Vers une algèbre XML-OLAP : État de l'art

Marouane Hachicha, Hadj Mahboubi, Jérôme Darmont
Université de Lyon (ERIC Lyon 2)
5 avenue Pierre Mendès-France
69676 Bron Cedex
France
Courriel : prenom.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 basées sur le Web. Dans ce contexte, il devient nécessaire de permettre des analyses OLAP sur des cubes de données XML. Pour cela, des extensions du langage XQuery sont nécessaires. 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). L'objectif de cet article est, en amont de ces recherches, de présenter et d'analyser l'état de l'art en la matière, et plus précisément les algèbres OLAP, les algèbres XML et les algèbres tendant vers XOLAP. Nous présentons également brièvement notre propre proposition d'algèbre XOLAP.
Mots-clés : Entrepôts de données, OLAP, XML, Requêtes XQuery décisionnelles, Algèbres.

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. C'est pourquoi les éditeurs de solutions décisionnelles commencent à développer des interfaces web simples et conviviales (Lawton, 2006). Dans ce contexte, une nouvelle 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 (groupement, agrégation, etc.) (Borkar et Carey, 2004; Beyer et al., 2005; 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 des hiérarchies multiples, imbriquées et incomplètes (*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 objectif pour développer un tel outil est triple : (1) définir un cadre formel actuellement inexistant dans le contexte XOLAP ; (2) soutenir les efforts visant à étendre le langage XQuery pour permettre des analyses OLAP, notamment avec des opérateurs spécifiques à XML ; (3) permettre l'optimisation de requêtes OLAP exprimées en XQuery. Les Systèmes de Gestion de Bases de Données (SGBD) natifs XML présentent en effet des limitations en terme de performance et, bien qu'en constant progrès sur ce point, bénéficieraient grandement d'une optimisation automatique des requêtes, et particulièrement des requêtes décisionnelles qui sont en général très coûteuses. En préalable à nos travaux, nous avons dressé un panorama aussi exhaustif que possible des recherches connexes de la littérature. L'objet de cet article est de présenter et d'analyser cet état de l'art en regard de nos objectifs ; ainsi que de présenter brièvement l'avancée actuelle de notre travail.

Le reste de cet article est par conséquent organisé comme suit. La Section 2 repose sur la définition des opérateurs OLAP que nous considérons et que nous illustrons par des exemples. La Section 3 détaille les algèbres développées pour l'OLAP. La Section 4 présente les algèbres d'interrogation de données XML. La Section 5 introduit les travaux, moins nombreux, qui visent à permettre des analyses OLAP sur des cubes XML. Finalement, nous concluons cet article, présentons l'avancée de nos propres recherches ainsi que leurs perspectives dans la Section 6.

2 Définitions préalables

Dans cette section, nous définissons les opérateurs OLAP classiques de base, classés dans trois familles. Chaque opérateur prend en entrée un cube OLAP et fournit un autre cube en sortie. Un cube OLAP représente dans ses cellules des faits à analyser (matérialisés par des mesures numériques) en fonction de dimensions (axes d'analyse) décrites par des attributs membres et susceptibles d'être hiérarchisées (par exemple, une dimension géographique ville, région, pays). Soit un entrepôt de données *Ventes* très simplifié, constitué de trois dimensions *Ville*, *Produit* et *Année*, ainsi que d'une table de faits *Vente*. Chacune des dimensions est caractérisée par deux modalités : *Ville*(Paris, Lyon), *Produit*(Ecran, Clavier) et *Année*(2004, 2005). La mesure étudiée est le *Prix* des produits vendus par ville et par année. Un exemple de cube OLAP modélisant ces ventes est disponible dans la Figure 1 (a).

2.1 Opérateurs liés à la structure

Rotation (rotate) : *Rotate(C, D)* permute les dimensions d'un cube *C* autour d'une d'entre elles (*D*). La *rotation* est une sorte de sélection de faces et non de membres ; elle déplace deux dimensions ou plus dans le cube sans modifier les valeurs des mesures correspondantes. Dans la Figure 1 (b), nous représentons une rotation du cube de la Figure 1 (a) selon l'axe *Produit* en appliquant ce principe.

Changement de position des modalités d'une dimension (switch) : *Switch(C, D, A, O)* échange les positions d'un ensemble d'attributs *A* au sein d'une dimension *D* d'un cube *C* selon l'ordre O. Comme la rotation, cette opération se caractérise principalement par un effet visuel puisque les valeurs des mesures ne sont pas modifiées avec le changement de position des faits correspondants.

Transformation d'une dimension en mesure (push) : *Push(C, D, A)* consiste à combiner un ensemble *A* de membres d'une dimension *D* aux mesures du cube *C*, c'est-à-dire à transformer ces membres en contenu de cellules de C.

Transformation d'une mesure en dimension (pull) : Opérateur réciproque de la transformation d'une dimension en mesure, *Pull(C, M, f, D)* transforme une mesure *M*, à l'aide d'une fonction de conversion *f*, en dimension *D* du cube *C* selon les exigences du modèle de données support de l'algèbre qui la met en œuvre.

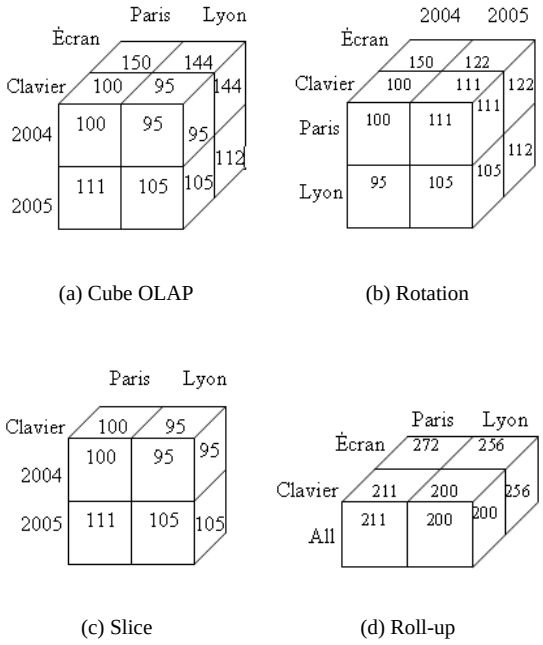
2.2 Opérateurs ensemblistes

Découpage de cube en tranches (slice) : *Slice(C, D, A)* dissocie une ou plusieurs tranches d'un cube de données multidimensionnel *C* selon un ensemble d'attributs *A* d'une seule dimension *D*, et ce pour tous les attributs des autres dimensions. Un cube peut d'ailleurs être vu comme un ensemble de tranches superposées les unes sur les autres (horizontalement) ou les unes à coté des autres (verticalement). Le *découpage de cube en tranches* s'effectue donc dans l'un de ces deux sens. Dans la Figure 1 (c), nous découpons le cube *Ventes* selon le *Produit* clavier en applicant ce principe.

