RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINSTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ Ammar TELIDJI - LAGHOUAT
Faculté des Sciences
Département de Mathématique et Informatique

MÉMOIRE DE MASTER

Domaine : Mathématique et Informatique
Filière : Informatique
Option : Système d'Information & Décision
Thème

Modélisation conceptuelle et logique d'un entrepôt de données XML

Présenté par :
BOUCHAREB Sara LATROUCI Imane

Soutenu devant le jury composé de :

Mr. XXXXXXXXXXX	Président	U. Amar Telidji, Laghouat
Mr. XXXXXXXXXXX	Examinateur	U. Amar Telidji, Laghouat
Mr. XXXXXXXXXXX	Examinateur	U. Amar Telidji, Laghouat
Mlle N.HAMINI	Encadreur	U. Amar Telidji, Laghouat

Année / ...../2012-2013

Modélisation Conceptuel et Logique des entrepots de données XML.

Realiser par :Bouchareb SARAH & LATROUCI Imane
Dirige par:Melle N.HAMINI

3 octobre 2013

Remerciement

Résumé

De nos jours, nous assistons à une évolution des systèmes d'information et de l'internet qui ne cesse d'engendrer des flux de données de plus en plus hétérogènes (structurées, semi-structurées, non structurées). XML a été largement adopté comme la langue unifiée permettant de représenter ces sources hétérogènes afin de les intégrer dans un entrepôt de données. De nombreuses sources XML Sont disponibles sur le web pour les applications de commerce électronique « business to business », etc. D'autre part, l'UML à été utilisé afin de mieux comprendre et documenter ces aspects business. Nous proposons dans ce mémoire une méthode de conception d'un entrepôt de données XML, basé sur XML et UML, que nous appelée XStar-Warehouse. Le schéma XML représentant la source de données est converti en un diagramme UML en étoile puis exprimé par un schéma XML en étoile.

Mots-clés : Entrepôt de données, XML, Entrepôt de données XML, schéma XML, UML, Mapping UML vers XML.

Abstract

Nowadays, we assist an evolution of the information systems and Internet which a large number of heterogeneous data sources(structured, semi-structured, not structured). XML was largely adopted like the unified language to represent these heterogeneous sources in order to integrate them in a data warehouse. Many XML sources Are available on the Web for the applications of E-commerce "business to business", etc. On the other hand, UML has been used in order to well understand and document business aspect. In this paper, we propose a conceptual and logical methodology to design XML Warhouse using XML and UML which we call ≪ XStar-Warhouse ≫. Our methodology starts by converting XML schemas of data sources into a star UML schema using a set of mapping and then expressed them using star XMLschema.

Keywords : : Data Warehouse, XML, XML data warehouse, UML, UML to XML Mapping.

Table des matières

Table des figures iii

Liste des tableaux v

Introduction 1

1 Les Entrepôts de Données 5
- 1.1 Introduction : . . . 5
- 1.2 SID vs transactionnels . . . 5
  - 1.2.1 Système d'information (SI) : . . . 5
  - 1.2.2 Système d'information décisionnel(SID) . . . 6
  - 1.2.3 Architecture des systèmes décisionnels . . . 7
- 1.3 Outils De Stockage pour l'aide à la décision . . . 8
  - 1.3.1 Les entrepôts de données *(Datawarehouse)* . . . 8
  - 1.3.2 Magasin de données(*DATA MART*) . . . 10
- 1.4 Entrepôt de données vs Base de données . . . 11
- 1.5 Modélisation multidimensionnelle . . . 11
  - 1.5.1 Concepts de bases . . . 12
  - 1.5.2 Niveau conceptuel . . . 12
  - 1.5.3 Niveau logique . . . 13
  - 1.5.4 Niveau physique . . . 15
- 1.6 Approches de conception des entrepôts de données . . . 15
  - 1.6.1 Approche ascendante . . . 16
  - 1.6.2 L'approche descendante . . . 16
  - 1.6.3 L'approche hybride (ou mixte) . . . 16
- 1.7 Conclusion . . . 16

2 Base de données XML 17
- 2.1 Introduction . . . 17
- 2.2 Avantage des bases de données XML . . . 17
- 2.3 Stockage des contenus XML . . . 18
  - 2.3.1 Stockage comme un BLOB . . . 18
  - 2.3.2 SGBD a support relationnel . . . 18
  - 2.3.3 SGBD natif XML . . . 19
- 2.4 Exemple SGBD XML . . . 20
  - 2.4.1 eXist . . . 20
  - 2.4.2 Xindice de Apache . . . 20
  - 2.4.3 Software A.G.Tamino . . . 20
  - 2.4.4 GoXML de XML Globale . . . 20

2.4.5 Oracle XML DB . . . 20
2.4.6 IBM DB2 . . . 21
2.4.7 Microsoft SQL Server . . . 21
2.5 Conclusion . . . 21

3 Entrepôts de données XML **23**
3.1 Introduction . . . 23
3.2 Système d'analyse (OLAP) . . . 23
3.2.1 Sources et résultats en XML . . . 23
3.2.2 Système natifs XML . . . 24
3.3 Entreposage de données XML . . . 24
3.3.1 Entrepôts de données XML . . . 24
3.3.2 Entrepôts de document XML . . . 24
3.3.3 Entrepôt de données vs entrepôt de données XML . . . 24
3.4 Modélisation multidimensionnelle en XML . . . 25
3.4.1 Modélisation Conceptuelle et/ ou Logique . . . 27
3.4.2 Niveau physique . . . 37
3.4.3 Synthèse générale . . . 39

4 XStar-Warehouse(XSW) : Modélisation Conceptuelle et Logique d'un Entrepôt de Données XML **41**
4.1 Introduction . . . 41
4.2 Approche "XStar-Warehouse" de conception d'un entrepôt XML . . . 41
4.2.1 Génération de schéma XML à partir d'une source . . . 43
4.2.2 Passage de schéma XML vers diagrame de classe UML . . . 44
4.2.3 Génération de schéma multidimensionnel . . . 45
4.2.4 De l'UMLStar vers L'XML . . . 49
4.3 Formalisme . . . 51

Conclusion et perspectives **53**

A Le format de stockage de données XML(eXtensible Markup language) **55**
A.1 Introduction . . . 55
A.2 Format de stockage de données XML . . . 55
A.2.1 Définition . . . 55
A.2.2 Composants et syntaxe d"un document XML valide . . . 55
A.2.3 Document bien formé . . . 56
A.2.4 Les DTD et les schemas XML . . . 57
A.2.5 Le langage d"interrogation des documents XML . . . 58
Bibliographie . . . 60

Table des figures

1.1 Architecture d'un Système d'aide à la décision.[bs12] 7
1.2 Exemple de modèle en cube [Kec11] 12
1.3 Exemple d'un schéma en étoile [Kec11] 13
1.4 Exemple d'un schéma en flocon [Kec11] 14
1.5 Exemple d'un schéma en constellation [Kec11] 14

3.1 Un DTD où les relations sont définit par sous-éléments [GRV01] 27
3.2 Graphe de DTD [GRV01] 28
3.3 Arbre d'attribut DTD [GRV01] 29
3.4 Modèle XML-Star [Pok01] 30
3.5 Structure d'un cube XCubeSchema[HBH03] 31
3.6 Structure d'un document XML de Dimension XCube[HBH03] 32
3.7 Structure d'un document XML Fait XCube[HBH03] 32
3.8 Une vue d'ensemble de X-WAREHOUSE[MJ04] 33
3.9 Les étapes de l'approche X-Warehousing[BMCA06] 35
3.10 Exemple d'un modèle conceptuel du cube de données[BMCA06] 35
3.11 Description d'un cube par un schéma en étoile XML[BMCA06] 36
3.12 Architecture générale DAWAX[XZ03] 37
3.13 Exemple de requête XQuery de création de dimension temps[RRT04] 38
3.14 Modèle multidimensionnel d'entrepôt XML-OLAP[PHS05] 39

4.1 Processus de conception de l'entrepôt de données XML : XStar-Warehouse 43
4.2 Le passage d'un schéma XML vers un diagramme de classe UML. 47
4.3 Masures potentielles. 47
4.4 Dimension potentielle d'un fait. 48
4.5 Diagramme UML Star. 49
4.6 Le schéma XML en étoile est généré. 52

A.1 Exemple d'un document XML 56

Liste des tableaux

1.1 SID vs transactionnels : la différence par les données. 6
1.2 SID vs transactionnels : la différence par l'usage 7
1.3 Comparaison entre les entrepôts de données et les bases de données.. 11

2.1 La différence entre les bases de données relationnelles,XML et documentaires . . 17

3.1 Comparaison entre entrepôt de données et entrepôt de données XML 25
3.2 Les travaux sur la modélisation multidimensionnelle des données XML 26
3.3 Comparaison des travaux d'entreposage XML. 40

4.1 les règles de passages de schéma XML vers UML 45

Introduction

Problématique et objectifs

En raison de la mondialisation, les entreprise doivent être en mesure de livrer une vive concurrence sur les marchés mondiaux. Elles doivent être ouvertes à de nouvelles technologies, pour leurs permettre de prendre des décisions pertinentes, de façon rapide. La mise en place d'un processus décisionnel est alors nécessaire pour gérer une masse de données de plus en plus conséquente.

Le stockage et la centralisation de ces données se fait dans un entrepôt de données(data warehouse), celui-ci peut être défini comme " une collection de données orientées sujet, intégrées, non volatiles, historiées et organisées pour supporter un processus d'aide à la décision" [Inm96].

En plus de son objectif de stockage, un entrepôt vise aussi l'exploitation des données dans un processus d'analyse et d'aide à la décision. Des structurations particulières, telles que le schéma en étoile ou le schéma en flocon de neige, ont été afin de rendre les données d'un entrepôt prêtes à l'analyse. Ces structurations offrent la possibilité de créer des vues multidimensionnelles des données appelées aussi : les cubes de données, ces cubes sont très adaptés à des manœuvres d'analyses souples et rapides assurées par des opérateurs d'analyse en ligne OLAP(On-Line Analysis Processing).

Cependant, les données sources sont souvent complexes et hétérogènes car elles ont été définies indépendamment les unes des autres et avec des sémantiques variables. Ces données sont également représentées dans différents formats. Elles peuvent être structurées par les bases de données classiques, semi-structurées, ou encore non structurées tels que les images, les vidéo. Le défi de conception d'un entrepôt de sources de données est de faire cohabiter ces sources hétérogènes, réparties et nombreuses. Pour exploiter de telles données à des fins décisionnelles, il est nécessaire de les structurer et de les homogénéiser. Le langage XML (eXtensible Markup Language) s'avère comme une solution appropriée. Grâce à ça structure souple et générique, il permet de représenter la plupart des structures et sources de données. Ces dernières sont alors stockées dans des documents XML validés et formés conformément selon une DTD(document type definition) ou un schéma XML.

Les schémas XML se distinguent des DTD par leurs types de données de base utilisables dans les DTD (PCDATA, ANY, EMPTY) ont été enrichis (entiers, réels, chaines de caractères, date,...). Ils est possible de définir précisément le nombre d'occurrences d'un élément au sein d'un document XML. En particulier, un utilisateur peut, avec un schéma XML, définir de nouveaux types existants. Ce qui est une caractéristique de l'approche objet, par exemple UML (Unified Modeling Language). Ce dernier est un langage de modélisation objet qui offre la possibilité de

spécifier et de visualiser les systèmes logiciels et ou matériels selon différents aspects par le biais de différents diagrammes.

Concevoir un entrepôt de données XML représente un enjeu important pour la communauté des bases de données. Cette conception doit concerner les trois niveaux de la modélisation :(1) *conceptuelle* qui consiste à trouver le schéma adapté. (2)*logique* schéma en étoile, flocon de neige, etc. (3)*physique* qui consiste à proposer des techniques d'optimisation adaptées aux données XML.

Dans un système d'entreposage de données XML, l'hétérogénéité des documents XML sources, se situe à deux niveaux : de *la structure* et de *la sémantique*, un même attribut peut avoir deux types différents selon la source. Pour la sémantique, la même information peut être représentée par des identificateurs différents, un nom d'attribut peut avoir différentes significations dans deux schémas sources différentes.

Le moyen possible de faire face à ces problèmes de l'hétérogénéité des documents XML en entrées, soit d'utiliser un schéma existant, ou de traiter chaque type de source et identifier sa propre schéma, ou de définir un schéma intermédiaire global unifiant toute les schémas sources. Nous distingons deux types d'entrés pour modéliser et concevoir un entrepôt de données XML :

1. L'ensemble des documents XML en entrée sont décrit et validé par un seul schéma global(xsd).
2. Plusieurs schémas (XSDs)décrivant chacun un document XML. Dans ce cas, nous proposons de créer un schéma XML globale intermédiares qui est une manière d'extraire l'information nécessaire, et avoir une vue commune sur les différentes sources et une seule entrée dans le processus de modélisation.

Précédemant, nous avons précisé que notre travail s'encadre principalment aux étapes de modèlisation. Pour cela, nous avons procédé comme suivant : proposition d'un modèle orienté objet, de schéma XML globale(dans notre cas en suppose l'existence de ce schéma pour validé la solution) basé sur le diagramme de classe UML(Unified Modeling Language). Ce dernier est un language de modélisation offre la possibilité de spécifier et visualiser le schéma XML. Puis l'extraction des classes multidimensionnelles pour la génération de schéma en étoile de diagramme de classe, enfin nous organisons ce diagramme en XML schéma en étoile.

Contribution

Notre travail se concentre autour de l'entreposage de données XML à des fins d'analyse en ligne (OLAP). Dans ce domaine, quelques travaux ont été effectués sans qu'aucun standard ne soit encore fixé.
C'est pourquoi nous nous intéressons aux travaux émergents sur les entrepôt de données XML qui est pratiquement un domaine de recherche vierge.

Dans ce contexte, nous avons proposé une méthode de modélisation des entrepôts de données XML, nous avons identifié un schéma XML pour représenter les données sources de nature base de données relationnelle.
Notre approche se décompose de quatre étapes :

1. **Construction de schéma XML globale intermédiaire**

   Pour la construction d'un entrepôt de données XML, la première étape de notre approche consistent en la définition du schéma XML représentant la source(BDR) qui va transformer à XML schéma. Le choix de XML schéma est justifié par sa flexibilité, extensibilité et sa puissance. Avec l'incorporation du typage de données et la gestion des espaces de nommage (namespaces).

2. **Génération de schéma XML à partir d'une source**

   Pour la construction d'un entrepôt de données XML, la première étape de notre approche consistent en la définition du schéma XML représentant la source(BDR) qui va transformer à XML schéma. Le choix de XML schéma est justifié par sa flexibilité, extensibilité et sa puissance. Avec l'incorporation du typage de données et la gestion des espaces de nommage (namespaces).

3. **Passage de schéma XML vers diagramme de classe UML**

   Les XSD générés lors de l'étape précédente est complexe pour comprendre la structure des données des sources XML. Il est difficile d'avoir une compréhension rapide et globale d'un vocabulaire XML basé sur un schéma XML. C'est pourquoi, il est intéressant de représenter ce vocabulaire en UML. Ce dernier avec sa notation graphique permet d' exprimer visuellement une solution objet.

4. **Génération de schéma multidimensionnel**

   Nous avons présenté un ensemble de règles pour la conversion de diagramme UML, généré à partir de source XML en un ou plusieurs diagrammes UML étoile. Nous avons proposé des règles pour l'extraction des propriétés multidimensionnelles (faits,dimension ,...), puis les représentées en un schéma en étoile.

5. **Génération de schéma XML en étoile**

   Nous avons présenté un ensemble des règles de transformation des entités du diagramme de classe UML (telles que les classes, les attributs, les associations, etc.) en leurs équivalents dans XML schéma. Contrairement à d'autres travaux qui ne s'intéressent qu'a une partie.

Organisation du mémoire

Nous avons retenu pour ce mémoire une organisation en quatre chapitres :

Le premier chapitre traite les entrepôts de données, nous analyserons les notions nécessaires à leur compréhension et nous étudierons ensuite les différents concepts gravitant de l'entrepôt de données.

Le deuxième chapitre consiste en une analyse des différents types de bases de données permettant de gérer des documents XML.

Le troisième chapitre, le noyau du mémoire, nous permettra de présenter une synthèse des différents domaines de recherche liés à nos contribution, à savoir des travaux de recherche autour de la conception d'un datawarehouse XML.

Le quatrième chapitre, présente notre proposition de conception d'un entrepôt de données XML basé sur XML et UML.

Nous conclurons notre travail par un récapitulatif sur ce que nous avons pu résumer sur la conception de l'entrepôt de données XML et présente quelques possibilités d'améliorations futures.

Chapitre 1

Les Entrepôts de Données

1.1 Introduction :

En raison de la mondialisation, les entreprises doivent être en mesure de livrer une vivre concurrence sur les marchés mondiaux, qui est de plus en plus complexes changeants. Les entreprises sont confrontées à une concurrence de plus en plus forte, des clients de plus en plus exigeants, pour cela les entreprises doivent s'appuyer sur des informations pertinentes, pour être efficaces et faire face aux nouveaux enjeux économiques.

Mais souvent, les données dans les entreprises sont éparpillées dans de multiples systèmes hétérogènes et ne sont pas organisées dans une vision décisionnelle. Il est alors indispensable de les homogénéiser et de les regrouper afin de faciliter la prise de décision. Cela ne peut ce faire sans la définition d'une architecture qui serve de fondations aux applications décisionnelles. D'où le nouveau concept « d'informatique décisionnelle».

Dans ce chapitre, nous nous attacherons a présenter les différents concepts liés aux entrepôts, que ce soit au niveau de la modélisation, de la stratégie de conception, de l'exploitation, etc.

1.2 SID vs transactionnels

L'informatique décisionnelle est un secteur en plein développement, la demande de systémes d'aide à la décision (SAD) et de systémes interactifs d'aide à la décision(SIAD) est de plus en plus forte, au vu de la croissance exponentielle des données manipulées par les entreprises.

1.2.1 Système d'information (SI) :

Le système d'information transactionnel est composé de l'ensemble des composants et applications métiers de l'organisation qui gèrent les données au quotidien. Il assure la gestion de toutes les transactions qui ont lieu au sein de l'organisation. Cette deuxième brique intègre la dimension métier de l'organisation. Ces systèmes ne sont pas adaptes des analyses complexes de données [Cod93] et ne préparent pas les données pour la prise de décision. Pour assurer une plus grande réactivité et une plus grande compétitivité. Les approches traditionnelles s'avérer donc rapidement insuffisantes et les décideurs requièrent des systèmes qui facilitent leur processus de prise de décision. Il système d'information pourrait être défini comme suit :

« Un système d'information est l'ensemble des méthodes et moyens recueillant, contrôlant, et distribuant les informations nécessaires à l'exercice de l'activité en tout point de l'organisation. Sa fonction et de produire et de mémoriser les informations, représentation de l'activité du systéme opérant (systéme opérationnel), puis de les mettre à disposition du systéme de décision (systéme de pilotage)» [Moi77].

1.2.2 Système d'information décisionnel(SID)

Le Systéme d'information décisionnel (SID) est un ensemble organisé d'information facilement accessible, qui adapté à la prise de décision. Il est obtenu à partir de traitements sur les données de systémes d'information transactionnels internes ou externes à l'organisation. Il pourrait être défini comme suit :

« Un système d'information décisionnel(SID) est un système qui réalise la collecte, la transformation des données brutes issues de sources de données et le stockage dans d'autre espace ainsi que la caractérisation des données résumées en vue de faciliter le processus de prise de décision ».

Ce qui caractérise les systèmes d'information décisionnel, est la possibilité de poser une grande variété de questions au système, certaines prévisibles et planifiées comme des tableaux de bord et d'autres imprévisibles, et de permettre à l'utilisation d'effectuer les requêtes qu'il souhaite, par lui-même, sans l'intervention de programmeurs. Ils doivent assurer une cohérence globale des données. Pour ce faire, leur alimentation doit être une opération réfléchie et planifiée dans le temps. Les transferts de données du système opérationnel vers le système décisionnel seront réguliers avec une périodicité bien choisie et dépendante de l'activité de l'entreprise. A l'inverse des systèmes transactionnels, aucune information n'y est jamais modifiée.

