Modèles d'arbre pour XOLAP

Marouane Hachicha, Jérôme Darmont

Université de Lyon (ERIC Lyon 2)
5 Avenue Pierre Mendès-France
69676 Bron Cedex
Courriel : prénom.nom@univ-lyon2.fr

Résumé. Avec l'avènement de XML comme standard de représentation de données décisionnelles, les entrepôts de données XML trouvent leur place dans le développement de solutions décisionnelles. Dans ce contexte, il devient nécessaire de permettre des analyses OLAP sur des cubes de données XML. Afin de contribuer à ces recherches, de définir un cadre formel et de permettre l'optimisation indispensable des requêtes décisionnelles exprimées en XQuery, nous travaillons à définir une algèbre XML-OLAP (ou XOLAP). En premier lieu, nous avons exprimé avec l'algèbre XML TAX (*Tree Algebra for XML*) les opérateurs OLAP usuels. Il s'agit maintenant de prendre en compte les structures complexes permises par XML. Comme premier pas, nous proposons dans cet article un opérateur *rollup* basé sur un modèle d'arbre qui prend en compte des données XML multidimensionnelles organisées en hiérarchies complexes.

Mots clés : XML, Entrepôts de données, XOLAP, Modèles d'arbre, Hiérarchies complexes, Algèbre, opérateur *rollup*

1 Introduction

La complexité des technologies décisionnelles telles que les entrepôts de données et l'analyse en ligne (OLAP, *On-Line Analytical Processing*) les rend peu attractives pour de nombreux utilisateurs potentiels. Dans ce contexte, une nette tendance à l'entreposage de données en ligne se dégage, avec notamment l'entreposage XML (Park et al., 2005; Zhang et al., 2005; Boussaïd et al., 2006). Le langage XML (*eXtensible Markup Language*) est en effet de plus en plus utilisé pour représenter des données décisionnelles (Beyer et al., 2005) et se montre particulièrement adapté pour modéliser des données dites complexes (Darmont et al., 2005) issues de sources hétérogènes et notamment du Web. Ainsi, plusieurs travaux visent à étendre le langage XQuery pour supporter des requêtes de type OLAP (Kay, 2006). Ces extensions doivent non seulement permettre d'effectuer des analyses OLAP classiques, mais aussi de prendre en compte dans l'analyse en ligne les spécificités des données XML, comme par exemple les structures hiérarchiques complexes telles que les *ragged hierarchies* (Beyer et al., 2005), qui seraient très difficiles à gérer dans un environnement relationnel.

Nous travaillons dans ce contexte à concevoir une algèbre XML-OLAP -ou XOLAP (Wang et al., 2005) permettant d'exécuter des requêtes OLAP sur des données natives XML. Notre but général sera de définir un cadre formel actuellement inexistant dans le contexte XOLAP.

Dans une première étape, nous avons exprimé les opérateurs OLAP usuels (structurels, ensemblistes et liés à la granularité) avec l'algèbre XML TAX (Jagadish et al., 2001) sur des arbres de données XML exprimés sous forme de hiérarchies simples (Hachicha et al., 2008). Notre objectif est d'étendre, dans une deuxième étape, ces opérateurs afin qu'ils fonctionnent sur des données organisées selon des hiérarchies complexes[1]. Dans cet article, nous exploitons le modèle d'arbre pour proposer un opérateur *rollup* capable de traiter des données XML multidimensionnelles organisées en hiérarchies complexes. Michiels et al. (2007) insistent d'ailleurs sur l'efficacité d'utiliser des modèles d'arbre dans les langages de requêtes en général et dans les algèbres XML en particulier.

Le reste de cet article est par conséquent organisé comme suit. Un état de l'art sur les modèles d'arbre employés dans les algèbres d'arbre XML est présenté dans la Section 2. Nous nous limitons à ces modèles comme nous nous situons dans un contexte algébrique. La Section 3 repose sur les définitions formelles des hiérarchies complexes, les arbres de données XML et les modèles d'arbre. Nous exposons notre approche dans la Section 4 avant de conclure et de présenter les perspectives de ce travail dans la Section 5.

