Modélisation des transformations pour l'évolution de modèles multidimensionnels

Saïd Taktak*, Jamel Feki*, Gilles Zurfluh**

*Université de Sfax, FSEGS, Laboratoire Miracl, 3018 Sfax, P.O. Box 1088, Tunisie
{Said.taktak, Jamel.feki}@fsegs.rnu.tn,
**Université de Toulouse 1 Capitole, IRIT, Toulouse, France
Gilles.zurfluh@ut-capitole.fr

Résumé. La modélisation et l'entreposage des données ont constitué, depuis plus d'une décennie, une problématique de recherche pour laquelle différentes approches ont été proposées. Ces approches se focalisent sur des aspects statiques de l'entrepôt de données. Or, l'évolution du système d'information qui alimente un entrepôt peut avoir un impact sur ce dernier et peut conduire, par conséquent, à l'évolution de son modèle multidimensionnel. Dans ce contexte évolutif, nous proposons une démarche dirigée par les modèles pour automatiser la propagation de l'évolution du modèle de la source de données relationnelle vers l'entrepôt. Cette démarche est fondée sur deux modèles d'évolution ainsi qu'un ensemble de règles de transformation formalisées en Query/View/Transformation. Nous développons un prototype logiciel nommé DWE (« Data Warehouse Evolution ») qui supporte cette démarche.

1 Introduction

L'entrepôt de données (ED) est caractérisé par une architecture complexe où les données issues des sources transactionnelles sont extraites, transformées, nettoyées et finalement chargées dans les tables de faits et de dimensions (Inmon, 2002).

Les méthodes de conception d'ED proposées depuis plus d'une décennie ainsi que la plupart des technologies d'entreposage disponibles sur le marché du décisionnel présument que le modèle conceptuel de l'ED est figé. Toutefois, en pratique, l'ED est caractérisé par une dynamique qui touche non seulement les données stockées mais aussi leurs structures. En fait, il est difficile de déterminer d'une manière définitive le modèle d'un ED dès sa phase de conception, et il est souvent indispensable de le modifier après sa mise en place. En effet, l'évolution au cours du temps des processus métier du système opérationnel de l'entreprise peut entraîner une évolution du schéma de la source (SD). L'ED dépendant de cette source ne peut résister à cette évolution qui peut affecter inévitablement son schéma, son processus ETL de chargement ainsi que les données entreposées.

Dans la littérature des systèmes d'information décisionnels, l'évolution des entrepôts dirigée par le modèle de données de la source d'alimentation et dans le contexte MDA (« MDA : Model Driven Architecture ») a été peu prise en compte. Cet article traite cette problématique. Plus précisément, nous nous intéressons à l'étude de l'impact de l'évolution du modèle de la

source relationnelle sur le modèle d'un ED multidimensionnel. Cette évolution devrait être répercutée rapidement, facilement et efficacement pour garantir une continuité des services redus par l'entrepôt. Pour atteindre cet objectif, nous adopterons l'approche dirigée par les modèles.

Ce papier est organisé comme suit. Dans la section 2, nous faisons un tour d'horizon des travaux relatifs au problème d'évolution des EDs. La section 3 décrit notre approche basée sur MDA pour la propagation de l'évolution du modèle de la source vers l'entrepôt multidimensionnel. La section 4 définit la transformation de modèle permettant le passage automatique entre le modèle d'évolution de la source vers le modèle de l'entrepôt. La section 5 présente le prototype supportant notre démarche. Enfin, la section 6 conclut l'article et donne un aperçu des perspectives à court terme.

2 Etat de l'art

L'évolution de l'ED est une problématique qui a intéressé certains chercheurs. Les travaux ont abordé différents aspects touchant notamment : (i) l'évolution des modèles multidimensionnels, (ii) la maintenance des vues matérialisées, et (iii) l'adaptation du processus de chargement ETL (« Extract, Transform and Load »). Nous passons en revue ces travaux afin de nous positionner par rapport à leur contribution et tout en focalisant sur l'évolution du modèle de données de l'entrepôt.

