AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur : ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite de ce travail expose à des poursuites pénales.

Contact : portail-publi@ut-capitole.fr

LIENS

Code la Propriété Intellectuelle – Articles L. 122-4 et L. 335-1 à L. 335-10
Loi n°92-597 du 1er juillet 1992, publiée au *Journal Officiel* du 2 juillet 1992
http://www.cfcopies.com/V2/leg/leg-droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 1 Capitole (UT1 Capitole)*

Présentée et soutenue le *05 décembre 2013* **par :**

FATEN ATIGUI

Approche dirigée par les modèles pour l'implantation et la réduction d'entrepôts de données

---

JURY

CLAUDE CHRISMENT	Professeur, Université Toulouse 3	Président du jury
RAFIK BOUAZIZ	Professeur, Université de Sfax	Examinateur
JEAN-MICHEL BRUEL	Professeur, Université Toulouse 2	Examinateur
MOURAD OUSSALAH	Professeur, Université de Nantes	Examinateur
JÉRÔME DARMONT	Professeur, Université Lyon 2	Rapporteur
FRANÇOIS PINET	Directeur de Recherche, Irstea Clermont-Ferrand	Rapporteur
GILLES ZURFLUH	Professeur, Université Toulouse 1	Directeur
FRANCK RAVAT	Professeur, Université Toulouse 1	Co-directeur

École doctorale et spécialité :
MITT : Image, Information, Hypermedia
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeur(s) de Thèse :
Gilles ZURFLUH et *Franck RAVAT*
Rapporteurs :
Jérôme DARMONT et *François PINET*

Remerciements

Je tiens à remercier très sincèrement Claude CHRISMENT Professeur à l'Université Toulouse 3, Josiane MOTHE Professeur à l'ESPE de Toulouse et Gilles ZURFLUH Professeur à l'Université Toulouse 1, qui ont tous trois dirigé l'équipe « Système d'Informations Généralisées » (SIG) de l'IRIT, pour m'avoir si bien accueillie au sein de leur équipe.

J'adresse mes remerciements les plus sincères à Monsieur Gilles ZURFLUH, pour avoir dirigé et encadré cette thèse, pour sa rigueur scientifique, pour ses critiques constructives ainsi que pour ses précieux conseils. Je le remercie aussi pour la confiance qu'il m'a accordée, pour son soutien et ses encouragements. Je le remercie d'autant plus pour ses qualités d'écoute et sa bonne humeur. Qu'il soit assuré de ma profonde reconnaissance et de mon très grand respect.

Je remercie Franck RAVAT, qui a codirigé la thèse, pour le temps qu'il m'a consacré lors de la rédaction des articles, pour sa lecture attentive de mon mémoire et pour la préparation de la soutenance.
Je n'oublierai pas Olivier TESTE pour ses remarques pertinentes et sa bonne humeur tout au long de la préparation de ma thèse.

Je tiens à mentionner le plaisir et l'honneur que m'ont fait Messieurs Jérôme DARMONT, Professeur à l'Université Lyon 2 et François PINET Directeur de Recherche à Irstea de Clermont-Ferrand, qui ont accepté d'évaluer cette thèse et d'en être rapporteur.

Je remercie également Messieurs Claude CHRISMENT Professeur à l'Université Toulouse 3, Rafik BOUAZIZ Professeur à l'Université de Sfax, Jean-Michel BRUEL Professeur à l'Université Toulouse 2 et Mourad OUSSALAH Professeur à l'Université de Nantes d'avoir accepter d'être membres de mon jury de thèse.

Résumé

Nos travaux se situent dans le cadre des systèmes d'aide à la décision reposant sur un Entrepôt de Données multidimensionnelles (ED). Un ED est une collection de données thématiques, intégrées, non volatiles et historisées pour des fins décisionnelles. Les données pertinentes pour la prise de décision sont collectées à partir des sources au moyen des processus d'Extraction-Transformation-Chargement (ETL pour Extraction-Transformation-Loading). L'étude des systèmes et des méthodes existants montre deux insuffisances. La première concerne l'élaboration d'ED qui, typiquement, se fait en deux phases. Tout d'abord, il faut créer les structures multidimensionnelles ; ensuite, il faut extraire et transformer les données des sources pour alimenter l'ED. La plupart des méthodes existantes fournit des solutions partielles qui traitent soit de la modélisation du schéma de l'ED, soit des processus ETL. Toutefois, peu de travaux ont considéré ces deux problématiques dans un cadre unifié ou ont apporté des solutions pour automatiser l'ensemble de ces tâches. La deuxième concerne le volume de données. Dès sa création, l'entrepôt comporte un volume important principalement dû à l'historisation régulière des données. En examinant les analyses dans le temps, on constate que les décideurs portent généralement un intérêt moindre pour les données anciennes.

Afin de pallier ces insuffisances, l'objectif de cette thèse est de formaliser le processus d'élaboration d'ED historisés (il a une dimension temporelle) depuis sa conception jusqu'à son implantation physique. Nous utilisons l'Ingénierie Dirigée par les Modèles (IDM) qui permet de formaliser et d'automatiser ce processus ; ceci en réduisant considérablement les coûts de développement et en améliorant la qualité du logiciel. Les contributions de cette thèse se résument comme suit :

1. Formaliser et automatiser le processus de développement d'un ED en proposant une approche dirigée par les modèles qui inclut :
   - un ensemble de métamodèles (conceptuel, logique et physique) unifiés décrivant les données et les opérations de transformation.
   - une extension du langage `OCL` (`Object Constraint Langage`) pour décrire de manière conceptuelle les opérations de transformation d'attributs sources en attributs cible de l'ED.

- un ensemble de règles de transformation d'un modèle conceptuel en modèles logique et physique.
- un ensemble de règles permettant la génération du code de création et de chargement de l'entrepôt.

2. Formaliser et automatiser le processus de réduction de données historisées en proposant une approche dirigée par les modèles qui fournit :
   - un ensemble de métamodèles (conceptuel, logique et physique) décrivant les données réduites,
   - un ensemble d'opérations de réduction,
   - un ensemble de règles de transformation permettant d'implanter ces opérations au niveau physique.

Afin de valider nos propositions, nous avons développé un prototype comportant trois parties. Le premier module réalise les transformations de modèles vers des modèles de plus bas niveau. Le deuxième module transforme le modèle physique en code. Enfin, le dernier module permet de réduire l'ED.

Mots clés : Entrepôt de données, Multidimensionnel, Extraction-Transformation-Chargement, Réduction de données, Ingénierie Dirigée par les Modèles

Abstract

Our work handles decision support systems based on multidimensional Data Warehouse (DW). A Data Warehouse (DW) is a huge amount of data, often historical, used for complex and sophisticated analysis. It supports the business process within an organization. The relevant data for the decision-making process are collected from data sources by means of software processes commonly known as ETL (Extraction-Transformation-Loading) processes. The study of existing systems and methods shows two major limits. Actually, when building a DW, the designer deals with two major issues. The first issue treats the DW's design, whereas the second addresses the ETL processes design. Current frameworks provide partial solutions that focus either on the multidimensional structure or on the ETL processes, yet both could benefit from each other. However, few studies have considered these issues in a unified framework and have provided solutions to automate all of these tasks. Since its creation, the DW has a large amount of data, mainly due to the historical data. Looking into the decision maker's analysis over time, we can see that they are usually less interested in old data. To overcome these shortcomings, this thesis aims to formalize the development of a time-varying (with a temporal dimension) DW from its design to its physical implementation. We use the Model Driven Engineering (MDE) that automates the process and thus significantly reduce development costs and improve the software quality. The contributions of this thesis are summarized as follows :

1. To formalize and to automate the development of a time-varying DW within a model-driven approach that provides :
   - A set of unified (conceptual, logical and physical) metamodels that describe data and transformation operations.
   - An OCL (Object Constraint Language) extension that aims to conceptually formalize the transformation operations.
   - A set of transformation rules that maps the conceptual model to logical and physical models.
   - A set of transformation rules that generates the code.
2. To formalize and to automate historical data reduction within a model-driven approach that provides :

– A set of (conceptual, logical and physical) metamodels that describe the reduced data.
– A set of reduction operations.
– A set of transformation rules that implement these operations at the physical level.

In order to validate our proposals, we have developed a prototype composed of three parts. The first part performs the transformation of models to lower level models. The second part transforms the physical model into code. The last part allows the DW reduction.

Key words : Data Warehouse, Multidimensional, Extraction-Transformation-Loading, Data Reduction, Mode-Driven Engineering

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Contexte et problématique

1.1 Introduction

Les systèmes décisionnels sont mis à disposition de l'entreprise pour permettre d'exploiter les données utiles pour la prise de décision. Ces systèmes doivent fournir un espace de stockage, typiquement un entrepôt de données, permettant de conserver de manière permanente un volume important de données issues des sources de production.
De nos jours, les entrepôts de données sont considérés comme un moyen essentiel pour garantir une meilleure prise de décision [Grabova et al., 2010]. Selon une étude menée dans le « `Data Warehouse Satifaction survey` [1] », environ 73% des utilisateurs qui ont participé à l'enquête admettent que l'entreposage de données a été utile pour la prise de décision et a permis de bien orienter les opérations commerciales. Cependant, la mise en œuvre d'un entrepôt de données reste une tâche complexe.

L'objectif de cette thèse est de répondre à cette complexité ; nous proposons une solution pour la conception et l'implantation automatiques d'un système décisionnel basé sur un entrepôt tout en réduisant la volumétrie des données mises à disposition des décideurs.

Ce chapitre est organisé de la façon suivante. Nous présentons dans la section 1.2 le contexte de notre étude en définissant ce qu'est un entrepôt de données ainsi que les processus permettant d'alimenter l'entrepôt à partir des sources. Nous abordons des aspects spécifiques de notre étude, à savoir la modélisation d'entrepôt de données et le problème du volume de données. Avant d'étudier notre problématique, nous présentons en section 1.3 un ensemble de définitions issu du domaine du génie logiciel portant sur les mécanismes de modélisation. En section 1.4, nous décrivons la problématique de recherche traitée dans le cadre

1. http ://www.information-management.com/specialreports/20071002/1093126-1.html

de cette thèse. La section 1.5 présente l'ensemble de nos contributions ainsi que la façon dont ce mémoire est organisé.

1.2 Systèmes décisionnels

Comme le montre la figure 1.1, dans un système décisionnel, un entrepôt contient des données extraites des systèmes de production d'une organisation (les sources) via des processus d'extraction de transformation et de chargement, couramment connus sous le nom des processus `ETL` (`Extraction-Transformation-Loading`). Dans cette section, nous présentons le cadre général de notre étude. Nous décrivons les différents concepts inhérents au développement d'un système décisionnel, à savoir : les sources, l'entrepôt et les processus `ETL`.

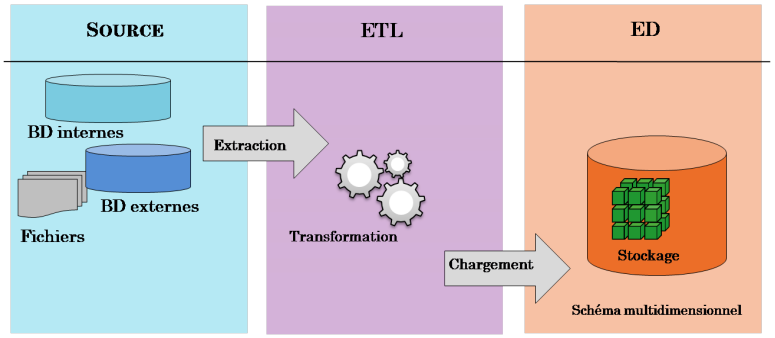


FIGURE 1.1 – Contexte de notre étude : entreposage de données

1.2.1 Source de données

Les sources à partir desquelles les données sont extraites et chargées dans l'entrepôt peuvent être :

- *internes* telles que les bases de production de l'entreprise ou,
- *externes* telles que les données provenant du Web, des méls, bases de partenaires, etc.

1.2.2 Entrepôts de données

Un entrepôt de données est défini comme une collection de données thématiques, intégrées, non volatiles et historisées pour des fins décisionnelles [Inmon, 1996]. IL permet de stocker les données pertinentes pour la prise de décision. A cette fin, les données de l'entrepôt sont souvent structurées selon un format particulier. Il s'agit du modèle multidimensionnel souvent utilisé dans les

bases de données décisionnelles. Ce modèle présente l'avantage d'être facilement compréhensible par les décideurs. Il organise l'information en termes de sujets d'analyse et d'axes d'analyse [Kimball, 1996]. Notons que dans la littérature, on parle aussi de bases de données multidimensionnelles ou de base décisionnelles, dans le cadre de cette thèse nous utilisons plutôt le terme « `Entrepôt de Données multidimensionnelles` » (`ED`). Le développement d'un ED est une tâche qui couvre deux aspects différents mais interdépendants que nous pouvons qualifier de `statique` et de `dynamique`. L'aspect statique présente la structure des données de l'entrepôt décrites via un schéma multidimensionnel. Quant à l'aspect dit dynamique, il correspond aux caractéristiques comportementales liées à l'extraction et à la transformation d'un attribut multidimensionnel de l'ED à partir des attributs sources.

1.2.2.1 Modélisation d'entrepôts de données

Les modèles de données d'un ED ont été conçus pour faciliter les analyses décisionnelles [Kimball, 1996]. Typiquement, la modélisation d'un ED repose sur trois niveaux d'abstraction : (1) le niveau conceptuel qui prend en compte les besoins des décideurs en faisant abstraction des aspects techniques, (2) le niveau logique où l'on fait le choix d'un modèle de stockage et (3) le niveau physique où l'ED est implanté sur un Système de Gestion de Base de Données (SGBD) particulier. Dans ce qui suit, nous présentons les pratiques les plus courantes lors de la modélisation d'un ED.

Niveau conceptuel

Au niveau conceptuel, les données de l'entrepôt sont représentées par des schémas en `étoile`, en `constellation` ou en `flocon`. Ces schémas organisent les données en termes de sujets d'analyse et d'axes d'analyses [Kimball, 1996]. Un sujet d'analyse appelé également `Fait` est composé par un ou plusieurs indicateurs d'analyse ; ce sont les `Mesures`. Les axes d'analyse appelés `Dimensions` sont composés de `Paramètres` en fonction desquels les mesures sont étudiées. Les paramètres sont organisés selon des `hiérarchies`, de la granularité la plus faible (paramètre racine) à la granularité la plus haute.

Niveau logique

À ce niveau, plusieurs possibilités sont envisageables pour modéliser la structure d'un entrepôt. Le plus courant est d'utiliser un modèle relationnel dit `ROLAP` (`Relational OnLine Analytical Processing`). Dans les modèles ROLAP, les faits et les dimensions du niveau conceptuel sont traduits en tables [Kimball, 1996]. Il existe deux variantes principales des modèles ROLAP ; ce sont les modèles ROLAP `dénormalisés` et ROLAP `normalisés`. Dans un schéma ROLAP dénormalisé, chaque fait et chaque dimension du schéma conceptuel sont traduits en tables. Dans la modélisation ROLAP normalisé, les tables de dimensions sont normalisées. Conformément aux hiérarchies associées aux dimensions, chaque niveau d'agrégation est transformé en une table. L'intérêt des modèles

ROLAP est la possibilité de réutiliser les principes de manipulation des bases de données relationnelles.

Niveau physique

Suite à une modélisation ROLAP, la modélisation physique consiste à choisir un mode d'implantation particulier (notamment le SGBD) et à définir des scripts `SQL` pour la création et l'alimentation de l'entrepôt.

1.2.2.2 Volume de données

Un entrepôt contient un volume important de données qui peut croître de manière régulière. En effet, les données historisées de l'entrepôt sont conservées de manière perpétuelle. De plus, les entrepôts ont tendance à « vivre » longtemps. Une étude menée dans [Agosta et al., 2007] a montré que plus de 56% des entrepôts de données ont été en production pendant plus de six ans et que 19% ont été en production depuis environ trois ans. Cette même étude a montré que non seulement 78% des ED dépasse le `téraoctet` mais aussi que le taux de croissance des ED de plus de 100 téraoctets atteint les 35%. Certes, les coûts de stockage ont diminué de façon sensible au cours des dernières années. Cependant, l'importance du volume de données qui doit être disponible en ligne, rend le stockage coûteux [Pöss and Potapov, 2003]. Evidemment, plus le volume des données est important, plus le temps de réponse aux requêtes des décideurs est important. Ainsi, la gestion et le contrôle de ces volumes représentent une problématique majeure.

1.2.3 Processus d'extraction, de transformation et de chargement

Les données pertinentes pour la prise de décision sont collectées à partir des sources et intégrées dans l'entrepôt. L'entreposage de ces données se fait par le biais des processus d'extraction de transformation et de chargement ou processus `ETL` [Vassiliadis, 2009]. Le processus de chargement des données à partir des sources vers l'entrepôt comporte trois étapes principales :

- **Extraction de données :** définit les sources de données (généralement hétérogènes) à utiliser pour alimenter l'ED.
- **Transformation de données :** une fois les données extraites, elles peuvent être transformées. Les opérations les plus courantes à ce niveau incluent des opérations de filtrage de données, de dérivation de valeurs, de transformation de formats de données, de génération automatique de numéros de séquence etc. Durant cette phase, différentes sources peuvent être fusionnées afin de charger l'ensemble des données dans une cible unique. Aussi, les éléments cible de l'ED à charger sont sélectionnés. Il est également indispensable d'établir les relations de correspondance entre les attributs sources et les attributs cibles.

– **Chargement de données :** c'est la dernière phase de l'alimentation d'un ED. Il s'agit d'insérer les données dans l'ED. Les tâches de chargement de l'entrepôt ont généralement lieu de façon périodique.

En ce qui concerne la conception des processus ETL, il n'y a pas un consensus sur la façon dont ils sont modélisés. En revanche, dans la littérature, à un niveau d'abstraction conceptuel, la modélisation des processus ETL vise à décrire et à formaliser les liens de correspondance entre les éléments cibles de l'ED et les éléments sources [Vassiliadis, 2009]. Au niveau logique, les processus ETL sont généralement modélisés par un ensemble d'opérations. Au niveau physique, ces modèles sont mis en œuvre sous forme de systèmes tels que `Oracle`[2] ou `Business Object`[3] permettant l'extraction, la transformation et le chargement des données dans l'ED.

1.3 Ingénierie Dirigée par les Modèles (IDM)

Face à la complexité et à la diversité des applications informatiques à développer, les recherches dans le domaine du génie logiciel ont abouti à un nombre important de méthodes et de langages de modélisation. Après le paradigme `Objet` où « Tout est objet », la communauté du génie logiciel s'est orientée vers le l'Ingénierie Dirigée par les Modèles (`IDM`) et le principe du « Tout est modèle » [Combemale, 2008]. Il s'agit d'un cadre où l'application est produite à partir de modèles. L'IDM a permis plusieurs améliorations significatives dans le développement logiciel en séparant la logique métier des plateformes techniques. L'idée phare de l'IDM est d'utiliser les modèles dans les différentes phases du cycle de développement de l'application et de fournir un ensemble de règles de transformation de modèles permettant d'aboutir au code final de façon quasi-automatique voire automatique.

L'Architecture Dirigée par les Modèles (`MDA` pour `Model Driven Architecture`) est la vision IDM de l'`Object Management Group`[4] (`OMG`). MDA s'appuie sur trois types de modèles. Le modèle des besoins, dit `CIM` pour `Computation Independent Model`, décrit les services que doit fournir l'application pour répondre aux besoins des utilisateurs. Le modèle d'analyse et de conception appelé `Platform Independent Model` (`PIM`) définit la structure et le comportement du système sans indiquer la plateforme d'exécution. Le modèle de code dit, `Platform Specific Model` (`PSM`) est la projection d'un PIM sur une plateforme donnée. Nous présentons plus de détails sur ces différents modèles dans la section 1.3.2. Dans cette section, nous citons les principes généraux de l'IDM. Ensuite, nous présentons les différentes parties de l'architecture dirigée par les modèles. Enfin, nous mettons en relief les avantages de l'approche IDM. Les différentes définitions présentées dans cette section sont établies en se référant

2. http ://www.oracle.com
3. http ://www.sap.com/pc/analytics/business-intelligence.html
4. http ://www.omg.org/

principalement aux travaux de [Bézivin and Gerbé, 2001], [Kleppe et al., 2003] et évidemment la spécification MDA de l'OMG [OMG, 2003]. Nous nous référons également aux travaux de synthèse sur l'IDM proposés par [Combemale, 2008] et [Diaw et al., 2010].

1.3.1 Principes généraux

Pour développer un système quelconque, l'IDM propose de décrire les connaissances métier de manière indépendante des plateformes techniques. Une démarche dirigée par les modèles fournit un ensemble de modèles qui permet de décrire différents aspects du système sous formes différentes selon les préoccupations de ses différents acteurs (concepteurs, programmeurs, utilisateur final, etc.). Un modèle se définit comme suit :

> **Définition 1** *Un `modèle` est une image simplifiée d'un système, autrement dit, une abstraction d'un aspect particulier d'un système. L'aspect système capturé dépend de l'objectif du modèle. [Bézivin and Gerbé, 2001] définit le modèle comme une simplification d'un système élaboré pour répondre à un objectif précis.*

Cette définition induit qu'un système est représenté par un ensemble de modèles, où chacun capture un aspect particulier. On dit qu'un modèle « `représente` » un système.

Remarque. Notons qu'en dehors des définitions dans un cadre dirigé par les modèles, par exemple dans un paradigme Entité/Association ou objet, on utilise plutôt le terme `Schéma` que `Modèle`. Ainsi dans ce mémoire de thèse, les deux termes sont considérés comme synonymes vu que modèle correspond bien à un schéma dans une vision dirigée par les modèles ; tous les deux appartiennent au même niveau d'abstraction.

Afin de comprendre et utiliser un modèle, on a besoin de la description de la sémantique de ses différents éléments (concepts, associations et graphique). Ceci se fait au travers d'un métamodèle défini comme suit :

> **Définition 2** *Un `métamodèle` est un modèle spécifique qui définit la syntaxe abstraite d'un langage de modélisation. [Bézivin and Gerbé, 2001] définit le métamodèle comme la spécification explicite d'une abstraction (simplification). Un métamodèle utilise un langage spécifique pour exprimer cette abstraction.*

Un modèle est dit « `conforme` » à son métamodèle si l'ensemble des éléments du modèle respecte les définitions du métamodèle. De la même manière qu'un métamodèle facilite la compréhension et l'utilisation d'un modèle, une description de la syntaxe du métamodèle est indispensable pour pouvoir l'instancier. Ceci se fait à travers d'un méta-métamodèle que l'on peut définir comme suit :

> **Définition 3** *Un `méta-métamodèle` est un modèle qui définit la syntaxe d'expression d'un métamodèle. Un méta-métamodèle doit être auto-descriptif, c'est-à-dire que l'on peut utiliser les concepts qu'il définit pour le décrire et le comprendre.*

Il existe aussi un concept souvent utilisé dans la communauté IDM, il s'agit de `Domain Specific Modeling Languages` (`DSML`) qui peut être défini comme suit :

> **Définition 4** *Un `Langage de Modélisation Spécifique au Domaine (DSML)` est un modèle qui décrit un domaine d'application particulier. Un DSML est défini par une syntaxe abstraite (un ensemble de concepts et d'associations) et une syntaxe concrète (graphique et textuelle).*

1.3.2 Architecture Dirigée par les Modèles (MDA)

MDA est la norme proposée par l'OMG pour le développement dirigé par les modèles. MDA préconise l'utilisation des modèles dans les différentes phases de développement logiciel. L'idée de base est de séparer la description de la logique métier de toute plateforme technique. Bien que cette idée ne soit pas nouvelle, privilégier les modèles contribue à cet objectif.

Architecture à quatre niveaux

L'OMG définit une architecture de modélisation à architecture à quatre niveaux présentée par la figure 1.2 :

1. Niveau M_0 : c'est le niveau des données réelles (concrètes) qu'on souhaite modéliser, ces données présentent une instance du modèle du niveau suivant M_1.
2. Niveau M_1 : c'est le niveau modèle qui permet de décrire les données du niveau M_0, typiquement un modèle UML appartient à ce niveau. Un modèle appartenant à ce niveau est décrit par un métamodèle du niveau M_2.
3. Niveau M_2 : à ce niveau sont décrits des métamodèles de description de langages, typiquement, le métamodèle UML.
4. Niveau M_3 : c'est le niveau méta-métamodèle. L'OMG définit un langage unique de définition de méta-métamodèle appelé le `Meta-Object-Facility (MOF)` [OMG, 2011a]. Le MOF est le méta-métamodèle qui permet de créer des métamodèles. Il permet aussi d'étendre ou de modifier un métamodèle existant. Le MOF est auto-descriptif, il permet de décrire sa propre sémantique.

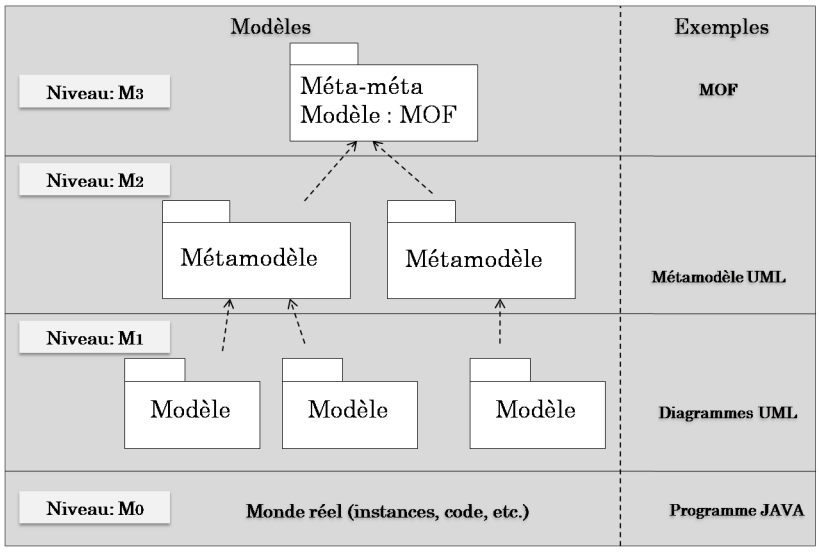


FIGURE 1.2 – Architecture de modélisation à quatre niveaux [OMG, 2003]

Modèles MDA

Le cycle de développement MDA ne semble pas être très différent du cycle de développement classique qui se base principalement sur les phases d'analyse, de conception, de codage, de test et de déploiement. En revanche, une différence majeure réside dans la nature des artefacts créés lors de la phase de développement. Dans MDA, les artefacts sont des modèles formels interprétables par machine. Les modèles sont au cœur de MDA [Kleppe et al., 2003]. Dans un premier temps, ces modèles vont servir à décrire le système. En outre, ces modèles vont servir à générer du code par transformations successives. La figure 1.3 montre les différents types de ces modèles que nous décrivons comme suit :

- CIM (Computation Independant Model) : est une vue système indépendante de tout calcul. Le CIM ne montre pas la structure du système. Ce modèle appelé aussi modèle métier présente un vocabulaire compréhensible par un spécialiste du domaine. Ce dernier ignore les modèles ainsi que tous les éléments utilisés pour réaliser ses besoins et ses exigences. Le CIM permet de rapprocher les visions des experts du domaine, de celles des concepteurs d'applications informatiques.
- PIM (Platform Indepent Model) : ce modèle indépendant de toutes plateformes techniques, définit le comportement et/ou la structure du système sans indiquer la plateforme d'exécution.
- PSM (Platform Specific Model) : ce modèle est spécifique à une plateforme technique particulière, il est fondamental pour générer du code. Un PSM est élaboré pour spécifier le système en termes d'éléments d'implémentation adaptés à une technologie spécifique.

– Code : représente le code de l'application généré à partir du PSM.
– Platform Model (PM) : appelé aussi Platform Description Model (PDM), ce modèle décrit une plateforme de manière à générer un PSM à partir d'un PIM et d'un PDM. Un modèle de plateforme fournit un ensemble de concepts techniques, représentant les parties et les services d'une plateforme. Il fournit, également, les concepts représentant les différents éléments permettant de spécifier l'utilisation de la plateforme par une application. Ces concepts seront utilisés par un PSM.

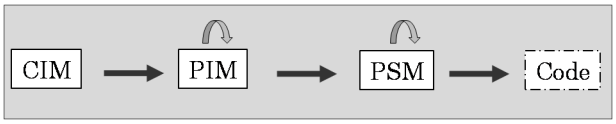


FIGURE 1.3 – MDA : modèles et transformations [OMG, 2003]

MDA définit les modèles CIM, PIM, PSM et aussi le code. MDA définit également la façon dont un modèle est obtenu à partir d'un autre. Le passage entre ces différents modèles se fait via une succession de transformation. L'étape la plus critique dans un processus MDA est la transformation d'un PIM en un ou plusieurs PSM [Kleppe et al., 2003]. Les différentes catégories des approches de transformation font l'objet de la section suivante.

Transformation

Dans un processus MDA, les transformations se font de manière automatique [Kleppe et al., 2003]. Tout comme les modèles, les transformations sont considérées comme éléments clé de l'IDM. L'objectif principal des transformations est d'automatiser les aspects fastidieux du développement logiciel. MDA permet plusieurs possibilités de transformations entre modèles, essentiellement entre les PIM et les PSM ; notons toutefois que tous les sens de transformation sont permis (cf. figure 1.3).

[Kleppe et al., 2003] définit la transformation comme la génération automatique d'un modèle cible à partir d'un modèle source selon une définition de transformation. Cette dernière présente un ensemble de règles qui décrivent comment obtenir le modèle cible à partir du modèle source. Une règle de transformation spécifie comment un (ou plusieurs) élément(s) du modèle source est transformé en un (ou plusieurs) élément(s) du modèle cible. Les approches de transformation de modèles peuvent être classées selon plusieurs critères. La transformation est dite `Endogène` si les métamodèles cibles et sources sont identiques, sinon elle est dite `Exogène`. On parle aussi de transformation « Modèle-Vers-Modèle » (Model-To-Model dite `M2M`) quand le résultat généré suite à la transformation est un modèle et de « Modèle-Vers-Texte » (Model-To-Text dite `M2T`) lorsque le résultat est du code. Si on considère le type de la transformation elle-même, on peut distinguer quatre catégories principales [Czarnecki and Helsen, 2006].

1. Approches opérationnelles : dans ces approches, les transformations sont

généralement basées sur un formalisme de métamodélisation étendu afin de permettre la modélisation du comportement des métamodèles. Il existe des langages et systèmes qui utilisent ce genre d'approches notamment, `QVT ((QueryViewTransformation)) Operational Mapping` [OMG, 2011b] et `Kermeta` [Muller et al., 2005].

2. Approches relationnelles : ces transformations sont basées sur la notion de `relation` et utilisent des règles déclaratives. Il s'agit de préciser les relations entre les éléments des modèles sources et cibles. Cette catégorie des approches permet les transformations multidirectionnelles et sépare les modèles sources des modèles cibles. Un exemples de mise en œuvre de ces approches est présentés par `QVT Relations` [OMG, 2011b] et AMW [Del Fabro et al., 2005].
3. Approches basées sur la transformation de graphes : dans ce type d'approches, les métamodèles source et cible de la transformation sont présentés par des graphes. Viatra [Varró et al., 2002], `AGG` [Taentzer, 2003], `AToM3` [Lara and Vangheluwe, 2002] et `Moflon` [Amelunxen et al., 2006] présentent des systèmes qui utilisent cette approche.
4. Approches de manipulation directe : dans cette catégorie, la mise en œuvre des règles de transformation, la planification (c'est à dire l'enchaînement des règles contradictoires) et le traçage sont effectués par l'utilisateur dans un langage de programmation donné. Les transformations sont ensuite mises en œuvre dans un cadre orienté objet qui fournit seulement une représentation du modèle interne et certaines API pour sa manipulation. Nous pouvons citer dans cette catégorie les travaux de [Djeddai, 2013].

D'autres approches dites `hybrides` combinent plusieurs types de transformations, nous pouvons citer ATL [Jouault and Kurtev, 2006] qui combine les règles impératives et déclaratives lors de la définition d'une transformation. Ceci est également le cas de la norme QVT [OMG, 2011b] composée du QVT relationnel et du QVT opérationnel.

1.3.3 Intérêts

Les principes de base de l'IDM peuvent se résumer en : (1) l'indépendance de la logique métier des plateformes techniques en utilisant des modèles, (2) l'automatisation des transformations entre modèles. De ces deux principes découlent de nombreux avantages. En effet, grâce aux transformations automatiques, les gains en termes de temps et d'effort lors du développement de l'application sont considérables. Une étude de la `Middleware Company` a montré qu'en utilisant un outil de développement dirigé par les modèles (`Compuware's OptimalJ`[5]), le gain atteint les 35% en terme de productivité et les 37% en terme de maintenance [Hutchinson et al., 2011]. Le passage par les modèles oblige les utilisateurs à favoriser les phases d'analyse et de modélisation. Bien que ceci implique une phase

5. http ://www.omg.org/mda/mda_files/MDA_OptimalJ.pdf

de spécification plus longue, les bénéfices sont tangibles lors de la maintenance de l'application. Les transformations offrent désormais la possibilité de développer plusieurs produits pour différentes plateformes, et de migrer un logiciel vers de nouvelles plateformes de manière plus aisée. Il ne faut pas non plus occulter les améliorations en termes de productivité, de portabilité, d'interopérabilité et de maintenance [Kleppe et al., 2003].

– **Productivité :** l'effort de développement se concentre sur l'élaboration du PIM. Les PSM, qui d'ailleurs sont aussi nécessaires, sont générés par transformation du PIM. Bien que la définition de la transformation ne soit pas une tâche simple, la transformation est définie une fois pour toute et peut être réutilisée dans le développement de nombreux systèmes. Ceci permet d'améliorer la productivité de deux manières différentes. D'un côté, l'élaboration du PIM requiert moins d'effort vu que les détails techniques spécifiques à des plateformes particulières ne seront pris en compte que dans la définition de la transformation. Aussi, au niveau du PSM et du code, une grande partie est générée de manière automatique à partir du PIM. La deuxième amélioration provient du fait que l'on se concentre sur le PIM plutôt que sur le code, ainsi plus d'attention est rendue à la résolution du problème applicatif. Il en résulte un système de meilleure qualité, plus rapidement développé et qui répond mieux aux besoins des utilisateurs finaux.

– **Portabilité :** la portabilité est garantie par le développement du PIM qui est défini indépendamment de la plateforme. Le même PIM peut être automatiquement transformé en plusieurs modèles spécifiques à différentes plateformes (PSM). Tout ce qui est spécifié au niveau du PIM est donc intégralement portable.

– **Interopérabilité :** les PSM générés à partir d'un même PIM ciblent différentes plateformes et ne peuvent pas communiquer directement les uns avec les autres. D'une façon ou d'une autre, nous devons transformer les concepts d'une plateforme vers les concepts utilisés dans une autre plateforme ; c'est l'interopérabilité. MDA répond à ce problème en générant non seulement des PSM, mais aussi les éléments qui les relient.
S'il est possible de transformer un PIM en deux PSM différents qui représentent deux plateformes différentes, toute l'information dont on a besoin pour relier les deux PSM est disponible. Pour chaque élément du PSM, l'élément du PIM à partir duquel il a été généré est connu. A partir des éléments du PIM, les éléments correspondant dans le second PSM sont aussi connus. On peut donc en déduire comment des éléments d'un PSM peuvent être reliés aux éléments d'un autre PSM.

– **Maintenance et documentation :** utiliser une démarche IDM permet de se centrer sur les modèles indépendants de plateformes (PIM) qui relèvent d'un niveau d'abstraction plus élevé que le code. A partir du PIM, les PSM sont générés, qui à leur tour sont utilisés pour générer le code. Le modèle

est une représentation fidèle du code. Ainsi, le PIM remplit la fonction de documentation de haut niveau nécessaire pour tout système logiciel. Dans la pratique, la différence avec un cycle de développement classique est que le PIM n'est pas abandonné après avoir élaboré le code. Les modifications que peut subir le système seront éventuellement effectuées en modifiant le PIM et en régénérant le PSM et le code. Ainsi, dans une approche dirigée par les modèles, la documentation abstraite est naturellement disponible.

1.4 Problématique : formaliser et automatiser l'élaboration d'ED historisés

Un entrepôt de données est créé afin de répondre à un besoin d'analyse pour l'aide à la prise de décision. La réussite d'un projet d'entreposage de données dépend de la phase de conception du schéma multidimensionnel et de la façon dont la correspondance entre l'ED et les sources est prise en compte [Vassiliadis et al., 2002]. Un autre point non négligeable est lié à la façon dont le schéma multidimensionnel de l'entrepôt est mis en œuvre au niveau physique. Autrement dit la qualité du schéma implanté au niveau physique qui doit être entièrement fidèle au schéma conceptuel. Ceci permet de garantir une réponse fiable aux besoins des décideurs. Compte tenu de tous ces aspects, le développement d'ED s'avère une tâche complexe, coûteuse et fastidieuse. D'autant plus, que l'évolution technologique perpétuelle contraint les organisations à migrer leurs logiciels vers de nouvelles plateformes, bien que globalement la logique métier ne change pas.

Il n'y a pas un consensus sur la méthode d'élaboration d'ED. Les méthodes actuelles proposent des solutions pour concevoir et implanter le schéma multidimensionnel d'une part, les processus d'alimentation d'autre part, sans pour autant combiner les deux. En effet, à l'heure actuelle, l'élaboration d'ED (y compris la création des structures et leur alimentation à partir des sources) nécessite l'utilisation de modèles voire de méthodes différent(e)s. Les problèmes d'intégration doivent être considérés afin de sélectionner les modèles appropriés. D'autre part, la plupart des approches existantes manque de mécanismes pour générer automatiquement ou documenter tous ces aspects. Le développement d'ED s'avère coûteux et voué à l'échec lorsqu'il est fait de manière manuelle [Kimball, 1996].

Données historisées. Un ED contient des données historisées pour répondre aux besoins des décideurs. Mais ces données deviennent s'accumulent et deviennent obsolètes au fil du temps. La maîtrise et la gestion de ces volumes deviennent quasi-impossibles et les temps de réponse deviennent de plus en plus importants. D'autant plus que le taux de données inactives (inutiles pour la prise de décision) est de plus en plus important. Selon l'institut Forrester

Research [6], le volume des données hébergées dans des applications métiers volumineuses, notamment les entrepôts de données, s'accroît de 65% chaque année. Cette croissance est due principalement à l'accumulation de données inactives. On estime que le taux de ces données atteint les 85%. Même si le coût de stockage baisse avec l'évolution technologique et qu'il existe des moyens pour optimiser les temps de réponse tels que les index, les pré-agrégats, etc. Il s'avère judicieux de supprimer des données inutiles. Il convient de noter que les données obsolètes qui décrivent la réalité à un niveau détaillé, peuvent rester utiles pour l'analyse à un niveau supérieur.

De plus, les données historisées perdent de leur intérêt avec le temps : alors que la granularité des informations doit généralement être importante pour des données récentes [Skyt et al., 2008] ; elle peut être plus faible pour des données anciennes. Par exemple, un décideur peut analyser ses ventes au niveau de la granularité produit sur les cinq dernières années tandis que pour les périodes antérieures, ces analyses au niveau du produit seraient absurdes (les produits n'existent plus à l'heure actuelle) et donc le décideur fera des analyses au niveau de la gamme du produit (qui n'a pas évolué dans le temps). Afin de faciliter la tâche du décideur et de mieux répondre à ses besoins, il est préférable de garder uniquement l'information nécessaire à ses analyses. L'idée est donc d'offrir un environnement d'analyse multidimensionnelle adapté aux besoins des décideurs en leur permettant de supprimer dans le temps les niveaux de granularité inutiles pour leurs analyses.

Partout dans le monde, il existe des lois pour réglementer les droits de conservation des données. A titre d'exemple, pour des raisons de confidentialité, les données personnelles d'un client ne peuvent être conservées que pour une durée limitée. En France, une des clauses de la loi `Informatique et Liberté` [7] précise que les données personnelles doivent être supprimées dès que la personne ne fait plus partie du système. Le problème qui se pose est comment conserver uniquement le sous ensemble de données nécessaires pour la prise de décision tout en garantissant une conservation légale des données. Aussi, comment gérer l'ensemble de ces données au fil du temps.

En synthétisant les données, il est également possible de rendre les données anonyme (ville de client au lieu de son nom), par conséquent, dans certains cas, ceci permet d'éviter la nécessité de supprimer complètement des données [Skyt et al., 2008].

Afin de répondre à cette problématique, nous souhaitons proposer une approche « unifiée » et « automatique » pour la modélisation et l'implantation d'entrepôt de données historisées. Il s'agit d'une approche « unifiée » car elle doit fournir un seul cadre pour la conception conjointe du schéma multidimensionnel et des opérations ETL. Cette approche est dite « automatique » car elle doit proposer un ensemble de règles permettant la génération automatique des schémas lo-

6. http ://www.forrester.com/home
7. http ://www.cil.cnrs.fr/CIL/spip.php ?rubrique281

giques et physiques à partir du schéma conceptuel, permettant ainsi de réduire le temps et l'effort requis pour élaborer un ED. Notons que nous partons du schéma multidimensionnel de l'entrepôt et que nous ne traitons pas la façon dont ce schéma a été obtenu (à partir des besoins et/ou des sources).

Dans ce contexte, il est reconnu que l'IDM (cf. section 1.3) est un cadre adapté pour gèrer la complexité de développement et de maintenance [Palyart et al., 2011]. L'IDM réduit considérablement les coûts, améliore la qualité du logiciel et met en œuvre la réutilisation [Kleppe et al., 2003], [OMG, 2003]. Ainsi en utilisant cette démarche, le développement d'ED nécessite moins d'efforts et de temps. L'IDM permet d'aboutir au code de l'application en partant des modèles conceptuels grâce à la définition d'une série de transformations. Par ailleurs, l'IDM offre un support à l'intégration, l'interopérabilité, l'adaptabilité et la portabilité des systèmes d'informations [Kleppe et al., 2003].

En plus de ce processus IDM pour l'élaboration d'ED, nous nous intéressons au problème de réduction de données. Notre objectif est de proposer une approche dirigée par les modèles qui permet de formaliser et d'automatiser le processus de réduction de données afin de conserver uniquement les données nécessaires aux analyses décisionnelles. Cette approche permet de réduire les données les plus anciennes inutiles pour la prise de décision. De plus, cette solution permettrait d'anticiper le problème de temps de réponse aux requêtes multidimensionnelles dès les premières phases de modélisation multidimensionnelle.

1.5 Contributions et organisation du mémoire

Nous exposons dans cette section les principales propositions de cette thèse. Nous présentons par la suite, l'organisation de ce mémoire.

