Du XML au multidimensionnel : Conception de magasins de données

Yasser Hachaichi, Jamel Feki, Hanene Ben-Abdallah

Laboratoire MIRACL, Faculté des Sciences Economiques et de Gestion de Sfax
Route de l'Aérodrome km 4, B.P. 1088, 3018 Sfax, Tunisie
{Yasser.Hachaichi, Jamel.Feki, Hanene.Benabdallah}@fsegs.rnu.tn

Résumé. Avec l'ouverture des entreprises sur l'Internet, les sources de données englobent davantage des données échangées avec les partenaires et/ou issues du Web. Dans ce cadre organisationnel ouvert, les documents XML constituent des sources de données plus utilisées aussi bien pour le stockage que pour les échanges transactionnels. Ce nouveau format de données a motivé des propositions de modèles pour les entrepôts de données XML et des démarches de conception d'entrepôts à partir de sources XML. Cependant, les démarches proposées requièrent une intense intervention qui nécessite une double expertise dans le domaine des sources XML et celui d'entreposage de données. Cet article propose une démarche automatique de conception de schémas multidimensionnels ayant un potentiel analytique à partir de sources XML. Il détaille les étapes de la démarche et l'illustre sur un exemple.

1 Introduction

Dans un contexte organisationnel sans cesse complexe et mouvant, les entreprises sont confrontées à une concurrence sévère et à des clients de plus en plus exigeants. Pour faire face, l'entreprise doit prévoir et planifier en s'appuyant sur l'analyse d'informations soigneusement collectées et préparées. Dans ce contexte, l'informatique décisionnelle offre des bases de données rigoureusement conçues pour ces analyses, appelées entrepôts de données (ED) et magasins de données (MD). Traditionnellement, les ED/MD sont alimentés par des informations pertinentes issues de sources de données opérationnelles, internes à l'entreprise.

Récemment, et avec l'ouverture des entreprises sur l'Internet, les sources de données englobent davantage des données échangées avec les partenaires et/ou issues du Web. Dans ce nouveau cadre organisationnel ouvert, les documents XML constituent des sources de données de plus en plus fréquentes, utilisées aussi bien pour le stockage que pour les échanges transactionnels. En effet, grâce à sa flexibilité quant à la modélisation des données, XML est devenu le langage élite pour décrire des informations et les échanger avec les partenaires. Par conséquent, il convient de disposer de mécanismes adéquats pour extraire et structurer les informations pertinentes à la prise de décision à partir de sources XML.

Face à cette problématique, plusieurs travaux ont proposé soit des modèles pour les entrepôts de données XML (*cf.*, (Hümmer et al., 2003) et (Boussaid et al., 2006)), soit des démarches semi-automatiques de conception de ED/MD à partir de sources XML (*cf.*, (Golfarelli et al., 2001), (Vrdoljak et al., 2003) et (Jensen et al., 2007)). Etant semi-automatiques, les démarches proposées requièrent une intense intervention pour

l'identification des concepts multidimensionnels. Une telle intervention nécessite une double expertise dans le domaine des sources XML ainsi que dans celle du domaine décisionnel.

Cet article traite cette limite en présentant une méthode *automatique* de conception de schémas multidimensionnels à partir de sources XML décrites par des DTD (« Document Type Definition »). Notre méthode est constituée de trois étapes : *prétraitement*, *construction* et *validation*. Le prétraitement vise d'abord à dompter la complexité de la DTD initiale ; ensuite, il traduit automatiquement la DTD simplifiée en des *arbres de transition* qu'il enrichit par les cardinalités et les types des éléments en consultant des documents XML. La construction exploite les arbres de transition pour en extraire automatiquement les concepts multidimensionnels (faits, mesures, dimensions…). Elle est fondée sur un ensemble de règles d'extraction que nous définissons pour chaque concept multidimensionnel. Elle produit des schémas multidimensionnels en étoile, qui peuvant être ultérieurement intégrés pour construire des schémas en constellation. Finalement, l'étape de validation assiste le concepteur à adapter les schémas obtenus à des besoins analytiques particuliers.

Cet article décrit brièvement l'étape de prétraitement où il introduit le concept d'arbre de transition et les notations employées, ensuite il détaille les règles d'extraction pour la construction de schémas et les illustre sur un exemple. Il est organisé comme suit : La section 2 donne un aperçu sur l'état de l'art des méthodes de conception de systèmes décisionnels. La section 3 décrit notre approche. La section 4 présente l'étape de prétraitement. La section 5 détaille la construction de schémas multidimensionnels et énumère ses règles. Finalement, la section 6 résume les travaux réalisés et ceux en cours.

2 Etat de l'art et motivations

Durant la dernière décennie, les méthodes d'entreposage de données ont adopté essentiellement trois approches : 1) L'approche descendante (Kimball, 1997) qui part d'une étude des besoins analytiques exprimés par les futurs utilisateurs décisionnels ; 2) l'approche ascendante qui commence à partir du modèle informatique du système d'information (SI) opérationnel de l'entreprise (Golfarelli et al., 1998), (Cabbibo et Torlone, 1998), (Moody et Kortink, 2000), (Husemann et al., 2000), (Pang et al., 2006), (Feki et Hachaichi, 2007) ; ou encore, 3) l'approche mixte qui combine les deux approches précédentes (Böhnlein et al., 1999), (Phipps et Davis, 2002), (Bonifati et al., 2001), (Ghozzi, 2004), (Soussi et al, 2005).

La première approche requiert une expertise dans l'expression de besoins analytiques pour lesquels le système d'information (SI) source contient les données nécessaires. En revanche, l'approche ascendante extrait des besoins analytiques potentiels à partir desquels le décideur peut définir ses propres besoins. Ainsi, cette deuxième approche jouit d'un double avantage : elle allège la tâche du décideur en lui proposant des besoins analytiques et garantit que le SI de l'entreprise peut alimenter les besoins choisis. Vu ces avantages, nous présentons dans cet article une méthode de conception de systèmes d'information décisionnels (SID) selon l'approche ascendante et, en particulier, pour les sources XML. Nous nous limitons dans cette section à passer en revue quelques travaux connus qui ont adopté cette approche sur les sources XML.

