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

Hélène Jaudoin
LIMOS-Cemagref
hjaudoin@isima.fr

Collaborations

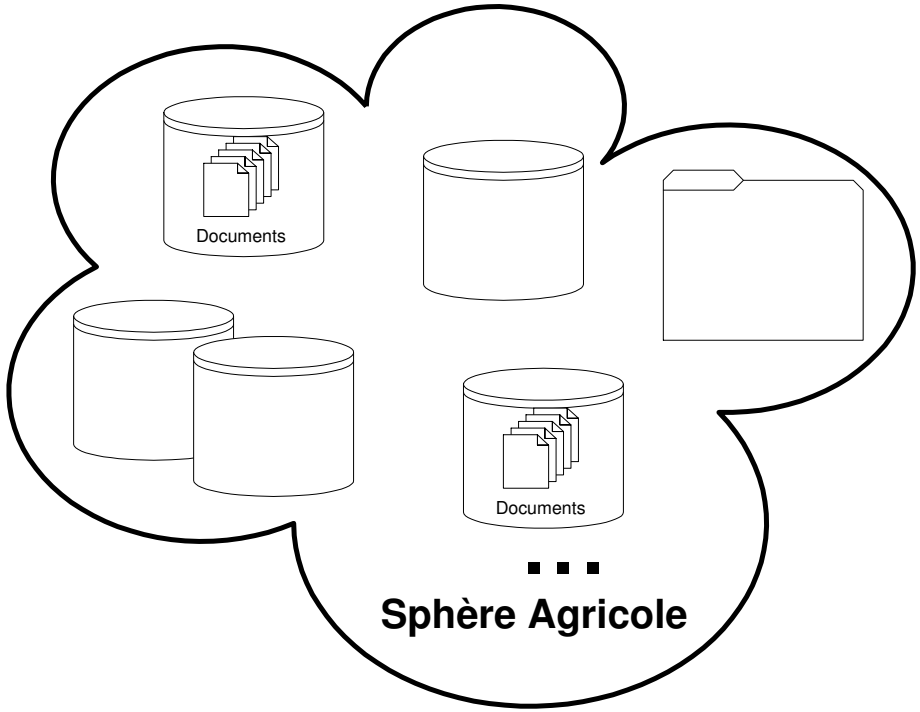
Laboratoires

- Cemagref
- LIMOS
- LIRIS

Membres

- F. Flouvat, M. Schneider, F. Toumani (LIMOS)
- J-M. Petit (LIRIS)

Application dans le domaine du développement durable



Charge:

Transparence des pratiques/qualité des produits
=> traçabilité

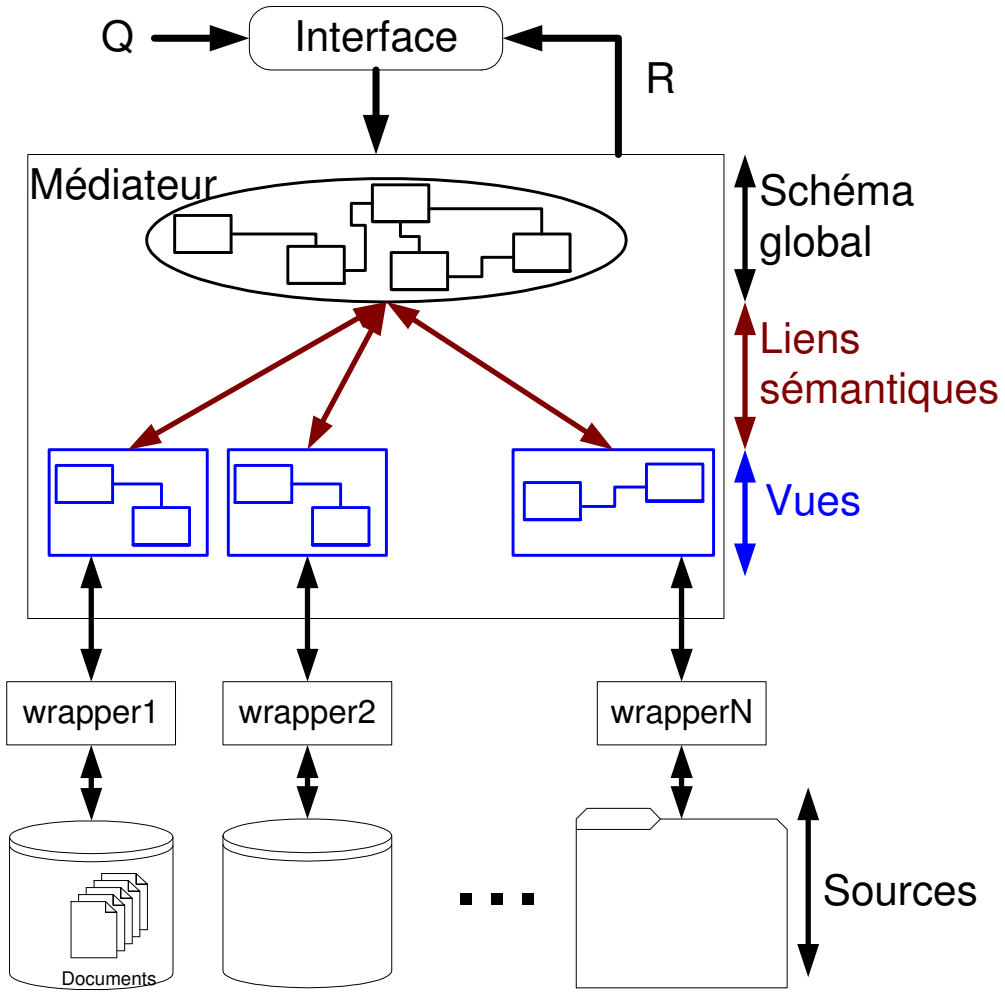
Conformité aux règlementations ministérielles

Caractéristiques du domaine

Système d'intégration de systèmes agricoles capable :

- accepter de nouvelles sources de données aisément
  - domaine versatile : processus d'informatisation du secteur agricole
- passer à l'échelle : $\Rightarrow$ traiter un grand nombre de sources de données
  - par exemple : 367000 exploitations agricoles professionnelles.

Système de médiation



Réécriture de requêtes en termes de vues

- Approche LAV et OWA
- Problème de réécriture en termes de vues
- En présence de contraintes de valeurs
  - Type de donnée énuméré : Oracle, DB2, MySQL
  - *Les voyages dont la destination est soit l'Europe, soit l'Australie.*
  - Type défini dans OWL : langage d'ontologie du Web du W3C
- Motivations :
  - distinguer des sources de données menant à des descriptions similaires
  - expression de requêtes typiques
  - optimisation du processus de réécriture

Plan

Plan

Logiques de description

- Formalisme de représentation des connaissances
  - Famille de logique
  - Pourvu d'une sémantique formelle $\Rightarrow$ raisonnements
  - A la base du langage OWL (W3C)
- Motivations pour l'utilisation des LD
  - Modélisation des contraintes de valeurs : constructeur $\mathcal{O}$
  - Raisonnement en présence des contraintes
- Formalisation et étude de la décidabilité du problème de réécriture

La logique $\mathcal{ALN}(\mathcal{O}_v)$

- Sous-langage de CLASSIC (Borgida & Patel-Schneider 94) : host individuals
  - Constructeurs $\mathcal{ALN}$: $\mathcal{P}, \neg A, \sqcap, \forall R.C, \leq n\ R, \geq n\ R$
  - Contraintes de valeurs $\mathcal{O}_v$: $\forall R.\{o_1, ..., o_n\}$
- Exemple de concept : $ParcelleCulturale \sqcap \forall arecu.categorie.\{C_1, C_4, C_8\} \sqcap \forall arecu. \geq 1\ Categorie$
- Forme normale : $\sqcap_{i=1}^{n} \forall w_i.P_i$
- Caractérisation de la subsomption structurelle adaptée au problème de réécriture :