Détachement d'un dé de cube (dice) : *Dice(C, P)* extrait un sous-cube du cube de données initial *C* selon un ensemble P de prédicats définis sur ses dimensions. Cette opération est assurée en sélectionnant les mesures du cube pour un certain nombre d'attributs dimensionnels, qui doit être le même pour tous les axes (dimensions) du cube. De plus, les cellules du dé résultant peuvent être reliées ou non au cube original.

2.3 Opérateurs liés à la granularité

Forage vers le haut (roll-up) : *Roll-up(C, D, n, f_{ag})* représente les données du cube *C* à un niveau n de granularité supérieur selon une dimension *D* et conformément à sa hiérarchie. Le but est de passer d'un niveau de granularité à un autre plus général, par exemple, d'une représentation multidimensionnelle des données suivant les villes à une autre suivant les pays. Ce passage nécessite l'association d'une fonction d'agrégation f_{ag} (somme, moyenne, etc.) à l'opération de groupement pour indiquer comment sont calculées les valeurs du niveau supérieur à partir de celles du niveau inférieur. Techniquement, un *forage vers le haut* consiste à aplatir (ou plier) les différentes tranches du cube en une seule tout en suivant la direction de l'axe représentant la dimension concernée. Chaque cellule du résultat final contient le résultat de l'agrégation des mesures correspondantes. L'exemple de la Figure 1 (d) illustre un forage vers le haut sur le cube *Vente* selon la dimension *Année*. Nous sommons ici les prix de vente des différents produits dans les différentes villes durant les années 2004 et 2005.



(a) Cube OLAP (b) Rotation

(c) Slice (d) Roll-up

FIG. 1 – *Exemples*

Forage vers le bas (drill-down) : Défini comme l'opérateur réciproque du forage vers le haut, *Drill-down(C, D, n)* permet d'obtenir des détails sur la signification d'un résultat en affinant une dimension *D* du cube *C* à un niveau d'agrégation inférieur n, par exemple, passer d'une représentation multidimensionnelle des données suivant les pays à une autre représentation suivant les villes. D'un point de vue technique, si le forage vers le haut permet de plier un cube, le *forage vers le bas* le déplie en détaillant les données suivant une ou plusieurs dimensions.

Construction de cube (cube) : *Cube(C,* $\mathcal{D}$*,* f_{ag}*)* peut être défini comme une opération d'agrégation généralisant le groupement, le forage vers le haut et la jointure sur un cube de données afin de produire un cube plus général. En premier lieu, un nombre de forages vers le haut égal au nombre de dimensions du cube est réalisé sur le cube initial *C*, pour chacune de ses dimensions $D_i \in \mathcal{D}$, où $\mathcal{D}$ est l'ensemble des dimensions de C. Dans une deuxième étape, afin de joindre les différents résultats de forage dans leurs dimensions communes, un calcul des mesures du forage est exigé pour chaque attribut de chaque axe, selon une fonction d'agrégation f_{ag} (somme, moyenne, etc.). Pour compléter le cube, il faut finalement grouper toutes les mesures du cube en une seule cellule en utilisant la même fonction d'agrégation. L'association de cette cellule avec les deux résultats précédents permet d'obtenir le cube de données agrégé.

3 Algèbres OLAP

Malgré le développement des systèmes décisionnels, les bases de données multidimensionnelles et la technologie OLAP souffrent d'un manque de formalisation précise. Nous nous intéressons dans cette section aux travaux qui proposent des modèles pour les données multidimensionnelles supportant une algèbre OLAP. Nous les classons en deux familles, selon les analogies qu'ils présentent avec les concepts relationnels et multidimensionnels, respectivement.

3.1 Extensions des concepts relationnels

3.1.1 Algèbres pour les cubes de données

Agrawal et al. (1997) modélisent les cubes multidimensionnels sous forme de repères à trois dimensions et proposent sur cette base une algèbre riche d'opérateurs inspirés essentiellement du relationnel. *Push* et *pull* suivent le principe décrit à la Section 2.1. *Destruction d'une dimension* supprime une dimension d'un cube. *Restriction* élimine dans une dimension certains attributs spécifiés. Le principe relationnel de *jointure* est transcrit en reliant deux cubes via leurs dimensions communes suivant une condition de jointure. *Fusion* (*merge*) réunit deux dimensions d'un cube en une seule. Les mesures résultantes sont calculées par l'intermédiaire d'une fonction d'agrégation. Les auteurs définissent ensuite des opérateurs OLAP basés sur cette algèbre. *Projection* utilise *fusion* puis *destruction d'une dimension*. *Union* et *intersection* emploient la *jointure*. Enfin, la *différence* entre deux cubes $C1$ et $C2$ est classiquement une opération d'*intersection* suivie d'une *union* du résultat avec le premier cube ($(C1 \cap C2) \cup C1$).