Les auteurs (Pokorny, 2001), (Hümmer et al., 2003) et (Boussaid et al., 2006) ont proposé de modéliser les schémas multidimensionnels sous format XML sans présenter comment ces schémas sont construits. L'objectif principal de leurs travaux était de proposer des modèles pour les ED XML, plutôt qu' une démarche pour leur conception.

Par ailleurs, (Golfarelli et al., 2001) ont proposé une méthode semi-automatique de construction de schéma de MD à partir de sources XML. La méthode proposée est basée sur deux hypothèses : (1) l'existence d'une DTD pour la source XML et (2) la conformité des documents XML à cette DTD. La démarche proposée comprend essentiellement les trois étapes suivantes : une simplification de la DTD, la création d'un graphe de DTD et le choix des faits. La simplification de la DTD aplatit les définitions imbriquées, groupe les éléments de même nom, remplace une succession d'opérateur par un seul et l'opérateur + par *. Le graphe de DTD prépare à la création d'un arbre d'attributs et à la sélection manuelle des faits. Pour chaque fait choisi l'arbre d'attributs est réarrangé pour définir les dimensions et les mesures. La méthode exige soit l'intervention d'un expert dans le domaine des documents XML alimentant le futur ED, soit un concepteur décisionnel ayant une bonne connaissance du domaine.

(Vrdoljak et al., 2003) ont développé un processus semi automatique de conception d'ED directement à partir de sources XML. Ce processus est composé des trois étapes suivantes : établissement du graphe de dépendance à partir du graphe du schéma XML, réarrangement du graphe de dépendance, définition des dimensions et des mesures, et création du schéma logique. Par rapport à (Golfarelli et al., 2001), ces auteurs proposent de détecter les hiérarchies partagées et la convergence des dépendances et tentent de résoudre le problème de modélisation des relations plusieurs à plusieurs dans les hiérarchies. Cependant, ce processus nécessite aussi une intervention intensive du concepteur. En effet, outre l'identification manuelle des faits, mesures et dimensions, lorsque l'examen des documents XML est insuffisant pour dégager les liens multiples (n-n) le concepteur doit se référer à ses connaissances pour décider si les relations sont multiples et si elles sont intéressantes pour l'agrégation.

D'autre part, les auteurs de Rusu et al., (2004, 2005) proposent une méthode générique de construction d'un ED XML pour les documents XML. La méthode se compose principalement de cinq étapes : nettoyage et intégration des données, réduction synthétique des données, création de documents XML intermédiaires, mise à jour/enchaînement des documents existants et finalement création d'un ED complet. Dans cette méthode, la majorité des étapes sont accomplies manuellement par un expert du domaine d'application des documents XML de départ.

De leur part, (Jensen et al., 2004) ont étudié comment un cube OLAP peut être obtenu à partir de données XML. Pour atteindre leur objectif, ils procèdent comme suit : D'abord la DTD des documents XML est transformée en diagramme de classes UML (Unified Modeling Language) en utilisant l'ensemble de règles de transformation décrites dans (Jensen et al., 2001). Ensuite, le concepteur utilise ce diagramme de classe pour spécifier un modèle de données OLAP, appelé "UML snowflake diagram", manuellement et moyennant une interface graphique. Finalement, le "UML snowflake diagram" est transformé en structures relationnelles pour préparer l'implémentation du système OLAP.

(Ouaret et al., 2007) proposent un processus pour la modélisation conceptuelle et logique d'un ED XML. Leur processus comporte quatre étapes : (1) extraire et sélectionner les structures et le schéma XML des documents XML (Avec un arbitrage humain nécessaire), (2) construire le diagramme de classes UML global en utilisant les techniques de mapping de (Li et al., 2004), (3) construire le diagramme de classes UML en étoile (faits et dimensions) à partir du diagramme de classes global, et (4) générer pour chaque classe de fait le schéma en étoile XML équivalent.

Etant semi-automatiques, les démarches proposées requièrent une intense intervention pour l'identification du fait à analyser et de ses dimensions.

Notre proposition pallie à cette limite en présentant une méthode *automatique* de conception de schémas multidimensionnels à partir de DTD.

3 Aperçu de la démarche proposée

La force de XML réside dans sa capacité de pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. En fait, XML permet de structurer et de définir le vocabulaire et la syntaxe des données qu'un document légal peut contenir. La définition du vocabulaire et de la syntaxe (*i.e.*, la grammaire) d'une famille de documents XML est décrite à travers une DTD ou un Schéma XML.

Dans ce qui suit, nous décrivons brièvement les composants d'une DTD puisqu' ils constituent la base de notre méthode.

3.1 Aperçu sur les DTD

Une DTD définit la structure de documents à l'aide d'une liste d'*éléments* légaux pouvant avoir des *attributs typés* ; Le mot-clé !ELEMENT déclare un élément et !ATTLIST précise ses attributs. La DTD permet aussi de déclarer le nombre de fois qu'un élément fils apparaît dans un élément parent : un ou plus (+), zéro ou plus (*) ou zéro ou un (?). En outre, il existe cinq types d'attributs : les types ID et IDREF permettent de créer des relations entre les attributs, à l'image des clés étrangères utilisées dans les systèmes relationnels ; le type #PCDATA désigne des données textuelles devant être traitées par l'analyseur, tandis que le type #CDATA désigne les données textuelles qui ne doivent pas être traitées par l'analyseur ; et le type EMPTY signifiant que l'élément est vide. Pour de plus amples détails sur les DTD le lecteur peut se référer à (w3c, 2008).

