Techniques de fouille de données pour la réécriture de requêtes en présence de contraintes de valeurs

Hélène Jaudoin*, Frédéric Flouvat*

*Laboratoire LIMOS, UMR CNRS 6158
Université Blaise Pascal - Clermont-Ferrand II,
63 177 Aubière cedex, France
{hjaudoin,flouvat}@isima.fr

Résumé. Dans cet article, nous montrons comment les techniques de fouilles de données peuvent résoudre efficacement le problème de la réécriture de requêtes en termes de vues en présence de contraintes de valeurs. A partir d'une formalisation du problème de la réécriture dans le cadre de la logique de description $\mathcal{ALN}(\mathcal{O}_v)$, nous montrons comment ce problème se rattache à un cadre de découverte de connaissances dans les bases de données. L'exploitation de ce cadre nous permet de bénéficier de solutions algorithmiques existantes pour la résolution du problème de réécriture. Nous proposons une implémentation de cette approche, puis nous l'expérimentons. Les premiers résultats démontrent l'intérêt d'une telle approche en termes de capacité à traiter un grand nombre de sources de données.

1 Introduction

Aujourd'hui, les techniques d'analyse et d'intégration de données sont devenues des atouts majeurs pour les entreprises et les services gouvernementaux. En effet, ces techniques permettent un gain de temps pour regrouper et croiser l'information distribuée. Dans le domaine du développement durable, ces techniques sont notamment indispensables afin de rassembler et d'analyser les pratiques agricoles et ainsi garantir la traçabilité des pratiques. Plus précisément nos travaux se situent dans le cadre d'un projet [1] visant à mettre en place un système d'intégration pour interroger les sources de données agricoles distribuées. Le système doit être flexible pour permettre l'arrivée de nouvelles sources de données afin de suivre le processus d'informatisation du domaine agricole. En effet entre 2000 et 2003, le nombre d'exploitations ayant un accès à Internet a triplé [2]. Il doit de plus permettre de traiter un grand nombre de sources de données car le domaine est susceptible d'accueillir, en plus des services déconcentrés des ministères, un grand nombre d'exploitations.

Dans cet article, nous nous plaçons dans le cadre d'un système de médiation suivant une approche Local As View (LAV), où les vues sont décrites via des requêtes sur le schéma global. Cette approche est connue pour être flexible car l'ajout et la suppression de sources de

[1] Ce projet est réalisé en collaboration avec le Cemagref, http ://www.cemagref.fr/
[2] http ://www.acta-informatique.fr/

données n'affectent pas le schéma global. Nous nous intéressons plus particulièrement au problème de la réécriture de requêtes en termes de vues en présence de *contraintes de valeurs*. Les contraintes de valeurs correspondent à la notion de type énuméré en base de données. Elles permettent de spécifier les valeurs autorisées pour un attribut donné. Les contraintes de valeurs sont utiles dans beaucoup d'applications, comme par exemple pour la vérification des contraintes d'intégrité ou pour exprimer une forme d'information incomplète (Borgida et Patel-Schneider, 1994). De plus, ce type de contraintes connaît actuellement un regain d'intérêt, impulsé notamment par les travaux liés au web sémantique. En effet, les types énumérés font partie intégrante du langage d'ontologie OWL [3], un standard émergeant du web sémantique [4]. L'utilisation des contraintes de valeurs est motivée par notre cadre applicatif. En effet, de nombreuses vues ont souvent la même description et se distinguent uniquement par les valeurs autorisées pour certains de leurs attributs. De plus, les contraintes de valeurs interviennent naturellement dans l'expression de requêtes typiques du domaine d'application.

Une version préliminaire de notre travail concernant le problème de réécriture en termes de vues dans le cadre du langage $\mathcal{FL}_0(\mathcal{O}_v)$ a été publié à BDA'2004 (Jaudoin et al., 2004). L'article présenté ici étend ces résultats dans les directions suivantes :

- Nous considérons ici le problème de la réécriture en termes de vues dans le cadre du langage $\mathcal{ALN}(\mathcal{O}_v)$. En plus des constructeurs de quantification universelle et de conjonction du langage $\mathcal{FL}_0$, le langage $\mathcal{ALN}$ permet une forme de négation et des contraintes de cardinalité. Le langage $\mathcal{ALN}(\mathcal{O}_v)$ augmente le langage $\mathcal{ALN}$ avec le constructeur $(\mathcal{O}_v)$ qui permet d'exprimer les contraintes de valeurs. L'investigation théorique du problème de réécriture est plus complexe dans le cadre du langage $\mathcal{ALN}(\mathcal{O}_v)$ du fait notamment des intéractions des contraintes de valeurs avec les contraintes de cardinalité, ainsi que de la nécessité de tenir compte de l'inconsistance. La résolution de ce problème nécessite une nouvelle caractérisation de la subsomption dans $\mathcal{ALN}(\mathcal{O}_v)$ adaptée au problème de réécriture ainsi qu'une caractérisation des nouvelles formes de réécritures qui apparaissent dans $\mathcal{ALN}(\mathcal{O}_v)$ (c.f. lemme 1).
- Nous étudions ensuite les problèmes algorithmiques engendrés par les nouveaux cas de réécritures dans $\mathcal{ALN}(\mathcal{O}_v)$. Nous présentons un nouveau prédicat, nommé P_2, issu de la formalisation du problème du calcul de ces nouveaux cas de réécriture dans le cadre de Mannila et Toivonen (1997). Ce prédicat P_2, en conjonction avec le prédicat P_1 présenté dans (Jaudoin et al., 2004) et rappelé ici, permet de résoudre le problème de réécriture.
- Enfin, nous décrivons succintement une implémentation de notre approche de réécriture qui exploite et adapte un algorithme de fouille de données existant, $Apriori$. Nous présentons ensuite les résultats de nos expérimentations qui viennent conforter l'intérêt des techniques de fouilles de données dans notre cadre. Les premiers résultats montrent la capacité de notre prototype à passer à l'échelle en supportant le traitement d'un grand nombre de vues (jusqu'à 15000). Notons que dans la littérature, peu d'articles présentent les résultats expérimentaux de leurs approches de réécriture. Ils se concentrent généralement sur les résultats théoriques. A notre connaissance, (Pottinger et Halevy, 2001) est une des rares références qui décrit l'évaluation des performances d'un algorithme de réécriture. Aussi la réalisation d'un prototype et son expérimentation constituent à notre avis une contribution dans le domaine de la réécriture.

