Modèle unifié pour la transformation des schémas en constellation

Faten Atigui[*,**], Franck Ravat[*,**], Olivier Teste [*,***], Gilles Zurfluh[*,**]

[*]IRIT (UMR 5505), Institut de Recherche en Informatique de Toulouse,
118 route de Narbonne, 31062 Toulouse, France
[**]Université Toulouse 1 Capitole, 2 rue G. Marty, 31042 Toulouse Cedex9, France
[***]Université Toulouse 3 Paul Sabatier 118 route de Narbonne, 31062 Toulouse, France
{atigui, ravat, teste, zurfluh}@irit.fr

Résumé. Au cours de ces dernières années, plusieurs approches ont abordé la modélisation et le développement des Entrepôts de Données (ED). La plupart de ces approches fournit des solutions partielles qui traitent soit la modélisation multidimensionnelles, soit la modélisation des processus d'Extraction-Transformation-Loading (ETL). Toutefois, peu de travaux ont visé à unifier ces deux problématiques dans un cadre structuré ou à automatiser le processus d'entreposage. Afin de pallier ces limites, nous proposons dans ce papier une démarche unifiée et semi-automatique qui intègre la modélisation des ED et des processus ETL. Cette démarche est définie dans le cadre d'une Architecture Dirigée par les Modèles (MDA). Elle permet (i) de formaliser les besoins des décideurs, ensuite (ii) de générer les modèles conceptuel, logique et physique de l'ED et des processus ETL conjoints (iii) ainsi que les codes de création et d'alimentation (ETL) des structures multidimensionnelles. Les règles de transformations entre modèles sont formalisées en Query/View/Transformation.

1 Introduction

Un entrepôt de données (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 couramment connus sous le nom ETL (Extraction-Transformation-Loading) (Vassiliadis, 2009). Les données extraites sont souvent structurées selon un format multidimensionnel qui organise l'information en termes de faits et de dimensions (Kimball, 1996).

De manière générale, le processus d'entreposage se fait en deux temps. Tout d'abord, il faut créer les structures multidimensionnelles de l'ED. Ensuite, il faut extraire les données des sources pour alimenter l'ED. La première phase repose sur trois étapes principales, à savoir la modélisation conceptuelle, la modélisation logique et la modélisation physique. La modélisation conceptuelle vise à fournir une représentation commune des données provenant de diverses sources opérationnelles. Le schéma conceptuel est transformé en un schéma logique puis en un schéma physique spécifique à une plateforme donnée (Romero et Abelló, 2009). La seconde phase vise à décrire et à implanter les processus ETL. Pour ce faire, il est

également nécessaire de suivre un ensemble d'étapes de modélisation conceptuelle, logique et physique (Simitsis et Vassiliadis, 2003).

Les méthodes actuelles proposent des solutions pour concevoir et implanter soit le schéma multidimensionnel soit les processus d'alimentation, sans pour autant amalgamer les deux. En effet, à l'heure actuelle pour développer un projet d'entreposage complet (de création des structures multidimensionnelles et d'alimentation), le concepteur doit combiner au moins deux démarches. Le choix de ces démarches doit prendre en compte les problèmes d'intégration et d'interopérabilité. En outre, la plupart des méthodes existantes manquent de mécanismes pour générer automatiquement du code et/ou pour décrire tous les aspects du processus d'entreposage. Ce dernier s'avère coûteux et souvent voué à l'échec lorsqu'il est fait de manière manuelle (Kimball, 1996).

Afin de répondre aux limites des approches existantes, nous souhaitons proposer une démarche (i) *mixte*, (ii) *unifiée* et (iii) *semi-automatique* pour la conception et l'alimentation d'ED multidimensionnelles. Il s'agit d'une démarche de modélisation « mixte » puisque le schéma de l'entrepôt doit être construit à partir des besoins des décideurs et validé par rapport aux sources de données. Nous qualifions cette démarche d'« unifiée » car elle doit fournir un seul cadre pour la conception conjointe des ED et des processus ETL. Enfin, cette démarche est dite « semi-automatique » car elle doit proposer un ensemble de règles permettant la génération automatique des schémas logiques et physiques à partir du schéma conceptuel, permettant ainsi de réduire le temps et l'effort important requis dans un processus d'entreposage.

Dans ce contexte, il est reconnu que l'Architecture Dirigée par les Modèles (MDA : Model Driven Architecture), standard de l'OMG (Object Management Group), est un cadre qui gère la complexité, réduit considérablement les coûts de développement, améliore la qualité du logiciel et réalise des niveaux élevés de réutilisation (Bettin, 2003), (Kleppe et al., 2003), (Object Management Group, 2003). Ainsi, en utilisant MDA le processus d'entreposage nécessite moins d'efforts et de temps. MDA permet d'aboutir au code de l'application en partant des modèles conceptuels grâce à la définition d'une série de transformations. Par ailleurs, MDA offre un support à l'intégration, l'interopérabilité, l'adaptabilité, la portabilité et la réutilisation des systèmes d'informations (Kleppe et al., 2003). MDA est basée sur l'utilisation de modèles aux différentes phases du cycle de développement d'une application. En effet, MDA préconise l'élaboration de trois types de modèles : 1) Le modèle des exigences (Computation Independent Model : CIM) décrit les services que doit fournir l'application pour répondre aux besoins des utilisateurs, sans spécifier les détails de sa construction. 2) Le modèle d'analyse et de conception (Platform Independent Model : PIM) définit la structure et le comportement du système sans indiquer la plateforme d'exécution. 3) Le modèle de code ou de conception concrète (Platform Specific Model : PSM) est la projection d'un PIM sur une plateforme donnée (Blanc et Salvatori, 2005).

Le papier est organisé comme suit. En section, 2 nous présentons les travaux qui ont traité la modélisation des ED et des processus ETL. En section 3, nous présentons un aperçu de notre démarche. La section 4 décrit le modèle conceptuel unifié. La section 5 présente le modèle logique ainsi que les règles de transformation vers le modèle physique. En section 6 nous présentons l'implantation des règles de transformations. La section 7 conclut le papier et annonce les futurs travaux.