Les deux tableaux 1.1,1.2suivants montrent la différence entre les systèmes transactionnels et ceux décisionnels.

Système transactionnels	***SID***
Orienté applications	Orienté thèmes et sujets
Situation instantanée	Situation historique
Données détaillées et codées non redondantes	Information agrégées cohérentes souvent avec redondance
Données changeant constamment	Information stable et synchronisées dans le temps
Pas de référentiel commun	Un référentiel unique

Table 1.1 – SID vs transactionnels : la différence par les données.

Système transactionnels	SID
Assure l'activité au quotidien	Permet analyse et prise de décision
Pour les opérationnels	Pour les décideurs
Mise à jour et requêtes simples	Lecture uniquement et requêtes complexes transparentes
Temps de réponse immédiats	Temps de réponse moins critiques
Faible volume à chaque transaction	Large volume manipulé
Conçue pour la mise à jour	Conçue pour l'extraction
Usage maitrisé	Usage aléatoire

TABLE 1.2 – SID vs transactionnels : la différence par l'usage

En résume contrairement au système d'information classique où les données opérationnelles sont souvent disséminées dans l'entreprise dans des systèmes hétérogènes, le SID se construit à partir de ces données. Il récupère ces données provenant de ces sources et les prépare pour le système de décision, pour ce faire, le système d'Information Décisionnel va remplir trois fonctions : extraction des données, leurs stockages et la restitution des données sous une forme exploitable. Le stockage sert à structurer les données au sein d'un entrepôt de données. Il s'agit de mettre en place un schéma relationnel orienté décisionnel, ce dernier constitue la partie la plus importante et la plus complexe du processus de développement des SID. C'est la raison pour laquelle, nous nous intéressons à cette partie du système d'information décisionnel.

1.2.3 Architecture des systèmes décisionnels

Afin de mieux comprendre la finalité des systèmes décisionnels, nous devons les placer dans leurs contextes.

Dans la figure 1.1 nous présentons une architecture simplifiée d'un système d'aide à la décision. Les différents composants ont été intégrés dans quatre parties : sources de données, outils ETL, outils de stockage et outils de restitution et d'analyse.

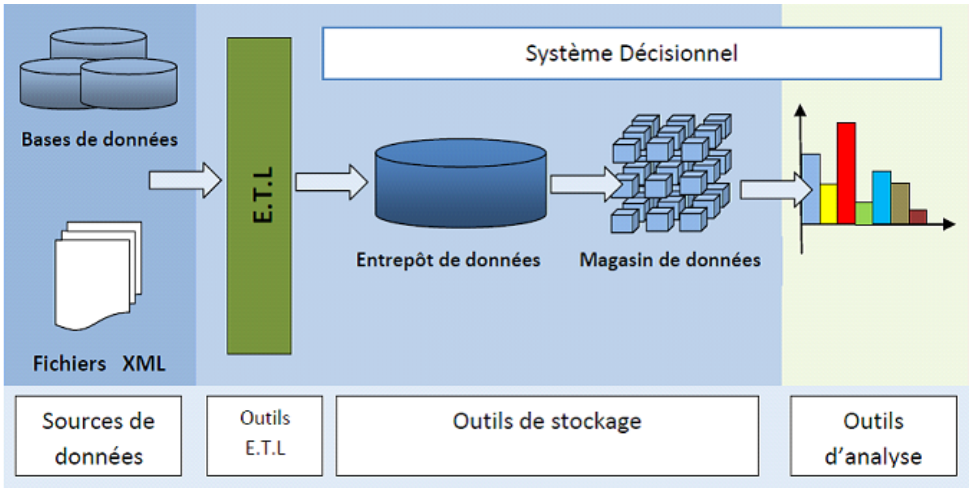


FIGURE 1.1 – Architecture d'un Système d'aide à la décision.[bs12]

- ***Sources de données :*** C'est l'ensemble des sources de données qui contiennent les informations nécessaires pour l'alimentation de l'entrepôt de données, elles sont souvent réparties et hétérogènes. Leur provenance peut être interne (bases de production, fichiers ...) ou externe (Internet, bases des partenaires ...) à l'entreprise.

- ***Outils ETL (Extraction, Transformation et Chargement) :*** Outre le fait que les sources de données utilisées soient hétérogènes et diffusées, elles contiennent des données qui ne seront pas utilisées par l'entrepôt. Il est donc nécessaire de disposer d'outils performants et rapides permettant de :

  - ***Extraire les données (Extract) :*** Des applications et des bases de données de production (SGBDR, fichiers ...).

  - ***Transformer (Transform) :*** Toutes les données ne sont pas utilisables telles qu'elles. Elles doivent être vérifiées, reformatées, nettoyées pour respecter le format requis par le système cible.

  - ***Charger (Load) :*** C'est la phase la moins complexe, elle consiste à Insérer les données normalisées dans l'entrepôt ou le magasin.

- ***Outils de stockage :*** Un système décisionnel repose sur deux catégories d'espaces de stockage [RT99] : l'entrepôt de données et les magasins de données.

- ***Outils de restitution et d'analyse :*** Ces derniers mettent à la disposition des utilisateurs (souvent non informaticiens) les données selon des axes d'analyses. L'objectif prioritaire est de segmenter les données en contextes informationnels fortement cohérents, simples à utiliser et correspondant à une activité décisionnelle particulière.

1.3 Outils De Stockage pour l'aide à la décision

Les données du système décisionnel sont organisées et stockées dans l'une des deux structures suivantes :

- Les entrepôts de données (datawarehouse).
- Les magasins de données (datamarts).

1.3.1 Les entrepôts de données *(Datawarehouse)*

1.3.1.1 Définition

Un entrepôt de données est l'espace de stockage centralisé d'un extrait des sources de données pertinentes pour les décideurs. Son organisation doit faciliter la gestion des données sous la forme d'une vision unifiée et doit permettre la conservation des évolutions nécessaires pour les prises de décisions.

Un entrepôt de données est vu comme [Kim96] : ***"Une copie de données transactionnelles spécifiquement structurée pour l'interrogation et l'analyse".***

Par ailleurs, Bill Inmon étant l'un des premiers à avoir utilisé le terme "Data Warehouse" a défini les entrepôts de données comme étant :≪***une collection de données orientées sujet, intégrées, variant selon le temps et non volatiles, qui sert de support au processus de prise de décision des acteurs du management (les décideurs)***≫[Inm96].

1.3.1.2 Caractéristiques

La définition précédente d'ED de Bill Inmon [Inm92] englobe différentes caractéristiques d'un entrepôt de données comme suit :

- **Orientée sujet :** les données constituent des granules d'information concernant des sujets d'analyse plutôt que les opérations de gestion se déroulant au sein de l'entreprise.

- **Intégrées :** Parmi tous les aspects d'un entrepôt de données, l'intégration est la plus importante. Les données de l'entrepôt proviennent de différentes sources éventuellement hétérogènes, l'intégration consiste à résoudre les problèmes d'hétérogénéité des systèmes de stockage, des modèles de données et de sémantique de données.

- **Non volatile :** Les données d'un entrepôt sont stables. Il est possible d'ajouter des données, mais on ne modifie pas les données déjà intégrées. Il est toutefois possible de les archiver.

- **Historisées :** La prise en compte de l'évolution des données est essentielle pour la prise de décision, qui par exemple utilise des techniques de prédiction en s'appuyant sur les évolutions passées pour prévoir les évolutions futures.

- **Aide à la décision :** Par sa fonction d'aide a la décision, l'entrepôt de données est amené à effectuer des traitements qui sont différents de ceux effectués par les systèmes transactionnels. En effet, les traitements transactionnels en ligne concernent un sous ensemble de données, portant des changements dans la base de données et retournant une réponse quasi instantanée, contrairement aux traitements pour l'aide à la décision qui impliquent la lecture de nombreuses données sans apporter de changement dans la base de données, et la réponse n'est pas forcement instantanée.

1.3.1.3 Objectifs

Les objectifs d'un entrepôt de données par les points suivant [Kim02] :

- L'entrepôt de données rend l'information d'une organisation facilement accessible : les données de l'entrepôt sont compréhensibles et navigable,

- Dans L'entrepôt de données, l'information de l'organisation est consistante :
  si deux mesures dans une organisation ont le même nom, alors elles doivent représenter la même chose,

- L'entrepôt de données doit être adaptatif et résilient au changement : Lorsque de nouvelles données sont ajoutées à l'entrepôt, les données existantes et les technologies ne sont pas modifiées ou perturbées,

- L'entrepôt de données doit être un bastion sûr, qui protège les biens d'informations,

- L'entrepôt de données doit servir de base à la prise des bonnes décisions.

Ces objectifs sont déterminés à partir des soucis des gestionnaires d'entreprise concernant le volume d'informations mal géré et mal exploité pour des fin décisionnelles.

1.3.1.4 Problématiques liées aux ED

Malgré l'efficacité des entrepôts de données, ils présentent quelques problèmes, on cite parmi d'autres :

- **L'hétérogénéité** Les objets de l'univers sont des données complexes [Bou09] ; l'objectif est d'adapter les ED à ces données complexes (comme le cas des données à référence spatiale) qualifiées comme :

  - ***Multiformes :*** des données qui n'ont pas le même format (par exemple les données sont de format raster dans une source et de format vecteur dans une autre).
  - ***Multi structures :*** les données différent dans leurs types, leurs unités,leurs granularité. . ..etc.
  - ***Multi sources :*** telles que les données provenant d'internet et qui sont de natures différentes.
  - ***Multi modale :*** les données contenant plusieurs facettes de présentation hétérogènes.
  - ***Multi versions :*** les données qui changent dans le temps (évolution temporelle).

- **La mise à jour** Plusieurs temps orthogonaux coexistent [PB04], nous citons :

  - **Le temps de validité** correspond au temps pendant lequel l'information est considérée valide dans la réalité.
  - **Le temps d'entreposage** est le temps pendant lequel l'information est active dans la réalité.
  - **La série temporelle** des temps d'extraction correspond à la série des temps de demande d'information de l'entrepôt (mise à jour).
  - **Le rafraichissement des données de l'entrepôt** est abordé selon deux méthodes spécifiques : le rafraichissement périodique et le rafraichissement incrémental.

- **L'interopérabilité** La problématique d'interopérabilité est liée au transfert des mises à jour, cela peut entrainer des difficultés [PB04] telles que :

  - Grande quantité de données à transmettre ,
  - Pas de liens avec les anciennes données donc pas de suivi de l'historique possible ,
  - Données précédentes entièrement remplacées chez l'utilisateur (Perte des liens, Pertedes données ajoutées).

1.3.2 Magasin de données(*DATA MART*)

Un magasin de données ou DATA MART est un référentiel de données recueillies à partir des données opérationnelles et d'autres sources, conçu pour servir les décideurs. Les données peuvent provenir d'un entrepôt ou d'une base de données au niveau de l'entreprise. Le rôle d'un DATA MART est de répondre aux exigences spécifiques des décideurs en termes d'analyse, de contenu et de présentation.

1.4 Entrepôt de données vs Base de données

Les SGBD (Système de gestion de base de données) sont utilisés pour la gestion de volumes importants de données se trouvant dans les différents systèmes opérationnels au sein de l'entreprise. Ces données sont manipulées en utilisant des processus transactionnels en ligne permettant des transactions en temps-réel et des traitements factuels tel que, l'ajout, la suppression, la modification [Cod93]. Par ailleurs, les entrepôts de données ont été conçus pour l'aide à la prise de décision. Les bases de données de productions s'opposent aux entrepôts de données sur plusieurs aspects [CF07].

Le tableau 1.3comparatif ci-dessous résume ces différences de caractéristiques [DM98] :

	Bases de données	***Entrepôt de données***
Objectifs	Gestion de la production	Consultation et analyse
Utilisateurs	Gestionnaires de productions Nombreux Accès restreint aux informations	Décideurs, analystes Peu Accès a de nombreuses informations
Requêtes	Prédéfinies Réponses immédiates	Imprévisibles Réponses moins rapides
Données	Exhaustives, détaillées Orientées applications Mises à jour De l'ordre des gigaoctet	Résumées, agrégées Orientées sujets d'analyse Recalculées De l'ordre des téraoctets

TABLE 1.3 – Comparaison entre les entrepôts de données et les bases de données..

1.5 Modélisation multidimensionnelle

Les modèles de données utilisés pour concevoir des systèmes transactionnels traditionnels OLTP [Che76], ne sont pas bien adaptés pour les systèmes décisionnels. En effet, les transactions dans les systèmes transactionnels, sont constituées des requêtes simples et prédéfinies. Par contre, dans un environnement d'entrepôt de données, les requêtes sont plus complexes, utilisent des opérations de jointure et des fonctions d'agrégation, et ont un temps de traitement plus élevé. Pour ce type d'environnement OLAP, une nouvelle approche de modélisation a été proposée :***la modélisation multidimensionnelle***.

1.5.1 Concepts de bases

Le modèle multidimensionnel, parfois désigné sous le nom de cube de données, s'est révélé être un modèle satisfaisant. Le cube correspond au sujet d'analyse appelé fait, les cellules du cube contiennent des mesures décrivant le fait. Les axes du cube appelés dimensions, représente les différents façons d'analyser les données, Chaque dimension est liée à un ou plusieurs points de vue définissant ainsi le degré de granularité des données (hiérarchies). Dans se qui suit nous allons définir ces concepts de base.

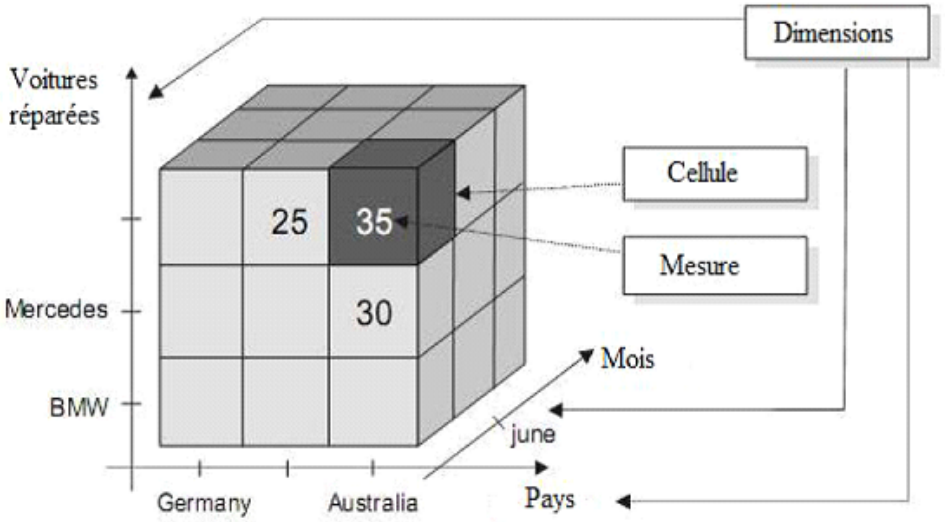


FIGURE 1.2 – Exemple de modèle en cube [Kec11]

- **Fait :** Un fait représente un sujet d'analyse et est caractérisé par une ou plusieurs mesures (ou indicateurs). Les mesures d'un fait sont :

  – Numériques pour permettre de résumer un grand nombre d'enregistrements en quelques enregistrements (on peut les additionner, les dénombrer ou bien calculer le minimum, le maximum ou la moyenne),
  – Valorisées de façon continue car il est important de ne pas valoriser le fait avec des valeurs nulles.

- **Dimension et hiérarchie :** Une dimension est définie comme un axe d'observation selon lequel un fait est observé. Une dimension peut comporter plusieurs hiérarchies impliquant différents niveaux de précision dans la description des faits. Ces hiérarchies permettent d'observer des mesures selon plusieurs niveaux de granularité et de construire des agrégats à partir des faits du niveau le plus fin.

1.5.2 Niveau conceptuel

À ce jour, aucun consensus n'existe sur la modélisation conceptuelle des entrepôts de données comme cela peut être le cas avec la méthode MERISE pour la conception des bases de données relationnelles. Différents travaux ont été proposés dans la littérature [Riz06]. Les auteurs, classent les modèles multidimensionnels en deux classes : les modèles qui étendent les modèles existant (Entité-Association, Orienté Objet), et les modèle spécifiquement multidimensionnels. Ces modèles permettent de représenter tous les concepts de base de la modélisation multidimensionnelle, ils diffèrent dans la possibilité de représenter des notions plus avancées telles que les

hiérarchies, la cardinalité plusieurs-vers-plusieurs...etc.

- **Extension des modèles existants :**

– ***Entité-Association.*** Différents travaux étendent le paradigme Entité-Association [TBC99] et [SHD98]. Les modèles emploient le même formalisme et reposent sur des relations d'associations entre sujets (faits) et axes d'analyse (dimensions). Une dimension est représentée par un ensemble s'entités. Chacune correspond un niveau hiérarchique et les différents niveaux sont reliés par des associations binaires. Les faits correspondent à des associations entres les différentes dimensions.
– ***Orienté-Objet.*** Ces approches se basent sur les modèles orientés objet, comme UML (Unifed Modeling Language), les travaux dans [TP98],[Tru01], proposent le model GOLD qui est une extension du diagramme de classe UML. Dans [Mor05], les auteurs proposent un profil (ensemble de stéréotypes, de valeurs typées et de contraintes) pour spécialiser UML à la modélisation multidimensionnelle. Le fait est représenté par une classe avec le stéréotype Fact. Une dimension est représentée par une classe avec le stéréotype Dimension.

- **Modèles spécifiques :**

Les modèles purement multidimensionnels se basent sur les concepts de fait et de dimension. La simplicité des notations et des concepts est le point fort de ces modèles. Dans cette catégorie de modèles, nous citons les travaux de [GMR98a] et [PJ99]. Cependant, les modèles spécifiques ne sont pas connus des concepteurs contrairement à ceux basés sur le modèle E/A et UML, ce qui constitue un obstacle.

1.5.3 Niveau logique

1.5.3.1 Modèles en étoile

Ce modèle représente visuellement une étoile, on parle de modèle en étoile (Star Schéma [Kim96]) où tous les faits sont définis dans une simple table relationnelle. Cette table va être reliée par sa clé primaire à d'autres tables correspondant aux dimensions. Comme illustré par la figure 1.3 ci-dessous. La table de faits Vente est jointe à un ensemble de dimensions Temps, Catégorie, Géographie. Le schéma en étoile est à la base des deux modèles suivants.

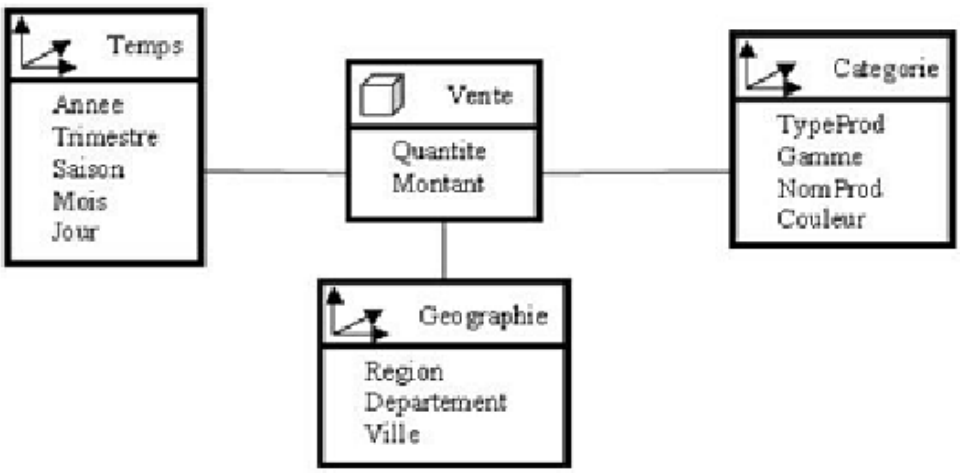


FIGURE 1.3 – Exemple d'un schéma en étoile [Kec11]

1.5.3.2 Modélisation en flocon de neige (snowflake)

Le modèle en flocon de neige, consiste à décomposer les dimensions dénormalisées du modèle étoile en sous hiérarchies. Les attributs redondants des dimensions sont supprimés et placés dans des tables secondaires normalisées. La figure 1.4 suivante illustre la modélisation en flocon.

