Modélisation et interrogation de données XML multidimensionnelles

Serge Boucher*, Boris Verhaegen*, Esteban Zimányi*

* Université Libre de Bruxelles (U.L.B.)
Service CoDE - CP165/15 - 1050 Bruxelles - Belgique
prénom.nom@ulb.ac.be

Résumé. XML étant devenu omniprésent et ses techniques de stockage et d'interrogation de plus en plus efficaces, le nombre de cas d'utilisations de ces technologies augmente tous les jours. Un sujet prometteur est l'intégration d'XML et des entrepôts de données, dans laquelle une base de données XML native stocke les données multidimensionnelles et exécute des requêtes OLAP écrites à l'aide du langage d'interrogation XML XQuery. Ce papier explore les questions qui peuvent survenir lors de l'implémentation d'un tel entrepôt de données XML.

1 Introduction

Les données qu'une entreprise rassemble d'années en années deviennent rapidement son atout le plus précieux. Les technologies des entrepôts de données aident à tirer le meilleur parti de ces données via le stockage et l'organisation des données opérationnelles d'une manière adaptée pour faciliter l'analyse et la décision. Un entrepôt de données est généralement modélisé de façon multidimensionnelle : l'ensemble des données est considéré comme un ensemble de *faits*. Ces faits contiennent des *mesures* qui peuvent être analysées suivant différentes *dimensions* ou perspectives. Chaque dimension peut être organisée en *hiérarchie* composée de plusieurs niveaux permettant d'analyser les mesures suivant différentes granularités. Un tel modèle multidimensionnel se prête à de nombreux types de requêtes qui peuvent avoir une valeur commerciale considérable. Leur traitement est appellé analyse multidimensionnelle ou *Online Analytical Processing (OLAP)*.

L'augmentation de l'échange de données entre applications a incité la création de standards tel que XML, aujourd'hui omniprésent. D'énormes quantités de données sont maintenant disponibles au format XML et les outils permettant d'utiliser ces données s'améliorent chaque jour. Plus particulièrement, les bases de données XML natives et le langage d'interrogation XQuery, sont aujourd'hui suffisamment avancés pour être utilisés dans un environnement de production. L'approche traditionnelle pour l'entreposage de données XML est de les convertir en données relationnelles. Cependant, une piste de recherche intéressante est de mettre en place un entrepôt de données utilisant uniquement les technologies XML : les données peuvent être modélisées en tant que documents XML stockés dans une base de données XML native et analysés à l'aide de requêtes XQuery.

La principale contribution de ce papier est une discussion sur les problèmes soulevés lors de l'implémentation d'un entrepôt de données utilisant uniquement des technologies XML.

2 Modélisation de données multidimensionnelles en XML

Dans cette section, nous présentons trois approches pour modéliser des données multidimensionnelles dans des documents XML. Pour illustrer, nous utiliserons tout au long de ce papier un entrepôt de données simple : les commandes (`Orders`) sont les faits, les dimensions sont le produit (`Product`) et le client (`Customer`) et les mesures sont la quantité d'articles (`Quantity`) et leur prix (`Price`). Les dimensions sont organisées suivant les hiérarchies suivantes : Produit-Catégorie-Famille (`Product-Category-Family`) et Client-Etat-Pays (`Customer-State-Country`). Ces dimensions sont strictes. Les dimensions de type graphe seront explorées dans une autre étude.

2.1 Modèle plat

Le modèle plat représente chaque fait par un élément XML unique, les mesures et hiérarchies de dimensions étant des sous-éléments de ce fait. Les hiérarchies de dimensions sont exprimées de manière plate de façon similaire à un schéma relationnel en étoile dénormalisé. Un exemple d'un fait représenté dans le modèle plat est présenté ci-dessous.

```
<order> <price>125,67</price>
        <quantity>3</quantity>
        <product_dimension> <product>Table</product>
                            <category>Kitchen</category>
                            <family>Furnitures</family>
        </product_dimension>
        <customer_dimension> ... </customer_dimension>
</order>
```

Chaque information à propos d'un fait et de ses dimensions est donc stockée dans le même élément XML. Comme nous le verrons dans la section suivante, cela permet une interrogation efficace car aucune jointure n'est nécessaire pour obtenir les différents niveaux hiérarchiques. D'un autre côté, le modèle plat implique de la redondance dans les données des dimensions et ne préserve pas la sémantique d'imbrication des niveaux des hiérarchies de dimensions.