2 Etat de l'art

La conception et le développement d'ED a fait objet de nombreux travaux. Dans la littérature, elle a été abordée selon deux approches complémentaires mais différentes (Rizzi et al., 2006). La première approche aborde la conception du schéma multidimensionnel qui vise à déterminer la structure de l'entrepôt (Romero et Abelló, 2009). Alors que la seconde vise à modéliser les traitements ETL afin de représenter les processus responsables de l'alimentation et de la mise à jour de l'entrepôt (Vassiliadis, 2009). Dans cette section, nous présentons brièvement les travaux qui ont traités la modélisation et le développement des ED et/ou des processus ETL.

En ce qui concerne la modélisation multidimensionnelle, les approches existantes peuvent être classées en trois grandes catégories (Rizzi et al., 2006) . Les approches « descendantes » (Tsois et al., 2001), (Prat et al., 2006) proposent de construire le schéma de l'entrepôt à partir d'une analyse détaillée des besoins des décideurs. Alors que les approches « ascendantes » (Golfarelli et Rizzi, 1998), (Hüsemann et al., 2000) partent d'une analyse détaillée des sources de données afin de sélectionner les données pertinentes pour la prise de décision. Enfin, les approches « mixtes » (Zepeda et al., 2008), (Romero et Abelló, 2010), (Mazón et Trujillo, 2009), (Carmè et al., 2010), (Essaidi et Osmani, 2010) considèrent à la fois les besoins des décideurs et la disponibilité des données au niveau des sources. Elles présentent l'avantage de concevoir des schémas multidimensionnels valides par rapport aux sources de données existantes.

Il existe plusieurs manières pour décrire et implanter les processus ETL. Les industriels proposent de nombreux outils (Barateiro et Galhardas, 2005) tels que Microsoft Integration Services, Oracle Warehouse Builder, etc. De manière générale, les travaux de recherche proposent soit des modèles spécifiques soit d'utiliser et/ou d'étendre des standards existants tels que UML ou le BPMN (Business Process Model Notation). D'une part, (Vassiliadis et al., 2002) proposent un modèle conceptuel qui fournit des notations graphiques spécifiques permettant de décrire les problèmes techniques souvent rencontrés dans les processus ETL. (Simitsis et Vassiliadis, 2003) complètent ce modèle via une méthode qui sert à identifier les relations entre les attributs. (Simitsis et al., 2010) exploitent les technologies du web sémantique afin de faciliter le processus de sélection et de transformation des données pertinentes à partir des sources. D'autre part, d'autres travaux proposent d'utiliser ou d'étendre UML pour décrire le workflow ETL. (Trujillo et Luján-Mora, 2003) définissent un ensemble d'activités ETL via des classes stéréotypées. Ils utilisent les notes UML pour spécifier les relations de correspondances entre les attributs cible de l'entrepôt et les attributs sources. (Luján-Mora et al., 2004) étendent UML via un diagramme de correspondance qui explicite les relations de transformations des attributs sources en attributs cible de l'ED. (Muñoz et al., 2008) utilisent les diagrammes d'activités UML afin de concevoir les processus ETL. Enfin, (Muñoz et al., 2009) proposent une approche dirigée par les modèles pour générer les processus ETL. (El Akkaoui et Zimanyi, 2009) présentent un modèle conceptuel basé sur le BPMN.

Par rapport aux travaux existants et particulièrement (Mazón et Trujillo, 2008) (Pardillo et al., 2011), l'avantage de notre approche réside dans la description unifiée des structures multidimensionnelles et des processus d'alimentation. En effet, nous suivons la même démarche de conception que l'équipe de Trujillo qui est une démarche dirigée par les modèles cependant, plutôt que définir deux processus indépendants pour la modélisation de la structure multidimensionnelle d'un ED et pour la spécification des traitements ETL, nous

avons unifié dès les premières étapes ces deux processus. Cette solution permet de limiter des étapes redondantes et coûteuses de la double confrontation du schéma multidimensionnel et de ses processus ETL avec les sources de données. La confrontation des sources est effectuée dès la définition du CIM et évite ainsi la définition dans un ED multidimensionnel d'attributs dont il serait impossible de définir le processus d'alimentation et de rafraichissement. Ce modèle permet également d'éviter les problèmes d'incohérences, d'intégration et d'interopérabilité qui s'imposent si on utilise des modèles et/ou approches distincts. Par ailleurs, il permet de définir des concepts multidimensionnels (données et traitement) prêt à être réutilisés pour définir de nouveaux schémas d'entrepôts. Notre approche présente aussi l'avantage de réutiliser et d'adapter des modèles et langages existants, notamment avec les modèles en constellation (Golfarelli et al., 1998) affirmé pour la conception multidimensionnelle, et le langage de contrainte OCL (Object Constraint Language) standardisé par l'OMG (Object Management Group, 2010).

3 Démarche de modélisation dirigée par les modèles

Notre approche est abordée dans un cadre dirigée par les modèles. La figure 1 présente un aperçu de notre proposition. Partant d'une représentation formelle des besoins des décideurs au niveau du CIM la démarche permet d'aboutir aux modèles physiques spécifiques à différentes plateformes. Les principes de MDA simplifient la tâche du concepteur. Ce dernier construit un modèle des besoins (CIM) et les transformations automatiques le traduisent en une succession de modèles de façon à obtenir un modèle physique adapté à la plateforme choisie. Pour ce faire, d'abord, un premier modèle multidimensionnel (PIM_1 conceptuel) est dérivé à partir du CIM. Deux types de modèles multidimensionnels sont définis en fonction des préférences du concepteur. Le premier est basé sur UML, alors que le second est basé sur les modèles en constellation présenté dans (Golfarelli et al., 1998), (Ravat et al., 2008). Quel que soit son type, ce PIM est complété via des contraintes ETL-OCL, définies en se basant sur le langage OCL. Une contrainte ETL-OCL permet de décrire de manière conceptuelle la formule d'extraction de transformation (projection, conversion, sélection et agrégation) des attributs multidimensionnels. Par la suite, plusieurs modèles logiques ($PIMs_2$) (CWM[1]: relationnel, CWM :: XML, etc.) peuvent être générés à partir des modèles conceptuels. Par ailleurs, les expressions ETL-OCL sont traduites en un ensemble d'expressions en algèbre relationnelle. Ensuite, différents modèles physiques (PSM) (Oracle, Mondrian, etc.) peuvent être générés à partir des modèles logiques et conceptuels en fonction de la plateforme de déploiement choisie par le concepteur. Les expressions en algèbre relationnelles sont traduites à leur tour en un ensemble de modèles de requêtes SQL et/ou de procédure PL/SQL. Enfin, la démarche fournit le code de création et d'alimentation (ETL) des structures multidimensionnelles de l'ED. Notons que nous ne présentons pas les modèle de description de plateformes (PDM : Platform Description Model). En effet, la transformation entre modèles repose sur des méta-modèles, un PDM est implicitement représenté par les méta-modèles physiques (qui décrivent le modèle physique : PSM). Ces modèles ne sont pas nécessaires dans le cadre de notre contribution (Bézivin et Blanc, 2002).