[3] http ://www.w3.org/2004/OWL/

[4] OWL est actuellement une recommandation du W3C (http ://www.w3.org/).

La suite de l'article est organisée comme suit. Dans la section 2, nous donnons les prérequis nécessaires à la formalisation du problème de la réécriture de requêtes en présence de contraintes de valeur dans le cadre des logiques de description. La section 3 reformule ce problème dans un cadre de découverte de connaissances dans les bases de données. La section 4 présente l'implémentation et les expérimentations réalisées. En section 5, une conclusion et des perspectives sont données. Les démonstrations des lemmes et théorèmes de cet article sont donnés dans Jaudoin et al. (2005).

2 Réécriture de requêtes en présence de contraintes

2.1 Prérequis sur les logiques de description

Les logiques de description sont un formalisme de représentation des connaissances qui permet de représenter des structures complexes et de raisonner avec elles (Baader et al., 2003). Elles permettent de décrire un domaine d'application à l'aide de concepts (prédicats unaires) et de rôles (prédicats binaires). Une logique de description est définie par un ensemble de constructeurs. Dans cet article, nous nous intéressons à la logique $\mathcal{ALN}(\mathcal{O}_v)$, dont les constructeurs sont listés dans la table 1, colonne 1, où $\mathcal{C}$ est un ensemble de noms de concepts, $\mathcal{N}$ un ensemble de noms de valeurs, $\mathcal{R}_C$ un ensemble de noms de rôles dont l'image est un concept de $\mathcal{C}$ et $\mathcal{R}_V$ un ensemble de noms de rôles qui prennent leurs valeurs dans $\mathcal{N}$.

Constructeurs	*Sémantique*
$\top_C$	δ_C
$\top_V$	δ_V
$\top$	$\Delta^{\mathcal{I}} = \delta_C \cup \delta_V$
$\bot$	$\emptyset$
P	$P^{\mathcal{I}}$
$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\forall R_c.C$	$\{x \in \delta_C \mid \forall y : (x,y) \in {R_c}^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
$\forall R_v.\{o_1 \ldots, o_n\}$	$\{x \in \delta_C \mid \forall y : (x,y) \in {R_v}^{\mathcal{I}} \rightarrow y \in \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\} \subseteq \delta_V\}$
$\leq nR$	$\{x \in \delta_C \mid \lvert\{y \mid (x,y) \in R^{\mathcal{I}}\}\rvert \leq n\}$
$\geq nR$	$\{x \in \delta_C \mid \lvert\{y \mid (x,y) \in R^{\mathcal{I}}\}\rvert \geq n\}$

avec $A \in \mathcal{C}$, n un entier positif, $o_i \in \mathcal{N}$, $R \in \mathcal{R}_C \cup \mathcal{R}_V$, $R_c \in \mathcal{R}_C$ et $R_v \in \mathcal{R}_V$.

TAB. 1 – *Syntaxe et sémantique de la logique* $\mathcal{ALN}(\mathcal{O}_v)$

Exemple 1 *Exemples de concepts pouvant être formés à l'aide des constructeurs de* $\mathcal{ALN}(\mathcal{O}_v)$.

- $ParcelleCulturale \sqcap \forall aRecuE.\neg EffluentElevage \sqcap \geq 1\ aRecuE$ *spécifie les parcelles culturales qui ont reçu au moins un épandage qui n'était pas un effluent d'élevage.*
- $\forall numDepartement.\{63, 43, 03\} \sqcap \forall aRecu. \leq 3\ typeProduit$ *spécifie les individus qui sont dans un des départements* 63 *ou* 43 *ou* 03 *et qui ont reçu moins de* 3 *types de produit. Les contraintes de valeurs ont permis ici de restreindre l'image du rôle* $numDepartement$.

La sémantique des concepts est donnée par une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ où $\Delta^{\mathcal{I}}$ est un ensemble non vide, appelé *domaine d'interprétation* et $.^{\mathcal{I}}$ est une *fonction d'interprétation*. Nous supposons que le domaine d'interprétation $\Delta^{\mathcal{I}}$ est partionné en deux ensembles disjoints : δ_C décrivant l'ensemble des individus du domaine et δ_V décrivant l'ensemble des valeurs. Un concept est interprété comme étant un sous-ensemble de $\Delta^{\mathcal{I}}$. Un rôle est interprété comme un sous-ensemble de $\delta_C \times \Delta^{\mathcal{I}}$. L'interprétation $\mathcal{I}$ associe chaque valeur $o_i \in \mathcal{N}$ à un élément $o_i^{\mathcal{I}}$ de δ_V tel que $o_i \neq o_j$ implique que $o_i^{\mathcal{I}} \neq o_j^{\mathcal{I}}$. Plus précisément, la sémantique des constructeurs de $\mathcal{ALN}(\mathcal{O}_v)$ est donnée dans la colonne 2 de la table 1. Une interprétation est un *modèle* pour un concept C ssi $C^{\mathcal{I}} \neq \emptyset$. Un concept est inconsistant ssi $C^{\mathcal{I}} \neq \emptyset$ pour toute interprétation $\mathcal{I}$.

Etant donnée cette sémantique, il est possible de définir la notion de subsomption et d'équivalence comme suit. Un concept C est *subsumé* par un concept D, noté $C \sqsubseteq D$, ssi $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ $\forall \mathcal{I}$. Un concept C est *équivalent* à un concept D, noté $C \equiv D$, ssi $C^{\mathcal{I}} = D^{\mathcal{I}}$ $\forall \mathcal{I}$.