2.2 Modèle hiérarchique

Nous proposons un modèle hiérarchique dans lequel la redondance des données est évitée en stockant les faits et les hiérarchies de dimensions dans des éléments XML différents. Un fait y est représenté par un élément XML contenant ses mesures et des références vers les niveaux les plus fins de ses hiérarchies de dimensions. Chaque hiérarchie de dimension est réprésentée dans un élément XML dans lequel l'imbrication entre les niveaux hiérarchiques est exprimée à l'aide d'une relation XML père-fils. Une feuille de cet arbre XML correspond au niveau le plus fin de la hiérarchie de dimension correspondante et est représentée par un identificateur comme présenté ci-dessous :

```
<family name="Furniture">
  <category name="Kitchen">
    <product name="Table" id="p98"/> ...
```

Le principal avantage de modéliser les dimensions de cette façon est que l'utilisation de la structure arborescente naturelle de XML rend la traversée des niveaux hiérarchiques via les langages d'interrogation XML comme XPath et XQuery très facile. De plus, l'imbrication des niveaux y est clairement exprimée et aucune redondance n'est introduite dans les données des dimensions.

2.3 Modèle XCube

XCube (Hümmer et al., 2003) est un autre formalisme pour réprésenter des donnes multidimensionneles dans des documents XML. Dans ce modèle, chaque fait et chaque niveau hiérarchique d'une dimension est représenté par un élément XML distinct. La différence principale entre notre modèle hiérarchique et XCube est que les liens entre les niveaux des hiérarchies ne sont plus exprimées par des relations père-fils mais par des identificateurs.

Comme dans notre modèle hiérarchique, XCube n'introduit pas de redondance dans les données des dimensions. Cependant, l'usage d'identificateurs entre les niveaux hiérarchiques au lieu des relations XML père-fils rend la navigation dans les hiérarchies plus compliquée.

3 Interrogation multidimensionnelle avec XQuery

Dans cette section, nous comparons l'efficacité de l'interrogation de ces modèles en montrant comment exprimer des requêtes OLAP typiques sur ces modèles en utilisant le langage XQuery. Nous utiliserons pour notre comparaison la requête OLAP suivante : *Pour chaque état de client et pour chaque catégorie de produit, donner le nombre de produits commandés.*

La traduction de notre requête de travail en XQuery 1.0 pour notre exemple d'entrepôt modélisé suivant le modèle plat est montrée ci dessous :

```
for $category in distinct-values(//category)
  for $state in distinct-values(//state)
    let $facts := //order[(//category eq $category) and (//state eq $state)]
    return <group> <category>{$category}</category>
                   <state>{$state}</state>
                   <sum>{sum($facts/quantity)}</sum> </group>
```

Pour répondre à cette requête à l'aide de XQuery 1.0, nous devons tout d'abord obtenir toutes les combinaisons de catégories et d'états en utilisant une double boucle `for` qui itère sur toutes les commandes pour trouver les valeurs distinctes des catégories et des états. Ensuite, nous devons obtenir toutes les commandes correspondantes à ces combinaisons de catégories et d'états pour enfin agréger les mesures.

Ce genre de requête souffre de deux problèmes. Premièrement, leur structure basée sur l'énumération des combinaisons de dimensions les rend peu commode à écrire et difficiles à comprendre. Deuxièmement, sans optimisation interne du moteur de requêtes XQuery, cette requête nécessite de traverser plusieurs fois l'entièreté du document XML. De plus, l'optimisation automatique de ce type de requête est une opération très complexe comme montré dans Beyer et al. (2005).

La requête correspondante traduite pour le modèle hiérarchique est très similaire à celle présentée ci-dessus. La principale différence est que les jointures entre les faits et les dimensions sont plus compliquées et plus coûteuses car celles-ci utilisent la comparaison générale XPath plutôt que la comparaison de valeurs. En ce qui concerne la requête écrite pour le modèle XCube, le principe reste également le même. La différence majeure est que du fait de la représentation des niveaux hiérachiques par des éléments XML différents, le nombre de jointures augmente significativement.

3.1 Clause de groupement pour XQuery

Les requêtes OLAP classiques posent des problèmes de lisibilité et de performances lorsqu'elles sont exprimées avec XQuery 1.0. La complexité en temps de ces requêtes augmente significativement avec le nombre de dimensions à analyser car ces requêtes requièrent une clause `for` additionnelle pour chaque dimension. De plus, commencer par déterminer toutes les combinaisons des dimensions avant d'agréger les faits n'est pas naturel.