[1] Common Warehouse Metamodel : http://www.omg.org/spec/CWM/

Les transformations entre les modèles sont formalisés au moyen du langage QVT (Query/View/Transformation) de telle sorte que le code final soit généré automatiquement (Object Management Group, 2009). Principalement, trois types de transformations de modèles sont définis. Les transformations « intra-niveau », graphiquement représentées par des cercles dans la figure 1, permettent de convertir un modèle source en un modèle cible dans le même niveau d'abstraction (par exemple du modèle logique CWM :: relationnel vers le modèle logique CWM :: XML). Alors que, les transformations « inter-niveaux » représentées par des triangles, permettent de transformer un modèle source en un modèle cible à deux niveaux d'abstraction différents (par exemple, un PIM conceptuel est transformé en un PIM logique). Enfin, lors de la « fusion de transformations » représentée par des losanges dans la figure 1, plusieurs modèles sources sont combinés en un modèle cible unique (par exemple les PIM conceptuel et logique sont transformés en un PSM). Les transformations « inter-niveaux » et la « fusion de transformations » sont responsables d'automatiser le processus d'entreposage, car elles permettent la génération automatique du code final à partir des besoins des décideurs. Alors que, les transformations « intra-niveaux » permettent d'améliorer le niveau d'intégration et de portabilité dans le système. En effet, elles permettent de déduire un PSM à partir d'un autre directement ou en exécutant les transformations « inter-niveaux ». La figure suivante représente les différentes étapes de notre approche.

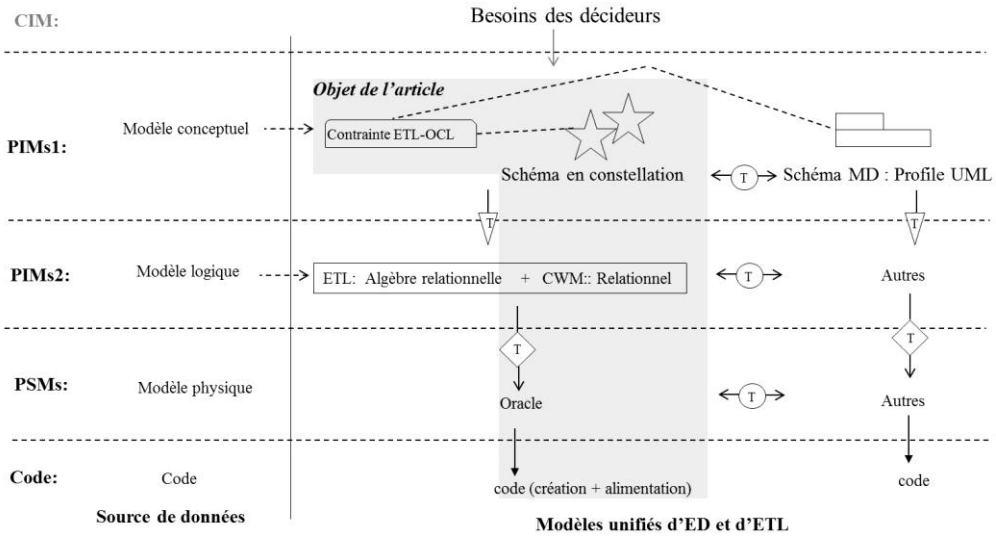

FIG. 1 – *Démarche dirigée par les modèles pour le développement d'ED et d'ETL.*

Dans les sections suivantes, nous présentons le modèle multidimensionnel unifié basé sur les modèles en constellation. Ensuite, nous présentons le modèle logique et nous détaillons les transformations vers le modèle physique. Le modèle conceptuel basé sur les profils UML ainsi que les transformations de ce modèle vers le modèle logique sont détaillés dans (Atigui et al., 2010) (Atigui et al., 2011).

4 Modèle multidimensionnel unifié

Dans cette section, nous présentons le modèle conceptuel qui décrit les structures multidimensionnelles et les processus ETL en se basant sur les modèles en constellation (Golfarelli et Rizzi, 1998) (Ravat et al., 2008). Des expressions formelles sont attachées à ce modèle afin de décrire les formules d'extraction des attributs multidimensionnels. Pour ce faire, nous proposons d'adapter le langage OCL.

4.1 Transformation d'attributs sources

Dans un processus ETL, les données extraites des sources subissent une suite de transformations avant d'être chargées dans l'ED. La modélisation conceptuelle des processus ETL vise à spécifier la relation de transformation entre les attributs multidimensionnels cibles et les attributs sources. En se basant sur les cardinalités des attributs et des valeurs sources, ces opérations peuvent être de trois types :

- La valeur d'un attribut cible est calculée à partir de la valeur d'un attribut source unique (opération identité) (DW.Produit.codeP = SR.Produit.codeP) ;
- La valeur d'un attribut cible est calculée à partir de plusieurs valeurs d'un attribut source unique (DW.Ventes.quantité = Sum(SR.LigneCde.quantité)) ;
- La valeur d'un attribut cible est calculée à partir des valeurs de deux ou plusieurs attributs sources (DW.Ventes.montant = Sum(SR.LigneCde.quantité) * SR.Produit.prixUnit).