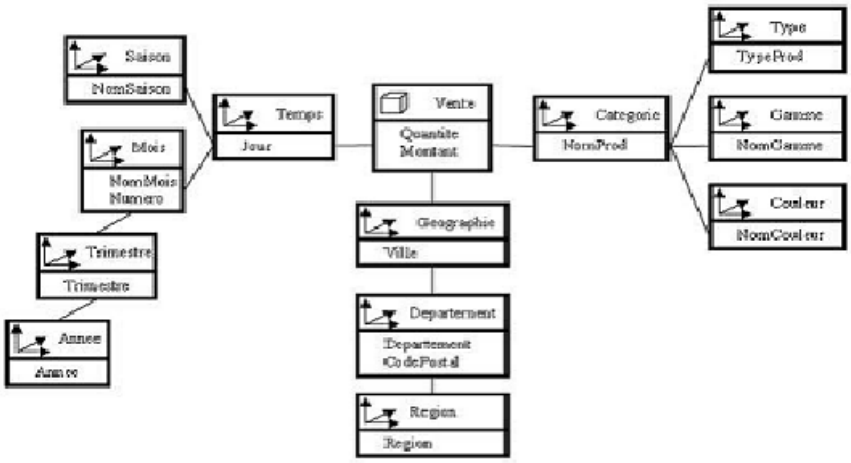
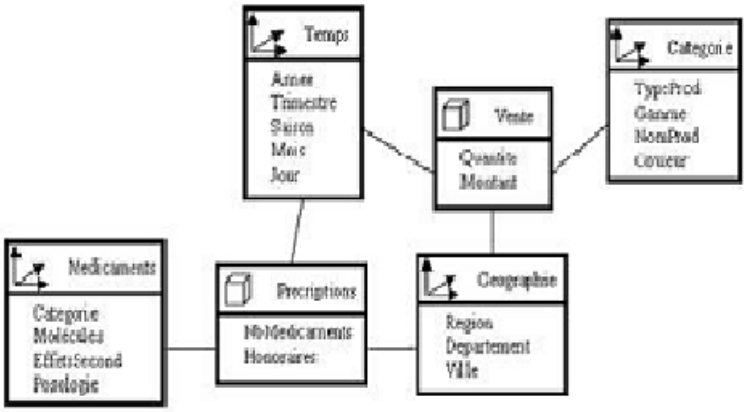


FIGURE 1.4 – Exemple d'un schéma en flocon [Kec11]

1.5.3.3 Modélisation en constellation

Il s'agit de fusionner plusieurs modèles en étoile qui utilisent des dimensions communes. Un modèle en constellation comprend donc plusieurs faits et des dimensions communes ou non.

FIGURE 1.5 – Exemple d'un schéma en constellation [Kec11]



1.5.4 Niveau physique

Le stockage de données et les méthodes d'accès associées(les techniques d'indexation) ont été étudiés depuis plusieurs années. Cependant le gros volume géré par les entrepôts et les besoins d'accès performants de l'interrogation ont remis en cause les techniques traditionnelles. Dans cette section nous présentons quelque techniques de stockage et d'accès aux données adaptées aux entrepôts.

1.5.4.1 Le stockage

Le stockage des modèles multidimensionnels(les cubes)est réduit au problème de stockage d'un vecteur multidimensionnel où chaque dimension de cube est associées à un axe du vecteur et chaque valeur d'une dimension est associée a une position de l'axe correspondant) la dimension. Les cellules du vecteur i.e. les intersections entre les positions des axes contiennent les mesures. Le problème principal posé par la représentation d'un cube sous forme d'un vecteur multidimensionnel est que celui-ci est creux. Ceci est du au fait qu'en général seulement un nombre réduit de cellules d'un cube ont une valeur de mesure associée.

1.5.4.2 L'indexation

Les techniques d'indexation utilisées dans les base de données OLTP ne sont pas bien adaptées aux environnements des entrepôts de données. En effet la plupart des transactions OLTP accèdent) un petit nombre de n-uplets techniques utilisées(index B+ par exemple) sont adaptées à ce type de situation. Les requêtes décisionnelles adressées à un entrepôt de données accèdent au contraire à un très grand nombre de n-uplets. les différents types d'index utilisés dans les entrepôts de données sont :index binaire, index de jointure et index de jointure en étoile.

1.5.4.3 Le partitionnement

Les entrepôts de données se prêtent bien à l'utilisation des techniques de partitionnement, car des requêtes complexes sur des grandes quantités de données peuvent être exécutées en parallèle améliorant éventuellement les temps de réponse.
Les travaux récent dans le contexte des entrepôts de données visent au partitionnement vertical de la relation de fait d'un schéma en étoile. Dans le système multidimensionnel, il est possible de distinguer : La fragmentation horizontale se décline en deux versions : les fragmentation primaire et dérivées.
La fragmentation primaire d'une relation est effectuée grâce à des prédicats de sélection définis sur une autre relation. La fragmentation horizontale dérivée s'effectuée avec des prédicats de sélection définis sur une autre relation.

1.6 Approches de conception des entrepôts de données

L'entreposage de données repose sur la conception du schéma, car c'est le schéma qui conditionne les possibilités d'analyse[CF07]. Il existe trois stratégies pour la conception d'entrepôts de données :

- L'approche ascendante (pilotée par les données).
- L'approche descendante (pilotée par les besoins d'analyse).
- L'approche hybride qui combine les deux approches précédentes.

1.6.1 Approche ascendante

Cette approche est proposée dans[Inm96], elle repose sur les données et ne s'intéresse par aux besoins d'analyse en premier lieu. En effet, la construction du schéma de l'entrepôt repose sur l'hypothèse que toutes les requêtes d'analyses pourront être prises en charge par cet entrepôt. Une fois créé, l'entrepôt contiendra des données structurées dimensionnelles (magasins de données) mais aussi des données non dimensionnelles (documents web, tableurs...). Dans cette approche, l'entrepôt a pour fonction la Centralisation des données.

1.6.2 L'approche descendante

Cette approche est proposée par Kimball, elle est guidée par les besoins d'analyse. Elle propose un schéma de l'entrepôt en fonction des besoins d'analyse et suppose que les données disponibles permettront la construction d'un tel schéma [CF07].

Avant la modélisation de l'entrepôt, les besoins d'analyses doivent être définis. Une fois mis en œuvre, l'entrepôt de données apparait au niveau logique comme une collection de magasins de données dimensionnels liés entre eux.

1.6.3 L'approche hybride (ou mixte)

Cette dernière, considère à la fois les besoins d'analyse et les données pour la construction du schéma. L'idée générale est de construire des schémas candidats à partir des données (démarche ascendante) et de les confronter aux schémas définis selon les besoins (démarche descendante). Ainsi, le schéma construit constitue une réponse aux besoins réels d'analyse et il est également possible de le mettre en œuvre avec les sources de données [CF07].

1.7 Conclusion

Les entrepôts de données et leurs outils deviennent indispensables dans le monde des entreprises. De nombreuse sources d'informations et de données semi-structurées particulièrement XML sont échangées et existent dans des systèmes de production.

Dans notre travail, nous allons étudier ce que XML peut apporter à ce domaine. Pour ce faire, dans les chapitres suivants, nous allons introduire la notion des bases de données XML, puis les travaux de recherches sur la conception des entrepôts de données XML.

Chapitre 2

Base de données XML

2.1 Introduction

Les bases de données XML se développent selon plusieurs directions ; celles permettant de manipuler des documents XML en bases relationnelles ou objets et celles étant des bases natives XML. Dans ce chapitre nous avons étudié les différents SGBD XML dont le leader du marché Oracle. De toutes façons il faut un langage de requêtes pour interroger les collections de documents XML, XQuery de W3C est bien parti pour devenir le standars.

Les bases de données XML se situent à la croisée des chemins des systèmes documentaires et des bases de données.

Bases de données relationnelles	***Bases de données XML***	***Bases de données documentaires***
Absence de sémantique	Il permet la séparation nette entre la sémantique et les données	Mélange de sémantique et données
Recherche définie par la structure de stockage	Recherche structurée et plein texte,multicritères par indexation	Recherche plein texte multicritères par indexation

TABLE 2.1 – La différence entre les bases de données relationnelles,XML et documentaires .

2.2 Avantage des bases de données XML

Un grand nombre de données commerciales sont stocker dans les base de données relationnelles, plusieurs raisons justifient leur exportation vers XML :

- Le partages de ces données avec d'autre système.
- L'interopérabilité avec des systèmes incompatibles.
- L'utilisation de donnée réelles par des applications faisant appel à XML.
- Les transactions entre entreprises.
- La pérennité de objets grâce à XML.
- Le regroupement de données éparses.

2.3 Stockage des contenus XML

Plusieurs solutions de stockage plus ou moins orientées document ou données existent. Pour exploiter les documents XML, une solution de stockage devient indispensable. Les entreprises doivent choisir entre plusieurs techniques, les plus importante, les bases native XML et les relationnelles compatibles XML [Bou03].

- *Stockage sans décomposition en BLOB :* il s'agit du stockage du document sous forme de données « données brutes » dans un SGBD classique ou dans un système de fichiers, un peu comme les fichiers HTML sur le web. Si le stockage et le chargement permet de retrouver intégralement le document, il est très difficile de faire des requêtes dessus.
- *Stockage sous forme décomposée relationnelle :*il s'agit de faire correspondre certains ou tous les attributs d'un document XML à des tables relationnelles existantes. Le stockage dans ce cas est orienté données.
- *Stockage dans une base de données objets :*il s'agit de stocker un document XML comme des objets persistants.
- *Stockage brut dans une base de données native semi-structurée :*dans ce cas un SGBD est dédié au stockage et aux requêtes sur des données semi-structurées.

2.3.1 Stockage comme un BLOB

Un BLOB (Binary Large Object) est un terme provenant du monde des bases de données et signifiant une séquence d'octets représentant des données. Le stockage des documents comme des BLOB(s) dans une base relationnelle apporte certains avantages propres aux bases de données [Bou03], tel que le contrôle de transaction, sécurité et l'accès multi utilisateur. En outre, de nombreuses bases de données relationnelles possèdent des outils de recherches compatibles avec XML, ce qui élimine les problèmes liés à la recherche des documents XML en tant qu'un simple texte.

Le stockage des documents XML sous forme de BLOBs permet au développeur d'implémenter facilement son propre indexation, même si la base de données ne sait pas indexer du XML, l'une des méthodes de réaliser cela, consiste à créer deux tables, une table d'index et une table des documents. La table des documents contient une clé primaire et une colonne BLOB où le document est stocké. La table d'index contient une colonne contenant la valeur à indexer et une clé étrangère pointant sur la clé primaire de la table des documents.

2.3.2 SGBD a support relationnel

Ce type de SGBD utilise une base de données relationnelle pour stocker les données XML. Ce procédé est motivé par :

- La réutilisation des techniques de stockage est d'interrogation efficaces développées en relationnel.
- L'enjeu économique en raison de la nature relationnelle de la majorité des base de données existantes. On peut trouver ces systèmes dans Oracle, DB2 et Microsoft SQL Server.

2.3.2.1 Génération de schéma XML à partir de schéma relationnel et vice-versa

La génération de schéma XML à partir de schéma relationnel et vice-versa est une opération intervenant lors de la conception[Bou03], parce que la plupart des applications orientées données travaillent avec un corpus bien établi de schémas XML et de schémas de base de données. Elles

ne nécessitent pas de générer des schémas lors de l'exécution.
Pour générer un schéma relationnel à partir d'un schéma XML il convient de :

- Créer une table et une colonne clé primaire pour tout type d'élément complexe.
- Pour chaque type d'élément possédant un contenu mixte créer une table séparée dans laquelle est stockée les PCDATA ; cette table est liée à la table parente grâce à la clé primaire de celle-ci.
- Pour chaque attribut de ce type d'élément qui possède une valeur unique, et pour chaque élément fils simple présentant une seule occurrence, créer une colonne dans cette table. Si le schéma XML contient des informations concernant le type de données, affecter le type de données de la colonne au type qui lui correspond. Dans le cas contraire, affecter lui un type prédéterminer comme "Character Large Objects" CLOB ou une variable de type caractère longue VARCHAR. Si l'occurrence de l'élément fils ou de l'attribut est optionnel, attribuer à la colonne la possibilité d'y affecter des valeurs nulles.
- Pour chaque attribut possédant plusieurs valeurs et pour chaque élément fils simple présentant plusieurs occurrences, créer une table séparée pour stocker ces valeurs. Cette table est liée à la table parente grâce à la clé primaire de celle-ci.
- Pour chaque élément complexe, lier la table de l'élément parent à la table de l'élément fils à l'aide de la clé primaire de la table parente.

Pour générer un schéma XML à partir d'un schéma relationnel, il convient de :

- Créer un type d'élément par table.
- Pour chaque colonne de cette table qui ne soit pas une clé et pour la (les) colonne(s) correspondante(s) à la clé primaire ajouter un attribut au type d'éléments ou ajouter un élément fils de ce type PCDATA seul à son modèle de contenu.
- Pour chaque clé étrangère, ajouter un élément fils au contenu du modèle et traiter récursivement la table de la clé étrangère.

2.3.3 SGBD natif XML

Les bases de données XML natives sont des bases conçues spécialement pour stocker et interroger des données XML sans faire subir à ses données des transformations de formats. Comme toutes les autres bases, elles possèdent des fonctionnalités telles que les transactions, la sécurité, les accès multi-utilisateurs, un ensemble d'APIs, des langages de requête, etc. La seule différence par rapport aux autres bases, c'est qu'elles sont basées sur XML, et pas sur autre chose comme dans le cas des bases relationnelles. Les bases de données XML natives sont plus franchement utiles pour le stockage des contenus orientés document, en raison du fait qu'elles préservent des choses telles que l'ordre interne du document, les instructions de traitement, les commentaires, les sections CDATA, l'utilisation des entités[Bou03]. Il existe deux grandes catégories d'architectures de bases XML natives : les architectures basées sur le texte et celles qui sont basées sur un modèle.

- **Les bases de données XML natives basées sur le texte :** Une base de données XML native basée sur le texte stocke le XML en tant que texte. Les index sont communs à toutes les bases de données XML natives basées sur le texte. Ils permettent au moteur de recherche de naviguer facilement en tout point d'un document XML quelconque.
- **Les bases de données XML natives basées sur un modèle :**La seconde catégorie est constituée des bases XML natives basées sur un modèle. Plutôt que de stocker un document XML en tant que texte, elles construisent un modèle objet interne du document et stockent ce modèle.

2.4 Exemple SGBD XML

Dans cette section nous étudions les différents types de SGBD pour XML disponible sur le marché. Il s'agit soit de systèmes documentaires étendus soit d'extensions aux SGBD relationnels ou objets ou encore de système natifs conçus pour XML.

2.4.1 eXist

Le projet eXist [1] est une implémentation open source d'un système de gestion de bases de données XML natives, interfaçable à l'aide de XPath de XQuery et de XUpdate. Le projet a été entamé en 2000 par Wolgang Merier. Aujourd'hui eXist n'utilise plus de relationnel et fonctionne sur un système de stockage propre. EXist fournit un stockage sans schéma des documents XML dans des collections hiérarchiques. Une collection est un ensemble qui peut contenir d'autres collections ou documents XML. En utilisant une syntaxe étendue d'XPath et d'XAuery, les utilisateurs peuvent interroger différences parties de la hiérarchie de collections, ou tous les documents contenus dans la base de données.

2.4.2 Xindice de Apache

Xindice [2] est un serveur de base de données open source pour stocker et retrouver des documents XML ; Xindice stocke les documents en natif sans faire de mapping. Les documents sont stockés dans des collections qui peuvent être imbriquées, la collection racine portant le nom db. Le langage d'interrogation XPath ; il gère des index sur les éléments et les attributs. Mise a jour des documents stockés à l'aide de la commande XUpdate.

2.4.3 Software A.G.Tamino

Tamino [3] est un serveur connu pour son SGBD Adabs, qui est un SGBD natif XML. Il propose un moteur spécialisé. XML Engine pour gérer du XML avec un modèle de stockage particulier. Tamino propose un module pour intégrer des données issues des tables relationnelle avec les documents natifs appelé X-Node.

2.4.4 GoXML de XML Globale

GoXML [4]est un SGBD natif supportant XQuery. Il gere volumes de documents XML. Il support le XML schéma et des extensions XQuery pour les opérations de mise à jours de type XUpdate. Il permet une indexation en plein texte st la recherche par mots clés.

2.4.5 Oracle XML DB

Depuis la version 2 d'Oracle [5] 9i un type de données XMLType a été rajouté. Ce type stocke les fragments XML soit comme un LOB(LongObjet) soit comme des colonnes de tables objets relationnels. Avec Oracle 11g les données XML sont stockées et manipulées nativement. De même la base supporte les interfaces standard XQuery, JvaSpecificationRquests(JSR)et SQL/XML.

1. http://exist.sourceforge.net
2. http://www.apache.org
3. http://www.softwareage.com
4. http://exist.xmlglobal.com
5. http://www.oracle.com

2.4.6 IBM DB2

Avec DB2[6] il est possible d'utiliser Text Extender pour stocker un document XML, utiliser le stockage en colonne avec CLOB (Character Long Objet) on le stockage en collection par un mapping entre un document XML et une collection de tables.

2.4.7 Microsoft SQL Server

Les instances XML sont stockées dans les colonnes de type XML sous forme de BLOB(Binary Large Objets volumineux binaires). Mais avec la nouvelle version SQL serveur 2008[7], les données sont stockées dans une représentation interne qui conserve le contenu XML des données. Cette représentation interne inclut des informations à propos de la hiérarchie de relations contenant–contenu. l'ordre des documents et les valeurs d'éléments et d'attributs. Plus précisément le contenu InfoSet des données XML est préservé.

2.5 Conclusion

Dans ce chapitre nous avons présenté les différents types de systèmes de gestion de bases de données permettant de gérer des documents XML en détaillant principalement les systèmes de stockage centrés données. sous forme BLOB, dans un SGBD Relationnel ou Objet, ou dans un SGBD natif semi-structuré. Le stockage par BLOB convient à un stockage axé texte, c'est-à-dire de type documentaire. Le stockage SGDB Relationnel convient plutôt à un stockage axé données et le stockage dans un SGBD natif peut convenir aux deux types de stockage. Enfin, nous avons présenté quelques SGBD XML.

6. http://www.ibm.com
7. http://www.microsoft.com

Chapitre 3

Entrepôts de données XML

3.1 Introduction

Dans ce chapitre, nous allons présenté en détailles les différentes approches de modélisation multidimensionnelle des données XML proposées. Pour ce faire, nous commencerons par une bref description de systèmes d'analyse (OLAP) pour les données XML, puis nous détaillons les différentes travaux antérieures. Enfin nous tenterons d'analyser ces travaux selon plusieurs critères.

3.2 Système d'analyse (OLAP)

Pour commencer, notons qu"il y a deux manières d"utiliser XML dans les systèmes d"analyse. La première consiste à utiliser XML pour représenter les données sources et /ou les résultats des requêtes, tout le traitement des requêtes se fait à l'aide d"outils OLAP existants basés sur des structures relationnelles ou multidimensionnelles. La seconde approche utilise XML aussi bien pour la représentation des données que pour leur traitement. Nous détaillerons ici ces deux approches, en nous concentrant sur la deuxième, et présenterons les outils et études existants.