1.5.1 Contributions

Les propositions de cette thèse se résument comme suit :

1. Formaliser et automatiser le processus de développement d'un ED en proposant une approche dirigée par les modèles qui inclut :
   - un ensemble de métamodèles (conceptuel, logique et physique) décrivant les données et les opérations de transformation.
   - une extension du langage `OCL` (`Object Constraint Langage`) pour décrire de manière conceptuelle les opérations de transformation d'attributs sources en attributs cible de l'ED.
   - un ensemble de règles de transformation d'un modèle conceptuel en modèles logique et physique.
   - un ensemble de règles permettant la génération du code de création et de chargement de l'entrepôt.

2. Formaliser et automatiser le processus de réduction de données historisées en proposant une approche dirigée par les modèles qui fournit :
   - un ensemble de métamodèles (conceptuel, logique et physique) décrivant les données réduites.
   - un ensemble d'opérations de réduction.
   - un ensemble de règles de transformation permettant d'implanter ces opérations au niveau physique.
3. Un prototype permettant de valider ces différentes contributions. Cet outil est composé principalement de deux modules :
   - le module de transformation IDM restitue le code SQL de création et d'alimentation de l'entrepôt de données à partir du schéma multidimensionnel entré par le concepteur. Pour ce faire, ce module fournit un ensemble de règles de transformation de modèles vers modèles et de modèle vers texte.
   - le module de réduction implante l'ensemble d'opérateurs de réduction et de vérifier l'ensemble des contraintes définies pour garantir la bonne formation d'un schéma réduit.

1.5.2 Organisation du mémoire

Le chapitre 2 est consacré à l'état de l'art. Nous y présentons les travaux portant sur la modélisation et l'implantation d'ED historisées. Dans ce chapitre, nous commençons par présenter les propositions de modélisation de schémas d'ED. Ensuite, nous présentons les travaux qui traitent de la modélisation des opérations ETL. Enfin nous présentons les solutions proposées afin de réduire les données d'un entrepôt. Nous présentons également une synthèse sur ces différents travaux et nous discutons des insuffisances décelées suite à l'étude de ces approches et qui nous ont permis de fonder nos propositions.

Le chapitre 3 présente notre approche dirigée par les modèles pour l'élaboration d'entrepôts de données. Cette approche présente les différents niveaux de modélisation d'ED et des opérations ETL de manière conjointe dans un cadre IDM. Le premier niveau de modélisation multidimensionnelles permet de décrire l'ED en termes de faits et de dimensions, et les opérations via une extension du langage `OCL`. Ces concepts sont décrits dans un métamodèle. Ensuite, les niveaux logique et physique qui présentent également les données et les opérations de manière conjointe sont décrits par des métamodèles. Dans ce chapitre, nous définissons aussi les transformations qui permettent de générer le code SQL à partir des modèles supérieurs.

Le chapitre 4 présente notre approche IDM pour la modélisation et l'implantation d'entrepôts réduits. Ce chapitre définit un ensemble d'opérateurs de réduction de schémas multidimensionnels et de contraintes pour définir des schémas réduits cohérents. Nous adaptons les métamodèles et les règles de transformations définis dans le chapitre 3 pour permettre la génération de code de création de d'alimentation d'un schéma réduit de manière automatique.

Le chapitre 5 valide nos contributions à travers la réalisation d'un prototype. Ce prototype utilise un ensemble de techniques permettant le développement dirigé par les modèles des applications. Il permet la modélisation, la métamodélisation et la transformation de schémas d'entrepôts pour générer du code.

Le chapitre 6 conclut ce mémoire en rappelant nos contributions, l'originalité de notre travail, tout en précisant ses limites. Nous terminons ce mémoire en explicitant les différentes perspectives.

Chapitre 2

Etat de l'art

2.1 Introduction

La prise de décision s'appuie sur un entrepôt contenant des données pertinentes collectées à partir des sources au moyen des processus ETL. Notre problématique porte sur le développement des entrepôts et plus particulièrement sur la modélisation des schémas et des processus ETL ainsi que sur la réduction de données. Dans la littérature, de nombreux travaux proposent des solutions pour concevoir et implanter le schéma multidimensionnel ainsi que les processus d'alimentation. D'autres travaux s'intéressent au problème de la réduction du volume de données dans l'entrepôt.

Dans ce chapitre, nous présentons les travaux qui ont traité ces différentes problématiques. La section 2.2 présente les approches de modélisation de schémas d'entrepôt. La section 2.3 montre les approches de modélisation des processus ETL. En section 2.4, nous présentons une synthèse sur ces différentes contributions. Enfin, la section 2.5 aborde les approches qui visent à gérer de gros volumes de données en les réduisant et donne une synthèse sur ces approches.

2.2 Elaboration de schémas d'entrepôt

La conception d'entrepôts de données a fait l'objet de plusieurs travaux de recherche [Romero and Abelló, 2009]. Généralement, les approches existantes peuvent être classées en trois grandes catégories [Rizzi et al., 2006] : `ascendantes`, `descendantes` et `mixtes`. Les approches ascendantes partent d'une analyse détaillée des sources de données, mais occultent les besoins des décideurs. Les travaux de [Golfarelli and Rizzi, 1998], [Moody and Kortink, 2000] et [Song et al., 2007] proposent différentes démarches qui permettent de générer

le schéma de l'entrepôt à partir de schémas `Entité/Association` décrivant les sources. Les travaux de [Jensen et al., 2004] et de [Romero and Abelló, 2007] exploitent respectivement des techniques de fouille de données et des ontologies afin de générer le schéma multidimensionnel. Les approches descendantes [Tsois et al., 2001] et [Prat et al., 2006] permettent de construire le schéma de l'entrepôt à partir d'une analyse détaillée des besoins des décideurs. Les problèmes de correspondance entre ces besoins et les sources existantes ne sont traités qu'à postériori lors du chargement des données dans l'entrepôt. Les approches mixtes [Hüsemann et al., 2000], [Bonifati et al., 2001], [Phipps and Davis, 2002], [Giorgini et al., 2005] [Romero and Abelló, 2010] et [Abdelhédi and Zurfluh, 2013] considèrent à la fois les besoins des décideurs et les données sources. Elles présentent donc l'avantage de concevoir des schémas multidimensionnels valides par rapport aux sources de données.

Les approches que nous venons de présenter fournissent des contributions différentes pour construire le schéma multidimensionnel de l'entrepôt. Rappelons que parmi les problèmes traités, nous considérons la formalisation et l'automatisation de l'implantation du schéma multidimensionnel au niveau physique, ceci dans un cadre dirigé par les modèles. Ainsi, dans cette section, nous détaillons d'abord les travaux effectués par [Prat et al., 2006] qui proposent une démarche pour la formalisation du processus de conception de schémas d'entrepôt. Ensuite nous présentons les approches abordées dans un cadre IDM notamment les travaux de [Mazón and Trujillo, 2009] qui utilisent cette approche pour transformer les schémas.

2.2.1 Approches de conception de schémas

Les auteurs de [Prat et al., 2006] présentent une démarche pour élaborer les schémas conceptuel, logique et physique de l'entrepôt. A partir des besoins des décideurs, la modélisation conceptuelle fournit un diagramme de classes UML. Les auteurs estiment que le schéma multidimensionnel relève d'un niveau d'abstraction logique. Ainsi, lors de la phase de modélisation logique, le diagramme de classes est enrichi pour obtenir un schéma multidimensionnel. Le modèle logique est ensuite traduit au niveau physique en schéma MOLAP. Les auteurs définissent les règles de traduction d'un niveau à un autre.

1. **Modèle conceptuel**
   A ce niveau, les auteurs décrivent initialement les éléments de l'entrepôt issus de l'analyse des besoins des décideurs via un diagramme de classes UML. Par la suite, ce diagramme est enrichi afin de faciliter la transformation vers le niveau logique. Pour ce faire, les auteurs proposent d'étendre le métamodèle UML en définissant un ensemble de stéréotypes. Les principales extensions consistent en l'ajout de deux attributs booléens dans la métaclasse « Attribut ». Le premier attribut appelé « mesure » indique si l'attribut représente (ou non) une mesure d'intérêt. Le deuxième attribut

indique si l'attribut est identifiant de la classe à laquelle il appartient ou non. Seules les classes dites ordinaires (les classes d'association sont exclues) doivent avoir un attribut identifiant. Si aucun attribut ne peut être considéré comme identifiant, la création d'un identifiant est requise. Les auteurs décrivent ces contraintes en utilisant le langage OCL. Quant à la définition des attributs « mesure », celle-ci doit être faite manuellement par le concepteur en fonction des besoins du décideur. En outre, dans la mesure où le diagramme initial peut contenir des classes d'association ayant des attributs, le concepteur doit vérifier la validité de cette représentation. Ensuite, les attributs des classes d'association de type `1-N` doivent migrer vers la classe du coté `N`. Lorsque l'association est de type `1-1`, les attributs migrent d'un coté ou d'un autre. Le résultat de cette étape est que toutes les classes d'association ayant des attributs sont transformées (grâce à la migration de leurs attributs) en associations ordinaires (c'est-à-dire n'ayant pas d'attributs). Pour transformer les liens de généralisation qui représentent d'éventuelles hiérarchies au niveau logique, les auteurs proposent de créer une classe dont le nom est précédé par « type ».

2. **Modèle logique (multidimensionnel)**
   Les auteurs proposent un ensemble d'étapes pour transformer le diagramme de classes enrichi en schéma multidimensionnel. Dans un premier temps, les associations `N-M` sont converties en fait. Si ces associations contiennent des attributs, ces derniers sont convertis en mesures. Ces faits sont analysés selon des dimensions issues des classes ordinaires impliquées directement ou indirectement dans l'association. Toute classe ordinaire ayant un attribut dont la valeur de l'attribut mesure est vraie, est transformée en fait. Ces attributs sont convertis en mesures. Pour chaque niveau de dimension lié à un fait, d'abord, un ensemble de hiérarchies est défini en se basant sur les chemins d'association `1-N`. Ensuite, la dimension est créée pour l'ensemble des hiérarchies définies. Enfin, pour chaque mesure et chaque dimension liée au fait, l'ensemble des fonctions d'agrégation applicable sur la mesure est défini.

3. **Modèle physique**
   Les auteurs considèrent une implantation MOLAP au niveau de la plateforme Oracle. Une dimension logique est convertie en une dimension du niveau physique. Les mesures et les attributs relatifs aux niveaux de dimensions sont convertis en variables. Les hiérarchies sont transformées en relations. Une relation représente une dépendance fonctionnelle entre une dimension et une dimension de référence. Les dimensions peuvent être temporelles ou non. Dans le cas d'une dimension temporelle, celle-ci est fournie par le système avec les niveaux jour, semaine, mois, trimestre et année.

Les travaux de [Prat et al., 2006] fournissent un ensemble de modèles et de formalisation des transformations permettant l'implantation de schémas d'entrepôt au niveau physique. Cependant, l'automatisation du processus d'implantation

n'a pas été considérée. Afin de tenir compte de cet aspect, les approches de [Zepeda et al., 2008], [Mazón and Trujillo, 2009], [Carmè et al., 2010] sont abordées dans un cadre dirigé par les modèles et sont détaillées dans la section suivante.

2.2.2 IDM pour la modélisation de schémas d'entrepôts

Afin de formaliser et d'automatiser le processus de conception des schémas multidimensionnels, des approches de modélisation d'ED ont été abordées dans un cadre IDM. Certaines se contentent de fournir un ensemble d'étapes pour guider le concepteur à élaborer le modèle conceptuel, d'autres présentent également le niveau logique et éventuellement le niveau physique.

En ce qui concerne la modélisation multidimensionnelle, l'article de [Zepeda et al., 2008] propose une approche qui vise à élaborer le modèle conceptuel de l'entrepôt en partant d'un modèle conceptuel source et des besoins des décideurs. La méthode est basée sur trois étapes. La première analyse le schéma `Entité/Association` de la source et applique un ensemble de règles de transformation défini entre le métamodèle `Entité/Association` et le métamodèle `OLAP`. La seconde phase permet de collecter les besoins des décideurs et de les décrire sous forme d'un modèle de buts. La dernière étape permet la confrontation du modèle des buts et du modèle des sources pour aboutir au modèle multidimensionnel final. Les auteurs de [Carmè et al., 2010] présentent une approche qui vise à formaliser la détection de faits à partir de sources de données relationnelles. L'approche est basée sur des heuristiques issues d'un ensemble de cas réels. D'abord, les tables de la BD source sont identifiées. Ensuite, le concepteur détermine les éléments de cette BD pouvant être considérés comme faits. Enfin, il définit l'ensemble des dimensions et des mesures pour chaque fait du modèle. Les travaux de [Zepeda et al., 2008] et [Carmè et al., 2010] utilisent le langage QVT pour formaliser la correspondance entre les éléments sources et cibles de l'ED.

D'autres approches permettent de modéliser différents niveaux d'abstraction de l'entrepôt. Les auteurs de [Mazón and Trujillo, 2008] et [Mazón and Trujillo, 2009] proposent une démarche mixte pour l'élaboration et l'implantation de schémas multidimensionnels d'ED dans un cadre IDM. Cette approche définit les modèles et les transformations comme suit.

1. **CIM**
   Les auteurs définissent un « CIM multidimensionnel » décrivant les besoins décisionnels grâce à un modèle de buts. Ce modèle définit les besoins à trois niveaux : « besoins stratégiques », « besoins décisionnels » et « besoins informationnels ». Les besoins stratégiques représentent les principaux objectifs de l'entreprise, par exemple « augmenter les ventes ». Les besoins décisionnels visent à définir les actions à faire pour réaliser un besoin stratégique, par exemple « ouvrir un nouveau magasin ». Les besoins informationnels sont reliés à l'information requise pour réaliser un besoin

décisionnel ; « analyser les ventes ». Une fois les besoins informationnels définis, les éléments du schéma multidimensionnel (faits et dimensions) peuvent être déterminés. Afin de modéliser ces besoins, les auteurs proposent un profil UML qui permet d'adapter les éléments du modèle `i*` [Yu, 1997] au domaine de l'ED. Le modèle i* fournit les mécanismes pour représenter les différents acteurs de l'ED ainsi que leurs dépendances. Il permet également de structurer les objectifs à atteindre. Le profil proposé définit principalement les stéréotypes (de besoins) « Stratégique »,« Décisionnel » et « Informationnel ». Ce profil définit également un ensemble de stéréotypes multidimensionnels, notamment « Processus Métier » relié aux objectifs des décideurs, « Mesure » relié aux besoins informationnels et « Contexte » requis pour analyser une mesure. Les relations d'agrégation entre contexte définissent l'organisation hiérarchique de ses données. Ainsi, le CIM est défini en appliquant les étapes suivantes : définir les acteurs, définir leurs objectifs (et les classifier), dériver les besoins informationnel à partir des buts informationnels et obtenir les concepts multidimensionnels reliés à chaque besoin informationnel.

2. **PIM**
   A ce niveau, les auteurs définissent un PIM dit initial dérivé à partir du CIM. Ce PIM est par la suite confronté au modèle source pour générer un PIM hybride.

   – **PIM multidimensionnel initial**
     Afin de décrire ce modèle, les auteurs proposent un profil UML multidimensionnel. Ce profil définit principalement les concepts de faits et de dimensions. Un fait est composé d'un ensemble d'attributs. Une dimension est décrite comme un ensemble de bases. Une base est composée d'un identifiant, d'un descripteur et d'un ou plusieurs attributs de dimensions. Le PIM multidimensionnel initial est obtenu à partir du CIM suite à une série de transformations QVT. Dans le CIM, l'ensemble de buts et de « Processus Métier » est transformé en fait au niveau PIM. Les mesures reliées à des besoins informationnels sont transformées en attributs associés au fait correspondant. Les contextes sont transformés en dimensions, les hiérarchies sont déduites à partir des relations inter-contexte du CIM.

   – **PIM multidimensionnel hybride**
     Le PIM initial est confronté avec la source de données pour générer un PIM dit hybride. Pour ce faire, dans un premier temps, le modèle logique de la source est transformé en appliquant des règles QVT en un modèle annoté par des éléments multidimensionnels (faits, dimensions, mesures, etc.). Dans un second temps, ce modèle annoté est confronté au le PIM initial pour obtenir le PIM hybride. Les transformations se basent sur les dépendances fonctionnelles dans le modèle source et dans le PIM initial.

3. **PSM**
   A ce niveau, les auteurs utilisent le `CWM` (Common Warehouse Metamodel [1]) comme métamodèle du PSM. Ils présentent une instance du schéma ROLAP. Ce dernier est généré en appliquant un ensemble de règles QVT où les faits et les dimensions sont transformés en tables.

4. **Transformation**
   Une fois le CIM défini, les auteurs proposent de générer les modèles PIM initial, PIM hybride et PSM en appliquant des transformations de modèles vers modèles formalisées en QVT.

Les approches de conception d'entrepôts visent à définir et à implanter le schéma de l'entrepôt. Certaines approches présentent l'avantage de définir des schémas multidimensionnels au niveau conceptuel et d'aboutir aux modèles logique et physique. Cependant, une fois les structures de l'entrepôt créées, on se heurte à l'alimentation de l'ED à partir des sources. Dans la section suivante, nous présentons les approches proposant des solutions à ce problème.

2.3 Modélisation des processus d'extraction, de transformation et de chargement

Les processus d'extraction de transformation et de chargement (ETL) sont des processus logiciels qui permettent l'alimentation initiale de l'entrepôt et son rafraîchissement périodique à partir des sources. Les travaux de recherche et les outils commercialisés dans le domaine ETL sont foisonnants et décrits dans [Vassiliadis, 2009]. D'une part, les industriels proposent de nombreux outils [Barateiro and Galhardas, 2005] tels que `Microsoft Integration Services` [2] et `Oracle Warehouse Builder` [3]. D'autre part, dans la littérature de nombreuses contributions ont traité la modélisation des processus ETL. Dans notre travail, nous nous intéressons au problème particulier de modélisation et d'implantation des processus de transformation ; nous présentons dans cette section, les principales contributions en la matière. De manière générale, les travaux de recherche proposent soit des modèles spécifiques [Simitsis and Vassiliadis, 2003] (travaux de `Vassiliadis et al.`) soit d'utiliser et/ou d'étendre des standards existants tels que `UML` ou le `Business Process Model Notation` (BPMN) [4], notamment les travaux de `Trujillo et al.` [Trujillo and Luján-Mora, 2003], [Muñoz et al., 2008]. D'autres travaux [Muñoz et al., 2009], [El Akkaoui et al., 2011] ont adopté un cadre IDM pour formaliser et automatiser le processus de transformation.

Dans cette section, nous présentons les principales contributions qui visent à modéliser les processus de transformation et de chargement.

1. http ://www.omg.org/spec/CWM/
2. http ://www.microsoft.com
3. http ://www.oracle.com/technetwork/developer-tools/warehouse/overview/introduction/index.html
4. http ://www.omg.org/spec/BPMN/2.0/

2.3.1 Approches spécifiques ou basées sur des standards

Les auteurs de [Vassiliadis et al., 2002] proposent un modèle conceptuel des opérations ETL (appelées aussi `mécanismes`, `fonctions` ou `activités` ETL) qui vise à décrire les relations de correspondance entre les éléments sources et les éléments cibles de l'entrepôt. Pour ce faire, ils définissent un métamodèle qui permet de relier les concepts sources (par exemple les tables) et les concepts cibles (faits et dimensions). Ce métamodèle définit également un ensemble d'opérations de filtrage, de nettoyage et de transformation de données sources présentées dans le tableau 2.1. A chaque opération, il est possible d'attacher une contrainte qui explique les conditions de fonctionnement de l'opération. Dans [Simitsis and Vassiliadis, 2003], les auteurs proposent une démarche pour la modélisation des processus ETL. L'objectif de cette proposition est de compléter le modèle via un ensemble d'étapes conduisant à la spécification d'attributs liés. La première étape consiste à identifier les sources appropriées, la seconde concerne l'identification des éléments sources susceptibles d'être transformés en éléments cibles. La troisième étape permet d'établir la correspondance entre les attributs sources et les attributs cibles. Enfin, la dernière étape permet d'annoter le diagramme via un ensemble de contraintes.

Au niveau logique, les auteurs de [Simitsis et al., 2005] proposent de modéliser les processus ETL sous forme de graphes. Le modèle permet d'expliciter les éléments sources, les éléments cibles de l'ED ainsi que les transformations qui ont eu lieu. La transformation du modèle conceptuel présenté dans [Vassiliadis et al., 2002] en un modèle logique se fait de manière semi-automatique [Simitsis, 2005]. A chaque entité du modèle conceptuel correspond une entité dans le modèle logique. Les concepts et les attributs sont convertis respectivement en un ensemble d'enregistrements et d'attributs. Les opérations de transformation, de filtrage, de nettoyage et les contraintes sont converties en un ensemble d'activités.

Afin de modéliser les processus ETL, certaines approches proposent d'utiliser ou d'étendre des normes existantes telles qu'UML. Notamment, dans [Trujillo and Luján-Mora, 2003], les auteurs proposent d'utiliser les diagrammes de classes UML comme modèle conceptuel des processus ETL. Ils définissent un ensemble de mécanismes ETL inspirés de ceux présentés dans le tableau 2.1. En particulier, un processus ETL est composé d'un ensemble de paquetages UML qui permettent au concepteur de décomposer un processus ETL en unités différentes. Chaque mécanisme ETL est représenté par le biais d'une classe stéréotypée. Ces mécanismes sont reliés en utilisant les relations de dépendances UML. Une note UML peut être attachée à tous les mécanismes afin d'expliquer leur fonctionnement et/ou définir les correspondances entre les attributs sources et les attributs cibles.

Dans [Luján-Mora et al., 2004], les auteurs proposent d'étendre UML via un diagramme de `Correspondance`. Ils définissent quatre niveaux différents de correspondance de données. Le premier niveau de « `base de données` » présente

les données concernées via des paquetages UML, les relations sont présentées par un paquetage de correspondance. Le deuxième niveau de « `flux de données` » montre les relations entre les tables sources et les tables cibles. Dans le troisième niveau « `table` », le diagramme montre les transformations sous forme de paquetages. Le dernier niveau des « `attributs` », présente les relations entre les attributs sources et les attributs cibles. Les relations entre attributs présentent les différents mécanismes ETL (agrégation, conversion, etc.). De même, dans [Muñoz et al., 2008], étant donné que les diagrammes d'activités UML permettent de décrire l'aspect dynamique d'un système, les auteurs proposent de les utiliser pour concevoir les processus ETL. Les mécanismes présentés dans [Trujillo and Luján-Mora, 2003] sont décrits comme un ensemble d'activités.

2.3.2 IDM pour la modélisation des processus ETL

Dans cette section, nous détaillons les approches qui proposent un cadre dirigé par les modèles pour décrire, à différents niveaux d'abstraction, les opérations ETL résumées dans le tableau 2.1. Dans [Muñoz et al., 2009], les auteurs proposent une approche qui définit le modèle conceptuel (PIM) sous forme d'un diagramme d'activités UML tel que présenté dans [Muñoz et al., 2008]. A partir de ce modèle, l'objectif est de générer le modèle logique (PSM) de manière automatique. Ces travaux présentent les modèles et les transformations suivants :

1. **Modèles**

   - **PIM**
     Un diagramme d'activités sert à décrire des aspects comportementaux d'un système et de montrer le déclenchement d'une suite d'évènements. Les auteurs proposent de concevoir les processus ETL comme un ensemble d'activités décrivant le comportement d'un processus. Un diagramme d'activités ETL comporte en entrée une table source, une liste des opérations à exécuter et une sortie montrant une table cible de l'entrepôt. Les actions d'un diagramme représentent les opérations d'agrégation, de conversion de filtrage et de jointure. Les auteurs définissent les activités suivantes :
     - Activité d'agrégation : l'entrée de cette activité montre une table source (Ventes par exemple). Dans un premier temps, l'ensemble des attributs de la table sont extraits. Dans un second temps, un ensemble d'actions de vérification des attributs clés et de calcul (utilisant les fonctions d'agrégation `Sum`, `Avg`, `Count`, `Min` et `Max`) est appliqué. Ensuite une action de regroupement (`Group by`) est exécutée.
     - Activité de conversion : après avoir extrait et vérifié les attributs sources, cette activité permet de transformer un attribut en précisant l'opération à appliquer (par exemple la concaténation d'attributs de type chaine de caractères).

- Activité de filtrage : permet de filtrer les données inutiles et vérifie que seules les données qui répondent aux critères de sélection sont chargées dans l'ED.
- Activité de jointure : cette activité prend en entrée une ou plusieurs tables sources et extrait les attributs (dont on a besoin) de chaque table. Ensuite, une action de vérification est appliquée pour aboutir à la jointure.

– **PSM**
A ce niveau, les auteurs proposent d'utiliser la plateforme Oracle pour l'alimentation de l'ED. L'ensemble des activités y compris les actions d'extraction à partir de la source et les opérations de transformations sont traduites en éléments du métamodèle de cette plateforme.

2. **Transformation**
Pour mettre en œuvre les transformations de modèles (du PIM vers le PSM), les auteurs présentent le métamodèle des diagrammes d'activités UML et le métamodèle du PSM décrivant les concepts de la plateforme Oracle. Le diagramme d'activités défini au niveau PIM est transformé au niveau PSM en appliquant un ensemble de règles QVT. Principalement, ces règles visent à convertir une activité du niveau PIM en une table de correspondance au niveau PSM. Les actions sont transformées en opérateurs de correspondance et les sources de données sont converties en tables. Enfin, les paramètres, regroupés en classes comportementales dans le diagramme des activités (PIM), sont transformées en groupe d'attributs du niveau PSM.

Les travaux de [El Akkaoui and Zimányi, 2009], [El Akkaoui et al., 2011] et de [El Akkaoui et al., 2012] proposent une approche qui vise à couvrir le cycle de développement des processus ETL et qui permet la génération semi-automatique du code. L'approche définit un modèle (PIM) dans lequel les auteurs utilisent le BPMN. Ce modèle est par la suite transformé pour générer le code.

1. **Modèle (PIM)**
Le BPMN est une norme de l'OMG qui permet de modéliser le déroulement des processus d'une entreprise dans un workflow. Cette norme décrit les processus en termes d'activités représentant le travail accompli par un processus. Une activité est décomposée en tâches. Lorsque les activités sont combinées, elles forment un sous-processus. Les auteurs définissent principalement trois types de tâches décrivant l'extraction (l'entrée qui représente la base de données et les fichiers sources), la transformation (jointure, filtrage, etc.) et le chargement (la base de données en sortie). Les tâches ETL sont les suivantes :
   - Tâche de dérivation : permet de dériver la valeur d'un attribut cible à partir d'attributs sources.
   - Tâche de jointure : permet de fusionner deux ou plusieurs éléments sources.

- Tâche de conversion de type : lorsque les données sont extraites d'un fichier, le type par défaut est une chaine de caractères. Cette opération permet d'affecter à chaque donnée son type d'origine.
- Tâche de filtrage : permet de sélectionner les données (notamment les valeurs non nulles).
- Tâche d'agrégation : applique une fonction d'agrégation pour calculer la valeur d'un attribut cible de l'entrepôt.

Au niveau PIM, le métamodèle proposé est structuré en un ensemble de paquetages. Le paquetage racine définit l'ensemble des processus d'alimentation de l'ED à partir des sources. Le deuxième paquetage décrit les données tout au long du processus, en allant des sources vers l'entrepôt. Il comporte les différentes tâches de jointure, de filtrage, de conversion, etc. Un événement est défini afin de personnaliser les exceptions levées par une tâche. Les tâches qui partagent les mêmes caractéristiques constituent un sous-processus. Le paquetage source décrit l'ensemble des sources utilisées pour alimenter l'ED. Les auteurs définissent également un métamodèle de vérification de cohérence grâce au langage `OCL`.

2. **Code**
   Les auteurs proposent d'implanter les processus ETL en utilisant la plateforme Oracle. Pour ce faire, ils définissent une grammaire qui décrit les tâches du niveau PIM. Cette grammaire est par la suite utilisée pour générer le code de manière semi-automatique.

3. **Transformation**
   Les auteurs définissent un ensemble de « templates » de transformation de modèle vers du texte. Les règles de correspondance sont établies entre les éléments du métamodèle PIM et la grammaire. Chaque « template » contient une partie du code qui correspond à un concept du métamodèle source.

2.4 Synthèse sur les approches de modélisation

Comme nous l'avons présenté dans les sections 2.2 et 2.3, nous pouvons constater que dans la littérature les problèmes de modélisation du schéma de l'ED et des processus ETL ont été traités de manière séparée ; pourtant les deux sont interdépendants. En effet, les approches actuelles offrent des solutions partielles qui se concentrent soit sur la modélisation des structures de l'ED soit sur la modélisation des opérations ETL, sans pour autant combiner les deux.

Les approches de modélisation d'ED visent à définir et à implanter le schéma de l'entrepôt. Certaines approches présentent l'avantage de définir des schémas multidimensionnels au niveau conceptuel et d'aboutir de manière automatique aux modèles logique et physique.

En étudiant les travaux de modélisation ETL, il s'avère que les premières contributions [Vassiliadis et al., 2002], [Simitsis and Vassiliadis, 2003] ont défini un ensemble d'opérations ETL. De manière générale, les contributions qui suivent présentent de nouveaux moyens pour formaliser ces opérations. Nous pouvons résumer ces opérations dans le tableau 2.1.

TABLE 2.1 – Récapitulatif des opérations ETL

Opération ETL		**Description**
Extraction	Wrapper	Transforme une donnée native en enregistrement source.
Transformation	Aggregation	Agrège les données basée sur des critères.
	Conversion	Modifie les types et les formats de données ou dérive de nouvelles données à partir de données existantes.
	Filter	Filtre et vérifie les données.
	Incorrect	Redirige les données incorrectes.
	Join	Jointure de deux sources de données reliées aux travers d'attributs.
	Merge	Intègre deux ou plusieurs sources de données en utilisant des attributs compatibles.
	Surrogate	Génère une clé unique.
Chargement	Loader	Charge les données dans la cible.
	Log	Stocke les activités d'un mécanisme ETL.

A notre connaissance, seuls les travaux de [Mazón and Trujillo, 2008] et de [Romero et al., 2011] évoquent la prise en compte simultanée des aspects de modélisation des structures de l'ED et des opérations ETL. Dans [Mazón and Trujillo, 2008], les auteurs présentent l'architecture générale de l'entrepôt de données partant des sources en allant jusqu'aux magasins, et ceci dans un cadre IDM. Bien que ces deux problèmes aient été évoqués, l'approche proposée fournit des modèles différents voire des approches différentes comme présentés dans [Mazón and Trujillo, 2008] et [Mazón and Trujillo, 2009] pour la modélisation des données multidimensionnelles et dans [Muñoz et al., 2009] pour la modélisation des processus ETL. Certes, la modélisation des structures multidimensionnelles a été bien traitée (différents niveaux de modélisation et génération de code ont été pris en compte), toutefois, celle des processus ETL se limite à décrire un ensemble d'activités ETL et leurs transformation au niveau logique. L'approche n'aboutit pas la génération du code. Aussi, à notre connaissance, les aspects de validation n'ont pas été présentés dans cet article.

Quant à l'article de [Romero et al., 2011], il propose un schéma conceptuel qui

décrit de manière conjointe les processus ETL et l'entrepôt. La sémantique, les caractéristiques et les contraintes des sources de données sont représentées au moyen d'une ontologie OWL [5]. Pour chaque besoin, l'approche identifie les éléments sources (concepts, attributs, propriétés, etc.) nécessaires pour y répondre. Par la suite, l'ensemble des dimensions et des faits est identifié à partir de ces concepts. Tous ces éléments sont exploités pour identifier les opérations ETL. Enfin, les schémas partiels relatifs à chaque besoin sont consolidés pour obtenir le schéma multidimensionnel et celui des processus ETL. L'avantage de ce travail est de traiter de manière conjointe les données et les processus. Cependant, la contribution se limite à la définition d'un schéma conceptuel. Aucune démarche n'a été fournie pour implanter ce schéma aux niveaux logique et physique.

Le tableau 2.2 montre l'ensemble des approches dirigées par les modèles proposées pour la modélisation de schéma d'entrepôts ou la modélisation des processus ETL. Ce tableau met en relief les caractéristiques liées à l'IDM. Ces caractéristiques montrent particulièrement les aspects de formalisation et de validation. Toutefois, les caractéristiques liées aux nombres et aux types d'opérations ETL considérées par chaque approche et que nous avons résumées dans le tableau 2.1 ne sont pas présentées dans ce tableau, étant donné que la plupart des travaux dirigés par les modèles traite principalement ces mêmes opérations mais les formalisent de manières différentes. Le tableau 2.2 montrent que les travaux de [Zepeda et al., 2008] et de [Carmè et al., 2010] proposent uniquement le niveau PIM qui décrit les sources (src) en utilisant les schémas Entités/Associations (E/A) et l'ED par un schéma multidimensionnel (noté MD). Ces travaux utilisent le langage QVT pour formaliser les correspondances entre le PIM source et le PIM cible. Les travaux de [Mazón and Trujillo, 2009] présentent une approche qui décrit le PIM de l'entrepôt via un profil UML multidimensionnel. Ce PIM est traduit en PSM relationnel en utilisant le langage QVT. Le PSM est à son tour traduit en code SQL en utilisant le langage MOFM2T (Meta Object Facility Model to Text Transformation Language).

D'autre part, les travaux de [Muñoz et al., 2009] et de [El Akkaoui et al., 2011] présentent des approches IDM pour la modélisation des opérations ETL. Les auteurs de [Muñoz et al., 2009] décrivent ces opérations au niveau PIM en utilisant les diagrammes d'activités UML. Le niveau PSM décrit la plateforme `Oracle Warehouse Builder`. Les transformations du PIM en PSM sont formalisées en QVT. Les auteurs de [El Akkaoui et al., 2011] utilisent le BPMN au niveau PIM, le PSM décrit la plateforme `Oracle MetaBase` (OMB). Les transformations entre ces deux modèles sont basées sur des patrons.

Si la validation a été considérée, la plupart de ces approches IDM utilise l'environnement `Eclipse Modeling Framework` (`EMF`) pour la modélisation et la métamodélisation et l'outil `MediniQVT` pour les transformations QVT.

En conclusion, aucun des travaux existants ne fournit une solution complète qui tient compte des trois aspects : modélisation de données, modélisation des

5. OWL :Web Ontology Language

opérations ETL et validation.

Dans la section suivante, nous évoquons le problème lié au volume de données dans un entrepôt et nous présentons les solutions proposées pour le résoudre.

2.5 Réduction de données

Dans un système décisionnel, certaines données abondent, alors qu'elles ne présentent pas un intérêt majeur pour les décideurs. La réduction de données vise à diminuer le volume de données non pertinentes et à obtenir des tailles gérables afin de maîtriser la dimension réelle des données [Udo and Afolabi, 2011]. Dans cette section, nous abordons le problème de volume de données dans les entrepôts. Puis, nous présentons les approches de réduction de données.

2.5.1 Volume de données

Au cours de ces dernières années, le volume des données générées et exploitées dans les entreprises a explosé. Le problème de stockage de données peut se manifester rapidement, surtout si les ressources de stockage des systèmes sont limitées ainsi que leurs capacités de traitement de requêtes. Le recours à des techniques telles que le `Cloud` [Armbrust et al., 2010] n'a pas résolu ce problème. En outre, comme les données vieillissent avec le temps, elle perdent leur intérêt progressivement et leur utilité pourrait s'amoindrir [Iftikhar and Pedersen, 2011]. En effet, le pourcentage de données inactives atteint les 85% [Informatica, 2010]. Les données inactives sont des données non utilisées, non touchées par des requêtes d'interrogation ni de mises à jour. Bien que ce chiffre diffère d'une application à une autre, les systèmes décisionnels qui ont été en production pendant plusieurs années sont susceptibles de contenir des volumes importants de données rarement utilisés, voire non utilisés [Agosta, 2008].

De manière générale, le cycle de vie des données de l'entrepôt commence lorsqu'elles sont insérées dans les systèmes opérationnels. En effet, pour répondre aux besoins des décideurs, ces données sont extraites, transformées et chargées dans l'entrepôt. Au fil du temps, certaines données deviennent obsolètes pour la prise de décision. En se basant sur leur utilité, les données d'un système décisionnel peuvent être :

- *Actives :* dans ce cas, elles doivent être accessibles aux requêtes d'interrogation en ligne (OLAP),
- *Disponibles :* l'accès à ces données est moins fréquent, mais possible.
- *Nécessaires mais inaccessibles :* ces données doivent être gardées en cas de besoins, mais elles ne sont interrogées que exceptionnellement.
- *Inactives :* ces données ne sont plus utiles pour la prise de décision.

Afin d'économiser les ressources et de privilégier les données les plus utiles, il existe deux principales options. La première solution, qui paraît la plus évi-

Caractéristiques	Modélisation de schémas d'ED				Modélisation ETL						Validation		
Approches IDM	PIM	PSM	Code	Transformation		PIM	PSM	Code	Transformation		Env.	Outils M2M	Outils M2T
				M2M	M2T				M2M	M2T			
[Zepeda et al., 2008]	ED : MD Src : ER	-	-	QVT opérationnel	-	-	-	-	-	-	-	-	-
[Carmè et al., 2010]	ED : profil UML MD/ Src : Relationnel	-	-	QVT opérationnel	-	-	-	-	-	-	EMF	Medini QVT	-
[Mazón and Trujillo, 2009]	ED : profil UML MD	Relationnel	SQL	QVT opérationnel	MOF M2T	-	-	-	-	-	EMF	Medini QVT	MOF Script
[Muñoz et al., 2009]	-	-	-	-	-	Activités UML	OWB	-	QVT Opérationnel	-	-	-	-
[El Akkaoui et al., 2011]	-	-	-	-	-	BPMN	OMB	SQL	Patron	MOF M2T	EMF	Grammaire OMB	Acceleo

TABLE 2.2 – Tableau comparatif sur les approches IDM pour la modélisation d'ED et des processus ETL

dente, est de supprimer des données inactives ; la deuxième permet d'agréger ces données. De plus, si on considère d'autres espaces de stockage, il est possible d'utiliser des techniques d'archivage ou de compression de données (par exemple présentes dans le SGBD `Oracle`). Ces techniques permettent de gagner de l'espace disque et de réduire le volume de données pour une meilleure interrogation :

- *Suppression de données :* ensemble de techniques et d'algorithmes permettant de supprimer les données inactives de l'ED [Garcia-Molina et al., 1998].
- *Archivage de données :* permet de déplacer les données dont l'accès est moins fréquent, à partir de l'entrepôt vers un deuxième niveau de stockage. Ceci permet de réduire les coûts, de favoriser la disponibilité du système, et d'améliorer les performances tout en satisfaisant les besoins de conservation et de sécurité des données [Informatica, 2010].
- *Compression de données :* la compression des tables (disponible dès la version `Oracle 9i`) consiste à éliminer les valeurs en doublon trouvées dans les tables d'une base de données [Pöss and Potapov, 2003].
- *Réduction de données :* vise à réduire les volumes importants de données (dans le domaine de fouille de données, de bases de données et des ED) à des tailles gérables et aussi à aider l'utilisateur à maîtriser la taille de la base de données [Udo and Afolabi, 2011].

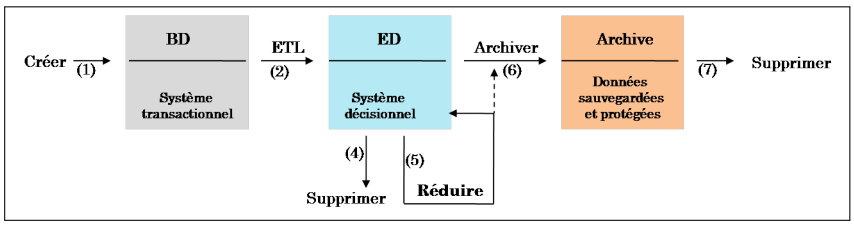


FIGURE 2.1 – Cycle de vie des données

Dans la mesure où nous nous intéressons au problème de réduction de données, nous présentons dans la section suivante, les approches de réduction dans le domaine des entrepôts.

2.5.2 Approches de réduction d'entrepôts

Les techniques de réduction de données ont été utilisées dans le domaine de fouille de données, nous pouvons citer à titre d'exemples les travaux de [Okun and Priisalu, 2007] et de [Udo and Afolabi, 2011] ainsi que dans les bases de données temporelles, notamment dans [Nørvåg, 2006] et [Roddick, 2009].

Dans le cadre des ED relationnels, les auteurs de [Boly et al., 2007] proposent un ensemble de fonctions d'« `Oubli` » permettant de supprimer, de résumer par agrégation et par échantillonnage les données anciennes d'un schéma rela-

tionnel. Sur chaque table, est défini au moyen de spécification, un ensemble de n-uplets à archiver. Parmi ces n-uplets, on peut conserver des échantillons dans l'ED. Les auteurs définissent la notion d'`Âge` de données définie en utilisant des estampilles. Une fonction d'`Oubli` peut contenir plusieurs spécification d'agrégation. Chaque spécification indique un niveau d'agrégation défini en fonction de l'âge de la donnée. Dans [Iftikhar and Pedersen, 2010], afin de garder les données les plus récentes disponibles pour de plus longues périodes, les données dites `vieilles` doivent être réduites progressivement. Avec le mécanisme d'`agrégation granulaire progressive`, les données les plus anciennes sont synthétisées à des niveaux temporels moins détaillés tout en conservant les données les plus récentes à des niveaux de granularité fine. Pour ce faire, la solution proposée introduit une table permettant de stocker les différentes granularités temporelles : `Seconde, Minute, Heure, Mois` et `Année`. Cette table permet de conserver les données anciennes à long terme et à différents niveaux de granularité.

Dans les ED multidimensionnelles, la réduction peut être réalisée par `agrégation` de données. Notamment, les auteurs de [Skyt et al., 2008] présentent une technique permettant l'agrégation progressive de données à des niveaux synthétisés. Ceci se fait en spécifiant des critères sur l'agrégation des données à des niveaux supérieurs dans les dimensions. Dans ce travail, la réduction de dimension se fait à travers un ensemble d'`Actions`. Une action permet d'agréger un fait par rapport à des niveaux de granularité moins détaillés. Un ensemble de contraintes est défini pour garantir que chaque action présente un niveau d'agrégation moins détaillé que l'action précédente. Le modèle proposé requiert que les hiérarchies de dimension soient strictes, c'est-à-dire que chaque valeur d'un niveau soit contenue dans une seule valeur du niveau père. Les hiérarchies doivent aussi être sûres ; chaque valeur d'un niveau est composée d'un ensemble de valeurs dans le niveau inférieur.