Il est possible d'appliquer une opération de « sélection » dans ces trois cas : seules les valeurs qui répondent à un critère donné sont extraites (DW.Produit.codeP = SR.Produit.codeP (critère : SR.Produit.description = « TV »)).

Afin de décrire la structure multidimensionnelle et les processus d'alimentation de manière conjointe, il est indispensable de fournir un moyen pour spécifier ces différentes opérations de manière à compléter le schéma multidimensionnel. Notons qu'UML ne propose aucun moyen pour exprimer des relations entre attributs appartenant à des classes différentes du même modèle, voire à des modèles différents (Simitsis et Vassiliadis, 2003). Ainsi, on a besoin d'un langage qui permet de décrire les opérations de transformation de manière indépendante des plateformes et qui relève du niveau d'abstraction conceptuel. Ce langage doit être formel pour permettre une génération automatique du code final. Il doit également fournir un moyen pour exprimer des relations de « dérivation » entre attributs. En outre, ce langage doit fournir un moyen pour naviguer entre les différentes concepts (classes, fait, dimensions, etc.) et d'accéder aux leurs attributs. Enfin, il doit permettre d'exprimer des opérations mathématiques (somme, produit, etc.) ainsi que des opérations de sélection et d'agrégation d'attributs.

Le langage OCL répond en grande partie à ces besoins. En effet, il permet d'exprimer des contraintes et des requêtes sur un modèle de manière précise et indépendante des plateformes. Contrairement aux notes UML, OCL est un langage formel qui peut être traduit de manière automatique vers les niveaux logique et physique.

4.2 Formalisation des opérations de transformation en ETL-OCL

Le langage OCL est formel, précis, simple, supporté par de nombreux outils et ne nécessite pas une expertise humaine importante (Warmer et Kleppe, 2003). Ce langage vise à raffiner les diagrammes UML en particulier les diagrammes de classes. Ce langage a été étendu pour exprimer des contraintes et des requêtes sur d'autres types de modèles, notamment, les modèles multidimensionnels. Ces extensions essentiellement orientées définition des contraintes (Bejaoui et al., 2010) ou interrogation (Pardillo et al., 2010) ne répondent pas à notre objectif d'extraction de données sources pour alimenter un ED.

En modélisation multidimensionnelle, chaque attribut de l'ED est dérivé à partir d'un ou plusieurs attribut(s) source(s). En conséquence, il est possible d'exploiter OCL pour exprimer des contraintes qui mettent en relation 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 (classes, 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 classes d'association s'il s'agit d'un diagramme de classes, les entités et les associations s'il s'agit d'un modèle E/A, etc.) dans chaque source. Le choix des concepts sources dépend des formules de transformation des attributs multidimensionnels. Pour chaque source de données, le concept choisi est celui qui a le plus d'attributs sources invoqués par les formules d'extraction de tous les attributs du concept multidimensionnel. Il est possible d'accéder à tous les attributs invoqués par la formule d'extraction en parcourant les liens.

Une expression OCL est définie dans un contexte (une classe, un attribut ou une opération dans un diagramme de classe). 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 suivant définit les principaux mots clés OCL utilisés pour définir une expression ETL-OCL.

Mot clé	Description
Context	Définit 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 (expression booléenne)	Permet de sélectionner un sous-ensemble qui répond à un critère donné.

TAB. 1– *Mots clés (ETL-) OCL.*

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 chaines de caractères (concat(), size(), substring(), toInteger(), toUpperCase(), etc.) ainsi que des fonctions mathématiques appliqués en particulier sur les attributs numériques (min(), max(), product(), sum(), etc.). OCL est assez riche pour permettre une description conceptuelle de la plupart des mécanismes ETL. Cependant, il ne fournit aucun moyen pour l'agrégation des données, fonction souvent utilisée pour transformer les données sources. Pour compléter les expressions ETL-OCL, nous définissons l'opération d'agrégation comme suit :

Définition. **AGG ($A_{SI} \rightarrow$ AF ; A_{SJ} [$\rightarrow$ select (P_{ASJ})]),** où :

- **A_{SI} :** attribut agrégé ;
- **AF :** fonction d'agrégation prédéfinie dans la bibliothèque OCL (sum, count, min, max, avg (sum / size)) ;
- **A_{SJ} :** attributs servant à définir le critère de regroupement ;
- **P_{ASJ}** : prédicat de sélection appliqué aux attributs de regroupement.

En fait, les expressions ETL-OCL permettent de formuler des extractions :

- Simples.
- Conversion :
  - Conversions arithmétiques (somme, division, multiplication, etc.) permettant de dériver les valeurs des attributs calculables et/ou de transformer leurs formats (conversion de devises par exemple).
  - Conversions de chaines de caractères (concaténation, substitution, taille, découpage, etc.).
  - Conversions de type (*AttributMD.oclAsType (nouveau type)*) : un attribut ne peut être ré-typé qu'en un type auquel il est conforme (un entier peut être transformé en un réel par exemple), sinon l'expression ETL-OCL est évalué comme invalide.

- Sélection d'attributs qui répondent à un ou plusieurs critère(s) donné(s) en utilisant la bibliothèque OCL, notamment les opérateurs : *select, reject, collect, exists, forAll*, etc.
- Agrégation : permet l'agrégation des données sources en fonction d'un ensemble de critères ou d'attributs d'agrégation en utilisant la fonction « AGG » définie ci-dessus.

Par ailleurs, il est possible de combiner plusieurs fonctions pour exprimer les formules d'extraction les plus complexes, notamment les mesures. Le concepteur peut également définir ses propres fonctions d'extractions et de transformations afin de gérer les cas particulier de son application.

Exemple. La figure suivante présente le schéma multidimensionnel (en haut de la figure) permettant l'analyse des montants et des quantités des ventes en fonction des produits, des clients des pays européens du bassin méditerranéen et des dates. En bas de la figure le diagramme de classes présente un exemple de source de données. Chaque concept cible (fait ou dimension) est relié à un concept de la source (classe ou classe d'association). D'un point de vue graphique, ces liens sont représentés par des lignes discontinues.