3.2.1 Sources et résultats en XML

XML peut être utilisé pour représenter les données sources d"un système d"analyse. Dans cette approche, les informations intéressantes des documents XML sont extraites vers un moteur OLAP existant via un langage de traitement pour XML comme XSLT ou XQUERY. L"analyse des données se fait à l'aide d"un moteur OLAP classique, basé sur des structures relationnelles ou multidimensionnelles comme SQL Server de Microsoft.

Les données sources d'un système d'analyse peuvent ne pas être uniquement des documents XML mais également des base de données relationnelles classiques. En général, les sources de données sont hétérogènes. Il convient donc de fédérer ces différentes sources de données afin de pouvoir les analyser uniformément. Des travaux à ce sujet ont été effectués par [PRP02] .les auteurs proposent une approche pour fédérer des sources de données XML avec les cubes OLAP existants. Ils proposent également une architecture permettant d"interroger différentes sources de données de manière transparente.
XML peut également servir de format d"échange entre les différents composants d"un système d'a'nalyse.

3.2.2 Système natifs XML

Cette approche utilise XML aussi bien pour la représentation des données que pour leur traitement. Les données sont représentées par des arbres XML dans le moteur OLAP et sont traitées à l"aide de langage d"interrogation pour XML comme XQuery. Cette dernière n'a pas encore été largement étudiée. En effet, très peu d'études sur le sujet ont été menées. Une de ces études menée par Bordawekar, chez IBM en 2005 [BL05]. Montre que le modèle multidimensionnel OLAP n'est pas adapté à l'analyse des documents XML en utilisant le modèle de données XML. Les différences principales entre les données XML et les données structurées classiques sont les suivantes :

- Les données XML ne respectent pas nécessairement un schéma rigide comme les données relationnelles et peuvent contenir une structure plus au moins irrégulière. C"est pour cela qu"on parle souvent de documents semi-structurés.
- Un nœud d'un arbre XML peut être accédé par de multiples chemins au contraire d"une case d"un hyper-cube dans un système OLAP .

3.3 Entreposage de données XML

Nous donnons ici une vue globale des travaux dans le cadre de l"entreposage de données XML. Les travaux menés dans le contexte de l'entreposage de données XML peuvent être divisés en deux familles[MMD09] :

- La première famille propose une modélisation multidimensionnelle pour les entrepôts de données XML. Elle se base sur les modèles classiques (schémas en étoile et dérivés). Ces travaux permettent ainsi une utilisation dynamique des dimensions et offrent un support pour des outils d'analyse.
- Les approches de la seconde famille abordent la problématique de l'entreposage de documents XML. Elles perçoivent un entrepôt XML comme une collection de documents XML sans se baser sur un modèle particulier.

3.3.1 Entrepôts de données XML

Un entrepôt de données XML est un entrepôt de données qui a les caractéristiques suivantes :

- Le stockage peut se faire dans une base de données XML, natifs ou relationnel.
- Il fournir en sortie les données ou document XML, en format adapté pour des requêtes OLAP.
- La possibilité d'échanger les informations par HTTP en Intranet ou Internet.
- Supporter les techniques d'interrogation des documents XML (XQuery ,XSLT,XPATH,...).

3.3.2 Entrepôts de document XML

Dans cette catégorie d'entrepôts, des collections de documents XML, contenant des données textuelles peut structurées, sont stockées de manière à optimiser la recherche d'information. L'analyse multidimensionnelle est une problématique secondaire car un tel entrepôt est inadapté pour des traitements de style OLAP.

3.3.3 Entrepôt de données vs entrepôt de données XML

Dans la table 3.1, nous reprenons une synthèse d'éléments distinguant, de manière générale, un entrepôt de données d'un entrepôt de données XML.

Caractéristiques	***Entrepôt de données***	***Entrepôt de données XML***
Entrés	Données relationnelles	Fichier XML, base de données XML natives web, données complexes
Sorties	Mesures ,Tables faits,Dimensions cubes	Dimensions, fait sont représenté par un seul document XML Dimensions, fait sont représenté par plusieurs documents XML
SGBD	SGBD relationnelles	SGBD XML natives,SGBD relationnelles
Interrogation	SQL	XQuery, XPath, XSLT

TABLE 3.1 – Comparaison entre entrepôt de données et entrepôt de données XML

3.4 Modélisation multidimensionnelle en XML

Les documents XML peuvent se classer en deux grandes catégories : orientés document et orientés données. Ces deux types documents XML, engendrent deux catégories d'entrepôt XML :

1. Le stockage des documents XML
2. Le partage et la transmission des données XML de l'entrepôt

Dans la première catégorie, les travaux consistent à organiser le stockage de collections de documents XML(essentiellement textuelle) de manière à optimiser la recherche d'information dans ces documentes. Dans cette catégorie on trouve par exemple, les entrepôts de données web. La restitution n'est pas vraiment orientée décideur car elle ne prend pas en compte l'OLAP. Dans ce type, Dan Sullivan recommande l'utilisation de la fouille de texte[Sul01].

Dans la deuxième catégorie, il s'agit de proposer des modèles XML afin de partager et transmettre des documents. Dans ce type d'entrepôts, les données XML sont des données principalement issues de documents orientées données. En faite. Le but de ce type d'entrepôts de données est de fournir une vision uniforme des sources de données XML, favorisant ainsi leur intégration au sein des systèmes d'analyse. D'une manière générale, le processus de modélisation d'un entrepôt de données XML peut être décomposé en trois étapes (étapes classiques : conceptuel, logique et physique) :

- **Modélisation conceptuelle :** la modélisation conceptuelle consiste en la détermination des faits, des dimensions et des hiérarchies de dimensions. A ce niveau, la plupart des travaux utilisent UML pour modéliser ces derniers et proposent des modèles proches de l'utilisateur et indépendants de l'implémentation. [LA05], Utilisent le diagramme de classe UML pour représenter et intégrer les sources de données XML, puis le transforme en schéma XML, [GRV01], présentent un modèle conceptuel graphique pour l'entreposage de données .
- **Modélisation logique :** inclut un ensemble d'étapes qui mènent à la définition d'un schéma logique (le schéma en étoile, le schéma en flocon de neige). La majorité des travaux ont abordé le sujet des entrepôts XML, en se concentrant uniquement sur le traitement des aspects de la modélisation logique. [Pok01] définit un schéma en étoile appelé XMLStar Schéma. [HBH03] ont présenté XCube, une famille de documents XML pour l'échange des cubes de données, à ce niveau, la majorité des travaux proposés utilisent les schéma XML et les DTDs pour décrire les entrepots souvent en étoile à ce niveau de modélisation on

peut cité les travaux de [BMCA06], qui proposent un modèle logique de cubes de données utilisant les schémas XML.

- **Modélisation physique :** ça consiste à décrire la manière dont les données seront stockées. Elle explique l'implémentation des cubes de données, dépendamment du SGBD utilisé, elle constitue un premier pas vers l'optimisation des performances des entrepôts XML. Quelque travaux ont été proposées a ce niveau, notamment [PHS05] et [RRT04] abordent la conception physique des entrepôts XML, [XZ03] ont proposé une architecture générale pour entreposer des documents XML appelé DAWAX.

Le tableau 3.2, résume la classification des travaux traitant le problème de conception et la construction des entrepôts de données XML en tenant compte des points discutés précédemment à savoir :

1. Le niveau de modélisation : modélisation conceptuelle, logique et physique.
2. Le type de l'approche : (A) : approche décisionnelle (analyse de données), (B) : approche documentaire (entreposage de documents XML), ou (C) : hybride.

Travaux	***Modélisation conceptuelle***	***Modélisation logique***	***Modélisation physique***	***Type de l'approche***	***Modèles***
[Golferlli et al ,2001]	*	*		A	Spécifique
[Pokorney ,2001]		*		A	
[Hummer et al, 2003]		*		A	
[Baril et al ,2003]		*	*	B	
[Rusu et al ,2004]		*	*	B	
[Zhang et al ,2004]	*	*		B	Spécifique
[Li and An ,2005]	*	*		A	UML
[Park et al ,2005]	*	*	*	A	UML
[Boussaid et al ,2006]	*	*		C	Spécifique

Table 3.2 – Les travaux sur la modélisation multidimensionnelle des données XML

En remarque dans ce tableau, que peu de cherches s'intéressent à la modélisation multidimensionnelle des données XML sur le plan conceptuelle. A ce niveau, la plus part des travaux utilisent le mode orienté objet UML. Les auteurs s'intéressent plutôt à la modélisation logique. En plus, en peut clairement voir qu'ils s'intéressent beaucoup plus aux entrepôts orienté données XML. Dans ce qui suit nous allons présenter en détaille ces travaux, et leurs démarche de construction d'entrepôts de données XML (sans tenir compte de type de document XML orienté données et /ou documents), pour avoir une vision globale sur les démarches suivies, en tenant compte de niveau de modélisation (conceptuel et logique, physique).

3.4.1 Modélisation Conceptuelle et/ ou Logique

A cette section on évoque les travaux au niveau conceptuel et ou logique, ces deux niveaux sont souvent imbriqué dans une seul phase.

3.4.1.1 Modèle Dimensionnel de Faits. Gollfareli et al.

Les auteurs [GRV01], ont adopté la modélisation basée sur les arbres d'attributs proposée dans le contexte des entrepôts de données relationnel [GMR98b], pour concevoir un entrepôt de données XML. Introduisent un processus de modélisation d'un entrepôt XML à partir des DTDs des documents XML sources conformément à un Modèle Dimensionnel de Faits .Dans leur approche les auteurs présent les différentes manières d'exprimer des relations dans un schémas XML et les DTD à l'aide de sous-éléments et de clef REF/IDREF, pour représenter des modèles multidimensionnels. Dans [GMR98c], pour produire le schéma d'entrepôt de données, les auteurs

```
<!DOCTYPE webTraffic [
   <!ELEMENT webTraffic (click*)>
   <!ELEMENT click (host, date, time, url)>
   <!ELEMENT host (category | nation)>
   <!ATTLIST host
      hostId ID #REQUIRED>
   <!ELEMENT category (#PCDATA)>
   <!ELEMENT date (#PCDATA)>
   <!ELEMENT time (#PCDATA)>
   <!ELEMENT url (site, fileType,
urlCategory+)>
   <!ATTLIST url
      urlId ID #REQUIRED>
   <!ELEMENT site (nation)>
   <!ATTLIST site
      siteId ID #REQUIRED>
   <!ELEMENT nation (#PCDATA)>
   <!ELEMENT fileType (#PCDATA)>
   <!ELEMENT urlCategory (#PCDATA)>
]>
```

FIGURE 3.1 – Un DTD où les relations sont définit par sous-éléments [GRV01]

présentent une méthode composée par les étapes suivantes :

1. Détermination des faits.
2. Construction d'un arbre des attributs (attribute tree) pour chaque fait.
3. Suppression des attributs inutiles.
4. Définition des dimensions et des mesures (fact attributes).
5. Définition des hiérarchies.

Arbre d'attribut est un arbre tel que :

- Chaque sommet correspond à un attribut –simple ou composé- du schéma.
- La racine correspond à l'identifiant de fait de F
- Pour chaque sommet V, l'attribut correspondant détermine fonctionnellement tous attributs correspondant aux descendants du V.

Une fois que l'arbre d'attributs a été construit, quelques nœuds qui ne sont pas intéressants peuvent être supprimer de l'arbre.

Pour construire le schéma conceptuel de l'entrepôt de données, la méthodologie comprend les étapes suivantes :

1. Simplification de la DTD.
2. Créer le graphe de la DTD.
3. Sélectionner Faits
4. Pour chaque fait :
   - Construction de l'arbre d'attribut du graphe de DTD.
   - Réarrangement de l'arbre d'attribut.
   - Définition des dimensions et des mesures.

1. **Simplification de la DTD :** Les sous-éléments dans les DTDs peuvent être déclarés d'une manière compliquée et superflue. Cependant la DTD peut être simplifié par la transformation de la DTD en une représentation plate comme suit :
   - Les sous-éléments ayant le même nom sont regroupés.
   - Beaucoup d'operateurs unaires sont réduits à un opérateur unaire simple.
   - Tous les opérateurs « + » sont transformés en des operateurs « * ».
2. **Créer un graphe de DTD :** Après la simplification de la DTD, un graphe représentant sa structure peut être créer : ses sommets correspondent aux éléments, aux attributs et aux opérateurs dans la DTD.

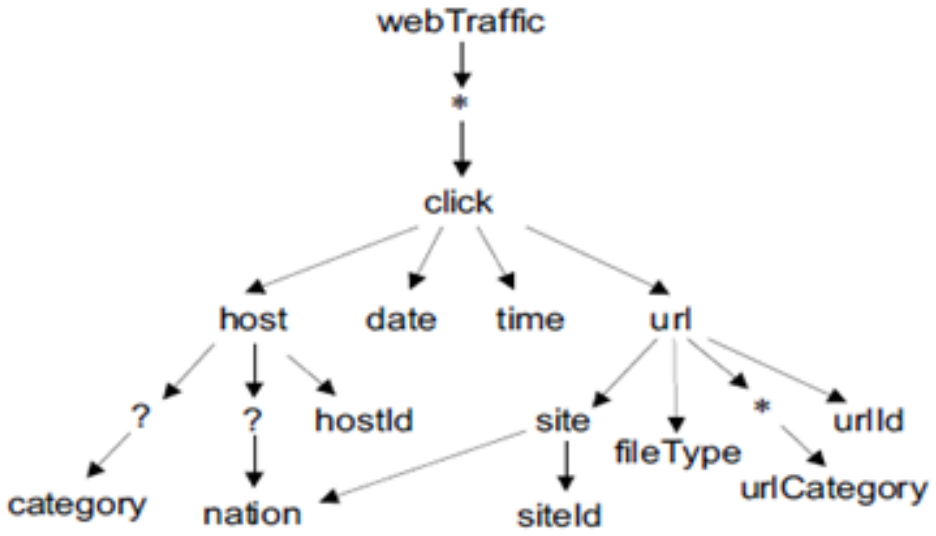


Figure 3.2 – Graphe de DTD [GRV01]

3. **Définition des Faits :** Le concepteur choisit un ou plusieurs sommets du graphe de la DTD comme faits ; chacun d'eux devient la racine d'un schéma de fait.

Construction de l'arbre d'attribut :

Les sommets de l'arbre d'attributs sont un sous-ensemble des sommets d'éléments et d'attributs du graphe de DTD. L'arbre d'attributs est initialisé avec le sommet F de fait ; puis il est agrandi en dirigeant périodiquement les dépendances fonctionnelles entre les sommets du graphe de DTD. Chaque sommet V inséré dans l'arbre d'attributs est augmenté comme suit(le procédé augmente) :

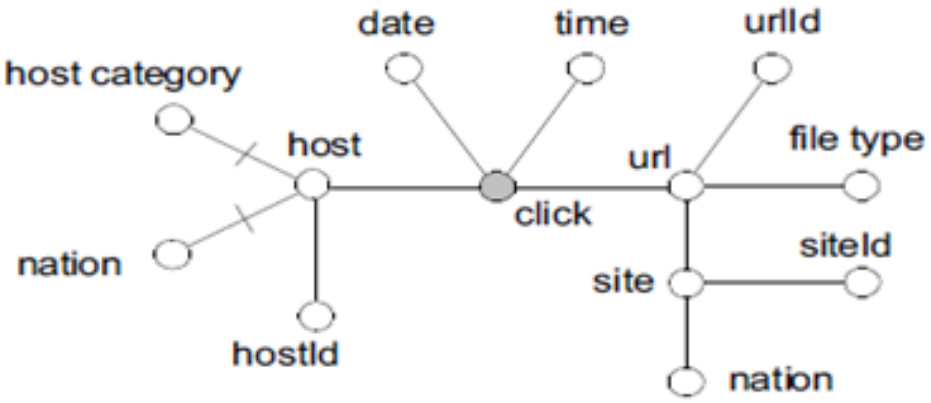


FIGURE 3.3 – Arbre d'attribut DTD [GRV01]

- Pour chaque sommet W(W correspond à un élément ou à un attribut qui est un enfant de V dans le graphe de DTD), on l'ajoute à l'arbre d'attribut en tant qu'enfant de V, si W est un operateur « ? », on ajoute son fils à l'arbre d'attribut en tant qu'enfant de V, si W est « * » ,on ne l'ajoute pas à l'arbre d'attribut.
- Puis on procède au réarrangement de l'arbre d'attribut, puis la définition des dimensions et des mesures par la sélection parmi les enfants de la racine.

Cette approche est caractérisé par :

Avantages :
- Basé sur le schéma source.
- Document XML orienté données.
- Modélisation au niveau conceptuel de l'entrepôt de données XML

Inconvénients :
- La structure des sources est décrite par une DTD, le cas de plusieurs schémas n'a pas été abordé dans cette démarche.
- Mapping entre les documents XML sources et le schéma cible doit être défini par le concepteur. Ils s'intéressent à la représentation multidimensionnelle des données XML au lieu de proposer un modèle conceptuel multidimensionnelle.

3.4.1.2 XML–star. Pokorny.

Jaroslav Pokorny propose une approche pour la construction d'un entrepôt de données XML qui se base sur l'utilisation des vues XML. L'approche consiste à reconstruire les hiérarchies des dimensions à partir de fragment de diverses données XML sources. Les dimensions constituant l'entrepôt doivent être conformes à une DTD. Cette DTD est déterminée par la fusion des fragments de DTD des documents XML intervenant dans la construction de la vue. Le plus déterminant pour construire une hiérarchie H de dimensions est de trouver les relations logiques entre les dimensions voisines dans H, une fois les données qui caractérisent chaque dimension sont déterminé, il ne reste qu'à représenter les faits par une DTD [Pok01].

Selon cette approche, un schéma en étoile sera défini comme un triplet ¡ H,F,IC ¿ où H est un ensemble de hiérarchies de dimensions, F une DTD représentant la table des faits et IC un ensemble de contraintes référentielles. Un exemple de schéma en étoile XML est montré dans la figure 3.4.

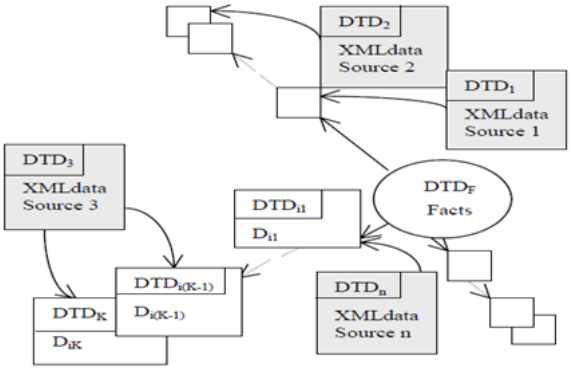


FIGURE 3.4 – Modèle XML-Star [Pok01]

L'approche proposé commence par reconstruire la hiérarchie H des dimensions à partir de n collections de documents XML C1...Cn, n¿1 dotées des DTDs, DTD1...DTDn,

$$n \geq 1$$

respectivement. Une collection de document XML C contiendra des XML valides dans un élément de DTDc (la DTD collection).

Pokorny suppose que chaque collection de documents XML est localement homogène, et donc dotée d'une seule DTD. Chaque DTD peut être la source d'un ou plusieurs niveaux de hiérarchie d'une dimension D. Les éléments PCDATA(Description Parsed Character Data) de la DTD choisis par l'utilisateur, sont utilisés pour décrire la dimension D. Ces éléments extraits de la DTD constituent une sous-DTD dénotée DTDd, de la dimension D.

- **Les dimensions :** Les dimensions sont des documents XML. Modélisée sous forme de séquence de DTDs qui sont logiquement associées au niveau conceptuel. Ce lien logique est similaire à l'intégrité référentielle définie dans les bases de données relationnelles.
- **Membres de dimension Sous-DTD :** Chaque DTD peut être la source pour un ou plusieurs membres de H. DTDD est une partie du DTD, décrivant un membre D de H. Il y'a de deux conditions qui devraient être remplies :
  - DTDD est une DTD.
  - Les éléments de DTDD indiqués comme PCDATA sont suffisant pour décrire D.