D'autres travaux proposent de résumer les données dans les entrepôts de flux de données. Les travaux présentés dans [Han et al., 2005] et [Pitarch et al., 2009] visent à développer des cubes de données matérialisées pour l'agrégation multidimensionnelle et multi-niveaux de flux de données. Les travaux de [Han et al., 2005] sont basés sur un modèle temporel. Tandis que, dans [Pitarch et al., 2009], les auteurs utilisent un modèle temporel et intègrent les préférences de l'utilisateur ainsi que des fonctions de précision pour matérialiser uniquement les parties utiles des flux de données historisées. Le but principal de ces deux travaux consiste à construire des cubes de données compacts pour l'analyse en ligne de flux de données. L'article de [Cuzzocrea, 2009] présente une approche basée sur les hiérarchies pour traiter l'agrégation de données dans les systèmes OLAP. Ces travaux s'intéressent plutôt à l'exécution des requêtes OLAP sur les entrepôts de flux de données.

2.5.3 Synthèse sur les approches de réduction

Les travaux présentés dans la section précédente fournissent différentes solutions permettant de résoudre le problème de réduction dans les entrepôts. Les techniques décrites dans [Han et al., 2005] et [Pitarch et al., 2009] traitent les entrepôts de flux de données et présentent des aspects manuels, le code doit être modifié ou réécrit suite à la moindre modification [Iftikhar and Pedersen, 2010]. Les travaux de [Iftikhar and Pedersen, 2010] présentent des solutions de réduction dans les ED relationnels. Ces travaux se contentent d'agréger les données par rapport à la table temporelle. Alors que les travaux de [Skyt et al., 2008] présentent l'avantage d'agréger les données du fait par rapport à la dimension temporelle, mais aussi aux autres dimensions. Cependant, ces travaux se limitent à agréger les données du fait de manière progressive. Ils ne fournissent aucun moyen pour supprimer des éléments multidimensionnels tels que les dimensions, les faits ou les attributs. D'autre part, ces travaux sont théoriques et la validation n'a pas été traitée [Iftikhar and Pedersen, 2010].

L'étude des approches existantes montre que la réduction se fait principalement par agrégation progressive des données du fait. De manière générale, les approches existantes fournissent un ensemble de techniques pour agréger les données du fait. A notre connaissance, il n'existe pas de démarche qui vise à formaliser, à décrire et à automatiser l'implantation d'un ED réduit.

2.6 Conclusion

Dans ce chapitre, nous avons présenté les approches de modélisation et d'implantation d'ED historisées. Les approches existantes proposent des solutions partielles qui traitent la modélisation de schémas d'entrepôt, indépendamment de la modélisation des opérations ETL. Les approches d'élaboration de schémas d'entrepôts sont classées en trois grandes catégories ; les approches ascendantes qui construisent le schéma multidimensionnel à partir des sources de données, les approches descendantes qui construisent ce schéma à partir des besoins des décideurs et les approches mixtes qui considèrent à la fois les besoins et les sources. Nous avons également présenté les approches de modélisation dirigée par les modèles qui présentent l'avantage de formaliser et d'automatiser l'élaboration du schéma de l'ED. Par la suite, nous avons présenté les approches de modélisation des processus ETL. Certaines ont proposé des modèles spécifiques pour décrire les opérations ETL. D'autres, présentent l'avantage d'exploiter des normes de modélisation notamment, l'IDM.

Nous avons aussi présenté les travaux menés afin de résoudre le problème de réduction de données historisées. La réduction des données multidimensionnelles se fait principalement par agrégation des données de faits. A notre connaissance, aucun de ces travaux n'a fournit une démarche qui vise à formaliser ou à automatiser le processus de réduction à différents niveaux d'abstraction.

Pour pallier ces limites, nous proposons dans le chapitre 3 une démarche dirigée par les modèles qui permet de formaliser et d'automatiser la modélisation et l'alimentation de l'entrepôt. Ensuite, nous présentons dans le chapitre 4 une démarche pour la réduction d'entrepôt de données.

Chapitre 3

Approche dirigée par les modèles pour l'élaboration d'entrepôts de données

3.1 Introduction

Nous avons exposé dans les chapitres précédents les problématiques de recherches auxquelles nous nous sommes intéressés. Le premier problème traité porte sur la modélisation et l'implantation conjointes du schéma multidimensionnel et des opérations de transformation. Ce chapitre présente les solutions que nous proposons afin de répondre à ce problème. Il s'agit d'une approche dirigée par les modèles qui vise à formaliser à automatiser l'élaboration d'ED. Ce chapitre est organisé comme suit. La section 3.2 décrit notre approche. La section 3.3 présente notre modèle multidimensionnel. La section 3.4 définit les transformations automatiques vers les niveaux logique et physique.

3.2 Apperçu de notre approche

Notre objectif est de formaliser et d'automatiser l'implantation de schémas d'ED et des opérations de transformation. Pour ce faire, nous proposons une approche dirigée par les modèles qui fournit un ensemble de modèles permettant de décrire conjointement les structures multidimensionnelles et les opérations de transformation du niveau conceptuel au niveau physique (code d'implantation). La figure 3.1 montre un aperçu de notre proposition. L'entrée de l'utilisateur présente un modèle conceptuel qui décrit l'aspect statique (structures multidimensionnelles) et l'aspect dynamique (opérations de transformation). L'approche fournit

un ensemble de métamodèles et de transformations qui permettent d'aboutir au code de création et d'alimentation de l'ED. Les principes de l'IDM simplifient la tâche du concepteur. Ce dernier construit un premier modèle multidimensionnel (PIM : Platform Independent Model) et les transformations automatiques le traduisent en une succession de modèles de façon à obtenir un modèle physique adapté à la plateforme choisie.

De manière générale, une démarche dirigée par les modèles repose sur les modèles CIM, PIM et PSM ainsi que les transformations automatiques entre modèles. Cependant, chaque démarche présente ses propres caractéristiques, notamment le nombre et le type des modèles et des transformations. Etant donné que le niveau CIM a été traité dans les travaux de [Salinesi and Gam, 2009], [Mazón and Trujillo, 2009], nous nous intéressons aux niveaux PIM, PSM et code. Notre approche repose sur les niveaux suivants :

1. Niveau PIM : Le nombre de modèles indépendants de la plateforme qu'on peut avoir dans une démarche IDM peut varier d'une application à une autre [Xavier, 2005]. Dans notre approche, nous proposons deux niveaux différents de PIM. Le premier niveau présente le modèle conceptuel (PIM multidimensionnel), le second (PIM ROLAP) présente le modèle logique. Ces modèles sont définis comme suit :
   - $PIMmultidimensionnel$: il s'agit du modèle conceptuel de l'ED qui vise à décrire les structures multidimensionnelles de l'entrepôt et à spécifier les opérations de transformation associées. Afin de décrire les structures et les opérations de manière conjointe, nous proposons d'étendre les modèles en constellation [Golfarelli and Rizzi, 1998] [Ravat et al., 2008] via un ensemble d'opérations et d'expressions ETL-OCL, définies en se basant sur le langage `OCL` (Object Constraint Language). Une expression ETL-OCL permet de décrire de manière conceptuelle la formule de transformation (projection, conversion, sélection et agrégation) des attributs multidimensionnels.
   - $PIMROLAP$: le deuxième type de modèles considéré comme indépendant des plateformes décrit le niveau logique. A ce niveau, en fonction de l'ED à développer, le modèle logique peut être un schéma ROLAP, un schéma XML ou un autre. Dans notre cas, le modèle conceptuel est traduit en un modèle ROLAP. Par ailleurs, les expressions ETL-OCL sont traduites en un ensemble d'expressions en algèbre relationnelle.
2. Niveau PSM : à ce niveau les modèles dépendent d'une plateforme particulière telle que `Oracle`[1], `Mondrian`[2], etc. Nous choisissons de traduire le modèle ROALP en un modèle de vues matérialisées spécifique à Oracle. Les expressions algébriques liées au PIM ROLAP sont traduites à leur tour en un ensemble de modèles de requêtes SQL.

1. http ://www.oracle.com/index.html
2. http ://mondrian.pentaho.com/

3. Niveau code : l'approche fournit en résultat le code de création et d'alimentation des structures multidimensionnelles de l'entrepôt.

La transition d'un niveau à un autre se fait de manière automatique en utilisant des transformations de modèles. Nous rappelons qu'il existe quatre types d'approches de transformation (approches opérationnelles, approches relationnelles, approches basées sur la transformation de graphes et approches de manipulation directes). Nous avons opté pour la norme `QVT` [OMG, 2011b] qui combine les aspects relationnels et opérationnels.

Dans notre approche, nous utilisons les transformations de modèles dites `M2M` (`Model-To-Model`) formalisées en QVT pour atteindre deux objectifs :

1. La transformation de modèles qui transforme un modèle source en un modèle cible ; par exemple le PIM multidimensionnel est transformé en PIM ROLAP.

2. La fusion de modèles qui combine plusieurs modèles sources en un modèle cible. Les deux PIM sont fusionnés pour générer le PSM.

Afin de générer le code d'implantation, le PSM est ensuite transformé en code `SQL`. Ce modèle physique est transformé en script SQL via des transformation de modèles vers texte (code) dites `M2T` (`Model-To-Text`).

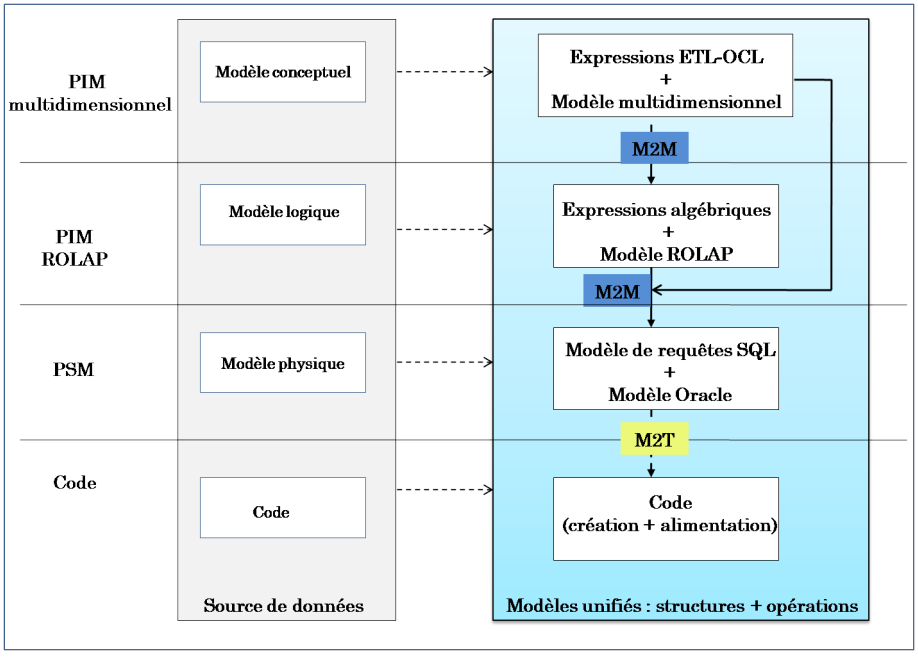


FIGURE 3.1 – Approche dirigée par les modèles pour l'implantation d'ED

Dans les sections suivantes, nous détaillons les différents niveaux de modélisation de notre approche. La section suivante traite du premier niveau PIM

multidimensionnel qui permet de décrire les structures (aspects statiques) et les opérations de transformation (aspects dynamiques) au niveau conceptuel.

3.3 PIM multidimensionnel

La modélisation conceptuelle fournit un haut niveau d'abstraction et vise à faire abstraction des problèmes de déploiement [Rizzi et al., 2006] [Kheir et al., 2013]. Bien qu'il n'y ait pas de modèle standard pour la conception des entrepôts de données [Sen and Sinha, 2005], les modèles en constellation (appelé aussi en étoile) [Golfarelli and Rizzi, 1998] [Ravat et al., 2008] sont largement acceptés pour représenter les bases de données multidimensionnelles. Cependant, ces modèles permettent de représenter les aspects structurels des entrepôts et manquent de mécanismes pour décrire les aspects comportementaux. En effet, un modèle en constellation décrit les besoins décisionnels en termes de faits et de dimension, sans se soucier de la manière dont les attributs cibles de l'entrepôt ont été extraits et transformés à partir des sources. Pour pallier cette limite, nous proposons d'associer à ces structures un ensemble d'opérations et d'expressions qui ont pour objectif la définition des formules de transformation des attributs multidimensionnels.

3.3.1 Structures multidimensionnelles

A l'instar des autres modèles multidimensionnels [Romero and Abelló, 2009], notre modèle repose sur les concepts de `faits` et de `dimensions`. Les définitions suivantes explicitent ces concepts [Ravat et al., 2008].

Définition 5 *Un schéma multidimensionnel S est défini par $(F^S, D^S, Star^S)$ où :*
- $F^S \neq \emptyset$ *: est un ensemble non vide de faits,*
- $D^S \neq \emptyset$ *: est un ensemble non vide de dimensions,*
- $Star^S : F^S \mapsto 2^{D^S}$ *associe chaque fait à un ensemble de dimensions.*

Définition 6 *Un fait noté $F_i \in F^S$ est défini par (N^{F_i}, M^{F_i}) où :*
- N^{F_i} *: est le nom du fait,*
- $M^{F_i} \neq \emptyset$ *: $\{(m_1^{F_i}, f_1), ..., (m_w^{F_i}, f_p)\}$ est un ensemble non vide de couples de mesures $m_j^{F_i}$ associées à des fonctions d'agrégation f_k,*

Définition 7 *Une dimension notée $D_i \in D^S$ est définie par $(N^{D_i}, A^{D_i}, H^{D_i})$ où :*
- N^{D_i} *: est le nom de la dimension,*
- $A^{D_i} \neq \emptyset$ *: $\{a_1^{D_i}, ..., a_u^{D_i}\}$ est un ensemble non vide d'attributs de dimension,*
- $H^{D_i} \neq \emptyset$ *: $\{H_1^{D_i}, ..., H_w^{D_i}\}$ est un ensemble non vide de hiérarchies,*

Définition 8 *Une hiérarchie notée* $H_j^{D_i} \in H^{D_i}$ *est définie par* ($N^{H_j^{D_i}}$, $P^{H_j^{D_i}}$, $AF^{H_j^{D_i}}$, $\{(p_u, af_v), ...\}$) *où :*

- $N^{H_j^{D_i}}$ *: est le nom de la hiérarchie,*
- $P^{H_j^{D_i}} : < p_1, ..., p_n > \neq \emptyset$ *: est une liste non vide de paramètres (attributs identifiant un niveau de granularité d'analyse) organisés du niveau de granularité la plus basse vers la plus élevée,*
- AF^{H_j} *: un ensemble d'attributs faibles permettant de compléter la sémantique des paramètres,*
- $\{(p_u, af_v), ...\}$ *: permet l'association d'attributs faibles aux paramètres.*

D'un point de vue graphique, comme le montre la figure 3.2 un fait est représenté par une boîte rectangulaire colorée en vert et comporte deux sections. La section supérieure montre le nom du fait. La section inférieure définit les mesures du fait. Un fait est lié à une ou plusieurs dimensions représentées par une boîte rectangulaire colorée en rouge et affichant le nom de la dimension. Quant à ses attributs, ils sont décrits par des ronds jaunes qui montrent une organisation hiérarchique de la granularité la plus faible vers la plus haute.

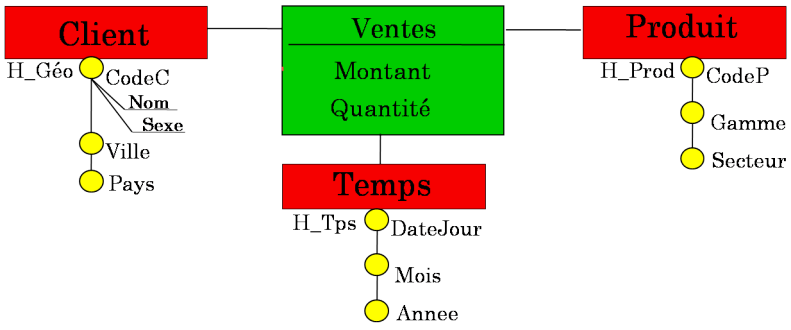


FIGURE 3.2 – Exemple de schéma multidimensionnel

3.3.2 Modélisation des opérations de transformation

Avant d'être chargées dans l'entrepôt, les données extraites des sources subissent un ensemble d'opérations de transformation. L'entreposage de données se fait par le biais des processus d'extraction de transformation et de chargement appelés aussi processus `ETL` [Vassiliadis, 2009]. Comme leur nom l'indique, ces processus permettent (1) d'extraire les données à partir des sources et (2) de les transformer. Lors de cette deuxième étape, les données subissent un ensemble de transformation qui visent à les rendre conformes à un schéma commun ; le schéma de l'entrepôt. Les opérations de transformation peuvent inclure des opérations de conversion, de filtrage, etc. Enfin, une fois les données transformées, celles-ci peuvent être (3) chargées dans l'entrepôt.

La modélisation conceptuelle des processus ETL vise à décrire les correspondances et les transformations nécessaires qui montrent la relation de chaque élément multidimensionnel avec les sources [Simitsis, 2005]. Notre objectif est de fournir une description conceptuelle des opérations de transformation des attributs sources en attributs multidimensionnels de l'ED. Chaque attribut de dimension et chaque mesure sont associés à une expression de transformation. Dans ce qui suit, nous présentons les différentes opérations de transformation. Au fur et à mesure, nous illustrons chaque opération pour expliquer son utilisation. Dans ces exemples les attributs de l'entrepôt sont précédés par `DW`, les attributs sources sont précédés par `SR`.

3.3.2.1 Opérations de transformation de données

Cette section présente les opérations de transformation couramment utilisées [Simitsis and Vassiliadis, 2003] [Trujillo and Luján-Mora, 2003].

1. **Identité**
   Cette opération présente le cas de transformation où la valeur d'un attribut cible est calculée à partir de la valeur d'un attribut source unique, par exemple `DW.codeProduit = SR.codeP`. Par exemple, cette opération est utilisée pour transformer les attributs textuels.

2. **Agrégation**
   Il s'agit d'agréger, à l'aide d'une fonction, un ensemble de valeurs de données sources. L'utilisateur peut définir les attributs à agréger, la fonction d'agrégation à utiliser `(SUM, AVG, MAX, MIN, COUNT, etc.)` ainsi que les critères de regroupement.

3. **Conversion**
   L'opération de conversion modifie les types et les formats de données extraites ou aussi à calculer et dériver de nouvelles données à partir de données existantes. La syntaxe d'une conversion est de la forme `DW.Attribut = Fonction (SR.Attribut)`. Il existe différents types d'opération de conversion :
   - ***Conversion de types de données :*** convertit des données d'un type en un autre type de données ; par exemple convertir un entier en une chaîne de caractères.
   - ***Conversion arithmétique :*** effectue des opérations arithmétiques (addition, multiplication, etc.) sur les données numériques, par exemple `DW.Montant = Produit(SR.Quantité * SR.Prix)`.
   - ***Conversion de chaînes de caractères :*** effectue des transformations sur les chaînes de caractères (la concaténation, majuscule/minuscule, remplacement d'une chaîne par une autre, etc.), par exemple `DW.Nom = Concat (SR.Nom, SR.Prénom)`.

- ***Conversion de format :*** convertit une valeur (monnaie, date, durée, etc.) d'un format à un autre, par exemple `DW.Prix = DollarToEuro (SR.Prix)`.
- ***Conversion de normalisation :*** permet de normaliser les attributs qui contiennent des valeurs identiques pour des éléments de données équivalentes, par exemple on peut substituer les valeurs `Janv` ou `1` avec `Janvier` en utilisant `DW.Mois = NormdMonth (SR.Mois)`.
- ***Génération de valeur :*** génère une valeur constante ou variable à partir d'une fonction, sans forcément être lié à un attribut source, par exemple `DW.DateJ = SysDate()`
- ***Valeur par défaut :*** si une valeur est manquante (null, chaîne de caractère vide, etc.), il est possible de définir une valeur par défaut, par exemple `DW.Attribut = valeur` ou encore `type = unknown`.

4. **Sélection**
   Cette opération filtre les données inutiles et vérifie que les données respectent les contraintes définies. Cette opération vise à sélectionner les données qui répondent à un critère donné. Les données qui ne satisfont pas la vérification peuvent être redirigées vers l'opération `Incorrecte`.

5. **Jointure**
   Cette opération permet de joindre deux entités ou classes sources ayant un ou plusieurs attributs communs qui seront utilisés pour établir la jointure.

6. **Fusion**
   Cette opération vise à établir la connexion entre le modèle de la source et le modèle cible de l'ED. Ce lien se fait en terme de modèle et par la suite au niveau concept jusqu'à atteindre le niveau attribut. L'objectif étant de relier chaque attribut cible de l'ED avec les attributs correspondants au niveau des sources.

7. **Incorrecte**
   Cette opération est utilisée pour générer et gérer les exceptions levées lors de l'exécution d'autre type d'opérations telle que la conversion. Ce mécanisme peut également être utilisé avec l'opération de `Sélection`, comme les données traitées par ces opérations sont contraignantes.

8. **Substitution**
   Cette opération permet de générer des identifiants uniques et uniformes qui remplacent les identifiants sources.

3.3.2.2 Langage ETL-OCL

Dans la littérature, OCL a été étendu pour exprimer des contraintes et des requêtes sur plusieurs types de modèles, notamment, les modèles multidimensionnels. Ces extensions essentiellement orientées définition des contraintes [Pinet

and Schneider, 2009], [Bejaoui et al., 2010] ou interrogation [Pardillo et al., 2010] ne répondent pas à notre objectif de transformation de données sources pour alimenter un ED.

Pourquoi OCL ?

Notre objectif est de fournir un langage qui décrit les opérations présentées précédemment de manière à ce qu'elles soient intégrées avec les structures multidimensionnelles. Nous avons besoin d'un langage qui décrive les opérations de transformation de manière indépendante des plateformes et qui relève du niveau d'abstraction conceptuel. Ce langage devrait faire partie des langages évoqués dans le cadre d'une architecture dirigée par les modèles. Il devrait permettre la navigation entre les différents concepts (classes, faits, dimensions, etc.) et d'accéder à leurs attributs. Enfin, il devrait permettre d'exprimer des opérations de conversion, de sélection et d'agrégation d'attributs.

OCL est un langage précis, simple et supporté par de nombreux outils [Warmer and Kleppe, 2003] et qui répond en grande partie à ces besoins. C'est un langage formel défini par une syntaxe, une grammaire et une sémantique manipulable par machine. Il peut s'appliquer sur différents types de modèle, particulièrement, les modèles `UML` et `MOF`. Il vise à compléter ces diagrammes via des descriptions précises et non ambiguës. Il permet d'exprimer des contraintes et des requêtes sur un modèle de manière précise et indépendante des plateformes. Dans le cadre de l'ingénierie des modèles, l'utilisation du langage OCL est requise pour garantir la transformation automatique entre les modèles.

Formalisation des opérations de transformation en ETL-OCL

Dans un contexte multidimensionnel, chaque attribut de l'ED est dérivé à partir d'un ou plusieurs attribut(s) source(s). Nous envisageons d'utiliser le langage OCL pour formaliser les relations entre des attributs appartenant à des modèles différents. Pour ce faire, il est indispensable d'expliciter les relations entre attributs en termes de relations entre concepts (classe, fait, dimension, etc.). L'objectif est d'établir des liens entre les deux modèles pour que la navigation entre les concepts cibles et les concepts sources soit possible. Afin de définir des expressions ETL-OCL, chaque concept cible (fait ou dimension) est relié à zéro ou un concept (en fonction du type du schéma source : les classes et les associations s'il s'agit d'un diagramme de classes, les classes d'entités et les classes d'associations s'il s'agit d'un modèle E/A, etc.) dans chaque source.

OCL est assez riche pour permettre une description conceptuelle de la plupart des opérations de transformation présentées précédemment. Afin de transformer les données sources, les expressions ETL-OCL utilisent les différentes opérations fournies par la bibliothèque standard d'OCL pour décrire des fonctions sur des chaînes de caractères `(concat(), size(), substring(), toInteger(), toUpperCase(), etc.)` ainsi que des fonctions mathématiques appliquées en particulier sur les attributs numériques `(min(), max(), product(), sum(), etc.)`. Cependant, l'agrégation de données telle qu'elle existe ne permet pas de décrire toutes les possibilités d'agrégation des données sources. Pour pallier cette

limite, nous proposons d'étendre le langage OCL via une opération d'agrégation de données `AGG` définie comme suit :

$AGG(A_{si} \rightarrow AF; A_{sj}[; P_{asj}])$où :

- A_{si} : attribut agrégé,
- AF : fonction d'agrégation prédéfinie dans la bibliothèque OCL `(sum, count, min, max, avg (sum/size), etc.)`,
- A_{sj} : attributs servant à définir le critère de regroupement,
- P_{asj} : prédicat optionnel appliqué aux attributs de regroupement.

De manière générale, une expression OCL est définie dans un contexte (une classe, un attribut ou une opération dans un diagramme de classes). Elle permet d'exprimer des invariants sur un objet ou encore de spécifier que la valeur d'un attribut est dérivée à partir d'un ou plusieurs autres attributs. Elle permet également de définir des post et des pré-conditions sur des opérations. Une expression ETL-OCL est définie dans le contexte d'un attribut multidimensionnel. Elle exprime une relation de dérivation entre cet attribut et un ou plusieurs attributs sources.

Le tableau 3.1 donne les principaux mots clés OCL utilisés pour définir une expression ETL-OCL.

TABLE 3.1 – Mots clés ETL-OCL

Mot clé	**Description**
Context	Donne le contexte de la contrainte ETL-OCL : les mesures d'un fait ou les attributs (paramètres ou attributs faibles) d'une dimension.
Derive	Indique que la valeur de l'attribut multidimensionnel est dérivée à partir des valeurs des attributs sources.
Select (Boolean expression)	Sélectionne un sous-ensemble de données qui répond à un critère donné.

Le tableau 3.2 contient les principales opérations conceptuelles présentées précédemment ainsi que leurs formalisations en ETL-OCL. Le tableau 3.3 présente les différents types d'opération de conversion et les opérations ETL-OCL utilisées pour les formaliser.

TABLE 3.2 – Formalisation des opérations de transformation en ETL-OCL

Opérations de transformation	**Description**	**ETL-OCL**
Identité	Relation de correspondance simple entre un attribut source et un attribut cible de l'ED.	Opérateur d'égalité.
Agrégation	Agrégation des attributs sources en fonction d'autres attributs sources.	L'opération `AGG`.
Conversion	Conversion de la valeur, du type ou du format d'un attribut source.	Opérations définies dans la bibliothèque OCL
Sélection (filtre)	Sélection des attributs qui répondent à critère donné.	Les opérations : `Select`, `Reject`, `If...else`, `collect`, `exists`, etc.
Jointure (classes/ associations)	Jointure de classes ou associations sources ayant des liens directs ou indirects.	Navigation à travers les associations et les classes pour accéder aux attributs.
Fusion (modèles)	Connecte le modèle source et le modèle cible.	Liens établis entre le modèle cible (faits et dimensions) et les modèles sources (classes et associations).
Incorrecte	Détecte les types et les formats de données incorrects.	Exceptions levées quand l'expression ETL-OCL est évaluée invalide.
Substitution	Génère des identifiants uniques et uniformes pour les données cibles de l'ED.	

TABLE 3.3 – Formalisation des opérations de conversion en ETL-OCL

Opérations de conversion	**Description**	**ETL-OCL**
Conversion de types	Modification du type d'un attribut vers un nouveau type auquel il est conforme (un entier peut être transformé en un réel par exemple), sinon l'expression ETL-OCL est invalide.	`Attribut.oclAsType (nouveau type)`
Conversion arithmétique	Conversion d'attributs numériques	Opérations de la bibliothèque OCL : `sum()`, `product()`, `max()`, `min()`,etc.
Conversion de chaines de caractères	Conversion d'attributs textuels	Opérations de la bibliothèque OCL : `concat()`, `size()`, `substring()`, `toUppercase()`, etc.
Conversion de format	permet de calculer une nouvelle valeur à partir d'une valeur existante en appliquant une formule de calcul.	Opération de la bibliothèque OCL.
Génération de valeur	Génère une valeur à partir d'une valeur existante en appliquant une opération.	Opération de la bibliothèque OCL.
Valeur par défaut	Affectation d'une valeur par défaut à un attribut.	Initialisation la valeur de l'attribut.

3.3.3 Exemple d'application

Une centrale de ventes qui alimente des clients a élaboré une base de données pour ses différents clients et produits. Cette base de données permet de suivre les commandes de produits faites par les clients ainsi que l'origine de ces clients. Le diagramme de classes de la source est décrit en bas de la figure 3.3. Chaque produit appartient à une seule gamme et chaque gamme appartient à seul secteur. La centrale des ventes s'approvisionne en effectuant des commandes auprès de ses clients. Chaque commande est passée à une date donnée et comporte une ligne par produit et par client. Chaque ligne indique la quantité commandée. Chaque client est géographiquement positionné par rapport à sa ville et son pays.

Cette source sert de support à l'élaboration d'un entrepôt de données multidimensionnelles pour le directeur des ventes. Ce directeur souhaite analyser les ventes de produits à des clients dans le temps. Plus précisément, ces ventes s'analysent à l'aide des indicateurs (mesures) `quantité` et `montants` des commandes en fonction des axes d'analyse (dimensions) `Client`, `Produit` et `Temps`.

La dimension `Produit` affiche les informations liées au produit, sa gamme et son secteur. La dimension `Client` montre les informations liées au client (`CodeC`, `Nom` et `Sexe`) ainsi qu'à sa position géographique (`Ville` et `Pays`). La dimension Temps présente les niveaux `DateJour`, `Mois` et `Annee`. En haut de la figure 3.3 s'affiche le schéma multidimensionnel qui répond à ces besoins.

Nous complétons ce schéma pour définir les opérations de transformation à partir des attributs sources. Le directeur précise que ses analyses portent uniquement sur les pays européens du bassin méditerranéen et que le montant des ventes correspond à la quantité des ventes multiplié par le prix unitaire de produit. Il précise aussi que l'attribut `Nom` du client doit correspondre à la concaténation de son `nom` et de son `prénom` au niveau de la source.

Pour établir les liens de correspondance source/entrepôt, chaque concept cible (fait ou dimension) est relié à un concept de la source (classe ou association) à partir duquel sont extraits ses attributs. D'un point de vue graphique, ces liens sont représentés par des lignes discontinues.

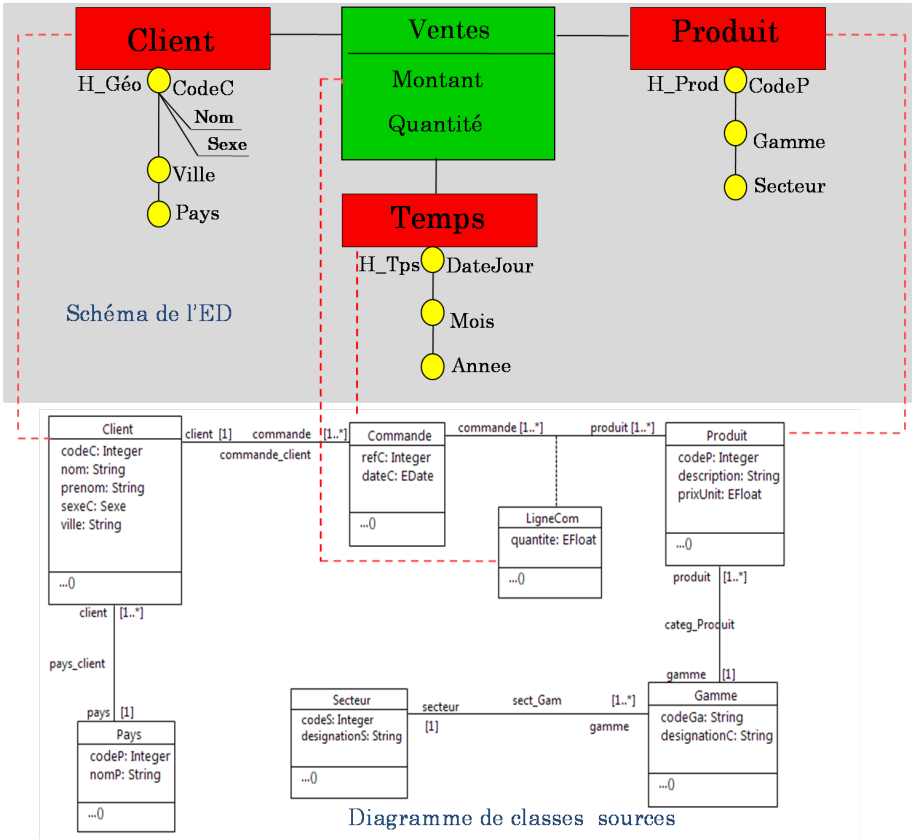


FIGURE 3.3 – PIM multidimensionnel et diagramme de classes source

Les expressions ETL-OCL qui formalisent les transformations sur les différents attributs de ce schéma sont présentées par les figures 3.4, 3.5, 3.6 et 3.7. Dans cet exemple, les classes provenant de la source sont précédées par `SR`. Les faits et les dimensions de l'ED sont précédés par `DW`.

Dans la dimension `Produit` (cf. figure 3.4), les attributs `CodeP`, `Gamme` et `Secteur` proviennent des attributs sources comme suit. L'opération de transformation du premier attribut montre un exemple d'utilisation de l'opération `Identité`. La transformation de l'attribut `Gamme` utilise une opération de `Jointure` des classes sources `Produit` et `Gamme`. Comme défini dans le langage OCL [OMG, 2010], ceci se fait par navigation au travers des rôles.

Context $DW.Produit :: CodeP : Integer$
derive : $SR.produit.codeP$

Context $DW.Produit :: Gamme : String$
derive : $SR.produit.gamme.designationG$

Context $DW.Produit :: Secteur : String$
derive : $SR.produit.gamme.secteur.designationS$

FIGURE 3.4 – Expressions ETL-OCL relatives à la dimension `Produit`

Les transformations qui portent sur les attributs de la dimension `Temps` (cf. figure 3.5) utilisent principalement des opérations de `Conversion` de l'attribut source `dateC` de type `Date`.

Context $DW.Temps :: Jour : Date$
derive : $SR.commande.dateC$

Context $DW.Temps :: Mois : Integer$
derive : $SR.commande.dateC.Month()$

Context $DW.Temps :: Annee : Integer$
derive : $SR.commande.dateC.Year()$

FIGURE 3.5 – Expressions ETL-OCL relatives à la dimension `Temps`

Les expressions ETL-OCL de la figure 3.6 définissent les opérations de transformation des attributs de la dimension `Client` (`CodeC`, `Nom`, `Sexe`, `Ville` et `Pays`). Le paramètre `CodeC` est construit en appliquant l'opération `Identité` à l'attribut source `codeC`. La transformation de l'attribut faible `Nom` utilise une opération de conversion de chaînes de caractères permettant la concaténation

des attributs sources `nom` et `prenom`. L'attribut faible `Sexe` est obtenu en utilisant une opération de conversion permettant de générer la valeur `H` si l'attribut source de la classe `Client` est égal à `Masculin`, la valeur `F` sinon. Le paramètre `Ville` est égal à l'attribut `ville` de la classe `Client`. Le paramètre `Pays` est égal à l'attribut `nomP` de la classe `Pays`. Seuls les pays européens du bassin méditerranéen sont sélectionnés. La jointure se fait en suivant les rôles. L'expression de transformation requiert une opération de `Sélection`.

$$
\begin{array}{l}
\textit{Context } DW.Client :: CodeC : \textit{Integer} \\
\quad \textit{derive} : SR.client.codeC \\
\\
\textit{Context } DW.Client :: Nom : \textit{String} \\
\quad \textit{derive} : SR.client.nom.concat('.').concat(SR.client.prenom) \\
\\
\textit{Context } DW.Client :: Sexe : \textit{String} \\
\quad \textit{derive} : \textit{If } SR.client.sexe =' masculin' \textit{ then } 'H' \\
\textit{else } 'F' \textit{ endif} \\
\\
\textit{Context } DW.Client :: Ville : \textit{String} \\
\quad \textit{derive} : SR.client.ville \\
\\
\textit{Context } DW.Client :: pays : \textit{String} \\
\quad \textit{derive} : \ SR.client.pays \rightarrow \textit{select}(nomP =' France' \textit{ or } nomP =' Italie' \\
\quad \textit{or } nomP =' Espagne' \textit{ or } nomP =' Portugal' \textit{ or } nomP =' Grece')
\end{array}
$$

FIGURE 3.6 – Expressions ETL-OCL relatives à la dimension `Client`

Les expressions ETL-OCL de la figure 3.7 montrent les opérations de transformation des mesures du fait `Ventes`. La `Quantité` est égale à la somme des quantités de la classe d'association source `LigneCom` calculée par code de produit, code de client et date de commande. La transformation de la mesure implique une opération d'`Agrégation`. La transformation de la mesure `Montant` requiert en plus de ces opérations, une opération de `Conversion` qui calcule le produit des quantités et des prix unitaires.

3.3.4 Métamodèle du PIM multidimensionnel

Dans le cadre de l'IDM, tout modèle est instancié à partir d'un métamodèle. Dans notre approche, le métamodèle du PIM multidimensionnel décrit les faits et les dimensions ainsi que les expressions ETL-OCL. Comme le montre la figure 3.8, ce métamodèle présente principalement trois sous-ensembles : les éléments cibles du schéma multidimensionnel (colorés en jaune), les éléments relatifs aux opérations ETL-OCL (colorés en violet) et les éléments relatifs au métamodèle

$Context\ DW.Ventes :: quantit : Real$
$derive : AGG(SR.LigneCom.quantite \rightarrow sum();$
$SR.ligneCom.produit.codeP,\ SR.ligneCom.commande.client.codeC,$
$SR.ligneCom.commande.dateC;\ )$

$Context\ DW.Ventes :: montant : Real$
$derive : AGG((SR.ligneCom.quantite * SR.ligneCom.produit.prixUnit;\ )$
$\rightarrow sum();\ SR.ligneCom.produit.codeP,$
$SR.ligneCom.commande.client.codeC,\ SR.ligneCom.commande.dateC;\ )$

FIGURE 3.7 – Expressions ETL-OCL relatives au fait `Ventes`

source (colorés en bleu). Notons que ce métamodèle, ainsi que tous les métamodèles présentés dans ce mémoire de thèse sont implantés et validés en utilisant la plateforme `Eclipse Modeling Framework` (`EMF`) présentée dans le chapitre 5.

Le métamodèle du PIM multidimensionnel montre les structures présentées dans la section 3.3.1. Un modèle multidimensionnel est composé de `Faits` et de `Dimensions`. Un fait est composé par une ou plusieurs mesures. Une dimension présente une ou plusieurs hiérarchies, composées de plusieurs niveaux. Les attributs multidimensionnels présentent un `Contexte` d'une expression de transformation formalisée en ETL-OCL.

La modélisation de ces expressions réutilise des parties du métamodèles du langage OCL, notamment les opérations standards de la bibliothèque OCL, les exceptions et les types de données. Comme nous l'avons présenté dans la section 3.3.2, une opération ETL-OCL peut être de type `Identité`, `Conversion`, `Sélection` ou `Agrégation`. Les opérations de sélection et d'agrégation utilisent des prédicats. L'agrégation utilise également une fonction de la bibliothèque OCL (`OCLOperation`). Toutes les opérations ETL-OCL font référence à un ou plusieurs attributs sources. Les opérations de `Jointure` et de `Fusion` sont présentées respectivement par des références (associations) inter-classes et inter-modèles (modèle multidimensionnel et paquetage source). L'opération `Incorrect` fait appel une exception OCL. L'opération `UserOperation` permet au concepteur de définir sa propre expression de transformation. En plus de la possibilité d'étendre le métamodèle, cette opération fournit un moyen pour traiter des cas particuliers qui dépendent du cas étudié.

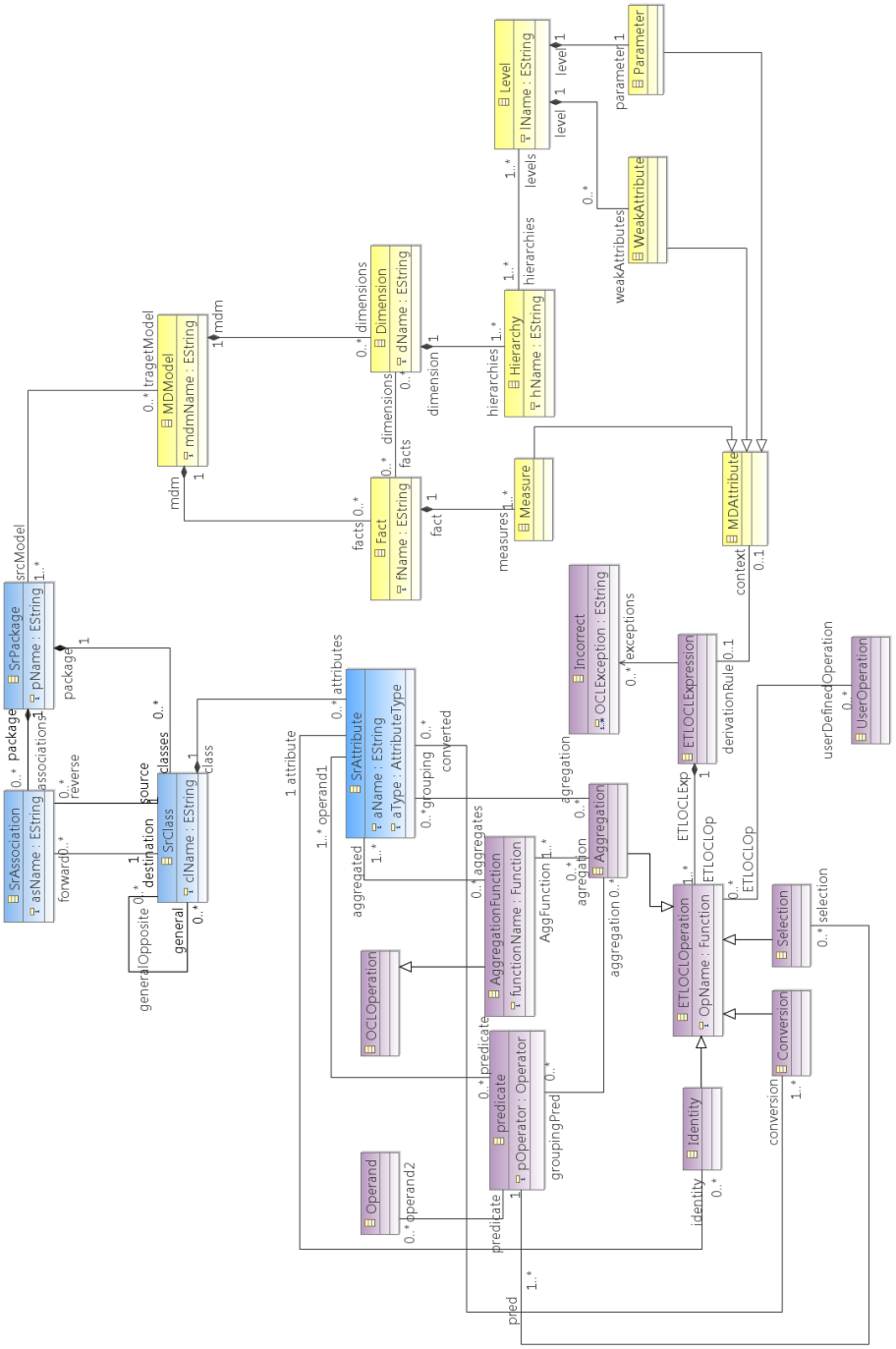


FIGURE 3.8 – Métamodèle du PIM multidimensionnel

3.4 Transformation automatique

Dans cette section, nous présentons les règles de transformation du modèle conceptuel (PIM multidimensionnel) en un modèle logique (PIM ROLAP), puis en modèle physique (PSM) et en script SQL. Pour chaque modèle, nous explicitons les transformations des structures multidimensionnelles (faits et dimensions) et des opérations de transformation.