2.1 Évolution du modèle de l'entrepôt

Nous distinguons deux courants qui ont étudié les problèmes d'évolution des modèles multidimensionnels : les travaux intéressés par la *mise à jour du modèle*, et les travaux de modélisation *temporelle/versionnement*.

2.1.1 Mise à jour du modèle

Les travaux menés dans cette catégorie adoptent l'hypothèse du schéma unique, c'est-à-dire que le schéma ne possède qu'une seule version à tout instant : c'est la version courante sous laquelle sont stockées toutes les données du système d'information (SI). En conséquence, toute évolution du schéma se traduit par la modification du schéma courant qui passe à une nouvelle version. Ce passage est accompagné par l'adaptation des instances afin de les faire migrer de l'ancienne version vers la nouvelle version de schéma.

Hurtado et al. (1999) ont traité le problème d'évolution des schémas multidimensionnels. Ils ont proposé des opérateurs d'évolution spécifiques au modèle multidimensionnel et ont étudié leurs effets sur le modèle ainsi que sur ses instances. Ils ont également traité l'effet de ces mises à jour sur les vues matérialisées et ont proposé un algorithme pour la maintenance de ces vues. Pour ce faire, ils ont défini un ensemble d'opérateurs d'évolution permettant la mise à jour des dimensions et de leurs hiérarchies, comme par exemple l'ajout et la suppression. A notre connaissance, l'approche proposée n'a pas été implantée.

Blaschka et al. (1999) se sont intéressés à l'évolution touchant les tables de dimension et de fait. De plus, ils ont enrichi les opérations d'évolution proposées par (Hurtado et al., 1999) en permettant d'insérer un nouveau niveau de hiérarchie à n'importe quelle position. Les auteurs ont développé l'outil *FIESTA* pour valider leur approche et l'ont présenté comme un

environnement permettant de gérer les évolutions des schémas multidimensionnels. *FIESTA* présente une méthodologie pour l'évolution des schémas multidimensionnels permettant le passage du niveau conceptuel au niveau logique. Les opérations d'évolution supportées sont dédiées au schéma en étoile.

Benitez-Guerrero et al. (2004) ont proposé un prototype appelé *WHES* (« WareHouse Evolution System ») qui permet la création et l'évolution des entrepôts de données en se basant sur un modèle d'évolution et le langage MDL (« Multidimensional Data Language »). Les opérations d'évolution proposées sont inspirées des travaux des auteurs de (Hurtado et al., 1999) avec l'avantage d'assurer la consistance du schéma modifié suite à des évolutions successives.

Favre et al. (2007) ont associé l'évolution de l'ED à une évolution des besoins d'analyse des décideurs. Les auteurs ont proposé une approche qui permet aux utilisateurs d'intégrer leurs propres connaissances afin d'enrichir les possibilités d'analyse de l'entrepôt en faisant évoluer son schéma. Les opérations d'évolution suggérées touchent principalement les dimensions et leurs hiérarchies. Pour valider leur approche, les auteurs ont développé une plateforme nommée *WEDriK* (« Warehouse Evolution Driven by Knowledge »).

D'autre part, Talwar et Gosain (2012) se sont intéressés principalement à l'évolution au niveau des hiérarchies et ont défini un ensemble d'opérations d'évolution et de contraintes qui doivent être satisfaites pour assurer l'intégrité des données et la cohérence du schéma. Les opérations et les contraintes ont été définies via ULD (« Uni-Level Description language ») et MDD (« Multilevel Dictionary Definition »).

2.1.2 Modélisation temporelle et versionnement

Le deuxième courant de travaux d'évolution repose soit sur des extensions temporelles qui intègrent explicitement la dimension temporelle au schéma du SI soit sur l'historique des différentes versions de schémas dans des espaces de stockage physiques différents. Contrairement au principe d'évolution du schéma, le versionnement de schéma a pour but de garder trace des changements apportés aux différentes versions du schéma d'une base de données. Plusieurs auteurs ont proposé la modélisation temporelle pour résoudre le problème d'évolution de l'ED.