Les descriptions intentionnelles d'un domaine d'application sont définies à l'aide d'une *terminologie*. Une terminologie est un ensemble fini d'axiomes terminologiques de la forme : *(i)* $A \sqsubseteq D$ (Spécification primitive de concept) ou *(ii)* $A \equiv D$ (Définition de concept), où A est un nom de concept et D un concept dans $\mathcal{ALN}(\mathcal{O}_v)$. Dans cet article, nous supposons que les terminologies sont acycliques, i.e. aucun concept ne fait référence à lui-même directement ou indirectement dans sa définition ou dans sa spécification. La sémantique des terminologies est obtenue en étendant la notion d'interprétation aux axiomes terminologiques comme suit. Une interprétation $\mathcal{I}$ est un modèle pour une terminologie $\mathcal{T}$ ssi $\mathcal{I}$ est un modèle pour chaque axiome de $\mathcal{T}$.

Pour traiter le problème de la réécriture dans la logique $\mathcal{ALN}(\mathcal{O}_v)$, nous nous appuyons sur la forme normale donnée dans Jaudoin et al. (2005). La forme normale d'un concept permet d'exprimer un concept sous une forme canonique. Cette forme normale transforme tout concept C en concept $\top$ ou en une conjonction d'*atomes* de la forme $\forall w.P$ avec w un mot défini sur l'ensemble des rôles de $\mathcal{R}_C \cup \mathcal{R}_V$ [5] et P est soit un concept atomique A ou une restriction de cardinalité ($\leq nR$ ou $\geq nR$) ou un ensemble de valeurs E. Par la suite, on note $\forall w.P \in C$ si $\forall w.P$ apparaît dans la description du concept C.

2.2 Réécriture dans $\mathcal{ALN}(\mathcal{O}_v)$

Dans cette section, nous définissons le cadre de médiation et le problème de la réécriture dans le cadre des logiques de description. Puis nous donnons les caractéristiques des réécritures dans ce contexte. Le schéma global $\mathcal{S}$ est une terminologie formée de définitions de concepts dans $\mathcal{ALN}(\mathcal{O}_v)$. Une requête Q est un concept dans $\mathcal{ALN}(\mathcal{O}_v)$. Q est décrite en termes des concepts de $\mathcal{S}$. De plus Q est supposée être dans sa forme normale. $\mathcal{V}$ est une terminologie formée de spécifications primitives dans $\mathcal{ALN}(\mathcal{O}_v)$. Les spécifications primitives de $\mathcal{V}$ permettent de décrire les *vues*. Les vues sont décrites en termes de $\mathcal{S}$ et sont supposées être données dans leur forme normale.

Dans ce contexte, on cherche à répondre à une requête Q en ayant uniquement connaissance des vues de $\mathcal{V}$. Une technique pour répondre à Q est de reformuler Q en une expression qui utilise uniquement les vues de $\mathcal{V}$. L'expression obtenue est appelée *réécriture*. On s'in-

[5] Soit r_1 et r_2 deux rôles. $\forall r_1.r_2.A$ dénote le concept $\forall r_1.\forall r_2.A$. Dans ce cas, $r_1.r_2$ est un mot où r_1 appartient à $\mathcal{R}_C$ uniquement et r_2 appartient à $\mathcal{R}_C \cup \mathcal{R}_V$.

téresse ici aux réécritures maximales de Q, i.e. les réécritures qui permettent de calculer le maximum de réponses à une requête donnée. La définition de réécriture maximale est donnée ci-dessous :

Définition 2 (Réécriture maximale) *Soient $\mathcal{V}$ une terminologie de vues et Q un concept dans $\mathcal{ALN}(\mathcal{O}_v)$. Q' est une réécriture maximale de Q en termes de $\mathcal{V}$ ssi :*

1. *Q' est un concept dans le langage $\{\sqcap, \sqcup\}$ qui réfère uniquement les noms des vues de $\mathcal{V}$,*
2. *$Q' \sqsubseteq Q$.*
3. *il n'existe pas de concept Q_1 dans $\{\sqcap, \sqcup\}$ qui réfère uniquement les noms des vues de $\mathcal{V}$, tel que $Q' \sqsubseteq Q_1 \sqsubseteq Q$ et $Q' \not\equiv Q_1$.*

Par la suite, on appelle *réécriture conjonctive maximale* de Q, toute réécriture de Q dans le langage $\{\sqcap\}$. Comme montré dans Jaudoin et al. (2005), une requête Q donnée a une unique réécriture maximale formée de l'union de ses réécritures conjonctives maximales. L'ensemble des réécritures conjonctives maximales de Q dans le contexte de la logique $\mathcal{ALN}(\mathcal{O}_v)$ est fini. Aussi pour calculer les réponses d'une requête Q, on cherche à calculer les réécritures conjonctives maximales de Q.

Pour calculer les réécritures conjonctives maximales d'une requête donnée Q, on suit une approche classique de réécriture basée sur l'algorithme des paniers (Levy et al., 1996). Informellement, cette approche fonctionne de la manière suivante. Etant donné une requête $Q \equiv \forall w_1.P_1 \sqcap \ldots \sqcap \forall w_n.P_n$, la principale idée est de considérer chaque atome $\forall w_i.P_i$ de Q isolément. A chaque atome $\forall w_i.P_i$ de la requête est associé un panier/ensemble qui contient toutes les réécritures conjonctives maximales de cet atome. Ensuite dans une deuxième étape, les réécritures candidates de Q sont calculées en effectuant le produit cartésien entre les paniers. Ceci permet d'obtenir un sur-ensemble de toutes les réécritures conjonctives maximales de Q. Pour obtenir effectivement les réécritures conjonctives maximales, les réécritures inconsistantes et non maximales doivent ensuite être supprimées de ce sur-ensemble.