2 État de l'art : Modèles d'arbre dans les algèbres d'arbres XML

L'objectif d'une algèbre XML est de présenter un jeu d'opérateurs permettant de manipuler des données XML, souvent modélisées en collection d'arbres étiquetés et ordonnés. Représenter les données XML en arbres facilite la compréhension et la conception des opérateurs d'une algèbre. Le résultat d'une requête relationnelle est toujours structuré en format tabulaire. De même, comme nous représentons les données XML avec des arbres, le format de sortie d'un opérateur est un arbre. Dans TAX (Jagadish et al., 2001), le rôle du modèle d'arbre est de spécifier le format et le contenu de l'arbre en sortie d'un opérateur.

Les premières algèbres XML sont apparues en 1999 (Beech et al., 1999) en parallèle avec les efforts visant à définir un langage de requêtes pour XML (Clark et DeRose, 1999; Ishikawa et al., 1999), et surtout avant la première spécification de XQuery (Chamberlin et al., 2001), qui est désormais le langage de requêtes standard pour XML.

Considérons comme premier exemple de modèle d'arbre dans un contexte algébrique XML celui de TAX, qui est l'un des plus simples. La principale caractéristique de ce modèle est de conserver la structure (l'ordre des nœuds et leur position dans l'arbre, ainsi que les relations entre eux : parent-enfant ou ancêtre-descendant) de la collection d'arbres de données XML ordonnée d'origine, et satisfait une formule associée au modèle. Une formule est, par exemple, un prédicat de la forme (*Where* Titre du livre = "XML").

La Figure 1 (b) illustre un exemple de modèle d'arbre de TAX sélectionnant des livres par leur titre, leurs auteurs et leur résumé depuis l'arbre de données XML de la Figure 1 (a). Nous remarquons que seul le deuxième livre (titre = "A dummy for a computer") a été sélectionné, dans le résultat de la Figure 1 (c), puisque le premier (titre = "Maktub") ne contient pas de résumé.

Le modèle d'arbre de TAX a été généralisé en modèle d'arbre généralisé (GTP : *Generalized Tree Pattern*) par l'attribution du statut obligatoire ou facultatif aux différentes arêtes (Chen

[1] Les hiérarchies complexes sont formellement définies dans la Section 3.1

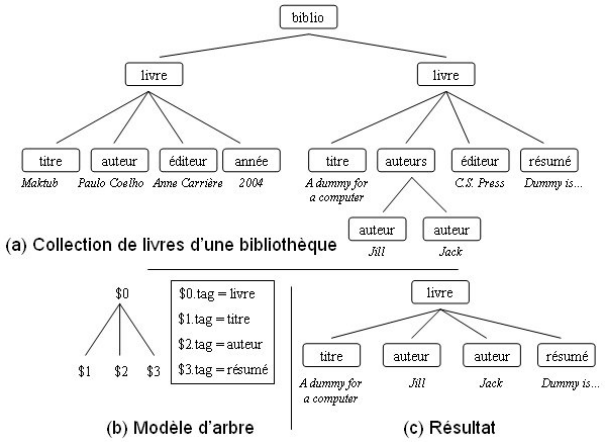

FIG. 1 – *Arbre de données XML, modèle d'arbre et résultat*

et al., 2003). Par exemple, si nous attribuons le statut facultatif à l'arête reliant le nœud *résumé* avec son nœud-parent, le livre (titre = "Maktub") apparaît aussi dans le résultat.