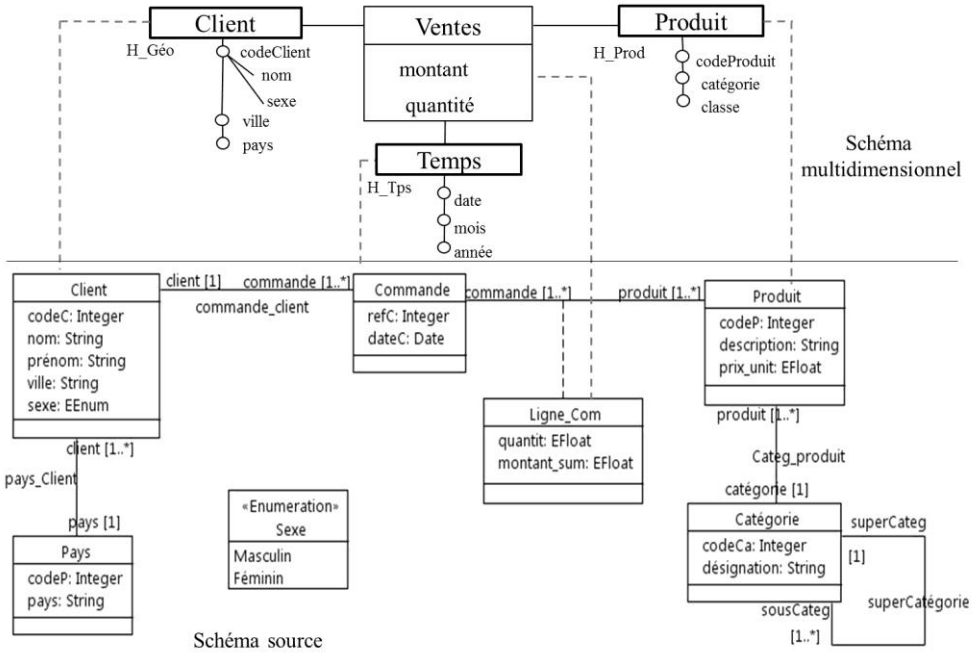

FIG. 2 – *PIM multidimensionnel (analyse des montants et des quantités des ventes selon les clients, les produits et les dates).*

Les expressions ETL-OCL définies sur les attributs de la dimension « Client » et les mesures du fait « Ventes » sont les suivantes :

Context Client :: codeClient : Integer *derive : SR.client.codeC*	-- Le paramètre codeClient est égal à l'attribut codeC de la classe source Client
Context Client :: nom : String *derive : SR.client.nom.concat('.').concat (SR.client.prénom)*	-- L'attribut faible « nomClient » est égal à la concaténation : « nom.prénom »
Context Client :: sexe : String *derive : If SR.client.sexe = 'masculin' then 'H'* *else 'F' endif*	-- L'attribut faible sexe vaut « H » si l'attribut source de la classe « Client » est égal « masculin » et vaut « F » sinon.
Context Client :: ville : String *derive : SR.client.ville*	-- Le paramètre ville est égal à l'attribut ville de la classe Ville.
Context Client :: pays : String *derive : SR.client.pays→ select(pays = 'France' or pays = 'Italie' or pays = 'Espagne' or pays = 'Portugal' or pays = 'Grèce').*	-- Le paramètre pays est égal à l'attribut pays de la classe « pays » de la classe « Pays ». Seul les pays européens du bassin méditerranéen sont sélectionnés. La navigation à partir de la classe « Client » se fait en suivant les rôles.
Context Ventes ::quantité : Real *derive : AGG (SR.ligne_Com.quantité→ sum() ; SR.ligne_Com.produit.codeP, SR.ligne_Com.commande.client.codeC, SR.ligne_Com.commande.dateC)*	-- La mesure quantité est égale à la somme des quantités de la classe d'association source « Ligne_Com » calculée par code produit, code client et date.
Context Ventes ::montant : Real *derive : AGG ((SR.ligne_Com.quantité *SR.ligne_Com.produit.prixUnit) → sum() ; SR.ligne_Com.produit.codeP, SR.ligne_Com.commande.client.codeP, SR.ligne_Com.commande.dateC)*	-- La mesure montant est égale à la somme des quantités multipliée par le prix unitaire du produit, l'agrégation se fait par code produit, code client et date.

5 Génération automatique des modèles logiques et physiques

Dans cette section, nous présentons les règles de transformations du modèle conceptuel (PIM_1) en modèle logique, puis en modèle physique. Dans ce qui suit nous nous intéressons uniquement sur la transformation des structures multidimensionnelles (faits et dimensions).

5.1 PIMs logiques

Les modèles logiques sont générés automatiquement à partir des modèles conceptuels en appliquant un ensemble de règles. Afin de couvrir autant que possible d'applications

d'entreposage, notre approche fournit un ensemble varié de modèles logiques de l'ED. Le concepteur peut choisir le modèle le mieux adapté à l'application qu'il a à développer, à savoir le ROLAP[2] normalisé, dénormalisé ou optimisé (R-OLAP avec définition des treillis de données) ou encore des schémas XML, etc. Les règles de transformation du modèle conceptuel vers les modèles logiques ROLAP normalisé et ROLAP dénormalisé ainsi que leur formalisation en QVT ont été détaillées dans (Atigui et al., 2010).

5.2 PSM

Dans cette section, nous présentons comment obtenir le modèle physique (PSM) à partir des PIMs conceptuel et logique. Les règles de transformation sont formalisées en QVT (Object Management Group, 2009). Les modèles physiques sont dépendants d'une plateforme spécifique (Oracle, Mondrian, etc.). Afin d'illustrer notre démarche, nous choisissons de détailler les règles de transformations 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 effectuées automatiquement par le SGBD. Dans cette section, nous présentons les règles de transformation modèle-à-modèle permettant de générer la structure du modèle physique.