  Une DTDD peut être indiqué par de divers moyens. Dans le cas le plus simple, c'est un sous-ensemble de !L'ELEMENT du DTDc de la collection.
- **Fait :** Dans cette approche, les auteurs n'ont pas bien expliqué, comment construire les faits, mis à part que les faits sont aussi modélisés sous forme d'un document XML. La clés de cette approche, est l'utilisation des vues XML pour les documents XML. Le besoin de ce concept (vue) est le même avec les bases de données relationnelles. Les vues XML couvre une collection C de document XML. Définit par des DTDs, avec l'utilisation de la requêtes V (langage de requête pour des documents XML). La matérialisation V(C) de la vue se fait par l'évaluation de V sur C. Les vues XML, peuvent être utilisées pour modéliser les associations qui peuvent exister entre deux documents XML décrits par deux DTD différentes. Ces vues aident à vérifier l'intégrité référentielle et accélèrent l'exécution des requêtes.

Cette approche est caractérisé par :

Avantages :

- Basée sur les schémas source.
- Orienté données.

- Proposition d'un schéma en étoile au niveau logique.
- Intégration physique des documents XML.

Inconvénients :

- Basées sur les DTDs, (hérite les insuffisances en terme de description,N'est pas adapté à la source de données hétérogène(c-à-d chaque document associées un schéma), or il suppose que les documents XML sont validé par seul DTD.
- Rien n'a été donné sur la construction des faits.

3.4.1.3 XCube. Hummer et al.

XCube utilise des documents XML pour le stockage, l'échange et l'interrogation des données d'un ou plusieurs entrepôts[HBH03]. XCube est organisé en fusion de modules ou formats : Schéma XCube, Dimensions XCube et Faits XCube.

– **Schéma XCube :**Le schéma XCube est le format central pour décrire la structure multidimensionnelle d'un cube de données. Il modélise les dimensions et les mesures contenues dans un cube. La structure type d'un document de schéma XCube est décrite à la figure 3.5 .

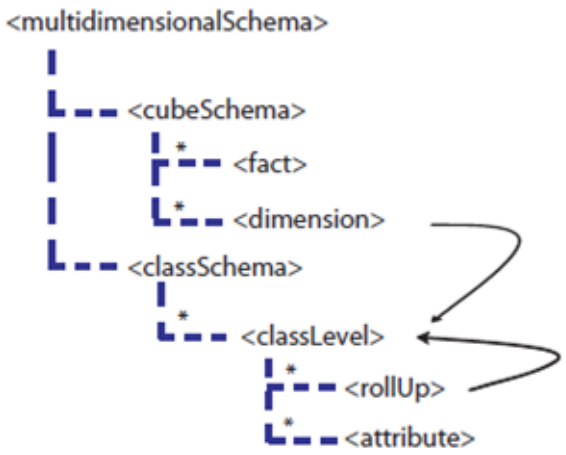


FIGURE 3.5 – Structure d'un cube XCubeSchema[HBH03]

Les lignes discontinues de la figure 3.5, représentent les relations père-fils et le symbole "*" indique que le nombre d'occurrences du fils peut être arbitraire (0 ou plusieurs).
Sous la racine MultidimentionalSchéma, existent deux blocs(sous-éléments) ; cubeSchema et clssSchema. La section cubeSchema contient une collection de faits (l'élément fact) et de dimensions (l'élément dimension). L'élément dimension possède des attributs pour référencer la dimensions et la granularité la plus fine de celle-ci. L'élément fact possède un attribut qui référence le nom du fait.
La section classSchema décrit la classification des niveaux des dimensions par l'élément classLevel. Ce dernier est constitué de l'élément attribute qui détermine l'attribut de la dimension et de l'élément rollup qui pointe vers la classification du niveau plus haut dans la hiérarchie des dimensions. La connexion entre la dimension et la classification des hiérarchies est établie par une référence depuis cubeSchema /dimension vers classSchema/classLevel(a/b : l'élément b est le fils de l'élément a).
Le schéma XCube peut aussi fournir les définitions des unités et des types des mesures,

la classification des dimensions ainsi que celles des nœuds et des attributs par un XML schéma. Il peut aussi définir les opérations d'agrégation définis sur les mesures du cube.

– **Dimension XCube :** Dimension XCube formalise la structure des dimensions. Un document de dimension XCube contient des nœuds appartenant à la classification des niveaux définie sur le schéma XCube.

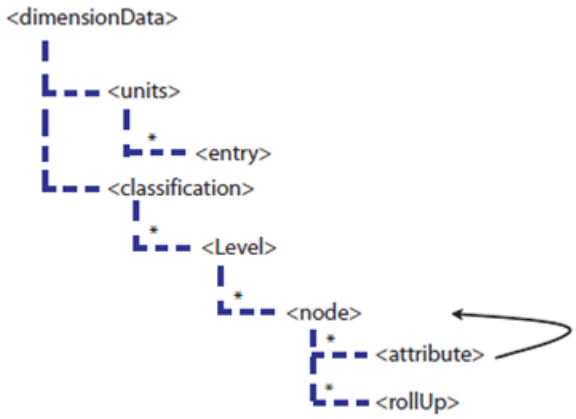


Figure 3.6 – Structure d'un document XML de Dimension XCube[HBH03]

La structure de base d'un document Dimension XCube est présentées à la figure 3.6. La racine dimensionData a deux fils : l'élément units, importé depuis les unités définies dans le schéma XCube et l'élément classification. Chaque élément Level est composé d'éléments node. Ces éléments décrivent les attributs (éléments attribute) des dimensions et leurs hiérarchies (éléments rollUp).

– **Fait XCube :** Un fait XCube définit la collection des cellules des cubes de données de l'élément cube. Chaque cellule est constituée, comme la figure 3.7, de deux sous-éléments :dimension, représentant les coordonnées dimensionnelles et fact renseignant la valeur du fait.

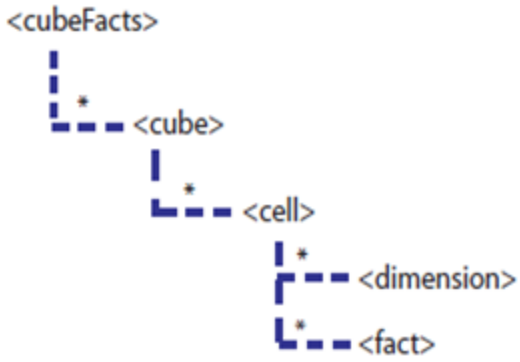


Figure 3.7 – Structure d'un document XML Fait XCube[HBH03]

En plus de la description des données du cube, XCube contient d'autres formats XCube-Text, XCubeQuery, XCubeFunction.

- XcubeText fournit des descriptions textuelles et des commentaires sur les formats schémaXCube et dimension XCube.
- XCubeQuery est un moyen d'échange d'information s entre un serveur et un client. Il fournit au site du client les données nécessaires à un besoin défini. XCube Query contient

plusieurs types de requête(getCubeSchema-List, getCubeSchema, getClassSchema, getClassNode,XCubeQuery,getFacts,...), dans ce qui suit quelque exemple d'utilisation de ces requêtes.
  - **getCubeSchema List :**le serveur retourne la liste des cubes.
  - **getCubeSchema :** pour explorer un cube spécial. La requete est un document XQuery qui contient getCubeSchema avec le IDde cube. Le résultat est une partie de document XCubeSchema qui contient une liste de fait et une liste de dimensions correspondant à l'ID spécifié dans la requête.
  - **getClassScheme :** récupère le détail de schéma de classification d'une dimension.
- XCubeFunction est en cours de développement. Il devrait permettre d'interroger un serveur XCube afin d'obtenir des cubes complets.

Cette approche est caractérisé par :

Avantages :
- Propose une meilleure modélisation en niveau logique(étoile ou en constellation) d'un entrepôt de données.
- Supporter des modèles de données multidimensionnelles.
- Propose un langage d'interrogation pour des cubes de données.
- Document XML orienté données.
- L'absence de redondance dans les données. Ce dernier point facilite grandement les mises à jour et la maintenance de la base de données.
- Le format proposé est transportable, extensible, facilement convertissable à différentes sources de données et formats.

Inconvénients :
- Les faits et les dimensions sont représenté dans des documents séparés et sont liés via des identificateurs et les instances des niveaux sont liées entre elles par des identificateurs, ce qui engendre un nombre important de jointures pour naviguer dans les hiérarchies de dimensions.

3.4.1.4 X-WAREHOUSE. Zhang et al.

Les auteurs [MJ04] proposent un entrepôt, qui se base sur l'historique des requêtes d'utilisateurs, appelé X-Warehouse, afin de construire l'entrepôt par la découverte des chemins de requêtes les plus fréquentes.

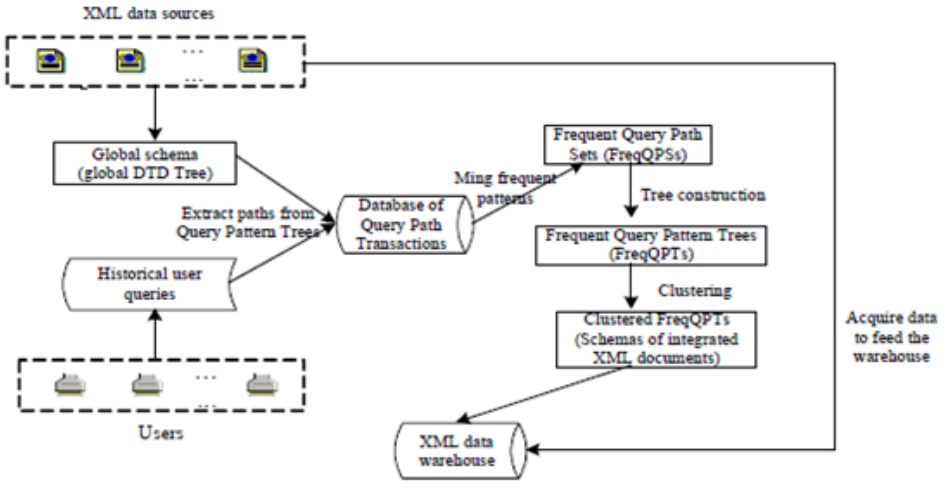


FIGURE 3.8 – Une vue d'ensemble de X-WAREHOUSE[MJ04]

Les auteurs ont proposé une approche pour matérialiser des entrepôts de données XML basés sur les modèles de requête fréquentes. Requêtes publiées par des utilisateurs (historiques).L'entrepôt de données X-Warehouse est basé sur les modèles de requêtes fréquentes des utilisateurs. Ça construction implique les quatre étapes suivantes :

- **Transformer les requêtes d'utilisateurs en Chemins de requêtes :** chemin de requête QP est une expression de chemin d'un arbre de DTD qui commence à partir de la racine de l'arbre. Une requête émise par l'utilisateur peut être transformée en un certain nombre de QPs, appelé une transaction de QP. Après transformé un ensemble de requêtes en transactions de QP, nous obtenons maintenant une base de données contenant toutes ces transactions de QP, dénotées comme DQPTra.
- **Découvrir un ensemble de Chemin de requête Fréquent dans une Base de données de Transactions de Chemins de requêtes :** Application des techniques d'extraction dans DQPTra pour découvrir l'ensemble de chemins de requêtes fréquents(FreqQPSs) dans DQPTra. Des arbres de modèle de requête fréquents(FreqQPTs) servent de modules pour des schémas des documents XML intégrés dans l'entrepôt.
- **Schémas des documents XML a intégré :** La construction de schéma de document XML à intégrer dans l'entrepôt pour toute les FreqQPTs extraite.
- **Acquérir les données pour alimenter l'entrepôt :** La dernière étape de la construction de l'entrepôt X-Warehouse, c'est la récupération des données des documents XML source, quand les schémas des documents XML dans l'entrepôt sont prêts.

Cette approche est caractérisé par :

Avantages :

- Basé sur les requêtes utilisateurs.
- Document XML orienté données.
- Intégration au niveau physique..

Inconvénients :

- Il faut reconsidérer le problème avec les nouvelles requêtes en entrée.
- Problème de mise à jours.
- Rien n'a été donné sur la DTD globale des sources de données.

3.4.1.5 X-Warehousing. Boussaid et al.

Les auteurs [BMCA06] ont définit une démarche méthodologique, appelée X-Warehousing, pour la modélisation multidimensionnelle des données complexes. C'est une approche basé entièrement sur XML, c.-à-d, elle permet de concevoir un entrepôt(ou un magasin), de représenter son schéma conceptuel à l'aide de schéma XML et enfin d'alimenter la structure multidimensionnelle à l'aide de données initialement stockées dans des documents XML. Elle part des objectifs d'analyse d'un utilisateur représentés par un modèle conceptuel multidimensionnelle(MCM). Elle utilise un ensemble de données complexes structurées dans des documents XML pour générer une base organisée de façon multidimensionnelle exprimant un contexte d'analyse. Dans ce modèle une instance d'une dimension est représentée par l'instance de son niveau le plus haut dans la hiérarchie. Une mesure d'un fait est représentée par un attribut du nom de la mesure et la valeur de la mesure est assignée à cet attribut. Enfin un fait est modélisé par un élément qui porte le nom du fait et qui contient toutes ses mesures en attributs et les instances de ses dimensions.
Méthode de conception :

Les auteurs [BMCA06],ont proposée une approche pour construire des cubes de données avec des documents XML sources pour faire des analyses en ligne(OLAP). Cette dernière prend comme entrées le modèle conceptuel multidimensionnel(MCM), représentant les objectifs d'analyse exprimé par l'utilisateur et les documents XML qu'il veut analyser, puis transformé le MCM

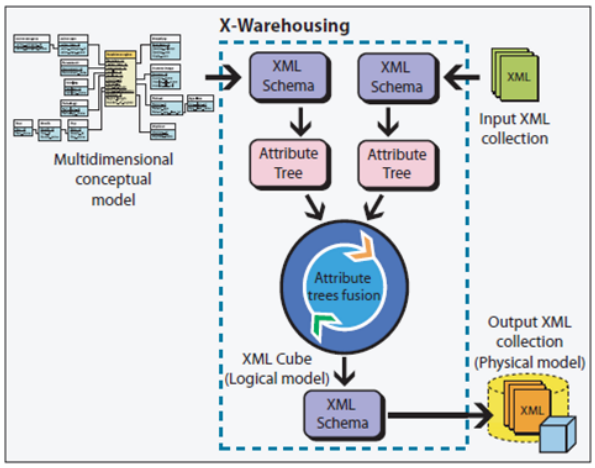


FIGURE 3.9 – Les étapes de l'approche X-Warehousing[BMCA06]

et chaque document XML en un arbre d'attributs, et les comparés un par un (Chaque arbre d'attributs d'un document XML en entrée est comparé à l'arbre d'attributs du MCM). Si le document en entré contient une information minimale(un ensembles d'éléments obligatoire qui portent sur les mesures, les dimensions ou les hiérarchies de dimensions et leurs attributs)requise dans le MCM, alors une instance de ce document sera crée et validée en accord avec le schéma XML du cube XML que génère l'application X-Warehousing. Sinon, dans le cas ou le document XML en entrée ne contient pas assez d'information requise par le MCM, le document est rejeté. Cette approche permet d'obtenir un ensemble homogène de données avec des contraintes strictes sur leurs contenus. Ces données sont structurées dans des documents XML et orientées vers des analyses en ligne. Cette collection de documents XML représente un cube OLAP que les auteurs désignent par cube XML. La figure 3.9 résume les différentes étapes de l'approche X-Warehousing.

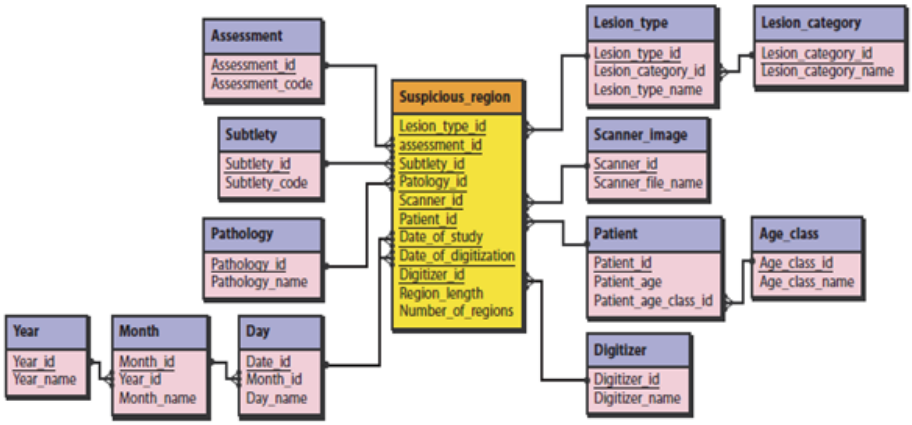


FIGURE 3.10 – Exemple d'un modèle conceptuel du cube de données[BMCA06]

– **Modèle Conceptuel Multidimensionnel (MCM) :**

Un MCM est une description d'objectifs d'analyse. La finalité de cette dernière est de décrire un contexte d'analyse. Il s'agit de représenter un ou plusieurs indicateurs et un ensemble d'axes d'observation et cela d'un point de vue strictement conceptuel, indépendamment de toute considération logique ou physique. Pour représenter ces indicateurs et leurs descripteurs, ceux qui sont à l'origine des schémas en étoile les auteurs

ont proposé d'utiliser des concepts empruntés au formalisme relationnel : des tables pour regrouper les indicateurs d'une part et leurs descripteurs d'autre part. De plus, pour relier ces différentes tables, ils ont utilisés le mécanisme des clefs (principales, étrangères). Dans leur approche une description des schémas en étoile et en flocon de neige a été donnée.

– **Méthode de conception :**

Dans leur approche une description des schémas en étoile et en flocon de neige a été donnée. La définition des dimensions hiérarchisées XML qui est une partie d'un schéma XML et le fait XML est un document XML valide conformément au modèle en étoile ou au modèle XML en flocon de neige correspondant au modèle conceptuel du cube de données(MCM).

– **Construction des cubes XML :**

Le MCM et les documents XML sources sont exprimés à l'aide de schémas XML puis transformés en arbres d'attributs pour être comparés. Des algorithme d'appariement permettent grâce à des opérateurs de fusion par élagage qui consiste à supprimer des parties des deux arbres, ou par greffe, elle s'effectue lorsque des sous arbres communs n'ont pas la même structure de relation dans les deux arbres en entrée, proposés par Golfarelli, de traiter les arbres d'attributs pour générer le schéma XML du cube XML .

```
<xs:element name="F">
      <xs:complexType>
            <xs:sequence>
                  <xs:element name="D1" type="D1_Type" />
                  <xs:element name="D2" type="D2_Type" />
                  <xs:element name="D3" type="D3_Type" />
                  <xs:element name="D3" type="D3_Type" />
                  <xs:element name="D4" type="D4_Type" />
            </xs:sequence>
            <xs:attribute name="F.M1" type="xs:integer" />
            <xs:attribute name="F.M2" type="xs:integer" />
      </xs:complexType>
</xs:element>

<xs:complexType name="D1_Type">
      <xs:attribute name="D1.A1" type="xs:string" />
</xs:complexType>

<xs:complexType name="D2_Type">
      <xs:attribute name="D2.A1" type="xs:string" />
      <xs:attribute name="D2.A2" type="xs:string" />
</xs:complexType>

<xs:complexType name="D3_Type">
      <xs:attribute name="D3.A1" type="xs:string" />
      <xs:attribute name="D3.A2" type="xs:string" />
</xs:complexType>

<xs:complexType name="D4_Type">
      <xs:attribute name="D4.A1" type="xs:string" />
</xs:complexType>
```

FIGURE 3.11 – Description d'un cube par un schéma en étoile XML[BMCA06]

Cette approche est caractérisé par :

Avantages :

- Basée entièrement sur XML, qui permet une bonne représentation des données complexe, et leur intégration facilement.
- Basée sur les besoins utilisateurs.
- Orienté données/documents deux formes sont proposées. La première est basée sur un schéma en étoile et l'autre sur un schéma en constellation.
- Ne nécessite pas de requêtes complexes de jointures parce que les informations nécessaires à propos d'un fait se retrouvent dans le même document.
- Grâce à l'imbrication d'éléments XML pour modéliser les dimensions, on peut connaître la profondeur des niveaux des hiérarchies.