Dans (Golfarelli et al., 2006), il a été défini un modèle conceptuel graphique pour représenter le modèle de base de l'ED ainsi qu'un ensemble d'opérations d'évolution pour la création de nouvelles versions. L'approche proposée permet la gestion des différentes versions de l'ED et assure leur interrogation à travers des requêtes de type « cross-version ».

Dans (Wrembel et Bebel, 2007), un modèle formel pour un ED multi-version est présenté avec un ensemble d'opérations d'évolution affectant le schéma et les instances d'un ED. Les auteurs font la distinction entre deux types de version d'ED : version réelle et version alternative. La version réelle est créée pour tenir compte des changements dans l'environnement réel de l'entreprise. La version alternative vise à assurer la simulation du processus de changement en se basant sur l'analyse « What-If ». Ils ont développé un outil nommé *MVDW* permettant la maintenance de l'ED et la gestion de ses versions.

Les auteurs de (Solodovnikova, 2008) ont proposé un outil pour l'évolution de l'ED, supportant la création et la manipulation de plusieurs versions ainsi que la construction et l'exécution des rapports associés. Les auteurs ont également défini une représentation physique des schémas de versions dans la source de données ainsi qu'une représentation logique de l'ED.

Ils ont situé les modifications susceptibles d'affecter un ED sur les trois niveaux : physique, logique et sémantique.

2.2 Maintenance des vues

La deuxième classe se base sur l'hypothèse qu'un ED est un ensemble de vues matérialisées construites directement à partir de sources de données. Il en résulte que toute modification au niveau de la source nécessite des opérations de maintenance de ces vues.

Dans (Rundensteiner et al., 1997), les auteurs développent un prototype nommé *EVE* (« Evolvable View Environment ») permettant d'automatiser la réécriture des définitions des vues pour assurer leur cohérence avec les changements structuraux réalisés sur des sources. L'outil *EVE* comprend deux modules de base : le premier permet à l'utilisateur d'évoluer les vues à travers un langage étendu de SQL nommé *Evolvable SQL* (E-SQL) et le deuxième module est dédié à la description des informations relatives aux changements au niveau source de données.

Dans (Bellahsene, 2002) l'auteur suggère une approche pour l'adaptation dynamique des vues matérialisés suite à des changements des sources de données. Cette approche concerne non seulement la maintenance de schémas mais aussi des données de l'entrepôt. L'idée principale de ce travail est d'éviter le re-calcul des vues après chaque modification, et de déduire le schéma de la nouvelle vue à partir de l'ancienne.

Les détails sur les travaux relatifs à la maintenance des vues matérialisées dans le contexte des ED sont disponibles dans l'article de (Jain et Gosain, 2012).

2.3 Adaptation du processus de chargement ETL

Plusieurs chercheurs se sont intéressés à la modélisation du processus de chargement (Atigui et al., 2012) mais très peu de travaux ont proposé des solutions pour l'adaptation de l'ETL lors de l'évolution de l'ED ou des sources de données. Parmi ces travaux nous citons (Papastefanatos et al., 2010). Les auteurs traitent le problème d'incohérences qui peuvent apparaître dans le processus de chargement de données ETL (Extract-Transform-Load) suite à des opérations d'évolution dans les sources de données. Ils ont proposé un outil nommé *HECATAEUS* offrant aux concepteurs un mécanisme permettant une adaptation des activités ETL aux changements qui se produisent sur les schémas des sources de données ainsi que la détection précoce des parties évolutives dans le SI. L'approche proposée se base sur une représentation technique qui fait correspondre tous les composants essentiels du processus d'alimentation ETL sous forme d'un modèle appelé graphe d'évolution. Suite à un changement d'un élément du graphe, l'outil détecte les parties touchées par ce changement et met en évidence les modifications à apporter selon des règles définies à l'avance pour assurer la cohérence des activités ETL.