Li et Wang (1996) formalisent un cube de données relationnel pour une analyse OLAP et proposent deux algèbres dans ce contexte : une algèbre de groupement, extension de l'algèbre relationnelle, et une algèbre multidimensionnelle pour manipuler les cubes multidimensionnels. Nous ne nous intéressons ici qu'à la deuxième. *Ajout d'une dimension* génère un nouveau cube *C1* à partir d'un cube existant *C0* en y insérant une dimension. *Transfert* modifie l'emplacement d'un attribut dans une dimension. L'*union* de deux cubes (de même structure) s'effectue selon une ou plusieurs dimensions de jointure. Plusieurs opérations d'agrégation sont possibles sans passer par le groupement. L'*agrégation de cube* réduit la taille d'un cube en compressant les mesures relatives à ses dimensions. L'opérateur *rc-join* permet de joindre une relation avec une dimension d'un cube. Finalement, la *construction* génère un cube à partir d'une relation. Les dimensions et les mesures concernées sont spécifiées en entrée de l'opérateur.

Pour terminer, Gyssens et Lakshmanan (1997) modélisent les cubes de données multidimensionnelles en tables relationnelles tout en conservant les noms des dimensions, des faits et des attributs. Très proche de l'algèbre relationnelle, cette algèbre se compose de deux ensembles d'opérateurs. Le premier repose sur des opérateurs classiques qui fonctionnent comme ceux de l'algèbre relationnelle (*union, intersection, différence* et *produit cartésien*) et des opérateurs unaires (*sélection, projection* et *renommage*). Le second ensemble d'opérateurs propose deux opérateurs de restructuration : *unfold* ajoute une nouvelle dimension dans le schéma relationnel d'un cube tandis que *fold* élague une dimension d'un cube.

3.1.2 Algèbre pour l'analyse en ligne OLAP

Thomas et Datta (2001) proposent un modèle de cube de données qui respecte la symétrie entre les dimensions et les mesures (et permet donc la *transformation d'une mesure en dimension* et *vice versa*) et supporte une algèbre OLAP. Un cube est défini par ses dimensions (*D*), ses mesures (*M*), les attributs des dimensions (*A*), la relation entre les attributs et les dimensions (f), l'ensemble de valeurs utilisées dans le cube (*V*) et leurs domaines (g). Le premier opérateur de cette algèbre, *restriction*, limite un ensemble des valeurs *V* pour obtenir en sortie un ensemble $V0 \subseteq V$. L'opérateur d'*agrégation* permet d'agréger une mesure en groupant les attributs de la dimension correspondante. *Produit cubique* relie deux cubes en unifiant les dimensions et les mesures concernées. Les deux opérateurs suivants nécessitent que les cubes en entrée soient union-compatibles. *Union* permet d'unir deux cubes. C'est une opération différente du *produit cubique* puisqu'il s'agit d'unifier les ensembles de valeurs de deux cubes en une seule valeur *V0* dans le cube résultant. *Différence* consiste à soustraire l'ensemble de valeurs correspondant au deuxième cube du premier. *Projection métrique* réduit le nombre d'attributs correspondant aux mesures prises en compte dans la représentation des données, tandis que *renommage* permet de renommer les éléments d'un ensemble d'attributs ou de caractéristiques (mesures ou dimensions). Les deux derniers opérateurs, *extraction de dimensions* (*pull*) et *de mesures* (*push*), suivent le principe énoncé à la Section 2.1.

3.2 Extensions des concepts multidimensionnels

3.2.1 Algèbre pour les matrices bidimensionnelles

Gyssens et al. (1996) proposent une algèbre pour les données multidimensionnelles (noms, valeurs numériques ou nulles) modélisées de façon tabulaire. En l'absence d'une structure figée, ces tables bidimensionnelles peuvent prendre plusieurs formes tout en respectant l'aspect multidimensionnel des données. Les opérateurs traditionnels proposés (*union*, *différence*, *produit cartésien*, *renommage*, *projection* et *sélection*) s'inspirent du relationnel. De plus, des opérateurs de restructuration (*groupement*, *fusion*, *éclatement* et *compression*) sont proposés. *Groupement* réunit dans un tableau les informations en fonction d'un attribut donné. *Fusion* effectue l'opération réciproque. *Éclatement* permet de transformer une table en plusieurs, tandis que *compression* fusionne des tables. *Groupement* et *fusion* sont deux opérateurs qui manipulent essentiellement les données, alors qu'*éclatement* et *compression* manipulent leur structure. Grâce à l'opérateur de *transposition* (*transposing*), il est possible d'exprimer, pour toute opération sur les lignes d'une matrice, une opération similaire sur ses colonnes. Finalement, afin d'offrir une meilleure représentation des données, *suppression des redondances* (*clean-up*) élimine dans une table les valeurs nulles.

3.2.2 Algèbre pour les informations réparties en niveaux

Cabibbo et Torlone (1998) proposent un modèle logique des systèmes OLAP pour la conception des bases de données multidimensionnelles. Ainsi, ils modélisent les dimensions en hiérarchies, suivant un ordre prédéfini exprimé par une relation de forage vers le haut. Le schéma d'un modèle multidimensionnel consiste alors en un ensemble fini de dimensions et un ensemble fini de tables de faits. Malgré l'intérêt de cette approche, les auteurs se limitent à la formalisation de la famille de fonctions de forage vers le haut.

3.2.3 Algèbre pour les tables multidimensionnelles

Ravat et al. (2006) proposent un modèle qui sert de support à une algèbre OLAP (opérateurs classiques et avancés) et à un langage graphique pour l'analyse et la visualisation des données. Ils modélisent ainsi un cube de données par une structure de visualisation proche des arbres d'attributs, sous la forme d'un tableau à double entrée hiérarchisé appelé table multidimensionnelle (TM). L'algèbre proposée est riche et repose sur un opérateur de *construction* (*display*) qui produit une TM à partir d'une base de données multidimensionnelle et sur un ensemble de onze opérateurs fondamentaux portant sur les TM et facilitant la manipulation OLAP. *Rotation*, *changement de position des modalités dans une dimension*, *forage vers le haut* et *vers le bas*, *transformation de dimensions en mesures* et *de mesures de dimensions* suivent le principe décrit à la Section 2. L'opérateur d'*imbrication* (*nest*) enrichit une dimension par des attributs supplémentaires. La *sélection* restreint l'ensemble des valeurs affichées des attributs dimensionnels et des mesures correspondantes. Le calcul d'*agrégats* permet d'ajouter dans une TM des calculs agrégeant ses lignes et/ou ses colonnes. Enfin, les opérateurs d'*ajout* (*addm*) et de *suppression* (*delm*) *de mesures* permettent de modifier l'ensemble des mesures calculées.