  $\rightarrow$ calculer les réécritures

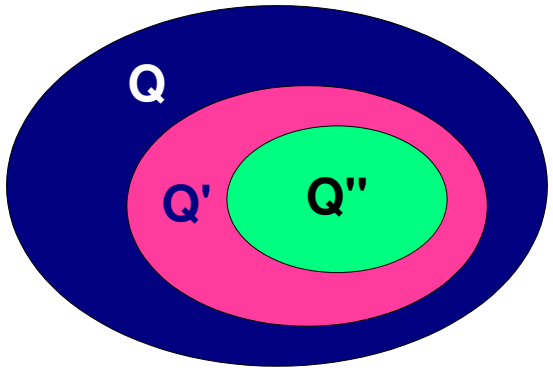
Modélisation du système de médiation

- Exemple de requête :
  $Q \equiv ParcelleCulturale \sqcap \forall sous.contratAR \sqcap$
  $\forall arecu.categorie.\{C_1, C_4, C_8\} \sqcap \forall arecu. \geq 1\ Categorie$
- Exemple de vues (OWA), $V_1, V_2 \in \mathcal{V}$
  - $V_1 \sqsubseteq ParcelleCulturale \sqcap \forall sous.contratAR$
  - $V_2 \sqsubseteq ParcelleCulturale \sqcap \forall arecu.categorie.\{C_1, C_4, C_8\} \sqcap$
    $\forall arecu. \geq 1\ Categorie$

Plan

Définition du problème

- Problème de réécriture de requêtes dans $\mathcal{ALN}(\mathcal{O}_v)$
- Reformuler Q en une réécriture Q' telle que
  - Q' est une expression en termes de $\mathcal{V}$
  - Q' est une disjonction de conjonctions de vues (langage : $\{\sqcap, \sqcup\}$)
  - $Q' \sqsubseteq Q$,
- Une réécriture conjonctive est une réécriture dans $\{\sqcap\}$
- Réécriture maximale
  - Q' est une réécriture
  - Q' est maximalement contenue dans Q



Problème de réécriture

- Problème : Calculer les réécritures conjonctives maximales de Q

$\Rightarrow$ i.e. calculer "les plus petites" conjonctions de vues subsumées par Q.

- Espace de recherche : $2^{|\mathcal{V}|}$
  i.e. toutes les conjonctions de vues de taille 1 à $|\mathcal{V}|$

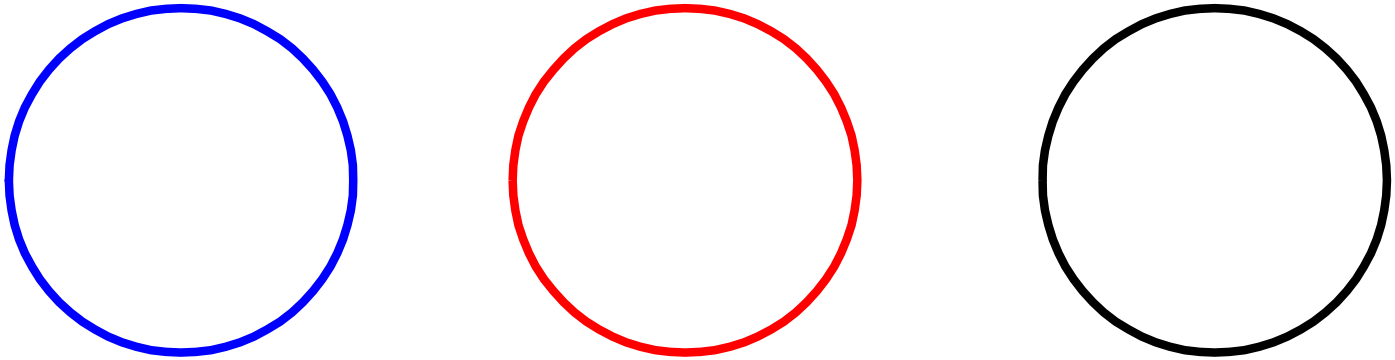
$\Rightarrow$ Approche de l'algorithme des paniers (Levy-Rajaraman-Ordille VLDB'96)

- Principe : réduire l'espace de recherche
  - identification des vues pertinentes à la résolution de Q
  - à l'aide de critères syntaxiques