3.4.1 PIM ROLAP

Conformément aux principes de l'IDM, les modèles logiques sont produits sans l'intervention de l'utilisateur. Le modèle logique de l'ED peut varier d'une application à une autre. Dans un ED, il est possible d'utiliser le `ROLAP` normalisé, dénormalisé ou encore d'autres formalismes tels que les schémas `XML`, etc.

Il existe différents types de schémas `ROLAP`, par souci de clarté, nous détaillons les règles de transformation vers le ROLAP dénormalisé. Les transformations vers le `ROLAP` normalisé ont été présenté dans [Atigui et al., 2010].

3.4.1.1 Transformation de structures

Le modèle multidimensionnel que nous avons présenté dans les sections précédentes, décrit les données en termes de faits et de dimensions. Les règles suivantes montrent comment transformer les faits et les dimensions en un ensemble de tables du schéma ROLAP dénormalisé.

- R1 : Tout schéma multidimensionnel est transformé en un schéma relationnel.
- R2 : Chaque dimension est transformée en une table où :
  - Les attributs correspondent à tous les paramètres et les attributs faibles relatifs aux différentes hiérarchies composant cette dimension,
  - La clé primaire correspond au paramètre appartenant au niveau de granularité le plus bas (appelé paramètre racine).
- R3 : Chaque fait est transformé en une table où :
  - Les attributs correspondent aux mesures et aux paramètres racines relatifs aux dimensions liées au fait,
  - La clé primaire correspond à la concaténation des paramètres racines des dimensions liées au fait.
  - Les clés étrangères correspondent aux paramètres racines des dimensions liées au fait.

Dans la section suivante, nous présentons les règles de transformation des opérations ETL-OCL au niveau PIM ROLAP.

3.4.1.2 Transformation des opérations

Cette section montre comment passer des opérations de transformation du niveau conceptuel en opérations relationnelles du niveau logique [Chrisment et al., 2008]. Ensuite, nous illustrons ces règles. Le tableau 3.4 montre pour chaque opérations du niveau conceptuel, l'équivalent au niveau logique. L'opération `Incorrecte` est une opération qui relève du niveau conceptuel, aucune opération n'est définie pour la traduire au niveau logique. En effet, cette opération lève une exception sur la validité des types et des formats de données : le passage vers le niveau logique n'a lieu que si cette exception est traitée. Aussi, nous précisons que l'opération de `Substitution` est spécifique au modèle ROLAP (relationnel) qui requiert l'identification unique des données d'une table par le biais de clés primaires.

TABLE 3.4 – Expression des opérations de transformation en algèbre relationnelle

Opérations conceptuelles	**Opérations relationnelles**
Identité	Projection (Π [attribut] R).
Agrégation	Fonctions d'agrégation (AGG(R ; group ; attributs ; prédicats)).
Conversion	Conversion (mathématiques, de chaînes de caractères, etc.) et fonctions d'agrégation.
Sélection (critère)	Sélection (critère) : (σ[critère] R)
Jointure (classes)	Jointure de tables reliées via un ou plusieurs attributs ($R_1 \underset{R_1.a_1=R_2.a_2}{\bowtie} R_2$)
Fusion (modèles)	Crée des liens inter-modèles.
Incorrecte	–
Substitution	Génère des clés primaires uniques.

Nous définissons l'ensemble des règles permettant de générer les opérations relationnelles décrites au niveau du métamodèle PIM ROLAP (cf.figure 3.11) à partir des opérations ETL-OCL présentées par le métamodèle du PIM multidimensionnel de la figure 3.8 comme suit :

- R1 : Toute opération de type `Identité` est transformée en une `Projection` où :
  - L'attribut source projeté correspond à l'attribut source référencé par l'opération `Identité`.
  - La table source correspond à la table de l'attribut source.
- R2 : Toute opération de type `Agrégation` est transformée en une projection d'attribut agrégé où :
  - L'attribut source projeté correspond à l'attribut source référencé par l'Agrégation.
  - La table source correspond à la table de l'attribut source.

  - La fonction d'agrégation correspond à la fonction d'agrégation liée à l'opération d'`Agrégation`.
- R3 : Toute opération de type `Sélection` est transformée en une `Sélection` relationnelle où :
  - L'attribut source correspond à l'attribut source référencé par la `Sélection`.
  - La table source correspond à la table de l'attribut source.
  - Le prédicat défini sur l'agrégation correspond à celui défini par l'opération d'agrégation ETL-OCL.
- R4 : Toute opération de type `Conversion` est transformée en une `Conversion` et une `Projection` où :
  - L'attribut source projeté correspond à l'attribut source référencé par la `Conversion`.
  - La table source correspond à la table de l'attribut source.
  - La fonction de `Conversion` correspond à la fonction de `Conversion` ETL-OCL.

3.4.1.3 Exemple d'application

Les tables `Produit`, `Client`, `Temps` et `Ventes` ci-dessous présentées correspondent au résultat de la transformation du schéma multidimensionnel de la figure 3.3 en un schéma `ROLAP` dénormalisé. Les attributs soulignés présentent les clés primaires des tables, les attributs suivis de `#` présentent des clés étrangères.

Produit (CodeP, Gamme, Secteur)
Client (CodeC, Nom, Sexe, Ville, Pays)
Temps (Jour, Mois, Annee)
Ventes (CodeP#, CodeC#, Jour#, Montant, Quantite)

FIGURE 3.9 – Tables du niveau PIM ROLAP relatives à l'exemple des `Ventes`

Les expressions ETL-OCL relatives au modèle conceptuel présentées dans la section précédente sont traduites en expressions algébriques (cf. figure 3.10). Les tables sources préfixées par `SR` correspondent au modèle relationnel de la source, dont le diagramme de classes est présenté dans la figure 3.3. Les tables cibles de l'entrepôt sont précédées par `DW`.

3.4.1.4 Métamodèle du PIM ROLAP

Cette section montre le métamodèle décrivant les structures et les opérations du niveau PIM ROLAP. A ce niveau, les tables représentent les structures cibles de l'entrepôt et celles de la source. La figure 3.11 montre que ce métamodèle est composé de deux sous-ensembles principaux. Les structures (sources et cibles)

DW.Produit = Π[codeP AS CodeP, designationG AS Categorie, designationS AS Secteur]
SR.Produit $\underset{\text{codeGa}}{\bowtie}$ SR.Gamme $\underset{\text{codeS}}{\bowtie}$ SR.Secteur

DW.Temps = Π[dateC AS Jour, Month(dateC) AS Mois, Year(dateC) AS Annee] SR.Commande

DW.Client = Π[codeC AS CodeC, nom ||'.'|| prenom AS Nom, ville AS Ville, nomP AS Pays, sexeC AS Sexe]
SR.Client $\underset{\text{codeP}}{\bowtie}$ SR.Pays

DW.Ventes = Π[dateC AS Jour, codeP AS CodeP, codeC AS CodeC, Quantite, Montant]
((Sum(SR.Commande $\underset{\text{refC}}{\bowtie}$ SR.LigneCom.codeP, SR.Commande.refC, SR.Commande.dateC ; SR.LigneCom.quantite AS quantite ;)) $\underset{\text{refC}}{\bowtie}$
(Sum(SR.Commande $\underset{\text{refC}}{\bowtie}$ SR.LigneCom $\underset{\text{codeP}}{\bowtie}$ SR.Produit ;
SR.LigneCom.codeP, SR.Commande.codeC, SR.Commande.dateC ;
SR.LigneCom.quantite * SR.Produit.PrixUnit AS Montant ;)))

FIGURE 3.10 – Expressions algébriques relatives au schéma des `Ventes`

sont colorées en jaune. Les opérations de transformation sont colorées en violet. Une table est composée d'une ou plusieurs colonnes (attributs) ; elle contient une clé primaire et éventuellement une ou plusieurs clés étrangères. Les opérations de transformation sont représentées par des expressions algébriques. Une expression algébrique définit une table cible (« `TableDefinition` »). Typiquement, elle est composée des opérations de projection, de sélection et de jointure. Ces opérations utilisent des tables et des colonnes sources. Une opération de jointure (« `Join` ») est définie par le biais d'un prédicat de jointure (« `joinPredicate` »). Une opération de sélection (« `Select` ») utilise un prédicat qui définit une condition sur une colonne source. La projection (« `Project` ») peut porter sur un attribut éventuellement agrégé. Les éléments colorés en gris présentent un ensemble d'énumération permettant de définir les types de données (« `DataType` »), les fonctions d'agrégation (« `AggFunction` ») et les opérateurs de comparaison (« `Operator` »).

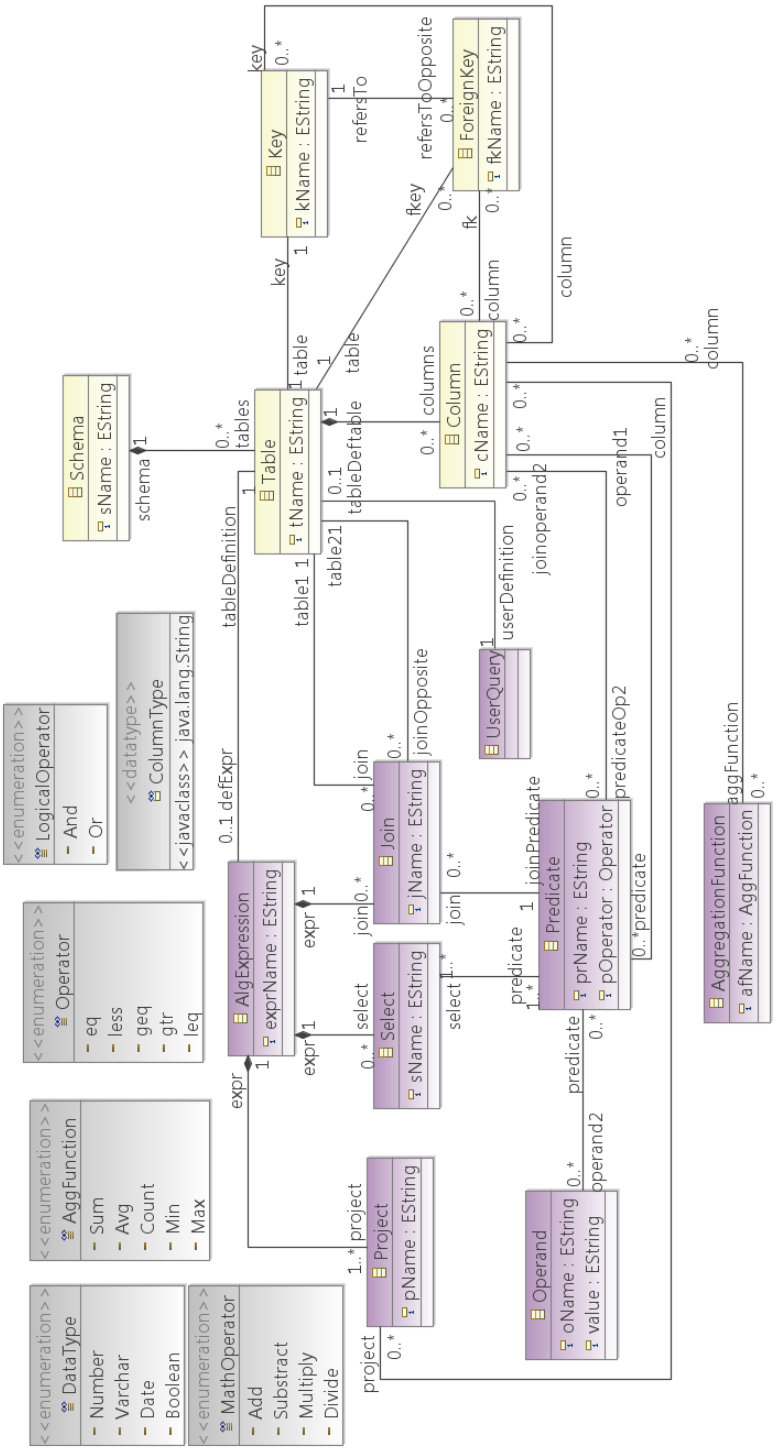


FIGURE 3.11 – Métamodèle du PIM ROLAP

Un PIM peut générer plusieurs PSM qui dépendent de la plateforme choisie. Dans la section suivante, nous présentons un cas d'application utilisant la plateforme `Oracle`.

3.4.2 PSM

Les modèles physiques sont dépendants d'une plateforme spécifique. Nous avons choisi de détailler les règles de transformation permettant de générer les vues matérialisées `Oracle`. L'utilisation des vues matérialisées est avantageuse puisque le calcul, le stockage, la mise à jour et le rafraîchissement sont effectués automatiquement par le SGBD. Dans cette section, nous présentons les règles de transformation de modèles permettant de générer les structures et les opérations du modèle physique.

3.4.2.1 Transformation de structures

La plateforme `Oracle` fournit des techniques spécifiques pour le stockage des données d'un ED, notamment des vues matérialisées et des dimensions que nous utilisons pour implanter l'ED. Ces deux concepts sont définis comme suit.

- **Vue matérialisée :** permet de créer une vue physique d'une table. Le contenu d'une vue matérialisée est calculé dès sa définition (requête de définition) et stocké dans la base de données. Une vue matérialisée permet la duplication des données. Elle peut être utilisée à des fins d'optimisation et de performance dans le cas où la requête associée est particulièrement complexe ou lourde, ou pour faire des réplications de table.
- **Dimension :** est un objet ROLAP en oracle. Il s'agit d'une structure du dictionnaire de données qui identifie les différents éléments multidimensionnels (hiérarchies, niveaux, paramètres et attributs faibles). La création d'une dimension suppose la création de la vue matérialisée correspondante en amont [Bello et al., 1998]. La définition des hiérarchies est basée sur des colonnes existantes dans la table dimension.

Le PSM Oracle est généré de manière automatique par fusion des deux modèles PIM multidimensionnel et PIM ROLAP. En effet, ce modèle physique combine des caractéristiques du modèle logique (tables, colonnes, clés primaires, clés étrangères, etc.) ainsi que des caractéristiques du modèle multidimensionnel (dimensions, hiérarchie, niveau, etc.). Ainsi, les deux PIM (multidimensionnel et ROLAP) sont fusionnés pour générer le PSM. Les transformations utilisent ainsi les deux métamodèles en entrée pour générer des éléments du métamodèle en sortie décrit par la figure 3.16. Le PSM (appelé aussi modèle de code) permet par la suite de générer ultérieurement le script SQL permettant la création et le chargement des structures de l'ED.

Le PSM est composé de vues matérialisées et de dimensions. La définition des vues matérialisées dépend des tables ROLAP et des expressions algébriques, alors que la définition des dimensions dépend des hiérarchies du PIM multidimensionnel. Par conséquent, le PSM est généré par fusion des deux PIM en appliquant les règles suivantes :

- R1 : Tout schéma ROLAP est transformé en un schéma PSM.
- R2 : Chaque table ROLAP est transformée en une vue matérialisée. Les colonnes, la clé primaire et les clés étrangères de la table correspondent respectivement aux colonnes, à la clé primaire et aux clés étrangères de la vue matérialisée.
- R3 : Chaque dimension du PIM multidimensionnel est transformée en une dimension Oracle. Ses hiérarchies, ses paramètres et ses attributs faibles sont respectivement transformés en hiérarchies, niveaux et attributs de la nouvelle dimension.

3.4.2.2 Transformation des opérations

La modélisation conjointe des structures et des opérations au niveau PSM, se fait de la même manière qu'aux niveaux PIM multidimensionnel et PIM ROLAP, les requêtes de création des vues matérialisées permettent aussi leurs définitions. Au niveau ROLAP, les opérations sont décrites par le biais d'opérations relationnelles. Typiquement, au niveau PSM, ces opérations sont exprimées en termes de requêtes (SQL) de définition des vues matérialisées. Ces requêtes permettent de sélectionner et de transformer les données sources requises pour la définition d'une vue matérialisée.

Les opérations relationnelles sont traduites de manière classique comme suit. Les attributs de la clause `Select` correspondent aux attributs de la projection. Les tables de la requête algébrique sont utilisées dans la clause `From`. Les prédicats de jointure et de sélection correspondent aux prédicat de la clause `Where`. Enfin, les attributs de la clause `Group By` sont issus des attributs utilisés par l'opération d'agrégation au niveau logique. Si cette dernière présente une condition sur le regroupement, celle-ci est traduite par la clause `Having`.

3.4.2.3 Exemple d'application

Le code SQL présenté par les figures 3.12, 3.13, 3.14 et 3.15 correspond au résultat de la transformation du schéma des `Ventes` figurant dans les exemples précédents. Le PSM est généré par fusion de deux PIM et traduit par le bais de transformation de modèle vers code. Les figures 3.12 et 3.13 montrent le code de création des vues matérialisées. La figure 3.14 définit les différentes contraintes de définition de clés primaires et étrangères relatives aux différentes vues matérialisées. La figure 3.15 présente le code de création des dimensions.

```
Create MaterializedView   Produit
Build Immediate
Refresh Complete
On Demand
AS Select                 codeP AS CodeP, designationG AS Gamme,
                          designationS AS Secteur
From                      SR.Produit, SR.Gamme
Where                     SR.Produit.codeGa = SR.Gamme.codeGa ;

Create MaterializedView   Temps
Build Immediate
Refresh Complete
On Demand
As Select                 dateC AS DateJour, Month(DateC)
                          AS Mois, Year(DateC) AS Annee
From                      SR.Commande ;

Create MaterializedView   Client
Build Immediate
Refresh Complete
On Demand
As Select                 codeC AS CodeC, concat(nom, prenom) AS Nom
                          , SexeC AS Sexe, ville AS Ville, nomP AS Pays

From                      SR.Client, SR.Pays
Where                     SR.Client.codeP = SR.Pays.codeP And
                          (nomP =' France' or nomP =' Italie'
                          or nomP =' Espagne'
                           or nomP =' Portugal' or nomP =' Grece') ;
```

FIGURE 3.12 – Exemple de script SQL de création et de chargement d'ED (1/4)

3.4.2.4 Métamodèle du PSM Oracle

La figure 3.16 présente le métamodèle du PSM qui décrit principalement deux sous-ensembles. Le premier sous ensemble coloré en jaune définit les structures cibles de l'ED, les vues matérialisées et les dimensions. Une vue matérialisée est une table composée par une ou plusieurs colonnes. Le deuxième sous-ensemble montre les requêtes SQL de définitions des vues colorées en violet. Typiquement, une requête est composée des clauses `Select`, `From`, `Where` et `Group By` qui présente une éventuelle restriction définie au niveau du `Having`. Les structures sources à partir desquelles sont extraites les colonnes cibles, sont des tables.

```
Create Materialized View   Ventes
Build Immediate
Refresh Complete
On Demand
As Select                  codeP AS CodeP, dateC AS DateJour,
                           codeC AS CodeC, Sum(quantite) As quantite,
                           Sum(quantite * prixUnit) As montant
From                       SR.Produit, SR.Client, SR.LigneCom,
                           SR.Commande
Where                      SR.Produit.codeP = SR.LigneCom.codeP And
                           SR.LigneCom.codeC = SR.Commande.refC
                           And SR.Commande.refC = SR.Client.refC
Group                      By SR.LigneCom.codeP,
                           SR.Commande.dateC, SR.Client.CodeC ;
```

FIGURE 3.13 – Exemple de script SQL de création et de chargement d'ED (2/4)

```
ALTER TABLE Produit ADD
CONSTRAINT Produitpk PRIMARY KEY (CodeP) ;
ALTER TABLE Client ADD
CONSTRAINT Clientpk PRIMARY KEY (CodeC) ;
ALTER TABLE Temps ADD
CONSTRAINT Tempspk PRIMARY KEY (DateJour) ;
ALTER TABLE Ventes ADD
CONSTRAINT Ventespk PRIMARY KEY (CodeC, CodeP, DateJour) ;
ALTER TABLE Ventes ADD
CONSTRAINT VentesProduitfk FOREIGN KEY (CodeP)
RFERENCES Produit (CodeP) ;
ALTER TABLE Ventes ADD
CONSTRAINT VentesClientfk FOREIGN KEY (CodeC)
RFERENCES Client (CodeC) ;
ALTER TABLE Ventes ADD
CONSTRAINT VentesTempsfk FOREIGN KEY (DateJour)
RFERENCES Temps (DateJour) ;
```

FIGURE 3.14 – Exemple de script SQL de création et de chargement d'ED (3/4)

```
CREATE DIMENSION  DimProduit
LEVEL             CodeP IS (Produit.CodeP)
LEVEL             Gamme IS (Produit.Gamme)
LEVEL             Secteur IS (Produit.Secteur)
HIERARCHY         HProd (CodeP CHILD OF Gamme
                  CHILD OF Secteur);

CREATE DIMENSION  DimTemps
LEVEL             DateJour IS (Temps.DateJour)
LEVEL             Mois IS (Temps.Mois)
LEVEL             Annee IS (Temps.Annee)
HIERARCHY         HTps (DateJour CHILDOF Mois
                  CHILD OF Annee);

CREATE DIMENSION  DimClient
LEVEL             CodeC IS (Client.CodeC)
LEVEL             Ville IS (Client.Ville)
LEVEL             Pays IS (Client.Pays)
HIERARCHY         HGeo (CodeC CHILD OF Ville
                  CHILD OF Pays)
ATTRIBUTE         CodeC DETERMINES (Nom, Sexe);
```

FIGURE 3.15 – Exemple de script SQL de création et de chargement d'ED (4/4)

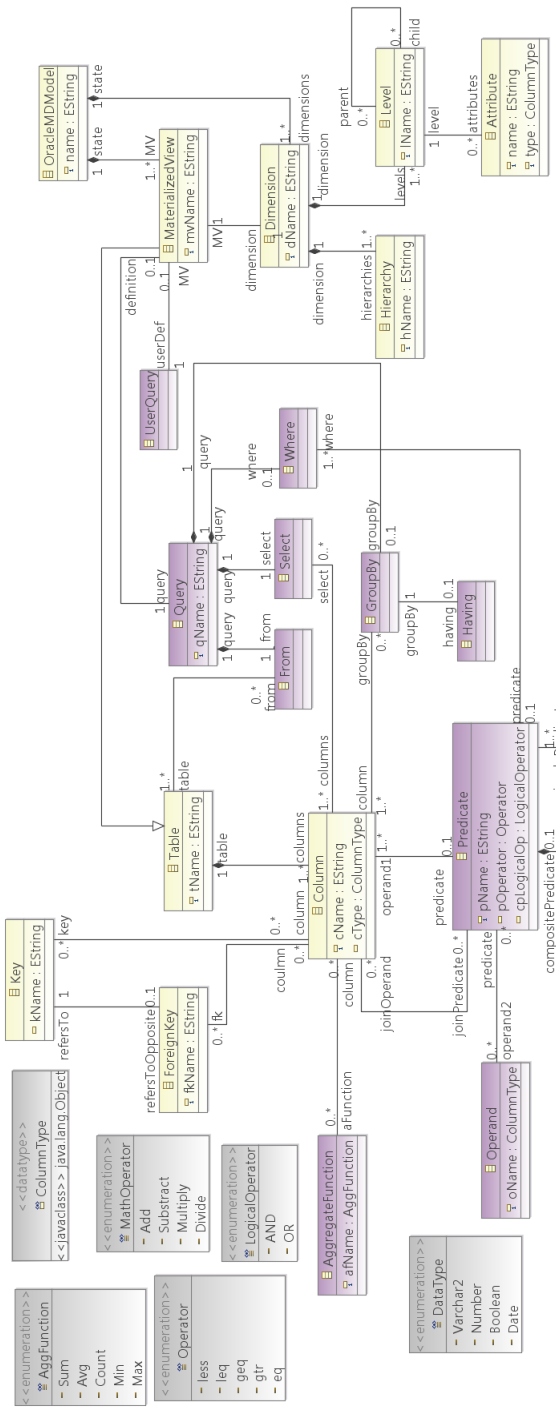


FIGURE 3.16 – Métamodèle du niveau PSM

3.5 Bilan et positionnement

3.5.1 Contributions

Afin de faciliter la tâche du concepteur, nous avons présenté une approche dirigée par les modèles pour l'élaboration automatique d'entrepôts de données. Les données et les opérations présentent deux aspects complémentaires et interdépendants. En considérant la modélisation conjointe et l'implantation automatique de ces deux aspects, nous évitons les problèmes d'intégration et d'interopérabilité rencontrés lors de l'utilisation d'approches différentes. Notre approche IDM est basée sur les éléments suivants :

1. **Niveaux de modélisation**
   - **PIM multidimensionnel :** décrit les données en termes de faits et de dimensions et les opérations de transformation en ETL-OCL. Ce dernier présente une extension du langage OCL permettant de formaliser les expressions de transformation d'attributs multidimensionnels à partir des sources.
   - **PIM ROLAP :** décrit les données en termes de tables et les transformations en termes d'opérations relationnelles.
   - **PSM :** définit le modèle physique de la plateforme Oracle qui décrit les données en termes de vues matérialisées et de dimensions. Les opérations sont formalisées en modèles de requêtes de définitions des vues matérialisées.
   - **Code :** présente le script SQL de création des vues matérialisées et des dimensions au niveau de la plateforme Oracle.
2. **Transformations automatiques**
   - **Transformation de modèles :** Le PIM multidimensionnel est transformé en PIM ROLAP. Les faits et les dimensions sont convertis en tables. Les opérations ETL-OCL sont converties en opérations relationnelles.
   - **Fusion de modèles :** Le PIM multidimensionnel et le PIM ROLAP sont fusionnés pour générer le PSM. Les tables ROLAP sont converties en vues matérialisées. Les dimensions du PIM multidimensionnel sont converties en dimensions Oracle.
   - **Transformation de modèles vers code :** Le PSM est traduit en script SQL.

Les contributions de ce chapitre ont été présentées dans les publications suivantes : [Atigui et al., 2010], [Atigui et al., 2011b], [Atigui et al., 2011a], [Atigui et al., 2012a] et [Atigui et al., 2012b].

3.5.2 Discussion

L'étude de l'état de l'art montre que les travaux existants relatifs à l'élaboration d'entrepôts de données sont nombreux et variées. Nous avons identifié deux limites principales que nos travaux ont permis de pallier : l'automatisation du processus de modélisation et la modélisation conjointe des données et des processus.

1. **Automatisation du processus de modélisation**
   Les approches existantes fournissent différentes solutions pour la modélisation (conceptuelle, logique et physique) des entrepôts de données. Cependant, la plupart d'entre elles omettent l'automatisation du processus d'implantation des modèles conceptuels jusqu'au niveau physique.
   Les travaux de [Prat et al., 2006] fournissent un ensemble de modèles et de formalisation des transformations permettant l'implantation de schémas d'entrepôt au niveau physique. Mais les processus d'implantation n'ont pas été automatisés. Les contributions de [Simitsis, 2005] et de [El Akkaoui et al., 2011] ont permis respectivement de semi-automatiser la transformation du modèle conceptuel en modèle logique et en code. Les travaux de [Mazón and Trujillo, 2009] et de [Muñoz et al., 2009] ont accompli une automatisation partielle du processus d'implantation. Ils ont notamment permis d'automatiser la transformation des modèles conceptuels en modèles physiques, mais n'ont pas abouti à la génération de code.

   Nos travaux, tels qu'ils sont décrit dans ce chapitre, permettent de formaliser et d'automatiser avec IDM le processus de transformation du modèle conceptuel en modèle logique puis au niveau physique en générant le code.

2. **Modélisation conjointe des données et des processus**
   Les problèmes de modélisation du schéma de l'ED et des processus ETL ont été traités de manière séparée ; pourtant les deux sont interdépendants puisque l'alimentation de l'ED repose sur la correspondance entre l'ED et la source. A notre connaissance, seuls les travaux de [Mazón and Trujillo, 2008] et de [Romero et al., 2011] évoquent la prise en compte simultanée de la modélisation des données et des opérations. Dans [Mazón and Trujillo, 2008], les auteurs se limitent à présenter l'architecture générale de l'entrepôt de données partant des sources en allant jusqu'aux magasins. Bien que ces deux problèmes ont été évoqués, l'approche proposée fournit des modèles différents voire des approches différentes comme présentés dans [Mazón and Trujillo, 2008] et [Mazón and Trujillo, 2009] pour la modélisation des données multidimensionnelles et dans [Muñoz et al., 2009] pour la modélisation des processus ETL. Les travaux de [Romero et al., 2011] proposent une démarche pour élaborer le schéma conceptuel de l'entrepôt et des processus d'extraction à partir des sources. Ces travaux ne portent pas sur l'implantation automatique de ce schéma au niveau physique.

Nos contributions, telles qu'elles sont décrites, permettent de formaliser à la fois les schémas de l'entrepôt et les opérations de transformation. La définition conjointe des données et des processus permet d'éviter les problèmes d'incohérence et d'intégrité.

Chapitre 4

Réduction d'entrepôts de données

4.1 Introduction

Dans un entrepôt, les données historisées sont conservées de manière permanente et sont rafraîchies de manière récurrente. De ce fait, l'ED présente un volume croissant de données dans lequel le décideur risque de « se perdre » lors de ses analyses. De plus, il est couramment admis que les données historisées perdent de leur intérêt avec le temps : alors que la granularité des informations doit généralement être importante pour des données récentes [Skyt et al., 2008] ; elle peut être plus faible pour des données anciennes. Par exemple, un décideur peut analyser ses ventes par produit sur les cinq dernières années tandis que, pour les périodes antérieures, ces analyses au niveau du produit seraient sans intérêt ; des analyses au niveau de la gamme suffiraient. Afin de faciliter la tâche du décideur et d'améliorer les performances du système, il est préférable de garder uniquement l'information pertinente. L'idée est donc d'offrir un environnement d'analyse multidimensionnelle adapté aux besoins des décideurs en leur permettant de supprimer dans le temps les niveaux de granularité inutiles pour leurs analyses. La figure 4.1 montre comment les besoins du décideur sont différents d'une période à l'autre. Durant les cinq dernières années, l'analyse des ventes se fait par rapport aux niveaux de granularité les plus bas ; le produit, le client et la date de vente. Alors que durant la période antérieure de 2005 à 2008, ces analyses sont synthétisées par rapport aux gammes de produits en fonction des mois de ventes. Avant 2005, seules les ventes annuelles par secteurs de produits sont conservées.

Notre objectif est donc de proposer un modèle de données multidimensionnelles permettant de prendre en compte la réduction des données afin de ne conser-

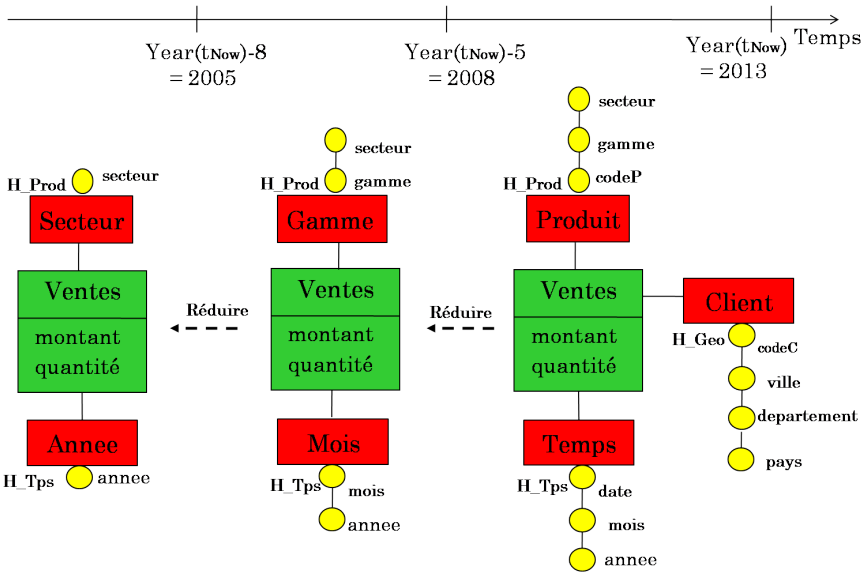


FIGURE 4.1 – Exemple d'application

ver que les données nécessaires aux analyses décisionnelles. Plus précisément, le processus de réduction des données s'effectuera en appliquant un ensemble d'opérations synthétisant sur les données anciennes. Une fois les besoins du décideur définis, nous proposons une approche pour l'implantation automatique d'un ED réduit basée sur l'IDM.

Pour ce faire, nous définissons dans ce chapitre une approche dirigée par les modèles pour la réduction de données multidimensionnelles. En section 4.2, nous présentons un aperçu de notre solution de réduction d'ED. La section 4.3 détaille le premier niveau de modélisation (PIM multidimensionnel) qui porte principalement sur un modèle de données réduites ainsi que sur un ensemble d'opérations et de contraintes. En section 4.4, nous présentons la génération automatique des modèles logique (PIM ROLAP) et physique (PSM) avec réduction de données. Nous présentons au fur et à mesure des exemples d'application permettant d'illustrer nos propositions.

4.2 Processus de réduction

Comme annoncé dans les chapitres précédents, les travaux menés dans le cadre de cette thèse sont formalisés avec l'IDM. Dans cette section, nous présentons le processus de réduction de données. Ensuite, nous exposons les différents niveaux de modélisation IDM d'un ED réduit.

Les données d'un entrepôt sont décrites par un schéma multidimensionnel. La

réduction d'ED porte à la fois sur les données et sur le schéma. Pour ce faire, nous définissons le concept d'`Etat` d'entrepôt réduit. La figure 4.2 montre le processus permettant de transformer un ED (schéma et données) dont les données datent de l'année 2000 en un ED réduit. On souhaite réduire les données sur des intervalles temporels précis. Les données les plus récentes qui datent de plus de 2010 sont conservées de manière détaillée. Les données qui datent de moins de 2010 subissent une réduction croissante dans le temps : plus les données sont anciennes plus elles sont réduites. Le processus de réduction se fait en définissant un ensemble d'états. Un état est défini par un schéma multidimensionnel et un ensemble de données ainsi qu'un intervalle temporel de validité (`T`). Au niveau de l'`état courant`, on conserve le même schéma de l'entrepôt en entrée. Le schéma courant représente le schéma indiquant le plus de niveaux de granularité d'analyse et stocke les données les plus récentes de l'entrepôt. Ensuite une succession d'`états réduits` est définie. Un état réduit contient les données synthétisées durant une période donnée. Celui-ci est toujours construit à partir d'un état origine : le premier état réduit est construit à partir de l'état courant, le second est construit à partir du premier état réduit, etc. Dans une même période de temps, un seul état est défini.

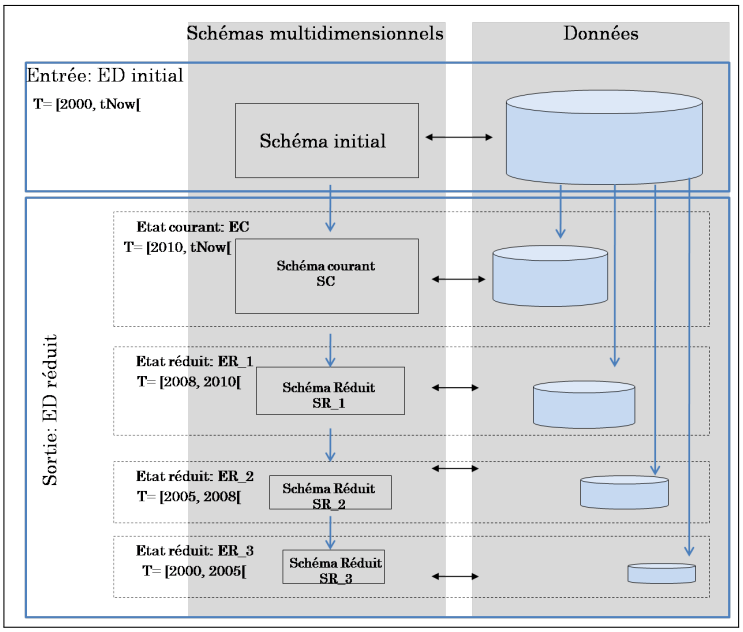


FIGURE 4.2 – Processus de réduction d'ED

Dans le chapitre 3, nous avons présenté notre approche de modélisation et de chargement d'ED qui fournit un ensemble de modèles IDM. Au niveau du `Platform Independent Model` (PIM), l'approche présente un modèle multidimensionnel ainsi qu'un modèle ROLAP. Au niveau du `Platform Specific Model`

(PSM), nous avons présenté un modèle spécifique à la plateforme `Oracle` permettant d'aboutir au code `SQL`. Dans cette même perspective dirigée par les modèles, nous introduisons notre approche de réduction (cf. figure 4.3) qui présentent les mêmes niveaux de modélisation (PIM multidimensionnel, PIM ROLAP, PSM et code) que l'approche présentée précédemment.

Au niveau du PIM multidimensionnel, à partir d'un schéma courant noté SC, un ensemble de schémas réduits notés SR_i est créé. La création d'un schéma réduit à partir d'un schéma origine se fait en appliquant un ensemble d'opérateurs. Les correspondances entre les attributs d'un schéma origine sont formalisées en ETL-OCL. Ensuite, au niveau PIM ROLAP les schémas multidimensionnels réduits sont traduits en schémas `ROLAP` dénormalisés. Les expressions ETL-OCL sont transformées en expressions algébriques. Les schémas réduits du niveau logique sont transformés en schémas spécifiques à la plateforme `Oracle` associés à un ensemble de requêtes `SQL`. Enfin, la démarche fournit le code de création des différentes structures des schémas réduits. Les transformations entre les modèles sont formalisées au moyen du langage QVT de telle sorte que le code final soit généré automatiquement. La génération du code se fait en utilisant le langage de transformation MOF M2T. La section suivante détaille le niveau PIM multidimensionnel qui présente l'entrée au processus de transformation IDM.

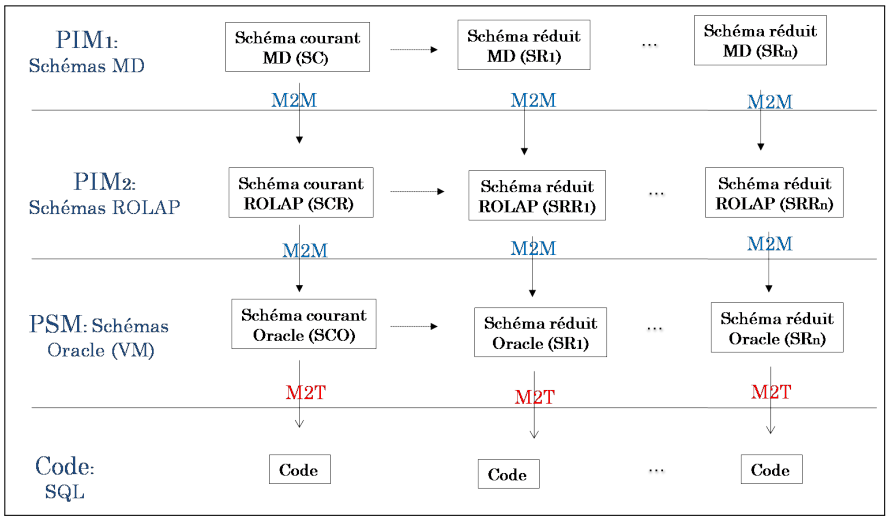


FIGURE 4.3 – Approche dirigée par les modèles pour la réduction d'ED

4.3 PIM multidimensionnel réduit

La réduction de données est un mécanisme qui vise à conserver uniquement l'information utile pour la prise de décision en agrégeant des données anciennes.

Ce mécanisme permet donc de synthétiser les données les moins récentes. L'objet de cette section est de définir les modèles et les opérateurs de réduction de données multidimensionnelles à un niveau conceptuel. La première sous-section présente l'hypothèse préliminaire sur laquelle est fondée notre approche de réduction d'ED. La sous-section 4.3.2 définit les concepts de notre modèle de données multidimensionnelles qui sert de base pour la réduction de données. La sous-section 4.3.3 montre les différents opérateurs permettant de créer un schéma multidimensionnel réduit à partir d'un schéma initial. La sous-section 4.3.4 définit un ensemble de contraintes. La sous-section 4.3.5 met en évidence des aspects particuliers liés à la réduction des modèles en constellation. La dernière sous-section illustre nos différentes propositions.

4.3.1 Entrepôt de données temporelles

Le processus de réduction de données reflète le niveau d'agrégation des données auquel le décideur s'intéresse dans le temps. Ce processus ne peut être défini que dans le cadre d'un entrepôt de données **temporelles**. La dimension temporelle est indispensable pour définir les différents états réduits. Notons que la notion du `Temps` a été traitée dans le cadre des bases de données temporelles [Mkaouar et al., 2011] ainsi que les entrepôts de données temporelles [Golfarelli and Rizzi, 2009]. Deux principales dimensions temporelles peuvent être considérées, à savoir :

- **Le temps de validité :** temps de l'occurrence d'un évènement dans le monde réel ; par exemple la date de vente d'un produit à un client.
- **Le temps de transaction :** temps de stockage de l'événement dans une base de données opérationnelle.

Si l'on souhaite historiser les données d'un entrepôt, il n'est possible d'utiliser le temps de validité que si les sources soient déjà historisées. Dans le cas contraire, il est possible d'utiliser le temps d'extraction des données sources vers l'entrepôt. Dans le cadre de cette thèse et plus particulièrement dans ce chapitre, nous partons de l'hypothèse suivante :

> **Hypothèse :** l'entrepôt de données comporte au moins une dimension temporelle qui correspond au temps de validité.

