraintes d'intervalle portant sur l'âge et le salaire. Le degré obtenu est: $\alpha_4 = \frac{|[22,35]|}{|[18,38]|} * \frac{|[1200,2300]|}{|[1000,2400]|} = 14/21 * 1101/1401 = 0.52$.*

Ainsi, $Q_1(nss, nom)$ obtient le degré $1 * 0.67 * 0.15 = 0.1$ alors que $Q_2(nss, nom)$ obtient $1 * 0.52$ ($Q_1 \sqsubseteq_{0.1} Q$ et $Q_2 \sqsubseteq_{0.52} Q$). Les tuples issus de Q_1 risquent fort de ne pas satisfaire la requête alors que ceux issus de Q_2 ont une probabilité de plus de 50% de satisfaire Q.◇

3 Un algorithme pour calculer les k meilleures réécritures d'une requête

Plutôt que de calculer toutes les réécritures tolérantes d'une requête puis de les classer, nous proposons d'adapter un algorithme bien connu de réécriture classique de requête, l'algorithme `Minicon` [15], afin de générer directement les k meilleures réécritures classées dans l'ordre décroissant de leur degré.

`MiniCon` calcule la réécriture maximalement contenue d'une requête donnée Q, i.e., celle qui fournit toutes les réponses certaines de Q, dans le cas où les requêtes et les vues comprennent des contraintes d'intervalle via l'union des requêtes conjonctives contenues dans Q. La première étape de cet algorithme consiste en l'énumération de *toutes* les *descriptions MiniCon*, appelées *MCDs*, de Q. De manière intuitive, l'existence d'un MCD associé à une vue dénote que cette vue couvre un sous-ensemble des sous-buts de la requête. La deuxième étape est dédiée au calcul des combinaisons de MCD qui couvrent tous les sous-buts de Q de telle sorte que les MCD de ces combinaisons forment une partition des sous-buts de Q.

Lorsqu'on souhaite calculer les réécritures tolérantes d'une requête, la première étape de l'algorithme `MiniCon` doit être légèrement modifiée afin d'associer un degré α_M à chaque MCD M d'une vue donnée V. Tout d'abord, la procédure $formMCD(V, M)$ de la version classique du `MiniCon` est utilisée pour vérifier si un MCD peut être créé avec V, i.e., pour vérifier si V peut réécrire une partie des sous-buts de Q. Si c'est le cas, le degré à associer à M est calculé sur la base des propositions 1 ou 2 selon le nombre de variables contraintes par V. Plus précisément, le degré est obtenu à partir des contraintes exprimées dans V sur des variables aussi contraintes par Q. Pour chacune des contraintes de V, le processus suivant est appliqué. Si la contrainte porte sur une variable distinguée (lignes $8-12$ de l'algorithme ci-après), le degré associé au MCD M est 1 ou 0. Si le degré est 0, le MCD M est supprimé. Si la contrainte porte sur une variable existentielle (ligne 14), le degré du MCD est modifié conformément aux degrés d'inclusion calculés à partir de chaque contrainte de Q également spécifiée dans V conformément aux équations 1 et 2.

Tout comme la deuxième étape de l'algorithme du `MiniCon`, la seconde étape de notre algorithme peut être modélisée par un hypergraphe [3] où les sous-buts de Q correspondent aux sommets et chaque ensemble de sous-buts de requête couverts par un MCD correspond à un arc. Toute combinaison satisfaisante de MCDs, i.e., toute partition des sous-buts de Q à l'aide des MCDs, correspond à une *couverture exacte* de cet hypergraphe. Ce type d'objet est fortement connecté à la notion de *transversal minimal* [9, 10] dans la théorie des hypergraphes. Même si déterminer si un hypergraphe donné possède au moins une couverture exacte est un problème NP-Complet [12], des algorithmes qui calculent toutes les couvertures exactes d'un hypergraphe et qui passent à l'échelle existent [13]. De tels algorithmes peuvent être utilisés pour obtenir une implémentation efficace du `MiniCon`. L'algorithme 2 présenté ci-dessous est inspiré de ces algorithmes; il génère soit les k meilleures réécritures d'une requête, soit toutes les réécritures tolérantes dont le degré atteint un certain seuil α_{min}. Les réécritures sont générées dans l'ordre décroissant de leur degré. Ainsi, les k meilleures réécritures de la requête considérée sont les k premières produites.