Les modèles d'arbre de TAX ont également été étendus par des annotations (APT : *Annotated Pattern Trees*) (Paparizos et al., 2004). Quatre annotations sous la forme de signes sont associées aux différentes arêtes du modèle :

- + permet une ou plusieurs correspondances (*matching*) ;
- avec -, ne seule correspondance est autorisée ;
- * permet de zéro à un nombre illimité de correspondances ;
- avec ?, zéro ou une seule correspondance est autorisée.

Finalement, comme ces trois modèles d'arbre (modèle de TAX, GTP et APT) préservent l'ordre des nœuds en sortie, quel que soit l'ordre des nœuds dans le modèle, les modèles d'arbre annotés (APT) ont été aussi enrichis avec un paramètre *ord* permettant de spécifier l'ordre en sortie d'une opération de sélection ou de jointure (Paparizos et Jagadish, 2005).

2.1 Synthèse

Nous synthétisons, dans la Table 1, les caractéristiques des modèles d'arbre étudiés selon trois critères : les options liées à la correspondance (*matching*) du modèle avec l'arbre de données en entrée, la possibilité de modifier l'ordre des nœuds en sortie de requête et la flexibilité du modèle par rapport aux niveaux hiérarchiques, absente pour tous les modèles.

	Options	**Ordre**	**Hiérarchies**
TAX	Non	Non	Non
GTP, APT	Oui	Non	Non
Extension de APT avec *ord*	Oui	Oui	Non

TAB. 1 – *Comparaison des modèles d'arbre*

3 Définitions et notations

3.1 Hiérarchies complexes

Entrepôt de données. Un entrepôt de données E modélisé en flocon de neige (c'est-à-dire avec des dimensions hiérarchisées) est défini par un triplet $E = (\mathbb{F}, \mathbb{D}, *)$ où :
- $\mathbb{F} = \{F_i\}_{i=1,f}$ est l'ensemble des faits (avec $|\mathbb{F}| = f$) ;
- $\mathbb{D} = \{D_i\}_{i=1,d}$ est l'ensemble des dimensions (avec $|\mathbb{D}| = d$) ;
- $* : \mathbb{F} \rightarrow \mathbb{D}$ est une fonction qui associe chaque fait à l'ensemble des dimensions qui le décrivent (Teste, 2009).

Fait. $\forall i \in [1,f]$, un fait F_i est défini par un n-uplet $F_i = (\Delta_i, \Lambda_i)$ où :
- $\Delta_i = \{\delta_{ij}\}_{j=1,d}$ est l'ensemble de références aux dimensions $\mathbb{D}$ résultant de la fonction $*$ (notons que $|\Delta_i| = d$) ;
- $\Lambda_i = \{\mu_{ij}\}_{j=1,m}$ est l'ensemble des mesures qui caractérisent F_i (avec $|\Lambda| = m$).

Dimension et hiérarchie. $\forall i \in [1,d]$, une dimension D_i est définie par une hiérarchie constituée de n_i niveaux : $D_i = \{H_{ij}\}_{j=1,n_i}$. $\forall j \in [1,n_i]$, un niveau hiérarchique H_{ij} est défini par un n-uplet $H_{ij} = \{\alpha_{ijk}\}_{k=1,a}$ constitué de a attributs α_{ijk}. Les liens entre les niveaux hiérarchiques de la dimension D_i sont établis grâce à une fonction $rollup_i : D_i \rightarrow D_i$.

Hiérarchie complexe. Nous qualifions une hiérarchie de dimension D_i de complexe si elle est à la fois non stricte et non couvrante.

Hiérarchie non stricte. Une hiérarchie est non stricte (Torlone, 2003) ou à arcs multiples (Rizzi, 2007) lorsque la fonction $rollup_i$ retourne un résultat multivalué. En d'autres termes, d'un point de vue conceptuel, une hiérarchie est non stricte si l'association entre deux niveaux hiérarchiques est de type plusieurs-à-plusieurs et non de type un-à-plusieurs. Par exemple, dans une dimension décrivant des produits, un produit donné peut appartenir à plusieurs catégories et non une seule. De même, une association plusieurs-à-plusieurs entre faits et instances de dimension peut exister (Torlone, 2003). Par exemple, dans un entrepôt de données de ventes, un fait peut être associé à une combinaison d'offres promotionnelles plutôt qu'une seule. Formellement, c'est ici la fonction $*$ qui retourne un résultat multivalué.