5.2.1 Règles de transformation des PIMs en PSM

Le schéma physique Oracle est composé de vues matérialisées et de dimensions. La définition des vues matérialisées dépend des tables logiques ROLAP et des expressions ETL-OCL, alors que la définition des dimensions dépend des dimensions du schéma conceptuel. Par conséquent, le modèle physique est généré en fusionnant les deux schémas conceptuel (schéma en constellation) et logique (schéma ROLAP dénormalisé) en appliquant les règles suivantes :

1. Un schéma ROLAP est transformé en un schéma Oracle.
2. Chaque table de dimension du schéma ROLAP est transformée en une vue matérialisée. Les colonnes et la clé primaire de la table correspondent respectivement aux colonnes et à la clé primaire de la vue matérialisée.
3. Chaque table de faits est transformée en une vue matérialisée. Les colonnes de la table sont transformées en colonnes de la nouvelle vue. La clé primaire et les clés étrangères sont transformées en clé primaire et clés étrangères référençant la vue matérialisée appropriée.
4. Chaque dimension du modèle conceptuel 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.

5.2.2 Formalisation des règles de transformation en QVT

Nous avons formalisé ces règles en utilisant la syntaxe graphique de QVT. Une transformation QVT entre deux modèles candidats est décrite par un ensemble de relations. Ces dernières sont définies par deux ou plusieurs domaines qui spécifient un modèle candidat

[2] Relational On Line Analytical Processing.

et un ensemble d'éléments à transformer ainsi qu'un ensemble de pré-conditions « When » et de post-conditions « Where ».

Nous présentons la relation « main » (1) et les relations de transformation des tables dimensions (2) et des dimensions conceptuelles (3).

Relation « Main ». Cette relation est le point d'entrée au processus de transformation. Elle permet de fusionner deux modèles sources en un modèle cible. La partie gauche de la figure suivante montre deux domaines différents, le premier fait partie du PIM logique (drolap : ROLAP), le second domaine fait partie du modèle conceptuel PIM (md : MD). La partie droite de la figure explicite le modèle 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 de la constellation sont transformées en dimensions Oracle.

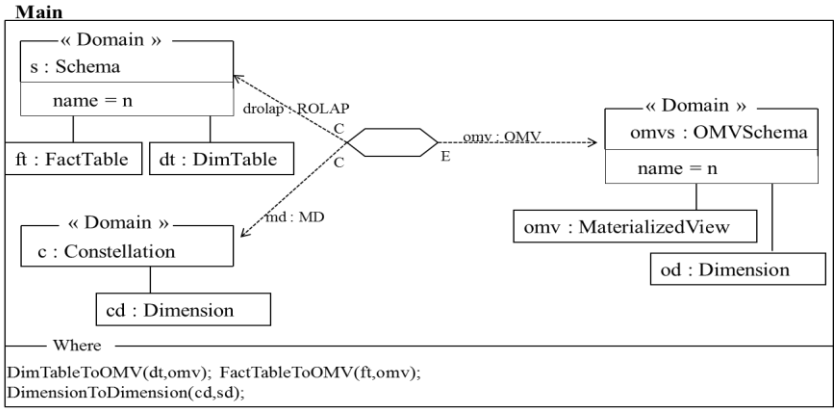

FIG. 3 – *Relation « Main »*

Relation « Table dimension en vue matérialisée ». Chaque table de dimension du modèle ROLAP est transformée en une vue matérialisée ayant le même nom. La clause «Where» identifie les relations de transformations des colonnes et des clés primaires.

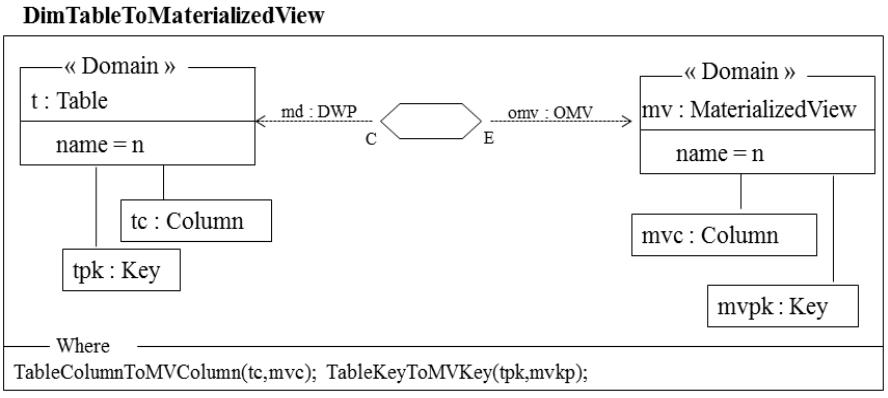

FIG. 4 – *Relation table dimension en vue matérialisée*

Relation « Dimension multidimensionnelle en dimension Oracle». Cette relation permet detransformer chaque dimension conceptuelle en une dimension Oracle ayant le même nompréfixé par « _dim ». La clause « When » permet de préciser que les tables ROLAP appropriées doivent être déjà transformée en vue matérialisée. Alors que, la clause « Where » permet de préciser que les hiérarchies, les paramètres et attributs faibles sont repectivement convertis en hiérarchies, niveaux et attributs Oracle.

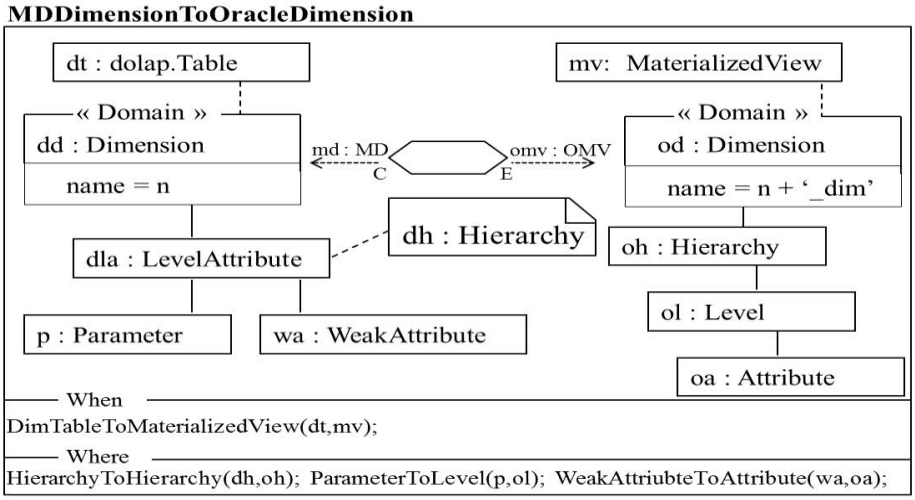

FIG. 5 – *Relation dimension multidimensionnelle en dimension Oracle*

6 Implantation

Les règles de transformation inter et intra modèles ainsi que de fusion de modèles ont été implantées en utilisant Medini-QVT[3]. Ce dernier est un plug-in Eclipse[4] qui permet d'implanter les règles de transformation modèles-à-modèles en utilisant la syntaxe textuelle de QVT.