Les travaux présentés dans Jaudoin et al. (2005) montrent en s'appuyant sur une caractérisation de la subsomption dans $\mathcal{ALN}(\mathcal{O}_v)$, que dans le cadre de médiation défini précédemment, les réécritures conjonctives maximales d'un atome donné sont formées de sous-ensembles minimaux [6] de $\mathcal{V}$ qui vérifient les conditions énoncées dans le lemme qui suit.

Lemme 1 *Soit $Q \equiv \forall w.P$. Soient l la cardinalité du plus grand des ensembles de valeurs qui apparaît dans les vues ou la requête, et p la profondeur maximale[7] des atomes apparaissant dans les requêtes et les vues. Si $Q' \equiv V_{i_1} \sqcap \ldots \sqcap V_{i_k}$ où $\{V_{i_1}, \ldots V_{i_k}\}$ est un sous-ensemble **minimal** de $\mathcal{V}$ tel que Q' est subsumée par $\forall w.P$ alors une des conditions suivantes est vérifiée :*

a) $k = 1$,
a.1) $P \in \{A, \neg A\}$ et $\forall w.P \in V_{i_1}$ ou,
a.2) $P = (\geq nR)$ et $\forall w.(\geq pR) \in V_{i_1}$ avec $p \geq n$ ou,
a.3) $P = (\leq nR)$ et $\forall w.(\leq pR) \in V_{i_1}$ avec $p \leq n$
b) $1 \leq k \leq l + 1$,

[6] au sens de l'inclusion ensembliste
[7] la profondeur d'un atome $\forall w.P$ est égal à la longueur du mot w, i.e. le nombre de rôles dans w.

b.1) $P = E$ *et* $\{V_{i_1}...V_{i_k}\}$ *est tel que : pour* $j \in \{1,\ldots,k\}$, $\forall w.E_{i_j} \in V_{i_j}$, *et* $\cap_{j=i_1}^{i_k} E_j \subseteq E$.

b.2) $P = (\leq\ n\ R_v)$, *avec* $R_v \in \mathcal{R}_v$, $\{V_{i_1}...V_{i_k}\}$ *est tel que : pour* $j \in \{1,\ldots,k\}$, $\forall w.E_{i_j} \in V_{i_j}$, *et* $|\cap_{j=i_1}^{i_k} E_j| \leq n$.

c) $1 \leq k \leq l+p$ *et il existe un préfixe* $w'v$ *de* w *tel que* $\forall w'.(\leq 0v) \in \sqcap_{j=i_1}^{i_k} V_j$.

Ce lemme donne les conditions nécessaires pour qu'un sous-ensemble de vues de $\mathcal{V}$ forme une réécriture conjonctive maximale de $Q \equiv \forall w.P$, i.e. pour qu'une conjonction de vues appartiennent au panier $B(w,P)$ de l'atome $\forall w.P$. Ainsi une réécriture d'un panier peut être formée d'une seule vue (cas (a) du lemme) ou d'au plus $l+1$ (cas (b) du lemme) ou d'au plus $l+p$ (cas (c) du lemme). Un examen plus attentif du lemme montre que lors de la construction des paniers, on calcule des cas classiques de réécriture dans le langage $\mathcal{ALN}$ (cas (a) et (c) du lemme) et des cas de réécritures spécifiques à la présence des contraintes de valeurs (cas (b.1) et (b.2)) [8]. Aussi, dans la suite de l'article, nous nous concentrons sur les problèmes du calcul des réécritures engendrées par les contraintes de valeurs car ces problèmes posent de nouvelles difficultés en termes de réécriture de requêtes.

Soit une requête $Q \equiv \forall w.P$ où P est un ensemble de valeurs E ou une restriction de cardinalité $\leq\ n\ R_v$. Considérons maintenant le problème de la création du panier $B(w,P)$. Nous nous intéressons aux problèmes du calcul des réécritures des cas (b.1) et (b.2) du lemme 1. Pour définir plus précisément ces problèmes, nous introduisons l'ensemble $\mathcal{V}_w = \{V_1,...,V_p\}$ qui désigne le sous-ensemble de vues de $\mathcal{V}$ telles que $\forall i \in \{1,...,p\}, \exists E_i \mid (\forall w.E_i) \in V_i$.

Le problème *E_conj_rewrite(E,w)* correspond au calcul des réécritures de type (b.1). Il est défini comme suit :

Problème 1 *Soit* $Q \equiv \forall w.E$ *et une terminologie* $\mathcal{V}$ *dans*$\mathcal{ALN}(\mathcal{O}_v)$. *Soit* $\mathcal{V}_w \subseteq \mathcal{V}$.

Le problème E_conj_rewrite(E,w) consiste à calculer les plus petites conjonctions de vues de $\mathcal{V}_w$ *subsumées par* $\forall w.E$.

Le problème *N_conj_rewrite(n,wR)* correspond au calcul des réécritures de type (b.2). Il est defini comme suit :

Problème 2 *Soit* $Q \equiv \forall w.\leq\ n\ R$ *et une terminologie* $\mathcal{V}$ *dans*$\mathcal{ALN}(\mathcal{O}_v)$. *Soit* $\mathcal{V}_{wR} \subseteq \mathcal{V}$.

Le problème N_conj_rewrite(n,wR) consiste à calculer les plus petites conjonctions de vues de $\mathcal{V}_{wR}$ *subsumées par* $\forall w.\leq\ n\ R$.

L'exemple suivant illustre les solutions des problèmes présentés ci-dessus.