2.4 Bilan

Nous avons présenté un état de l'art des travaux traitant le problème d'évolution dans les systèmes d'information décisionnels. Ces travaux ont traité l'effet d'évolution de la source de données sur l'entrepôt selon différents points de vue : évolution du modèle de l'ED, évolution des vues matérialisées, et évolution du processus de chargement ETL. Nous constatons que peu de travaux se sont intéressés à l'étude de l'effet de l'évolution du modèle des sources

de données sur le modèle multidimensionnel ainsi que sur le processus ETL. Néanmoins, la plupart de ces travaux fournit des solutions qui touchent quelques aspects. Par ailleurs, ils sont réalisés dans un contexte de conception et de développement classique et sont par conséquent fortement dépendants d'une plateforme spécifique. Pour pallier ces problèmes de dépendance, nous proposons une approche s'appuyant sur le modèle MDA où la mise à jour de l'ED est préconisée. Afin de comparer ces travaux et nous positionner, nous avons élaboré le tableau 1.

Critères / Travaux	Modèle Multidimensionnel		Vues M.	ETL	Évolution dirigée par		Usage MDA
	MAJ du modèle	Version-nement			Besoins des décideurs	Modèle de la source	
Hurtado 1999	✓		✓		✓		
Blaschka 1999	✓				✓		
Benitez 2004	✓				✓		
Favre 2007	✓				✓		
Talwar 2012	✓				✓		
Golfareli 2006		✓			✓		
Wrembel 2007		✓			✓		
Solodovnikova 2008						✓	
Rundensteiner 1997			✓			✓	
Bellahsene 2002			✓			✓	
Papastefanatos 2009				✓		✓	
Notre approche	✓			✓		✓	✓

TAB. 1 – *Approche d'évolution proposée.*

Nous abordons le problème d'évolution des systèmes d'information décisionnels en adoptant l'Architecture Dirigée par les Modèles (« MDA : Model Driven Architecture »). Ce choix est motivé par le fait que MDA présente un support souple et efficace pour la gestion des évolutions. En effet, le processus d'évolution d'un entrepôt permet d'épargner des efforts, de réduire les délais et les coûts et améliore la qualité lorsqu'il est traité à un niveau d'abstraction élevé, c'est-à-dire en utilisant des modèles. Ceci est particulièrement avantageux car MDA offre des mécanismes de transformations automatiques entre modèles situés à différents niveaux d'abstraction, contrairement aux approches classiques qui touchent directement le niveau implantation. De plus, MDA permet de fournir un support pour l'évolution, l'intégration, l'interopérabilité, l'adaptabilité, la portabilité et la réutilisabilité des systèmes d'information.

3 Approche proposée

Nous adoptons la démarche MDA pour automatiser la propagation des évolutions du schéma d'une source de données vers l'entrepôt multidimensionnel.

MDA propose une démarche de développement de systèmes informatiques basée sur les modèles et des transformations de modèles conformément à un ensemble de standards de l'OMG (OMG, 2004). Cette démarche permet de séparer les spécifications fonctionnelles d'un système des détails de son implantation. En effet, tout est considéré comme modèle, aussi bien les schémas que le code. La figure 1 montre les différentes étapes de l'approche proposée où

nous retrouvons les trois niveaux de modèles et les deux types de transformations : Verticale et Horizontale.

La transformation verticale met en jeu différents niveaux d'abstraction. Elle permet le passage du modèle des exigences CIM (« Computation Independent Model ») vers le modèle d'analyse et de conception PIM (« Platform Independent Model ») puis vers le modèle de conception concrète PSM (« Platform Specific Model ») pour aboutir enfin à un code d'évolution de l'ED. Le passage entre ces modèles se fait à travers des règles de transformations. Dans la figure 1 :

- CIM : est un modèle indépendant de tout système informatique. C'est le modèle métier ou le modèle du domaine d'application. Il représente le point de départ du processus d'altération de l'ED en permettant de décrire les besoins de l'administrateur en termes d'évolution du système d'information décisionnel ;
- PIM : est un modèle indépendant de toute plateforme technique. Il définit la structure et le comportement du système sans faire référence à la plateforme d'exécution. Ce niveau constitue le cœur de la démarche proposée. Il permet la gestion des évolutions dans les différents niveaux du SID, à savoir l'ED et le processus de chargement ETL ;
- PSM : il s'agit d'un modèle dépendant des plates-formes technologiques ; il présente une projection d'un PIM sur une plateforme donnée pour la génération du code exécutable correspondant.