La figure 6 présente un extrait du code QVT qui implémente les règles de fusion des PIMs conceptuel et logique en un PSM Oracle. Elle présente également les règles de transformation des tables ROLAP en une vue matérialisée et des dimensions de la constellation en dimension Oracle. Afin de transformer le schéma en constellation des « Ventes » en un schéma physique, les métamodèles sources et cibles doivent être décrit dans des fichiers « Ecore ». L'exécution du code QVT génère le schéma physique correspondant.

[3] http://projects.ikv.de/qvt
[4] http://www.eclipse.org/

7 Conclusion

Dans cet article nous avons présenté une approche dirigée par les modèles pour la conception et le développement conjoint des ED et des processus ETL. L'utilisation d'un cadre MDA permet de générer les modèles logique et physique de manière automatique. En effet, chaque phase de modélisation (analyse des exigences, modélisation conceptuelle, logique et physique) est décrite par un ou plusieurs modèles. La démarche applique une série de transformations en vue de générer automatiquement le code final. Nous avons présenté le modèle conceptuel unifié qui réutilise et adapte les schémas en constellation et le langage OCL. Ensuite, nous avons détaillé les règles QVT permettant de transformer les PIMs conceptuel et logiques en PSM. Par ailleurs, nous avons implanté les transformations modèle-à-modèle formalisées en syntaxe textuelle de QVT en utilisant MediniQVT.

Dans les futurs travaux, nous envisageons, dans un premier temps, de travailler les phases amont aux processus présentés dans cet article. Plus particulièrement nous aborderons la formalisation des exigences (CIM) et de transition entre le CIM et le PIM conceptuel en utilisant un ensemble de règles QVT. Dans un second temps, nous aborderons un ensemble d'extensions relatif aux processus présenté dans cet article. Notamment, nous envisageons de compléter la démarche et de fournir les règles de transformation des contraintes ETL-OCL vers les modèles logique et physique. Nous traiterons la prise en compte de plusieurs sources en adaptant les processus ETL avec des liens inter bases de données, des espaces de stockage intermédiaires pour écrire les requêtes de traitement. Nous avons également l'intention de développer notre méthode en considérant d'autres plates-formes de déploiement. En outre, nous envisageons d'appliquer notre approche sur des cas d'études du monde réel et de l'évaluer par différents utilisateurs (étudiants en master ou consultants dans une société de service).

```
transformation MDPIManddROLAP2OMV(md:Multidimensional, drolap:ROLAP, omv:OracleMV) {
    -- mapping a ROLAP schema to Oracle schema
    top relation main {
      sn : String;
        checkonly domain md rs : ROLAP::Schema {
        name = sn };
      enforce domain omv os : OracleMV::Schema {
        name = sn };
    }
-- mapping ROLAP dimension table into Oracle MV
    top relation DimTableToMaterializedView {
      dtn : String;
      checkonly domain drolap t : ROLAP::Table {
        column = co : ROLAP::Column { },
        pKey = k : ROLAP::Key {},
        name = dtn };
enforce domain omv o : OracleMV::MaterializedView {
        column = mvc : OracleMV::Column { },
        pKey = mvpk : OracleMV::Key { },
        name = dtn + '_dim'  };
      where {
        TableColumnToMVColumn(co,mvc);
        TableKeyToMVKey (k,mvpk);
       }}
--mapping a MD dimension into oracle dimension
    top relation MDDimensionToOracleDimension {
      mdn : String;
      dt:ROLAP::Table;
checkonly domain md d:Multidimensional::Dimension {
ownedHierarchy=h:Multidimensional::Hierarchy {},
ownedParameter =p:Multidimensional::Parameter{},
ownedWeakAttribute=wa:Multidimensional::WeakAttribute{},
       mdName = mdn };
    enforce domain omv od:OracleMV::Dimension {
        hierarchies = oh : OracleMV::Hierarchy { },
        levels = ol : OracleMV::Level { },
        attributes = oa : OracleMV::Attribute {  },
         dimMV = dmv : OracleMV::MaterializedView {  },
        name = mdn  };
       when {
        DimTableToMaterializedView(dt,dmv); }
      where {
        MDHierarchyToOHierarchy(h,oh);
        ParameterToLevel(p,ol);
        WeakAttributeToAttribute(wa,oa); }}
```

Règle Main (figure 3)

Règle de transformation des tables de dimension en vue matérialisée (figure 4)

Règle de transformation des dimensions en dimension Oracle (figure 5)

FIG. 6 – *Implantation des règles de transformation (QVT textuel)*

Références

Atigui, F., F. Ravat, O. Teste et G. Zurfluh (2010). *Démarche dirigée par les modèles pour la conception d'entrepôts de données multidimensionnelles*. 26ème journées des Bases de Données Avancées, Toulouse.

Atigui, F., F. Ravat, R. Tournier et G. Zurfluh (2011). *A unified model driven methodology for Data Warehouses and ETL design*. (A paraître) 13th International Conference on Enterprise Information Systems, Beijing, China.

Barateiro, J. et H. Galhardas (2005). *A survey of data quality tools.* Datenbank-Spektrum.

14, 48.

Bejaoui, L., F. Pinet, M. Schneider et Y. Bédard (2010). *OCL for formal modelling of topological constraints involving regions with broad boundaries*. Geoinformatica 14, p.353-378.

Bettin, J. (2003). *Model-Driven Architecture Implementation & Metrics*. SoftMetaWare, Ltd., Version 1.

Bézivin, J. et X. Blanc (2002). *Promesses et interrogations de l'approche MDA*. Développeur Référence, Septembre.

