Fragmentation des entrepôts de données XML

Hadj Mahboubi, Jérôme Darmont

Fragmentation des entrepôts de données XML

Hadj Mahboubi et Jérôme Darmont

Université de Lyon (ERIC Lyon 2)
5 avenue Pierre Mendès-France
69676 Bron Cedex
{hmahboubi,jdarmont}@eric.univ-lyon2.fr

Résumé. Les entrepôts de données XML proposent une base intéressante pour les applications décisionnelles qui exploitent des données hétérogènes et provenant de sources multiples. Cependant, les performances des SGBD natifs XML étant actuellement limitées en termes de temps de réponse et de volume des données, il est nécessaire de trouver des moyens pour les optimiser. Dans cet article, nous proposons une adaptation de la fragmentation horizontale dérivée définie dans le contexte relationnel aux entrepôts de données XML pour traiter ces deux problèmes. Nos expérimentations montrent que cette démarche permet de réduire le temps de traitement de requêtes décisionnelles XQuery de façon significative.

Mots clés : Entrepôts de données XML, performance, fragmentation.

1 Introduction

Actuellement, les applications décisionnelles exploitent de plus en plus de données hétérogènes et provenant de sources variées. Dans ce contexte, le langage XML peut grandement aider à l'intégration et à l'entreposage de données en vue de la fouille ou de l'analyse statistique ou en ligne (Beyer et al., 2005). Cependant, les requêtes décisionnelles sont généralement complexes du fait qu'elles impliquent de nombreuses jointures et agrégations. Par ailleurs, les systèmes de gestion de bases de données (SGBD) natifs XML présentent actuellement des performances médiocres quand les volumes de données sont importants ou que les requêtes sont complexes. Il est donc crucial lors de la construction d'un entrepôt de données XML de garantir la performance des requêtes XQuery qui l'exploiteront.

La fragmentation est une technique d'optimisation de performance qui consiste à diviser un ensemble de données en plusieurs fragments de telle façon que la combinaison de ces fragments produit l'intégralité des données source, sans perte ou ajout d'information. Dans le contexte relationnel, ce sont les tables (relations) qui sont partitionnées ; dans le contexte XML, ce sont les documents XML. La fragmentation peut intégrer plusieurs scénarios : la conception de systèmes distribués, le traitement des flux de données et l'échange de données dans les systèmes pair-à-pair.

Les travaux qui traitent de la fragmentation dans les entrepôts de données relationnels s'inspirent de ceux proposés dans les bases de données relationnelles et orientées objets. Cette

fragmentation peut être horizontale, verticale ou hybride. La fragmentation des données XML s'inspire également des travaux du domaine relationnel. Elle sert à améliorer les performances de requêtes XML distribuées. Elle peut être horizontale, verticale, hybride ou un *split* (spécifique à XML).

Cependant, à notre connaissance, il existe aucune approche pour la fragmentation des entrepôts de données XML. Nous proposons donc dans cet article une adaptation de la fragmentation horizontale dérivée aux entrepôts de données XML. Cette fragmentation consiste à partitionner les faits d'un entrepôt XML en fonction d'un ou plusieurs fragments horizontaux des dimensions.

Pour valider cette approche, nous avons généré un entrepôt de données XML et une charge de requêtes décisionnelles XQuery grâce au banc d'essais XWB (Mahboubi et Darmont, 2006). Nous avons ensuite évalué les temps de réponse de la charge avec et sans application de notre démarche de fragmentation. Nous avons pu vérifier qu'appliquer une fragmentation horizontale dérivée à un entrepôt de données XML réduit le temps d'exécution des requêtes XQuery de façon significative.

Le reste de cet article est organisé comme suit. Nous présentons en premier lieu un aperçu des travaux existants pour la fragmentation des entrepôts relationnels et des bases de données XML (Section 2). Nous introduisons ensuite une formalisation des documents XML et des expressions de chemin, et nous proposons une architecture de référence pour les entrepôts de données XML (Section 3). Puis nous présentons notre démarche de fragmentation des entrepôts de données XML (Section 4) et les expérimentations que nous avons menées pour la valider (Section 5). Finalement, nous concluons et présentons des perspectives de recherche (Section 6).

2 Etat de l'art

2.1 Fragmentation des entrepôts de données relationnels

Dans le contexte des entrepôts de données relationnels, Bellatreche (2003) identifie trois types de fragmentation : la fragmentation horizontale, la fragmentation verticale et la fragmentation hybride (fragmentation horizontale suivie d'une fragmentation verticale ou l'inverse).

Les travaux qui traitent ce problème sont peu nombreux. Certains utilisent une fragmentation verticale afin de construire des index. En effet, un index de projection ressemble à un fragment vertical de relation. Datta et al. (1999) ont par exemple développé un index, nommé *Curio*, en se basant sur une fragmentation verticale de la table des faits. Noaman et Barker (1999) utilisent la fragmentation horizontale pour construire un entrepôt de données réparti. Pour cela, ils emploient une stratégie descendante de répartition qui est couramment utilisée pour la conception des bases de données réparties. Bellatreche et Boukhalfa (2005) appliquent une fragmentation horizontale sur un schéma en étoile en se basant sur un ensemble de requêtes. Cette approche est basée sur des algorithmes génétiques et vise à contrôler le nombre de fragments afin d'optimiser leur coût de maintenance.