4.3.2 Modèle de données multidimensionnelles réduites

Comme nous l'avons introduit en section 4.2 et afin de réduire un entrepôt (schéma multidimensionnel et données), nous définissons un ensemble d'`Etats`. Un état est défini par un schéma multidimensionnel, un ensemble d'instances et un intervalle de validité noté T. Le premier état, désigné par `Etat Courant`, est construit à partir de l'entrepôt de données initial duquel il acquiert le schéma ;

mais il conserve uniquement les données correspondantes à son intervalle de validité. Ensuite une suite d'`Etats Réduits` est définie. Un état réduit est construit à partir d'un état précédent appelé `Etat de Référence`. Dans un état réduit, tout élément du schéma (`Fait`, `Dimension` et `Hiérarchie`) est construit à partir d'un élément dans le schéma de l'état de référence appelé composant de `Référence`. Notamment, un fait réduit est issu d'un fait origine dont il acquiert le nom. Le nombre de dimensions associées à ce fait est inférieur ou égal au nombre de dimensions reliées à son fait origine. De même, il peut conserver toutes ou une partie des mesures du fait origine. Toute dimension réduite est construite à partir d'une dimension dans le schéma de référence (dimension origine). Par rapport à son origine, une dimension peut être construite en supprimant des hiérarchies, des niveaux de granularité et/ou des attributs faibles.

L'objectif de cette section est de définir un entrepôt de données réduit. En section 3.3.1 du chapitre précédent, nous avons défini le schéma multidimensionnel. Par rapport à un entrepôt non réduit (identifié par un nom, un schéma et un ensemble d'instances), un entrepôt réduit est défini comme un ensemble d'états dont le premier (ayant la date de validité la plus récente) est appelé état courant, les autres états sont dits états réduits. Un état réduit présente de la même manière un nom et un schéma, mais aussi :

- un état de référence à partir duquel il est dérivé,
- un ensemble d'instances dont la validité est fixée,
- une fonction `Map` qui permet de dériver cet état à partir de son état origine,
- un intervalle temporel de validité.

Un état qu'il soit courant ou réduit est défini comme suit :

Définition 9 *Un entrepôt de données réduit EDR est défini par le couple (N^{EDR}, E^{EDR}) où :*
- *N^{EDR} : est le nom de l'entrepôt,*
- *$E^{EDR} : < E_1^{EDR}, ..., E_n^{EDR} >$: est une liste d'états.*

Définition 10 *Un état $E_i \in E^{EDR}$ est construit à partir d'un état de référence $E_{Ref}^{E_i}$ et défini par $(N^{E_i},\ E_{Ref}^{E_i},\ S^{E_i},\ Ext^{E_i},\ Map^{E_i},\ T^{E_i})$ où :*
- *N^{E_i} : est le nom de l'état,*
- *$E_{Ref}^{E_i}$: est l'état de référence à partir duquel est issu l'état réduit tel que :*

$$E_{Ref}^{E_i} = \begin{cases} Null & Si\ i = 1 \\ E_{i-1} & Sinon \end{cases}$$

- *S^{E_i} : est le schéma de l'état E_i et que l'on notera S_i défini par $(F^{S_i}, D^{S_i}, Star^{S_i})$ où :*
  - *F^{S_i} : l'ensemble de faits du schéma S_i. Soit $F_j \in F^{S_i}$, F_j est défini par (N^{F_j}, M^{F_j}) :*

  - N^{F_j} *: est le nom du fait,*
  - M^{F_j} *:* $\{(m_1^{F_j}, f_1), ..., (m_w^{F_j}, f_p)\}$ *est un ensemble de couples de mesures* $m_k^{F_j}$ *associées à des fonctions d'agrégation* f_x*.*
- D^{S_i} *: est l'ensemble de dimensions du schéma* S_i*. Soit* $D_j \in D^{S_i}$*,* D_j *est définie par* $(N^{D_j},\ A^{D_j},\ H^{D_j})$ *où :*
  - N^{D_j} *: le nom de la dimension,*
  - A^{D_j} *:* $\{a_1, a_2, ..., a_m\}$ *: est un ensemble d'attributs de dimensions,*
  - H^{D_j} *:* $\{H_1, H_2, ..., H_n\}$ *: est un ensemble de hiérarchies. Soit une hiérarchie* $H_w^{D_j} \in H^{D_j}$*.* $H_w^{D_j}$ *est définie par* $(N^{H_w^{D_j}},\ P^{H_w^{D_j}},\ AF^{H_w^{D_j}},\ \{(p_u, af_v),\ ...\})$ *où :*
    - $N^{H_w^{D_j}}$ *: est le nom de la hiérarchie réduite,*
    - $P^{H_w^{D_j}}$ *: est une liste de paramètres de la hiérarchie* $H_w^{D_j}$*,*
    - $AF^{H_w^{D_j}}$ *: est l'ensemble d'attributs faibles de la hiérarchie* $H_w^{D_j}$*,*
    - $\{(p_u, af_v),\ ...\}$ *: permet l'association d'attributs faibles aux paramètres.*
  - $Star^{S_i} : F^{S_i} \mapsto 2^{D^{S_i}}$ *associe chaque fait à un ensemble de dimensions du schéma* S_i*.*
- Ext^{E_i} *: est l'extension de l'état représentant l'ensemble des instances du schéma* S^{E_i}*,*
- Map^{E_i} *:* $\{Op_1\ o\ Op_2\ ...\ o\ Op_n\}$ *: est une fonction qui combine un ensemble d'opérateurs de réduction (défini dans la section 4.3.3) et permettant de créer l'état* E_i *à partir de l'état de référence* E_{Ref}*,*
- $T^{E_i} = [T_{Debut}, T_{Fin}[$ *: intervalle temporel durant lequel l'état est valide. Le début d'un état* E_i *représente la fin de l'état* E_{i+1}*.*
  Pour définir T_i*, nous adoptons un modèle temporel numérique, linéaire et discret qui approche le temps de manière granulaire au travers d'unités temporelles d'observation [Wang et al., 1997]. Un grain temporel est un entier défini relativement à une unité temporelle ; nous adoptons les unités temporelles standard manipulées au travers de fonctions :* `Year`*,* `Quarter`*,* `Month`*,* `Day`*, etc. Par exemple, Year(1990) définit l'instant* 1990 *à l'unité temporelle année. Un instant est un grain temporel. On note* t_{Now} *l'instant présent qui se caractérise par son caractère dynamique, c'est-à-dire que* t_{Now} *change perpétuellement en fonction de l'écoulement du temps. Un intervalle temporel est donc défini par un couple d'instants* t_{deb} *et* t_{fin}*. Ces instants peuvent être fixes (grains temporels) ou bien dynamiques (définis relativement à l'instant* t_{Now}*).*

4.3.3 Opérations de réduction de données

Dans cette section, nous définissons un ensemble d'opérations de réduction permettant de produire un état réduit à partir d'un état de référence. Dans chaque nouvel état réduit, le schéma est construit à partir d'une restriction du schéma de référence. Pour ce faire, nous définissons un ensemble d'opérations de création de faits et de dimensions restreints où seuls les attributs choisis par le décideur sont conservés. Nous définissons également des opérations d'agrégation permettant de recalculer les valeurs des mesures par rapport à un niveau d'agrégation supérieur. Il est aussi possible d'utiliser une opération de sélection de données en appliquant un ensemble de critères. Le domaine de valeurs d'un attribut existant peut aussi être modifié.

4.3.3.1 Opération de création de dimensions

Afin de créer une dimension par restriction d'une dimension de référence, nous définissons l'ensemble des opérateurs élémentaires suivants.

Définition 11 *$CreateD(D, D_{Ref})$: permet de créer la dimension D dans le schéma de l'état réduit à partir d'une restriction de la dimension de référence D_{Ref}.*

Définition 12 *$AddA(D, \{a_1, ..., a_n\})$: permet d'ajouter un ensemble d'attributs $A = \{a_1, ..., a_n\}$ (paramètres et attributs faibles) à la dimension D.*

Définition 13 *$AddH(D, \{h_1, ..., h_m\})$: permet d'ajouter un ensemble de hiérarchies $H = \{h_1, ..., h_m\}$ à la dimension D.*

Définition 14 *$AddL(D, H, < (p_1, \{wa_1, ..., wa_n\}), (p_2, \{wa_1, ..., wa_m\}), (...) >)$: permet d'ajouter une liste de niveaux d'attributs composés d'un paramètre p et d'un ensemble d'attributs faibles (qui peut être vide) $\{wa_1, ..., wa_n\}$ à la hiérarchie H de la dimension D.*

4.3.3.2 Opération de création de faits

Une fois que les dimensions sont créées, l'ensemble des opérateurs suivants est appliqué pour créer un fait à partir d'une restriction d'un fait de référence.

Définition 15 *$CreateF(F, F_{Ref})$: permet de créer le fait F dans le schéma de l'état réduit à partir d'une restriction du fait de référence F_{Ref}.*

Définition 16 *AddM*$(F, \{f_1(m_1), ... f_n(m_l)\})$ *: permet d'ajouter un ensemble de mesures* m_i *avec leurs fonctions d'agrégation* f_i *au fait* F*.*

Définition 17 *Connect*(F, D) *: permet de relier un fait* F *à une dimension* D*.*

4.3.3.3 Opération de sélection

Cette opération permet de sélectionner l'ensemble des valeurs à conserver. La restriction porte aussi bien sur les valeurs des attributs des dimensions que sur celles des mesures du fait. Cette opération est définie comme suit :

Définition 18 *Select*$(pred)$ *où :*
- *pred est un prédicat permettant de préciser que seules les données qui respectent ce prédicat seront conservées. Il est de la forme :*
  - *`Dimension.paramètre opérateur valeur` ou*
  - *`Fait.mesure opérateur valeur`.*

4.3.4 Contraintes

Dans cette section, nous définissons un ensemble de contraintes permettant de compléter les concepts et les opérations définies dans les sections précédentes. Un état réduit est issu d'un état de référence en appliquant un ensemble d'opérateurs. Ces états doivent respecter un ensemble de contraintes que nous définissons comme suit. Soit E_i un état (réduit) défini par (N^{E_i}, $E^{E_i}_{Ref}$, S^{E_i}, Ext^{E_i}, Map^{E_i}, T^{E_i}) dont le schéma réduit S^{E_i} que l'on notera S_i, est défini par $(F^{S_i}, D^{S_i}, Star^{S_i})$ et le schéma de référence que l'on notera S_{Ref} de l'état de référence $E^{E_i}_{Ref}$, est défini par (F_{Ref}, D_{Ref}, $Star_{Ref}$) (cf. définition 10).

1. ***Contraintes non vide :***
   Les éléments (faits et dimensions) d'un schéma multidimensionnel doivent respecter les contraintes suivantes :
   - L'ensemble de faits du schéma S_i doit être non vide : $F^{S_i} \neq \emptyset$,
   - Pour chaque fait $F_j \in F^{S_i}$ l'ensemble des mesures M^{F_j} doit être non vide : $M^{F_i} \neq \emptyset$,
   - L'ensemble des dimensions du schéma S_i doit être non vide : $D^{S_i} \neq \emptyset$,
   - Pour chaque dimension $D_j \in D^{S_i}$ l'ensemble de ses hiérarchies H^{D_j} doit être non vide : $H^{D_j} \neq \emptyset$,
   - Pour chaque dimension, l'ensemble des attributs doit être non vide : $A^{D_j} \neq \emptyset$,
   - La liste des paramètres d'une hiérarchie $H_k \in H^{D_j}$ doit être non vide : $P^{H_k} : < p_1, ..., p_n > \neq \emptyset$.

2. ***Contraintes d'irréversibilité :***
Une des caractéristiques inhérentes à la réduction de données est `l'Irréversibilité` [Skyt et al., 2008]. L'irréversibilité implique que les opérations de réduction doivent spécifier des niveaux d'agrégation supérieurs pour les données les plus anciennes. Dans notre proposition, comme la réduction de données est réalisée via des opérations d'agrégation mais aussi de restriction qui portent sur les schémas et les données, l'irréversibilité implique que plus l'état réduit est récent, plus ses données sont détaillées. La succession d'états réduits dans le temps, où chaque état est construit à partir d'un état de référence, permet d'assurer cette contrainte. Nous définissons l'ensemble de contraintes d'irréversibilité sur un état réduit comme suit. Reprenons la définition de l'état E_i, les contraintes d'irréversibilité ne peuvent être évoquées que dans le cas d'un état réduit où l'état de référence n'est pas nul : $E_{Ref} \neq Null$.

   * L'ensemble des faits F^{S_i} de l'état E_i est issu des faits de l'état de référence tel que : $F^{S_i} \subseteq F_{Ref}$ où chaque fait est défini comme suit :
     - Un fait $F_j \in F^{S_i}$ est défini par (N^{F_j}, M^{F_j}) tel que : $\forall j, \forall k, \exists! F_k \in F_{Ref}$ où :
       - $N^{F_j} = N^{F_k}$: le fait réduit F_j possède le même nom du fait de référence F_k,
       - M^{F_j} : $\{(m_1^{F_j}, f_1), ..., (m_w^{F_j}, f_p)\}$ est un ensemble de couples de mesures $m_u^{F_j}$ associées à des fonctions d'agrégation f_v tel que : toute mesure du fait F_j est issue de l'ensemble des mesures du fait correspondant F_k dans l'état de référence : $\forall x, \forall y, (m_x, f_y) \in M^{F_j}, \forall u, \forall v \exists (m_u, f_v) \in M^{F_k}, m_u = m_x$,
       - Un fait F_j n'est réduit que par rapport aux dimensions auxquelles il est lié, mais pas nécessairement toutes :
         $\forall k, \exists D_k \in Star^{SR}(F_j) \Rightarrow D_k \in Star_{Ref}(F_j)$.
   * L'ensemble de dimensions D^{S_i} de l'état réduit E_i est issu de l'état de référence tel que : $D^{S_i} \subseteq D_{Ref}$ où chaque dimension est définie comme suit :
     - Une dimension $D_j \in D^{S_i}$ est définie par $(N^{D_j}, A^{D_j}, H^{D_j})$ tel que : $\forall j, \forall k, \exists! D_k \in D_{Ref}$ où :
       - N^{D_j} est le nom de la dimension réduite et correspond au nom de la dimension origine,
       - A^{D_j} est un ensemble d'attributs issu des attributs de la dimension de référence D_k : $A^{D_j} \subseteq A^{D_k}$,
       - H^{D_j} est un ensemble de hiérarchies issu des hiérarchies de la dimension de référence H^{D_k} : $H^{D_j} \subseteq H^{D_k}$,
       - Une hiérarchie $H_w^{D_j} \in H^{D_j}$ est définie par $(N^{H_w^{D_j}}, P^{H_w^{D_j}}, AF^{H_w^{D_j}}, \{(p_u, af_v), ...\})$ tel que : $\forall x, \forall k, \exists! H_x^{D_k} \in H^{D_{Ref}}$ où :
         - $N^{H_w^{D_j}}$ est le nom de la hiérarchie réduite $H_w^{D_j}$ identique au nom de la hiérarchie de référence $H_x^{D_k}$ issue du schéma de référence

$S_{Ref} : N^{H_w^{D_j}} = N^{H_x^{D_k}}$,
- $P^{H_w^{D_j}}$ est une liste de paramètres de la hiérarchie $H_w^{D_j}$ issue de la liste des paramètres de la hiérarchie $H_x^{D_k} : P^{H_w^{D_j}} \subseteq P^{H_x^{D_k}}$,
- $AF^{H_w^{D_j}}$ est l'ensemble des attributs faibles de la hiérarchie $H_w^{D_j}$ issu des attributs faibles de la hiérarchie $H_x^{D_k}$ provenant du schéma de référence $S_{Ref} : AF^{H_w^{D_j}} \subseteq AF^{H_x^{D_k}}$,
- $\{(p_u, af_v), ...\}$: permet l'association d'attributs faibles aux paramètres tel que : $\forall y,\ \forall z,\ \exists! u,\ \exists! v,\ p_y \in P^{H_w^{D_j}},\ af_z \in AF^{H_w^{D_j}},\ p_u \in P^{H_x^{D_k}},\ af_v \in AF^{H_x^{D_k}}\ (p_u, af_v) = (p_y, af_z)$.

3. ***Contraintes temporelles :***
   La réduction de données ne peut être appliquée que dans le cadre d'un ED temporelles (cf. section 4.3.1), particulièrement :
   - Dans un même intervalle de temps T, un seul état est défini,
   - Un fait F_i ne peut être réduit que s'il est lié à une dimension temporelle notée : Soit TD un ensemble de dimensions temporelles, $\forall\ i,\ \exists j,\ D_j \in Star(F_i) \wedge D_j \in TD$.

4.3.5 Schéma en constellation et hiérarchies multiples

Dans un schéma en constellation, plusieurs faits peuvent partager une ou plusieurs dimensions, notamment la dimension temporelle. Aussi, les hiérarchies d'une même dimension partagent un ou plusieurs niveaux de granularité. Dans ce cas, la réduction de schéma présente quelques particularités.

- Cas d'une dimension partagée :
  - tous les faits liés à la dimension sont réduits par rapport à des niveaux de granularité identiques : la dimension est conservée avec les niveaux choisis,
  - les faits liés à la dimension sont réduits par rapport à des niveaux de granularité différents : la dimension est dupliquée en plusieurs dimensions ; chaque dimension est liée à son propre fait réduit par rapport à ses niveaux de granularité.
- Cas d'une dimension à hiérarchies multiples :
  - le niveau de granularité plus faible choisi pour la réduction est un niveau commun à toutes les hiérarchies. Ce niveau est considéré comme la nouvelle racine de la dimension et seules les hiérarchies ayant des niveaux de granularité supérieurs sélectionnés dans la réduction sont conservées,
  - un niveau de granularité plus faible est choisi par hiérarchie (gamme et marque de produit). Dans ce cas, on sélectionne le niveau de granularité plus faible et partagé par toutes les hiérarchies. Ce niveau est considéré comme le niveau racine de la dimension.

Les dimensions issues d'une même dimension sont définie comme suit :

Soit un état réduit ER_i défini par (N^{ER_i}, $E^{ER_i}_{Ref}$, S^{ER_i}, Ext^{ER_i}, Map^{ER_i}, T^{ER_i}) dont le schéma S^{ER_i} est défini par ($F^{ER_i}, D^{ER_i}, Star^{ER_i}$) et le schéma de l'état de référence est noté S_{Ref}

$\forall j, D^{ER_i}_j \in D^{ER_i} = \{D_1, ..., D_x\}$ un ensemble de dimensions, D^{ER_i} issue de $D_k \in D_{Ref}$ $\forall j \in [1, x], D_j$ est définie comme suit :

– Les noms des dimensions correspondent à la concaténation du nom de la dimension de référence et d'un compteur allant de 1 à x.

$$N^{D_j} = \begin{cases} N^{D^{SO}_k} & \text{Si } \|D^{SR_i}\| = 1 \\ N^{D^{SO}_k} + j & j \in [1, x] \text{ Sinon} \end{cases}$$

– $A^{D_j} \in A^{D^{SO}_k}$: l'ensemble d'attributs de la dimension est issu de l'ensemble des attributs de la dimension correspondante dans le schéma origine,

– $H^{D_j} \in H^{D^{SO}_k}$: l'ensemble de hiérarchies appartenant à la dimension est issu des hiérarchies de la dimension correspondante dans le schéma origine.

Lorsqu'une dimension est dupliquée, un même fait ne peut pas être associé à deux dimensions provenant de la même dimension :
Soit D_x et D_y deux dimensions provenant de la même dimension D :
– C1 : chaque dimension est associée à un fait qui ne peut pas être lié à une autre dimension issue de la même dimension :
$Star^{-1}(D_x) \cap Star^{-1}(D_y) = \emptyset$
– C2 : la liste des attributs d'une dimension est différente de celle des attributs d'une autre dimension provenant de la même dimension orgine :
$\exists i, A_i \in A^{D_x} \wedge A_i \notin A^{D_y}$.

4.3.6 Métamodèle d'ED réduits

Afin de décrire un ED réduit, le métamodèle présenté par la figure 3.8 du chapitre précédent est adapté comme le montre la figure 4.4. Ce métamodèle présente principalement trois sous-ensembles. Le premier sous-ensemble, coloré en jaune décrit la structure de l'ED, comme un ensemble d'états composés de faits et dimensions. Les éléments colorés en bleu montrent le métamodèle (diagramme de classes UML) de la source à partir de laquelle est alimenté l'ED avant réduction. L'ensemble des opérations est coloré en violet.

Comme présenté dans le chapitre précédent, au niveau PIM multidimensionnel, les données sont décrites en termes de faits et de dimensions. A ce niveau, les transformations des sources vers l'ED sont formalisées via un ensemble d'opérations ETL-OCL. Dans le contexte d'un ED réduit, nous avons défini un ensemble

d'états (« `State` ») défini par un nom et un intervalle de validité (« `Tstart` » et « `Tend` »). Les données d'un état sont extraites à partir des données d'un état de référence décrit, lui-même, par un schéma multidimensionnel. Ainsi, les éléments multidimensionnels peuvent être cibles (appartenant à l'état défini) ou sources (appartenant à l'état de référence).

Par rapport au métamodèle initial, un attribut multidimensionnel (« `MDAttribute` ») est un attribut qui peut être un attribut à définir (à transformer), dans ce cas, il représente le contexte d'une expression ETL-OCL. Il peut également, être utilisé dans la formule de transformation d'un autre attribut, dans ce cas il est référencé par une opération ETL-OCL. La classe « `Attribute` » représente la classe mère d'un attribut multidimensionnel (« `MDAttribute` ») et d'un attribut source « `SrcAttribute` ». Cette classe permet de préciser qu'un même attribut multidimensionnel peut être un attribut cible, ou un attribut source appartenant à un état de référence. C'est au niveau de cette classe que les associations avec les opérations de transformation se font. Ces associations montre que l'attribut est utilisé dans une expression de définition d'un attribut cible. L'association « `context` » montre qu'un attribut multidimensionnel est défini via une expression ETL-OCL « `ETLOCLExpression` ». La création d'un schéma multidimensionnel sans réduction reste possible en définissant un état initial à partir de la source. En ce qui concerne les métamodèles du PIM ROLAP et du PSM, la prise en compte de la réduction est plus simple dans la mesure où les métamodèles source et cible sont les mêmes. Une table cible est alimentée à partir d'une table appartenant à la source (de l'ED initial) ou à l'état de référence. Les métamodèles présentés dans le chapitre précédent ont été adaptés en ajoutant simplement le concept d'état et les intervalles de validité.

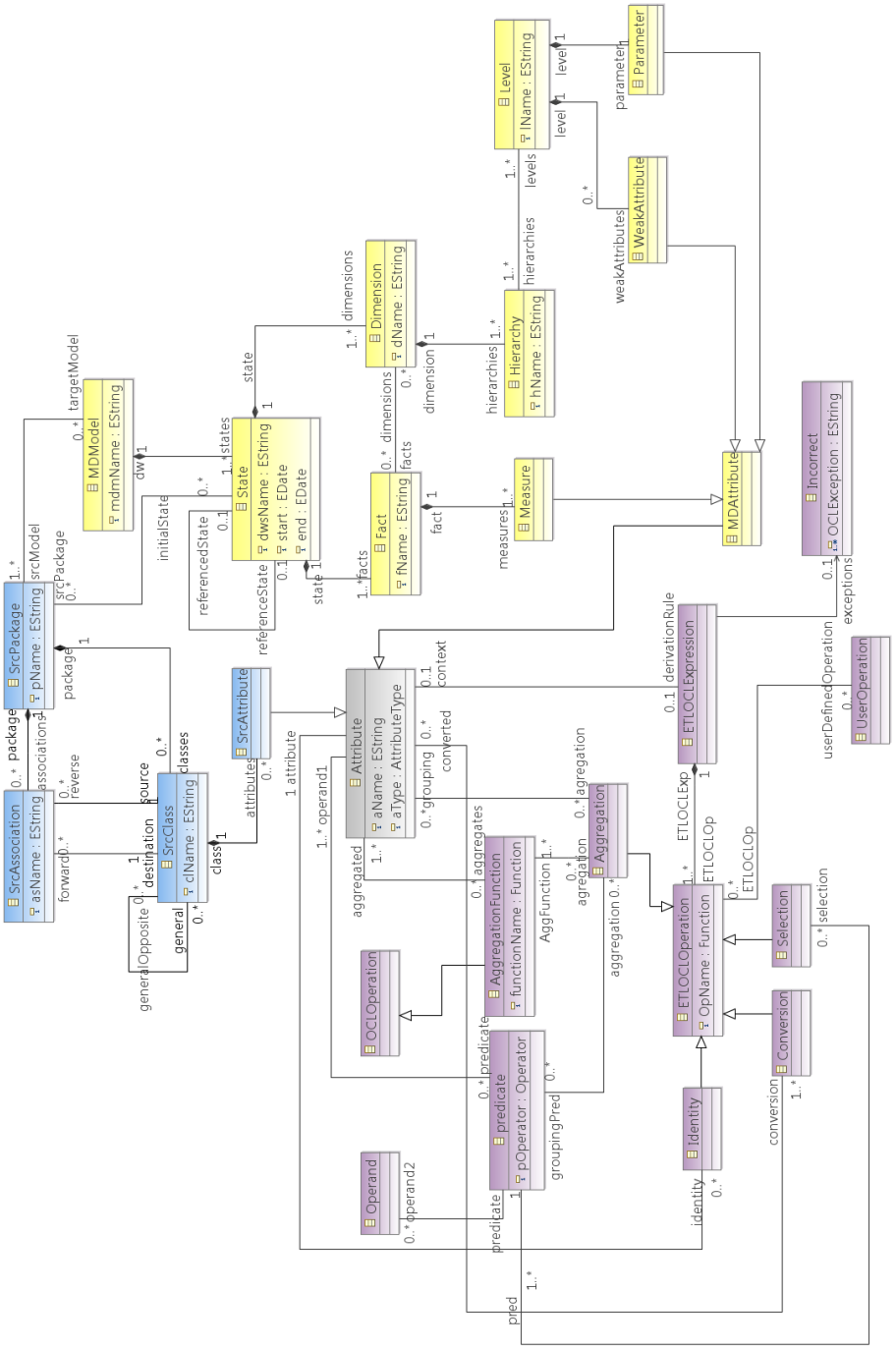


FIGURE 4.4 – Métamodèle du niveau PIM multidimensionnel avec réduction

4.3.7 Exemple d'application

Afin d'illustrer les contributions de ce chapitre, nous étendons l'exemple de l'application commerciale des ventes pour considérer également l'analyse des achats. Plus précisément, l'entrepôt élaboré permet l'analyse des montants et des quantités de ventes ainsi que les prix d'achats de produits qui datent de l'année 2000 ($T = [2000, t_{Now}[$). Comme le montre la figure 4.5, l'analyse des `Ventes` se fait par rapport aux dimensions `Clients`, `Produits` et `Temps`. Ces deux dernières sont également utilisées dans l'analyse des `prix` d'`Achats`. La dimension `Produit` est identifiée par les attributs suivants : `codeP`, `gamme`, `marque` et `secteur` organisés selon les hiérarchies `H_marque` et `H_gamme`. Les niveaux de granularité `codeP` et `secteur` sont partagés par les deux hiérarchies, alors que les niveaux se trouvant au milieu sont des niveaux spécifiques à chaque hiérarchie. La dimension `Temps` contient une seule hiérarchie `H_Tps` présentant les niveaux de granularité `date`, `mois` et `annee`. Enfin la dimension `Client` présente trois hiérarchies différentes, à savoir, `H_age` (`codeC`, `age` et `tranche`), `H_pays` (`codeC`, `ville`, `pays` et `continent`)) et `H_zone` (`codeC`, `ville`, `zone` et `continent`).

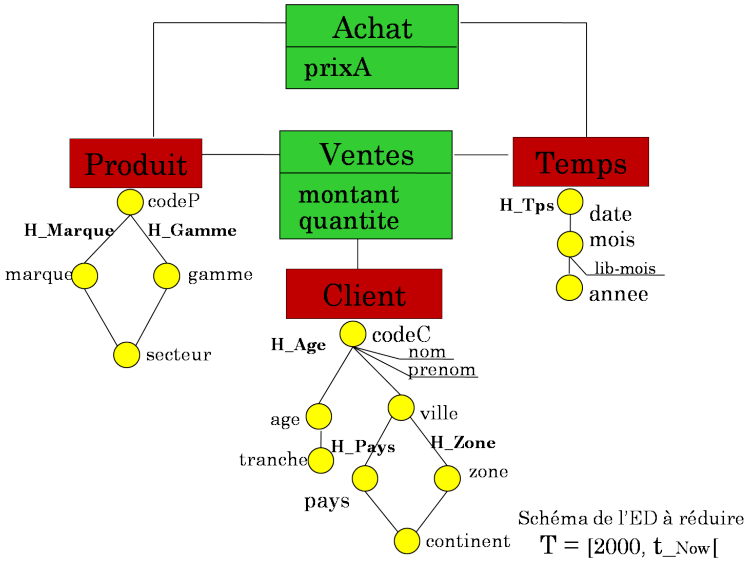


FIGURE 4.5 – Schéma multidimensionnel de l'entrepôt analysant les `Ventes` et les `Achats`

A partir de cet ED, l'objectif est de construire un ensemble d'états (courant et réduits) permettant de synthétiser les données dans le temps pour réduire les données tout en répondant aux besoins d'analyse. Le décideur souhaite garder les données détaillées depuis 2007. Ainsi, l'état courant (EC) est créé pour stocker uniquement les données valides durant $[2007, t_{Now}[$. EC est défini par : (EC, ED_VA, S^{EC}, Ext^{EC}, Map^{EC}, $[2007, t_{Now}[$).

Le schéma multidimensionnel de cet état S^{EC} est identique à celui de l'ED de départ présenté par la figure 4.5. La fonction Map^{EC} est composée d'un ensemble d'opérateurs permettant de créer exactement le même schéma que l'ED de départ.

Les niveaux de détails présentés dans l'état courant n'ont pas d'intérêt pour les analyses décisionnelles portant sur les données antérieures à 2007. Dans un premier temps, le décideur souhaite conserver une synthèse des quantités et des montants **mensuels** de ventes en fonction des produits et des **villes** de clients. Ceci est également le cas pour les prix d'achats analysés de manière **mensuelle** selon les produits. Pour ce faire, un premier état réduit est construit à partir de l'état courant EC. Celui-ci, présenté par la figure 4.7, décrit les données conservées à $T_1 = [2005, 2007[$. L'état réduit ER_1 est défini par (ER_1, EC, S^{ER_1}, Ext^{ER_1}, Map^{ER_1}, $[2005, 2007[$). La figure 4.6 décrit la fonction Map^{ER_1} permettant de dériver ER_1 à partir de son état de référence EC. Le schéma de cet état (cf. 4.7) permet l'analyse des `Ventes` et des `Achats` en fonction des dimensions `Temps` et `Produits`. Durant cette période, la marque de produits n'étant pas utile pour l'analyse des ventes et des achats, ceci est également valable pour les dates. De même, l'analyse des ventes selon les clients n'a d'intérêt que par rapport à sa position géographique et non pas par rapport à son âge ; ainsi seules les hiérarchies `H_pays` et `H_zone` sont conservées. Le niveau de granularité plus bas et commun aux deux hiérarchies : `ville`, représente le paramètre racine de la dimension `Client`.

Dans un second temps, le décideur souhaite synthétiser les données les plus anciennes. Un état réduit ER_2 est donc créé à partir de l'état ER_1 afin de décrire les données valides durant $[2000, 2005[$. Cet état permet de stocker les montants et les quantités des ventes **annuels** par **gamme** de produit et par **pays** de client. Le décideur souhaite par contre garder les prix d'achats **mensuels** par **gamme** de produit. Seules les ventes dont le montant dépasse mille euros sont conservées.

Comme le montre la figure 4.9, les `Ventes` et les Achats sont analysés en fonction des `gammes` et des `secteurs` de `Produit`. Cependant, alors que l'analyse des ventes se fait de manière **annuelle**, celle des `Achats` se fait de manière **mensuelle**. Ainsi, à partir de la dimension `Temps` sont créées les deux dimensions : `Temps_1` et `Temps_2` avec, respectivement, les paramètres racines `mois` et `annee`. Dans la dimension `Client`, seule la hiérarchie `H_Pays` est conservée avec les niveaux `pays` et `continent`.

L'état réduit ER_2 est défini par (ER_2, ER_1, S^{ER_2}, Ext^{ER_2}, Map^{ER_2}, $[2000, 2005[$) tel que la fonction Map^{ER_2} qui permet de dériver ER_2 à partir de son état de référence ER_1 est composée de l'ensemble des opérateurs de réduction présentés dans 4.8.

$$
\begin{aligned}
Map^{ER_1} = \; & \{Create(Produit,\, EC.Produit) \\
& \circ\, AddA(Produit,\, \{codeP,\, gamme,\, secteur\}) \\
& \circ\, AddH(Produit,\, \{H\_Gamme\}) \\
& \circ\, AddL(Produit,\, H\_Gamme, \\
& \quad < (codeP, \emptyset), (gamme, \emptyset), (secteur, \emptyset) >) \\
& \circ\, CreateD(Client,\, EC.Client) \\
& \circ\, AddA(Client,\, \{ville,\, zone,\, pays,\, continent\}) \\
& \circ\, AddH(Client,\, \{H\_Pays,\, H\_Zone\}) \\
& \circ\, AddL(Client,\, H\_Pays, \\
& \quad < (ville, \emptyset), (pays, \emptyset), (continent, \emptyset) >) \\
& \circ\, AddL(Client,\, H\_Zone, \\
& \quad < (ville, \emptyset), (zone, \emptyset), (continent, \emptyset) >) \\
& \circ\, CreateD(Temps,\, EC.Temps) \\
& \circ\, AddA(Temps,\, \{mois,\, lib\_mois,\, annee\}) \\
& \circ\, AddH(Temps,\, \{H\_Tps\}) \\
& \circ\, AddL(Temps,\, H\_Tps,\, < (mois, \{lib\_mois\}), (annee, \emptyset) >) \\
& \circ\, CreateF(Ventes,\, EC.Ventes) \\
& \circ\, AddM(Ventes,\, \{Sum(quantite),\, Sum(montant)\}) \\
& \circ\, Connect(Ventes,\, \{Produit,\, Client,\, Temps\}) \\
& \circ\, CreateF(Achats,\, EC.Achats) \\
& \circ\, AddM(Achats,\, \{Sum(prix)\}) \\
& \circ\, Connect(Achats,\, \{Produit,\, Temps\}) \\
& \circ\, Select((Temps.annee < 2007)\, and \\
& \quad (Temps.annee >= 2005))\}
\end{aligned}
$$

FIGURE 4.6 – Fonction Map du premier état réduit (Map^{ER_1})

4.4 Transformation automatique

Nous avons défini dans la section précédente un ED réduit comme un état courant et une suite d'états réduits. L'objectif de cette section est de présenter le mode de génération automatique du code de création et de chargement des différents états élaborés. L'idée est donc de transformer l'ensemble des états définis au niveau du PIM multidimensionnel en un ensemble de tables du PIM ROLAP dénormalisé et par la suite en vues matérialisées du niveau PSM.

Au niveau PIM multidimensionnel, chaque état est défini à partir d'un état de référence ; il en est de même pour son schéma. Ce cas de figure peut être considéré comme un cas particulier de création et de chargement d'un état cible (dans le chapitre précédent : l'ED) à partir d'un état source (dans le chapitre précédent : la source de données). Afin d'implanter les états du PIM multidimensionnel au niveau physique, il suffit de formaliser les formules d'extraction des attributs de l'état cible à partir de l'état de référence en ETL-OCL et d'appliquer les mêmes règles de transformation pour générer le code `SQL` ; ceci tout en suivant les mêmes

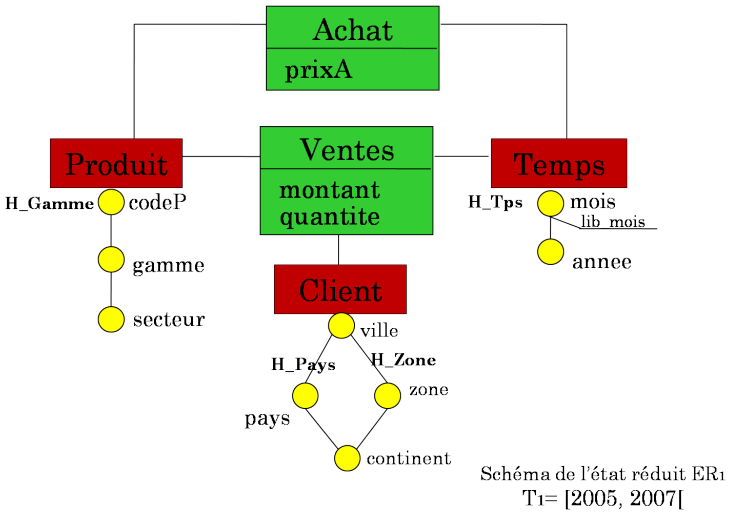


FIGURE 4.7 – Réduction de l'entrepôt analysant les ventes et les achats : schéma de l'état réduit ER_1

étapes définies dans le chapitre précédent. En effet, le langage `ETL-OCL` a été initialement défini pour formaliser les relations de correspondance des attributs de l'ED avec les attributs sources ; nous pouvons l'appliquer pour formaliser la transformation d'un attribut d'un état donné à partir des attributs de l'état de référence.

Dans ce qui suit, nous illustrons l'utilisation du langage ETL-OCL dans le cadre d'un ED réduit. Nous rappelons au fur et à mesure les règles de transformation et les niveaux IDM de cet exemple (PIM multidimensionnel et expressions `ETL-OCL`, PIM ROLAP et les expressions algébriques, le niveau PSM et le code `SQL`).

Nous reprenons l'exemple de la section 4.3.7 qui présente un état courant (EC) et deux états réduits (ER_1 et ER_2). Un état est construit à partir d'un état de référence, afin d'illustrer les différents niveaux de modélisation IDM relatifs à cet état. Le niveau PIM multidimensionnel de cet état dont le schéma est présenté dans la figure 4.7 est décrit également par les expressions ETL-OCL de la figure 4.10.

Afin de transformer le schéma de l'état ER_1 en schéma `ROLAP` dénormalisé, nous appliquons les règles présentées dans la section 3.4 du chapitre précédent. La transformation de l'ensemble des schémas des états du niveau PIM multidimensionnel, qu'ils soient courants ou réduits, se fait en suivant les mêmes règles de transformation d'un schéma multidimensionnel sans réduction. Ceci a été présenté dans la section 3.4 du chapitre 3.

Les tables de la figure 4.11 correspondent au résultat de la transformation du

$$
\begin{aligned}
Map^{ER_2} = \; & \{CreateD(Produit,\ ER_1.Produit) \\
& \circ\, AddA(Produit,\ \{gamme,\ secteur\}) \\
& \circ\, AddH(Produit,\ \{H\_Gamme\}) \\
& \circ\, AddL(Produit,\ H\_Gamme, \\
& \quad < (gamme, \emptyset), (secteur, \emptyset) >) \\
& \circ\, CreateD(Client,\ ER_1.Client) \\
& \circ\, AddA(Client,\ \{pays,\ continent\}) \\
& \circ\, AddH(Client,\ \{H\_Pays\}) \\
& \circ\, AddL(Client,\ H\_Pays,\ < (pays, \emptyset), (continent, \emptyset) >) \\
& \circ\, CreateD(Temps\_1,\ ER_1.Temps) \\
& \circ\, AddA(Temps\_1,\ \{mois,\ lib\_mois\}) \\
& \circ\, AddH(Temps\_1,\ H\_Tps) \\
& \circ\, AddL(Temps\_1,\ H\_Tps,\ < (mois, \{lib\_mois\}) >) \\
& \circ\, CreateD(Temps\_2,\ ER_1.Temps) \\
& \circ\, AddA(Temps\_2,\ \{annee\}) \\
& \circ\, AddH(Temps\_2,\ H\_Tps) \\
& \circ\, AddL(Temps\_2,\ H\_Tps,\ < (annee, \emptyset) >) \\
& \circ\, CreateD(Ventes,\ ER_1.Ventes) \\
& \circ\, AddM(Ventes,\ \{Sum(quantite),\ Sum(montant)\}) \\
& \circ\, Connect(Ventes,\ \{Produit,\ Client\, Temps\_2\}) \\
& \circ\, CreateF(Achats,\ ER_1.Achats) \\
& \circ\, AddM(Achats,\ \{Sum(prix)\}) \\
& \circ\, Connect(Achats,\ \{Produit,\ Temps\_1\}) \\
& \circ\, Select(Temps.annee\ >=\ 2000)\}
\end{aligned}
$$

FIGURE 4.8 – Fonction Map du deuxième état réduit (Map^{ER_2})

schéma multidimensionnel de l'état ER_1 présenté dans l'exemple de la section 4.3.7 en schéma `ROLAP` dénormalisé. Ce résultat est obtenu suite à un ensemble de transformations qui vise à convertir les faits et les dimensions du schéma multidimensionnel en tables du niveau logique. Il en est de même pour les états ER_2 et ER_3.

Les expressions `ETL-OCL` relatives à l'état ER_1 et présentées dans la figure 4.10, sont traduites en expressions algébriques présentées par la figure 4.12, ceci en appliquant les règles figurant dans le chapitre précédent (cf. section 3.4.1.2).

La dernière étape vise à générer le modèle physique (PSM) de la plateforme `Oracle`. Le processus de réduction se fait de manière séquentielle : chaque état est construit à partir d'un état de référence. Les données d'un état sont extraites des données précédentes. Le schéma conceptuel est implanté au niveau physique en utilisant les vues matérialisées. Ceci permet de garantir la cohérence des données du schéma courant et des différents états réduits. Nous rappelons que le PSM est composé de vues matérialisées et de dimensions. La définition des vues matérialisées dépend des tables ROLAP et des expressions algébriques, alors

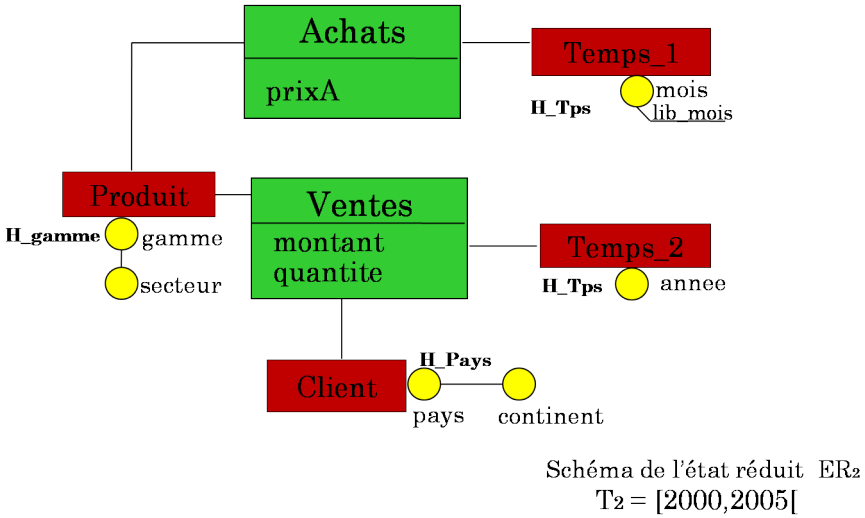


FIGURE 4.9 – Réduction de l'entrepôt analysant les ventes et les achats : schéma de l'état réduit ER_2

que la définition des dimensions dépend du schéma conceptuel. Par conséquent, comme dans le chapitre précédent, le modèle physique est généré par fusion des modèles conceptuel (schéma multidimensionnel) et logique (schéma ROLAP dénormalisé) en appliquant les règles de la section 3.4.2 du chapitre précédent. A partir de ce modèle, sont générés les codes de création et de chargement des vues matérialisées présentées dans la figure 4.14 ainsi que le code de création des dimensions (cf. figure 4.15). Il en est de même pour les états ER_2 et ER_3.