Hiérarchie non couvrante. Une hiérarchie est non couvrante (Torlone, 2003) ou effilochée (*ragged*) (Rizzi, 2007) si la fonction $rollup_i$ permet de relier un niveau hiérarchique H_{ij} à un niveau $H_{ij'}$ en "sautant" un ou plusieurs niveaux intermédiaires, c'est-à-dire : $rollup_i(H_{ij}) = H_{ij'}$ et $\exists H_{ij''} / rollup_i(H_{ij''}) = H_{ij'}$. Ce cas arrive, par exemple, si dans une dimension décrivant des magasins, la hiérarchie magasin-ville-région permet qu'un magasin soit localisé dans une région donnée sans être associé à une ville (cas des magasins situés en zone rurale). De même, les faits peuvent être décrits à des niveaux de granularité hétérogènes. Par exemple, toujours dans notre entrepôt de données de ventes, le volume de vente peut être connu au niveau des magasins dans une région du monde, mais seulement à un niveau plus agrégé (comme le pays) dans d'autres régions. Cela signifie que, $\forall i \in [1,d]$ la fonction $*$ peut associer un fait donné à un niveau hiérarchique H_{ij} de la dimension D_i tel que $j > 1$.

Notes :
- La notion de hiérarchie effilochée (*ragged hierarchy*) a différentes acceptions dans la littérature. Par exemple, Beyer et al. (2005) la définissent comme une hiérachie qui est à la fois non stricte et non couvrante alors que Rizzi (2007) la définit comme uniquement non couvrante. C'est pourquoi nous lui préférons et définissons les nouveaux termes de hiérarchie complexe.
- La mise en œuvre de hiérarchies complexes pose d'importants problèmes d'agrégation (*summarizability*) (Mazón et al., 2009). Toutefois, leur prise en compte dans un environnement XOLAP est pertinente (les cas réels existent) et nécessaire. Des travaux qui se consacrent à la normalisation de modèles conceptuels présentant des problèmes d'agrégation (Mazón et al., 2008) peuvent d'ailleurs être exploités en ce sens.

3.2 Arbres de données et modèles d'arbre

Arbre de données. Un arbre de données t est un triplet (r, N, E) où N est un ensemble de nœuds, $r \in N$ est la racine de t et E est un ensemble d'arêtes reliant des couples de nœuds.

Modèle d'arbre. Un modèle d'arbre est un arbre am = (r, N, E) dont l'ensemble d'arêtes appartient à celui de la collection d'arbres de données $C = \mathrm{U}_{i=1,n} t_i$ / $t_i = (r_i, N_i, E_i)$ modélisant les données XML.

4 Modèle d'arbre pour XOLAP

4.1 Exemple de hiérarchie complexe

Soit la Figure 2 (a) où sont représentés des faits de vente de livres décrits par leur titre, les catégories auxquelles ils appartiennent et les prix de vente. Les catégories de tous les livres sont représentées par une hiérachie complexe dans la Figure 2 (b). La catégorie la plus générale est représentée par `C1[valeur]`, celle du niveau immédiatement inférieur par `C2[valeur]` et ainsi de suite. Dans cet exemple, nous ne disposons que de trois niveaux, du plus général (C1) au plus détaillé (C3).