Finalement, Wu et Buchmann (1997) recommandent d'utiliser une fragmentation à la fois horizontale et verticale dans un entrepôt de données, afin d'optimiser le traitement des requêtes et d'éviter le balayage de grandes tables. Les auteurs utilisent une fragmentation horizontale dérivée qui consiste à partitionner horizontalement la table des faits en fonction d'une ou plu-

sieurs tables de dimension. La table des faits peut aussi être fragmentée verticalement. Toutes les clés étrangères de la table de faits y sont alors partitionnées. Aucun algorithme de fragmentation n'est cependant proposé.

2.2 Fragmentation des bases de données XML

Plusieurs travaux traitent de la fragmentation des documents/données XML. Bremer et Gertz (2003) proposent une approche de distribution de données XML par fragmentation. Les auteurs proposent une fragmentation verticale et horizontale. La spécification (définition) des fragments XML est basée sur un langage d'expression de chemins, dérivé de *XPath*, nommé *XF*. Un fragment vertical XML est le résultat d'évaluation d'une expression *XF* sur un schéma global des données XML, nommée *RG*. Afin de garantir la cohérence et la disjonction des fragments, les auteurs associent à chaque expression de sélection une expression d'exclusion. Ces expressions définissent les fragments à exclure ou à ignorer. Un fragment horizontal est obtenu par une expression *XF* plus complexe. Elle intègre des conditions sur les éléments, attributs et valeurs et utilisent la notion de voisinage ("branching [ ]").

Bonifati et al. (2004) modélisent des fragments XML verticaux pour une architecture XML pair-à-pair. Un fragment XML est obtenu et identifié par une expression de chemin (*XP, root-to-node path expression*). Il est ensuite stocké dans un site de l'architecture. De plus, et pour chaque fragment, deux types d'expressions de chemins sont associées : des expressions de chemins *super fragments* et *child fragments* reliées. Cet aspect permet d'identifier les différents sites ou fragments nécessaires pour le traitement d'une requête ou l'expression de chemins sur l'architecture. Dans d'autre travaux, Bonifati et al. (2006) traitent la problématique de la limitation d'espace mémoire pour le traitement de requêtes XML. Pour cela, les auteurs utilisent une fragmentation dirigée par les contraintes structurelles du document XML (taille, largeur et profondeur), qui utilise des heuristiques et des statistiques pour partitioner le document XML ciblé par une requête.

Bose et Fegaras (2005) utilisent des fragments XML pour le transfert de données XML dans une architecture pair-à-pair. Chaque fragment est identifié par un identifiant *ID*. Les fragments sont aussi reliés. Les auteurs proposent un schéma de fragmentation, nommé *Tag Structure*, qui définit la structure des données et fournit des informations sur la fragmentation.

En plus de la fragmentation horizontale et verticale, Ma et Schewe (2003) définissent un nouveau type de fragmentation, nommé *split*. Cette fragmentation est inspirée de celle utilisée dans les bases de données orientées objets. Elle consiste à partitioner un élément du document XML en sous-éléments et à leurs attribuer des références pour les relier. Ces références sont aussi ajoutées à la DTD qui définit le document XML source. Pour cela, les auteurs définissent une extension de la DTD et une autre du langage de requêtes XML-QL.

Finalement, Andrade et al. (2006) proposent une fragmentation pour une collection XML (ensemble documents XML homogènes) et non pour un seul document XML. Elle est basée sur l'algèbre TLC (Paparizos et al., 2004). Les auteurs présentent une formalisation des techniques de fragmentation traditionnelles pour une collection XML (horizontale, verticale et hybride) et des règles qui assurent une fragmentation correcte. Des expérimentations sur les trois types de fragmentation ont été menées sur un système nommé PartiX implémenté pour l'occasion (Andrade et al., 2005). PartiX assure l'exécution parallèle de requêtes XQuery sur un ensemble de SGBD natifs XML. Les résultats des expérimentations ont montré un gain de performance considérable pour les trois types de fragmentation.

Les approches présentées précedemment sont inspirées de celles avancées dans les contextes relationnel et orienté objets. Ils diffèrent dans la manière de définir formellement les fragments (algèbre TLC, expressions de chemins *XF* et langage XML-QL). Ma et Schewe (2003) et Gertz et Bremer (2003) ont également défini des règles pour une fragmentation correcte : la complétude, la disjonction et la reconstruction. Ces règles garantissent une décomposition syntaxiquement correcte des requêtes distribuées et permettent d'assurer un résultat cohérent lors de la reconstruction. Par ailleurs, les approches proposées pour traiter les flux de données utilisent une fragmentation verticale et ne supportent pas la fragmentation horizontale. Or, cette dernière peut être plus performante dans le processus de reconstruction des résultats des sous-requêtes d'une requête distribuée. En effet, la reconstruction est réalisée par union. Pour des fragments verticaux, cette reconstruction nécessite des opérations de jointure qui peuvent être coûteuses.

À notre connaissance, il n'existe aucune approche pour la fragmentation des entrepôts de données XML. Cependant, les techniques existantes dans les contextes relationnel et XML peuvent être adaptées à ces entrepôts.

3 Définitions et contexte

Nous présentons dans cette section un cadre formel pour les documents XML et les expressions de chemin. Nous proposons ensuite une architecture de référence pour les entrepôts de données XML.