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT vente (date, client, produit+, vendeur)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT client (adresse)>
<!ATTLIST client id_client ID #REQUIRED nom CDATA #IMPLIED >
<!ELEMENT adresse (localité, ZIP)>
<!ELEMENT localité (#PCDATA)>
<!ELEMENT ZIP (#PCDATA)>
<!ELEMENT produit (id_produit, prix, quantites, description?)>
<!ELEMENT id_produit (#PCDATA)>
<!ELEMENT prix (#PCDATA)>
<!ELEMENT quantites (#PCDATA)>
<!ELEMENT vendeur EMPTY>
<!ATTLIST vendeur id_vendeur ID #REQUIRED nom CDATA #REQUIRED >
<!ELEMENT description (#PCDATA)>
```

FIG. 1 – *Un exemple de DTD pour les documents Ventes*.

A titre d'exemple, la DTD Vente (Figure 1) modélise la structure et le contenu d'un document XML d'une vente. Dans cet exemple, une *vente* est réalisée à une *date* par *un vendeur* pour *un client* qui achète *un ou plusieurs produits*. Un exemple de document XML valide par rapport à cette DTD est illustré dans la figure 2. Il concerne le client BRAYAN qui a acheté le 12/03/2008 auprès du vendeur SAMUEL un CD ORACLE 10G et un livre sur les bases de données.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vente SYSTEM "vente.dtd">
<vente>
        <date>12/03/2008</date>
        <client id_client="C_100" nom="BRAYAN">
                        <adresse>
                    <localité> immeuble INFO appartement DSS </localité>
                    <ZIP>E 12 DSS</ZIP>
                        </adresse>
        </client>
        <produit>
                        <id_produit>CDs_ORACLE 10G</id_produit>
                        <prix>160.16</prix>
                        <quantites>1</quantites>
                        <description>certifié oracle pour un seul usage interne</description>
        </produit>
        <produit>
                        <id_produit>livre_BD</id_produit>
                        <prix>20.00</prix>
                        <quantites>1</quantites>
                        <description>documentations sur les BD oracle</description>
        </produit>
        <vendeur id_vendeur="AC_10" nom="SAMUEL"/>
</vente>
```

FIG. 2 – *Un document XML conforme à la DTD Vente*.

3.2 Proposition d'une démarche

Nous proposons une démarche qui construit des schémas de MD candidats à partir de sources XML en examinant leur DTD. Cette démarche exploite les *liens structurels* entre les éléments de la DTD qui traduisent partiellement la *sémantique* du document. D'autre part, vu la pauvreté des DTD en information de typage, la démarche retrouve les types en interrogeant des documents conformes à une DTD. Ces liens structurels et ces informations de typage aident à l'identification des concepts multidimensionnels dans les documents XML. Plus précisément, notre démarche se compose des trois étapes suivantes :

- **Prétraitement.** Comme toute donnée semi structurée, la structure d'une DTD est généralement complexe (Beeri et al., 1999). Pour maîtriser les difficultés engendrées par cette complexité, cette étape simplifie chaque DTD utilisée dans la conception du système décisionnel, construit pour chaque DTD simplifié ses *arbres*

de transition, et enrichit ces derniers par les types de données retrouvées dans les documents XML conformes à la DTD. Les arbres de transition constituent une structure qui représente les liens structurels entre les éléments de la DTD qui facilite l'identification des concepts multidimensionnels.

- **Construction de schémas de MD.** Cette étape construit des *schémas en étoile candidats* en exploitant les arbres de transition. Elle extrait les concepts multidimensionnels : les faits, leurs mesures et dimensions ainsi que leurs attributs dimensionnels organisés en hiérarchies. Elle applique automatiquement un ensemble de règles qui associent à chaque concept extrait son correspondant dans la source (*i.e.*, élément de la DTD) ainsi qu'un niveau de pertinence indiquant son potentiel analytique. L'association concept-source prépare le passage vers le niveau logique en garantissant la faisabilité des opérations ultérieures de chargement. L'assignation d'un niveau de pertinence assiste l'étape suivante d'ajustement/validation.
- **Ajustement/Validation.** Dans cette étape, le concepteur décisionnel peut ajuster les schémas multidimensionnels candidats en les adaptant aux besoins analytiques du système de pilotage. Pour ce faire, nous définissons un ensemble d'opérateurs inspirés de nos travaux sur les patrons multidimensionnels (Feki et al., 2007). (Hachaichi et al., 2008-a) Ces opérateurs permettront la suppression, le renommage et/ou la restructuration des éléments multidimensionnels tout en préservant l'association de chaque concept du schéma avec sa source (composant de la DTD). De plus, le décideur tire profit des niveaux de pertinence dans la validation des concepts multidimensionnels d'un schéma candidat en focalisant ses efforts beaucoup plus sur les concepts ayant des niveaux de pertinence élevés.

Nous détaillons, dans le reste de l'article, les deux premières étapes (prétraitement et construction de schémas de MD) de cette démarche, qui génèrent automatiquement des schémas conceptuels de MD à partir de DTD et de documents conformes à ces DTD.

4 Prétraitement

Rappelons que l'étape de prétraitement vise à restructurer les DTD afin de préparer l'extraction des concepts multidimensionnels. Nous la divisons en trois sous étapes, à savoir : *Simplification d'une DTD*, *Construction de ses arbres de transition*, et *enrichissement des arbres* par ajout des types des données (Hachaichi et al., 2008-b).

4.1 Simplification d'une DTD

Cette simplification produit une DTD simplifiée notée DTD^S qui facilite le processus de construction des arbres de transition. La DTD^S est obtenue en éliminant de sa DTD initiale des composants inutiles au processus décisionnel et en réécrivant les déclarations des éléments de manière plus concise. Ainsi, elle compacte les liens entre les éléments et généralise les cardinalités[1]. En conséquence, une DTD^S n'est pas équivalente à son DTD ; ceci n'est pas problématique du fait qu'elle n'est pas considérée pour la validation de documents XML.

Nous effectuons cette simplification à travers les opérations suivantes :

[1] Cardinalité désigne le nombre d'occurrence d'un élément dans un autre.