Dans le monde relationnel, ces problèmes ont été résolus élégamment par l'introduction de la clause `GROUP BY`. L'extension de XQuery avec une clause de groupement a été proposée plusieurs fois dans la littérature (Beyer et al., 2005; Borkar et Carey, 2004). Récemment, le consortium W3C a publié une ébauche du futur XQuery 1.1 dans laquelle des fonctionnalités de groupement et de fenêtrage sont proposées. Une telle clause de groupement permet d'écrire ce type de requêtes d'une façon plus naturelle et facilite l'optimisation de leur traitement comme montré ci-dessous :

```
for $fact in //order
let $category := $fact//category, $state := $fact//state
group by $category, $state
return <group> ... </group>
```

L'avantage en lisibilité de cette nouvelle syntaxe est plus évident : elle est plus succinte et son but est immédiatement clair. De plus, il n'est plus nécessaire de calculer toutes les combinaisons d'états et de catégories et l'optimisation en est facilitée.

4 Expérimentation

Pour comparer nos modèles et nos requêtes, nous avons produit des ensembles de données de différentes tailles pour chacun des trois modèles. Nous avons égament implémenté une clause de groupement dans le moteur XQuery de la base de données XML native libre eXist[1], depuis la version 1.2. Pour chaque modèle, nous avons écrit huit requêtes avec une quantité de résultats attendus et un nombre de dimensions différents. La Fig. 1 montre le temps d'exécution de nos requêtes par rapport au nombre de résultats (groupes) sur une ensemble de données contenant 10 000 faits, pour chaque modèle et pour les requêtes utilisant ou non la clause de groupement.

Notre première observation est que les requêtes sur le modèle plat sont généralement plus rapides que celles sur le modèle hiérarchique ou le modèle XCube. Ce comportement est dû aux jointures nécessaires lors de l'exécution des requêtes sur les autres modèles à cause de la séparation entre les faits et les dimensions. Les requêtes sur le modèle XCube sont plus lentes que celle sur le modèle hiérarchique en raison du nombre de jointure nécessaire pour traverser ses hiérarchies.

Deuxièmement, nous remarquons que le temps d'exécution des requêtes sur le modèle plat sans clause de groupement augmente linéairement avec le nombre de résultats. Le temps de réponse pour les requêtes sur les autres modèles n'utilisant pas de clause de groupement augmente également avec le nombre de groupes mais présente un minimum local dû aux stratégies d'indexation d'eXist.

Comme attendu, le temps de réponse des requêtes utilisant la clause de groupement de XQuery 1.1 est dépendant uniquement du nombre de dimensions à analyser et de la taille

[1] exist.sourceforge.net/

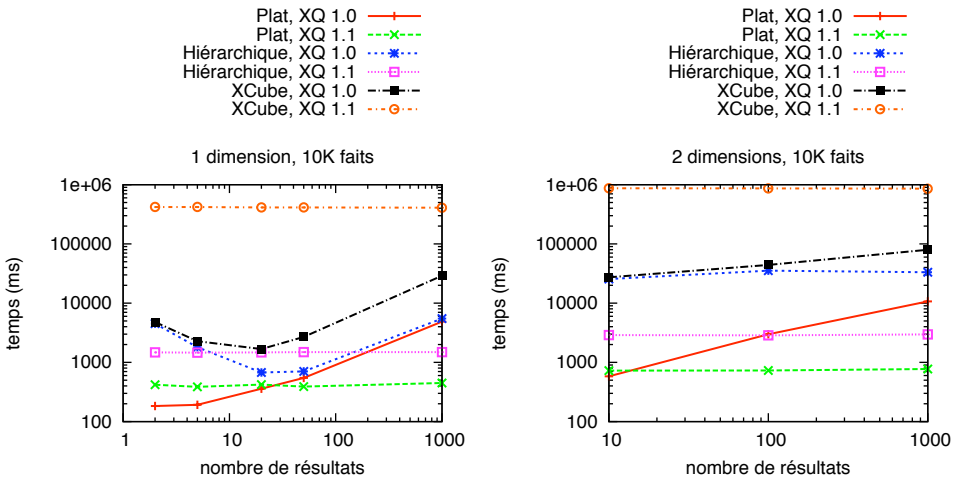

FIG. 1 – *Temps d'exécution par nombre de groupes, avec une et deux dimensions.*

de la base de données. De plus, le temps semble augmenter linéairement avec le nombre de dimensions et ces requêtes sont le plus souvent plus rapides que leur équivalente sans clause de groupement, excepté pour le modèle XCube.