3.1 Concepts de base

Définition 1 *Un document XML est représenté par un graphe étiqueté. Il est constitué de nœuds qui représentent les éléments et les attributs du document. Les arcs du graphe sont étiquetés par les noms des éléments et des attributs. Les nœuds feuilles du graphe stockent les valeurs des éléments et des attributs. Un nœud peut référencer un autre noeud du même document XML ou d'un autre document XML.*

Soient E l'ensemble des noms d'éléments distincts, A l'ensemble des noms d'attributs distincts et V l'ensemble des valeurs des éléments et des attributs. Un graphe de données XML peut se représenter par $\Gamma :=< t, l, \psi >$, où t est un arbre ordonné, l est la fonction qui étiquette un nœud de t avec des symboles appartenant à $E \cup A$ et ψ est la fonction de correspondance entre les nœud de t et leurs valeurs dans V. Le nœud racine de l'arbre t est dénoté $root_t$.

Définition 2 *Une expression de chemin P est une séquence : $root_t/e_1/.../\{e_n/@a_k\}$, où $\{e_1, ..., e_n\} \in E$ et $@a_k \in A$. L'expression P peut contenir le symbole $*$ qui indique un élément quelconque de E et "//" qui indique une séquence d'éléments $e_i/.../e_j$ telle que $i < j$. Le symbole [i] peut aussi être affecté à un élément e_i. Il indique alors la position de l'élément dans l'arbre.*

Définition 3 *Un prédicat simple est défini par l'expression $p := P\theta[value \mid \emptyset_{XPath}(P) \mid Q]$, où P et Q sont des expressions de chemins, $\theta \in \{=, <, >, \leq, \geq, \neq\}$, $value \in D$ et $\emptyset_{XPath}$ est une fonction XPath (W3C, 2007).*

Un prédicat de restriction peut donc être exprimé par : $P_\sigma := P_1\beta....\beta P_n$*, où* P_n *est un prédicat simple et* $\beta \in \{and, or, xor\}$.

Définition 4 *Un fragment d'un entrepôt de données XML est constitué d'un ensemble de graphes qui représentent les faits et les dimensions fragmentés. Il représente un sous-schéma de l'entrepôt de données XML.*

3.2 Entrepôt de données XML

3.2.1 Architectures existantes

Plusieurs travaux traitent de la conception et de la construction des entrepôts de données XML. Ils se basent sur un schéma en étoile pour modéliser un entrepôt de données XML et utilisent une collection de documents XML pour représenter les faits et les dimensions. Cet aspect favorise un stockage natif et une interrogation via des langages de requêtes XML.

Pokorný (2002) propose une structure de données, nommée *XML-star schema*. Un schéma en étoile XML, selon Pokorný, est représenté par un ensemble de documents XML logiquement reliés. Hümmer et al. (2003) utilisent des documents XML pour le transfert de cube de données. Les faits et les dimensions sont définis par un modèle nommé *XCube*. Rusu et al. (2005) se basent sur le langage XQuery pour construire un entrepôt de données XML. Park et al. (2005) proposent une plate-forme pour l'analyse en ligne des documents XML, nommée *XML-OLAP*. Les auteurs se basent sur un entrepôt de données XML où les faits sont représentés par une collection de documents XML et les dimensions et leurs hiérarchies sont stockées dans des documents XML. Finalement, Boussaïd et al. (2006) définissent un entrepôt par un schéma XML. Les auteurs implémentent un prototype, nommé *X-Warehousing*, pour l'entreposage des données complexes. Chaque fait, ainsi que les dimensions correspondantes sont stockés dans un document XML.

Ces travaux diffèrent principalement dans la manière de représenter les faits et les dimensions, ainsi que par le nombre de documents XML utilisés. Pour résumer, un entrepôt de données XML peut en effet avoir différentes représentations :

1. une collection de documents XML, où chaque document stocke un fait et les dimensions correspondantes (X-Warehousing) ;
2. un document XML qui représente les faits et un autre qui stocke toutes les dimensions (XCube) ;
3. une collection de documents XML avec un document par fait et par dimension (XML-OLAP).

3.2.2 Architecture de référence

Une évaluation de performance des différentes représentations d'entrepôts de données XML, citée dans la section 3.2.1, a été menée par Boukraa et al. (2006). Les auteurs ont généré trois entrepôts de données mamographiques XML qui respectent ces représentations. Les documents XML des entrepôts ont été stockés dans une base de données native XML, eXist (Meier, 2002). Les auteurs ont comparé les temps de traitement d'une charge de requêtes XQuery. Les résultats de cette évaluation ont montré que représenter les faits par un document XML et chaque dimension par un autre document XML permet d'obtenir les meilleures performances.

Nous avons donc choisi de représenter les faits par un seul document XML et chaque dimension par un document XML (Mahboubi et Darmont, 2006). Ce choix d'architecture permet de minimiser le coût de parcours des faits, de modéliser différents schémas (étoile, flocon de neige et constellation).

Les graphes correspondants aux documents XML qui constituent notre entrepôt XML sont représentés par les figures 1 et 2. Le graphe *facts.xml* représente les faits (Figure 1 (a)). Le nœud racine est *factdoc*. Il est composé de nœuds *fact* qui définissent les faits. Chaque nœud *fact* est composé d'un ou plusieurs nœuds *measure* qui représentent les mesures et leurs valeurs, ainsi que de nœuds *dimension* qui représentent les noms et les identificateurs des dimensions associées au fait.