- *Suppression des éléments vides.* Cette opération supprime les éléments EMPTY de la DTD. Par exemple, la déclaration <!ELEMENT test EMPTY> qui se traduit dans un document XML par l'un des deux formats <test></test> ou <test/> ne contient aucune information et ne présente, en conséquence, aucun intérêt pour la modélisation multidimensionnelle.
- *Substitution des références*. Une entité, définie dans une DTD par ENTITY, déclare un groupe d'éléments nommé afin d'éviter sa répétition dans la DTD en le référençant. Dans un document XML, une référence permet une meilleure lisibilité, un contrôle accru sur le contenu et une souplesse de mise à jour. L'opération de substitution des références vise à remplacer toute référence à une ENTITE par la valeur de cette entité. Egalement, elle supprime la déclaration, devenant inutile, de l'entité. Cette substitution garantira la désignation des concepts multidimensionnels identifiés par leur nom au lieu de références non significatives. Par ailleurs, elle facilitera les opérations ultérieures de chargement par une localisation plus directe de ces concepts dans les documents XML.
- *Simplification des déclarations d'éléments.* Dans une DTD, les éléments sont souvent imbriqués à plusieurs niveaux. Cette simplification aplatit les déclarations, réduit les cardinalités et compacte les écritures. Elle applique un ensemble de règles de transformation proches de celles présentées dans (Shanmugasundarma et al., 1999), (Yan et al., 2001) et similaires à celles de (Jensen et al., 2001). Nos règles sont classées en trois catégories (figure 3) : (a) *Règles d'aplatissement* qui transforment une déclaration d'élément complexe en son équivalent plat où les opérateurs "," et "|" n'apparaissent plus entre parenthèses ; (b) *Règles de réduction* qui remplacent plusieurs opérateurs consécutifs par un seul ; et (c) *Règles de groupement* qui compactent les apparitions multiples d'un même sous élément en une seule ayant la cardinalité la plus générique. Notons qu'en appliquant ces règles de groupement nous perdons l'ordre initial d'apparition des sous éléments dans l'élément conteneur. Cependant, cette perte n'affecte pas notre méthode d'extraction. Ces trois types de transformation sont appliqués dans l'ordre *a*, *b* puis *c*. Les règles de chacune doivent être appliquées de façon itérative jusqu'à ce qu'aucune d'elles ne puisse plus être appliquée. La figure 3 donne les règles de transformation de chacune des trois catégories ; la flèche (→) se lit '*se transforme en*'.

(a) Aplatissement	(b) Réduction	(c) Groupement
(e1, e2)*→e1*, e2*	e1**→ e1*	…,e1*,…,e1*, …→ e1*,…
(e1, e2)?→e1?, e2?	e1*?→ e1*	…,e1*,…,e1?, …→ e1*,…
(e1|e2) →e1?, e2?	e1?*→ e1*	…,e1?, …,e1*, …→e1*,…
	e1??→ e1?	…,e1?,…,e1?, …→ e1*,…
		…, e1, …, e1, …→ e1+,…

FIG. 3 – *Trois catégories de règles de simplification des déclarations d'éléments de DTD*

4.2 Construction des arbres de transition

Une DTD^S est utile pour construire des arbres de transition : Un arbre de transition *A* est un graphe dont les nœuds racine et intermédiaires représentent des éléments non PCDATA et où chaque feuille est soit un élément PCDATA, un attribut ou une référence vers le sommet d'un arbre de transition (éventuellement *A* en cas de récursivité).

Chaque arbre de transition représente une entité du domaine des documents XML utilisés pour la construction des MD. Par ailleurs, notons que l'existence d'une référence entre deux arbres de transition implique l'existence d'un lien entre leurs entités. Ainsi, les arbres de transition issus d'une même DTD^s se réfèrent. Nous exploitons cette caractéristique pour identifier les liens hiérarchiques entre les attributs dimensionnels.

Nous décrivons, dans ce qui suit, la construction des arbres de transition constituée de trois sous étapes : *détermination des sommets*, *construction* et *enrichissement*.

Détermination des sommets des arbres de transition. Nous identifions les sommets des arbres de transition parmi les éléments de la DTD^S en appliquant quatre règles. Ces dernières tiennent compte de la complexité des éléments et de leur fréquence d'apparition dans d'autres éléments. Nous nous limitons à énumérer ces règles, amplement détaillées dans (Hachaichi et al, 2008).

- **RD1 :** Tout élément qui n'apparaît pas dans les déclarations d'autres éléments est un sommet pour un arbre de transition.
- **RD2 :** Tout élément contenant au moins un élément non PCDATA est un sommet pour un arbre de transition.
- **RD3 :** Tout élément se trouvant dans la déclaration de n ($n \geq 2$) autres éléments est un sommet pour un arbre de transition.
- **RD4 :** Tout élément qui apparaît récursivement (directement ou indirectement) dans sa propre déclaration est un sommet pour un arbre de transition.

Construction des arbres de transition. Détermine pour chaque sommet identifié ses nœuds descendants (intermédiaires et terminaux) ainsi que ses arcs de transition reliant les nœuds.

Soit *R* un sommet déterminé par l'une des quatre règles précédentes, l'arbre de transition construit sur le sommet *R* est obtenu en appliquant l'algorithme *Createtree*(R, DTD^S) suivant.

```
CreateTree(E, DTD^S)  // E le nœud actuel du DTD^S
{
  pour chaque élément e dans la déclaration de E faire
  {
        addChildNode(E, e) //ajoute un enfant e au nœud E
        markArcCard (E, e) // annote l'arc de E vers e avec la cardinalité de e
        si ( e est identifié comme racine) alors
              markNode(e, #) //racine déjà identifiée avec RD1-RD4
        sinon si (e contient autres éléments ou attributs) alors
              CreateTree(e, DTD^S)
  }
  pour chaque attribut a dans la déclaration de E faire
  {
              addChildNode(E, a)
              markArcType(E, a)   //annote l'arc de E vers a avec le type de a
  }
}
```

Dans cet algorithme :

- La fonction markArcCard(*E, e*) annote l'arc partant de *E* vers *e* avec la cardinalité désignant les occurrences d'apparition de *e* dans *E*. Le symbole d'annotation de

cardinalité appartient à l'ensemble {?, *, +, 1, espace} où l'espace correspond à la valeur 1. La cardinalité maximale 1 (? et 1) et la cardinalité maximale plusieurs (+ et *) aideront à identifier les dimensions et les paramètres hiérarchiques (Golfarelli et al., 1998) (Husemann et al., 2000). Chacune des cardinalités minimales 0 (? et *) et 1 (+ et 1) permettront de classer certains concepts multidimensionnels selon leur pertinence dans la prise de décision.