L'algorithme maintient une liste ordonnée, par rapport à leurs degrés, de *réécritures partielles* de la requête. Ici, une réécriture partielle est formée d'un ensemble de MCDs,

Algorithm 1 ComputeApproximateMCD

Require: $\mathcal{V} = \{V_1, ..., V_m\}$ un ensemble de vues, Q une requête, α_{min} un seuil qui est de 0 quand les k meilleures réécritures sont recherchées.

Ensure: $\mathcal{M}$ l'ensemble des MCD pondérés de Q

```
1: for all V ∈ 𝒱 do
2:   Boolean b:= vérifier formMCD(V, M) pour V
3:   if b then
4:     α_M:=1
5:     𝒞:= Récupération des contraintes d'intervalle associées à V sur des attributs contraints dans Q
6:     for all C_i ∈ 𝒞 do
7:       Soit var la variable de V à laquelle C_i est associée
8:       if var est distinguée then
9:         if C_i est disjointe de la contrainte correspondante dans Q then
10:          α_M := 0
11:          EXIT
12:        end if
13:      else
14:        α_M := α_M * InclusionDegree(var)
15:      end if
16:    end for
17:    if α_M > α_min then
18:      Ajouter M à ℳ
19:    end if
20:  end if
21: end for
```

disjoints deux à deux, qui ne couvrent pas *tous* les sous-buts de la requête. Si on considère une réécriture partielle X et un MCD M couvrant d'autres sous-buts que X, on note par $(X + M)$ la réécriture (partielle) correspondant à la concaténation de X et M, et par $X.\alpha$ le degré d'une réécriture partielle de Q. De plus, on rappelle que d'après les propositions précédentes, le degré d'une concaténation $(X + M).\alpha$ est égal au produit des degrés de X et de M. L'algorithme 2 prend en entrée l'ensemble $\mathcal{M}$ des MCDs pondérés retournés par l'algorithme 1. Cet ensemble de MCDs est stocké dans une structure de données sur laquelle sont définies les opérations suivantes :

- cover(M) supprime tous les MCDs partageant un sous-but avec M,
- uncover(M) ré-insère les MCDs partageant un sous-but avec M, précédemment supprimés par cover(M).

Quand nous utilisons un MCD M dans une réécriture partielle, nous devons interdire l'utilisation de tous les MCD intersectant M. En effet, chaque sous-but de la requête devra être couvert une fois et une seule. L'opération *cover*() est utilisée pour interdire ce type de MCD. D'autre part, l'objectif de la fonction *uncover*() est de ré-autoriser l'utilisation de ces

MCDs dans la suite du processus. On remarque que $\mathcal{M}.cover(X).uncover(X) = \mathcal{M}$, et que l'utilisation d'*uncover*(X) suit toujours celle d'une opération *cover*(). Cette structure de données peut être implémentée très efficacement en utilisant des *dancing links* [13]. En effet, leur utilisation permet d'implémenter les opérations *cover* et *uncover* avec une complexité linéaire par rapport au nombre de MCDs supprimés/insérés.

La génération ordonnée des réécritures est fondée sur la propriété suivante : le degré global d'une réécriture partielle M peut uniquement décroître quand M est combinée à un MCD m. Cette propriété découle du fait que les degrés utilisés sont des probabilités, par conséquent des réels compris entre 0 et 1.

L'algorithme 2 considère les réécritures partielles dans l'ordre décroissant par rapport à leurs degrés et tente de les compléter pour générer l'ensemble $RW(Q, \mathcal{V})$ des réécritures de la requête Q considérée.