5 Travaux connexes

Au meilleur de notre connaissance, ce papier est le premier à comparer les performances de requêtes OLAP sur différents modèles multidimensionnels XML. Beyer et al. (2005) ont proposé une clause de groupement mais n'ont considéré que le modèle plat pour leurs cas d'utilisations et leur évaluation de performance. Boussaid et al. (2006) ont proposé une variante du modèle plat qui préserve la sémantique des hiérarchies de dimensions, sans pour autant régler les problèmes de redondance de ce modèle.

A propos du problème spécifique de l'interrogation de données multidimensionnelles XML, Bordawekar et Lang (2005) ont identifié d'importantes différences entre les données relationnelles et XML en ce qui concerne l'analyse de données.

6 Conclusions

Nous avons analysé différentes approches pour représenter des données multidimensionnelles dans des documents XML. Nous avons proposé un modèle hiérarchique contenant plus de sémantique et moins de redondance qu'un modèle plat classique et nous avons comparé ces deux modèles avec le modèle XCube. Nous avons montré que les requêtes OLAP typiques sont toujours plus rapide sur un modèle plat par rapport aux autres modèles grâce à l'absence de jointures complexes. De manière similaire, nous avons montré que le modèle XCube engendre les requêtes les plus lentes à cause de la quantité de jointures nécessaires pour traverser les

hiérarchies de dimensions. Deuxièmement, nous avons discuté la manière dont des requêtes OLAP typiques peuvent être implémentées sur des données représentées à l'aide de ces modèles. Nous avons montré que les opérations de groupement classiques sont difficiles à écrire avec XQuery 1.0 et coûteuses en temps, mais que l'implémentation de la clause de groupement proposée dans XQuery 1.1 les rendait plus faciles à écrire et plus rapides. Notamment, les requêtes sur le modèle relationnel deviennent plus rapides que leur équivalent sans clause de groupement sur le modèle plat. Nous proposons donc notre modèle hiérarchique associé à une clause de groupement comme le meilleur compromis entre lisibilité, redondance et performances d'interrogation pour les entrepôts de données XML. Une analyse plus détaillée est disponible dans le rapport technique Boucher et al. (2009).

Le succès des technologies XML a été spectaculaire depuis leur création. Comme le montre ce papier, il reste encore une quantité substantielle de recherche avant de pouvoir mener de gros projets d'entrepôts de données natifs XML. Cependant, ce thème de recherche est prometteur : au-delà de l'avantage évident d'en finir avec de longues conversions XML-relationnel, le design épuré de XQuery et ses liens étroits avec l'écosystème XML en font un outil très utile pour l'exploitation optimale des données d'une organisation.

Références

Beyer, K., D. Chamberlin, L. Colby, F. Özcan, H. Pirahesh, et Y. Xu (2005). Extending XQuery for Analytics. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 503–514.

Bordawekar, R. et C. Lang (2005). Analytical Processing of XML Documents : Opportunities and Challenges. *ACM SIGMOD Record 34*(2), 27–32.

Borkar, V. et M. Carey (2004). Extending XQuery for Grouping, Duplicate Elimination, and Outer Joins. *Proceedings of the XML 2004 Conference*.

Boucher, S., B. Verhaegen, et E. Zimányi (2009). XML Multidimensional Modelling and Querying. Technical report, Université Libre de Bruxelles (U.L.B.) - Code Departement. Available at http://arxiv.org/abs/0912.1110.

Boussaid, O., R. Messaoud, R. Choquet, et S. Anthoard (2006). X-Warehousing : an XML-Based Approach for Warehousing Complex Data. *Proceedings of the 10th East-European Conference on Advances in Databases and Information Systems*, 39–54.

Hümmer, W., A. Bauer, et G. Harde (2003). XCube : XML for Data Warehouses. *Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP*, 33–40.

Summary

As XML becomes ubiquitous and XML storage and processing becomes more efficient, the range of use cases for these technologies widens daily. One promising area is the integration of XML and data warehouses, where an XML-native database stores multidimensional data and processes OLAP queries written in the XQuery interrogation language. This paper explores issues arising in the implementation of such a data warehouse.