Blanc, X. et O. Salvatori (2005). *MDA en action: Ingénierie logicielle guidée par les modèles*. Eyrolles éditions, Paris.

Carmè, A., J. N. Mazón, et S. Rizzi (2010). *A Model-Driven Heuristic Approach for Detecting Multidimensional Facts in Relational Data Sources*. Data Warehousing and Knowledge Discovery, p.13–24.

El Akkaoui, Z. et E. Zimanyi (2009). *Defining ETL worfklows using BPMN and BPEL*. 12th international workshop on Data warehousing and OLAP. Hong Kong, China, p. 41-48.

Essaidi, M. et A. Osmani (2010). *Model driven data warehouse using MDA and 2TUP*. Journal of Computational Methods in Science and Engineering, 10, p.119-134.

Golfarelli, M., D. Maio et S. Rizzi (1998). *The dimensional fact model: a conceptual model for data warehouses*. International Journal of Cooperative Information Systems, 7, Issues 2, p. 215-247.

Golfarelli, M. et S. Rizzi (1998). *A methodological framework for data warehouse design*. 1st international workshop on Data warehousing and OLAP. Bethesda, Maryland, USA. p.3-9

Hüsemann, B., J. Lechtenbörger et G. Vossen (2000). *Conceptual data warehouse design*. 2nd International Workshop on Design and Management of Data Warehouses. Stockholm, Sweden. p. 3-9.

Kimball, R. (1996). *The data warehouse toolkit: practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc. New York, NY, USA.

Kleppe, A.G., J. Warmer et W. Bast (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Luján-Mora, S., P. Vassiliadis et J.Trujillo (2004). *Data mapping diagrams for data warehouse design with UML*. 23rd International Conference on Conceptual Modeling, p. 191-204.Shanghai, China.

Mazón, J. N. et J.Trujillo (2008). *An MDA approach for the development of data warehouses*. Decision Support Systems, 45. p.41-58.

Mazón, J. N. et J. Trujillo (2009). *A hybrid model driven development framework for the multidimensional modeling of data warehouses*. SIGMOD Record 38, p 12-17.

Muñoz, L., J.N Mazón, J. Pardillo et J. Trujillo (2008). *Modelling ETL processes of data warehouses with uml activity diagrams*. On the Move to Meaningful Internet Systems Workshops. p. 44-53.

Muñoz, L., J N. Mazón et J. Trujillo (2009). *Automatic generation of ETL processes from conceptual models.* 12th international workshop on Data warehousing and OLAP. Hong Kong, China, p.33.

Object Management Group (2003). *Model Driven Architecture* Version 1.0.1.

Object Management Group (2009). *Meta Object Facility* (MOF) 2.0 Query/View/Transformation Specification Version 1.1.

Object Management Group (2010). *Object Constraint Language,* Version 2.2.

Pardillo, J., J. N Mazón et J. Trujillo (2011). *An MDA Approach and QVT Transformations for the Integrated Development of Goal-Oriented Data Warehouses and Data Marts*. Journal of Database Management, 22, p. 43-68.

Pardillo, J., J. N Mazón et Trujillo (2010). *Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses*. Inf. Sci. 180, p. 584-601.

Prat, N., J. Akoka et I. Comyn-Wattiau (2006). *A UML-based data warehouse design method*. Decision Support Systems 42, p.1449-1473.

Ravat, F., O. Teste, R. Tournier et G. Zurfluh (2008). *Algebraic and graphic languages for OLAP manipulations*. International Journal of Data Warehousing and Mining 4, p.17- 46.

Rizzi, S., Abelló, A., J. Lechtenbörger et J. Trujillo (2006). *Research in data warehouse modeling and design dead or alive?* 9th international workshop on Data warehousing and OLAP. Arlington, Virginia, USA, p. 3.

Romero, O. et A. Abelló (2009). *A survey of multidimensional modeling methodologies*. International Journal of Data Warehousing and Mining 5, p.1-23.

Romero, O. et A. Abelló (2010). *Automatic validation of requirements to support multidimensional design*. Data & Knowledge Engineering, 69 (9), p.917-942.

Simitsis, A., D. Skoutas, M. Castellanos (2010). *Representation of conceptual ETL designs in natural language using Semantic Web technology*. Data & Knowledge Engineering 69 (9), p. 96-115.

Simitsis, A. et P. Vassiliadis (2003). *A methodology for the conceptual modeling of ETL processes*. Decision Systems Engineering, Velden, Austria.

Trujillo, J. et S. Luján-Mora (2003). *A UML based approach for modeling ETL processes in data warehouses*. Conceptual Modeling-ER, 2813-2003, p. 307-320.

Tsois, A., N Karayannidis et T. Sellis (2001). *MAC: Conceptual data modeling for OLAP*. International Workshop on DMDW. p. 28-55.

Vassiliadis, P. (2009). *A Survey of Extract-Transform-Load Technology*. International Journal of Data Warehousing and Mining 5, p. 1-27.

Vassiliadis, P., A., Simitsis et S. Skiadopoulos (2002). *Conceptual modeling for ETL processes*. 5th international workshop on Data Warehousing and OLAP. p. 14–21.

Warmer, J. et A. Kleppe (2003). *The object constraint language: getting your models ready for MDA*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Zepeda, L., M. Celma et R. Zatarain, R. (2008). *A mixed approach for data warehouse conceptual design with MDA*. Computational Science and Its Applications–ICCSA. p. 1204-1217.

Summary

During the last few years, several frameworks have dealt with Data Warehousing (DW) design issues. Most of these frameworks provide partial answers that focus either on multidimensional (MD) modelling or on Extraction-Transformation-Loading (ETL) modelling. However, less attention has been given neither to unifying both modelling issues into a single structured framework nor to automating the warehousing process. To overcome these limits, this paper provides a generic unified and semi-automated method that integrates DW and ETL processes design. The framework is handled within the Model Driven Architecture (MDA). It (i) first helps the designer in modelling the decision-makers requirements and then (ii) generates the MD model as well as (ii) the logical and the physical models and finally (iv) generates the code. In this approach, the transformation rules are formalized using the Query/View/Transformation (QVT) language.