4.5 Bilan et positionnement

4.5.1 Contributions

Nous avons présenté au cours de ce chapitre la deuxième partie de nos contributions portant sur la modélisation et l'implantation d'un entrepôt de données multidimensionnelles réduit. Notre approche dirigée par les modèles présente deux principales parties :

1. L'élaboration du PIM multidimensionnel : le concepteur construit l'ensemble des schémas multidimensionnel des états pour répondre aux besoins des décideurs. Un état est construit à partir d'un état de référence en appliquant un ensemble d'opérateurs de réduction. Dans un état donné, la relation de correspondance d'un élément du schéma multidimensionnel à partir d'un élément de référence, est formalisée en `ETL-OCL`.

2. Transformation automatique : à partir du niveau PIM multidimensionnel dans lequel chaque état est décrit par un schéma et des un ensemble d'expressions `ETL-OCL`, est généré :

   (a) Le niveau PIM ROLAP : ce niveau permet de décrire l'ensemble des états de l'ED sous forme de tables ROLAP dénormalisées et des expressions algébriques permettant d'extraire un état à partir de son état de référence.

   (b) Le niveau PSM : le modèle physique `Oracle` est généré à partir du PIM ROLAP ; il décrit les différents états en termes de vues matérialisées et de dimensions est généré. Le PSM est par la suite transformé en code SQL de création et de chargement des vues matérialisées relatives aux différents états de l'entrepôt.

Les contributions de ce chapitre ont été présentées dans les publications [Atigui et al., 2012c] et [Atigui et al., 2012d].

4.5.2 Discussion

La réduction de données vise à conserver uniquement l'information utile pour la prise de décision en agrégeant des données anciennes.

Les travaux existants consistent à formaliser uniquement les mécanismes de réduction de données relationnelles [Iftikhar and Pedersen, 2010] ou multidimensionnelles [Skyt et al., 2008]. Les contributions menées pour réduire des données multidimensionnelles ([Skyt et al., 2008]) se limitent à synthétiser les données du fait de manière progressive (agrégation d'un niveau de dimension à un niveau supérieur). Ils ne fournissent aucun moyen pour supprimer des éléments multidimensionnels tels que les dimensions, les faits ou les attributs. D'autre part, ces travaux sont théoriques et la validation n'a pas été fournie. A notre connaissance, aucune démarche n'a été proposée pour formaliser et implanter de manière automatique un ED réduit au niveau physique.

Pour pallier ces limites, nous nous appuyons sur une démarche IDM. Nous proposons de décrire les données réduites au niveau multidimensionnel par un ensemble d'états. Nous avons défini un ensemble d'opérateurs qui permet non seulement d'agréger progressivement les données du fait, mais aussi de supprimer des éléments multidimensionnels dans le temps. Nous avons également présenté une démarche IDM qui formalise et implante de manière automatique ces modèles au niveau physique.

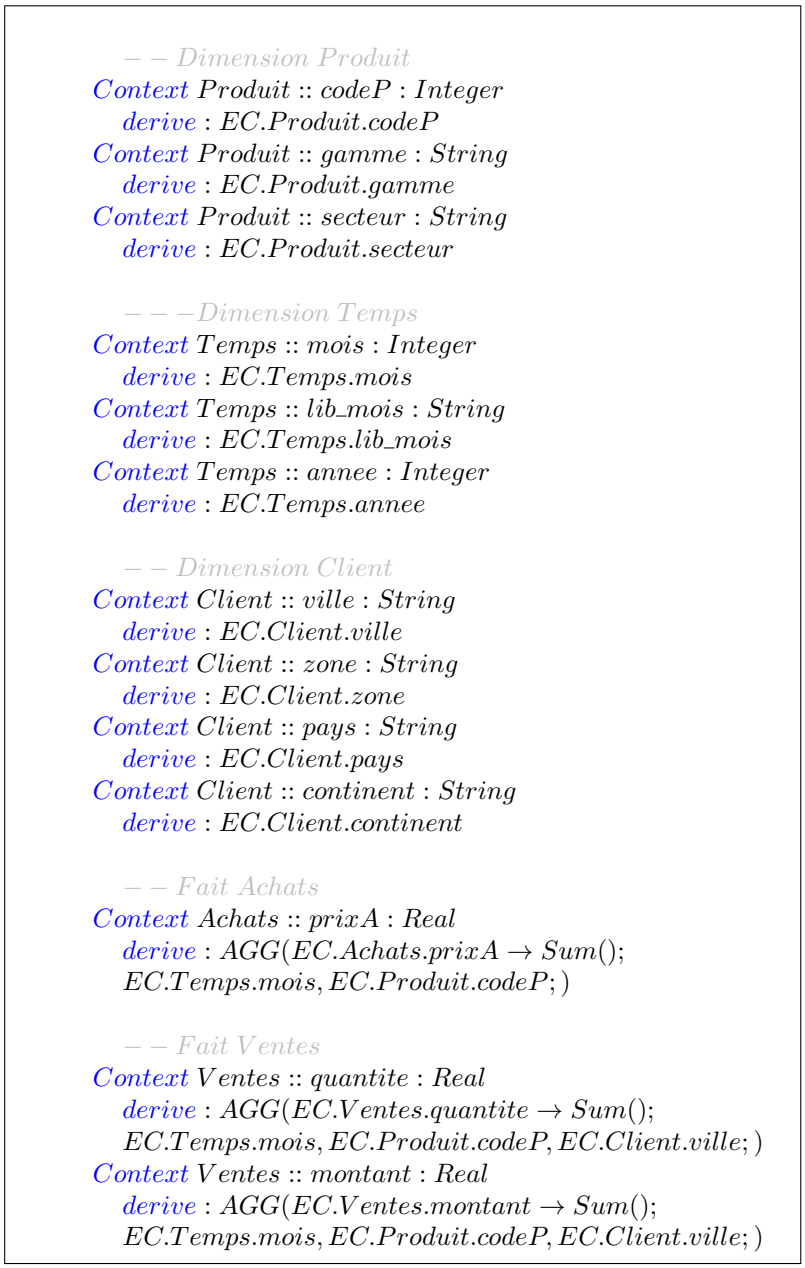


FIGURE 4.10 – Expressions ETL-OCL liées au premier état réduit (ER_1)

ER1.Produit (CodeP, Gamme, Secteur)
ER1.Client (Ville, Zone, Pays, Continent)
ER1.Temps (Mois, lib_mois, Annee)
ER1.Ventes (CodeP#, Ville#, Mois#, Montant, Quantite)
ER1.Achats (CodeP#, Mois#, PrixA)

FIGURE 4.11 – Tables du niveau PIM ROLAP relatives à l'état ER_1

$$
\begin{aligned}
ER_1.Produit &= \pi[codeP, gamme, secteur]\, EC.Produit \\
ER_1.Temps &= \pi[mois, lib\_mois, annee]\, EC.Temps \\
ER_1.Client &= \pi[ville, zone, pays, continent]\, EC.Client \\
ER_1.Achats &= \pi[codeP, mois, ville, prixA] \\
&\quad (Sum([EC.Achats \underset{codeP}{\bowtie} EC.Produit \\
&\quad \underset{date}{\bowtie} EC.Temps]; [codeP, mois,]; [prixA];)) \\
ER_1.Ventes &= \pi[codeP, mois, ville, quantite, montant] \\
&\quad (Sum([EC.Ventes \underset{codeP}{\bowtie} EC.Produit \\
&\quad \underset{date}{\bowtie} EC.Temps \underset{codeC}{\bowtie} EC.Client]; \\
&\quad [codeP, mois, ville]; [quantite, montant];))
\end{aligned}
$$

FIGURE 4.12 – Expressions algébriques liées au premier état réduit (ER_1)

```
Create Materialized View   ER1.Produit
Build Immediate
Refresh Complete
On Demand
As Select                  codeP, gamme, secteur
From                       EC.Produit ;

Create Materialized View   ER1.Temps
Build Immediate
Refresh Complete
On Demand
As Select                  mois, lib_mois, annee
From                       EC.Temps
Where                      EC.Temps.annee >= 2005 And
                           EC.Temps.annee < 2007 ;
```

FIGURE 4.13 – Script SQL de création et de chargement de l'état réduit ER_1 (1/3)

```
Create Materialized View   ER1.Client
Build Immediate
Refresh Complete
On Demand
As Select                  ville, zone, pays, continent
From                       EC.Client;

Create Materialized View   ER1.Achats
Build Immediate
Refresh Complete
On Demand
As Select                  codeP, mois, Sum(prixA) As prixA
From                       EC.Achats, EC.Produit, EC.Temps
Where                      EC.Achats.codeP = EC.Produit.codeP And
                           EC.Achats.date = EC.Temps.date And
                           EC.Temps.annee >= 2005 And
                           EC.Temps.annee < 2007
GroupBy                    codeP, mois;

Create Materialized View   ER1.Ventes
Build Immediate
Refresh Complete
On Demand
As Select                  codeP, mois, ville, Sum(quantite) As quantite,
                           Sum(montant) As montant
From                       EC.Ventes, EC.Produit, EC.Client, EC.Temps
Where                      EC.Ventes.codeP = EC.Produit.codeP And
                           EC.Ventes.codeC = EC.Client.codeC And
                           EC.Ventes.date = EC.Temps.date And
                           EC.Temps.annee >= 2005 And
                           EC.Temps.annee < 2007
Group By                   codeP, mois, ville;
```

FIGURE 4.14 – Script SQL de création et de chargement de l'état réduit ER_1 (2/3)

Create Dimension	$ER1.Produit\_Dim$
Level $codeP$	*Is* $(ER1.Produit.codeP)$
Level $gamme$	*Is* $(ER1.Produit.gamme)$
Level $secteur$	*Is* $(ER1.Produit.secteur)$
Hierarchy H_Gamme	$(codeP$ *Child Of* $gamme$ *Child Of* $secteur)$;
Create Dimension	$ER1.Temps\_Dim$
Level $mois$	*Is* $(ER1.Temps.mois)$
Level $annee$	*Is* $(ER1.Temps.annee)$
Hierarchy H_Tps	$(mois$ *Child Of* $annee)$
Attribute $mois$	*Determines* (lib_mois) ;
Create Dimension	$ER1.Client\_Dim$
Level $ville$	*Is* $(ER1.Client.ville)$
Level $pays$	*Is* $(ER1.Client.pays)$
Level $zone$	*Is* $(ER1.Client.zone)$
Level $continent$	*Is* $(ER1.Client.continent)$
Hierarchy H_Pays	$(ville$ *Child Of* $pays$ *Child Of* $continent)$
Hierarchy H_Zone	$(ville$ *Child Of* $zone$ *Child Of* $continent)$;

FIGURE 4.15 – Script SQL de création et de chargement de l'état réduit (ER_1) (3/3)

Chapitre 5

Implantation et validation

5.1 Introduction

Ce chapitre a pour objectif de présenter le prototype que nous avons développé afin d'expérimenter nos contributions présentées dans les chapitres 3 et 4. Ce prototype baptisé `DWAT` (`Data Warehouse Automatic Transformation`) fournit l'ensemble des mécanismes permettant d'élaborer un entrepôt de données réduit. Ce chapitre est organisé comme suit. En section 5.2, nous présentons les techniques utilisées pour élaborer notre système. La section 5.3 décrit notre système. Les sections 5.4 et 5.5 présentent des études expérimentales qui portent respectivement sur la transformation de schémas multidimensionnels et la réduction d'ED.

5.2 Environnement et choix techniques

Dans cette section, nous présentons l'ensemble des techniques utilisées afin de mettre en œuvre le prototype `DWAT`.

Etant donné que notre démarche d'élaboration d'ED est dirigée par les modèles, nous avons utilisé un cadre technique qui permet de mettre en œuvre les différents aspects liés à l'IDM, à savoir : les modèles, les métamodèles et les transformations. Dans le paragraphe suivant, nous explicitons nos besoins techniques. Ensuite, nous décrivons les différents outils que nous avons utilisés afin de développer le système `DWAT`.

5.2.1 Besoins

Nous avons besoin de langages et d'outils qui fournissent :

1. Un environnement convenable pour créer les métamodèles et les instancier de manière à garantir la relation de conformité d'un modèle avec son métamodèle.
2. Les mécanismes pour la transformation (endogène et exogène) d'un modèle source en un modèle cible, conformes respectivement à un métamodèle source et cible.
3. Les mécanismes pour la fusion de modèles où un ou plusieurs modèles sources sont transformés en un ou plusieurs modèles cibles.
4. Les moyens pour la transformation M2T (Model to Text) d'un modèle source (du niveau PSM) en un langage cible.
5. Un environnement permettant d'intégrer l'ensemble des techniques liées à la modélisation dirigée par les modèles, et qui permet également de tenir compte des standards souvent utilisés tel que `UML` et `OCL` que nous avons utilisés pour décrire les sources de données et les expressions de transformation des attributs sources.

5.2.2 Outils pour la mise en œuvre de notre système

Il existe plusieurs logiciels pour implanter une approche IDM [Diaw et al., 2010]. Cependant, certains outils se contentent de fournir un environnement restreint qui permet de répondre uniquement à un aspect particulier de modélisation ou de transformation. Notre objectif est de trouver un cadre technique convenable pour la modélisation, la métamodélisation ainsi que la transformation, y compris de modèles vers modèles et de modèle vers texte. Nous avons eu recours à la plateforme `Eclipse Modeling Framework`[1] (`EMF`) qui présente un environnement complet pour la mise en œuvre de notre démarche IDM. La figure 5.1 montre les langages et outils utilisés afin d'implanter notre démarche présentées dans la figure 3.1 du chapitre 3. Dans cette section, nous présentons les langages et les outils IDM qui ont permis la mise en œuvre de notre prototype.

5.2.2.1 Eclipse Modeling Framework (EMF)

Avant de présenter les fonctionnalités EMF utilisées dans le cadre de cette thèse, nous introduisons d'abord Eclipse. Eclipse[2] est un système logiciel sous licence « `Open Source` » initié par IBM[3] et dont l'objectif est de fournir un environnement de développement logiciel intégré libre et extensible. Cet environnement permet de fournir et de produire des outils pour la réalisation d'une application y compris les phases de modélisation, de programmation et de test. L'architecture d'`Eclipse` est basée sur la notion de « `Plugin` ». Chaque plugin fournit un ensemble particulier de fonctionnalités.

1. http ://www.eclipse.org/modeling/emf/
2. http ://www.eclipse.org/
3. http ://www.ibm.com

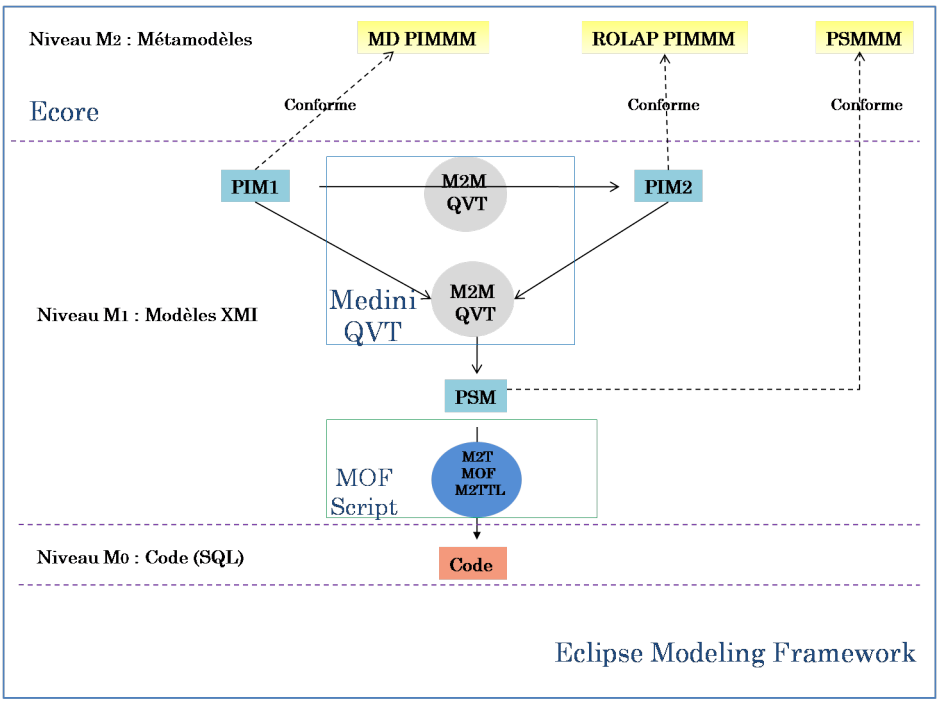


FIGURE 5.1 – Outils pour la mise en œuvre de notre système

`Eclipse Modeling Framework` (`EMF`) présente un cadre Eclipse permettant l'élaboration d'applications basée sur les modèles. Il présente un environnement technique assez complet couramment utilisé dans le développement dirigé par les modèles. EMF repose sur trois technologies : `Java`, `XML` et `UML`. Il permet ainsi de décrire un modèle sous forme d'un diagramme `UML` ou d'un schéma `XML` ou aussi en utilisant le langage `Java`. Il est possible d'utiliser l'une de ces représentations et de générer les deux autres.

5.2.2.2 Mise en œuvre des métamodèles (`Ecore`)

`EMF` fournit un environnement qui permet de créer et de valider les différents métamodèles. `EMF` utilise `Ecore` afin d'élaborer et manipuler les modèles.`Ecore` est le modèle auto-descriptif utilisé pour décrire et manipuler des modèles dans `EMF`. Il peut être considéré comme une implantation simplifiée du méta-métamodèle d'UML :`Meta-Object Facility` (`MOF`) [OMG, 2011a]. Nous pouvons trouver des similitudes dans leurs capacités à spécifier des classes, des caractéristiques structurelles et comportementales, l'héritage et les paquetages. Cependant, leurs différences résident dans les structures de type de données, les relations inter-paquetage et d'autres aspects complexes liées aux associations. `Ecore` est plus

proche du métamodèle Essentiel Meta-Object Facility (EMOF) [Budinsky et al., 2003].

La figure 5.2 montre un extrait simplifié du métamodèle Ecore. Nous présentons ici les éléments du métamodèle Ecore que nous avons utilisés pour élaborer les différents métamodèles de notre approche. Il s'agit principalement des éléments suivants :

- EPackage : est l'élément racine d'un modèle Ecore. Il est composé de Eclasse et de EDataType. Chaque paquetage est identifié par un nom, un URI (nsURI) et le préfixe du namespace XML (nsPrefix).
- EClass : cet élément représente une classe Ecore et permet de décrire la structure d'un objet à travers un ensemble d'attributs (EAttributes) et son comportement via un ensemble d'opérations. L'association réflexive (eSuperType) montre un lien de généralisation définissant la notion d'héritage. Une EClass hérite de toutes les caractéristiques structurelles et comportementales de sa EClass mère. Elle ne peut pas hériter de plusieurs EClass (l'héritage multiple n'est pas possible).
- EDataType : représente le type d'un attribut qui peut être prédéfini tel que les types primitifs : Integer, Boolean, String, etc. ou défini par l'utilisateur en utilisant une énumération EEnum.
- EEnum : une énumération permet de spécifier un ensemble de valeurs possibles pour un attribut donné.
- EReference : permet de spécifier un lien (une association en terme UML) entre deux instances du métamodèle. La composition au sens UML est présentée comme une propriété Containment d'une EReference.

5.2.2.3 Mise en œuvre des modèles : XMI

L'OMG propose la norme XML Metadata Interchange (XMI) [OMG, 2011c] pour représenter des modèles sous forme de documents XML. Cette représentation se fait par des mécanismes de sérialisation et de génération. XMI permet l'échange de métadonnées entre les outils de modélisation UML ou MOF basés dans des environnements distribués et hétérogènes. XMI est couramment utilisé dans le cadre de l'ingénierie dirigée par les modèles et fait partie intégrante de l'environnement EMF. Dans le cadre de cette thèse, XMI présente un moyen pour créer des modèles à partir de métamodèles Ecore. Ces modèles sont par la suite transformés pour obtenir du code.

5.2.2.4 Mise en œuvre des transformations

Il existe de nombreux langages de transformation de modèles [Ehrig et al., 2005]. Nous pouvons citer par exemple les langages QVT [OMG, 2011b] et ATL [Jouault and Kurtev, 2006], [ATL Development Team, 2013] pour les transformations M2M, et le langage MOF2T pour les transformations de types modèle vers du texte.

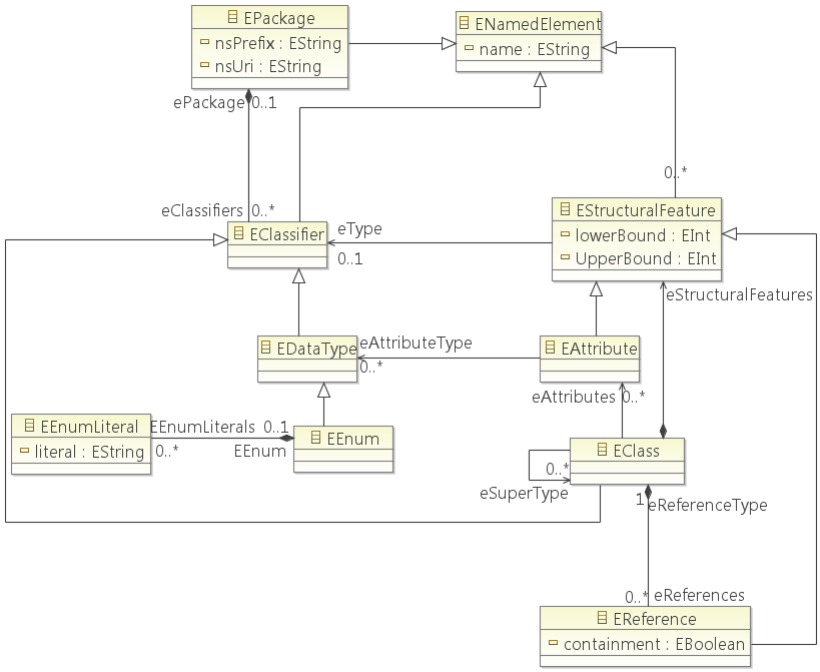


FIGURE 5.2 – Extrait du métamodèle Ecore [Budinsky et al., 2003]

Durant ces dernières années, de nombreux outils ont été créés pour mettre en œuvre ces langages. La progression de ces outils en termes de nombre et de qualité (d'une version à l'autre) a été remarquable durant les trois dernières années. Un choix entre ces différents langages peut être basé sur des études comparatives [Diaw et al., 2010], [Combemale, 2008], [Czarnecki and Helsen, 2006] qui montrent les différents aspects de ces outils, voire sur un test comparatif pour évaluer les fonctionnalités fournies par chaque système afin de les comparer et en tirer le plus adapté à notre problématique.

Notre choix du langage a été fondé sur des critères spécifiques à notre démarche. Le langage (par conséquent l'outil) utilisé doit permettre la transformation exogène ainsi que la fusion de modèles. L'outil doit être intégré dans l'environnement EMF pour qu'il soit utilisé aisément avec les outils de modélisation et de métamodélisation. Ainsi, nous avons utilisé le langage QVT [OMG, 2011b] pour les transformations `M2M` (du PIM multidimensionnel vers le PIM ROALP et des PIM vers le PSM) et le langage `MOF Model To Text Transformation Language` (`MOFM2Text`) [OMG, 2008] pour les transformations `M2T` (du PSM vers le code SQL). Dans la suite, nous présentons ces deux langages ainsi que les outils qui leurs sont associés.

Transformation de modèles vers modèles

QVT est le langage standardisé par l'OMG pour accomplir la transformation de modèles dans une démarche MDA. Le langage QVT présente un aspect hybride qui combine des langages déclaratifs (Relations et Core) et impératif (Operational QVT). La norme QVT a été proposée afin d'atteindre les objectifs suivants [Diaw et al., 2010] :

- Exprimer des requêtes (Query) afin de sélectionner des éléments d'un modèle,
- Créer des vues (Views) qui représente des parties de modèles pour en montrer des aspects spécifiques,
- Formaliser les expressions de correspondance (Transformations) entre modèles définis avec MOF.

Il existe des outils qui permettent de mettre en œuvre le langage QVT. Parmi eux, nous pouvons citer SmartQVT [Alizon et al., 2007] qui implante l'aspect opérationnel et MediniQVT [4] qui implante l'aspect relationnel. Nous avons choisi d'utiliser l'outil MediniQVT afin d'implanter les règles de transformation de type M2M, à savoir la transformation du PIM multidimensionnel en PIM ROLAP ainsi que la fusion de ces deux PIM en PSM.

MediniQVT utilise le langage QVT-Relations. Cet outil est fourni sous forme de plugin Eclipse basé sur le framework de métamodélisation EMF. Il est conçu pour implanter les transformations de modèles vers modèles. Cet outil inclut un éditeur pour exprimer les règles de transformations dans une syntaxe textuelle.

Transformation de modèles vers texte

Afin d'aboutir au code, un ensemble de transformations est appliqués au PIM pour générer des modèles de plateforme (PSM). Ces PSM sont par la suite transformés en code. Le langage « MOF Model to Text Transformation Language » (MOF2Text) est la norme proposée par l'OMG pour la transformation des modèles vers une représentation textuelle [OMG, 2008]. Ce langage utilise une approche basée sur les templates dans laquelle le texte généré à partir de modèles est spécifié comme un ensemble de templates de texte paramétrés avec des éléments du modèle. Un template spécifie pour chaque élément du modèle source, les éléments qui leurs correspondent au niveau du texte (code). Les règles de transformation sont spécifiées sur les entités des métamodèles sélectionnées par des requêtes qui permettent de sélectionner et d'extraire des valeurs à partir des modèles. Ces valeurs sont par la suite transformés en fragment de code en utilisant la bibliothèque de définition du langage cible. Les templates peuvent être composés pour répondre aux transformations complexes. Les grandes transformations peuvent être structurées en modules.

Pour la mise œuvre de ce langage, nous pouvons citer par exemple : Acceleo [5] et

4. http ://projects.ikv.de/qvt
5. http ://www.eclipse.org/acceleo/

MOFScript [6]. Nous avons choisi d'utiliser MOF Script. Ce dernier est un plugin Eclipse et fournit un environnement pour la compilation et l'exécution du code MOF2Text.

Sur le plan théorique, les choix du langage QVT pour les transformations de modèles vers modèles et du langage MOF2Text pour les transformations de modèles vers du code se justifient comme suit. Ces deux langages étant standardisés par l'OMG à l'instar d'OCL et d'UML, ceci permet d'éviter les problèmes d'interopérabilité. Sur le plan pratique, l'utilisation d'EMF et des plugins présentés précédemment (Medini-QVT et MOF Script) permet d'éviter les problèmes d'interopérabilité entre les différents modules de notre prototype. Si l'utilisation du XMI favorise la communication entre les différents modules (les entrées et les sorties échangés par les différents modules), EMF intègre tous ces outils et fournit ainsi un environnement adapté pour mettre en œuvre notre approche. La section suivante présente l'architecture de notre système.

5.3 Elaboration d'un outil pour la modélisation et l'implantation d'ED réduits

5.3.1 Architecture générale

La vocation principale de notre système est d'implanter un ED. Ce système fournit également les moyens pour réduire un ED. Ce système regroupe trois principaux composants : l'interface utilisateur, un espace de stockage et un module de réduction et de transformation. L'entrée présente un schéma multidimensionnel à réduire et/ou à implanter au niveau physique. L'outil permet à l'utilisateur de créer ce schéma et de formaliser les expressions de transformation des attributs sources. Ensuite, le système permet d'implanter ce schéma en fournissant le script de création et de chargement des différentes structures, ou bien de réduire le schéma en créant un ensemble d'états et par la suite de les implanter. La figure 5.3 montre l'architecture générale de notre système. Dans la suite, nous détaillons ses différentes composantes.

5.3.1.1 Entrée du système

L'utilisateur se connecte au système DWAT pour créer et charger un ED et, le cas échéant, pour le réduire. Quel que soit l'objectif final de l'utilisateur, l'entrée du système est un schéma multidimensionnel. Ce dernier donne les structures de l'ED ainsi que les expressions de transformation des différents attributs. L'utilisateur peut instancier aisément le métamodèle en entrée du niveau PIM multidimensionnel (cf. figure 4.4). Evidemment, la (ou les) source(s) de données à

6. http ://marketplace.eclipse.org/content/mofscript-model-transformation-tool#.Ubx9fdUHdc

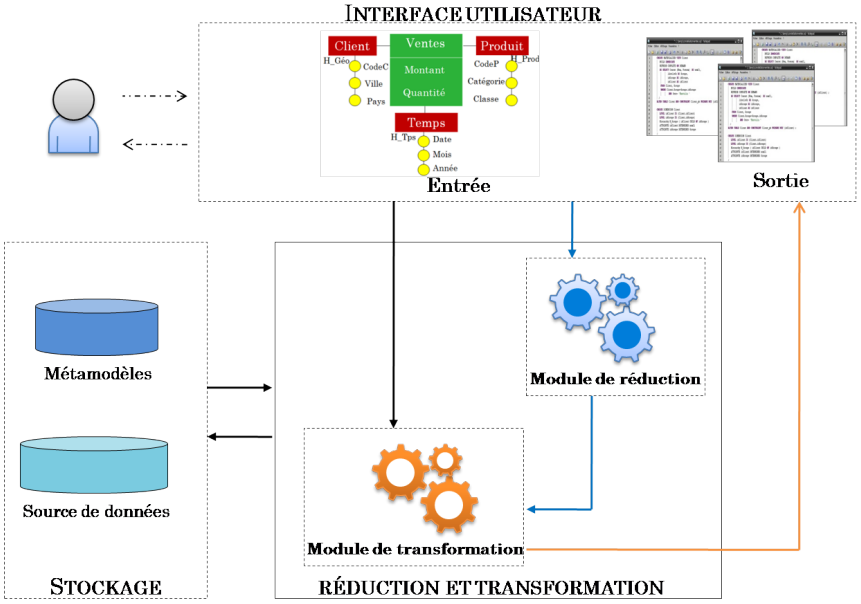


FIGURE 5.3 – Architecture générale du système

partir desquelles l'entrepôt est alimenté doivent être paramétrée(s) pour établir les liens source/entrepôt.

5.3.1.2 Sortie du système

Le résultat final est un script `SQL` qui permet de créer et d'alimenter des structures multidimensionnelles à partir d'un ensemble de structures sources. Si l'objectif était de construire et d'alimenter un ED à partir d'une BD source, le script permet de créer les tables de faits et de dimensions et de les alimenter à partir des tables correspondantes au niveau des sources. Lorsqu'il s'agit de la réduction de l'entrepôt, le script résultant permet de créer les différents états et leurs chargements à partir des états de référence.

5.3.1.3 Module de réduction

Ce module permet d'implanter l'ensemble d'opérateurs de réduction et de vérifier l'ensemble des contraintes définies pour garantir la bonne formation d'un schéma réduit. A partir du schéma de l'ED en entrée, l'utilisateur crée un ensemble de schémas d'états réduits. Ces schémas sont par la suite transformés en code SQL grâce au module de transformation.

5.3.1.4 Module de transformation

A partir d'un schéma multidimensionnel entré par le concepteur, le module de transformation MDA permet de restituer le code `SQL` de création et d'alimentation de l'entrepôt de données. Pour ce faire l'ensemble des règles de transformation de modèles vers modèles (`QVT`) et de modèles vers du texte (`MOF M2text`) sont implantées au préalable. Ces transformations font appel à un ensemble de métamodèles stockés en amont sous forme de fichiers `Ecore`. Les étapes suivantes présentent l'ordre de création des différents composants du système :

- **Etape 1 :** création des métamodèles (décrivant les structures et les opérations) sous forme de fichiers Ecore :
  - Métamodèle conceptuel des structures multidimensionnelles et des expressions `ETLOCL`.
  - Métamodèle logique des structures relationnelles et des expressions algébriques.
  - Métamodèle physique des structures physiques `Oracle` (vues matérialisées) et des requêtes `SQL`.
- **Etape 2 :** création des règles de transformation de modèles vers modèles (syntaxe textuelle du langage `QVT`) :
  - Règles de transformation du modèle conceptuel en modèle logique : des structures multidimensionnelles et des expressions `ETL-OCL` vers les structures relationnelles et les expressions algébriques.
  - Règles de fusion des modèles conceptuel et logique en modèle physique : (1) des structures relationnelles et des expressions algébriques vers les structures Oracle, (2) des dimensions et des hiérarchies conceptuelles vers les dimensions Oracle.
- **Etape 3 :** Création des règles de transformation de modèles vers texte (MOF M2T) : du modèle physique Oracle vers le code SQL. Le processus de transformation d'un modèle en texte prend en entrée le métamodèle et fournit en sortie le ou les fichier(s) du code SQL. Dans ce contexte, notre prototype prend en entrée le métamodèle qui décrit la plateforme `Oracle` et fournit en sortie le code SQL de création et d'alimentation des structures de l'ED.

5.3.2 Implantation

Cette section présente la mise en œuvre de notre système (cf. figure 5.3). Plus précisément, nous décrivons l'implantation du module de transformation. Ce module utilise l'ensemble des trois métamodèles présentés dans les chapitres précédents et présentés par les figures 4.4, 3.11 et 3.16. La figure 5.4 montre un extrait de ces trois métamodèles implantés en Ecore : PIM multidimensionnel (a), PIM ROLAP (b) et PSM (c). Ces métamodèles sont utilisés lors des transformations QVT et MOF2Text que nous détaillons dans les sous-sections suivantes.

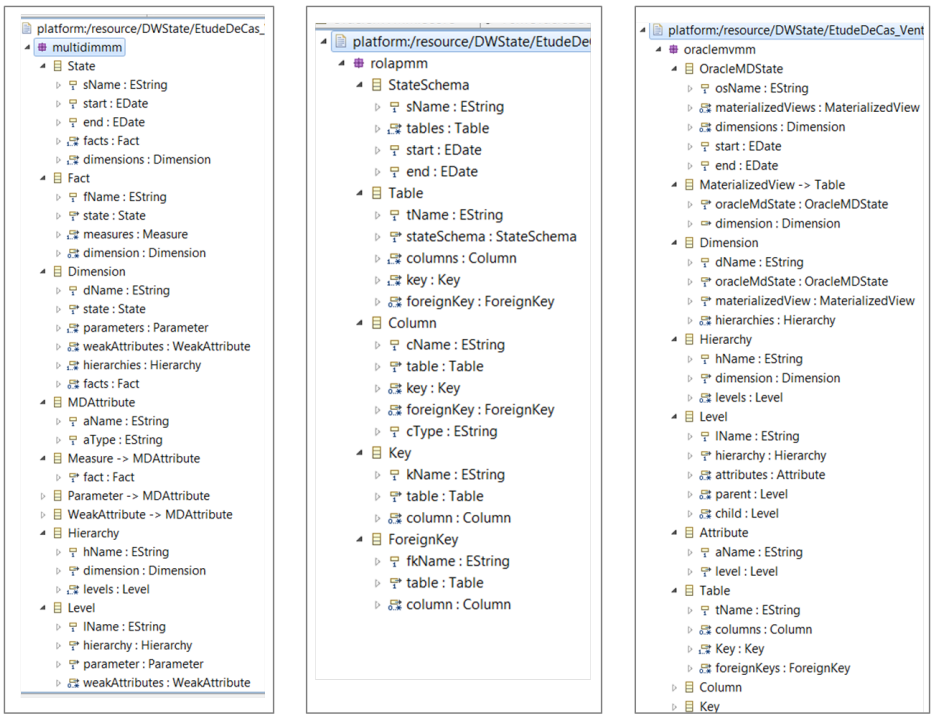


FIGURE 5.4 – Métamodèles de notre approche implantés en Ecore

5.3.2.1 Transformations QVT

Dans cette section, nous présentons les règles de transformation QVT qui permettent de transformer le PIM multidimensionnel en PIM ROLAP, et par la suite de fusionner ces deux PIM en un PSM Oracle. Pour expliquer ces règles, nous utilisons le formalisme graphique du langage QVT. Nous montrons également un extrait du code implanté en utilisant la syntaxe textuelle du langage QVT.

Une transformation `QVT` entre deux modèles candidats est spécifiée grâce à un ensemble de relations. Chaque relation est composée des éléments suivants :

- `Domains` : chaque domaine désigne un modèle candidat et un ensemble d'éléments à relier.
- `Relation Domain` : permet de spécifier le type de relation entre les domaines, elle peut être marquée comme `Checkonly` (C) ou `Enforced` (E). Un domaine `Chekonly` permet de vérifier s'il existe une correspondance valide qui satisfait la relation ; alors qu'un domaine `Enforced` permet de créer un élément dans le modèle si le lien de correspondance n'est pas vérifié. Pour chaque domaine

le nom de son métamodèle sous-jacent doit être spécifié.
- La clause `When` : décrit les pré-conditions qui doivent être remplies pour réaliser la transformation.
- La clause `Where` : détermine les post-conditions qui doivent être remplies par tous les éléments du modèle participant à la relation.

Transformation du PIM multidimensionnel en PIM ROLAP

Le PIM multidimensionnel est traduit en PIM ROLAP en appliquant un ensemble de règles de transformation. La figure 5.5 montre les relations QVT, les liens décrivent qu'une relation telle que la relation « `DimensionToTable` » implique une autre relation de transformation telle que « `ParameterToColumn` ». Dans la suite, nous détaillons certaines de ces relations.

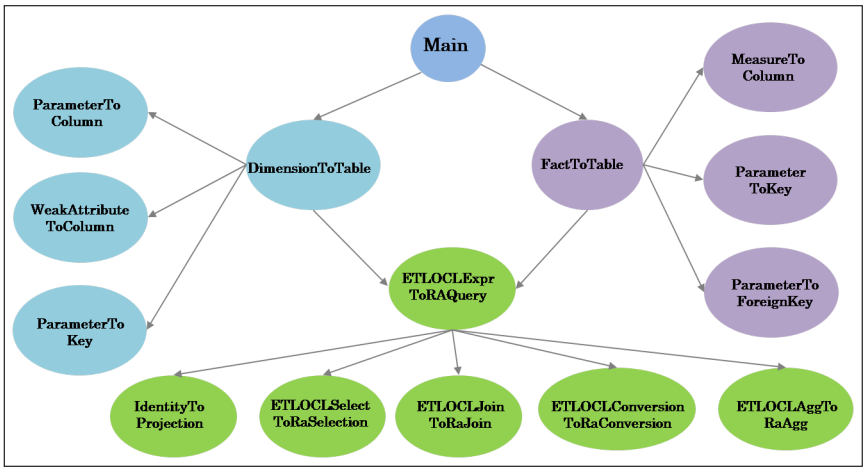


FIGURE 5.5 – Ensemble des règles de transformation du PIM multidimensionnel en PIM ROLAP

Relation « Main ». Cette relation est le point d'entrée au processus de transformation. La partie gauche de la figure 5.6 montre les éléments du modèle source `(md : MD)` transformés en éléments du modèle ROLAP dénormalisé `(drolap : ROLAP)` présenté par la partie droite de la figure. Un état multidimensionnel est transformé en un schéma ROLAP de même nom et ayant la même durée de validité. Chaque fait est transformé en une table via la relation « `FactToTable` ». Chaque dimension est traduite en une table via la relation « `DimensionToTable` » spécifiée au niveau de la clause « `Where` ».

Relation « DimensionToTable ». La figure 5.7 montre qu'une dimension est transformée en une table dont elle acquiert le nom. Tous les attributs (les paramètres et les attributs faibles) de la (ou des) hiérarchie(s) composant cette dimension, sont transformés en colonnes de la table en appliquant respectivement les règles « `ParameterToColumn` » et « `WeakAttributeToColumn` ». Le paramètre racine détermine la clé primaire de cette table via la relation « `Pa-`

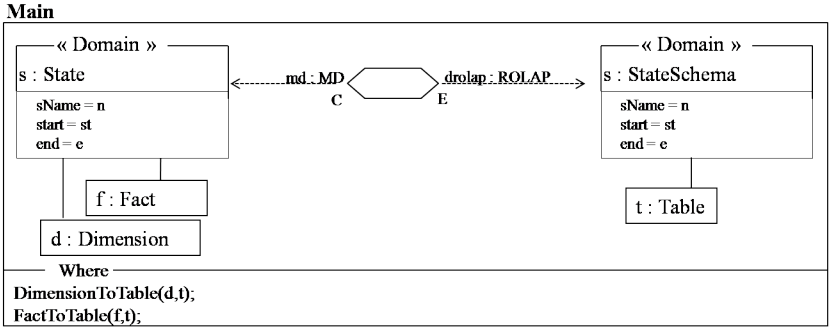


FIGURE 5.6 – Relation `Main` de la transformation du PIM multidimensionnel en PIM ROLAP

`rameterToKey` ». Enfin, les expressions ETL-OCL définissant les expressions de transformation des attributs de la dimension sont transformés en requêtes algébriques via la relation « `ETLOCLExprToRAQuery` » .

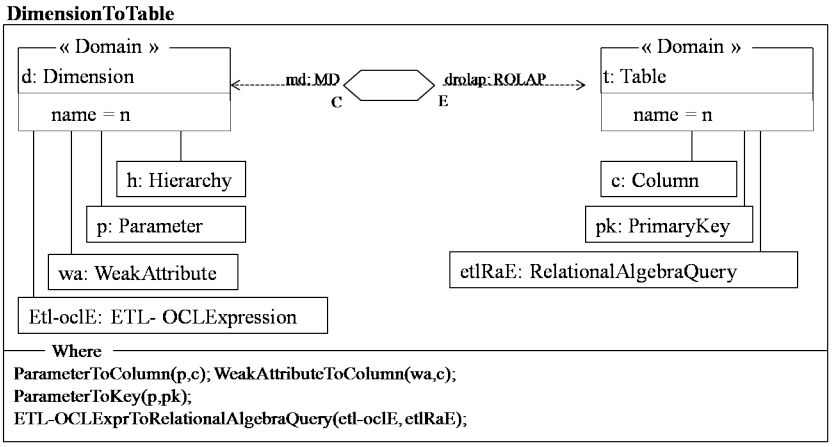


FIGURE 5.7 – Relation de transformation de dimensions en tables

Relation « FactTotable ». Une fois que les dimensions liées au fait ont été transformées (la pré-condition « `DimensionToTable` » de la clause « `When` »), le fait est converti en une table ayant le même nom (cf. figure 5.8). Les mesures sont transformées en colonnes au moyen de la relation « `MeasureToColumn` » de la clause « `Where` ». Cette clause implique aussi la transformation des expressions ETL-OCL relatives aux mesures en requêtes algébriques.