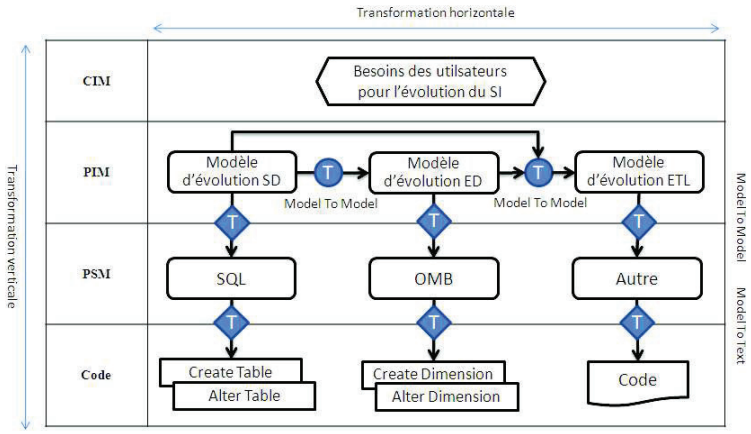

FIG. 1 – *Approche d'évolution proposée.*

La transformation horizontale permet le passage d'un ou de plusieurs modèles sources vers un modèle cible du même niveau d'abstraction mais plus raffiné. La figure 1 montre ce type de transformation au niveau du PIM. Nous distinguons trois modèles d'évolution à ce niveau :

- Modèle d'évolution source de données : Ce modèle décrit toutes les opérations susceptibles d'affecter une base de données relationnelle (e.g. table, colonne) ;
- Modèle d'évolution ED : Il décrit les différentes opérations susceptibles d'affecter les structures multidimensionnelles (e.g. dimension, faits). Il est déduit du modèle d'évolution source de données ;

– Modèle d'évolution ETL : Il décrit les opérations d'évolution applicables sur l'ETL. Il est déduit des deux modèles d'évolution : Source et ED.

Chacun de ces modèles doit être conforme à un méta-modèle qui doit être, à son tour, basé sur un méta-méta-modèle (méta-modèle MOF : Meta Object Facility) (OMG, 2009). Les règles de transformation sont définies avec le langage de transformation qui est aussi basé sur un méta-modèle MOF. Ces règles permettent de définir les correspondances entre les méta-modèles source et cible ; leur exécution génère un modèle cible à partir d'un modèle source.

4 Transformation de modèle

La transformation de modèle permet de générer un modèle (ou méta-modèle) cible à partir d'un modèle source. Elle se base sur des règles de transformation. Dans cet article, nous détaillons la transformation qui permet le passage entre le modèle d'évolution de la source de données et le modèle d'évolution de l'entrepôt. Cette transformation est définie à travers des règles de correspondance entre ces deux modèles et se fait de manière automatique conformément à la figure 2.

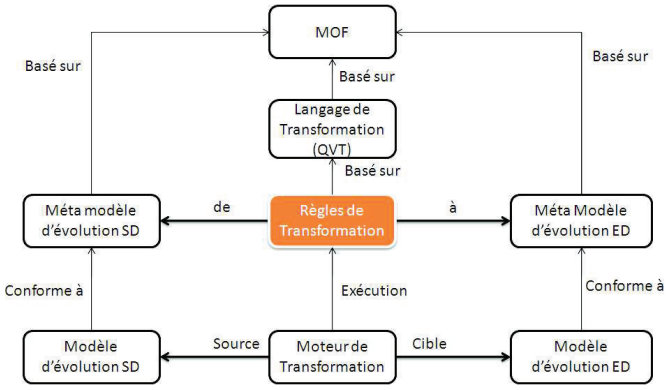

FIG. 2 – *Transformation de modèle.*

En fait, nous étudions l'automatisation de l'impact du processus d'évolution du modèle de la source sur le modèle multidimensionnel de l'entrepôt afin de déduire les changements que doit subir ce dernier. Pour atteindre cet objectif, nous définissons un modèle d'évolution de la source de données (MEvSD) et un modèle d'évolution de l'ED (MEvED). Chacun de ces deux modèles est conforme à un méta modèle d'évolution respectivement MMEvSD et MMEvED ; ces derniers doivent être, à leur tour, basés sur un méta-méta-modèle MOF. Les règles de transformation sont définies en utilisant un langage de transformation qui est aussi basé sur un méta-modèle MOF. Nous définissons ces règles en QVT (« Query/View/Transformation ») qui est un langage déclaratif standardisé par l'OMG (OMG, 2009).