- La fonction markNode(*e*, #) marque par le symbole # un noeud déjà identifié comme sommet d'un arbre de transition. Cette notation est empruntée à celle utilisée dans les bases de données relationnelles pour marquer les clés étrangères. En effet, elle permet de mettre en évidence les liens entre les arbres de transition appartenant à une même DTD^S et nous aide ultérieurement à établir des liens entre les paramètres hiérarchiques des dimensions.
- La fonction markArcType(*E, a*) permet d'étiqueter l'arc de transition de *E* vers l'attribut *a* par les types associés à la déclaration de *a* dans la DTD. Les abréviations d'étiquetage utilisées sont {I, R, F, ID, IDREF} où *I* désigne un attribut facultatif (spécifié par #IMPLIED), *R* un attribut qui doit être renseigné (#REQUIRED), *F* un attribut ayant toujours la même valeur (#FIXED), *ID* un identifiant unique, et *IDREF* un attribut qui référence un ID. Nous complétons une étiquette IDREF d'un arc de transition par l'attribut référencé placé entre parenthèses. L'introduction de ces types va nous permettre ultérieurement d'associer un niveau de pertinence aux concepts extraits et limiter le champ d'investigation de certains concepts multidimensionnels. Par exemple, lors de la détermination des mesures candidates nous écarterons les attributs facultatifs (étiquette I), constants (F) et les identifiants (ID) car ces attributs représentent des informations artificielles ou redondantes qui ne tracent pas l'activité de l'entreprise.

Notons que notre méthode construit des arbres de transition d'une profondeur maximale égale à quatre et faciles à parcourir en phase d'extraction des concepts multidimensionnels.

L'application de cet algorithme sur les sommets Vente et Client génère les arbres de transition de la figure 4.

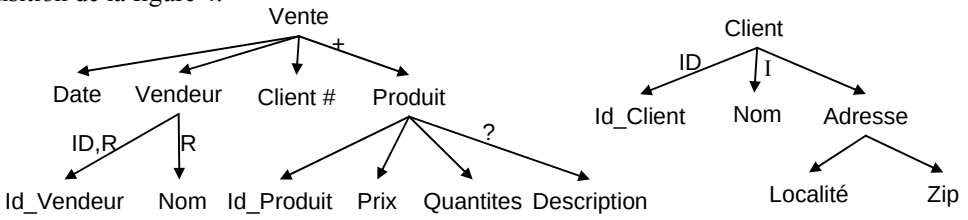

FIG. 4 –*Les arbres de transition construits à partir de la DTD Vente.*

Enrichissement de l'arbre de transition. En général, l'identification de certains concepts multidimensionnels se base sur les types de données (*e.g.,* numériques, date,…) (Feki et Hachaichi, 2007). Cependant, dans une DTD ce type n'est pas précis puisque tout est défini comme chaîne de caractères (PCDATA et CDATA).

Cette étape enrichit les arbres de transition construits par les types de données. Cet enrichissement se restreint aux feuilles (non marquées par le symbole #) d'un arbre puisqu'elles correspondent aux attributs et aux éléments PCDATA qui prennent des valeurs dans les documents XML et sont, par conséquent, susceptibles de fournir des concepts multidimensionnels.

Afin de déterminer ces types pour les arbres de transition issus d'une même DTD (via la DTD^s), nous interrogeons, moyennant un parseur (XQuery, XML-QL…), un ensemble de documents XML conformes à cette DTD de départ. Ainsi, pour chaque feuille (non marquée par #) nous scannons le texte contenu dans la balise correspondante dans les documents XML et nous décidons du type approprié de la feuille (Numérique, date ou chaîne de caractères).

Ainsi, l'enrichissement des arbres de transition de notre exemple de la figure 4 produit les résultats illustrés par la figure 5 où les types de données sont ajoutés aux feuilles non marquées par #.

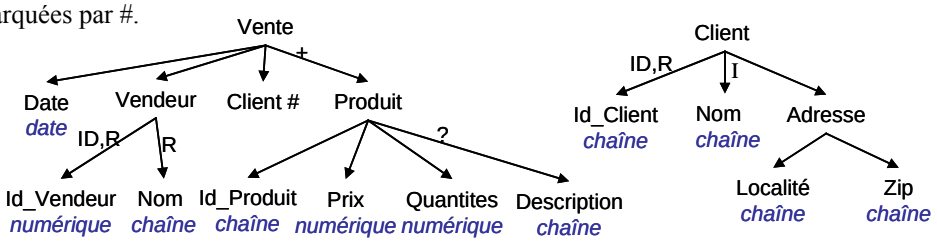

FIG. 5 –*Les arbres de transition construits à partir de la DTD Vente.*

Notons que les DCD (Document Content Descriptors) et XMLSchemas (extension de DTD) spécifient les types de données. Ainsi, si elles deviennent largement utilisées, alors l'étape d'enrichissement pourrait être intégrée à la construction ; ce qui préserve la méthode proposée.

Les arbres de transitions ainsi enrichis nous permettent maintenant d'entamer l'extraction des concepts multidimensionnels.

5 Construction des schémas de MD

Cette construction produit des schémas de MD candidats pour des documents XML conformes à une DTD. Elle part des arbres de transition construits en phase de prétraitement et identifie les concepts multidimensionnels moyennant un ensemble de règles d'extraction. Nos règles ont le mérite de préparer le passage au niveau logique en associant à chaque concept extrait son correspondant dans les documents XML et d'assister la phase de validation en attribuant un niveau de pertinence à chaque concept extrait. Cette construction commence par l'extraction des faits.