3.2.4 Discussion

Dans la Table 1, nous évaluons les algèbres OLAP selon trois critères de comparaison qui, à l'exception du premier, sont transversaux aux familles d'algèbres que nous avons identifiées (extensions des concepts relationnels d'une part et multidimensionnels d'autre part). Ces critères sont le modèle de données adopté, les opérateurs disponibles et la gestion de la granularité des données.

	Modèle don.	**Opérateurs**	**Granularité**
Agrawal, Gyssens-Lakshmanan, Li-Wang, Thomas-Datta	Tab. rel.	Rel.	Prise en compte sauf par Gyssens-Lakshmanan
Cabibbo-Torlone, Gyssens, Ravat	Tab. multidim.	Rel. + OLAP	Prise en compte sauf par Gyssens

TAB. 1 – *Comparaison des algèbres OLAP*

La diversité des modèles de données utilisées dans les différentes algèbres provient du fait que les systèmes d'aide à la décision reposant sur la technologie OLAP ont existé avant la définition d'un fondement théorique standard et reconnu par la communauté des bases de données. De plus, outre les opérateurs OLAP classiques (Section 2) et les opérateurs inspirés du relationnel, peu d'opérateurs ont été proposés pour enrichir l'analyse en ligne en général (comme la *suppression des redondances* dans l'algèbre de Gyssens et al.). Néanmoins, l'inconvénient de ces nouveaux opérateurs est qu'ils soient totalement dépendants du modèle de données. Leur utilisation dans le cadre d'une autre algèbre OLAP nécessite une adaptation au modèle qui lui est associée. Enfin, la prise en compte de la granularité des données varie largement selon les algèbres OLAP. Certaines ne considèrent d'ailleurs pas du tout la granularité des données dans leurs différents opérateurs (Gyssens et Lakshmanan, 1997; Gyssens et al., 1996), vraisemblablement du fait que ces algèbres s'inspirent (trop) directement du cas relationnel. Dans

les autres travaux, la prise en compte de la granularité des données est toujours accompagnée par une implémentation des opérateurs de *forage vers le haut* et/ou *vers le bas* (*roll-up* et/ou *drill-down*) et parfois de l'opérateur *cube*.

4 Algèbres XML

Dans cette section, nous étudions les algèbres XML présentées dans la littérature. Bien que la plupart puisent leur logique dans celle des opérateurs relationnels, de nouveaux opérateurs sont apparus avec XML.

4.1 Algèbres pour les langages de requêtes XML

Le but de ces algèbres est d'optimiser les langages de requêtes XML en définissant de nouveaux opérateurs. Fernández et al. (2000) ont soumis au W3C (*World Wide Web Consortium*) une algèbre pour les requêtes XML riche et facile à utiliser. Les documents et les schémas XML y sont transformés en nouveaux schémas adaptés à l'algèbre. Les auteurs s'inspirent essentiellement de l'algèbre relationnelle et de XPath. La plupart des opérateurs proposés sont classiques et s'inspirent du relationnel (*projection, itération, sélection, restructuration, quantification, jointure* et *agrégation*), à l'exception des fonctions qui sont spécifiques aux langages de requêtes XML comme XQuery.

Zhang et al. (2002) proposent pour leur part l'algèbre XAT (*XML Algebra Tree*). XAT modélise les requêtes XQuery en arbres avant de les optimiser et présente trois catégories d'opérateurs : *XML, SQL* et *spécifiques XAT*. Les opérateurs classiques (*projection, sélection*, etc.) sont exprimés en SQL. Les opérateurs XAT assurent les diférentes phases d'optimisation. Les opérateurs XML (*union, intersection, différence, agrégation, exposition, étiquetage* et *navigation*) permettent de manipuler les données XML.

Finalement, Novak et Zamulin (2006) proposent une algèbre pour XQuery qui se limite volontairement à des structures de premier ordre. Ils définissent une expression élémentaire comme une séquence de nœuds (*sequel*) de l'arbre qui représente le document XML. Par la suite, une expression FLWOR du langage XQuery peut être définie par la combinaison de plusieurs expressions élémentaires. Les auteurs définissent des opérateurs classiques associés aux différentes clauses FLWOR (*parcours, sélection, ordre, construction du résultat, projection* et *jointure*), ainsi que de nouveaux opérateurs de construction de nœuds (*constructeurs de documents, d'éléments, d'attributs* et *de nœuds textes*).

4.2 Algèbres sur les arbres de données XML

Un document XML peut être représenté sous forme d'arbre de données enraciné, ordonné et étiqueté. Dans cette représentation, la racine du document est le nœud racine de l'arbre et ses éléments fils sont les nœuds fils de cette racine, et ainsi de suite pour le reste des éléments. Plusieurs auteurs définissent des algèbres XML sur ce modèle de données. Par exemple, Pradhan (2006) s'inspire du relationnel et propose une algèbre composée de trois opérateurs : *sélection*, *jointure* et *plus puissante jointure* qui offre la possiblité de joindre plus de deux fragments.

Jagadish et al. (2001) proposent également une algèbre logique d'arbres nommée TAX (*Tree Algebra for XML*) comme une extension de l'algèbre relationnelle. Dans un arbre TAX,

chaque nœud est modélisé par une paire attribut-valeur. Pour chaque opérateur, un modèle d'arbre (*pattern tree*) est défini, qui doit être suivi par l'arbre témoin (*witness tree*) résultat. TAX présente d'une part un ensemble d'opérateurs inspirés du relationnel : *sélection, projection, produit*, opérateurs ensemblistes (*union,intersection* et *différence*), *groupement* et agrégations (*somme, minimum, maximum* et *comptage*). D'autre part, TAX offre aussi des opérateurs de mise à jour d'arbres qui n'ont pas d'équivalent dans les travaux similaires (*renommage, réordonnancement, copier-et-coller, mise à jour des valeurs, suppression de nœuds* et *insertion de nœuds*). Paparizos et al. (2002) exploitent TAX et proposent une algèbre physique pour la manipulation des bases de données XML. En adoptant les mêmes principes que TAX, les auteurs expriment chaque opération logique TAX sous forme physique.