Le graphe $dimension_d.xml$ (Figure 1 (b)) permet l'instanciation de la dimension d. Il permet de stocker cette dimension et ses niveaux hiérarchiques. Le nœud racine est *dimension*. Il est composé d'un ou plusieurs nœuds *Level*. Un nœud *Level* définit un niveau de la hiérarchie. Chaque nœud *Level* possède des nœuds fils *instance* qui contiennent les attributs et leurs valeurs. De plus, un nœud *instance* possède un nœud *rollup* qui assure le lien entre les niveaux de la hiérarchie.

Finalement, le graphe *dw-model.xml* (Figure 2) définit le schéma de l'entrepôt de données XML (métadonnées). La racine de ce graphe, *dw-model*, est composée de nœuds *dimension* et *factdoc*. Un nœud *dimension* représente une dimension, ses niveaux hiérarchiques et le type de ses attributs. Un noeud *factdoc* représente un document de faits.

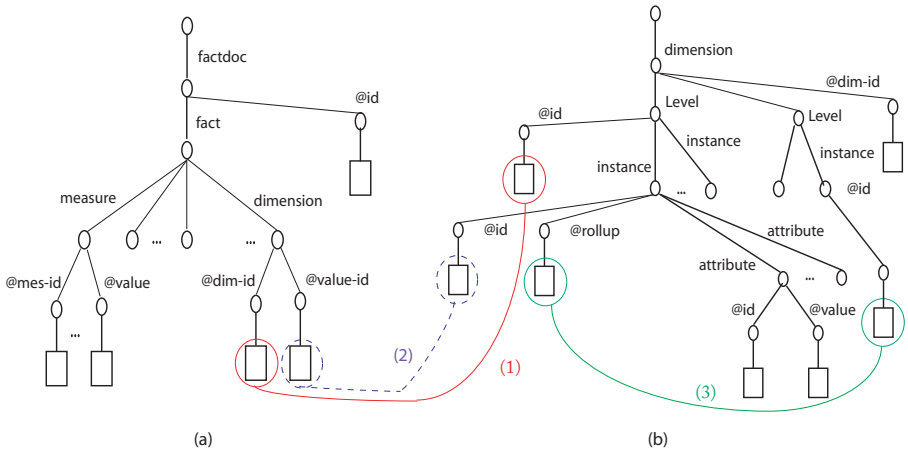

FIG. 1 – *Graphes des documents facts.xml (a) et* $dimension_d.xml$ *(b)*

4 Fragmentation des entrepôts de données XML

4.1 Motivation

Les approches qui traitent de la fragmentation des bases de données XML se limitent à la fragmentation horizontale primaire (Ma et Schewe, 2003; Gertz et Bremer, 2003), qui est appliquée sur un seul document XML. Cette fragmentation minimise le nombre d'entrées/sorties

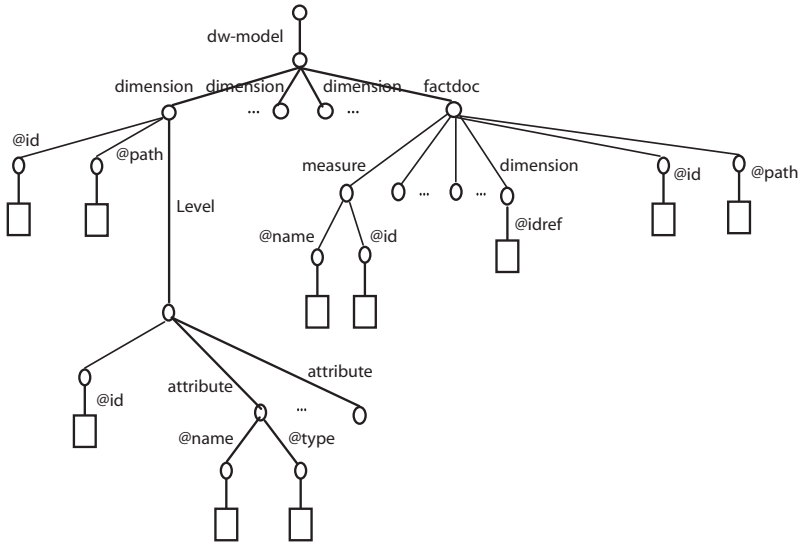

FIG. 2 – *Graphe du document dw-model.xml*

pendant l'interrogation du document XML. Or, les entrepôts de données XML exploitent des requêtes qui comportent de multiples opérations de jointures qui ciblent plusieurs documents à la fois. Ces travaux n'y sont donc pas adaptés.

Par ailleurs, les travaux qui proposent une fragmentation des entrepôts de données relationnels recommandent une fragmentation horizontale dérivée (Wehrle et al., 2005; Bellatreche et Boukhalfa, 2005). Elle consiste à diviser les faits en fonctions d'un ou plusieurs fragments obtenus par une fragmentation horizontale des dimensions. Cette fragmentation est utile lors du traitement de requêtes de jointure, ce qui est le cas des requêtes décisionnelles, en éliminant les jointures inutiles.