Inconvénients :

- Ce modèle implique beaucoup de redondance dans les données des dimensions.

- Les informations nécessaires à propos d'un fait se retrouvent dans le même document, ce qui peut impliquer des difficultés de mise à jour et de maintenance.
- Elle n'est pas adaptées pour l'analyse de données textuelles issues de documents XML.
- Le format physique de stockage du XML est volumineux, lorsque les instances sont stockées ensemble.

3.4.2 Niveau physique

3.4.2.1 DAWAX. Baril et al.

Les auteurs perçoivent un entrepôt de données comme une collection de vues XML matérialisées [XZ03]. Les vues XML constituent l'entrepôt et permettent de filtrer et de structurer les données pour un besoin d'analyse. Selon cette approche, les auteurs ont développé un système appelé DAWAX(DAta Warehouse for XML). DAWAX se base sur trois modules principaux.

1. **Module de spécification de l'entrepôt de données :** Ce module définit les vues XML qui constituent l'entrepôt de données. Deux sortes de vues peuvent être distinguées :
   - Des vues de sélection et des vues composées qui créent de nouveaux éléments et attributs à partir de plusieurs sources.
   - Des fonctions d'agrégation sont utilisées pour définir les nouvelles valeurs.
2. **Module de gestion des métadonnées :** La spécification de l'entrepôt de données est stockée dans un document XML. Ce document contient des informations sur le stockage des données, la provenance (URL) des sources des données et les spécifications des vues. Afin de fournir une intégration de vues sources hétérogènes, l'entrepôt de données est considéré comme un document XML union de toutes les vues. Un élément entrepôt est défini par la ligne de DTD suivante :

$$> 0, < \delta \geq,$$

$$<!ELEMENT datawarehouse(view1, view2, ..., viewN) >$$

3. **Module de stockage et de gestion des données :** Le stockage des données s'effectue grâce à un mapping vers une base de données relationnelle. Il en est de même pour les vues. Pour cela, trois tables sont utilisées :
   - Patterns pour le stockage des modèles.
   - Fragments pour le stockage des fragments .
   - Views pour le stockage des vues.

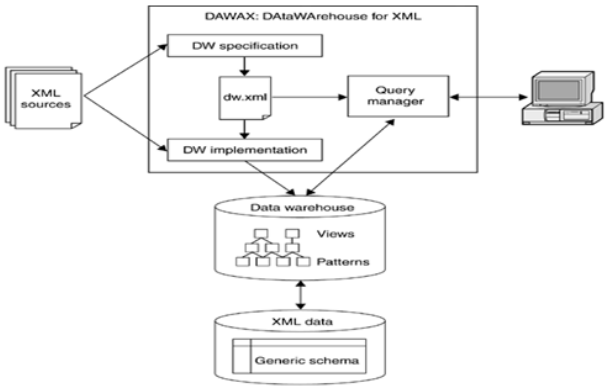


FIGURE 3.12 – Architecture générale DAWAX[XZ03]

Cette approche est caractérisé par :

Avantages :

- Basé sur les schémas sources.
- Orienté document.
- Proposition d'un modèle physique.

Inconvénients :
- Ne prend pas en charge les concepts de la modélisation multidimensionnelle
- Une base de données relationnelle pour le stockage des données XML .
- Impossibilité d'utiliser les langages d'interrogation XML.

3.4.2.2 Entrepôt de données XML. Rusu et al.

Rusu et al. Se basent sur le langage XQuery pour construire un entrepôt de données XML [RRT04]. La méthode proposée couvre toutes les étapes du processus d'ETL : l'extraction, la transformation et le chargement des données XML dans l'entrepôt. Toutes ces étapes sont assurées à l'aide de XQurey. Les auteurs définissent des règles de chargement pour faire face aux problèmes de conflits rencontrés lors du chargement des données dans l'entrepôt. Ces conflits sont de deux types :
(1) au niveau schéma lorsque ce dernier ne correspond totalement pas aux données, et
(2) au niveau données suite aux irrégularités structurelles des données XML. Les faits et les dimensions sont stockés dans des documents XML également construits par des requêtes XQuery. Un exemple de requête XQuery qui construit une dimension temps est présenté dans la figure 3.13. Cette approche est caractérisé par :

```
let $b:=0 document {
for $t in distinct-values(doc("libraryBooks.xml")//borrowing_date)
let $b:=$b+1
return <borrowtime>
<timekey>$b</timekey>
<borrowdate>$t</borrowdate>
<month>get-month-from-date($t)</month>
</borrowtime> }
```

FIGURE 3.13 – Exemple de requête XQuery de création de dimension temps[RRT04]

Avantages :
- Basé sur les sources de données.
- Document XML orienté données.
- Proposition d'un modèle physique, contrairement à la majorité des travaux qui ne s'intéressent qu'au niveau logique.
- intégration des données XML au niveau physique.

Inconvénients :
- La méthode génère un document XML de dimension et un autre de fait, ce qui impose des opérations de jointure très compliquées pour l'interrogation.
- Rien n'a été proposé au niveau conceptuel.

3.4.2.3 XML-OLAP. Park et al.

Les auteurs [PHS05] proposent une plateforme pour l'analyse en ligne de documents XML nommée XML-OLAP. Comme le montre la figure, les auteurs se basent sur un entrepôt de données XML où les faits sont représentés par une collection de documents XML. Les auteurs

représentent chaque fait par un document XML. Un document XML d'une collection de dimension stocke une instance de la hiérarchie d'une dimension .Les auteurs affirment que cela permet d'éliminer les opérations de jointure entre les niveaux hiérarchiques d'une dimension.

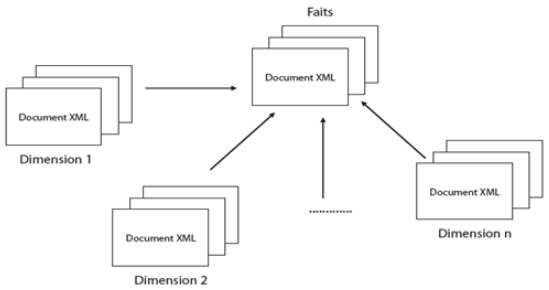


FIGURE 3.14 – Modèle multidimensionnel d'entrepôt XML-OLAP[PHS05]

Afin d'interroger l'entrepôt de données XML, les auteurs proposent un langage d'expression multidimensionnel : XML-MDX [PHS05]. Ils étendent en fait le langage relationnel MDX de Microsoft avec deux operateurs : CREATE XQ-CUBE pour la création de cubes XML et SELECT pour leur interrogation. De plus, les auteurs définissent sept operateurs d'agrégation : ADD, LIST, COUNT, SUMMARY, TOPIC, TOP KEYWORD et CLUSTER. Certains de ces operateurs sont inspirées du relationnel, d'autres utilisent des techniques de fouille de données textuelles pour agréger des valeurs non-additives. Cette approche est caractérisé par :

Avantages :
- Basée sur les schémas source.
- Orienté document.
- XQ-Cube est une généralisation d'un cube relationnel. Il devient un cube relationnel quand il utilise des données numériques, ou devient un cube en textes(vous avez dit relationnel dans le premier cas) dans le cas des données textes.
- Stockage de gestion en natifs des documents XML des bases de données XML.
- Utilisation des langage d'interrogation XML.
- Fournit un mécanisme d'agrégation sur des documents XML, utilisant le langage XML-MDX.

Inconvénients :
- Redondance dans le stockage des dimensions au niveau physique
- Ils ne proposent pas la modélisation pour les structures de base telle que la mesure.

3.4.3 Synthèse générale

Les travaux qui traitent de l'entreposage de données XML varient par leur approche de construction de l'entrepôt et couvrent plus ou moins les différentes étapes du processus d'entreposage : ETL (Extraction, Transformation et Chargement), modélisation et analyse. Certains proposent de traiter les données directement à partir de leurs sources, d'autres favorisent le chargement dans un entrepôt de données modélisé par un schéma en étoile. Certains proposent également des opérateurs d'analyse spécifiques aux données XML. Ces travaux sont résumés dans le tableau 3.3.

Les approches qui proposent de construire des entrepôts de documents XML sont basées sur des vues définies par l'utilisateur. Elles proposent des méthodes de conception et de construction d'un espace de stockage XML (XML repository) qui représente le contexte d'analyse de l'entrepôt

Familles	***Travaux***	***ETL***	***Entrepôt***	***Analyse***
Entrepôts de documents XML	Baril et al(2003)	*	*	*
	Zhang et al (2005)	*		
Entrepôts de données XML	Golfareli et al(2001)	*	*	
	Pokorny et al (2002)	*	*	*
	Hummer et al (2003)		*	*
	Rusu et al (2004)	*	*	
	Park et al (2005)	*	*	*
	Li and An (2005)	*	*	
	Boussaid et al (2006)	*	*	

Table 3.3 – Comparaison des travaux d'entreposage XML.

et ne définissent aucun modèle logique d'entrepôt XML. Ces approches sont orientées besoins et sont plutôt employées lorsque les spécifications de l'entrepôt sont peu susceptibles d'évoluer.

Les approches qui proposent de construire des entrepôts de données Web exploitent des schémas XML et des DTD à des fins de modélisation. Ces formalismes permettent de d'écrire la structure des données et de définir des relations entre les documents XML sources, qui sont souvent hétérogènes. Ces DTD/schémas sont transformés et fusionnés pour construire le schéma de l'entrepôt[GRV01].

Cependant, les mécanismes de transformation peuvent s'avérer complexes. Ces approches sont donc utilisées quand les dimensions sont statiques, c'est-à-dire que les besoins d'analyse sont fixés. De plus, ces besoins peuvent parfois ne pas être totalement satisfaits. Ce problème survient notamment quand les données ciblées ne sont pas disponibles. Ces travaux proposent une préparation de données à des fins d'analyse, mais n'incluent aucun outil ni opérateur d'analyse spécifique aux données XML.

Les approches de la famille des entrepôts de données XML respectent une modélisation multidimensionnelle pour construire un entrepôt ou un magasin de données. Ces approches différent principalement dans la manière de représenter les faits et les dimensions, ainsi que par le nombre de documents XML utilisés pour les stocker. Certains travaux se basent sur des critères de performance pour définir ces éléments. D'autres proposent des modèles simplifiés pour la modélisation de l'entrepôt ou respectent strictement une modélisation en étoile.

Ces approches proposent également des mécanismes d'extraction et de chargement des données dans l'entrepôt. Par exemple, Rusu et al. définissent des règles pour le nettoyage et le chargement des données et Park et al. s'inspirent des approches orientées objets pour la modélisation et l'intégration des données dans l'entrepôt. Ces approches utilisent des schémas UML construits à partir des DTD des données sources. De plus, Hummer et al. Proposent de représenter les métadonnées de l'entrepôt par un document XML, tandis que Pokorny valide les documents XML de l'entrepôt par des DTD. Ces métadonnées n'existent dans aucun autre travail. Or, ce type de document peut s'avérer utile pour l'interrogation et la mise à jour des données de l'entrepôt. Finalement, d'autres travaux proposent des solutions pour l'interrogation des données XML. Par exemple, Hummer et al, définissent un modèle pour l'interrogation de cubes de données XML et Park et al, proposent des operateurs d'analyse adaptés aux données XML.

La méthodologie utilisée pour la construction de DWAX (entrepôt de document), ne prend pas en charge les concepts d'une modélisation multidimensionnelle, en plus DWAX utilisent une

base de données relationnelles pour le stockage des données XML, ce qui écarte la possibilité d'interrogation avec un langage XML sans faire un mapping vers SQL .
Le symbole (+) y indique qu'une approche traite une étape du processus d'entreposage.

Chapitre 4

XStar-Warehouse(XSW) : Modélisation Conceptuelle et Logique d'un Entrepôt de Données XML

4.1 Introduction

Nous avons réalisé une analyse approfondie de l'état de l'art du domaine des entrepôts de données XML. Elle nous permet d'identifier les caractéristiques des entrepôts XML, les principaux niveaux de conception. Prenant comme base cette analyse, nous avons proposé une méthode de conception des entrepôts de données XML.

4.2 Approche "XStar-Warehouse" de conception d'un entrepôt XML

La méthodologie de conception est une condition essentielle pour assurer le succès d'un projet d'entreposage de données complexes. Le processus proposé pour la modélisation conceptuelle et logique de l'entrepôt de données XML est présenté dans la figure 4.1. Notre approche se décompose en quatre étapes :

1. **Construction de schéma XML global intermédiaire**

   Pour la construction d'un entrepôt de données XML, la première étape de notre approche consiste en la définition du schéma XML globale,unifiant les schémas des sources de documents XML(des DTD,des XSD,et des bases de données). Ces informations doivent être extraites et transformées au formats de représentation appropriés qui est le XML schéma. Le choix de XML schéma est justifié par sa flexibilité, extensibilité et sa puissance. Avec l'incorporation du typage de données et la gestion des espaces de nommage (namespaces).

   Définition : le schéma XML Global est l'union des schémas XSDi généré pour chaque source S, avec élimination des éléments et des parties de schéma qui ne sont pas pertinentes par rapport au domaine à analyser.

   De façon générale, cette partie de notre approche, illustrée dans figure 4.1, consiste en une séquence itérative des étapes suivantes :

Transformation : Permet de sélectionner et de transformer les données sources à des schémas XML équivalent. Le but de cette phase est la traduction des données sources(document XML, DTD, Schéma XML, BDD,...) à intégrer dans un modèle de données commun et d'homogénéisé leurs représentations en l'expriment sous la forme d'une définition de schéma XML(i.e. XSDi). Cette tache est appelée Extraction.

Nettoyage : Cette phase traite de bruits, d'erreurs et de données manquantes, les données dupliquées, elle consiste à nettoyer chaque schéma séparément. Dans cette étape en peut réutiliser quelques règles des approches proposées dans le domaine de Datawarehousing.

Fusion : Dans cette étape, nous cherchons à générer à partir des schémas XSDi générées lors de la phase précédente un schéma XML globale qui représente toute les sources de données. L'objectif principal est d'avoir un ensemble cohérent non redondant et une vue globale des schémas des sources de données. La construction d'un tel schéma est basée sur l'utilisation des types de données complexes.

2. **Passage de schéma XML vers diagramme de classe UML**

Les XSD générés lors de l'étape précédente est complexe pour comprendre la structure des données des sources XML. Il est difficile d'avoir une compréhension rapide et globale d'un vocabulaire XML basé sur un schéma XML. C'est pourquoi, il est intéressant de représenter ce vocabulaire en UML. Ce dernier avec sa notation graphique permet d'exprimer visuellement une solution objet.

3. **Génération de schéma multidimensionnel**

Nous avons présenté un ensemble de règles pour la conversion de diagramme UML, généré à partir de source XML en un ou plusieurs diagrammes UML étoile. Nous avons proposé des règles pour l'extraction des propriétés multidimensionnelles (faits,dimension,... ..), puis les représentées en un schéma en étoile.

4. **Génération de schéma XML en étoile**

Nous avons présenté un ensemble des règles de transformation des entités du diagramme de classe UML (telles que les classes, les attributs, les associations, etc.) en leurs équivalents dans XML schéma.

La figure4.1, montre le processus global de construction de l'entrepôt de données XML :

1. Processus de modélisation conceptuelle et logique de l'entrepôt (XStar-Warehouse), (objet de ce mémoire),
2. ETL : extraction et transformation pour le chargement dans l'entrepôt,
3. Implémentation physique et stockage dans un SGBD XML.

Notre problématique s'articule autour de l'entreposage des données XML, à des fins d'analyse. Nous nous intéressons plus particulièrement à la modélisation conceptuelle et logique. Pour cela en suppose que le schéma globale décrivant les schémas XML source, puis génére un schéma XML en étoile passant par UML et récupération des concepts multidimensionnels(le fait, les

dimensions). Pour la première étape concernant la construction de schéma XML globale, dans le cas où il existe plusieurs schémas, cette étape nécessitant l'intervention d'expert de domaine. Dans notre cas nous avons intérissé de définir un schéma XML présente une sources base de données relationnelle.

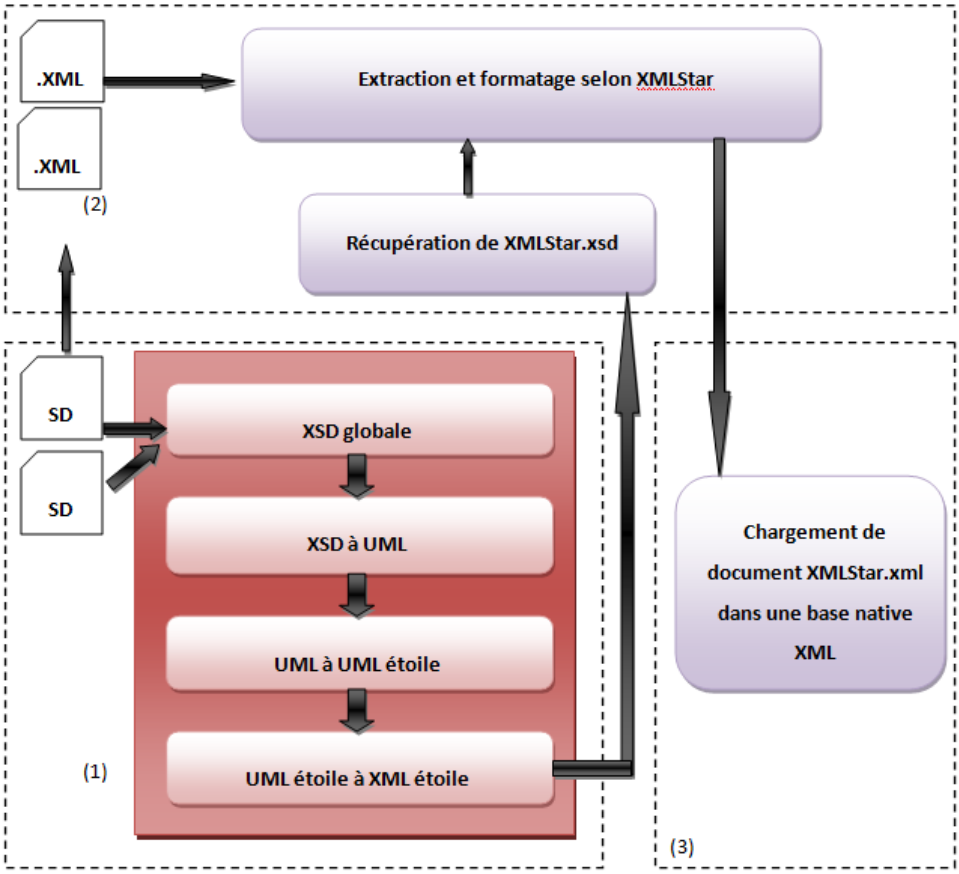


FIGURE 4.1 – Processus de conception de l'entrepôt de données XML : XStar-Warehouse .

4.2.1 Génération de schéma XML à partir d'une source

Pour la construction d'un entrepôt de données XML, la première étape de notre approche consistent en la définition un type de source, on à choisi "une Base de Données Relationnelle", qui va transformer à XML schéma.