Une catégorie regroupe plusieurs livres et un livre est décrit par plusieurs catégories, ce qui fait de cette hiérarchie une hiérarchie non stricte (Section 3.1). De plus, les livres de titres "SQL" et "Manag. S.I" sont décrits par des hiérarchies de catégories complètes (C3→C2→C1). Par contre, le livre de titre "PHP 5" est décrit par une hiérarchie de catégories incomplète (C3 [SQL]→C1 [Logiciels]). D'un autre côté, en comparant les livres de titres "SQL" et "PHP 5", ils sont décrits par la même sous-catégorie C3 [SQL] mais le niveau C2 [BD] est absent pour "PHP 5". La hiérarchie est donc une hiérarchie non couvrante (Section 3.1). La hiérarchie de la Figure 2 (b) est bien une hiérarchie complexe alors comme elle est non couvrante et non stricte à la fois.

4.2 Problématique

Principalement, les opérations OLAP qui seront intéréssantes sur les données de la Figure 2 (a) sont le forage vers le haut (*rollup*) et vers le bas (*drill-down*) ou aussi construction de

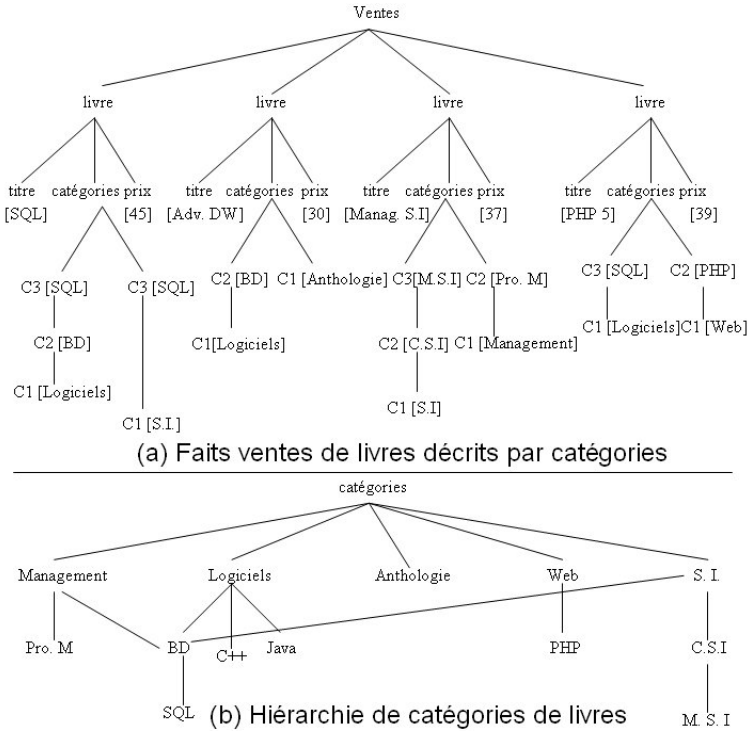

FIG. 2 – *Faits ventes de livres décrits par catégories organisées en hiérarchie complexe*

cube. Dans un travail précédent, nous avons exprimé le forage vers le haut (*rollup*) avec une combinaison d'opérateurs TAX de sélection, de groupement, de jointure, d'agrégation et de mise à jour des nœuds (Hachicha et al., 2008). Cela fonctionne pour une hiérarchie stricte.

Le problème que nous rencontrons avec les hiérarchies complexes est qu'au moment de l'agrégation des données, nous nous trouvons avec des faits décrits à différents niveaux de granularité. Il est donc difficile d'appliquer toutes les opérations, moyenne et comptage surtout, possibles sur les mesures. Un autre problème qui se pose est que certaines données risquent de ne pas être prises en compte à cause des niveaux de hiérarchie manquants, comme le cas du livre de titre "PHP 5" décrit par une hiérachie de catégorie incomplète (C3→C1).