Nous proposons donc d'appliquer une fragmentation horizontale dérivée à notre entrepôt de données XML. Notre démarche de fragmentation consiste à fragmenter les graphes des dimensions en fonction d'une charge de requêtes. Le graphe des faits est par la suite fragmenté en fonction de ces fragments horizontaux. La fragmentation horizontale exploite un algorithme défini dans le contexte des bases de données relationnelles (Özsu et Valduriez, 1999), qui a été adapté aux entrepôts de données relationnels par Noaman et Barker (1999) et Bellatreche (2000). Nous l'adaptons à notre tour aux entrepôts de données XML.

4.2 Définitions préalables

Une fragmentation horizontale dérivée pour un entrepôt de données XML, modélisé selon un schéma en étoile ou ses dérivés flocon de neige, nécessite trois informations (Bellatreche, 2000).

1. **Noms des graphes propriétaires et du graphe membre** : Les graphes propriétaires $G_{dimension_d}$ représentent les dimensions. Ces graphes seront horizontalement fragmentés. Le graphe membre G_{facts} représente les faits.

2. **Nombre de fragments horizontaux des dimension** : Il est obtenu par l'application d'une fragmentation horizontale sur un ou plusieurs graphes $G_{dimension_d}$.
3. **Qualification de la jointure** : Elle permet de fragmenter le graphe G_{facts} en fonction des fragments horizontaux des graphes $G_{dimension_d}$. Cette qualification (opération de semi-jointure) est représentée dans la figure 3. Elle est constituée d'une conjonction de deux expressions de chemin.

document($facts.xml$)/FactDoc/dimension/[@dim-id=document($dimension_i.xml$)
/dimension/Level/@id]
and
document($facts.xml$)/FactDoc/dimension/[@value-id=document($dimension_i.xml$)
/dimension/Level[@id=@dim-id]/instance/@id]

FIG. 3 – *Qualification de la jointure*

4.3 Démarche de fragmentation

La fragmentation horizontale des $G_{dimension_d}$ se base sur une charge de requêtes. Elle consiste à déterminer les prédicats de sélection utiles pour une fragmentation (Noaman et Barker, 1999; Özsu et Valduriez, 1999; Bellatreche, 2000, 2003) puis à construire des fragments horizontaux. Cette démarche comprend cinq étapes.

1. **Enumération des prédicats de sélection**. Cette étape consiste à identifier les prédicats de selection d'une charge de requêtes. Pour cela, nous effectuons une analyse syntaxique de la charge. La figure 4 montre un exemple de requête décisionnelle XQuery. Une expression p (prédicat de restriction) est extraite de la clause *where*. Dans cet exemple $p := [a/attribute/@name = 'cust\_city' and\ a/attribute/@value = 'Lyon']$.

```
for $a in //dimensionData/classification/Level
[@node='customers']/node, $x in //CubeFacts/cube/Cell
let $q := $b/attribute[@name='cust_name']/@value
let $q := $b/attribute[@name='cust_zip_code']/@value
where $a/attribute/@name='cust_city'
and $a/attribute/@value='Lyon'
and $x/dimension /@node=$a/@id
and $x/dimension/@id='customers'
group by(cust_name,@cust_zip_code)
return name='cust_name', aggregation(sum, quantity)
```

FIG. 4 – *Exemple de requête décisionnelle XQuery*

2. **Attribution des prédicats de selection aux graphes XML des dimensions**. Cette étape consiste à affecter à chaque graphe $G_{dimension_d}$ un ensemble de prédicats, $Exp_{dimension}$. Cette affectation est basée sur l'expression de chemin $root_t//dimension/@id$ *='Nom de la dimension'*. Ainsi, nous appliquons une identification des graphes de dimension à fragmenter. L'ensemble des graphes à fragmenter est noté $G_{candidat}$.
3. **Vérification des règles de complétude et de minimalité des prédicats de sélection**. L'objectif de cette étape est de s'assurer qu'un graphe candidat est divisé en au moins

deux fragments. Pour cela, nous appliquons l'algorithme des *COM-MIN* (Özsu et Valduriez, 1999; Bellatreche, 2000). Cet algorithme fournit en sortie un ensemble complet et minimal de prédicats pertinents pour la fragmentation $Exp'_{dimension}$.

4. **Fragmentation des graphes $\mathbf{G_{candidat}}$.** Cette étape consiste à construire un schéma de fragmentation des graphes de dimension $G_{dimension_d}$ par l'application d'un algorithme de fragmentation primaire (Özsu et Valduriez, 1999). Un fragment horizontal est donc obtenu par l'application des prédicats de $Exp'_{dimension}$ sur les graphes $G_{candidat}$.
5. **Fragmentation du graphe $\mathbf{G_{facts}}$.** Le graphe des faits est fragmenté en fonction des fragments définis par le schéma de fragmentation primaire horizontale. Dans cette étape, les fragments de faits sont obtenu par l'application de la qualification de jointure de la figure 3. Le nombre de fragments de faits peut être évalué comme suit :
   $N_{fact} = \prod_{d=1}^{n} N_d$, où n représente le nombre de dimensions et N_d le nombre de fragments pour une dimension d. Ce nombre représente aussi le nombre de sous-schémas obtenus par fragmentation.

La figure 5 montre un exemple de fragment XML. Les graphes $dimension'_d$*.xml* sont obtenus par fragmentation horizontale. Le graphe dérivé $facts'$*.xml* est obtenu par des opérations de semi-jointure entre les graphes $dimension'_n$*.xml* et *facts.xml*.