Relation « ETLOCLExprToRAQuery ». Cette relation est décrite par la figure 5.9 Toute expression ETL-OCL est transformée en une requête algébrique. La clause « `When` » précise que les dimensions et les faits du schéma conceptuel

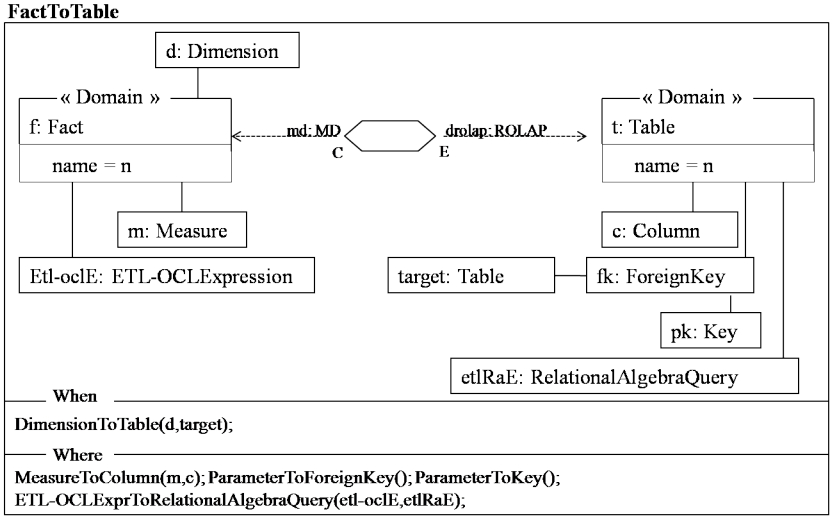


FIGURE 5.8 – Relation de transformation de faits en tables

doivent être déjà converties en tables. La clause « `Where` » identifie les relations de correspondances entre les différentes opérations ETL-OCL et algébriques.

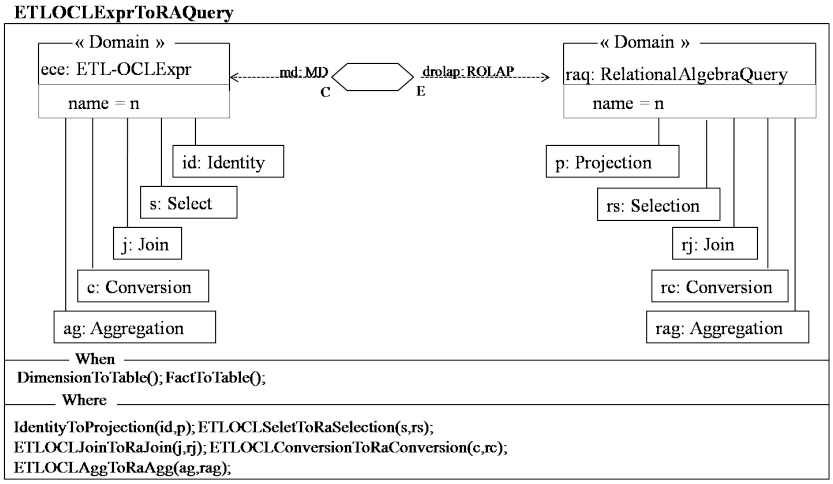


FIGURE 5.9 – Relation de transformation d'expressions ETL-OCL en requêtes algébriques

Relation « ETLOCLAggToRAAgg ». Cette relation (cf. figure 5.10) permet la transformation de l'opération d'agrégation définie sur les attributs appartenant à des classes sources en une opération d'agrégation définie sur les attributs des tables du modèle ROLAP. Tous les paramètres de cette opération

sont convertis en leurs équivalents du niveau ROLAP. La clause « `When` » précise que les faits et les dimensions doivent être convertis en tables au préalable.

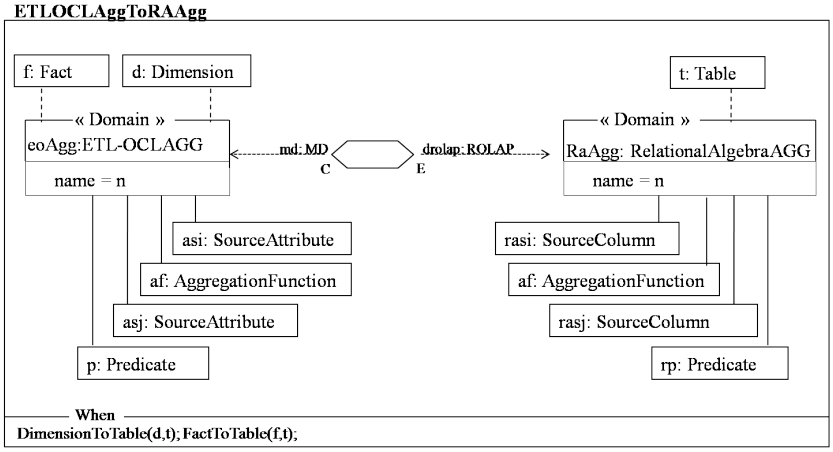


FIGURE 5.10 – Relation de transformation d'agrégations ETL-OCL en agrégations algébriques

Relation « ETLOCLSelectionToRASelection ». Comme le montre la figure 5.11, toute opération de sélection définie en ETL-OCL en utilisant les opérateurs « `Select` », « `Reject` » est transformée en une opération de sélection algébrique. L'attribut sur lequel porte la sélection indique la colonne sur laquelle est définie la sélection algébrique. L'opérande et le critère de sélection définissent respectivement l'opérande et le critère de sélection de la requête algébrique.

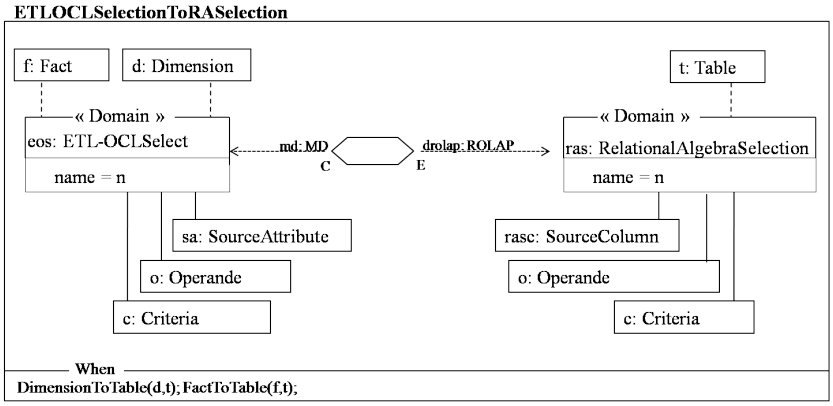


FIGURE 5.11 – Relation de transformation de sélections ETL-OCL en sélections algébriques

Fusion des PIM en PSM

Le PIM multidimensionnel et le PIM ROLAP sont fusionnés pou générer le PSM. La fusion de ces deux modèles se fait en appliquant un ensemble de relations comme présenté par la figure 5.12. Les liens décrivent qu'une relation implique une autre relation. Par exemple la transformation d'une table en une vue matérialisée (« `TableToMaterializedView` ») implique la transformation de toutes les colonnes de la tables en colonnes de la vue matérialisée (« `TableColumnToMVColun` »). Dans ce qui suit, nous détaillons certaines de ces relations.

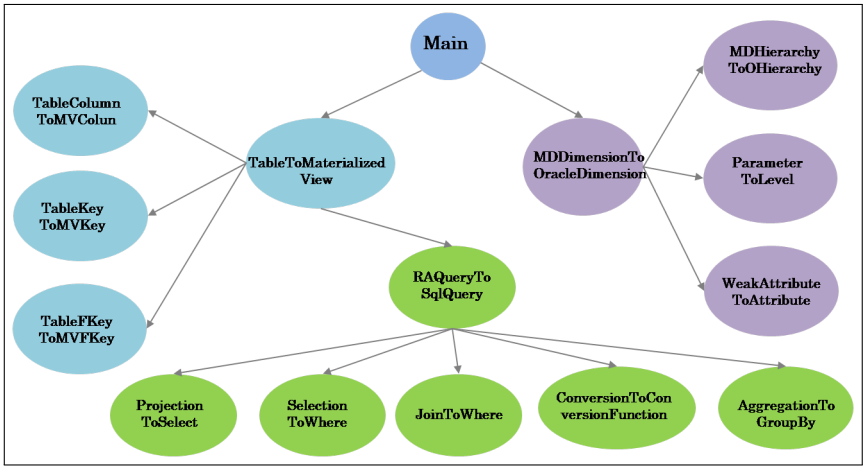


FIGURE 5.12 – Ensemble des règles de fusion des PIM en PSM

Relation « Main ». Cette relation permet de fusionner les deux PIM pour fournir le PSM Oracle. La partie gauche de la figure 5.13 montre deux domaines différents, le premier décrit le PIM ROLAP (« `drolap : ROLAP` »), le second montre le PIM multidimensionnel (« `md : MD` »). La partie droite de la figure explicite le PSM en sortie composé de vues matérialisées et de dimensions. La clause « `Where` » indiquent que les tables ROLAP sont transformées en vues matérialisées et que les dimensions du PIM multidimensionnel sont transformées en dimensions Oracle.

Relation « TableToMaterializedView ». La figure 5.14 montre que chaque table du modèle ROLAP est transformée en une vue matérialisée ayant le même nom. La clause « `Where` » identifie les relations de transformation des colonnes, des clés primaires et des expressions algébriques.

Relation « MDDimensionToOracleDimension ». Cette relation (cf. figure 5.15) permet de transformer les dimensions du PIM multidimensionnel en dimensions Oracle ayant le même nom préfixé par « `Dim` ». La clause « `When` » précise que les tables appropriées doivent être transformées en vues matérialisées. La clause « `Where` » montre que les hiérarchies, les paramètres et les attributs faibles sont respectivement convertis en hiérarchies, niveaux et attributs Oracle.

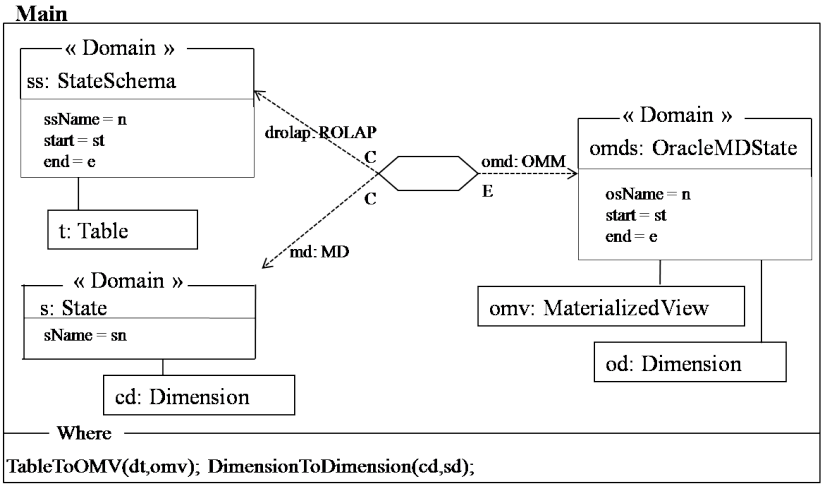


FIGURE 5.13 – Relation `Main` de la fusion des PIM en PSM

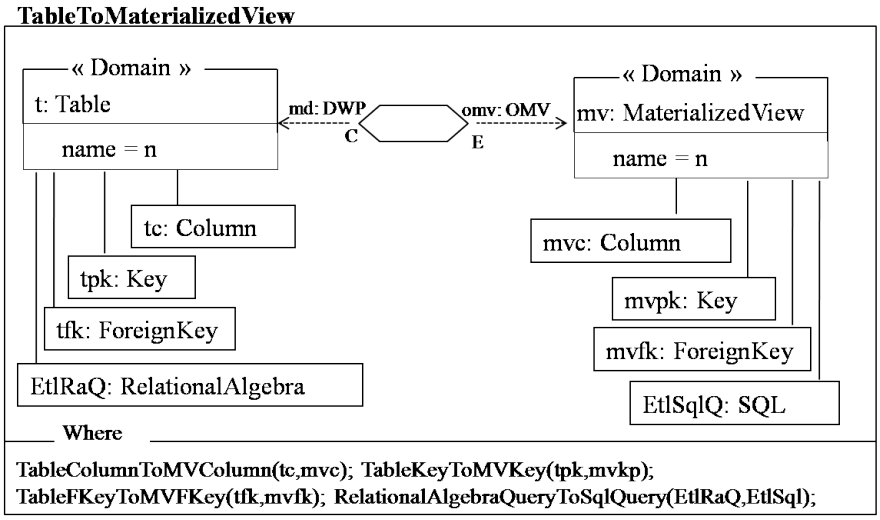


FIGURE 5.14 – Relation de transformation de tables en vues matérialisées

Relation « RelationalAlgebraQueryToSQLQuery ». Cette relation (cf. figure 5.16) montre la transformation des requêtes algébriques en requêtes SQL permettant la définition de la vue matérialisée appropriée. La clause « `When` » indique que les tables de dimension et de fait sont converties au préalable en vue matérialisées. La clause « `Where` » présente les différentes relations de correspondance entre les éléments d'une requête algébrique et les éléments d'une requête SQL.

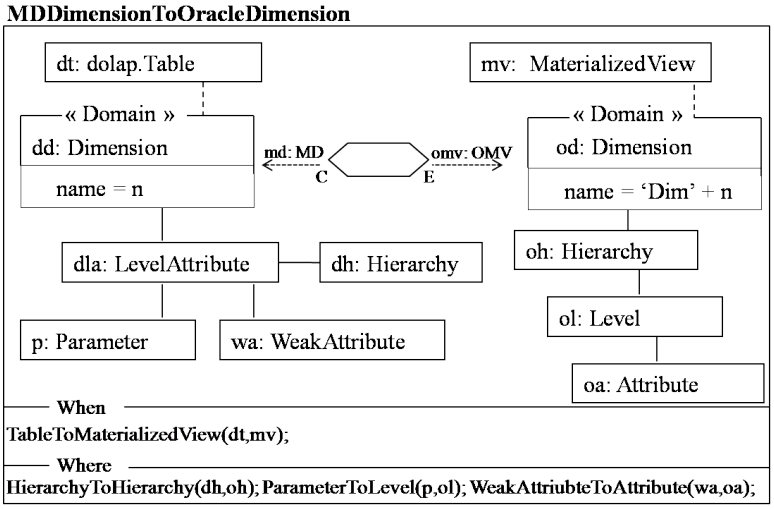


FIGURE 5.15 – Relation de transformation de dimension du PIM multidimensionnel en dimension Oracle

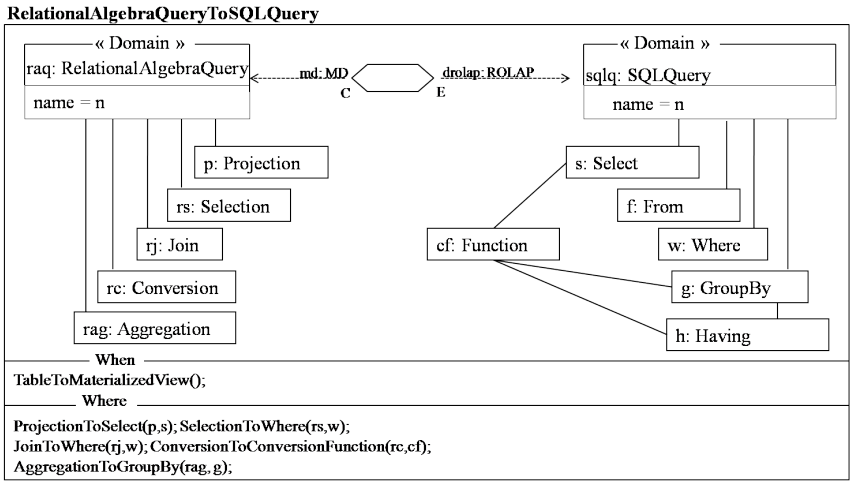


FIGURE 5.16 – Relation de transformation de requêtes algébriques en requêtes SQL

Relation « RelationalAlgebraAggToSqlGroupBy ». Cette relation est présentée par la figure 5.17 et convertit l'agrégation algébrique en clause « `Group By` ». L'attribut source « `si` » présente l'attribut agrégé du « `Select` » alors que l'attribut « `rasj` » présente l'attribut selon lequel se fait l'agrégation. Le prédicat présente une éventuelle condition définie sur les attributs d'agrégation. Il est ainsi converti en une clause « `Having` » qui peut faire appel à une fonction

d'agrégation SQL.

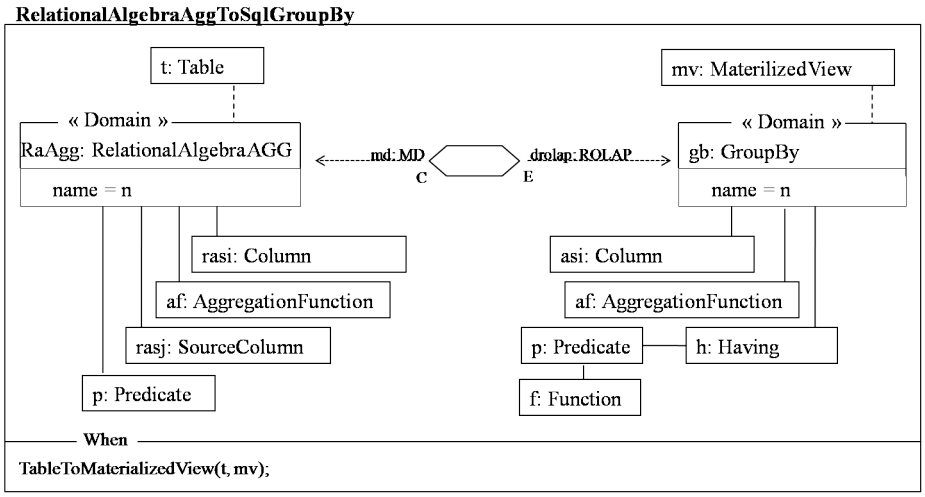


FIGURE 5.17 – Relation de transformation d'agrégations algébriques en agrégations SQL

Relation « RelationalAlgebraSelectionToSqlWhere ». Cette relation (cf. figure 5.18) permet la transformation de la sélection algébrique en un prédicat de la clause « `Where` ». Les différents éléments du prédicat algébrique sont transformés en leurs équivalents du prédicat SQL.

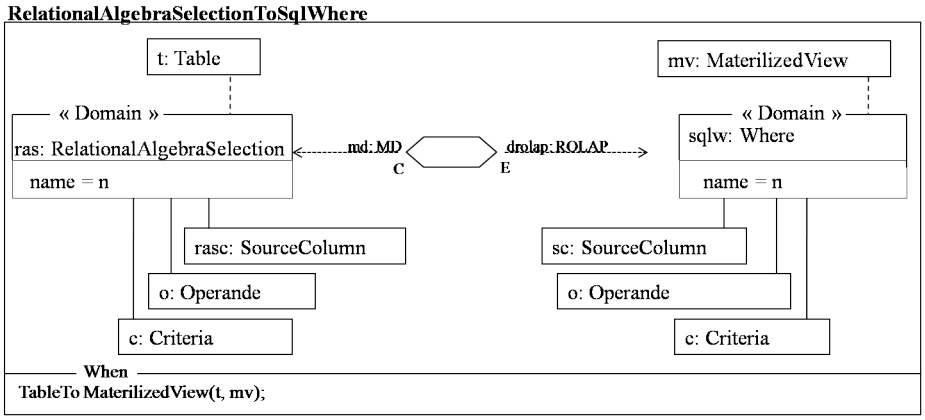


FIGURE 5.18 – Relation de transformation de sélections algébriques en prédicats SQL

Les figures 5.19 et 5.20 montrent des extraits du code QVT permettant la génération du PIM ROLAP à partir du PIM multidimensionnel. Le premier extrait

présente les relations de transformation des dimensions et des faits en tables. Le deuxième décrit la transformation des opérations ETL-OCL en opérations algébriques.

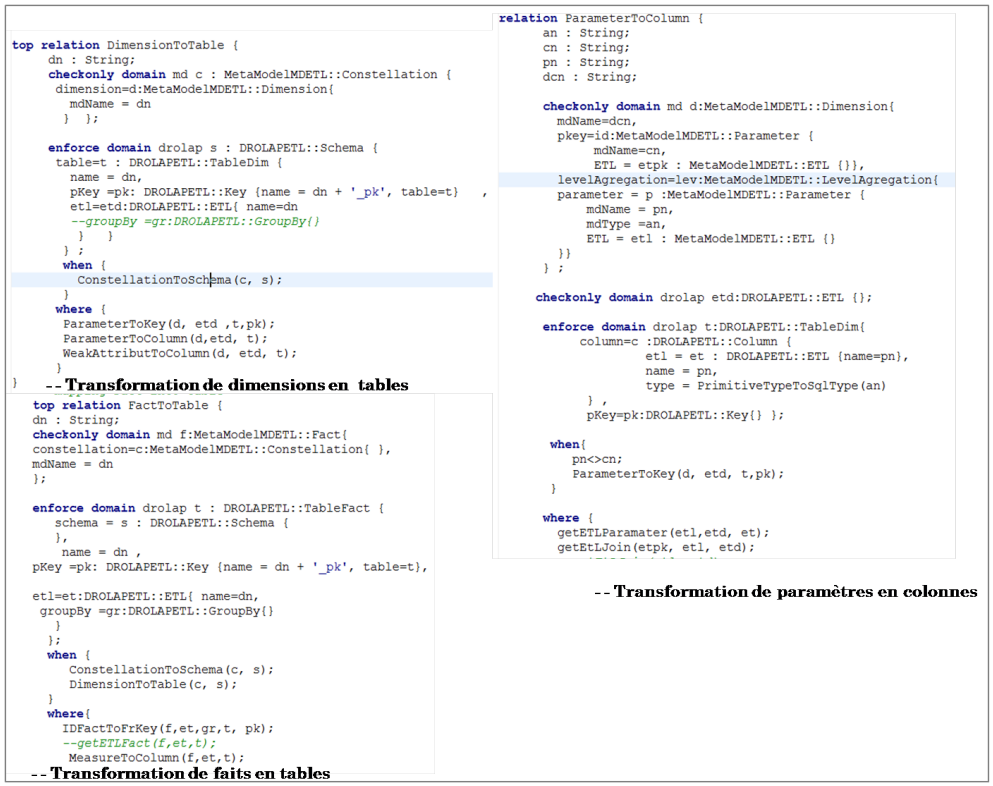


FIGURE 5.19 – Extrait du code QVT : transformation de faits et de dimensions en tables (syntaxe textuelle)

5.3.2.2 Transformations MOF2Text

Le code de création des vues matérialisées est généré en appliquant un ensemble de règles de transformation formalisées en MOF2Text. Ce langage permet de générer du code en appliquant les règles de transformation des éléments appartenant à un ou plusieurs modèle(s). Le processus de transformation d'un modèle en texte prend en entrée le modèle physique (PSM) et produit en sortie le ou les fichier(s) du code SQL permettant la création et le chargement des structures de l'entrepôt. La figure 5.21 montre un extrait du script MOF2Text.

FIGURE 5.20 – Extrait du code QVT : transformation des opérations ETL-OCL en opérations algébriques (syntaxe textuelle)

5.4 Etudes expérimentales sur les transformations

L'objectif de cette section est de montrer comment utiliser le système `DWAT` pour implanter un schéma multidimensionnel au niveau physique. Nous présentons les étapes d'implantation de l'étude de cas de l'application commerciale des ventes utilisée dans les chapitres précédents. Rappelons que cette étude de cas vise à analyser les `montants` et les `quantités` des « `Ventes` » en fonction des dimensions « `Client` », « `Produit` » et « `Temps` ».

L'utilisation de notre système revient à l'utilisation de l'environnement EMF. Comme le montre la figure 5.22, la première étape vise à instancier le métamodèle du PIM multidimensionnel (métamodèle en entrée). Un exemple de cette instance (`PIMMDVentes.xmi`) est présenté par la figure 5.23. Il est à noter que certaines informations du modèle telle que la liaison d'un fait à l'ensemble de ses dimensions (ou aussi la relation inverse) sont décrites au niveau des propriétés de l'élément. Par exemple, les propriétés du fait « `Vente` » (cf. figure 5.23) montre que ce fait est lié aux dimensions « `Client` », « `Produit` » et « `Temps` ». Une fois créée, l'instance doit être validée par l'utilisateur afin de vérifier la confor-

FIGURE 5.21 – Extrait du script MOF2Text

mité par rapport au métamodèle en cliquant sur le bouton « `Validate` » fourni par la plateforme EMF.

Le métamodèle du PIM multidimensionnel décrit les structures et les opérations ETL-OCL. Chaque attribut multidimensionnel est extrait à partir d'un ou plusieurs attribut(s) source(s). La liaison avec la source considérée au niveau métamodèle, est instanciée au niveau modèle en utilisant l'option « `Load Resource` » au niveau du fichier cible (`PIMMDVentes.xmi`). Cette option permet de charger la source de données sous forme d'un fichier `XMI`. Un extrait du diagramme de classes sources est montré par la figure 5.23 (b).

Afin d'exécuter les transformations QVT, il faut spécifier les chemins des métamodèles utilisés dans la transformation. Une démonstration de cette étape est disponible sur le site de documentation de l'outil MediniQVT [7]. La figure 5.24 (a) montre le PIM ROLAP et le PSM (b) obtenus respectivement par transformation du PIM multidimensionnel des `Ventes` et par fusion des deux PIM. A chaque résultat intermédiaire, il est recommandé au concepteur de valider le

7. http ://www.ikv.de/ikv_movies/mediniQVT.htm

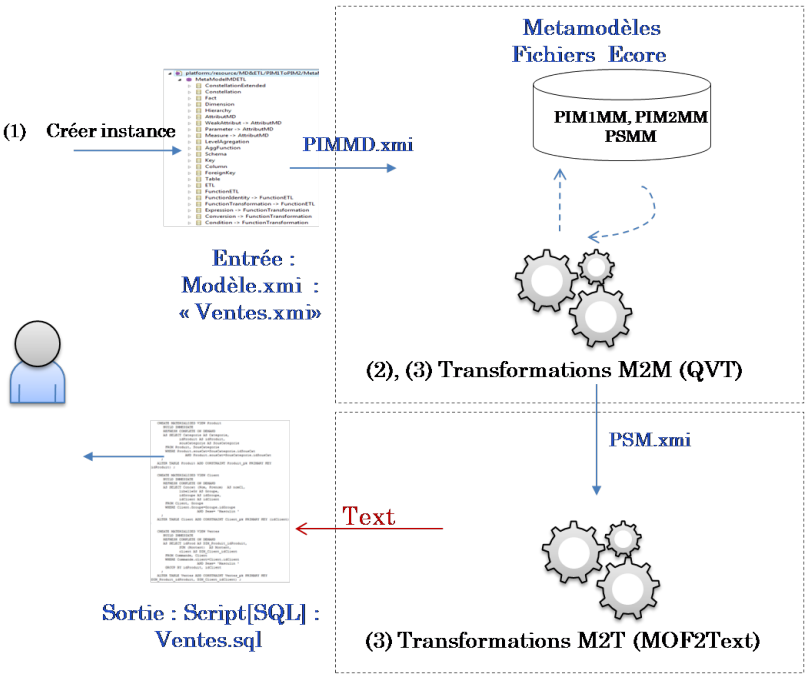


FIGURE 5.22 – Utilisation du système DWAT pour la transformation de schémas multidimensionnels

modèle en sortie. Notons qu'il est possible de modifier le modèle de manière manuelle.

Lors de la dernière étape, le PSM est transformé en code SQL. Le métamodèle et le modèle PSM sont passés en paramètres comme le montre le guide d'utilisation de l'outil MOFScript [8]. La figure 5.25 montre un extrait du code SQL fourni en résultat final.

5.5 Etudes expérimentales sur ED réduit

La réduction de données est un mécanisme qui vise à conserver uniquement l'information utile pour la prise de décision en utilisant les niveaux d'agrégation adéquats. Elle a pour but de garder uniquement l'information pertinente pour les décideurs afin d'obtenir les meilleures performances d'accès aux données. Il semble évident que toute réduction d'un entrepôt de données entraînera une diminution des temps de réponses aux requêtes dans la mesure où le volume des données est restreint. Nous avons cependant mesuré les gains de temps

8. http ://umt-qvt.sourceforge.net/mofscript/docs/MOFScript-User-Guide.pdf

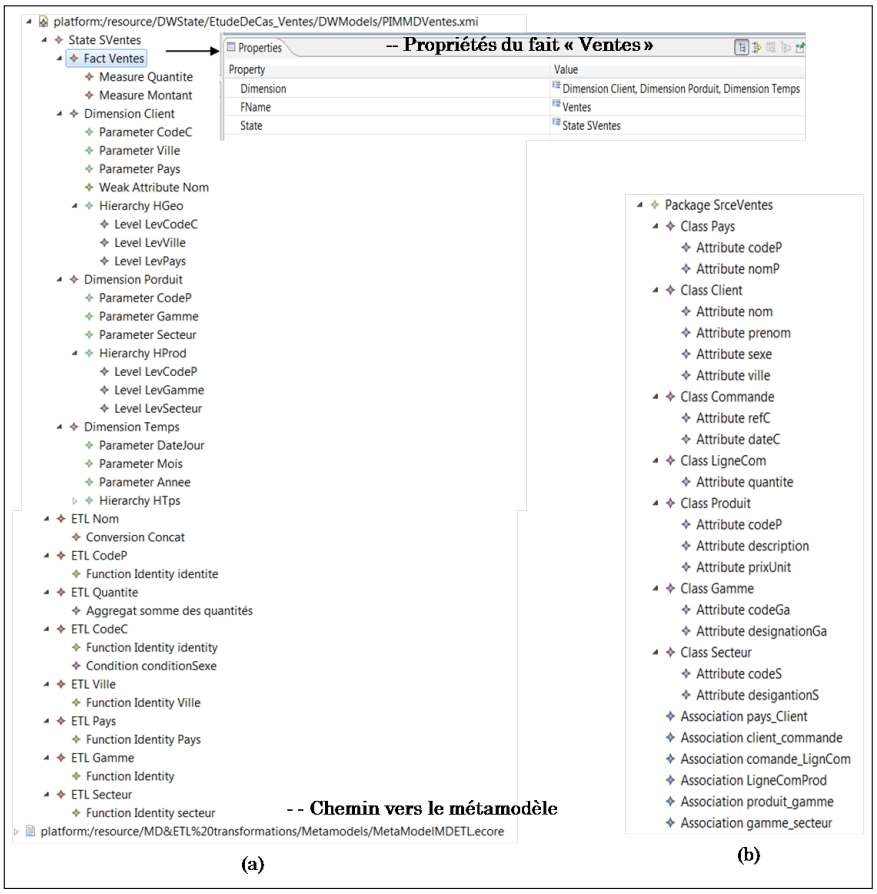


FIGURE 5.23 – Modèle multidimensionnel des `Ventes` entré par l'utilisateur (a) et PIM source (b)

de réponse que nous pourrions escompter après application du processus de réduction.

Dans cette section, nous présentons une étude comparative sur l'interrogation (1) d'ED non réduit, (2) d'ED réduit implanté en ROLAP dénormalisé et (3) d'ED réduit implanté en ROLAP normalisé. Cette étude est principalement basée sur deux critères : le temps d'exécution de requêtes et l'espace de stockage.

5.5.1 Collection de données

Prenons l'exemple du chapitre 3 (section 3.3.3) qui montre un entrepôt permettant l'analyse des montants et des quantités de ventes journalières selon les produits et les clients. Pour mener nos expérimentations, nous avons construit

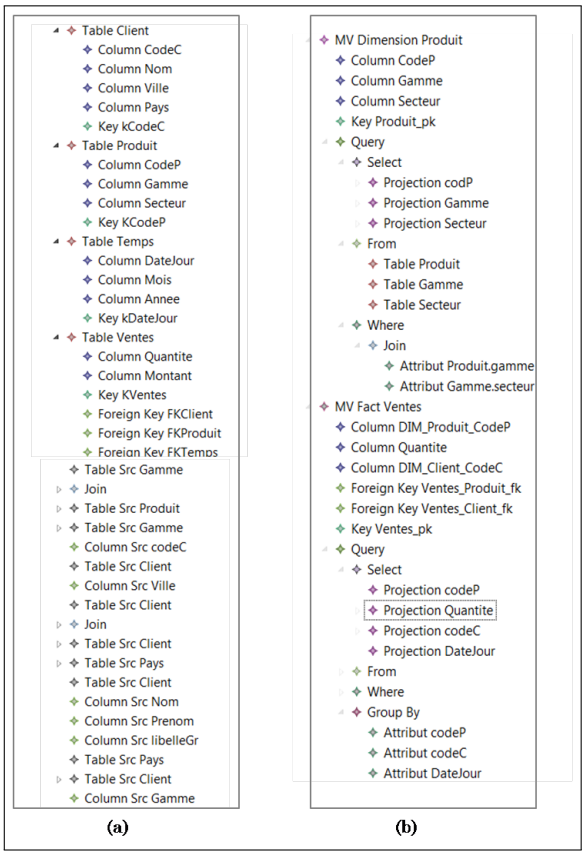
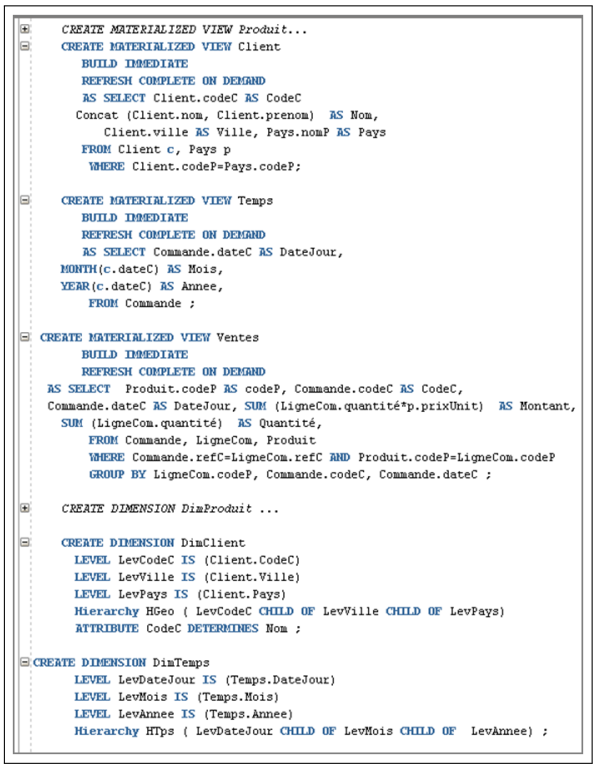


FIGURE 5.24 – Modèles générés par le système : PIM ROLAP (a) et PSM (b) de l'étude de cas des `Ventes`

un ED ROLAP avec le SGBD Oracle. L'alimentation de cet ED non réduit a été faite par le biais d'un programme PL/SQL qui charge les tables de dimension de manière aléatoire. Dans un second temps, le chargement du fait a été réalisé de façon systématique (produit cartésien des dimensions) ; cette solution pourrait nous écarter de la réalité, mais elle permet néanmoins de comparer les temps de réponse de façon significative.

La réduction de cet ED consiste à construire trois états conformes à l'étude de cas présentée dans le chapitre précédent. L'état courant noté E_1 est alimenté par des données générées aléatoirement. Les états réduits E_2 et E_3 sont dérivés conformément à l'exemple précédent.



```
CREATE MATERIALIZED VIEW Produit...
CREATE MATERIALIZED VIEW Client
   BUILD IMMEDIATE
   REFRESH COMPLETE ON DEMAND
   AS SELECT Client.codeC AS CodeC
   Concat (Client.nom, Client.prenom)  AS Nom,
       Client.ville AS Ville, Pays.nomP AS Pays
   FROM Client c, Pays p
    WHERE Client.codeP=Pays.codeP;

CREATE MATERIALIZED VIEW Temps
   BUILD IMMEDIATE
   REFRESH COMPLETE ON DEMAND
   AS SELECT Commande.dateC AS DateJour,
  MONTH(c.dateC) AS Mois,
  YEAR(c.dateC) AS Annee,
     FROM Commande ;

CREATE MATERIALIZED VIEW Ventes
     BUILD IMMEDIATE
     REFRESH COMPLETE ON DEMAND
 AS SELECT  Produit.codeP AS codeP, Commande.codeC AS CodeC,
 Commande.dateC AS DateJour, SUM (LigneCom.quantité*p.prixUnit)  AS Montant,
  SUM (LigneCom.quantité)  AS Quantité,
     FROM Commande, LigneCom, Produit
     WHERE Commande.refC=LigneCom.refC AND Produit.codeP=LigneCom.codeP
     GROUP BY LigneCom.codeP, Commande.codeC, Commande.dateC ;

CREATE DIMENSION DimProduit ...

CREATE DIMENSION DimClient
  LEVEL LevCodeC IS (Client.CodeC)
  LEVEL LevVille IS (Client.Ville)
  LEVEL LevPays IS (Client.Pays)
  Hierarchy HGeo ( LevCodeC CHILD OF LevVille CHILD OF LevPays)
  ATTRIBUTE CodeC DETERMINES Nom ;

CREATE DIMENSION DimTemps
     LEVEL LevDateJour IS (Temps.DateJour)
     LEVEL LevMois IS (Temps.Mois)
     LEVEL LevAnnee IS (Temps.Annee)
     Hierarchy HTps ( LevDateJour CHILD OF LevMois CHILD OF  LevAnnee) ;
```

FIGURE 5.25 – Extrait du code SQL généré par le système et relatif à l'étude de cas des `Ventes`

5.5.2 Protocole

L'expérimentation consiste à comparer un ED non réduit avec deux types d'implantation ROLAP. Nous proposons deux types de réduction d'ED : une implantation dénormalisée (schéma ROLAP en étoile) et une implantation normalisée (schéma ROLAP en flocon). Comme le montrent les figures 5.26 et 5.27, les schémas ROLAP des ED réduits sont définis selon trois états (E_1, E_2 et E_3).

L'ensemble des données de synthèse est généré en faisant varier le nombre de n-uplets des tables `Client` et `Produit` (les détails sont présentés dans le tableau 5.1) ; |Clients| et |Produits| varient de 10 à 40 n-uplets :

– |Client| = 10, 20, 30, 40 n-uplets
– |Produit| = 10, 20, 30, 40 n-uplets
– |Temps| = 8401 n-uplets (chaque jour, du 01/01/1990 jusqu'au 31/12/2012)
– |Ventes| = |Client| x |Produit| x |Temps| = de 840100 à 13441600 n-uplets.

La section suivante présente l'ensemble de requêtes utilisées pour évaluer les différentes implantations : ED sans réduction et ED réduit (dénormalisé ou

– – – – E1 –
Produit (codeP, gamme, marque, secteur)
Client (codeC, nom, prenom, ville, departement, region, type)
Temps (jour, mois, annee)
Ventes (codeP#, codeC#, jour#, montant, quantite)

– – – – E2 –
Produit (gamme, secteur)
Client (ville, departement, region)
Temps (jour, mois, annee)
Ventes (gamme#, ville#, jour#, montant, quantite)

– – – – E3 –
Produit (gamme, secteur)
Temps(annee)
Ventes (gamme#, annee#, montant)

FIGURE 5.26 – Schéma ROLAP dénormalisé

TABLE 5.1 – Taille des tables

|Client| x |Produit|	ED non réduit	ED réduit		
		E_1	E_2	E_3
10 x 10	|Ventes|=840.100 |Temps|=8.401	|Ventes|=109.600 |Temps|=1.096	|Ventes|=32.877 |Temps|=3.653	|Ventes|=30 |Temps|=10
20 x 20	|Ventes|=3.360.400 |Temps|=8.401	|Ventes|=438.400 |Temps|=1.096	|Ventes|=91.325 |Temps|=3.653	|Ventes|=50 |Temps|=10
30 x 30	|Ventes|=7.560.900 |Temps|=8.401	|Ventes|=986.400 |Temps|=1.096	|Ventes|=178.997 |Temps|=3.653	|Ventes|=70 |Temps|=10
40 x 40	|Ventes|=13.441.600 |Temps|=8.401	|Ventes|=1.753.600 |Temps|=1.096	|Ventes|=32.877 |Temps|=3.653	|Ventes|=90 |Temps|=10

normalisé).

5.5.3 Requêtes

Après avoir implanté les différents schémas et alimenté l'entrepôt de données, nous présentons l'ensemble des requêtes sur lesquelles est fondée l'étude comparative. Lorsque l'ED est réduit, l'utilisateur peut interroger un ou plusieurs états. Si la requête comporte un prédicat temporel, la portée de ce prédicat définit le nombre d'états interrogés. Lorsqu'aucun prédicat temporel n'est précisé, tous les états sont interrogés. L'interrogation des ED réduits (implantés en ROLAP dénormalisés et normalisés) se fait par le biais d'un programme PL/SQL. Lorsque la requête porte sur plusieurs états, dans un premier temps, cette requête est

– – – – E1 –
Produit (codeP, gamme#, marque#)
Gamme (gamme, secteur#)
Marque(marque, secteur)
Secteur(secteur)
Client (codeC, nom, prenom, ville#, type#)
Ville (ville, departement#)
Departement (departement, region#)
Region (region)
Type (type)
Jour (jour, mois#)
Mois (mois, annee#)
Annee (annee)
Ventes (codeP#, codeC#, jour#, montant, quantite)

– – – – E2 –
Produit (gamme, secteur#)
Secteur(secteur)
Client (ville, departement#)
Departement (departement, region#)
Region (region)
Jour (jour, mois#)
Mois (mois, annee#)
Annee (annee)
Ventes (gamme#, ville#, jour#, montant, quantite)

– – – – E3 –
Produit (gamme, secteur#)
Secteur(secteur)
Annee (annee)
Ventes (gamme#, annee#, montant)

FIGURE 5.27 – Schéma ROLAP normalisé

décomposée de manière manuelle afin d'établir une sous-requête pour chaque état. Le résultat final de la requête est obtenu par union des sous-requêtes.

Pour comparer les différentes implantations, nous avons défini un ensemble de quatorze requêtes exécutées sur les trois ED. Le tableau 5.2 montre ces requêtes, les états et les dimensions manipulés par chaque requête.