Dans la suite de cette section, nous détaillerons :

– Les modèles utilisés pour définir les opérations d'évolution au niveau source et ED ;
– Quelques règles de transformation en QVT qui serviront d'illustration.

4.1 Les modèles

Nous représentons les différents modèles d'évolution par des diagrammes de classes UML (« Unified Modeling Language ») (OMG, 2001). UML est un langage d'expression de modèles à objets ; il permet de modéliser les aspects structurels et comportementaux d'un système. Particulièrement, les modèles de classes permettent de représenter simultanément ces deux aspects. En effet, une classe UML est composée d'une liste de propriétés statiques et d'un ensemble d'opérations. Nous utilisons d'une part la liste des propriétés pour définir les structures des SD, ETL et l'ED, et d'autre part, les opérations pour définir les changements que peut subir chacune de ces structures.

4.1.1 Modèle d'évolution de la source de données

Le modèle d'évolution de la source de données est le modèle de base pour la déduction des modèles d'évolution du processus ETL et de l'ED. Il définit le schéma de la source (sous forme de tables, contraintes, ...) en utilisant les propriétés des classes ainsi que les opérations d'évolution qui lui sont applicables (ajout de table, ajout de colonne, ...).

Le modèle d'évolution des sources est décrit par le méta-modèle de la figure 3. Ce dernier est composé de la classe « DS_Schema » qui est une extension de la méta-classe « Package ». Le schéma de la source de données est, à son tour, composé de plusieurs tables. Chacune d'elles est définie par une classe « Table » qui est une extension de la méta-classe « Class ». Chaque table est composée d'une ou plusieurs colonnes, certaines peuvent être clé primaire ou clé étrangère. La classe « Column » est une extension de la méta-classe «StructuralFeature».

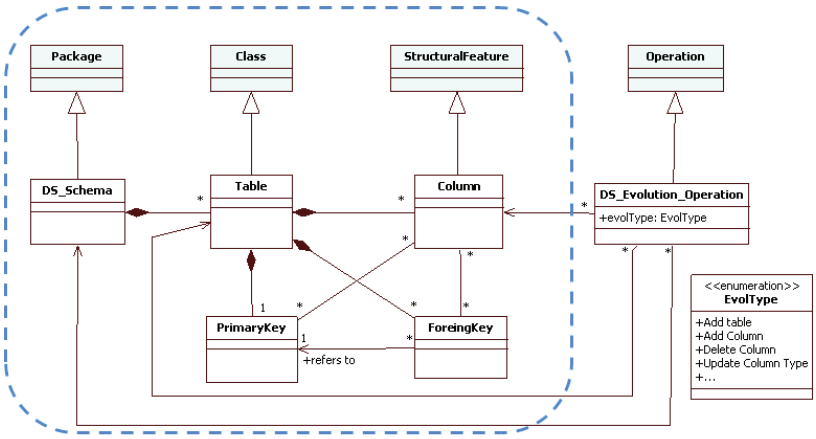

FIG. 3 – *Diagramme de classes UML du Méta modèle d'évolution SD.*

La partie en pointillé du modèle de la figure 3 représente la structure des sources. Nous lui ajoutons les opérations décrivant les modifications apportées aux schémas des sources. Ces opérations (ajout, suppression, modification,...) concernent principalement les tables et les colonnes. Nous les modélisons par la classe « DS_Evolution_Operation » qui est une extension de la méta-classe « Operation ».

Rappelons qu'une transformation permet la génération automatique d'un modèle cible à partir d'un modèle source en appliquant un ensemble de règles. Nous venons de définir le méta-modèle source (cf. figure 3). Nous continuons à définir le méta-modèle cible.