Algorithm 2 ComputeRW($\mathcal{M}$)

Require: $\mathcal{M}$ l'ensemble des MCDs pondérés
Ensure: L'ensemble $RW(Q, \mathcal{V})$ des réécritures de Q utilisant $\mathcal{V}$

```
1:  RW(Q,V) = ∅
2:  PartialRW = M
3:  while PartialRW ≠ ∅ do
4:    X = PartialRW.max();
5:    PartialRW = PartialRW \ X;
6:    if X couvre Q then
7:      RW(Q,V) = RW(Q,V) ∪ X;
8:    else
9:      M.cover(X);
10:     for M ∈ M do
11:       if X.α * M.α ≥ α_min then
12:         PartialRW.Add(X + M);
13:       end if
14:     end for
15:     M.uncover(X);
16:   end if
17: end while
```

L'algorithme commence par considérer les réécritures partielles formées d'un seul MCD (ligne 2). Puis, tant qu'il reste des réécritures partielles qui peuvent être complétées, il considère une réécriture partielle X de poids maximal (ligne 4), vérifie si c'est une réécriture de Q (ligne 6), et essaie de la compléter le cas échéant (ligne $9-15$). Pour cela, on commence par retirer de $\mathcal{M}$ tous les MCDs qui partagent un sous-but avec X, puis on essaie de former une réécriture (partielle) avec chacun des MCDs restants. Enfin, on ré-insère les MCDs précédemment supprimés avant de réitérer le processus.

4 Expérimentations

L'approche proposée a été implémentée dans un prototype nommé FlexIS utilisant Java SE 5 et le système de gestion de bases de données PostgreSQL 8.3 [5]. Il se présente sous la forme d'une interface où l'on trouve le schéma global du système d'intégration. Il permet la saisie d'une requête SQL et l'affichage des réponses certaines et probables. Il laisse à l'utilisateur la possibilité de modifier les paramètres k et α_{min}. La figure 1 présente une capture d'écran du prototype.