4.3 Proposition d'un opérateur *rollup* basé sur les modèles d'arbre

4.3.1 Principe

Le choix de l'extension d'un modèle d'arbre au sein d'un ou plusieurs opérateurs TAX éléments de forage (sélection, groupement, jointure, agrégation, ajout et suppression des nœuds) reste une bonne solution mais pas générale car plusieurs choix d'opérateurs sont possibles. Par exemple, nous pouvons sélectionner les données dont nous avons besoin selon un nouveau modèle que nous proposons comme première solution, ou aussi emlpoyer la jointure pour adapter son modèle d'arbre avec le format des hiérarchies complexes comme une autre solution. De

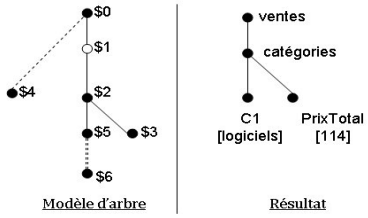

FIG. 3 – *Modèle d'arbre et résultat*

plus, nous voyons bien dans la table 1 que le modèle d'arbre de TAX et ses extensions présentés ne tiennent pas compte des hiérarchies, que nous avons traité dans un modèle à part dans (Hachicha et al., 2008).

Au lieu d'étendre un des modèles d'arbre utilisé dans un des opérateurs du forage vers le haut (*rollup*) alors, nous proposons un opérateur *rollup*, basé sur un modèle d'arbre et associé à un algorithme, permettant d'agréger les données XML multidimensionnelles représentées dans des hiérarchies complexes.

4.3.2 Modèle d'arbre

Dans le modèle d'arbre de la Figure 3 : $0 modélise la racine du document des faits. $1 modélise un fait décrit par des dimensions et des mesures. Ce nœud est de fond blanc, contrairement aux autres nœuds, car il peut ne pas apparaître dans le résultat. Son apparition dépend de l'opérateur. $2 modélise la racine de la hiérarchie complexe à manier. $5 modélise l'élément de la hiérarchie décrivant le fait. $6 représente un ancêtre quelconque de $5, il est utile pour fouiller dans la hiérarchie complexe et exploiter ses différents niveaux (le niveau le plus général est C1). $6 est relié à son nœud parent avec une arête discontinue comme il n'apparait pas dans le résultat final. $3 représente l'agrégat calculé, pour chaque fait $2, à partir de ses mesures. $4 compter les faits traités (utile dans des opérations de type comptage ou aussi moyenne). $4 est aussi relié à son nœud parent avec une arête discontinue pour la même raison que $6.

4.3.3 Algorithme

L'algorithme que nous proposons, basé sur le modèle d'arbre de la Figure 4.3.2, est disponible dans la Figure 4. Nous appliquons cet algorithme sur les données de la Figure 2 (a) dans la Section 4.3.4. Notons que cet algorithme peut prendre en compte un nombre infini de hiérarchies complexes hétérogènes. En effet, il ne passe d'un niveau hiérarchique à un niveau suivant ($0 est un niveau, $1 est le niveau suivant, etc.) qu'après le traitement de tous les éléments du niveau en cours.

4.3.4 Exemple

Soit la requête Q : calculer le total des ventes des livres de catégorie “Logiciels”.

En assurant la correspondance entre le modèle d'arbre de la Figure 3 et les données de la Figure 2 (a), le nœud $0 prend la valeur “ventes” et le nœud $1 prend la valeur “livre”.

```
Algorithme : Agrégation avec un modèle d'arbre

// Initialisation
 $4 = 0
 $3 = fonction d'agrégation = 0
chaine = ``élément de l'agrégation''

Début

Pour tout  $1 Faire // Test1
   Tant que ( $2 / $5 == True) // (Tant que  $2 a des enfants) // Test 2
        Si  $5 == chaine alors
           {
             $4++
             $3 + = la mesure correspondante
            exit //retour à Test 1
           }
          Sinon
           {
            Si  $6 == chaine alors
           {
             $4++
             $3 + = la mesure correspondante
            exit // retour à Test1
           }
            Sinon exit //retour à Test 2
            Finsi
        Finsi
   Fin Tant que
Fin Pour

Fin
```