Approche suivie : "Bucket Algorithm"

- Approche en deux étapes :
  1) construction des paniers

$Q \equiv ParcelleCulturale \sqcap \forall numCommune.\{63455, 63566, 63700\} \sqcap$
$\forall aRecu. \leq 4\, TypeProduit$



Approche suivie : "Bucket Algorithm"

- approche du "bucket algorithm"

  1) construction des paniers

$Q \equiv ParcelleCulturale \sqcap \forall numCommune.\{63455, 63566, 63700\} \sqcap \forall arecu. \leq 4\, TypeProduit$

Approche suivie

- approche du "bucket algorithm"

  1) construction des paniers

$Q \equiv ParcelleCulturale \sqcap \forall numCommune.\{63455, 63566, 63700\} \sqcap \forall arecu. \leq 4\, TypeProduit$

  2) produit cartésien : $\Rightarrow$ surensemble des réécritures maximalement contenues

Les "formes" des réécritures d'un panier $B(w, P)$

Soit $Q' \equiv V_{i_1} \sqcap \ldots \sqcap V_{i_k} \sqsubseteq Q$ (réécriture conjonctive maximale). Alors

- Q' est un cas classique de réécriture de $\mathcal{ALN}$
  - formée d'une seule vue
  - formée d'au plus p vues : inconsistance implicite $\forall v.\bot$
- Q' est un cas de réécriture engendré par les contraintes de valeurs, formée d'au plus $l+1$ vues :
  - $P = E$ et $\forall w.E_j \in V_j$ pour tout V_i, $j \in [i_1, i_k]$, et $\cap_{j=i_1}^{i_k} E_j \subseteq E$.
  - $P = (\leq\ n\, R_v)$, avec $R_v \in \mathcal{R}_v$, $\forall w R_v.E_j \in V_j$, pour tout V_j, $j \in [i_1, i_k]$, et $|\cap_{j=i_1}^{i_k} E_j| \leq n$.
  - nouvelle inconsistance implicite formée d'au plus $l+p$ vues.

Avec l cardinalité max. des contraintes de valeurs, p le nombre max de rôles des atomes.

Illustration (1) : les réécritures $\mathcal{R}_1$

- Entrée :
  - $Q \equiv \forall numDept.E \sqcap \forall aRecu. \leq 3 typeProduit$, $E = \{03, 43, 63\}$
  - $V_i, i \in \{1, ..., 3\}$ t.q. $V_i \sqsubseteq \forall numDept.E_i$,
    $E_1 = \{03, 63\}$ $\quad \mathcal{V}_{numDept} = \{V_1, V_2, V_3\}$
    $E_2 = \{01, 43, 63, 69\}$
    $E_3 = \{43, 55, 63\}$ $\quad F_{numDept} = \{E_1, E_2, E_3\}$
- Sortie : Réécritures $\mathcal{R}_1$ de $B(numDept, E)$

V_1 , $E_1 \subseteq E$
$V_2 \sqcap V_3$, $E_2 \cap E_3 \subseteq E$

Illustration (2) : les réécritures $\mathcal{R}_2$

- Entrée :
  - $Q \equiv \forall numDept.E \sqcap \forall aRecu. \leq 3typeProduit$
  - $V_i, i \in \{5,6,7\}$ t.q. $V_i \sqsubseteq \forall aRecu.typeProduit.E_i$,
    $E_5 = \{P_1, P_{10}, P_{15}, P_{20}, P_{27}\}$ $\quad \mathcal{V}_{aRecu.typeProduit} = \{V_5, V_6, V_7\}$
    $E_6 = \{P_1, P_{10}, P_{15}, P_{20}, P_{26}\}$
    $E_7 = \{P_1, P_{10}, P_{15}, P_{26}, P_{27}\}$ $\quad F_{aRecu.typeProduit} = \{E_5, E_6, E_7\}$
- Sortie : Réécritures $\mathcal{R}_2$ de $B(aRecu, (\leq 3typeProduit))$
  $B(aRecu, (\leq 3typeProduit)) = \{V_5 \sqcap V_6 \sqcap V_7\}$
  car $E_5 \cap E_6 \cap E_7 = \{P_1, P_{10}, P_{15}\}$ et
  $|E_5 \cap E_6 \cap E_7| \leq 3)$