5.1 Extraction des faits

Le fait représente un centre d'intérêt pour la prise de décision. En effet, il modélise un sujet d'analyse représentant un événement qui se produit au sein d'une organisation.

Dans le cadre des méthodes de conception ascendantes les faits sont identifiés manuellement/semi-automatiquement soit parmi les représentations conceptuelles (entités ou associations) (Kimball 1997), (Bonifati et al. 2001), (Soussi et al. 2005), soit parmi les relations (Feki et Hachaichi, 2007) possédant au moins un attribut numérique non clé (primaire ou étrangère). Cependant, pour les sources XML (Golfarelli et al., 2001) (Vrdoljak et al., 2003) les faits sont sélectionnés manuellement.

Dans notre méthode, nous retrouvons les faits candidats à partir des sommets des arbres de transition en appliquant la règle *Rf*.

Rf : Le sommet *A* de tout arbre de transition contenant au moins une feuille vérifiant les trois conditions i) non marquée par #, ii) de type numérique et iii) dont l'arc de transition est sans étiquette ou étiqueté *R* constitue un fait candidat pertinent nommé *F-A*.

Afin d'assister le concepteur décisionnels, nous classons l'ensemble des faits extraits par *Rf* en deux sous classes selon leur pertinence en appliquant la règle *Rcf*. suivante.

Rcf : Un fait est considéré pertinent à la prise de décision si l'arbre de transition correspondant possède au moins une feuille annotée par #.

La complexité d'un arbre possédant des feuilles annotés par # garantit pour le fait construit sur cet arbre une multiplicité d'axes d'analyses et ainsi un privilège par rapport aux autres faits candidats.

L'application des règles *Rf* et *Rcf* sur les arbres de transition de la DTD Vente (figure 5) dégage le nœud *Vente* comme fait pertinent à la prise de décision.

L'extraction des faits se complète par celle des mesures.

5.2 Extraction des mesures

Chaque fait possède une ou plusieurs mesures. Il s'agit d'observations régulières des évolutions d'un phénomène (le fait) par rapport à des objectifs fixés. Les mesures d'un fait sont identifiées par la règle suivante :

Rm : Dans un arbre de transition dont le sommet est identifié comme fait, toute feuille non marquée par #, de type numérique et reliée par un arc non marqué {I, F, ID, IDREF} est une mesure candidate pour ce fait.

Généralement, les mesures d'un fait sont très proches du nœud fait. Ceci n'exclut pas l'existence de mesures plus distantes mais moins pertinentes. Ainsi, nous convenons de classer les mesures extraites par *Rm* en deux classes de pertinence.

Rcm : Une mesure identifiée est pertinente à la prise de décision si elle est liée directement au nœud fait.

L'application des règles Rm et Rcm sur l'arbre de sommet Vente (figure 5) identifie les nœuds *Prix* et *Quantites* comme mesures pertinentes pour le fait *Vente*.

Pour compléter la construction des schémas de MD, nous déterminons pour chaque fait ses dimensions.

5.3 Extraction des dimensions

Les dimensions sont les axes d'observation sur lesquels le décideur souhaite évaluer, quantifier ou qualifier les faits. Chaque dimension est caractérisée par un nom et possède un ensemble d'attributs. Nous définissons quatre règles pour identifier les dimensions.

Rd1: Tout nœud *N* de profondeur deux d'un arbre-fait (i.e., arbre dont le sommet est identifié comme fait) est une dimension nommée *D-N* pour le fait de cet arbre.

Rd2 : Tout nœud *N* de profondeur deux, annoté par # et appartenant à un arbre-fait est une dimension nommée *D-N* pour ce fait.

Rd3 : Tout nœud *N1* ayant un arc sortant annoté ID vers un nœud référencé par *N2* (un autre nœud) dont le parent est un fait *F* est une dimension (nommée *D-N1*) pour *F*.

Rd4 : Tout nœud *N* de profondeur deux, non annoté par #, de type *Date* et appartenant à un arbre-fait F estampille F et lui construit une dimension temporelle nommée *D-Date-N*.

La dimension temps figure systématiquement dans tout entrepôt (Kimball 1997). Ainsi, nous considérons pertinentes les dimensions obtenues avec Rd4. Quand aux dimensions obtenues avec Rd1 à Rd3, nous les classons avec la règle Rdc.

Rdc : Une dimension construite sur un nœud *N* avec l'une des règles Rd1 à Rd3 n'est considérée pertinente que si l'arc entrant à *N* est soit non étiqueté soit étiqueté 1 ou +.

Nous basons ce classement sur la présence obligatoire de l'information dans les documents XML exprimée par la cardinalité minimale 1 (arc non marqué ou marqué 1 ou +).

L'application de ces règles sur l'arbre de transition Vente (figure 5) dégage pour le fait *Vente* les dimensions pertinentes *Vendeurs* et *Produit* par Rd1 et *Client* et *Date* respectivement par Rd2 et Rd4.

La construction des dimensions se poursuit par celle de leurs hiérarchies.

5.4 Extraction des hiérarchies

Une hiérarchie organise les paramètres d'une dimension selon une relation *"est_plus_fin"* conformément à leur niveau de détail (Teste, 2000). Par ailleurs, toute hiérarchie d'une dimension *D* part de l'identifiant de *D* qui est le paramètre le plus fin (*i.e.*, de rang 1).

Extraction des identifiants. Nous définissons dans cette section deux règles d'extraction des identifiants de dimension selon que la dimension est extraite à partir d'un élément (avec Rd1 à Rd3) ou d'un attribut (Rd4).

Rid1 : Si dans un arbre de transition un nœud *N* est lié à un nœud-dimension *N1* (i.e., identifié par Rd1, Rd2 ou Rd3) via un arc annoté ID alors *N* est un identifiant de *N1*.

Notons que si la règle Rid1 n'extrait pas un identifiant pour une dimension de nom *D* obtenue par Rd1 à Rd3 alors nous associons à *D* un identifiant artificiel séquentiel nommé *id_D* .