4.1.2 Modèle d'évolution multidimensionnel

Le méta-modèle d'évolution multidimensionnel (cf. figure 4) comporte deux parties : le Méta-Modèle Multidimensionnel (M3) en pointillé, et le Méta-Modèle de ses Opérations d'évolution (MMO). M3 est instancié avec des schémas multidimensionnels alors que le MMO sera instancié automatiquement avec des opérations déduites à partir du modèle d'évolution de la SD et en appliquant les règles de transformation.

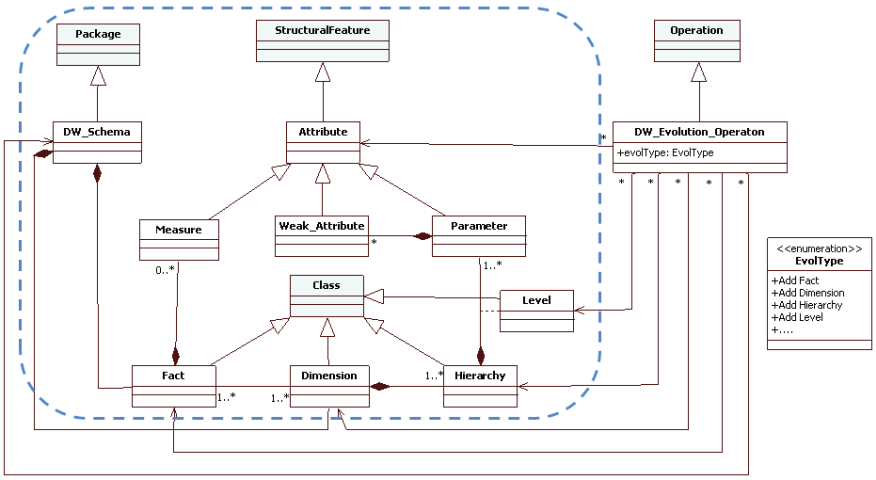

FIG. 4 – *Diagramme de classes UML du Méta-Modèle Multidimensionnel d'évolution.*

M3 est composé du schéma de l'entrepôt « DW_Schema» (extension de la méta-classe Package). Ce schéma se compose de tables de faits « Fact » et de dimensions « Dimension ». Une dimension est composée d'une ou plusieurs hiérarchies « Hierarchy » qui contient un ou plusieurs niveaux d'attributs « Level ».
« Fact », « Dimension », « Hierarchy » et « Level » sont des extensions de la méta-classe « Class ». La table de faits contient une ou plusieurs mesures « Measure ». Chaque niveau d'une hiérarchie se compose d'un paramètre « parameter » auxquels peut être associés un ou plusieurs attributs faibles « Weak_Attribute ».
« Measure », « Parameter » et « Weak_Attribute » héritent de la classe « Attributes » qui est une extension de la méta-classe « StructuralFeature ». Les opérations d'évolution au niveau de l'ED peuvent être appliquées sur les différents composants du modèle de l'ED (table de faits, dimensions Hiérarchies, niveau d'attribut, attribut). Ces opérations peuvent être constructives (ajout dimension, ajout mesure, etc.) ou destructive (suppression dimension, suppression hiérarchie, etc). Nous modélisons ces opérations à travers la classe « Evolution_Operation_ED » qui est une extension de la méta-classe « Operation ».

4.2 Les règles de transformation QVT

Les règles de transformation touchant l'évolution sont nombreuses (cf. (Taktak et Feki, 2012) ; (Taktak et al., 2014)). Elles peuvent concerner les tables, les colonnes, les clés... La figure 5 montre les transformations possibles applicables à l'ED suite à l'ajout d'une nouvelle table ou colonne à la source.