Exemple 3 *Soit une requête* Q *telle que* $Q \equiv \forall numDepartement.E \sqcap \forall aRecu.\ \leq 3\ typeProduit$, *avec* $E = \{63,43,03\}$. *Supposons qu'il existe* 4 *vues* $V_i, i \in \{1,...,4\}$ *telles que* $V_i \sqsubseteq \forall numDepartement.E_i$ *où* $E_1 = \{23,15,18,80,43,03\}$, $E_2 = \{03,63\}$, $E_3 = \{01,07,11,\ 43,63\}$, $E_4 = \{26,63\}$, *et* 3 *vues* $V_i, i \in \{5,6,7\}$ *telles que* $V_i \sqsubseteq \forall aRecu.typeProduit.E_i$ *pour* $i \in \{5,6,7\}$ *où* $E_5 = \{P_1,P_{10},P_{15},P_{20},P_{27}\}$, $E_6 = \{P_1,P_{10},P_{15},P_{20},P_{26}\}$, $E_7 = \{P_1,P_{10},P_{15},P_{26},P_{27}\}$.

[8] Notons cependant que les réécritures du cas (b.2) du lemme intéragissent avec les cas classiques de réécritures dans $\mathcal{ALN}$. Ces réécritures sont nécessaires pour obtenir effectivement les réécritures du cas (c) du lemme

Ici on a $\mathcal{V}_{numDepartement} = \{V_1, V_2, V_3, V_4\}$. *On calcule les solutions de E_conj_rewrite(E,numDepartement) à partir de* $\mathcal{V}_{numDepartement}$. *On obtient les conjonctions suivantes* V_2, $V_1 \sqcap V_3$, $V_3 \sqcap V_4$ *et* $V_1 \sqcap V_4$. *On peut par exemple vérifier que* $E_2 \subseteq E$. *Le panier* $B(numDepartement, E)$ *contient alors les conjonctions* V_2, $V_1 \sqcap V_3$, $V_3 \sqcap V_4$ *et* $V_1 \sqcap V_4$.

De la même manière à partir de $\mathcal{V}_{aRecu.typeProduit} = \{V_5, V_6, V_7\}$, *on obtient la solution de N_conj_rewrite(3,aRecu.typeProduit) :* $V_5 \sqcap V_6 \sqcap V_7$. *En effet, l'intersection des ensembles* E_5, E_6, E_7 *donne un ensemble dont la cardinalité est inférieure à* 3. *Ainsi* $V_5 \sqcap V_6 \sqcap V_7$ *appartient au panier* $B(aRecu, (\leq 3typeProduit))$.

Dans la section qui suit, nous montrons comment les deux problèmes *E_conj_rewrite(E,w)* et *N_conj_rewrite(n,wR)* se rattachent à un cadre de découverte des connaissances dans les bases de données.

3 Vers la mise en place de techniques de fouille de données

3.1 Le cadre de Mannila et Toivonen (1997)

Pour rattacher les problèmes énoncés précédemment à un cadre de découverte de connaissances, nous nous appuyons sur le cadre théorique introduit dans Mannila et Toivonen (1997). Il formalise un problème basique de découverte de connaissances dans des bases de données, qui peut être énoncé de la manière suivante : Soit r une base de données, $\mathcal{L}$ un langage pour exprimer des propriétés ou définir des sous-groupes des données, et P un prédicat de sélection. Le prédicat P permet d'évaluer si une phrase $X \in \mathcal{L}$ est "intéressante" dans r. L'objectif est de trouver la théorie de r selon $\mathcal{L}$ et P, i.e. l'ensemble $\mathcal{T}h(r, \mathcal{L}, P)$=$\{ \varphi \in \mathcal{L} \mid P(r, \varphi)$ est vrai$\}$, qui correspond à l'ensemble des phrases intéressantes de r.

Soit une relation de spécialisation/généralisation, i.e. un ordre partiel $\preceq$, sur les motifs de $\mathcal{L}$. On dit que X généralise Y et que Y spécialise X quand $X \preceq Y$. Soit $\mathcal{S}$ un ensemble de phrases de $\mathcal{L}$ tel que si $\varphi \in \mathcal{S}$ et $\gamma \preceq \varphi$ alors $\gamma \in \mathcal{S}$. Alors $\mathcal{S}$ peut être représenté par sa bordure positive $\mathcal{B}d^+(\mathcal{S})$ ou sa bordure négative $\mathcal{B}d^-(\mathcal{S})$.

$$\mathcal{B}d^+(\mathcal{S}) = \{\varphi \in \mathcal{S} \mid \text{ pour tout } \theta \in \mathcal{L} \text{ avec } \varphi \preceq \theta, \theta \notin \mathcal{S}\}$$

$$\mathcal{B}d^-(\mathcal{S}) = \{\varphi \in \mathcal{L} \setminus \mathcal{S} \mid \text{ pour tout } \gamma \in \mathcal{L} \text{ avec } \gamma \preceq \varphi, \gamma \in \mathcal{S}\}$$

La bordure positive correpond aux éléments les plus spécifiques de la théorie, tandis que la bordure négative correspond aux éléments les plus généraux de la théorie. Elles permettent chancune de retrouver toutes les phrases $X \in \mathcal{L}$ "intéressantes" dans r, i.e. celles pour lesquelles $P(r, X)$ est vrai. Notons que si le prédicat P de $\mathcal{T}h(r, \mathcal{L}, P)$ est anti-monotone par rapport à $\preceq$ (i.e. si $\forall X, Y \in \mathcal{L}$ tels que $X \preceq Y$ et $Pred(r, Y)$ est vrai alors $Pred(X, r)$ est vrai), alors la théorie peut être représentée par ces bordures.

Ce cadre peut être appliqué à de multiples problèmes (Mannila et Toivonen, 1997), comme par exemple le problème de la découverte des motifs fréquents (Agrawal et al., 1993). La section qui suit s'attache à montrer comment le problème de la réécriture peut se ramener à une formulation ensembliste, puis être transposé dans le cadre précédemment introduit.