Rid2 : Une dimension temporelle construite (par Rd4) sur un nœud nommé *D* aura pour identifiant le nom de ce nœud *D*.

Un nœud-dimension obtenu par *Rd4* est en réalité un attribut possédant une valeur dans un document XML ; ainsi il est utile à alimenter l'identifiant.

Identification des paramètres de rang 2. Pour construire les hiérarchies nous continuons à extraire les paramètres de rang supérieur à 1.

Notons que les dimensions extraites par Rd1 et Rd2 proviennent des nœuds correspondant à des éléments dans une DTD. Ces éléments peuvent être liés à d'autres ou contenir des attributs fournissant éventuellement des niveaux hiérarchiques. Par contre les dimensions extraites par Rd3 proviennent d'attributs et, en conséquence, ne peuvent s'étendre à d'autres niveaux de paramètres.

Nous présentons dans ce qui suit trois règles d'extraction des paramètres de rang 2, ensuite nous traitons l'extraction des paramètres de rang supérieur à 2.

Rp1: Tout nœud terminal, annoté par # et lié à un nœud-dimension *N* est un paramètre de rang 2 d'une hiérarchie définie sur *N*. Le nom de ce paramètre est celui du nœud terminal.

Rp2 : Tout nœud non terminal lié à un nœud-dimension *N* est un paramètre de rang 2, nommé *P-N*, d'une hiérarchie définie sur *N*.

Rp3 : Tout nœud *N1* ayant un arc sortant annoté ID vers un nœud référencé par *N2* (un autre nœud) dont le parent est une dimension *d* est un paramètre de rang 2 d'une hiérarchie définie sur *d*. Le nom de ce paramètre est celui du nœud *N1*.

Rp4 : Tout nœud *N* de type *Date* lié à un nœud-dimension *d* est un paramètre de rang 2, nommé *P-N*, d'une hiérarchie définie sur *d*.

Plus généralement, l'application récursive des règles Rp1 à Rp4 sur les nœuds identifiés par Rp1 à Rp3 produit des paramètres de rang supérieur à deux.

Notons que si une dimension est extraite à partir d'un nœud *N* annoté par # alors ces règles ne s'appliquent pas sur *N* mais s'appliqueront sur le sommet de l'arbre référé par *N*.

Pour l'association du concept source servant ultérieurement à l'alimentation, les paramètres identifiés par Rp4 ne posent pas de problèmes puisqu'ils correspondent soit à des attributs soit à des éléments PCDATA renseignés dans les documents. Tandis que ceux identifiés par Rp1 à Rp3 correspondent à des éléments non PCDATA et ne peuvent être renseignés dans les documents. Pour résoudre ce problème nous associons à chaque paramètre de ce type l'identifiant de l'élément source. Dans le cas de d'absence de spécification d'identifiant nous donnons la main au concepteur, en phase de *Validation/raffinement*, pour sélectionner un.

Par analogie aux dimensions, nous considérons que les paramètres obtenus avec Rp4 sont pertinents et nous classons ceux obtenus avec Rp1 à Rp4 selon la cardinalité minimale correspondante.

L'application de la règle Rp2 sur l'arbre Vente dégage le nœud *Adresse* comme paramètre de rang 2 pour la dimension *Client*.

Identification des attributs faibles. Les paramètres hiérarchiques peuvent être accompagnés de descripteurs appelés *attributs faibles*. Ces attributs ne sont pas utilisés dans les calculs d'agrégation. Ils ont un rôle informationnel permettant de faciliter la compréhension des résultats d'analyses. Nous présentons dans ce qui suit la règle d'extraction des attributs faibles.

Raf : Tout nœud terminal, non marqué par # et lié à un nœud-paramètre *P* (*i.e.,* identifié comme paramètre) via un arc non annoté par ID ou IDREF est un attribut faible pour *P*.

Vu que les attributs faibles sont descriptifs, nous considérons que ceux textuels sont plus pertinents que les attributs numériques (Hachaichi et al., 2007).

L'application de la règle Raf sur permet de dégager comme attributs faibles pertinents les nœuds *Zip* et *Localité* pour le paramètre *Adresse*, le nœud *Nom* pour la dimension *Client*, le nœud *Description* pour la dimension *Produit* et le nœud *Nom* pour la dimension *Vendeur*.

La figure 6 représente le MD construit pour la DTD Vente.

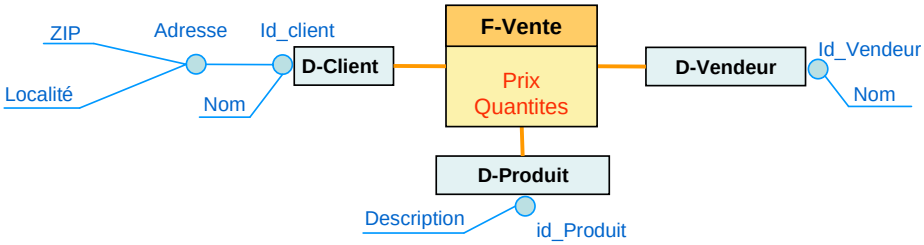

FIG. 6 – *Schéma en étoile construit à partir de la DTD Vente.*

6 Conclusion

Cet article a présenté une méthode automatisable, ascendante de construction de schémas en étoile à partir d'une source XML pour laquelle nous disposons d'un ensemble de documents XML conforme à une DTD. Cette méthode simplifie une DTD initiale en

éliminant des redondances, compactant ses déclarations et les aplatissant. Ensuite, elle génère un ensemble d'arbres de transition représentant les liens structurels entre les éléments des documents. Enfin, elle enrichit ces arbres par les types de données afin de faciliter l'extraction automatique des éléments multidimensionnels et la construction de schémas de magasins de données. Les types de données sont extraits en interrogeant des documents XML conformes à la DTD initiale.

Les schémas en étoile ainsi obtenus peuvent être adaptés par le décideur pour spécifier ses besoins particuliers. Une fois ajustés et validés, ces schémas seront modélisés logiquement par le développeur qui sera assisté par les associations entre les éléments multidimensionnels et les éléments sources ; ces associations constituent un deuxième avantage de notre méthode qui s'ajoute à son aspect automatique.