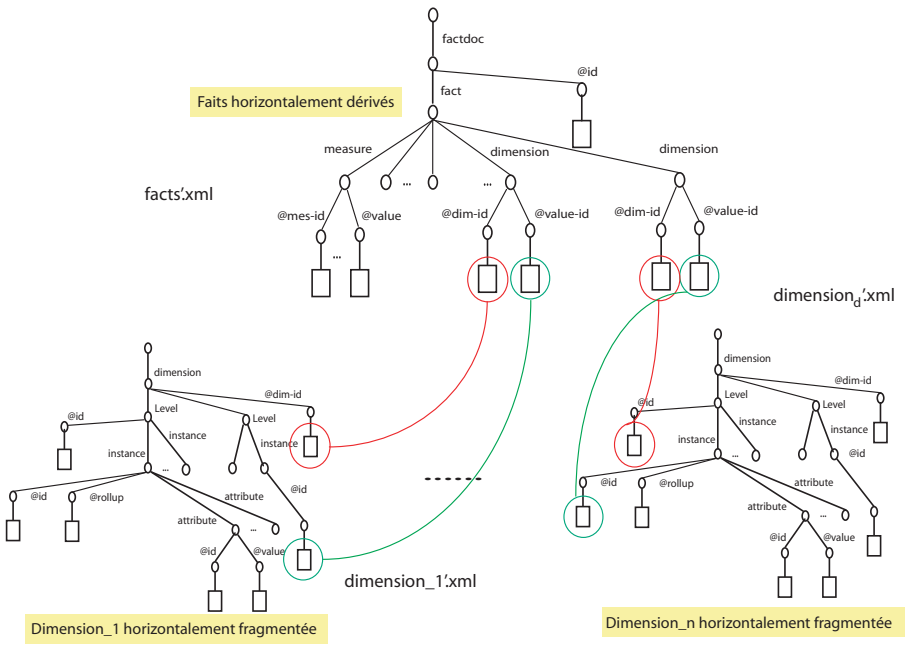

FIG. 5 – *Exemple de fragment d'entrepôt de données XML*

Finalement, nous représentons le schéma de fragmentation obtenu par un sous-graphe dans le graphe qui représente le document *dw-model.xml*. Ce sous-graphe représente les fragments obtenus par fragmentation horizontale dérivée (figure 6). Son nœud racine est *fragments*. Il est composé de nœuds *fragment* qui définissent les differents fragments. Le nœud *fragment* est à

son tour composé de nœuds *dimension* qui représentent les fragments horizontaux des dimensions. Chaque nœud *dimension* est identifié par les nœuds *attribute* et *value* qui définissent le prédicat de selection. Ce sous-graphe nous permet de simuler la répartition des fragments, chaque fragment (nœud *fragment*) étant identifié par un chemin (attribut *path*), et l'identification des fragments nécessaires à l'exécution d'une requête.

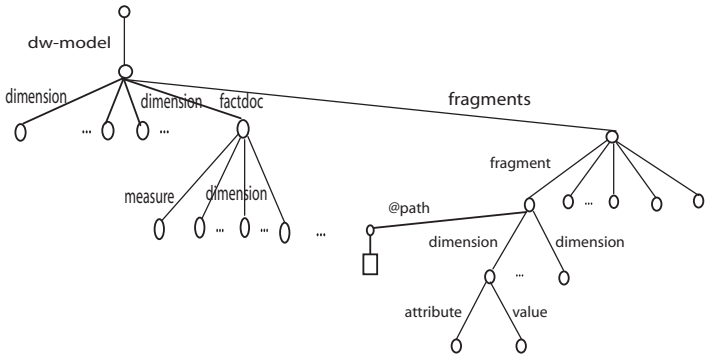

FIG. 6 – *Schéma de fragmentation*

5 Expérimentation

5.1 Conditions expérimentales

Nous avons généré un entrepôt de données XML à partir du banc d'essai XWB (*XML Data Warehouse Benchmark*) (Mahboubi et Darmont, 2006). XWB propose un entrepôt de données XML et une charge de requêtes décisionnelles XQuery. L'entrepôt XWB respecte la représentation présentée dans la section 3.2.2. Il est constitué d'une table de faits *sales* stockée dans le document *Fact.xml* et de quatre dimensions *products, customers, supplies* and *date* stockées dans les documents $dimension_{nom-dimension}.xml$. Ces documents sont stockés au sein du SGBD natif XML X-Hive (X-Hive-Corporation, 2007). Ce système permet le stockage de documents volumineux et implémente un moteur d'exécution de requêtes XQuery. Nous avons ajouté aux requêtes de la charge XWB des prédicats de restriction supplémentaires afin d'obtenir un nombre de fragments significatif. Cette nouvelle charge est constituée de dix requêtes qui comportent dix prédicats de sélection et quinze jointures. Pour des raisons de place, nous ne la représentons pas entièrement dans cet article, mais elle disponible en ligne[1]. Nous avons effectué nos expérimentations sur une machine dotée d'un processeur Intel Pentium 3 GHz avec 1 Go de mémoire et un disque dur IDE.

5.2 Résultats

Nous avons appliqué notre démarche de fragmentation sur l'entrepôt de données XWB. Nous avons obtenu seize sous-schémas (fragments de l'entrepôt) que nous avons stockés dans

[1] http ://eric.univ-lyon2.fr/∼ hmahboubi/file/XWB/workload.pdf

des collections différentes. Cela permet de simuler la répartition des fragments et l'exécution en parallèle des requêtes distribuées.