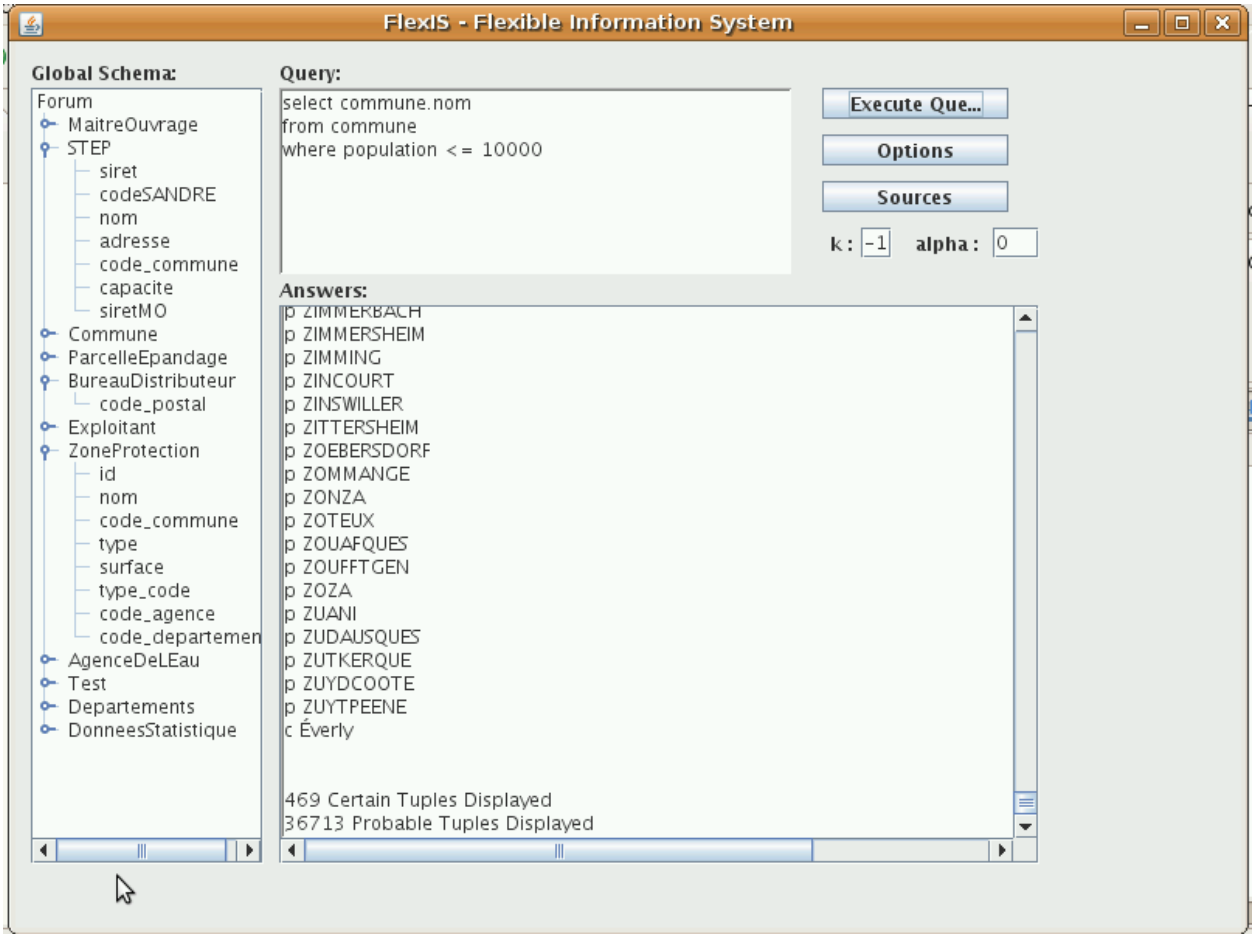

Figure 1: Capture d'écran de FlexIS

Les expérimentations ont été réalisées sur des sources de données agricoles fournies par le Cemagref dans le cadre du projet ANR FORUM (ARA - MDMSA - 2005) [4]. Ces sources se présentent sous la forme de 310 tables, contenant environ 600 Mb de données. Les requêtes sont posées sur un schéma global comprenant 10 tables. Chaque source de données est mappée sur le schéma global par l'intermédiaire d'un mapping LAV. L'exemple suivant montre un extrait du schéma global, quelques sources de données, ainsi que les vues permettant de les mettre en correspondance.

Exemple 4 *Schéma global*

```
step(siret,codeSandre,nom,adresse,codeCommune,capacite,siretMO)
commune(codeCommune,codePostal,departement,nom,population)
maitreOuvrage(siret,codeCommune,nom,adresse,siren)
parcelleEpandage(code,surface,surfaceSIG,codeCommune,typeProduit,departement)
```

Sources de données

```
stepRhin(sandre,lab,mOuvrage,moSiret)
communesPonctuel(nom,code,nbHab)
06stepRmc2(codeSandre,nom,moSiren,moSiret,moNom,moAdr,moCP)
```

Elles sont mises en correspondance avec le schéma global à l'aide des vues suivantes :

```
stepRhin(stepSandre,lbStep,moNom,moSiret) :-
maitreOuvrage(moSiret,_,moNom,_,_),
step(_,stepSandre,lbStep,_,_,capacite,moSiret),
capacite > 1000, capacite < 500000.

communesPonctuel(nom,codeCommune,pop) :-
   commune(codeCommune,_,"89",nom,pop), pop=<38900.

06stepRmc2(stpSandre,stpNom,moSiren,moSiret,moNom,moAdr,moCP) :-
   step(_,stpSandre,stpNom,_,_,capacite,moSiret),
   maitreOuvrage(moSiret,moCo,moNom,moAdr,moSiren),
   commune(moCo,moCP,_,_,_),
   bureauDistributeur(moCP),
   capacite=<1630000, capacite>=25.
```

Les requêtes utilisateur sont formulées dans les termes du schéma global. A titre d'exemple, on peut considérer la requête suivante qui retourne le code SANDRE, le nom, et la commune des stations d'épuration ayant une capacité comprise entre 25000 et 50000 :

```
SELECT step.codeSandre,step.nom,commune.nom
FROM step,commune
WHERE step.codeCommune = commune.codeCommune
      AND capacite > 25000 AND capacite < 50000
```

Avec les paramètres $k = \infty$ et $\alpha_{min} = 0$, cette requête fournit 732 réponses certaines et 4738 réponses probables.