3.2 Formulation des problèmes de réécriture dans un cadre de découverte de connaissances

3.2.1 Formulation ensembliste de la réécriture

Pour un mot w, un ensemble E, un entier n donnés, nous cherchons maintenant à donner une formulation ensembliste des problèmes *E_conj_rewrite(E,w)* et *N_conj_rewrite(n,w)*. Pour reformuler plus précisément ces problèmes, nous introduisons les définitions suivantes :

Définition 4 *Soient w un mot, E un ensemble et n un entier. Soit $\mathtt{F}_w = \{E_1, ..., E_p\}$ l'ensemble des E_i associés aux vues V_i de $\mathcal{V}_w$, et $\mathcal{P}(\mathtt{F}_w)$ l'ensemble des parties de $\mathtt{F}_w$.*

$S_1(w, E) = \{X \in \mathcal{P}(\mathtt{F}_w) | \ \cap\{x \in X\} \subseteq E$ *et* $\forall Y \subset X, \cap\{y \in Y\} \not\subseteq E\}$

$S_2(w, n) = \{X \in \mathcal{P}(\mathtt{F}_w) | \ |\cap\{x \in X\}| \leq n$ *et* $\forall Y \subset X, \ |\cap\{x \in X\}| > n\}$

$S_1(w, E)$ caractérise les plus petits sous-ensembles de $\mathtt{F}_w$ dont l'intersection des éléments est contenue dans E, tandis que $S_2(w, n)$ caractérise les plus petits sous-ensembles de $\mathtt{F}_w$ dont la cardinalité de l'intersection des éléments est inférieure à n.

Le lemme suivant caractérise les solutions de *E_conj_rewrite(E,w)* et *N_conj_rewrite(n,wR)* avec les ensembles $S_1(w, E)$ et $S_2(wR, n)$.

Lemme 2 *Soient n un entier, w et wR des mots et E un ensemble. Soient deux problèmes E_conj_rewrite(E,w) et N_conj_rewrite(n,wR). Soit $b = \sqcap_{j=i_1}^{i_k} V_j$.*

1. *b est une solution de E_conj_rewrite(E,w) ssi $\{V_{i_1}, ..., V_{i_k}\} \subseteq \mathcal{V}_w$ et $\{E_{i_1}, ..., E_{i_k}\} \in S_1(w, E)$*
2. *b est une solution de N_conj_rewrite(n,wR) ssi $\{V_{i_1}, ..., V_{i_k}\} \subseteq \mathcal{V}_{w.R}$ et $\{E_{i_1}, ..., E_{i_k}\} \in S_2(w.R, n)$*

La formulation ensembliste du problème *N_conj_rewrite(n,wR)* est illustrée dans l'exemple qui suit.

Exemple 5 *Reprenons l'énoncé de l'exemple 3. On a $F_{aRecu.typeProduit} = \{E_5, E_6, E_7\}$. On a $S_2(aRecu.typeProduit, 3) = \{\{E_5, E_6, E_7\}\}$. On retrouve alors la solution de N_conj_rewrite(3,aRecu.typeProduit) : $V_5 \sqcap V_6 \sqcap V_7$.*

Dans la section suivante, nous montrons comment cette représentation ensembliste peut être transposée dans le cadre de Mannila et Toivonen (1997).

3.2.2 Calcul des réécritures dans un cadre de découverte de connaissances

Identification de $S_1(w, E)$ Dans ce contexte, le premier ensemble $S_1(w, E)$ peut se ramener au cadre de découverte de connaissances précédent de la manière suivante :

- la relation r est vide.
- le langage $\mathcal{L}_w$ est l'ensemble des parties de $\mathtt{F}_w$, i.e. $\mathcal{P}(\mathtt{F}_w)$.
- le prédicat, noté P_1, est défini de la façon suivante :
  Soient $X \in \mathcal{L}_w$, $X = \{E_1, ..., E_k\}$ et E un ensemble de valeurs.
  $P_1(E, X)$ est vrai ssi $\cap_{i=1}^{k}\{E_i\} \not\subseteq E$.
- la relation d'ordre est la relation d'inclusion $\subseteq$.

La théorie $\mathcal{T}h(\emptyset, \mathcal{L}_w, P_1)$ est alors l'ensemble des éléments de $\mathtt{F}_w$ qui vérifient le prédicat P_1. De plus, le prédicat P_1 étant anti-monotone, les notions de bordure positive et négative s'appliquent ici. Le théorème suivant permet de caractériser $S_1(w, E)$ en fonction de la bordure négative.

Théorème 1 *Soit le problème E_conj_rewrite(E,w).* $S_1(w, E) = \mathcal{B}d^-(\ Th(\emptyset, \mathcal{L}_w, P_1)\ )$

Nous pouvons de la même manière caractériser $S_2(w, n)$ dans le cadre théorique de Mannila et Toivonen (1997).

Identification de $S_2(w, n)$ Comme précédemment, la relation r est vide, $\mathcal{L}_w$ consiste en l'ensemble des parties de $\mathtt{F}_w = \{E_1, ..., E_p\}$, et la relation d'ordre est l'inclusion. Nous introduisons un nouveau prédicat $P_2(n, X)$ défini comme suit : $X = \{E_1, ..., E_k\} \in \mathcal{L}_w$, $P_2(n, X)$ est vrai ssi $| \cap_{i=1}^{k} E_i | > n$. Le prédicat $P_2(n, X)$ est anti-monotone par rapport à l'inclusion, ce qui garantit l'existence des bordures. Par conséquent, le théorème suivant donne une caractérisation de $S_2(w, n)$ en fonction de la bordure négative.

Théorème 2 *Soit le problème N_conj_rewrite(n,w).* $S_2(w, n) = \mathcal{B}d^-(\ Th(\emptyset, \mathcal{L}_w, P_2)\ )$

L'exemple qui suit illustre la formulation de $S_2(w, n)$ dans le cadre introduit ci-dessus.