FIG. 4 – *Algorithme proposé*

Pour chaque nœud $2 "catégories" de chaque livre, nous testons si la valeur correspondante à la première catégorie du livre ($5) est égale à la valeur recherchée "Logiciels". Si oui, nous passons au livre suivant ($1 suivant). Sinon, on continue la recherche dans ses ancêtres ($6) sous condition qu'ils soient des "Logiciels". Au cas où ce livre de type "Logiciels" est trouvé, $4 est incrémenté de 1 et la fonction $3 prend comme première valeur le prix de vente de ce livre. Dans le cas contraire, nous continuons la recherche de la même façon pour toutes les sous-catégories. Au cas où le livre n'est pas de la catégorie recherchée (le parcours de toutes les catégories est assuré sans résultat), nous passons au livre ($1) suivant. À la fin du parcours de tout l'arbre de données, l'agrégat est calculé comme le montre le résultat de la Figure 3.

5 Conclusion et persperctives

Dans cet article, nous avons progressé vers la définition d'un cadre formel dans le contexte XOLAP. Notre contribution est double dans ce travail. En premier lieu, nous avons formalisé la notion de hiérarchies complexes. Nous avons proposé, en deuxième lieu, un premier opérateur permettant d'agréger des données multidimensionnelles décrites par ce type de hiérarchies en étendant la notion du modèle d'arbre.

Les perspectives que présente ce travail sont multiples. Il faut en premier lieu prendre en compte les importants problèmes d'agrégation (*summarizability*). Nos futurs opérateurs seront aussi basés sur des modèles d'arbre adaptés aux hiérarchies complexes. Enfin, une implémentation de nos travaux est indispensable. Les travaux d'optimisation des requêtes XQuery et des expressions de chemin XPath en parallèle aux travaux d'optimisation des modèles d'arbre

utilisés dans XML seront les éléments de départ de l'implémentation de notre prototype. Ces perspectives s'inscrivent, en effet, dans nos perspectives générales : soutenir les efforts visant à étendre le langage XQuery pour permettre des analyses OLAP, notamment avec des opérateurs spécifiques à XML, d'un côté ; et permettre l'optimisation de requêtes OLAP exprimées en XQuery, d'un autre côté.

Références

Beech, D., A. Malhotra, et M. Rys (1999). A formal data model and algebra for XML. Technical report.

Beyer, K. S., D. D. Chamberlin, L. S. Colby, F. Özcan, H. Pirahesh, et Y. Xu (2005). Extending XQuery for Analytics. In *ACM SIGMOD 24th International Conference on Management of Data (SIGMOD 05), Baltimore, USA*, pp. 503–514. ACM.

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 *LNCS*, pp. 39–54. Springer.

Chamberlin, D. D., D. Florescu, J. Robie, J. Siméon, et M. Stefanescu (2001). XQuery 1.0 : An XML Query Language. http ://www.w3.org/TR/xquery 2001. World Wide Web Consortium (W3C).

Chen, Z., H. V. Jagadish, L. V. S. Lakshmanan, et S. Paparizos (2003). From Tree Patterns to Generalized Tree Patterns : On Efficient Evaluation of XQuery. In *29th International Conference on Very Large Data Bases (VLDB 03), Berlin, Germany*, pp. 237–248.

Clark, J. et S. DeRose (1999). XML Path Language (XPath) Version 1.0. http ://www.w3.org/TR/xpath. World Wide Web Consortium (W3C).

Darmont, J., O. Boussaïd, J.-C. Ralaivao, et K. Aouiche (2005). An Architecture Framework for Complex Data Warehouses. In *7th International Conference on Enterprise Information Systems (ICEIS 05), Miami, USA*, pp. 370–373.

Hachicha, M., H. Mahboubi, et J. Darmont (2008). Expressing OLAP operators with the TAX XML algebra. In *3rd International Workshop on Database Technologies for Handling XML Information on the Web (DataX-EDBT 08), Nantes, France*.