Nous avons exécuté notre charge sur l'entrepôt original et l'entrepôt fragmenté. La figure 7 présente les temps d'exécution de chaque requête de la charge. Elle montre que notre approche de fragmentation permet d'obtenir une amélioration des temps de traitement de 46 % en moyenne.

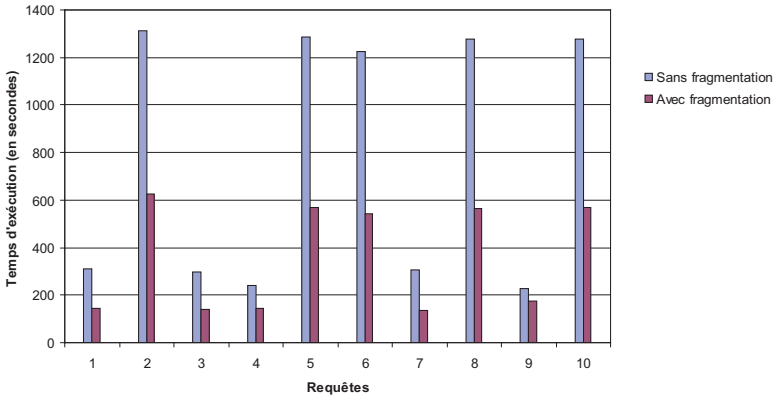

FIG. 7 – *Résultat des expérimentations*

6 Conclusion

Dans cet article, nous avons tout d'abord présenté les différentes techniques de fragmentation des entrepôts de données relationnels et des bases de données XML. Nous avons ensuite proposé une adaptation de ces techniques pour la fragmentation horizontale dérivée des entrepôts de données XML. Les premières expérimentations que nous avons menées pour valider notre approche montrent qu'elle permet de réduire le temps d'exécution de requêtes décisionnelles XQuery de façon significative.

Cependant, ces expérimentations doivent être approfondies. Pour cela, un mécanisme automatique de traitement de requêtes distribuées doit être défini pour permettre la décomposition des requêtes et leur distribution sur les différents fragments (Andrade et al., 2005). Des expérimentation sur différentes configuration d'entrepôts de tailles variés peuvent aussi être envisagées. Elles nous permettront d'observer les performances des requêtes lors d'une fragmentation de ces configurations d'entrepôt. Par ailleurs, les techniques existantes de fragmentation horizontale dérivée génèrent un grand nombre de fragments (sous-schémas), ce qui peut détériorer les performances du système. Cette problématique a été traitée par Bellatreche et Boukhalfa (2005), dans le contexte des entrepôts de données relationnels, et par Ma et Schewe (2003), dans le contexte des bases de données XML. Les solutions proposées permettent de sélectionner un schéma de fragmentation optimal en fonction du nombre de fragments afin d'optimiser les performance du système. Nous allons adapter ces travaux aux entrepôts de données XML.

D'autres perspectives se situent dans le cadre de la conception d'un système d'entrepôt de données XML distribué. En effet, la fragmentation constitue une première étape dans un processus de distribution de données. Il sera donc nécessaire d'identifier l'architecture de répartition des fragments XML : architecture de base de données répartie classique (Gertz et Bremer, 2003), grille de données (Wehrle et al., 2005) ou réseau pair-à-pair (Bonifati et al., 2006). Ce choix devra se baser sur l'évaluation des performances des requêtes sur ces architectures.

Références

Andrade, A., G. Ruberg, F. A. Baião, V. P. Braganholo, et M. Mattoso (2005). PartiX : processing XQuery Queries over Fragmented XML Repositories. Technical report, Computer Science Department, IM Federal University of Rio de Janeiro, Brazil.

Andrade, A., G. Ruberg, F. A. Baião, V. P. Braganholo, et M. Mattoso (2006). Efficiently Processing XML Queries over Fragmented Repositories with PartiX. In *Current Trends in Database Technology, EDBT 2006 Workshops PhD, DataX, IIDB, IIHA, ICSNW, QLQP, PIM, PaRMA, and Reactivity on the Web, Munich, Germany*, Volume 4254 of *Lecture Notes in Computer Science*, pp. 150–163. Springer.

Bellatreche, L. (2000). *Utilisation des Vues Matérialisées, des Index et de la Fragmentation dans la Conception logique et Physique d'un Entrepôt de Données*. Ph. D. thesis, Université de Clermont-Ferrand II.

Bellatreche, L. (2003). Techniques d'optimisation des requêtes dans les data warehouses. In *6th International Symposium on Programming and Systems (ISPS 03), Alger, Algérie*, pp. 81–98.

Bellatreche, L. et K. Boukhalfa (2005). An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse. In *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Copenhagen, Denmark*, Volume 3589 of *Lecture Notes in Computer Science*, pp. 115–125. Springer.

Beyer, K. S., D. D. Chamberlin, L. S. Colby, F. Ozcan, H. Pirahesh, et Y. Xu (2005). Extending XQuery for Analytics. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 05), Baltimore, Maryland*, pp. 503–514. ACM.

Bonifati, A., A. Cuzzocrea, et B. Zinno (2006). Fragmenting XML Documents via Structural Constraints. In *10th East European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece, Local Proceedings*, pp. 17–29.