Exemple 6 *Dans la suite de l'exemple 5, pour le mot* $aRecu.typeProduit$, $\mathcal{L}_{aRecu.typeProduit}$ *est l'ensemble des parties de* $F_{aRecu.typeProduit}$. *Le paramètre* n *du prédicat* P_2 *est égal à* 3. *La théorie* $\mathcal{T}h(\emptyset, \mathcal{L}_{aRecu.typeProduit}, P_2)$ *est égale à* $\{\{E_5\}, \{E_6\}, \{E_7\}, \{E_5, E_6\}, \{E_5, E_7\}, \{E_6, E_7\}\}$. *La bordure positive de* $\mathcal{T}h(\emptyset, \mathcal{L}_{aRecu.typeProduit}, P_2)$ *est donc* $\{\{E_5, E_6\}, \{E_5, E_7\}, \{E_6, E_7\}\}$, *et sa bordure négative est* $\{\{E_5, E_6, E_7\}\}$. *On retrouve le résultat de l'exemple 5.*

A partir de cette formalisation, il est possible d'utiliser des algorithmes de fouille de données pour résoudre ces problèmes, et plus particulièrement les algorithmes de découvertes des motifs fréquents.

4 Implémentation et expérimentations

Choix d'implémentation Un algorithme de réécriture pour un sous-langage de $\mathcal{FL}_0(\mathcal{O}_v)$ a été implémenté à l'aide d'un système de gestion de bases de données (SGBD), permettant ainsi de gérer efficacement de grands volumes de données et de traiter un grand nombre de transactions simultanées. Pour le moment, seul le calcul des solutions de *E_conj_rewrite(w,E)* a été implémenté sous forme d'un algorithme de fouille de données extérieur au SGBD. Les données sont expatriées vers ce programme chargé d'exécuter l'algorithme de fouille de données et de renvoyer les résultats au SGBD.

Nous utilisons l'algorithme *Apriori* (Agrawal et Srikant, 1994) pour trouver les solutions de *E_conj_rewrite(w,E)*. Cet algorithme est l'algorithme classique de découverte des motifs fréquents. Il effectue un parcours par niveau de l'espace de recherche, et utilise une stratégie d'élagage à partir de motifs de la bordure négative pour limiter le nombre de motifs générés. Les avantages de cet algorithme pour résoudre notre problème sont multiples. Cet algorithme,

tout en recherchant les motifs fréquents, découvre uniquement les motifs de la bordure négative, ce qui n'est pas le cas d'une grande partie des autres approches. De plus, sa stratégie et son efficacité ne dépendent pas du prédicat étudié. A l'opposé, une grande partie des autres algorithmes fondent leur efficacité sur des techniques propres au prédicat "être fréquent". L'efficacité de ce type d'approche pour un autre prédicat est donc difficilement prévisible.

L'implémentation d'$Apriori$ utilisée est une adaptation de l'implémentation C++ de Borgelt (2003). Cette implémentation est reconnue pour être l'implémentation la plus efficace d'$Apriori$ actuellement (Goethals et Javeed Zaki, 2003; Bayardo et al., 2004). L'implémentation initiale d'$Apriori$ a été modifiée de façon à rendre l'algorithme indépendant du prédicat étudié. Plus concrètement, pour pouvoir appliquer $Apriori$ à un nouveau prédicat, il suffit de définir les opérations propres à ce prédicat, et de le passer en paramètre de l'algorithme. Actuellement, en plus de différents prédicats liés aux motifs fréquents, le prédicat P_1 a été implémenté permettant ainsi de trouver les solutions de *E_conj_rewrite(w,E)* par $Apriori$. L'avantage de notre implémentation est donc de faciliter l'utilisation d'$Apriori$ pour résoudre d'autres problèmes que ceux de fouille de données, en évitant d'avoir à réécrire à chaque fois l'algorithme.

Expérimentations Nous nous concentrons ici sur l'expérimentation de la phase de résolution de *E_conj_rewrite(w,E)*. Cette phase étant l'une des plus coûteuse, son étude va nous permettre d'estimer le nombre de vues pouvant être traitées. Notons que cette borne correspond au nombre de vues que l'algorithme de réécriture a identifié comme étant pertinentes à la réécriture d'un atome de la forme $\forall mot.valeurs$, i.e. les vues de $\mathcal{V}_{mot}$. Ainsi cette borne conditionne uniquement la taille de l'entrée d'$Apriori$ et ne fait pas figure de limite sur le nombre de vues que l'algorithme de réécriture peut traiter.

Les expérimentations ont été réalisées sur des jeux de données synthétiques. Les jeux d'essais ont été créés à l'aide du générateur aléatoire d'Oracle, de façon à ce que la cardinalité des contraintes de valeurs soit égale à un entier n ou comprise entre 1 et un entier n tel que $n \in \{10, 20, 30, 40\}$. On a mesuré les temps d'exécution d'$Apriori$ sur ces jeux d'essais. Ces expérimentations ont été réalisées sur un pentium IV pro 2.6 Ghz avec 3 Go de mémoire.

Comme le montre la figure 1, lorsque la taille des contraintes de valeurs est petite, il est possible de prendre un grand nombre de vues en entrée (e.g. 15000 vues pour des contraintes de taille inférieure à 10). Dès que la taille des contraintes augmente, le nombre de vues que peut traiter l'algorithme, diminue (figure 1). En effet, plus la taille des contraintes est grande, plus les contraintes risquent de s'intersecter et que cette intersection ne soit pas incluse dans E. Par conséquent, le nombre de motifs intéressants est susceptible d'être important et le programmme dans ce cas, nécessite plus d'espace mémoire que disponible. Notons que lorsque les contraintes de valeurs sont de taille fixe (figure gauche de la figure 1), $Apriori$ est mis en difficulté plus rapidement que lorsque les contraintes sont de taille variable. Ceci s'explique par le fait que par exemple pour des contraintes de cardinalité au plus 10, les contraintes sont composées en moyenne de 5 valeurs.