Frasincar et al. (2002) proposent l'algèbre XAL (*XML Algebra*) qui comprend trois familles d'opérateurs. Les opérateurs d'extraction (*projection, sélection, distinction, jointure, désordonnancement* et *tri*) recherchent l'information dans les graphes XML et renvoient la collection de sommets correspondants. Les méta-opérateurs contrôlent l'évaluation des expressions et expriment les répétitions au niveau des opérateurs ou des entrées (*map* pour appliquer une fonction sur chaque élément d'entrée et *kleene star* pour répéter une fonction pour une collection d'entrée donnée). Enfin, les opérateurs de construction (*création de sommets, création d'arêtes* et *copie d'exemples*) permettent de manipuler la structure des documents XML, c'est-à-dire les sommets et les arêtes, sans modifier leurs valeurs.

Finalement, l'algèbre Niagara présente des opérateurs similaires à ceux que nous venons de présenter : *source* pour la manipulation des racines des fichiers XML/DTD, *suivi* pour la projection, *exposition* pour rechercher les éléments spécifiques, *sélection, jointure, union, intersection, produit cartésien, groupement, création de sommets* et *renommage* (Galanis et al., 2001).

4.3 Algèbres sur les arbres de données XML pour l'évaluation de XQuery

Paparizos et al. (2004) exploitent TAX et proposent une nouvelle algèbre pour les classes logiques d'arbres nommée TLC (*Tree Logical Class*). Une classe logique contient l'ensemble des nœuds communs entre les arbres modèles et témoins (Section 4.2). À partir d'exemples réels, les auteurs démontrent qu'une requête XQuery peut être vue comme une suite d'opérations logiques, traduisibles en une suite d'opérateurs TLC. Les opérateurs de TLC s'inspirent de TAX : *filtrage, sélection, projection, élimination de doublons*, fonctions d'agrégation (*count, min, max*, etc.), *construction, étiquetage, renommage* et *rassemblement* (jointure).

Bose et al. (2003) proposent une algèbre de requêtes pour XQuery qui opère sur des flux de données XML répartis sur le réseau. Les auteurs proposent en premier lieu leur algèbre pour les données XML stockées sur une seule machine (données non fragmentées) et fournissent des opérateurs d'extraction des arbres XML (*sélection, projection, fusion, réduction* et *agrégation*). Ils associent ensuite ces opérateurs (à l'exception des agrégations) aux données fragmentées et les enrichissent par les opérateurs de *jointure*, de *parcours d'arbre* (*unnest*) et d'*évaluation de groupes* (*nest*).

4.4 Discussion

Nous synthétisons dans la Table 2 les caractéristiques des algèbres XML selon cinq critères identifiés par Zhang et Yao (2004) : le modèle de données sur lequel se base l'algèbre, l'unité

de base d'information manipulée par les opérateurs, les opérateurs, l'expressivité de l'algèbre avec les langages de requêtes XML et les possibilité d'optimisation de requêtes qu'offre l'algèbre.

	Modèle don.	**Unité info.**	**Op.**	**Expr.**	**Opt.**
Fernández	Schéma XML	Schéma XML	Rel. + XML	Algèbre	Oui
XAT	Arbres de req.	Non défini	Rel. + XML	XQuery	Oui
Novak	XPath	Séq. de nœuds	Rel. + XML	XQuery	Oui
Pradhan	Arbres XML	Fragment d'arbre	Rel.	XQuery	Non
TAX	Arbres XML	Arbre XML	Rel. + XML	XQuery et +	Oui
XAL	Arbres XML	Sommet d'arbre	Rel. + XML	XQuery	Oui
Niagara	Arbres XML	Sommet d'arbre	Rel. + XML	Quilt	Oui
Bose	Arbres XML	Arbre XML	Rel. + XML	XQuery	Oui

TAB. 2 – *Comparaison des algèbres XML*

Le modèle de représentation des documents XML généralement adopté dans la littérature est un arbre de données enraciné, ordonné et étiqueté. Il n'existe pas pour autant de représentation standard. Tous les modèles proposés respectent l'ordre entre les nœuds de l'arbre, sauf celui de Pradhan qui néglige l'ordre des nœuds du fragment recherché dans l'arbre initial, mais les relie avec le plus proche nœud parent commun. Une originalité des arbres de Bose et al. est la présence de nœuds vides ou *trous* (*holes*), auxquels d'autres sous-arbres enracinés, les *remplisseurs* (*fillers*), peuvent être associés dans le but de relier les fragments des documents XML. Les autres auteurs adoptent des représentations ad-hoc. Par exemple, Fernández et al., ainsi que Novak et Zamulin se basent sur XPath ; Zhang et al. représentent les requêtes XQuery par des arbres.

Par analogie avec les opérateurs relationnels qui opèrent sur des collections de n-uplets, les opérateurs des algèbres XML agissent sur des collections d'entités selon le modèle de données utilisé. Dans XAL et Niagara, un opérateur traite une collection de sommets. Par contre, les opérateurs de TAX et de l'algèbre de Bose et al. traitent une collection d'arbres. Selon Pradhan, l'unité d'information de base est le fragment, qui représente une suite de nœuds de l'arbre représentant le document XML. Les opérateurs de l'algèbre de Fernández et al. manipulent en entrée le contenu d'un schéma XML pour fournir un autre en sortie. L'entité principale manipulée dans le modèle de Novak et Zamulin est l'expression (séquence de nœuds). Dans XAT, il n'existe pas d'unité d'information de base en raison de la variété d'opérateurs présentés (relationnels, XML et spécifiques).