Une première évaluation de la méthode sur des sources de la littérature des SID a produit des résultats encourageants. Nous comptons continuer cette évaluation sur des exemples plus complexes.

Références

Beeri, C. et T. Milo (1999). *Schemas for integration and translation of structured and semistructured data*. In Proc. of the Int. Conf. on Database Theory.

Böhnlein, M. et A. Ulbrich-vom Ende (1999). *Deriving Initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems*.

Bonifati, A., F. Cattaneo, S. Ceri, A. Fuggetta et S. Paraboschi (2001). Designing Data Marts for Data Warehouse, in ACM Transaction on Software Engineering and Methodology, ACM, vol. 10, Octobre 2001, 452-483.

Boussaïd O., R. BenMessaoud, R. Choquet et S. Anthoard (2006), *X-Warehousing: an XML-BasedApproach for Warehousing Complex Data*, 10th East-European Conference on Advances in Databases and Information Systems (ADBIS 06).

Cabibbo, L. et R. Torlone (1998). *A Logical Approach to Multidimensional Databases*, Conference on Extended Database Technology, Valencia, Spain,1998, 187-197.

Feki J. et H. Ben-Abdallah (2007). *Multidimensional Pattern Construction and Logical Reuse for the Design of Data Marts*, International Review on Computers and Software (IRECOS), vol 2 no. 2, 24-134.

Feki, J., et Y. Hachaichi (2007), *Du relationnel au multidimensionnel : Conception de magasins de données*, Revue des Nouvelles Technologies de l'Information : Entrepôts de données et analyse en ligne (EDA 2007 ; Broché), 5-19.

Golfarelli, M., D. Maio, et S. Rizzi (1998). *Conceptual Design of Data Warehouses from E/R Schemas*. Conference on System Sciences,Vol. VII, Kona, Hawaii.

Golfarelli, M., S. Rizzi et B. Vrdoljak (2001). *Data warehouse design from XML sources*. Proc. DOLAP'01, Atlanta, 40-47.

Ghozzi, F. (2004), *Conception et manipulation des bases de données dimensionnelles à contraintes*, Thèse de Doctorat, Université Paul Sabatier, France.

Hachaichi, Y., H. Ben-Abdallah et J. Feki (2008).*Vers une approche à la MDA pour le développement d'entrepôts de données*, Conférence internationale Systèmes d'Information et Intelligence Economique (SIIE'2008), Hammamet-Tunisie.

Hachaichi, Y., J. Feki et H. Ben-Abdallah (2008). *XML Source Preparation for Building Data Warehouses*. International Conference on Enterprise Information Systems and Web Technologies EISWT-08 (à paraître).

Hüsemann, B., J. Lechtenbörger et G. Vossen (2000). *Conceptual Data Warehouse Design*. Proc. of the Int'l Workshop on Design and Management of Data Warehouses, Stockholm, Sweden, 6.1-6.11.

Jensen, M., T. Møller et T.B. Pedersen (2001). *Specifying OLAP Cubes On XML Data*, Journal of Intelligent Information Systems.

Kimball, R. (1997). *The Data Warehouse Toolkit*, John Wiley and Sons, Inc.

Li, Y. et A. Aijun (2004). *Converting XML to UML for OLAP Using XML Schema and XMI*, Proceedings of the Third International Workshop on Cooperative Internet Computing (CIC'04), Hong Kong, China.

Moody, L.D. et M.A.R. Kortink (2000). *From Enterprise Models to Dimensional Models: A Methodology for Data Warehouses and Data Mart Design*. Proc. of the Int'l Workshop on Design and Management of Data Warehouses, Stockholm, Sweden.

Ouaret, Z., L. Bellatreche et O. Boussaid (2007). *XUML Star : Conception d'un entrepôt de données XML*, Atelier des Systèmes d'Information Décisionnels, Sousse – Tunisie.

Pang, C., K. Taylor, X. Zhang et M. Cameron (2004). *Generating Multidimensional Schemata from Relational Aggregation Queries.* LNCS 3306, 584–589,

Phipps, C. et K. Davis (2002). Automating data warehouse conceptual schema design and evaluation. DMDW'02, Canada.

Rusu, L.I., W. Rahayu et D. Taniar (2005). *A methodology for Building XML Data Warehouses*, International Journal of Data warehousing & Mining, 1(2), 67-92.

Shanmugasundarma J., K. Tufte, G. He, C. Zhang, D. DeWitt et J. Naughton (1999). *Relational database for querying xml documents: limitation and opportunities*. Procedins of the 25th VLDB Conferences, Edinburgh, Scotland.

Soussi, A., J. Feki et F. Gargouri (2005). *Approche semi-automatisée de conception de schémas multidimensionnels valides*, EDA 05, Revue RNTI vol B-1, 71-90.

Teste, O. (2000). *Modélisation et manipulation d'entrepôts de données complexes et historisées*. Thèse de l'Université Paul Sabatier (Toulouse III).

Vrdoljak, B., M. Banek et S. Rizzi (2001). *Designing Web Warehouses from XML Schema*, 5th International Conference Data Warehousing and Knowledge Discovery: DaWak,

World Wide Web Consortium XML Schema (2008), *W3C Candidate Recommendation*, www.w3.org/XML/Schema.html. Current as of January 1st, 2008.

Yan M.H et W.C.F. ADA (2001). *From XML to Relational Database*. CEUR Workshop Proceedings.

Summary

With the opening of enterprises on the Internet, data sources include more data exchanged with their partners and/or stemming from the Web. Within this free organizational framework, XML documents constitute a source of data more used as well for the storage as for the transactional exchanges. This new data format motivated propositions of XML data warehouse models and methods for data warehouse design from XML. However, proposed methods require an intensive intervention of expert in the domain of XML sources and in the decision-making domain. This paper proposes an automatic method to design XML multidimensional star schemes from XML documents. It details the steps of the method and illustrates it with an example.