D'une manière générale, ces jeux ont montré qu'il est possible de prendre en entrée pour la réécriture des atomes de la forme $\forall mot.valeurs$, jusqu'à 15000 vues. Néanmoins, il est difficile de comparer les performances de notre prototype avec d'autres applications de réécriture de requêtes dans la mesure où dans ce domaine les résultats théoriques ont toujours primé sur les résultats expérimentaux. A notre connaissance, (Pottinger et Halevy, 2001) est une des

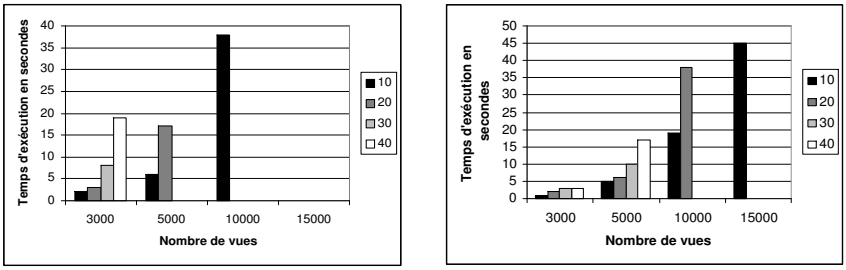

FIG. 1 – *Temps d'exécution d'*$Apriori$ *pour des contraintes de taille fixe puis variable*

rares références qui décrit l'évaluation des performances d'un algorithme de réécriture. Leur implémentation est considérée comme permettant le passage à l'échelle, car pour réécrire une requête, elle peut traiter près de 12000 vues. Aussi par rapport à ces résultats, notre approche de réécriture basée sur $Apriori$ permet d'envisager le passage à l'échelle.

5 Conclusions et perspectives

Dans cet article, nous avons confirmé l'intérêt des techniques de fouilles de données pour traiter le problème de la réécriture en présence de contraintes de valeurs. En effet dans $\mathcal{ALN}(\mathcal{O}_v)$, de nouveaux cas de réécritures engendrés par les contraintes peuvent bénéficier d'une formulation dans le cadre de découverte de connaissances de Mannila et Toivonen (1997). De plus, l'implémentation de notre approche basé sur une implémentation générique d'$Apriori$ permet de traiter un grand nombre de vues et d'envisager le passage à l'échelle de notre algorithme de réécriture. Toutefois, l'exécution d'$Apriori$ devient problématique quand une grande partie de l'espace de recherche doit être parcourue, i.e. quand il existe de grands motifs intéressants. Dans de telles configurations, pour le problème de la découverte des fréquents, des algorithmes ont été proposés afin de trouver plus efficacement les motifs de grande taille (Han et al., 2000; Uno et al., 2003; Flouvat et al., 2004). L'adaptation de certaines de ces approches pourrait donc permettre de traiter un nombre plus important de vues.

Références

Agrawal, R., T. Imielinski, et A. N. Swami (1993). Mining association rules between sets of items in large databases. In *SIGMOD conference, Washington*, pp. 207–216. ACM Press.

Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules in large databases. In *VLDB conference, Santiago de Chile, Chile*, pp. 487–499.

Baader, F., D. Calvanese, D. McGuinness, D. Nardi, et P. Patel-Schneider (Eds.) (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Bayardo, R., B. Goethals, et M. J. Zaki (Eds.) (2004). *ICDM Workshop on Frequent Itemset Mining Implementations*, Volume 126 of *CEUR Workshop Proceedings*.

Borgelt, C. (2003). Efficient implementations of Apriori and Eclat. In *ICDM Workshop on Frequent Itemset Mining Implementations*.

Borgida, A. T. et P. F. Patel-Schneider (1994). A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research 1*.

Flouvat, F., F. D. Marchi, et J.-M. Petit (2004). ABS: Adaptive borders search of frequent itemsets. In *ICDM Workshop on Frequent Itemset Mining Implementations*, Volume 126 of *CEUR Workshop Proceedings*.

Goethals, B. et M. Javeed Zaki (2003). Advances in frequent itemset mining implementations: Introduction to FIMI'03. In *ICDM Workshop on Frequent Itemset Mining Implementations*.

Han, J., J. Pei, et Y. Yin (2000). Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pp. 1–12.

Jaudoin, H., J.-M. Petit, C. Rey, M. Schneider, et F. Toumani (2005). Query rewriting using views in presence of value constraints. In F. W. Ian Horrocks, Ulrike Sattler (Ed.), *Description Logics*, pp. 112–119.

Jaudoin, H., F. Toumani, J.-M. Petit, et M. Schneider (2004). Utilisation d'un cadre de découverte des connaissances pour la réécriture de requêtes en présence de contraintes de valeurs. In J. L. Maitre (Ed.), *BDA 2004*, pp. 407–426.

Levy, A., A. Rajaraman, et J. Ordille (1996). Querying Heterogeneous Information Sources Using Source Descriptions. In *VLDB conference, Mumbai (Bombay), India*, pp. 251–262.

Mannila, H. et H. Toivonen (1997). Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery 1*(3), 241–258.

Pottinger, R. et A. Halevy (2001). Minicon : A scalable algorithm for answering queries using views. *The VLDB Journal 10*(2-3), 182–198.

Uno, T., T. Asai, Y. Uchida, et H. Arimura (2003). LCM: An efficient algorithm for enumerating frequent closed item sets. In *Workshop on Frequent Itemset Mining Implementations*.

Summary

In this paper, we show how data mining techniques can be used to solve efficiently the problem of query rewriting using views in presence of value constraints within the setting of description logic $\mathcal{ALN}(\mathcal{O}_v)$. Firstly, we give a formalization of this problem in the framework of $\mathcal{ALN}(\mathcal{O}_v)$. Then we show how to fit this formalization with a KDD framework. Therefore we reuse KDD scalable algorithmic solutions to solve this rewriting problem. We describe an implementation of this approach, and give some experimental results. The first results show the scability of this kind of approach.