Pour générer un schéma XML à partir d'un schéma relationnel, il convient de [Bou03] :

- **Créer un type d'élément par table.**
- **Pour chaque colonne de cette table qui ne soit pas une clé et pour la(les) colonne(s) correspondant à la clé primaire, ajouter un attribut au type d'élément ou ajouter un élément fils de type PCDATA seul à son modèle de contenu.**
- **Pour chaque table pour laquelle la clé primaire est exportée, ajouter un élément fils au modèle de contenu, puis traiter la table récursivement.**

- **Pour chaque clé étrangère, ajouter un élément fils au contenu du modèle et traiter récursivement la table de la clé étrangère.**

4.2.2 Passage de schéma XML vers diagrame de classe UML

Les schémas XML sont assez complexes pour comprendre la structure des données des sources XML, il est difficile d'avoir une compréhension rapide et globale d'un vocabulaire XML basé sur un schéma XML. C'est pourquoi il est intéressant de représenter ce vocabulaire en UML. Ce dernier avec sa notation graphique permet d'exprimer visuellement une solution objet. UML est un langage de modélisation commun, précis qui peut être facilement compréhensible par tous les membres d'une équipe de développement d'applications. Il est très commode d'employer UML pour représenter la sémantique des schémas XML favorables pour l'analyse multidimensionnelle. La modélisation des données XML n'est pas standardisée, mais il y'a des outils pour la génération des modèles pour les données XML.(Altova XML Spy, oXygen XML editor,etc), et des traveaux de recherches.

Dans notre approche nous avons utilisé quelques règles de passage nécessaires dans notre contexte (entrepôt de données XML) pour générer un diagramme de classe UML à partir d'un schéma XML :

- **Pour chaque déclaration d'élément complexe et pour chaque déclaration de type complexe sauf la racine, créer une classe avec une colonne clé primaire. Le nom de cette classe sera celui de l'élément où du type complexe.**

- **Les d'attributs de type simple (prédéfinis) et les éléments fils simples d'un élément complexe seront transformés en des attributs de la classe UML correspondante.**

- **Pour chaque élément fils complexe, lier sa classe UML correspondante à la classe UML correspondante à l'élément parent par le biais de la clé primaire de la classe parente.**

- **Les types de données prédéfinis tel que** $< xs : int >$**,** $< xs : double >$**, et** $< xs : string >$ **seront transformés en leurs équivalents UML int, double, et string.**

- **Les attributs** $< xs : minOccurs >$**, et** $< xs : maxOccurs >$ **portés par l'élément sont transformés en multiplicités UML.**

- **Pour chaque attribut possédant plusieurs valeurs et pour chaque élément fils simple présentant plusieurs occurrences, créer une classe séparée pour contenir des valeurs : cette dernier sera liée à la classe parente par le biais de la clé primaire de celle-ci.**

XML Schéma	***UML***
Un élément XML $< xs : element >$ ayant une structure complexe $< xs : complexTeype >$	Classe UML
$< xs : element >$ ayant pour valeur d'attribut $< xs : type >$ un type de donnée	Les attributs UML
Attributs $< xs : minOccurs >$, et $< xs : maxOccurs >$	Les cardinalités UML
$< xs : int >$, $< xs : double >$, $< xs : float >$, $< xs : string >$	Les types de données natifs UML tels que int,double,float ,string

TABLE 4.1 – les règles de passages de schéma XML vers UML

- **Etude de cas**

La base de donnée relationnelle source :
Produit(CodeP,DesigP,CodeCAT#)
Fournisseur(CodeF,RaisonF,CodeVI #)
Ville(CodeVI,NomVI)
Achat(CodeP#,CodeF#,DateA,QteA,PrixUnit)
Catégorie(CodeCAT,Categorie)

Achat.xsd (schéma XML décrivant l'activité d'une Base de données Achat au sein d'une entreprise).

Pour bien comprendre le passage de XML schéma vers UML,nous fournissons un exemple en deux figures dont la première représente un schéma XML(Achat.xsd) ; et la deuxième le résultat de la transformation en un diagramme de classe UMLdecelui-ci, selon les règles de mapping décrites ci-dessus :

4.2.3 Génération de schéma multidimensionnel

Dans cette étape de notre démarche, il existe deux principales approches : les approches [Kim96] guidé par les besoins des utilisateurs décisionnels, et les approches de [GMR98b]guidé par le schéma de système de production. on se base sur les travaux de [Kim96], qui consistent à la transformation du modèle E/R vers un modèle multidimensionnel, donc on se base sur ces travaux et faisant le parallèle entre le modèle E/R et le digramme de classes UML. Les types d'entités du modèle E/R sont modélisés en tant que classes. Les entités du modèle E/R correspondant à des objets en UML, etc. Nous avons proposé une démarche de conversion de diagramme UML généré à partir de sources XML en un diagramme UML en étoile. La démarche se décompose en trois étapes :

- **La séparation des diagrammes de classe UML en plusieurs diagrammes représentant chaque domaine (cette étape nécessite une intervention d'expert de domaine).**

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="achats">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="ville">
 <xs:complexType>
 <xs:sequence>
<xs:element type="xs:string" name="nomv"/>
<xs:element name="fournisseur" minOccurs="1" maxOccurs="unbounded" >
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="raisonf"/>
</xs:sequence>
<xs:attribute type="xs:string" name="codef" use="Required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="codev" use="Required"/>
</xs:complexType>
</xs:element>
<xs:element name="categorie">
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="nomcat"/>
<xs:element name="prouduit" minOccurs="1" maxOccurs="unbounded" >
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string"  name="designiation""/>
<xs:element name="Achat" minOccurs="1" maxOccurs="unbounded" >
<xs:complexType>
<xs:sequence>
<xs:element type="xs:integer" name="quantité"/>
<xs:element type="xs:float" name="prix"/>
<xs:element type="xs:string" name="codef"/>
<xs:element type="xs:date"   name="date" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="codeP" use="Required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="codecat" use="Required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

- **Extraction des concepts multidimensionnels (ex :Choisir les relations multiples contenant des faits numériques).**

- **Dénormaliser toutes les classes restantes dans les classes plates avec les clefs uniques qui se relient directement aux classes des faits. Ces classes deviennent des classes de dimensions.**

4.2.3.1 Extraction des concepts multidimensionnels

Dans cette section nous décrivons succinctement les étapes de notre approche de construction de schéma multidimensionnel candidats à partir de diagramme UML générer précédemment, en exploitant les relations entre les classes ainsi que les types de leurs attributs (numériques, temporelle,...).

- **Détermination de fait**

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 :

- Les représentations conceptuelles (entités ou association n-aire)[Kim02].

- Les associations (*,*) non porteuses de données dans le diagramme de classe UML [**?**].

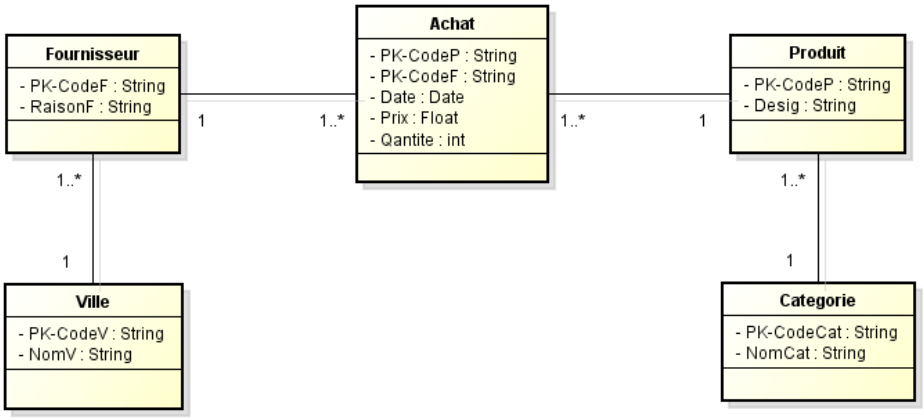


FIGURE 4.2 – Le passage d'un schéma XML vers un diagramme de classe UML.

Nous identifions les faits candidats à partir d'une classe contenant un attribut numérique non identifiant et contenant au moins une relation (1,*) vers une autre classes.

Nous pouvons améliorer ces règles par le critère suivant : un objet fait est un objet fréquemment mis à jour dans la source objet(un objet fait qui contient des attributs dynamiques)[Kim02]. Il s'agit d'un objet de transaction [GMR98b].

- **Détermination des mesures**

Généralement les mesures sont des attributs numériques qui permettent l'agrégation des données. Dans cette démarche, les données de type numériques sont considérées comme des mesures d'un fait donné F.

S'il existe une relation un-a-plusieurs entre le fait F et une classe C1 et que cette dernière est reliée par une relation a-un à une autre classe C2, alors tous les attributs numériques de C2(à l'exception des clés primaires) sont des mesures de fait 4.3.

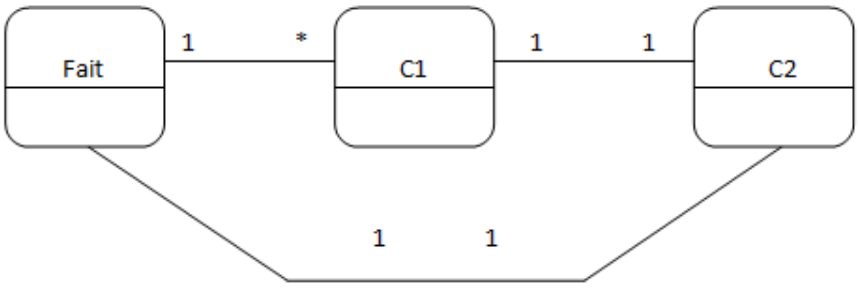


FIGURE 4.3 – Masures potentielles.

- **Détermination des dimensions**

Le sujet ou fait est observé et analysé suivant différentes dimensions, une dimension modélise un axe d'analyse et se compose d'attributs correspondant aux informations faisant varier les mesures de l'activité. Certains des attributs (dits paramètres) sont ordonnées pour former des perspectives d'analyse (i.e hiérarchies).

Dans notre méthode, les dimensions sont extraites à partir des relations avec le fait choisi, ou à partir d'un attribut d'un fait (ex : attribut temporel) :

Soit C1, C2, C3 un ensemble de classes d'un diagramme de classe UML tel que C1 est supposé être le fait. Deux cas de figure peuvent se présenter pour que C2 et C3 soient des dimensions de C1 4.4 :

- **Si C2 est reliée à C1 par une relation un-à-plusieurs (de cette manière la C2 détermine fonctionnellement le fait C1) ;**

- **Si C2 et reliée a C3 par une relation un-a-un, alors C3 sera une dimension de C1.**

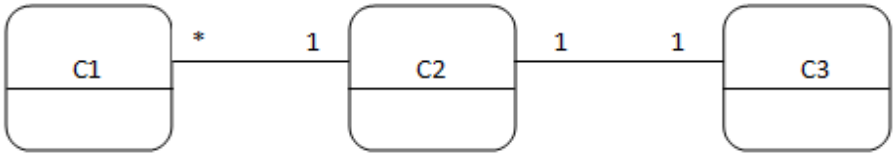


Figure 4.4 – Dimension potentielle d'un fait.

La figure suivante montre un exemple de passage d'un diagramme de classe UML à un diagramme de classe UML en étoile.

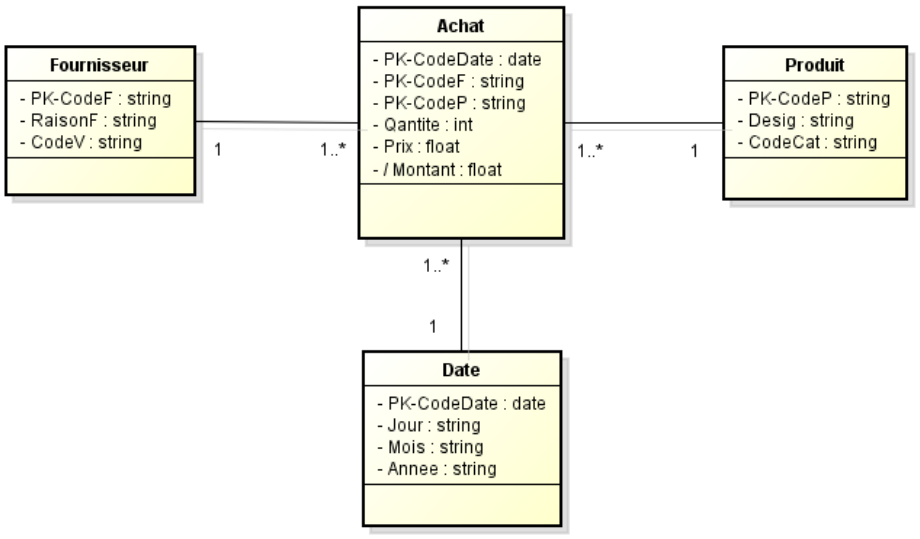


FIGURE 4.5 – Diagramme UML Star.

4.2.4 De l'UMLStar vers L'XML

Dans ce qui suit, nous présentons un ensemble de régles de transformation des entités du diagramme de classe UML (telles que les classes, les attributs, les associations, etc.)en leurs équivalents dans XML schéma. leur but était de décrire le mapping UML vers XML.

4.2.4.1 Mapping des classes UML

Une classe UML est transformée en un élément XML($< xs : element >$)et une déclaration de type complexe assortie ($< xs : complexType >$) dans XML schéma. Les complexType XML peuvent porter des attributs, d'autres éléments, des relations comme c'est le cas des classes UML.

4.2.4.2 Mapping des types de données UML

Les types de données encastrés comme integer (int), double et string sont représentés dans les schémas XML par les notations XSD standards, telles que $< xs : int >$, $< xs : double >$, et $< xs : string >$. Les types de données définis par l'utilisateur seront balisés dans UML en utilisant le stéréotype UML « XSDsimpleType ». Dans le schéma XML équivalent, de nouveaux types simples sont définis en les dérivant des types simples existants (encastrés et dérivés). La plage de valeurs que le nouveau type peut prendre est un sous-ensemble de la plage valeurs du type existant. L'élément $< xs : simpleType >$ et utilisé pour définir et nommer le nouveau type simple. L'élément $< xs : restriction >$ est utilisé pour indiquer le type existent (base), et identifier des facettes (telles que pattern et length) qui contraignent les plages de valeurs. Cette solution suppose que tout type de données encastré en UML a un type de données équivalent en XML. Et si un type de données existent n'est pas trouvé, il est traité comme un type de données défini par l'utilisateur. Les types de données énumérés sont dérivés par restriction, toute documentation est représentée par des annotations.

4.2.4.3 Mapping des attributs UML

En général, un attribut UML est transformé en un élément XML renfermé dans les balises $< xs : sequence >$ et c et avec attribut " type" pour indiquer son type de données. Si la validité du schéma XML n'est pas importante, $< xs : all >$ remplace $< xs : sequence >$. Si l'attribut est précédé par le stéréotype « XSDattribute » dans le modèle UML, il est converti en un attribut XML représenté par $< xs : attribute >$.

4.2.4.4 Mapping des attributs dérivés UML

Un attribut dérivé dans UML est soit un attribut précédé par '/', soit une méthode UML précédée par le stéréotype «XSDDerivedAttribute». La notation XML équivalente serait une définition d'élément XML avec des attributs "name" et "type" pour le type de données.

4.2.4.5 Mapping des associations générales

nous représentons une entité XML équivalente dans les deux classes impliquées par l'association générale. Pour chaque classe, celle-ci serait un élément XML avec les attributs name "association", ID, " ASSOCIATION-rolename, classename" et un attribut de référence pointant vers l'autre classe. Si les noms de rôle ne sont pas présents dans le modèle UML. La notation ID "ASSOCIATION-classenames" est utilisé (ASSOCIATION-A-B dans la classe A et ASSOCIATION-B-A dans la classe B. Sinon ID "ASSOCIATION-rolename" est utilisée.

Si les contraintes de multiplicité sont présentes dans le modèle UML, elles seront représentées par les attributs XML minOccurs et maxOccurs. Sinon * est utilisé dans les deux classes reliées par l'association.

4.2.4.6 Mapping d'agrégations

l'agrégation est représentée par une déclaration d'élément XML dans la classe agrégat avec des attributs : name "agrégation", ID "AGGREGATION-A-B" (A est la classe agrégat) et un attribut ref pointant vers la classe agrégée (i.e la classe B).

Si la contraintes de multiplicité sont présentes dans le modèle UML, elles seront représentés par les attributs minOccurs et maxOccurs dans le schéma XML. Si par contre, elles ne le sont pas. * est supposé au niveau de l'agrégat et 1 au niveau de la classe agrégée.

4.2.4.7 Mapping des compositions

Une composition est représentée par une déclaration d'élément XML dans la classe composite, avec des attributs : name "composition", ID "COMPOSITION-A-B" (A est la classe composite) et un attribut ref pour la classe composante (i.e la classe B).

Les remarques sur le mapping d'agrégations restent valables pour les compositions puisque ces dérniéres ne sont qu'un cas particulier d'agrégation.

4.3 Formalisme

Nous formalisons le schéma en étoile de par les définitions suivantes :
Définition1 (XStar-Warhouse) : Le schéma en étoile de l'entrepôt XStar-Warhouse est défini par le Quintuple (Noms,Fs,Ds,MRs) où :

- Noms est le nom de l'entrepôt
- Fs f1,f2,....,fn est l'ensemble de faits (mesures).

- Ds d1,d2,....,dn est l'ensemble de m dimension où chaque D contient un ensemble d'attributs. D est un élément XML de type complexe de schéma XML en étoile,aui contient des éléments simples correspondant au attribut de la dimension.

- MRS est un ensemble de régles de mapping, visant à définir les différentes transformation des données sources vers schéma XML en étoile.

Définition2 (Objet XStar-Warehouse) :

une instance de schéma XStar-Warehouse définit précédemment est un document XML contenant les mesures et l'ensembles des éléments XML qui représentent les dimensions, valide conformément a modèle XML en étoile.
Définition3 (Dimension Hiérarchisée) :

Une hiérarchie de dimension H est un partie de schéma XML décrivant une dimension contenant plusieurs éléments de type $< xs : complexType >$ imbriqués (niveaux de détails), noté H. D1...,Dt,...Dl, un membre Di ($i >$1,$i <$l) de H est sous ensemble d'élément de schéma XML vérifiant les conditions suivante :

- membre de Di de H est schéma XML valide.

- la clé primaire de Di est un attribut (clé étrangère) dans Dt-1.