Reformulation du calcul de $\mathcal{R}_1$ et $\mathcal{R}_2$

- Pour un mot donné v, on définit
  - $\mathcal{V}_v$ l'ensemble des vues subsumées par $\forall v.E_{i_j}$ (ex : $\{V_{i_1}, V_{i_2}\}$)
  - F_v l'ensemble des E_{i_j} associés à $\mathcal{V}_v$ (ex : $\{E_{i_1}, E_{i_2}\}$)

$\Rightarrow$ Formulation ensembliste du calcul de $\mathcal{R}_1$ et $\mathcal{R}_2$

- Trouver les sous-ensembles minimaux de F_v ($\{E_{i_1}...E_{i_k}\} \subseteq F_v$) t.q. :
  - $\cap_{j=1}^{j=k} E_{i_j} \subseteq E \Rightarrow S_1(E, v) \Rightarrow \mathcal{R}_1$
  - $|\cap_{j=1}^{j=k} E_{i_j}| \leq n \Rightarrow S_2(n, v) \Rightarrow \mathcal{R}_2$
- Problème posé : calculer $S_1(E, v)$ et $S_2(n, v)$

Vers une formulation des problèmes dans un cadre KDD

- Rappel des caractéristiques du domaine applicatif
  - Beaucoup de sources de données et de vues

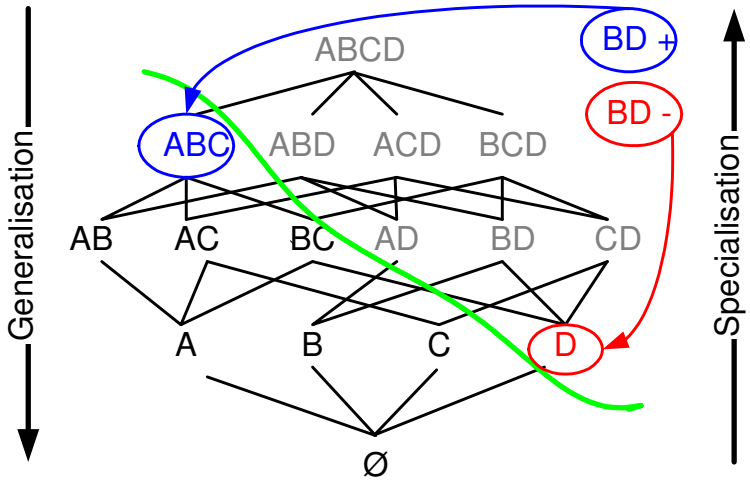
$\Rightarrow$ Problème du passage à l'échelle dans le calcul des réécritures $S_1(E, v)$ et $S_2(n, v)$

$\Rightarrow$ Idée : tirer profit des algorithmes de KDD pour la réécriture de requêtes

- Intérêt : existence d'algorithmes efficaces (site web : FIMI)

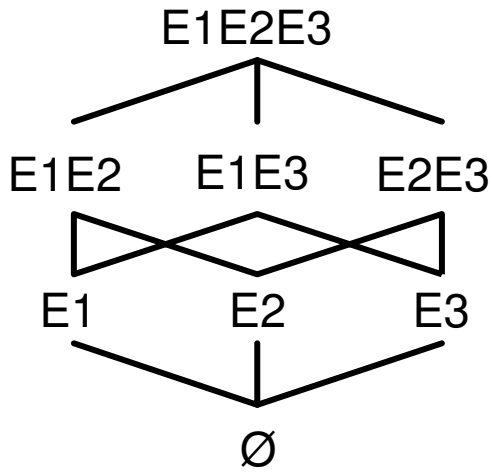
Un cadre de découverte des connaissances