Ishikawa, H., K. Kubota, Y. Kanemasa, et Y. Noguchi (1999). The Design of a Query Language for XML Data. In *10th International DEXA Workshop on Database and Expert Systems Applications (DEXA 99), Florence, Italy*.

Jagadish, H. V., L. V. S. Lakshmanan, D. Srivastava, et K. Thompson (2001). TAX : A Tree Algebra for XML. In *8th International Workshop on Database Programming Languages (DBPL 01), Frascati, Italy*, Volume 2397 of *LNCS*, pp. 149–164. Springer.

Kay, M. (2006). Positional Grouping in XQuery. In *3rd International Workshop on XQuery Implementation, Experience and Perspectives (XIME-P 06), Chicago, USA*.

Mazón, J.-N., J. Lechtenbörger, et J. Trujillo (2008). Solving summarizability problems in fact-dimension relationships for multidimensional models. In *ACM 11th International Workshop on Data Warehousing and OLAP (DOLAP 08), Napa Valley, California, USA*, pp. 57–64.

Mazón, J.-N., J. Lechtenbörger, et J. Trujillo (2009). A survey on summarizability issues in multidimensional modeling. *Data & Knowledge Engineering 68*(12), 1452–1469.

Michiels, P., G. A. Mihaila, et J. Siméon (2007). Put a Tree Pattern in Your Algebra. In *23rd International Conference on Data Engineering (ICDE 07), Istanbul, Turkey*, pp. 246–255. IEEE.

Paparizos, S. et H. V. Jagadish (2005). Pattern Tree Algebras : Sets or Sequences ? In *31st International Conference on Very Large Data Bases (VLDB 05), Trondheim, Norway*, pp. 349–360. ACM.

Paparizos, S., Y. Wu, L. V. S. Lakshmanan, et H. V. Jagadish (2004). Tree Logical Classes for Efficient Evaluation of XQuery. In *SIGMOD 23rd 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 *LNCS*, pp. 32–42. Springer.

Rizzi, S. (2007). Conceptual Modeling Solutions for the Data Warehouse. In R. Wrembel et E. Christian Koncilia (Eds.), *Data Warehouses and OLAP : Concepts, Architectures and Solutions*, pp. 1–26. IRM Press, Hershey, PA, USA.

Teste, O. (2009). *Modélisation et manipulation des systèmes OLAP : de l'intégration des documents à l'usager*. Mémoire d'HDR, Université Paul Sabatier - Toulouse III, France.

Torlone, R. (2003). Conceptual Multidimensional Models. In E. Maurizio Rafanelli (Ed.), *Multidimensional Databases : Problems and Solutions*, pp. 69–90. IDEA Group Publishing, Hershey, PA, USA.

Wang, H., J. Li, Z. He, et H. Gao (2005). OLAP for XML Data. In *1st International Conference on Computer and Information Technology (CIT 05), Shanghai, China*, pp. 233–237. IEEE Computer Society.

Zhang, J., W. Wang, H. Liu, et S. Zhang (2005). X-Warehouse : Building Query Pattern-driven Data. In *14th International Conference on World Wide Web (WWW 05), Chiba, Japan*, pp. 896–897. ACM.

Summary

With the growth of XML as a standard for representing business data, XML data warehousing appears as a suitable solution for decision-support applications. In this context, it is necessary to allow OLAP analyses on XML data cubes. Thus, XQuery extensions are needed. To define a formal framework and allow much-needed performance optimizations on analytical queries expressed in XQuery, defining an algebra is desirable. However, XML-OLAP (XOLAP) algebras from the literature still largely rely on the relational model and only feature a very small number of OLAP operators. Hence, we propose in this paper a *rollup* operator based on a pattern tree in order to handle multidimensional XML data expressed within complex hierarchies.