TABLE 5.2 – Liste des requêtes de test

	Requêtes	**Etats**	**Dimensions**
Q_{01}	Montant des ventes durant les trois dernières années	E_1	Temps
Q_{02}	Montant et quantité des ventes en 2008	E_2	Temps
Q_{03}	Montant des ventes annuelles antérieures à 2000	E_3	Temps
Q_{04}	Montant des ventes par ville de 2010 à 2012	E_1	Temps et Client
Q_{05}	Quantité des ventes mensuelles par département de 2000 à 2005	E_2	Temps et Client
Q_{06}	Montant des ventes annuelles par secteur avant 2000	E_3	Temps et Produit
Q_{07}	Montant des ventes par ville, secteur et mois en 2012	E_1	Temps, Produit et Client
Q_{08}	Montant des ventes annuelles par secteur et département de 2000 à 2005	E_2	Temps, Produit et Client
Q_{09}	Montant des ventes mensuelles depuis 2000	E_1 ; E_2	Temps
Q_{10}	Montant des ventes par année et ville de 2002 à 2012	E_1 ; E_2	Temps et Client
Q_{11}	Montant des ventes par année et gamme de 1990 à 2009	E_2 ; E_3	Temps et Produit
Q_{12}	Montant des ventes par ville et secteur de 2002 à 2012	E_1 ; E_2	Temps, Produit et Client
Q_{14}	Montant des ventes par année	E_1 ; E_2 ; E_3	Temps
Q_{15}	Montant des ventes par année et gamme	E_1 ; E_2 ; E_3	Temps et Produit

5.5.4 Résultats et interprétations

Le premier test compare les temps d'exécution théoriques des différentes requêtes (`Oracle DBMS explain plan`) en faisant varier la taille de l'ED suivant le protocole expérimental décrit précédemment. Dans chaque cas, on obtient un gain de temps significatif. Plus précisément, les requêtes sur un ED réduit sont exécutées avec un temps quatre à six fois plus court que sur un ED non réduit. Ceci s'explique par la taille des données manipulées (voir tableau 5.1).

La figure 5.29 montre le nombre de n-uplets manipulés par chaque requête : la quantité de données traitées dans un ED sans réduction est beaucoup plus élevée que dans un ED réduit. Comme le présente la figure 5.28, les coûts d'exécution sont beaucoup moins élevés dans un ED réduit par rapport à ceux d'un ED sans réduction.

En outre, nous notons (voir figure 5.30) que les coûts d'exécution des requêtes sont sensiblement les mêmes dans une implantation dénormalisée ou normalisée.

La figure 5.31 montre le gain de temps en pourcentage obtenu par l'exécution des requêtes sur les deux types d'ED ; avec et sans réduction. En moyenne, le gain atteint les 87% quelle que soit la taille de l'ED (de 840.100 à 13.441.600 de n-uplets dans la table du fait Ventes). Les courbes de tendance montrent que plus le nombre d'états interrogé est important (voir tableau 5.2), moins le gain en temps est significatif.

5.6 Conclusion

Le développement du prototype `DWAT` a permis de mettre en œuvre et de montrer la pertinence de notre approche dirigée par les modèles pour la formalisation et l'implantation conjointes d'ED réduits. L'implantation d'approches IDM nécessite l'utilisation d'environnements dédiés tels que la plateforme `Eclipse Modeling Framework`. Cette plateforme nous a permis de mettre en œuvre les métamodèles, les modèles et les transformations. Grâce à la mise en place de métamodèles et des transformations définissant les structures et les opérations de transformation de l'entrepôt, le système `DWAT` fournit en résultat le code SQL de création et de chargement de l'ED.

Nous avons mené des expériences qui portent sur la transformation de schémas d'ED ainsi que sur la réduction d'ED. Le temps d'exécution de l'implantation d'un schéma multidimensionnel au niveau physique est de l'ordre de quelques secondes par transformation. Notamment, le temps d'exécution de la transformation du PIM multidimensionnel vers le PIM ROLAP de l'application des Ventes est de l'ordre de `15ms`. Celui de la fusion de ces deux PIM est de l'ordre de `20ms`. Même si on fait varier l'exemple en rajoutant des faits et des dimensions, le temps d'exécution de l'ensemble des transformations reste de l'ordre de quelques secondes.

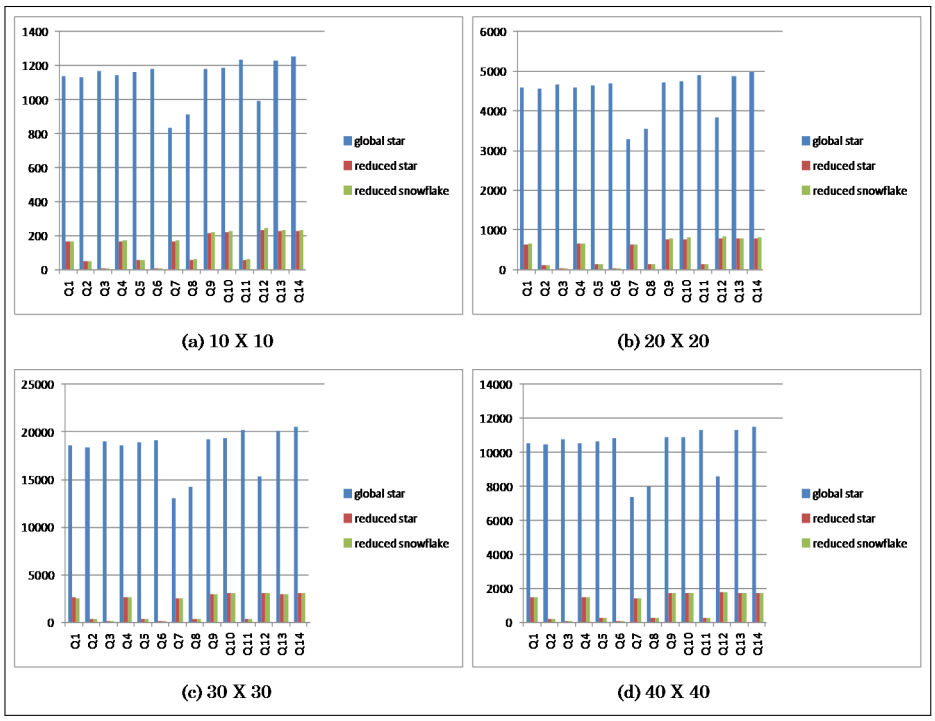


FIGURE 5.28 – Comparaison des coûts d'exécution des requêtes Q_1 à Q_{14}

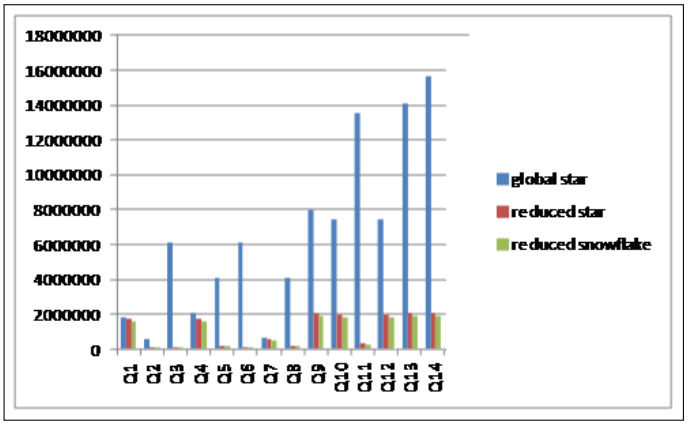


FIGURE 5.29 – Cardinalités (nombre de lignes) de requêtes Q_1 à Q_{14} pour une mise en œuvre 40x40

En ce qui concerne l'expérimentation de la réduction, nous avons comparé les temps d'exécution de requêtes sur un ED réduit (implanté en ROLAP dénormalisé et en ROLAP normalisé) avec un ED non réduit. Le premier test compare les

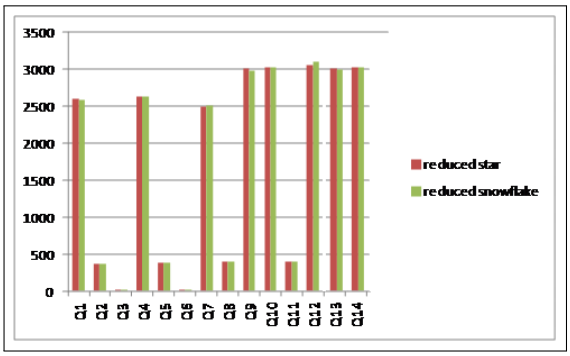


FIGURE 5.30 – Coûts d'exécution des requêtes Q_1 à Q_{14} pour une mise en œuvre 40x40

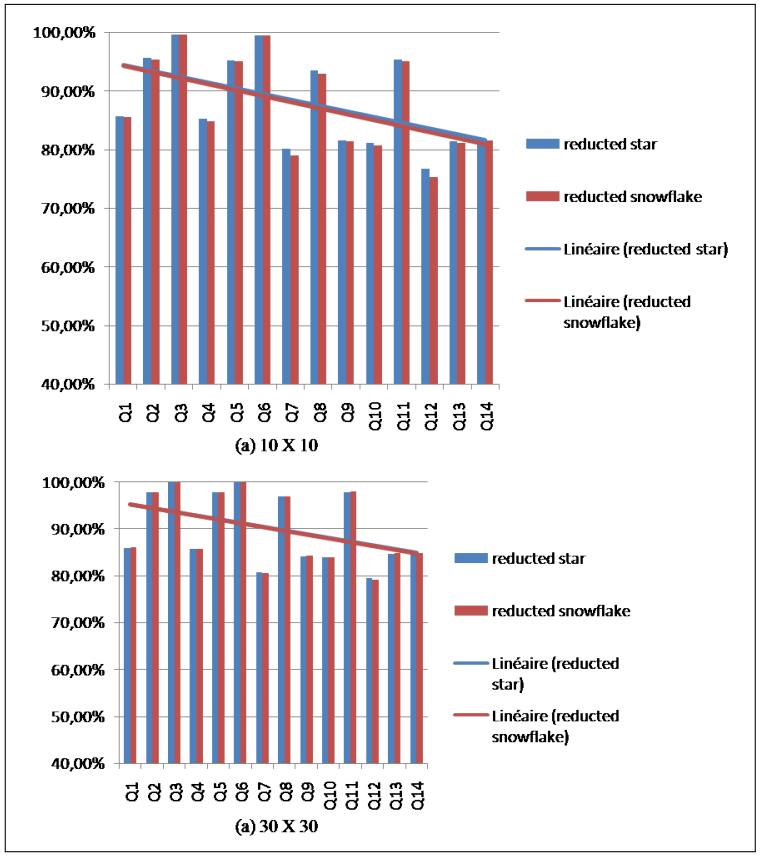


FIGURE 5.31 – Gain en temps d'exécution des requêtes Q_1 à Q_{14}

temps d'exécution théoriques des différentes requêtes en faisant varier la taille de l'ED. Dans les différents cas, le gain de temps obtenu est significatif. Les requêtes sur un ED réduit sont exécutées quatre à six fois plus rapidement que sur un ED non réduit. En moyenne, le gain de temps en pourcentage obtenu par l'exécution des requêtes sur les deux types d'ED ; avec et sans réduction atteint les 87% quelle que soit la taille de l'ED. Les coûts d'exécution sont beaucoup moins élevés dans un ED réduit par rapport à un ED sans réduction. En outre, nous notons que les coûts d'exécution des requêtes sont sensiblement les mêmes dans une implantation dénormalisée ou normalisée.

Chapitre 6

Conclusion générale

6.1 Bilan

Les travaux de recherche présentés dans ce mémoire de thèse s'inscrivent dans le cadre de systèmes d'aide à la décision reposant sur un entrepôt de données multidimensionnelles (ED). Jusqu'à présent peu de travaux ont considéré les problèmes de modélisation et d'implantation d'ED ainsi que les processus ETL de manière conjointe. Les méthodes actuelles proposent des solutions pour concevoir et implanter le schéma multidimensionnel d'une part, les processus d'alimentation d'autre part ; pourtant les deux traitements sont complémentaires et interdépendants. En outre, la plupart des approches manque de mécanismes pour générer automatiquement ou documenter ces deux aspects.

Dans cette thèse, notre premier objectif a été de pallier ces limites en fournissant une solution pour formaliser le processus d'élaboration d'ED historisés depuis sa conception jusqu'à son implantation physique et son alimentation. Afin de faciliter la tâche du concepteur, nous avons proposé une approche dirigée par les modèles pour l'élaboration automatique d'entrepôts de données. Notre approche est basée sur trois niveaux de modélisation. (1) Le PIM multidimensionnel décrit les données en termes de faits et de dimensions et les opérations de transformation en ETL-OCL. Ce dernier présente une extension du langage OCL permettant de formaliser les expressions de transformation des attributs multidimensionnels à partir des sources. (2) Le PIM ROLAP décrit les données en termes de tables et les transformations en termes d'opérations relationnelles. (3) Le PSM définit le modèle physique de la plateforme Oracle qui décrit les données sous forme de vues matérialisées et de dimensions. Les opérations sont formalisées en modèles de requêtes de définition des vues matérialisées. A partir du PSM est généré le script SQL de création des vues matérialisées et des dimensions au niveau de la plateforme Oracle. Notre approche fournit aussi un ensemble de transformations automatiques. (1) La transformation de modèles

où le PIM multidimensionnel est transformé en PIM ROLAP. Les faits et les dimensions sont convertis en tables. Les opérations ETL-OCL sont converties en opérations relationnelles. (2) La fusion de modèles où le PIM multidimensionnel et le PIM ROLAP sont fusionnés pour générer le PSM. Les tables ROLAP sont converties en vues matérialisées. Les dimensions du PIM multidimensionnel sont converties en dimensions Oracle. (3) La transformation de modèles vers code où le PSM est traduit en script SQL.

L'originalité de notre approche réside dans l'utilisation de l'ingénierie dirigée par les modèles (IDM) qui permet de formaliser et d'automatiser ce processus ; ceci en réduisant considérablement les coûts de développement et en améliorant la qualité du logiciel. Considérer la modélisation conjointe et l'implantation automatique du schéma de l'entrepôt ainsi que des opérations associées présente l'avantage d'éviter les problèmes d'intégration et d'interopérabilité rencontrés lors de l'utilisation d'approches différentes. Toutefois, modéliser conjointement les données et les opérations risque de produire des métamodèles complexes (nombre important d'éléments). La difficulté réside dans le fait de modéliser les opérations de transformation des attributs (opérations de conversion, de calcul, etc.) pour établir les règles de transformation. Par exemple, pour modéliser une expression simple telle que « $a + b = c$ », il faut définir au minimum deux classes pour modéliser les opérandes et les opérateurs ainsi qu'une interface pour énumérer les opérateurs.

D'autre part, au fil du temps, l'ED accumule d'importants volumes de données. Ceci est dû principalement à l'historisation régulière des données. En examinant les analyses dans le temps, on constate que les décideurs portent généralement un intérêt moindre pour les données anciennes ; mais ils souhaitent en conserver une trace. La gestion de ces volumes est lourde et entraîne des temps de réponse élevés. Pour répondre à ce problème, des travaux récents fournissent des solutions pour réduire ces données au fil du temps. Cependant, ces travaux se limitent à agréger progressivement les données du fait par rapport à ses dimensions. Ils n'offrent pas de moyens pour réduire les données en supprimant des éléments multidimensionnels tels que les dimensions, les faits ou les attributs. D'autre part, ces travaux manquent de mécanismes pour valider ou automatiser le processus de réduction. Aucune démarche n'a été fournie pour formaliser, décrire et automatiser l'implantation d'ED réduits.

Pour combler ces lacunes, nous avons proposé une approche dirigée par les modèles qui permet de formaliser et d'automatiser le processus de réduction de données afin de conserver uniquement les données nécessaires aux analyses décisionnelles. Cette approche présente deux étapes principales. (1) L'élaboration du PIM multidimensionnel où le concepteur construit l'ensemble des schémas multidimensionnels des états. Un état est construit à partir d'un état de référence en appliquant un ensemble d'opérateurs de réduction. Dans un état donné, la relation de correspondance d'un élément du schéma multidimensionnel à partir d'un élément de référence, est formalisée en ETL-OCL. (2) Les états du PIM multidimensionnel sont traduits en script SQL. Pour ce faire, d'abord le premier

PIM est transformé en PIM ROLAP qui permet de décrire l'ensemble des états de l'ED sous forme de tables ROLAP dénormalisées et des expressions algébriques permettant d'extraire un état à partir de son état de référence. Ensuite, les états du niveau PSM (Oracle) sont générés à partir du PIM ROLAP. Le PSM décrivant les différents états en termes de vues matérialisées et de dimensions, est généré. Ce modèle est par la suite transformé en code SQL de création et de chargement des vues matérialisées. L'avantage de notre approche est qu'elle fournit un cadre pour réduire un entrepôt de données multidimensionnelles et de l'implanter de manière automatique. Cependant, nous n'avons pas abordé l'interrogation d'ED réduit.

Afin de valider nos contributions, nous avons développé le prototype `DWAT`. Ce système est principalement composé de deux modules. (1) Le module de réduction qui permet d'implanter l'ensemble d'opérateurs de réduction et de vérifier l'ensemble des contraintes définies pour garantir la cohérence des schémas réduits. (2) Le module de transformation MDA qui permet de générer du code SQL à partir du PIM multidimensionnel entré par l'utilisateur.

6.2 Perspectives

Nos travaux ont permis de mettre en évidence certaines limites de notre contribution et nous a conduit à envisager quelques perspectives. Nous souhaitons aborder un ensemble d'extensions relatives aux opérations de transformation. Il est possible de considérer d'autres opérations de transformation ainsi que des opérations d'extraction et de chargement. Notamment dans le cas où l'on prend en compte plusieurs sources de données. Dans son état actuel, le système permet d'alimenter l'ED à partir de plusieurs modèles sources à condition que les opérations de transformation soient déjà définies dans le métamodèle et que la source soit relationnelle. Autrement dit, il est possible d'alimenter l'ED à partir de plusieurs instances de sources conformes aux métamodèles définis actuellement. La prise en compte d'autres types de sources de données requiert l'extension des différents métamodèles et des règles de transformations. Lorsque ces sources sont hétérogènes (structures différentes), il est indispensable de définir de nouveaux métamodèles et de nouvelles règles de transformation.

En ce qui concerne la réduction d'entrepôt, la question qui se pose est comment l'interroger. Les données qui intéressent le décideur peuvent être réparties entre l'état courant et les différents états réduits. Ainsi, des requêtes peuvent porter sur chacun des états pris séparément mais aussi sur plusieurs états. Nous envisageons de développer un langage d'interrogation adapté.

La génération automatique des modèles et du code procure un intérêt majeur à l'administrateur des données. Cependant, une question demeure : comment peut-on prouver la validité du résultat d'une transformation ? Plus précisément, est-ce qu'un modèle en sortie est cohérent et conforme au modèle en entrée et à

son métamodèle. En effet, dans le cas d'une table du modèle ROLAP en sortie qui n'a pas de clé primaire, la relation de conformité avec le métamodèle ROLAP n'est pas vérifiée. La vérification des transformations est une problématique de recherche connue dans le domaine de l'ingénierie dirigée par les modèles. Dans ce contexte, il existe des travaux de recherche qui ont traité des aspects liés à ce problème, notamment les travaux de [Giese et al., 2006], [Lano and Clark, 2008], [Cabot et al., 2010]. Nous envisageons d'améliorer notre approche IDM en définissant des contraintes de vérification formalisées en OCL pour garantir la qualité des modèles en sortie de chaque de transformation.

L'évolution d'un entrepôt représente une autre perspective liée à notre problématique d'élaboration d'ED . L'évolution peut se situer au niveau des besoins mais aussi au niveau des sources. Certes, l'étude de l'évolution dans les ED a fait l'objet de plusieurs travaux comme présentés dans [Golfarelli and Rizzi, 2009] et [Wrembel, 2009]. Cependant, l'évolution dans le cadre d'entrepôts de données dirigés par les modèles présentent des particularités [van Deursen et al., 2007], nous pouvons citer dans ce contexte les travaux de [Kurze et al., 2011] et de [Taktak and Feki, 2012]. Dans notre approche IDM, lorsque les besoins évoluent (ajout, suppression ou modification des éléments du schéma multidimensionnel), il suffit de relancer le processus de transformation pour aboutir au code SQL correspondant. Lorsqu'il s'agit de l'ajout, il faut évidemment considérer les expressions de correspondance avec la source. Si l'évolution a eu lieu au niveau de la source, dans un premier temps, les expressions ETL-OCL sont mises à jour pour adapter le modèle en entrée. Dans un second temps, les processus de transformation sont relancés. Le problème de l'évolution serait plus difficile à gérer lorsque les métamodèles évoluent.

Bibliographie

Fatma Abdelhédi and Gilles Zurfluh. User support system for designing decisional database. In *Proceedings of the 6th International Conference on Advances in Computer-Human Interactions, ACHI*, pages 377–382, Nice, France, 2013.

Lou Agosta. *Data Warehousing Meets Data Archiving in Information Lifecycle Management*, 2008. URL http://www.information-management.com/news/10001092-1.html?zkPrintable=true.

Lou Agosta, Marc Andrews, and Mark Ritzmann. *The Data Warehouse Satisfaction Survey*. IBM, 2007. URL http://www.information-management.com/specialreports/20071002/1093126-1.html.

Francis Alizon, Mariano Belaunde, Grégoire DuPre, Bertrand Nicolas, Sébastien Poivre, and Jacques Simonin. Les modèles dans l'action à France Télécom avec SmartQVT. In *Génie logiciel : Congrès Journées Neptune No5*, 2007. URL http://smartqvt.elibel.tm.fr.

Carsten Amelunxen, Alexander Königs, Tobias Rötschke, and Andy Schürr. MOFLON : A Standard-Compliant Metamodeling Framework with Graph Transformations. In *Proceedings of the 2nd European Conference on Model Driven Architecture - Foundations and Applications, ECMDA-FA*, pages 361–375, Bilbao, Spain, 2006.

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Communications of the ACM, CACM*, 53(4) :50–58, 2010.

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Démarche dirigée par les modèles pour la conception d'entrepôts de données multidimensionnelles (papier long). In *26ème Journées des Bases de Données Avancées, BDA*, page (support électronique), Toulouse, France, 2010. URL ftp://ftp.irit.fr/IRIT/SIG/2010_ARTZ_BDA.pdf.

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Modèle unifié pour la transformation des schémas en constellation. In *Actes des 7ème journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA*, pages 5–22, Clermont-Ferrand, France, 2011a.

Faten Atigui, Franck Ravat, Ronan Tournier, and Gilles Zurfluh. A unified model driven methodology for data warehouses and ETL design. In *Proceedings of the 13th International Conference on Enterprise Information Systems, ICEIS*, pages 247–252, Beijing, China, 2011b.

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Using OCL for automatically producing multidimensional models and ETL processes. In *Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 42–53, Vienna, Austria, 2012a.

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Modélisation conjointe des données et des processus pour l'implantation de schémas d'entrepôts. *Journal des Systèmes Décisionnels (Journal of Decision Systems), JDS*, 21(1) :27–49, 2012b.

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Archivage d'entrepôts de données multidimensionnelles. In *Actes des 8èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA*, pages 129–138, Bordeaux, France, 2012c.

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Modèle d'archivage d'entrepôts de données multidimensionnelles. In *Actes du XXXème Congrès INFORSID, INFORSID*, pages 473–488, Montpellier, France, 2012d.

ATL Development Team. ATL website, 2013. URL http://www.eclipse.org/atl/. Available from : http://www.eclipse.org/atl/.

José Barateiro and Helena Galhardas. A Survey of Data Quality Tools. *Datenbank-Spektrum*, 14 :15–21, 2005.

Lotfi Bejaoui, François Pinet, Michel Schneider, and Yvan Bédard. OCL for formal modelling of topological constraints involving regions with broad boundaries. *GeoInformatica*, 14(3) :353–378, 2010.

Randall G. Bello, Karl Dias, Alan Downing, James J. Feenan Jr., James L. Finnerty, William D. Norcott, Harry Sun, Andrew Witkowski, and Mohamed Ziauddin. Materialized Views in Oracle. In *Proceedings of 24th International Conference on Very Large Data Bases, VLDB*, pages 659–664, New York City, New York, USA, 1998.

Jean Bézivin and Olivier Gerbé. Towards a Precise Definition of the OMG/MDA Framework. In *Proceedings of the 16th International Conference on Automated Software Engineering, ASE*, pages 273–280, Washington, DC, USA, 2001.

Aliou Boly, Sabine Goutier, and Georges Hébrail. Des fonctions d'oubli intelligentes dans les entrepôts de données. In *Actes des 5ème journées francophones sur l'Extraction et la Gestion des Connaissances, EGC*, pages 223–234, Namur, Belgique, 2007.

Angela Bonifati, Fabiano Cattaneo, Stefano Ceri, Alfonso Fuggetta, and Stefano

Paraboschi. Designing data marts for data warehouses. *Transactions on Software Engineering and Methodology, TOSEM*, 10(4) :452–483, 2001.

Frank Budinsky, Stephen A. Brodsky, and Ed Merks. *Eclipse Modeling Framework*. Pearson Education, 2003. ISBN 0131425420.

Jordi Cabot, Robert Clarisó, Esther Guerra, and Juan de Lara. Verification and validation of declarative model-to-model transformations through invariants. *Journal of Systems and Software*, 83(2) :283–302, 2010.

Andrea Carmè, Jose-Norberto Mazón, and Stefano Rizzi. A Model-Driven Heuristic Approach for Detecting Multidimensional Facts in Relational Data Sources. In *Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 13–24, Bilbao, Spain, 2010.

Claude Chrisment, Karen Pinel-Sauvagnat, Olivier Teste, and Michel Tuffery. *Bases de données relationnelles : concepts, mise en œuvre & exercices*. Hermès Science, 2008.

Benoît Combemale. Ingénierie Dirigée par les Modèles (IDM) - État de l'art. État de l'art, 2008. URL http://hal.archives-ouvertes.fr/hal-00371565.

Alfredo Cuzzocrea. CAMS : OLAPing Multidimensional Data Streams Efficiently. In *Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 48–62, Linz, Austria, 2009.

Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3) :621–645, 2006.

Marcos D. Del Fabro, Jean Bézivin, Frédéric Jouault, Erwan Breton, and Guillaume Gueltas. AMW : a generic model weaver. In *Actes des 1ères Journée sur l'Ingénierie Dirigée par les Modèles, IDM*, 2005.

Samba Diaw, Redouane Lbath, and Bernard Coulette. Etat de l'art sur le développement logiciel basé sur les transformations de modèles. *Technique et Science Informatiques, Ingénierie Dirigée par les Modèles*, 29 (4-5) :505–536, 2010. URL ftp://ftp.irit.fr/IRIT/MACAO/Diaw_et_al-Diaw-et-al2010.pdf%20.

Selma Djeddai. *Combining Formal Verification Environments and Model-Driven Engineering*. PhD thesis, Université Paul Sabatier, Toulouse, 2013.

Karsten Ehrig, Esther Guerra, Juan de Lara, Laszló Lengyel, Tihamér Levendovszky, Ulrike Prange, Gabriele Taentzer, Dániel Varró, and Szilvia Varró-Gyapay. Model transformation by graph transformation : A comparative study. In *International Workshop on Model Transformations in Practice (Satellite Event of MoDELS 2005), MTiP*, Montego Bay, Jamaica, 2005.

Zineb El Akkaoui and Esteban Zimányi. Defining ETL worfklows using BPMN

and BPEL. In *Proceedings of the 12th International Workshop on Data warehousing and OLAP, DOLAP*, pages 41–48, Hong Kong, China, 2009.

Zineb El Akkaoui, Esteban Zimànyi, Jose-Norberto Mazón, and Juan Trujillo. A model-driven framework for ETL process development. In *Proceedings of the 14th International Workshop on Data warehousing and OLAP, DOLAP*, pages 45–52, Glasgow, Scotland, UK, 2011.

Zineb El Akkaoui, Jose-Norberto Mazón, Alejandro A. Vaisman, and Esteban Zimányi. BPMN-Based Conceptual Modeling of ETL Processes. In *Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 1–14, Vienna, Austria, 2012.

Hector Garcia-Molina, Wilburt Labio, and Jun Yang. Expiring Data in a Warehouse. In *Proceedings of 24th International Conference on Very Large Data Bases, VLDB*, pages 500–511, New York City, New York, USA, 1998.

Holger Giese, Sabine Glesner, Johannes Leitner, Wilhelm Schäfer, and Robert Wagner. Towards Verified Model Transformations. In *Proceedings of Modeva workshop associated to Models*, pages 78–93, Genova, Italy, 2006.

Paolo Giorgini, Stefano Rizzi, and Maddalena Garzetti. Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the 8th International Workshop on Data warehousing and OLAP, DOLAP*, pages 47–56, Bremen, Germany, 2005.

Matteo Golfarelli and Stefano Rizzi. Methodological Framework for Data Warehouse Design. In *Proceedings of the 1st International Workshop on Data warehousing and OLAP, DOLAP*, pages 3–9, Bethesda, Maryland, USA, 1998.

Matteo Golfarelli and Stefano Rizzi. A Survey on Temporal Data Warehousing. *International Journal of Data Warehousing and Mining, IJDWM*, 5(1) :1–17, 2009.

Oksana Grabova, Jerôme Darmont, Jean-Hugues Chauchat, and Iryna Zolotaryova. Business intelligence for small and middle-sized entreprises. *SIGMOD Record*, 39(2) :39–50, 2010.

Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, and Y. Dora Cai. Stream Cube : An Architecture for Multi-Dimensional Analysis of Data Streams. *Distributed and Parallel Databases*, 18(2) :173–197, 2005.

Bodo Hüsemann, Jens Lechtenbörger, and Gottfried Vossen. Conceptual data warehouse modeling. In *Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses, DMDW*, page 6, Stockholm, Sweden, 2000.

John Hutchinson, Mark Rouncefield, and Jon Whittle. Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on*

Software Engineering, ICSE, pages 633–642, Waikiki, Honolulu, HI, USA, 2011.

Nadeem Iftikhar and Torben Bach Pedersen. Using a Time Granularity Table for Gradual Granular Data Aggregation. In *Proceedings of the 14th East European Conference Advances in Databases and Information Systems, ADBIS*, pages 219–233, Novi Sad, Serbia, 2010.

Nadeem Iftikhar and Torben Bach Pedersen. A rule-based tool for gradual granular data aggregation. In *Proceedings of the 14th International Workshop on Data Warehousing and OLAP, DOLAP*, pages 1–8, Glasgow, United Kingdom, 2011.

Informatica. *White Paper : Data Warehouse Archiving : A Way to Optimize Data Warehouse Performance and Reduce Costs*, 2010.

Bill Inmon. *Building the Data Warehouse*, volume 2nd edition. John Wiley and Sons, New York, 1996.

Mikael R. Jensen, Thomas Holmgren, and Torben Bach Pedersen. Discovering Multidimensional Structure in Relational Data. In *15th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 138–148, Prague, Czech Republic, 2004.

Frédéric Jouault and Ivan Kurtev. Transforming models with ATL. In *International Workshop on Model Transformations in Practice (Satellite Event of MoDELS 2005), MTiP*, pages 128–138, Montego Bay, Jamaica, 2006.

Ahmad Kheir, Mourad Oussalah, and Hala Naja. Hierarchical Multi-Views Software Architecture. In *Proceedings of the 8th International Conference on Software Engineering Advances, ICSEA*, pages 478–484, Venice, Italy, 2013.

Ralph Kimball. *The Data Warehouse Toolkit : Practical Techniques for Building Dimensional Data Warehouses*. John Wiley, 1996. ISBN 0-471-15337-0.

Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained : The Model Driven Architecture : Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 032119442X.

Christian Kurze, Marcus Hofmann, Frieder Jacobi, André Müller, and Peter Gluchowski. Towards Model-driven Evolution of Data Warehouses. In *Proceedings of the 13th International Conference on Enterprise Information Systems, ICEIS*, pages 356–360, Beijing, China, 2011.

K. Lano and D. Clark. Model transformation specification and verification. In *Proceedings of the 8th International Conference Quality Software, ICSQ*, pages 45–54, Oxford, UK, 2008.

Juan de Lara and Hans Vangheluwe. Atom3 : A tool for multi-formalism and meta-modelling. In *Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering, FASE*, pages 174–188, London, UK, 2002.

Sergio Luján-Mora, Panos Vassiliadis, and Juan Trujillo. Data Mapping Diagrams for Data Warehouse Design with UML. In *Proceedings of the 23rd International Conference on Conceptual Modeling, ER*, pages 191–204, Shanghai, China, 2004.

Jose-Norberto Mazón and Juan Trujillo. An MDA approach for the development of data warehouses. *Decision Support Systems*, 45 :41–58, 2008.

Jose-Norberto Mazón and Juan Trujillo. A hybrid model driven development framework for the multidimensional modeling of data warehouses ! *SIGMOD Record*, 38(2) :12–17, 2009.

Mohamed Mkaouar, Rafik Bouaziz, and Mohamed Moalla. Querying and manipulating temporal databases. *The Computing Research Repository, CoRR*, abs/1103.0686 :arXiv preprint, 2011.

Daniel L. Moody and Mark A. R. Kortink. From enterprise models to dimensional models : a methodology for data warehouse and data mart design. In *Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses, DMDW*, page 5, Stockholm, Sweden, 2000.

Pierre-Alain Muller, Franck Fleurey, Didier Vojtisek, Zoé Drey, Damien Pollet, Frédéric Fondement, and Jean-Marc Jézéquel. On executable meta-languages applied to model transformations. In *International Workshop on Model Transformations in Practice (Satellite Event of MoDELS 2005), MTiP*, Montego Bay, Jamaica, 2005.

Lilia Muñoz, Jose-Norberto Mazón, Jesús Pardillo, and Juan Trujillo. Modelling ETL Processes of Data Warehouses with UML Activity Diagrams. In *Proceedings of the International Workshop On the Move to Meaningful Internet Systems, OTM Workshops*, pages 44–53, Monterrey, Mexico, 2008.

Lilia Muñoz, Jose-Norberto Mazón, and Juan Trujillo. Automatic generation of etl processes from conceptual models. In *Proceedings of the 12th International Workshop on Data Warehousing and OLAP, DOLAP*, pages 33–40, Hong Kong, China, 2009.

Kjetil Nørvåg. Granularity reduction in temporal document databases. *Information Systems Journal*, 31(2) :134–147, 2006.

Oleg Okun and Helen Priisalu. Unsupervised data reduction. *Signal Processing*, 87(9) :2260–2267, 2007.

Object Management Group OMG. *MDA Guide Version 1.0.1*, June 2003. URL http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf.

Object Management Group OMG. *MOF Model To Text Transformation Language (MOFM2T), Version 1.0*, January 2008. URL http://www.omg.org/spec/MOFM2T/1.0/PDF.

Object Management Group OMG. *Object Constraint Language (OCL) 2.2 Specification*, February 2010. URL http://www.omg.org/spec/OCL/2.2.

Object Management Group OMG. *Meta Object Facility (MOF) Core Specification, , Version 2.4.1*, August 2011a. URL http://www.omg.org/spec/MOF/2.4.1/PDF/.

Object Management Group OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification 1.1*, January 2011b. URL http://www.omg.org/spec/QVT/1.1/PDF/.

Object Management Group OMG. *OMG MOF 2 XMI Mapping Specification, Version 2.4.1*, August 2011c. URL http://www.omg.org/spec/XMI/2.4.1.

Marc Palyart, David Lugato, Ileana Ober, and Jean-Michel Bruel. Improving Scalability and Maintenance of Software for High-Performance Scientific Computing by Combining MDE and Frameworks. In *Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems, MoDELS*, pages 213–227, Wellington, New Zealand, 2011.

Jesús Pardillo, Jose-Norberto Mazón, and Juan Trujillo. Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses. *Information Sciences*, 180(5) :584–601, 2010.

Cassandra Phipps and Karen C. Davis. Automating data warehouse conceptual schema design and evaluation. In *Proceedings of the 4th International Conference on Design and Management of Data Warehouses, DMDW*, pages 23–32, Toronto, Canada, 2002.

François Pinet and Michel Schneider. A Unified Object Constraint Model for Designing and Implementing Multidimensional Systems. *Journal on Data Semantics XIII*, 5530 :37–71, 2009.

Yoann Pitarch, Anne Laurent, Marc Plantevit, and Pascal Poncelet. Multidimensional data stream summarization using extended tilted-time windows. In *Proceedings of the 23rd International Workshop on Advanced Information Networking and Applications, AINA*, pages 250–254, Bradford, United Kingdom, 2009.

Meikel Pöss and Dmitry Potapov. Data compression in oracle. In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB*, pages 937–947, Berlin, Germany, 2003.

Nicolas Prat, Jacky Akoka, and Isabelle Comyn-Wattiau. A uml-based data warehouse design method. *Decision Support Systems*, 42(3) :1449–1473, 2006.

Franck Ravat, Olivier Teste, Ronan Tournier, and Gilles Zurfluh. Algebraic and graphic languages for olap manipulations. *International Journal of Data Warehousing and Mining, IJDWM*, 4(1) :17–46, 2008.

Stefano Rizzi, Alberto Abelló, Jens Lechtenbörger, and Juan Trujillo. Research in data warehouse modeling and design : dead or alive ? In *Proceedings of the 9th International Workshop on Data Warehousing and OLAP, DOLAP*, pages 3–10, Arlington, Virginia, USA, 2006.

John F. Roddick. Schema vacuuming in temporal databases. *Transactions on Knowledge and Data Engineering*, 21(5) :744–747, 2009.

Oscar Romero and Alberto Abelló. Automating multidimensional design from ontologies. In *Proceedings of the 9th International Workshop on Data warehousing and OLAP, DOLAP*, pages 1–8, Lisbon, Portugal, 2007.

Oscar Romero and Alberto Abelló. A Survey of Multidimensional Modeling Methodologies. *International Journal of Data Warehousing and Mining, IJDWM*, 5(2) :1–23, 2009.

Oscar Romero and Alberto Abelló. Automatic validation of requirements to support multidimensional design. *Data & Knowledge Engineering*, 69(9) : 917–942, 2010.

Oscar Romero, Alkis Simitsis, and Alberto Abelló. GEM : Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. In *Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 80–95, Toulouse, France, 2011.

Camille Salinesi and Inès Gam. How Specific should Requirements Engineering be in the Context of Decision Information Systems ? In *Proceedings of the 3rd International Conference on Research Challenges in Information Science, RCIS*, pages 247–254, Fès, Morocco, 2009.

Arun Sen and Atish P. Sinha. A comparison of data warehousing methodologies. *Communication of the ACM*, 48(3) :79–84, 2005.

Alkis Simitsis. Mapping conceptual to logical models for ETL processes. In *Proceedings 8th International Workshop on Data Warehousing and OLAP, DOLAP*, pages 67–76, Bremen, Germany, 2005.

Alkis Simitsis and Panos Vassiliadis. A Methodology for the Conceptual Modeling of ETL Processes. In *Proceedings of the 15th Workshops on Advanced Information Systems Engineering, CAiSE Workshops*, Klagenfurt/Velden, Austria, 2003.

Alkis Simitsis, Panos Vassiliadis, Manolis Terrovitis, and Spiros Skiadopoulos. Graph-Based Modeling of ETL Activities with Multi-level Transformations and Updates. In *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, pages 43–52, Copenhagen, Denmark, 2005.

Janne Skyt, Christian S. Jensen, and Torben Bach Pedersen. Specification-based data reduction in dimensional data warehouses. *Information Systems Journal*, 33(1) :36–63, 2008.

Il-Yeol Song, Ritu Khare, and Bing Dai. Samstar : a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In *Proceedings of the 10th International Workshop on Data Warehousing and OLAP, DOLAP*, pages 9–16, Lisbon, Portugal, 2007.

Gabriele Taentzer. AGG : A Graph Transformation Environment for Modeling and Validation of Software. In *Proceedings of the 2nd International Workshop Applications of Graph Transformations with Industrial Relevance, AGTIVE 2003*, pages 446–453, Charlottesville, VA, USA, 2003.

Saïd Taktak and Jamel Feki. Toward Propagating the Evolution of Data Warehouse on Data Marts. In *Proceedings of the 2nd International Conference on Model and Data Engineering, MEDI*, pages 178–185, Poitiers, France, 2012.

Juan Trujillo and Sergio Luján-Mora. A UML Based Approach for Modeling ETL Processes in Data Warehouses. In *Proceedings of the 22nd International Conference on Conceptual Modeling, ER*, pages 307–320, Chicago, IL, USA, 2003.

Aris Tsois, Nikos Karayannidis, and Timos K. Sellis. Mac : Conceptual data modeling for OLAP. In *Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses, DMDW*, page 5, Interlaken, Switzerland, 2001.

James Udo, Ifiok and Babajide Afolabi. Hybrid Data Reduction Technique for Classification of Transaction Data. *Journal of Computer Science and Engineering*, 6(2) :12–16, 2011.

Arie van Deursen, Eelco Visser, and Jos Warmer. Model-Driven Software Evolution : A Research Agenda. In *Proceedings of the 1st International Workshop on Model-Driven Software Evolution, MoDSE*, pages 41–49, Amsterdam, The Netherlands, 2007.

Dániel Varró, Gergely Varró, and András Pataricza. Designing the automatic transformation of visual languages. *Science of Computer Programming Journal*, 44(2) :205–227, 2002.

Panos Vassiliadis. A Survey of Extract-Transform-Load Technology. *International Journal of Data Warehousing and Mining, IJDWM*, 5(3) :1–27, 2009.

Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual modeling for ETL processes. In *Proceedings 5th International Workshop on Data Warehousing and OLAP, DOLAP*, pages 14–21, McLean, Virginia, USA, 2002.

X. Sean Wang, Claudio Bettini, Alexander Broadsky, Sushil Jajodia, Name X. Sean Wang, and Name Sushil Jajodia. Logical Design for Temporal Databases with Multiple Granularities, 1997.

Jos Warmer and Anneke Kleppe. *The Object Constraint Language : Getting Your Models Ready for MDA*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2003.

Robert Wrembel. A survey of managing the evolution of data warehouses. *International Journal of Data Warehousing and Mining, IJDWM*, 5(2) :24–56, 2009.

Blanc Xavier. *MDA en action : ingénierie logicielle guidée par les modèles.* Eyrolles, 2005.

Eric S. K. Yu. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the 3rd International Symposium on Requirements Engineering, RE*, pages 226–235, Annapolis, MD, USA, 1997.

Leopoldo Zepeda, Matilde Celma, and Ramón Zatarain. A Mixed Approach for Data Warehouse Conceptual Design with MDA. In *Proceedings of the 14th International Conference on Computational Science and Its Applications, ICCSA (2)*, pages 1204–1217, Perugia, Italy, 2008.