Les expérimentations portent sur une vingtaine de requêtes fournies par des utilisateurs du domaine. Dans ce contexte expérimental, toutes les sources de données portent sur le même domaine d'application. Un tel contexte se prête bien au calcul des réécritures classiques d'une requête, et par conséquent au calcul des réponses certaines (ce qui explique le grand nombre de réponses certaines retrouvées). Les résultats suivants montrent l'évolution du *nombre moyen* de tuples correspondant aux réponses certaines et probables des requêtes.

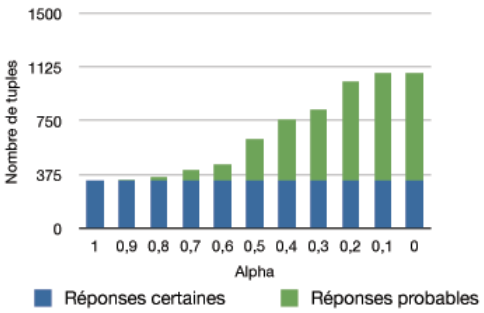

Figure 2: Impact du seuil α_{min} sur le nombre de réponses

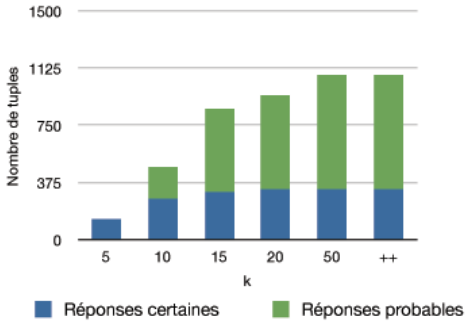

Figure 3: Impact de k sur le nombre de réponses

La première expérimentation montre l'évolution du nombre de réponses quand le seuil α_{min} change. On cherche alors les réécritures tolérantes dont le degré atteint α_{min}. Ainsi, le paramètre k est fixé à l'infini et toutes les réécritures sont générées. On peut noter que le nombre de réponses certaines reste constant. En effet, toutes les réécritures classiques sont calculées pour n'importe quelle valeur de α_{min}. D'autre part, le nombre de réponses probables décroît quand le seuil augmente. Quand α_{min} est égal à 1, seules les réponses certaines sont fournies.

Dans la seconde expérimentation, le seuil est positionné à 0 mais k, le nombre de meilleures réécritures recherchées, doit être spécifié. Lorsque k décroît, le nombre de réponses probables décroît aussi, ce qui montre que seules les meilleures réécritures sont générées. Lorsque k est inférieur à 10, le nombre de réponses certaines décroît aussi. En effet, pour certaines requêtes, un nombre de réécritures inférieur à 10 n'est pas toujours suffisant pour calculer toutes les réponses certaines lorsque le nombre de réponses certaines dépasse le seuil fixé. Un examen de la figure 3 laisse à penser lorsque le seuil passe de 10 à 15 réponses, que le logiciel retourne des réponses probables avant d'avoir donné toutes les réponses certaines. Ce résultat est dû au fait que le graphe présente une moyenne du nombre de réponses certaines et probables fournies par une vingtaine de requêtes. Lorsque le logiciel évalue une requête particulière, il retourne bien l'ensemble de toutes les réponses certaines, si le seuil le permet, avant de retourner des réponses probables.

Le prototype montre de très bonnes performances: le temps d'exécution est d'environ 2 secondes dans le pire des cas, et les performances de l'approche de réécriture tolérante sont comparables à celles de l'algorithme classique `MiniCon` (dans le pire cas, le surcoût est seulement de 10%). Comme mentionné précédemment, l'utilisation de structures de données appropriées permet à notre algorithme de passer à l'échelle: il peut gérer efficacement le nombre de MCDs supplémentaires induits par la nature tolérante du processus de réécriture.