Certaines algèbres ne représentent que la traduction des opérateurs relationnels dans un contexte XML (Bose et al., 2003; Pradhan, 2006). La plupart des autres approches étendent les concepts relationnels par de nouveaux opérateurs qui diffèrent d'un travail à un autre. Dans l'ensemble de ces travaux, chaque opérateur est toujours dépendant du modèle de données. Parmi les opérateurs inspirés du relationnel, la *sélection* suit le même principe, contrairement à la *projection*. En effet, en algèbre relationnelle, projeter élimine du résultat les champs non spécifiés. Dans XAL et Niagara, l'opérateur de *projection* est similaire à un parcours d'expressions de chemins dans XPath. Dans TAX et ses dérivées, la *projection* recherche les entités en entrée de l'opérateur qui correspondent à un modèle défini en sélectionnant les attributs cités

dans une liste de projection. Par ailleurs, nous avons souligné l'importance de certains opérateurs spécifiques à XML comme les fonctions de Fernández et al., *copier-et-coller* de TAX, *sommet* de Niagara ou encore les opérateurs de construction de XAL.

Puisque le but d'une algèbre XML est de représenter des requêtes utilisateurs grâce à ses opérateurs, elle est jugée plus performante si elle peut être exprimée dans divers langages de requêtes XML. TAX sort du lot selon ce critère, car cette algèbre est supportée par la plupart des langages de requêtes XML tels que XQuery ou XML-QL. Les opérateurs de Niagara sont exprimés en Quilt. XAL et les algèbres respectives de Pradhan, Bose et al., Zhang et al. et Novak et Zamulin supportent les requêtes XQuery. L'algèbre de Fernández et al. constitue un cas particulier, puisqu'elle est elle-même conçue comme un langage de requêtes XML.

Finalement, les règles d'optimisation désignent les rôles que prennent ces opérateurs au sein de la requête. Le but d'une optimisation est de réduire le nombre de résultats intermédiaires et, par la suite, le nombre d'opérateurs utilisés. Toutes les algèbres, à l'exception de celle de Pradhan, offre cette possiblité, mais d'une façon différente pour chacune. Par exemple, dans Niagara, les deux opérateurs de *projection* (*follow*), *unnesting follow* et *straight follow*, sont interchangeables.

5 Algèbres XOLAP

Dans cette section, nous nous intéressons aux travaux qui abordent XML et OLAP dans un même contexte. Nous partons des premières tentatives de définition d'un environnement XML-OLAP et des notions associées pour arriver aux algèbres les plus récentes.

5.1 Analyse multidimensionnelle de données XML

Jensen et al. (2001) proposent un travail qui se résume en trois points. Ils présentent tout d'abord une architecture d'intégration des données XML et relationnelles. Ils proposent ensuite une modélisation multidimensionnelle des bases de données (XML/relationnelles) OLAP avec UML. Enfin, ils illustrent les considérations nécessaires à prendre en compte pour concevoir des bases de données XML supportant des analyses OLAP. En sortie du système d'intégration proposé, les données sont présentées en relationnel et évaluées avec OQL. Dans la même logique, Niemi et al. (2002) présentent une architecture d'intégration de données XML et un modèle de base de données OLAP exploités par un langage dédié, MDX.

Pedersen et al. (2004) préconisent également d'intégrer des données XML et relationnelles au sein d'un même système. De plus, ils proposent une approche pour fédérer des sources de données XML avec des cubes OLAP existants. Une fédération se compose d'un cube, d'une collection de documents XML et de liens entre le cube et les documents. Un cube multidimensionnel est décrit par son nom et ses tables de dimensions et de faits. Chaque dimension est vue comme une hiérarchie de niveaux, où chaque niveau est associé à un groupe de valeurs dimensionnelles. La table de faits est une relation contenant un attribut pour chaque dimension et un attribut pour chaque mesure. Ce modèle de cube est associé à une algèbre composée de trois opérateurs. L'opérateur le plus fondamental dans une fédération OLAP-XML, *décoration*, attache une nouvelle dimension à un cube en se basant sur les valeurs des éléments XML liés. La *sélection* sert à filtrer les mesures. L'opérateur de *projection généralisée* (*generalized*

projection) a pour rôle l'agrégation des mesures. La *projection* est vue ici comme une opération de *sélection* employant une clause de *groupement* (*group by*). Ces opérateurs sont mis en œuvre sur les données OLAP-XML à l'aide d'une extension de SQL_M, nommée SQL_{XM}, qui permet d'associer des requêtes XPath aux requêtes SQL_M. SQL_M est lui-même une extension de SQL pour le traitement des données multidimensionnelles.

5.2 Agrégation, groupement et XML-OLAP

Park et al. (2005) proposent un cadre pour l'analyse en ligne OLAP des données XML appelé XML-OLAP et introduisent la notion de cube XML (XQ-Cube). Pour interroger un cube XML, les auteurs proposent un langage pour les expressions multidimensionnelles, XML-MDX, inspiré du langage d'expressions multidimensionnel MDX de Microsoft. Ils implémentent la création de cubes XML (*CREATE XQ-CUBE*) et leur interrogation (*SELECT*). En outre, ils proposent sept opérateurs d'agrégation : *ajout* (*ADD*), *listage* (*LIST*), *comptage* (*COUNT*), *récaputilation* (*SUMMARY*), *extraction du sujet* (*TOPIC*), *extraction du plus important mot-clé* (*TOP KEYWORD*) et *classification* (*CLUSTER*). Certains opérateurs sont inspirés du relationnel, d'autres sont conçus pour les données non-additives et exploitent des techniques de fouille de textes (*text mining*).

Dans le même contexte, Wang et al. (2005) présentent des concepts pour XOLAP (OLAP sur données XML). Ils définissent un opérateur général d'agrégation pour XML, *GXaggregation*, qui forme la base de *XCube*, une extension de l'opérateur traditionnel cube pour les données XML. Mis en œuvre avec une extension de XQuery, *GXaggregation* permet l'extraction de propriétés à partir des dimensions et des mesures suivant leurs expressions de chemin XPath. Par conséquent, calculer des statistiques sur les données de XML devient plus aisé. Enfin, ce processus est associé aux fonctions qui agrègent des données hétérogènes sur des hiérarchies.