- Mannila & Toivonen, DMKD'97
- une BD r, un langage $\mathcal{L}$, un prédicat P
- Objectif : $\mathcal{T}h(r, \mathcal{L}, P)$=$\{ \varphi \in \mathcal{L} \mid P(r, \varphi)$ est vrai$\}$
- Un ordre partiel $\preceq$ sur $\mathcal{L}$
- $P(r, X)$ doit être anti-monotone selon $\preceq$
- On peut définir $\mathcal{B}d^+(\mathcal{T}h(r, \mathcal{L}, P))$, $\mathcal{B}d^-(\mathcal{T}h(r, \mathcal{L}, P))$
- Exemple d'algorithme : Apriori



Passage vers le cadre de Mannila & Toivonen

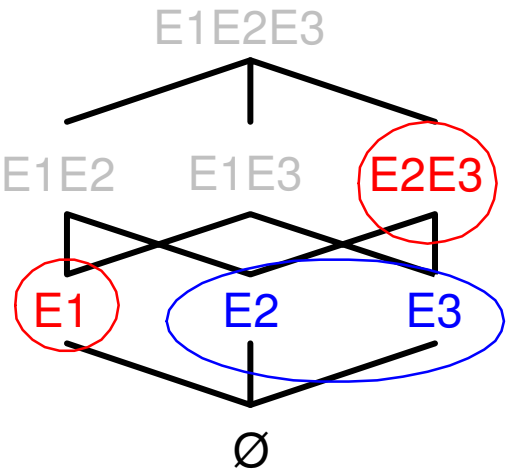
- Espace de recherche : sous-ensembles de F_v, i.e. $\mathcal{P}(F_v)$
- Ordre partiel : $\subseteq$
  - $\{E_1\} \subseteq \{E_1, E_2\}$
  - $\{E_1\}$ généralise $\{E_1, E_2\}$
- Intuition :
  - Calcul de $S_1(E, v)$ et $S_2(n, v)$ $\overset{?}{\leftrightarrow}$ Calcul de $\mathcal{B}d^-$



Exprimer $S_1(E, v)$ dans le cadre KDD

- BD : $\emptyset$; $\mathcal{L}_v = \mathcal{P}(F_v)$; ordre partiel : $\subseteq$
- Prédicat P_1 :
  $X \in \mathcal{L}_v$, $X = \{E_{i_1}, ..., E_{i_k}\}$, $P_1(E, X)$ est vrai ssi $\cap_{j=1}^{k} E_{i_j} \not\subseteq E$.

  - P_1 est antimonotone selon $\subseteq$
  - $S_1(E, v) = \mathcal{B}d^-(\mathcal{T}h(\emptyset, \mathcal{L}_v, P_1))$



Exprimer $S_2(n, v)$ dans le cadre KDD

- BD : $\emptyset$; $\mathcal{L}_v = \mathcal{P}(F_v)$; ordre partiel : $\subseteq$
- Prédicat P_2 : $X \in \mathcal{L}_v$, $X = \{E_{i_1}, ..., E_{i_k}\}$, $P_2(n, X)$ est vrai ssi $|\cap_{j=1}^{k} E_{i_j}| > n$
- P_2 est antimonotone selon $\subseteq$

$\Rightarrow$ $S_2(n, v) = \mathcal{B}d^-(\mathcal{T}h(\emptyset, \mathcal{L}_v, P_2))$

Plan

Perspectives d'implémentation

- A partir de cette formalisation dans le cadre de Mannila & Toivonen
- Possiblité d'utiliser plusieurs types d'algorithmes
  - par niveau : APriori
  - par sauts : Dualize and Advance
  - approche hybride : ABS
- Difficulté : adaptation des algorithmes à notre cadre
  - données : ensemble de valeurs
  - propriétés du Trie
  - test du prédicat