Bonifati, A., U. Matrangolo, A. Cuzzocrea, et M. Jain (2004). XPath lookup queries in P2P networks. In *6th ACM CIKM International Workshop on Web Information and Data Management (WIDM 04), Washington, USA*, pp. 48–55. ACM.

Bose, S. et L. Fegaras (2005). XFrag : A query Processing Framework for Fragmented XML Data. In *8th International Workshop on the Web and Databases (WebDB 05), Baltimore, Maryland*, pp. 97–102.

Boukraa, D., R. BenMessaoud, et O. Boussaïd (2006). Proposition d'un Modèle physique pour les entrepôts XML. In *Atelier Systèmes Décisionnels (ASD 06), 9th Maghrebian Conference on Information Technologies (MCSEAI'06), Agadir, Maroc*.

Boussaïd, O., R. BenMessaoud, R. Choquet, et S. Anthoard (2006). X-Warehousing : an XML-Based Approach for Warehousing Complex Data. In *10th East-European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece*, Volume 4152 of *Lecture Notes in Computer Science*, pp. 39–54. Springer.

Bremer, J.-M. et M. Gertz (2003). On Distributing XML Repositories. In *International Workshop on Web and Databases (WebDB 03), San Diego, California*, pp. 73–78.

Datta, A., K. Ramamritham, et H. M. Thomas (1999). Curio : A Novel Solution for Efficient Storage and Indexing in Data Warehouses. In *25th International Conference on Very Large Data Bases (VLDB 99), Edinburgh, UK*, pp. 730–733. Morgan Kaufmann.

Gertz, M. et J.-M. Bremer (2003). Distributed XML Repositories : Top-down Design and Transparent Query Processing. Technical report, Departement of Computer Science, University of California, USA.

Hümmer, W., A. Bauer, et G. Harde (2003). XCube : XML for data warehouses. In *6th International Workshop on Data Warehousing and OLAP (DOLAP 03), New Orleans, USA*, pp. 33–40. ACM.

Ma, H. et K.-D. Schewe (2003). Fragmentation of XML Documents. In *XVIII Simpósio Brasileiro de Bancos de Dados, Manaus, Amazonas, Brasil*, pp. 200–214. UFAM.

Mahboubi, H. et J. Darmont (2006). Benchmarking XML data warehouses. In *Atelier Systèmes Décisionnels (ASD 06), 9th Maghrebian Conference on Information Technologies (MCSEAI 06), Agadir, Maroc*.

Meier, W. (2002). eXist : An Open Source Native XML Database. In *Web, Web-Services, and Database Systems, NODe 2002 Web and Database-Related Workshops, Erfurt,Germany*, Volume 2593 of *Lecture Notes in Computer Science*, pp. 169–183. Springer.

Noaman, A. Y. et K. Barker (1999). A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse. In *The 1999 ACM CIKM International Conference on Information and Knowledge Management (CIKM 99), Kansas City, USA*, pp. 154–161. ACM.

Özsu, M. T. et P. Valduriez (1999). *Principles of Distributed Database Systems, Second Edition*. Prentice-Hall.

Paparizos, S., Y. Wu, L. V. S. Lakshmanan, et H. V. Jagadish (2004). Tree Logical classes for effecient evaluation of XQuery. In *SIGMOD International Conference on Management of Data (SIGMOD 04), Paris, France*, pp. 71–82. ACM.

Park, B.-K., H. Han, et I.-Y. Song (2005). XML-OLAP : A Multidimensional Analysis Framework for XML Warehouses. In *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05)*, Volume 3589 of *Lecture Notes in Computer Science*, pp. 32–42. Springer.

Pokorný, J. (2002). XML Data Warehouse : Modelling and Querying. In *5th International Baltic Conference (BalticDB&IS 06), Tallin, Estonia*, pp. 267–280. Institute of Cybernetics at Tallin Technical University.

Rusu, L. I., J. W. Rahayu, et D. Taniar (2005). A Methodology for Building XML Data Warehouse. *International Journal of Data Warehousing and Mining, 1(2)*, 67–92.

W3C (2007). XQuery 1.0 and XPath 2.0 Functions and Operators. W3C Recommendation. http ://www.w3.org/TR/xpath-functions/.

Wehrle, P., M. Miquel, et A. Tchounikine (2005). A Model for Distributing and Querying a Data Warehouse on a Computing Grid. In *11th International Conference on Parallel and Distributed Systems (ICPADS 05), Fuduoka, Japani*, pp. 203–209. IEEE Computer Society.

Wu, M.-C. et A. P. Buchmann (1997). Research Issues in Data Warehousing. In *Datenbanksysteme in Buro, Technik und Wissenschaft*, pp. 61–82.

X-Hive-Corporation (2007). The fastest and most scalable XML database powered by open standards. http ://www.x-hive.com/products/db/.

Summary

XML data warehouses form an in interesting basis for decision-support applications that exploit heterogeneous data from multiple sources. However, XML-native database systems currently bear limited performances in terms of response time and data volume and it is necessary to search for ways to optimize them. In this paper, we propose to adapt derived horizontal fragmentation (as defined in relational data warehouses) for XML warehouses, in order to address these two issues. Our experiments show that this approach helps reduce the response time of analytic queries expressed in XQuery significantly.

Keywords: XML data warehouses, performance, fragmentation.