BenMessaoud et al. (2006) proposent un opérateur d'agrégation OLAP basé sur une méthode automatique de classification (*clustering*) : *OpAC*. Cet opérateur permet des analyses précises et fournit les agrégats sémantiques sur des données complexes représentées dans des documents XML.

Beyer et al. (2005) affirment que les requêtes analytiques exprimées en XQuery sont difficiles à lire, à écrire et à manipuler d'une façon efficace. Afin d'améliorer cela, ils étendent les expressions FLWOR de XQuery avec une syntaxe explicite de groupement et de comptage des résultats. Face aux problèmes générés par le groupement avec XQuery, ils présentent également des solutions originales traitant l'aspect hétérogène et hiérarchique des données XML.

Finalement, Wiwatwattana et al. (2007) affirment que les extensions du modèle relationnel ne peuvent pas prendre en compte toutes les spécificités de XML. Ils proposent en conséquence un entrepôt de treillis de cubes XML, un opérateur cube intitulé X^3 et un mécanisme de spécification généralisé. Ils discutent également le mécanisme de construction de cube et comparent plusieurs algorithmes alternatifs. X^3 a été implémenté en C++ au sein du SGBD natif XML TIMBER. Ce travail est une extension de l'algèbre TAX.

5.3 Discussion

Nous évaluons, dans la Table 3, les algèbres XOLAP selon les critères suivants : le modèle de données support de l'algèbre, les opérateurs proposées et le langage utilisé pour l'exploita-

tion des données et l'expression des opérateurs.

	Modèle de données	**Opérateurs**	**Langage**
Jensen, Niemi	Intégration XML/rel.	Aucun	OQL/MDX
Pedersen	Intégration XML/rel.	Décoration (jointure), projection, sélection	SQL_{XM}
Park	Cubes XML	Cube, sélection, agrégation	XML-MDX
OpAC	Documents XML	Agrégation	Non défini
Wang	Documents XML	Agrégation, cube	XQuery
Beyer	Documents XML	Groupement, comptage	XQuery
X^3	Treillis de cubes XML	Cube	XQuery

TAB. 3 – *Comparaison des algèbres XOLAP*

L'ensemble des travaux de la Section 5.1 repose sur des architectures intégrant les données XML et relationnelles simultanément. Dans les travaux de Jensen et al. et Niemi et al., les opérateurs OLAP spécifiques à XML sont en effet absents, puisque les données OLAP sont exploitées avec des langages de requêtes pour les données relationnelles. Ainsi, les données XML sont traduites en relationnel avant leur utilisation au sein de ces architectures d'intégration. Le travail de Pedersen et al. représente la première véritable tentative de définition d'une algèbre permettant de réaliser une analyse en ligne OLAP sur des cubes XML. Toutefois, cette algèbre se caractérise par seulement deux opérateurs de base exprimés en SQL. Les travaux de la Section 5.2 ont adopté des structures multidimensionnelles de données représentées dans des documents ou des cubes XML. À l'exception du travail de Beyer et al., la prise en compte de la granularité des données dans ces algèbres est claire grâce à la présence d'opérateurs comme *agrégation* ou *cube*. L'ensemble d'opérateurs XOLAP disponibles reste cependant très restreint.

6 Conclusion et travaux futurs

Dans cet article, nous avons, avec en point de mire la définition d'une algèbre XOLAP, dressé un état de l'art que nous espérons le plus exhaustif possible au sujet des algèbres OLAP (Section 3) et des algèbres XML (Section 4). Les efforts fournis ces dernières années pour formaliser des algèbres OLAP nous permettent de disposer d'un cadre formel et d'opérateurs bien identifiés. Les opérateurs OLAP existants étant définis dans un contexte soit relationnel, soit multidimensionnel, il nous reste désormais à les adapter au modèle de données des documents XML (classiquement un arbre, plus généralement un graphe), ainsi qu'à les enrichir de nouveaux opérateurs spécifiquement adaptés au contexte XML. Les travaux existants qui tendent vers XOLAP (Section 5) ne satisfont pas complètement ces objectifs. Certains privilégient la traduction des cubes XML en relationnel et leur interrogation avec des extensions du langage SQL, d'autres tendent vers des solutions multidimensionnelles utilisant des langages de requêtes pour XML comme XQuery ou encore XML-MDX. Nous tendons pour notre part, comme les auteurs de X^3, vers une solution native XML exploitant le langage XQuery pour

l'interrogation. De plus, ces travaux ne proposent en terme d'algèbre qu'un nombre d'opérateurs assez limité.

Afin de définir notre algèbre XOLAP, nous avons donc choisi d'exprimer les opérateurs OLAP "de base" (Section 2) dans une algèbre XML. Notre choix s'est porté sur TAX. Notre motivation principale provient de la richesse de cette algèbre qui comprend, sous ses formes logique et physique, plus qu'une vingtaine d'opérateurs. Cela nous offre une grande liberté pour les combiner et exprimer les différents opérateurs OLAP que nous souhaitons mettre en œuvre. De plus, l'expressivité de TAX est largement reconnue, puisque cette algèbre peut s'exprimer avec la majorité des langages de requêtes XML, et notamment XQuery, qui est le langage que nous ciblons pour des analyses OLAP. Enfin, TAX et son algèbre dérivée TLC fournissent un cadre d'optimisisation de requêtes que nous pourrons exploiter, car la performance est un de nos principaux soucis pour des applications décisionnelles intégralement basées sur XML et XQuery.

Les premières perspectives qui s'offrent à nous sont de deux ordres. Premièrement, nous sommes en train d'implémenter notre proposition d'algèbre XOLAP logique au niveau physique, dans le cadre d'une plateforme d'entreposage XML de données complexes. Notre objectif est de permettre, à travers une interface simple et accessible depuis le Web, la construction et la manipulation de cubes XML. Dans un second temps, nous comptons incorporer dans notre algèbre et notre prototype de nouveaux opérateurs spécifiques au contexte XML. Ils nous paraît, par exemple, particulièrement intéressant de pouvoir effectuer des opérations de forage (*roll-up* et *drill-down*) sur des hiérarchies de dimensions complexes telles que les *ragged hierarchies* mentionnées par Beyer et al. (2005). L'objectif de ce travail est de venir en support des extensions en cours du langage XQuery pour les applications décisionnelles.