5 Conclusion et discussion

Ce papier traite du problème de calculer et d'ordonner les réécritures tolérantes d'une requête afin de n'exécuter que les meilleures, réduisant ainsi le coût de traitement d'une requête dans un contexte de bases de données décentralisées. Nous avons proposé une approche

pour obtenir de telles réécritures sur la base des contraintes d'intervalles intervenant dans la requête et les vues. Cette approche associe un degré $\in [0, 1]$ à chaque réécriture Q' d'une requête Q, qui correspond à la probabilité pour un tuple retourné par Q' d'être une réponse certaine à Q. Nous avons adapté l'algorithme `MiniCon` pour calculer les réécritures tolérantes d'une requête. De plus, nous avons proposé un algorithme qui permet de n'obtenir que les meilleures réécritures, réduisant ainsi la combinatoire relative à un nombre potentiellement élevé de MCDs. Les expérimentations montrent que le coût induit par la nature tolérante de l'appariement entre les contraintes des vues et celles de la requête est parfaitement acceptable, alors qu'un nombre significatif de réponses supplémentaires est obtenu.

Les expérimentations effectuées ont également permis d'estimer — dans un contexte spécifique — le nombre de réponses supplémentaires fournies avec une telle approche par rapport au nombre de réponses certaines. Cependant, il convient de noter que les réponses retournées sont susceptibles de ne pas satisfaire tous les critères de la requête utilisateur initiale. Pour faire prendre conscience de cet état de fait à l'utilisateur, une solution simple consiste à associer un degré à chaque réponse fournie, celui de la réécriture tolérante dont elle est issue. A ce propos, il serait intéressant de compléter les expérimentations par des tests réalisés avec des utilisateurs afin d'évaluer leur satisfaction globale quant aux réponses non-certaines qui leur sont retournées.

De manière générale, les travaux proposés ici ont vocation à s'appliquer dans un cadre d'intégration particulier, i.e., celui du Web, dans lequel les appariements exacts entre la description des sources de données et la requête ne sont pas toujours possibles. Dans un tel contexte, les réponses à une requête ne peuvent être calculées que sur la base des informations dont on dispose même si celles-ci sont teintées d'incertitude. Des approches de réécriture flexibles permettent alors de fournir aux utilisateurs des réponses là où les algorithmes de réécriture classiques basés sur la sémantique des réponses certaines échouent. L'idée de fournir aux utilisateurs des réponses *probables* n'est pas nouvelle puisqu'elle a été utilisée pour des contextes d'intégration ouverts, dans [6, 16]. La nature de la probabilité associée aux réponses fournies par les algorithmes de ces articles diffèrent cependant de celle utilisée dans nos travaux. En effet, dans [6], elle indique la probabilité d'*existence* des réponses retournées tandis que dans [16], elle découle de l'incertitude portant sur les appariements entre descriptions de sources de données.

En termes de perspectives, nous projetons maintenant d'étudier d'autres sémantiques, non-probabilistes, de réécriture tolérante de requêtes en utilisant les vues, sur la base de visions alternatives de la relaxation de requête.

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] C. Berge. *Hypergraphs*. North-Holland, 1989.

[4] Stephan Bernard, François Barnabé, and François. Pinet. Cas d'utilisation - projet forum, 2007.

[5] Pierre Colomb and Hélène Jaudoin. Flexis, vers un système d'intégration flexible. In *Demonstration Session Bases de Données Avancées*, 2008.

[6] N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *VLDB*, pages 805–816. VLDB Endowment, 2005.

[7] G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis. Answering top-k queries using views. In *VLDB*, pages 451–462. VLDB Endowment, 2006.

[8] Xin Dong, Alon Y. Halevy, and Cong Yu. Data integration with uncertainty. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 687–698. VLDB Endowment, 2007.

[9] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J.*, 24(6), 1995.

[10] T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Appl. Math. DOI*, 10, 2007.

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

[12] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 43:85–103, 1972.

[13] D.E. Knuth. Dancing links. *Arxiv preprint cs.DS/0011047*, 2000.

[14] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *SIGMOD Conference*, pages 256–276, 1984.

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

[16] Anish Das Sarma, Xin Dong, and Alon Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 861–874. ACM, 2008.

[17] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *SIGMOD*, pages 23–34, New York, NY, USA, 1979. ACM.