- les éléments de Di sont suffisants pour décrire un document XML de membre D.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema>
<xs:element name="Achat"/>
<xs:complexType name="Achat">
<xs:sequence>
<xs:element name="CodeDate" type=" xs:date"/>
<xs:element name="CodeF" type=" xs:string"/>
<xs:element name="CodeP" type=" xs:string"/>
<xs:element name="Qantite" type=" xs:int"/>
<xs:element name="Prix" type=" xs:float"/>
<xs:element name="Montant" type=" xs:float"/>
<xs:element name="association" id = "ASSOCIATION_Achat_Produit" />
<xs:element ref ="Achat" minOccurs = "1" maxOccurs = "unbounded" />
<xs:element name="association" id = "ASSOCIATION_Achat_Date" />
<xs:element ref ="Achat" minOccurs = "1" maxOccurs = "unbounded" />
<xs:element name="association" id = "ASSOCIATION_Achat_Fournisseur" />
<xs:element ref ="Achat" minOccurs = "1" maxOccurs = "unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Date"/>
<xs:complexType name="Date">
<xs:sequence>
<xs:element name="CodeDate" type=" xs:date"/>
<xs:element name="Jour" type=" xs:string"/>
<xs:element name="Mois" type=" xs:string"/>
<xs:element name="Annee" type=" xs:string"/>
<xs:element name="association" id = "ASSOCIATION_Date_Achat" />
<xs:element ref ="Date" minOccurs = "1" maxOccurs = "1" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Fournisseur"/>
<xs:complexType name="Fournisseur">
<xs:sequence>
<xs:element name="CodeF" type=" xs:string"/>
<xs:element name="RaisonF" type=" xs:string"/>
<xs:element name="CodeV" type=" xs:string"/>
<xs:element name="association" id = "ASSOCIATION_Fournisseur_Achat" />
<xs:element ref ="Fournisseur" minOccurs = "1" maxOccurs = "1" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Produit"/>
<xs:complexType name="Produit">
<xs:sequence>
<xs:element name="CodeP" type=" xs:string"/>
<xs:element name="Desig" type=" xs:string"/>
<xs:element name="CodeCat" type=" xs:string"/>
<xs:element name="association" id = "ASSOCIATION_Produit_Achat" />
```

Figure 4.6 – Le schéma XML en étoile est généré.

Conclusion et perspectives

XML est un format de plus en plus utilisé pour le stockage et la transmission d'information. Depuis que XML est devenu le langage de choix pour représenter les données, le besoin de pouvoir analyser ces données augmente constamment. Il convient donc de disposer d'outils adéquats pour stocker et interroger ces informations XML. Le stockage des ces données est le rôle d'un entrepôt de données et l'interrogation de celles-ci revient à des outils d'analyse OLAP et de data-mining.

Nous avons d'abord du nous familiariser aves les entrepôts de données et les technologies XML dans le domaine des bases de données : les différentes modélisations, les bases de données natives XML. Tout ceci étant récent et novateur.

Ensuite, nous avons étudié des différents articles publiés à propos de XML dans le domaine des entrepôts de données, qui est un domaine pratiquent vierge. Les travaux de recherche qu'y sont consacré sont de deux sort ; les premiers sont relative à l'entreposage de document XML. Il consiste à organiser le stockage de collections de documents XML essentiellement textuels de manière à optimiser la recherche d'information dans ces documents. Et le second type des travaux dans ce domaine consiste en des modèles XML de partage et de transmission d'entrepôt de données.

Finalement nous avons proposé une démarche de modélisation conceptuelle et logique d'un entrepôt de données XML basé sur XML et l'UML, appelée XU Etoile. Nous avons basé sur les bases de données relationnelles comme une source de base. Nous avons défini des correspondances entre XML et UML er vice-versa .

Nous savons que cette proposition n'est qu'un 1ier essai, les modifications et les améliorations sont toujours possibles. C'est la raison pour laquelle nous avons pensé à des améliorations et des perspectives futures :

- Définir un schéma XML global, unifiant les schémas des sources documents XML (DTD, XSD, document XML, BDR).

- Valider nos proposition par le développement d'un prototype.

Annexe A

Le format de stockage de données XML(eXtensible Markup language)

A.1 Introduction

Dans cette annexe , nous présentons le langage XML pour lequel nous allons proposer un modèle d'entrepôt de données XML.

A.2 Format de stockage de données XML

A.2.1 Définition

Le XML (Extensible Markup Language, « langage de balisage extensible ») est un langage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite extensible car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS... Elle est reconnaissable par son usage des chevrons (<>) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche...) entre systèmes d'informations hétérogènes (interopérabilité).

A.2.2 Composants et syntaxe d"un document XML valide

D'un point de vue formel, un document XML est un arbre. Un tel arbre accepte différentes incarnations (objet en mémoire, flux d'événements...), et surtout, une version texte. Les composants principaux de ce modèle sont les nœuds, qui peuvent être de différents types (éléments, attributs, commentaires...) présentés dans l'exemple suivant :

A.2.2.1 Le nœud document

Un document XML a toujours une et une seule racine, le noeud document. Dans le langage d'accès à un document XML, XPath, le noeud document est abrégé avec la barre oblique /, comme la racine de l'arborescence d'un système de fichiers Unix. La racine peut éventuellement comporter des enfants de type commentaire ou instruction de traitement, elle doit obligatoirement comporter un et un seul élément.

A.2.2.2 Les éléments

Un élément est un noeud, désigné par un nom qualifié au sein d'un espace de noms ¡espace : élément/¿ , pouvant contenir la plupart des autres noeuds : texte, éléments, attributs... (À

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <!-- Commentaire -->
  <ex:collection
    xml:lang="fr"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ex="http://exemple.org"

    <élément>Texte</élément>
    <dc:title>Astérix le Gaulois</dc:title>
    <ex:livre attribut="valeur" type="BD">
      <dc:title>Astérix chez les Belges</dc:title>
      <!-- élément répété -->
      <dc:creator>René Goscinny</dc:creator>
      <dc:creator>Albert Uderzo</dc:creator>
      <dc:description>
        <b>Astérix chez les Belges</b> est un album de
        <a href="http://fr.wikipedia.org/wiki/Bande_dessinée">bande dessinée</a>
        de la série Astérix le Gaulois créée par René Goscinny et Albert Uderzo.
        <br/><!-- élément vide -->
        Cet album publié en 1979 est le dernier de la série écrit par René Goscinny.
      </dc:description>
    </ex:livre>
  <ex:collection>
```

Figure A.1 – Exemple d'un document XML

l'exception du noeud document). Cette spécification formelle est à l'origine des particularités de XML comparé à d'autres formats.

A.2.2.3 Les balises

Une balise est un nom commode pour désigner les constructions entre deux chevrons ¡ ¿ dans un fichier XML. On distinguera les balises ouvrantes ¡élément attribut=”valeur”¿, les balises fermantes ¡/élément¿ (sans attribut et avec barre oblique au début), et les balises vides ¡élément attribut=”valeur”/¿ (barre oblique à la fin, attributs possibles). Il ne faut pas confondre les balises avec les éléments. Ces notations permettent de délimiter des éléments (ainsi que leurs attributs), mais les balises ne sont pas des noeuds dans le modèle abstrait du document.

A.2.2.4 Les attributs

Les attributs sont toujours associés aux éléments. Ils viennent en quelque sorte les qualifier. Un attribut est t une paire nom=valeur. Un élément ne peut pas avoir deux attributs de même nom. Un attribut doit toujours avoir une valeur, même si celle-ci est une chaîne vide.

A.2.2.5 Les commentaires

En XML, les commentaires sont délimités par ¡!– et –¿. Le contenu d'un commentaire ne sera pas interprété ¡!– cet ¡élément¿ n'est pas fermé mais cela est autorisé dans un commentaire –¿.La chaîne de caractères « – » ne peut apparaître dans le contenu du commentaire.

A.2.3 Document bien formé

Un document bien formé respecte les contraintes formelles définies ci-dessus. Certaines erreurs courantes se rencontrent particulièrement en (X)HTML, où les navigateurs sont plus tolérants qu'un processeur XML :

- Un fichier XML ne doit comporter que des caractères dans l'encodage déclaré (ex : pour un document UTF-8, certaines séquences sont interdites).
- Tout élément XML doit être fermé, les enchâssements forment un arbre strict, les chevauchements sont interdits.
- Les caractères de la syntaxe (ex : <, &), doivent être échappés s'ils ne servent pas à délimiter une balise ou une entité (à remplacer par l'entité, & lt ;=<, & amp ;=&) ou qu'ils ne sont pas dans un champ CDATA.

- Les éléments de commentaires et CDATA ne peuvent être enchâssés.
- Les noms (éléments et attributs) sont sensibles à la casse.
- Les valeurs d'attribut sont entre guillemets.
- Un document XML n'a qu'un seul élément racine.

A.2.4 Les DTD et les schemas XML

Les DTD et les XML schémas sont des langages utilises pour définir la structure des documents XML . Ils déterminent quels éléments peuvent être contenus dans un document XML, quels élément peuvent être imbriques dans d'autres, quelle valeur par défaut leurs attributs peuvent avoir, etc. A l'aide d'une DTD on d'un Schéma XML et du document XML correspondant, un analyseur syntaxique peut confirmer que le document est conforme à la structure et aux contraintes désirées. Un tel document est dit valide.

A.2.4.1 Les DTD

Une DTD est écrite selon la norme EBNF (Extended Backus Naur Form) et, à ce titre ,elle est donné héritière du langage SGML . Une DTD définit la structure du document à l'aide d'une liste d'éléments légaux . Le mot-clé !ELEMENT définit le type de l'élément et mot-clé ! ATTLIST précise ses attributs . La DTD permet également de déclarer le nombre de fois qu'un élément fils peut apparaitre dans un élément parent : un ou plus (+) , zéro ou plus (*) ou zéro ou un(?). Les attributs de type ID et IDREF permettent de créer des relations entre les attributs , à l'image des clés étrangères utilisées dans les systèmes relationnels. Les types d'éléments possibles sont les suivants :

- # PCDATA désigne les données textuelles devant être traitées par l'analyseur .
- # CDATA désigne les données textuelles qui ne doivent pas être traitées par l'analyseur.
- EMPTY signifie que l'élément ne peut rien contenir.
- ANY signifie que l'élément peut contenir ce que l'on veut.

Bien que la DTD soit l'un des langages de schémas le plus utilisé, elle a des limitations, parmi lesquelles on peut énumérer[Age04] :

- Une DTD définit très peu de types de données pour la validation du contenu. En fait , les DTD ont seulement le type de texte(chaine de caractère).
- Il est impossible d'intégrer d'autres définitions existantes puisque le support des namespaces différenciant les langages n'est pas supporté.
- Le langage n'est pas exprimé en XML, ce que est en contradiction avec la forme même du langage défini .

Du fait des limitations énumérer ci-dessus, d'autres schémas ont été proposés comme XML-Schéma.

A.2.4.2 Les XML Schema

Les schémas XML permettent, comme les DTD, de définir des modèles de documents. Il est ensuite possible de vérifier qu'un document donné est valide pour un schéma, c'est-à-dire respecte les contraintes données par le schéma. Les schémas ont été introduits pour combler certaines lacunes des DTD.

La première différence entre les schémas et les DTD est d'ordre syntaxique. La syntaxe des DTD est une syntaxe héritée de SGML qui est différente de la syntaxe XML pour le corps des documents. En revanche, la syntaxe des schémas est une syntaxe purement XML. Un schéma est, en effet, un document XML à part entière avec un élément racine xsd :schema et un espace de noms.

Un schéma XML se compose essentiellement de déclarations d'éléments et d'attributs et de définitions de types. Chaque élément est déclaré avec un type qui peut être, soit un des types prédéfinis, soit un nouveau type défini dans le schéma.

Parmi les types, les schémas XML distinguent les types simples introduits par le constructeur xsd :simpleType et les types complexes introduits par le constructeur xsd :complexType. Les types simples décrivent des contenus textuels, c'est-à-dire ne contenant que du texte. Ils peuvent être utilisés pour les éléments comme pour les attributs. Ils sont généralement obtenus par dérivation des types prédéfinis. Au contraire, les types complexes décrivent des contenus purs constitués uniquement d'éléments ou des contenus mixtes constitués de texte et d'éléments. Ils peuvent uniquement être utilisés pour déclarer des éléments. Seuls les types complexes peuvent définir des attributs.

Comme caractéristiques propos à XML Schema nous pouvons énumérer :

- Les schémas XML Schema sont écrits en utilisant le langage XML.
- XML Schema et les espaces de nommage(namespaces) :un schéma est constitué d'un ensemble de définitions de types et de déclarations d'élément. Les espaces de nommage permettent de fournir un contexte à un vocabulaire, ce qui facilite la création de schémas et la validation de documents.
- Contraintes d'occurences sur les éléments.
- Dérivation de types par restriction et extension.
- Réutilisation par importation et héritage.
- Davantage de souplesse dans les cardinalités.

Dans un schéma, les types sont fondamentaux. Ils ont la caractéristique d'être disponibles à la fois pour les éléments et les attributs. Examinons, tout d'abord, le typage des éléments. Pour cela, passons en revue les contenus possibles pour un élément :

- Vide (entraîne EMPTY avec une DTD) : pas de contenu.
- Simple (entraîne (#PCDATA) avec une DTD) : du texte.
- Complexe : un ou plusieurs éléments.
- Mixte : un mélange d'éléments et de texte.

A.2.5 Le langage d"interrogation des documents XML

De nombreux langages de requêtes ont fait objet de travaux pour l"interrogation des données semi-structurées particulièrement des documents XML. Parmi ceux-ci nous pouvons citer. SGMLQL réalisent des requêtes sur des documents LOREL, SGML permettant la construction de graphes généralisés sur le modèle de données OEM. Puis, XPath, un langage de requête simple pour exprimer des chemins dans un document. Plusieurs autres langage de requêtes plus complexes ont été proposés tel que XML-QL, XQL, QUILT, CDuce, XyQL, et enfin XQuery qui est devenu la standard du W3C pour l"interrogation de document XML, qu'on va utiliser dans notre approche pour l"extraction des données et les charger dans l'entrepôt. Et c'est pour cela que nous avons décidé de nous attarder un petit peu plus sur ce langage.

A.2.5.1 XPath

Le langage d'expression de chemins XPath, est utilisé localiser les éléments d'un document XML. Initialement créé pour fournir une syntaxe et une sémantique aux fonctions communes à XPointer et XSL, XPath a rapidement été adopté par les développeurs comme langage d'interrogation simple d'emploi.

Une expression XPath est un chemin de localisation, constitué de pas de localisation. Les pas de localisation sont séparés par le caractère « / ». Un chemin ressemble ainsi au chemin dans un système de fichiers.

A.2.5.2 XSLT

XSLT (Extensible Stylesheet Language Transformations) est un langage XML qui sert à passer d'un format XML à un autre format texte (XML, XHTML/HTML, CSV...).XSLT est conçu pour etre utilisé indépendamment de XSL. Cependant, XSLT n'est pas censé être utilisé comme un langage de transformation XML à vocation générale. Il a surtout été conçu pour les types de transformations nécessaires lorsque XSLT est utilisé comme une partie de XSL.

A.2.5.3 XQuery

XQuery est le langage aujourd'hui utilisé pour l'interrogation de documents XML. Il est issu de Quilt proposé par IBM et certains auteurs de XML-QL. Le premier document publie de travail du W3C. Il permet d'exprimer des sélections des chemins dans un document XML grâce à XPath. Chaque sélection est assignée à une variable définissant un ensemble d'arbres. Ces ensembles peuvent être manipulés par des contraintes, des ordonnancements, des constructions de résultat, des imbrications de requêtes, des quantificateurs, des agrégats, des séquences, des opérations ensemblistes et conditionnelles.

Bibliographie

[BL05] Rajesh Bordawekar and Christian A. Lang. *Analytical processing of XML documents : opportunities and challenges*, volume 34. ACM SIGMOD Record, 2005.

[BMCA06] Boussaid, Riadh Ben Messaoud, Rémy Choquet, and Stéphane Anthoard. X-warehousing an xml-based approach for warehousing complex data. In *10th East European Conf. on Advances in Databases and Information Systems (ADBIS) LNCS 4152*, page 39â“54, 2006.

[Bou03] R. Bourett. Xml et les bases de données. 2003.

[Bou09] Omar Boussaid. Conférence modélisation et analyse en ligne des objets complexes. In *Journée thématique sur la Simulation, Intégration et Fouille de données .*, université d´Oran, 2009.

[bs12] Benabderrahmane billel and Heddar sofiane. Implémentation d´une mesure d´agrégation textuelle dans les entrepots de documents xml pour la classification. Master's thesis, ESI Alger, 2012.

[CF07] O. Boussaid C. Favre, F. Bentayeb. *Evolution et personnalisation des analyses dans les entrepots de données : une approche orienté utilisateur.* 2007.

[Che76] Shan Chen. The entity-relationship model-toward a unified view of data. *ACM Trans*, 1 :9–36, March 1976.

[Cod93] Codd. Providing olap (on line analytical processing). rapport technique, an IT mandate, 1993.

[DM98] Devlin and Murphy. An architecture for a business and information system. *IBM System Journal*, pages 60–80, 1998.

[GMR98a] Golfarelli, Dario Maio, and Stefano Rizzi. Conceptual design of data warehouses from e/r schema. In *In Proceedings of the Thirty- First Annual Hawaii International Conference on System Sciences*, page 334, Washington, DC, USA, 1998.

[GMR98b] Golfarelli, Dario Maio, and Stefano Rizzi. *Conceptual Design of Data Warehouses from E/R Schema.* IEEE Computer Society, 1998.

[GMR98c] Golfarelli, Dario Maio, and Stefano Rizzi. *The Dimensional Fact Model : A Conceptual Model For Data Warehouses.* IEEE Computer Society, 1998.

[GRV01] Golfarelli, Stefano Rizzi, and Boris Vrdoljak. *Data Warehouse Design from XML Sources.* ACM Press, 2001.

[HBH03] Hummer, Andreas Bauer, and Gunnar Harde. *XCube : XML for data warehouses.* ACM Press, 2003.

[Inm92] Inmon. *Building the data warehouse*, chapter 1, pages 15–16. Wellesley, 1992.

[Inm96] Inmon. *Building the Data Warehouse.* 1996.

[Kec11] Mohamed Kechar. Fragmentation d'un entrepot de données xml sécurisé par xacml. Master's thesis, Université Es-Sénia Oran, 2011.

[Kim96] Kimball. *The data warehouse toolkit : Practical Techniques for Building Dimensional Data Warehouses.* 1996.

[Kim02] Kimball. *The data warehouse toolkit : The complete guide to dimensional modeling*, volume 2, chapter 1, pages 16–18. 2002.

[LA05] Li and Aijun An. *Representing UML Snowflake Diagram from Integrating XML Data Using XML Schema.* IEEE Computer Society, 2005.

[MJ04] Zhang M. and Yao J. *XML Algebras for Data Mining*, volume 5433 de SPIE proceedings series. in Data Mining and Knowledge Discovery : Theory, Tools, and Technology VI, USA, 2004.

[MMD09] Mahboubi.H, Hachicha. M, and Darmont.J. *XML Warehousing and OLAP*. 2009.

[Moi77] Moigne. *Théorie du systéme général - Théorie de la modélisation*, volume 2. 1977.

[Mor05] Lujan Mora. *Data Warehouse Design with UML*. PhD thesis, Université de Alicante, June 2005.

[PB04] Pouliot and Yvan Bédard. Expérimentation d´une approche incrémentielle de gestion et d´échange de mises á jour de données geospatiales. *Centre de recherche en géomatique, Département des sciences géomatiques, université laval Sainte-Foy, Quebec*, 58(2) :119–132, 2004.

[PHS05] Park, Hyoil Han, and Il-Yeol Song. Xml-olap a multidimensional analysis framework for xml warehouses. In *7th Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK)*, page 32â“42, 2005.

[PJ99] Pedersen and Christian S. Jensen. *Multidimensional Data Modeling for Complex Data*, pages 336–345. ICDE, 1999.

[Pok01] Jaroslav Pokorny. *Modelling Stars Using XML*. ACM Press, 2001.

[PRP02] Pedersen, Karsten Riis, and Torben Bach Pedersen. *Query optimization for OLAP-XML federations*. 5th ACM Intl. workshop on Data Warehousing and OLAP (DOLAP), ACM Press, 2002.

[Riz06] Rizzi. Research in data warehouse modeling and design : dead or alive ? *In DOLAP*, pages 3–10, 2006.

[RRT04] Rusu, J. Wenny Rahayu, and David Taniar. On building xml data warehouses. In *5th Intl. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL)*, page 293â“299, 2004.

[RT99] Ravat and Test. Towards data warehouse design. In *8th International Conference on Information and Knowledge Management*, 1999.

[SHD98] Markus Blaschka Sapia, Gabriele Hofling, and Barbara Dinter. Extending the e/r model for the multidimensional paradigm. *In ER Workshops*, pages 105–116, 1998.

[Sul01] Dan Sullivan. *Document Warehousing and Text Mining*. 2001.

[TBC99] Tryfona, Frank Busborg, and Jens G. Borch Christiansen. starer : A conceptual model for data warehouse design. *In DOLAP*, pages 3–8, 1999.

[TP98] Trujillo and Manuel Palomar. An object-oriented approach to multidimensional database conceptual modeling. *In DOLAP*, pages 16–21, 1998.

[Tru01] Jaime Gomez et Il-Yeol Song Trujillo, Manuel Palomar. Designing data warehouses with oo conceptual models. *IEEE Computer*, 34 :66–75, 2001.

[XZ03] Baril X. and Bellahsène Z. *Designing and Managing an XML Warehouse*. XML Data Management : Native XML and XML-enabled Database Systems, Addison Wesley, 2003.