Calcul des réécritures $\mathcal{R}_1$

Adaptation d'APriori

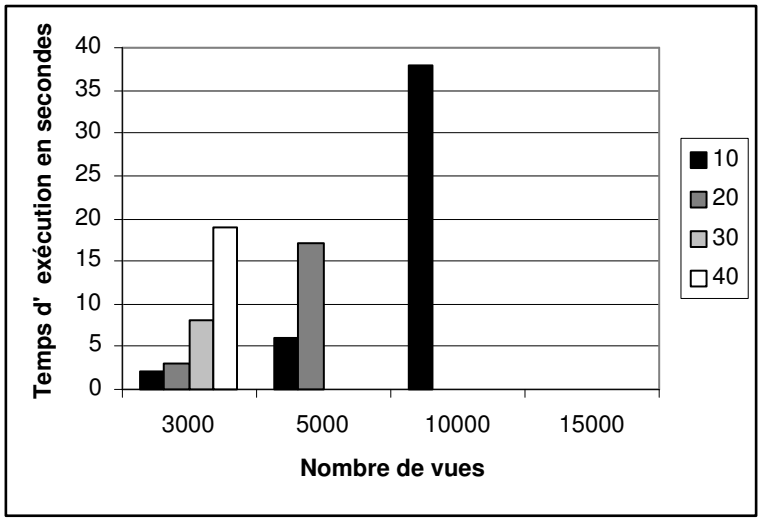
- Approche faiblement couplée : *ComputeEConj_1*
  - Procédures PLSQL
  - Trie relationnel : inspiré du système ATLaS (Wang, Zaniolo SDM 2003) Trie(Item, Pere, Niveau, values)
- Approche externe (F. Flouvat) : *ComputeEConj_2*
  - adaptation de l'implémentation de Borgelt (ICDM'03)
  - vers une implémentation générique ($\forall$ prédicat)

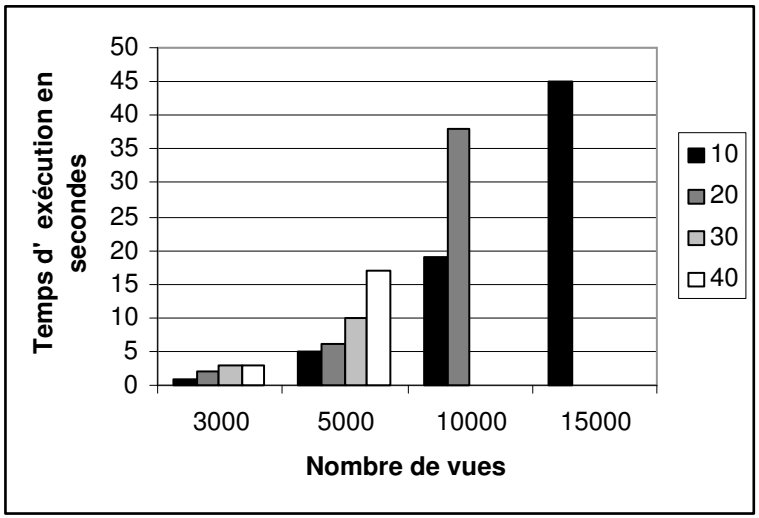
Comparaison des deux approches

	ComputeEConj_1	*ComputeEConj_2*
60 vues, 10 valeurs parmi 100	$21'09s$	$0s$
60 vues, 10 valeurs parmi 500	$1'13s$	$0s$
60 vues, 20 valeurs parmi 500	$17'33s$	$0s$
60 vues, 30 valeurs parmi 500	$56'58s$	$0s$
80 vues, 10 valeurs parmi 500	$8'29s$	$0s$

Résultat prévisible (Sarawagi, Thomas, Agrawal SIGMOD'98)

Expérimentations de *ComputeEConj_2*





Temps d'exécution d'Apriori pour des contraintes de taille fixe puis variable

Quelques remarques par rapport aux résultats

- Vers un algorithme de réécriture qui passe à l'échelle (15 000 vues pour traiter un atome)
- Pas de tests sur des jeux de données réels
  - taille des contraintes ?
  - taille des éléments dans la bordure (Configuration des vues)
- Tester d'autres algorithmes.

Conclusion

- Problème de la réécriture de requêtes en termes de vues en présence de contraintes de valeurs dans $\mathcal{ALN}(\mathcal{O}_v)$
- Solution : calcul des paniers, en utilisant une caractérisation en termes de bordures.
- Implémentation d'un algorithme : deux approches
  - faiblement couplé
  - externe (capable de traiter jusqu'à 15000 vues)
- Techniques de fouilles de données : le domaine de la réécriture, un autre champ d'application ?