Références

Agrawal, R., A. Gupta, et S. Sarawagi (1997). Modeling Multidimensional Databases. In *13th International Conference on Data Engineering (ICDE 97), Birmingham, UK*, pp. 232–243. IEEE Computer Society.

BenMessaoud, R., S. Rabaséda, et O. Boussaïd (2006). A Data Mining-Based OLAP Aggregation of Complex Data : Application on XML Document. *International Journal of Data Warehousing & Mining 2*(4), 1–26.

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

Borkar, V. R. et M. J. Carey (2004). Extending XQuery for Grouping, Duplicate Elimination, and Outer Joins. In *XML 2004, Washington DC, USA*.

Bose, S., L. Fegaras, D. Levine, et V. Chaluvadi (2003). A Query Algebra for Fragmented XML Stream Data. In *9th International Workshop on Database Programming Languages (DBPL 03), Potsdam, Germany*, Volume 2921 of *LNCS*, pp. 195–215. Springer.

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.

Cabibbo, L. et R. Torlone (1998). A logical approach to multidimensional databases. In *6th International Conference on Extending Database Technology (EDBT 98), Valencia, Spain*, Volume 1377 of *LNCS*, pp. 183–197. Springer.

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.

Fernández, M. F., J. Siméon, et P. Wadler (2000). An Algebra for XML Query. In *20th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 00), New Delhi, India*, Volume 1974 of *LNCS*, pp. 11–45. Springer.

Frasincar, F., G.-J. Houben, et C. Pau (2002). XAL : An Algebra For XML Query Optimization. In *13th Australasian Database Conference (ADC 02), Melbourne, Victoria*, Volume 5 of *CRPIT*. Australian Computer Society.

Galanis, L., E. Viglas, D. J. DeWitt, J. F. Naughton, et D. Maier (2001). Following the paths of XML Data : An algebraic framework for XML query evaluation. Technical report, University of Wisconsin, USA.

Gyssens, M. et L. V. S. Lakshmanan (1997). A foundation for multi-dimensional databases. In *23rd International Conference on Very Large Data Bases (VLDB 97), Athens, Greece*, pp. 106–115. Morgan Kaufmann.

Gyssens, M., L. V. S. Lakshmanan, et I. N. Subramanian (1996). Tables as a paradigm for querying and restructuring. In *15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 96), Montreal, Canada*, pp. 93–103. ACM.

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.

Jensen, M. R., T. H. Moller, et T. B. Pedersen (2001). Specifying OLAP cubes on XML data. *Journal of Intelligent Information Systems 17*(2-3), 255–280.

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

Lawton, G. (2006). Making business intelligence more useful. *Computer 39*(9), 14–16.

Li, C. et X. S. Wang (1996). A data model for supporting on-line analytical processing. In *5th International Conference on Information and Knowledge Management (CIKM 96), Rockville, USA*, pp. 81–88. ACM.

Niemi, T., M. Niinimäki, J. Nummenmaa, et P. Thanisch (2002). Constructing an OLAP cube from distributed XML data. In *5th International Workshop on Data Warehousing and OLAP (DOLAP 02), McLean, USA*, pp. 22–27. ACM.

Novak, L. et A. V. Zamulin (2006). An XML Algebra for XQuery. In *10th East European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece*, Volume 4152 of *LNCS*, pp. 4–21. Springer.

Paparizos, S., S. Al-Khalifa, H. Jagadish, A. Nierman, et Y. Wu (2002). A physical algebra for XML. Technical report, University of Michigan, USA.

Paparizos, S., Y. Wu, L. V. S. Lakshmanan, et H. V. Jagadish (2004). Tree Logical Classes for Efficient 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 *LNCS*, pp. 32–42. Springer.

Pedersen, D., J. Pedersen, et T. B. Pedersen (2004). Integrating XML Data in the TARGIT OLAP System. In *20th International Conference on Data Engineering (ICDE 2004), Boston, USA*, pp. 778–781. IEEE Computer Society.

Pradhan, S. (2006). An Algebraic Query Model for Effective and Efficient Retrieval of XML Fragments. In *32nd International Conference on Very Large Data Bases (VLDB 06), Seoul, Korea*, pp. 295–306. ACM.

Ravat, F., O. Teste, et G. Zurfluh (2006). *Constraint-Based Multi-Dimensional Databases*, pp. 323–368. Database Modeling for Industrial Data Management. Idea Group Publishing.

Thomas, H. et A. Datta (2001). A Conceptual Model and Algebra for On-Line Analytical Processing in Decision Support Databases. *Information Systems Research 12*(1), 83–102.

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

Wiwatwattana, N., H. V. Jagadish, L. V. S. Lakshmanan, et D. Srivastava (2007). X^3 : A Cube Operator for XML OLAP. In *23rd International Conference on Data Engineering (ICDE 07), Istanbul, Turkey*, pp. 916–925.

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.

Zhang, M. et J. Yao (2004). XML Algebras for Data Mining. In *Data Mining and Knowledge Discovery : Theory, Tools, and Technology VI, Orlando, USA*, Volume 5433 of *SPIE proceedings series*, pp. 209–217. SPIE.

Zhang, X., B. Pielech, et E. A. Rundesnteiner (2002). Honey, I shrunk the XQuery ! : an XML algebra optimization approach. In *4th International Workshop on Web Information and Data Management (WIDM 02), McLean, USA*, pp. 15–22. ACM.

Summary

With the rise of XML as a standard for representing business data, XML data warehouses appear as suitable solutions for Web-based decision-support applications. In this context, it is necessary to allow OLAP analyses on XML data cubes. Thus, XQuery extensions are needed. To help define a formal framework and allow much-needed performance optimizations on analytical queries expressed in XQuery, we work at defining and XML-OLAP (or XOLAP) algebra. The aim of this paper is to present, ahead of this research, the state of the art in the area, i.e., regarding OLAP algebras, XML algebras and algebras that tend toward XOLAP. We also briefly present our own XOLAP algebra proposal.