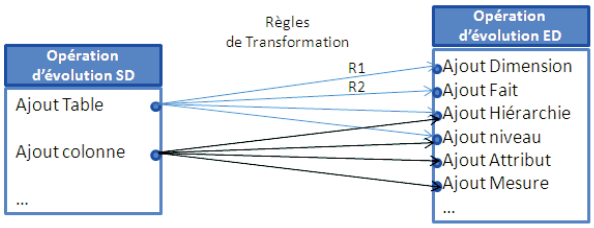

FIG. 5 – *Transformation possibles pour les opérations d'ajout de table ou de colonne.*

Par exemple, soit une table t ajoutée à la SD, nous définissons les deux règles suivantes :

- R1 : t peut se transformer en une dimension lorsqu'elle est référencée par une table alimentant un fait ;
- R2 : si t modélise une association alors elle sera proposée comme fait.

Cette section présente la formalisation en QVT de la règle R1 (avec ses sous règles). Chaque opération du modèle d'évolution source peut se convertir automatiquement en une ou plusieurs opérations d'évolutions dans le modèle cible (modèle d'évolution ED).

Une transformation QVT entre deux modèles candidats est spécifiée grâce à un ensemble de relations. Une relation est définie par les éléments suivants :

- Deux ou plusieurs domaines (domain) : chaque domaine comporte un ensemble d'éléments relatifs au modèle candidat ;
- Checkonly et enforce : un domaine peut être marqué par Checkonly ou bien Enforce. Checkonly (C) vérifie s'il existe une correspondance dans le modèle qui satisfait la relation. Dans le cas où la correspondance n'est pas vérifiée, le domaine ne doit pas être modifié. Par contre, lorsqu'un domaine est marqué par Enforce (E) et si la correspondance n'est pas vérifiée, alors le modèle doit être modifié afin de satisfaire la relation ;
- Pattern matching : un motif (pattern) apparaît dans un domaine et permet de sélectionner une partie du modèle candidat ;
- When : cette clause définit les conditions nécessaires à la vérification de la relation (i.e., Précondition) ;
- Where : définit les conditions à vérifier par tous les éléments d'un modèle qui participent à la relation (i.e. Post-condition) ;

Dans une transformation, nous trouvons deux types de relations : *top-level relations* et *non-top-level relations*. L'exécution d'une transformation entraîne l'exécution de tous ses top-level relations. Les non-top-level relations sont invoquées à partir des clauses Where des top-level relations ou bien d'autres non-top-level relations (OMG, 2009).

Nous présentons dans un premier temps la relation *Main* suivie des deux sous relations qu'elle invoque en cascade.

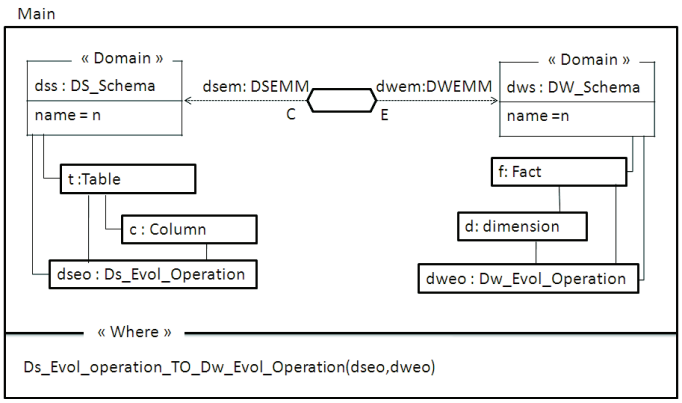

FIG. 6 – *Représentation graphique de la relation Main.*

Main. C'est la relation point d'entrée du processus de transformation. Elle comporte :

– les éléments du modèle «dsem» conforme au DSEMM (DS_Evolution_Meta_model) ;

– les éléments du modèle «dwem» conforme au DWEMM (DW_Evolution_Meta_model).

L'élément *Domain* du modèle « dsem » est marqué par « C » tandis que l'élément *Domain* du modèle « dwem » est marqué par « E ». La partie gauche de cette relation décrit les éléments du modèle source « dsem » transformés en éléments du modèle cible « dsem » (cf., figure 6). Plus particulièrement, une opération d'évolution de gauche « dseo : Ds_Evol_Operation » est transformée en une ou plusieurs opérations d'évolutions « dweo : Dw_Evol_Operation» en invoquant la relation « DS_Evolution_Operation_TO_Dw_Evolution_Operation (dseo, dweo) » spécifiée dans la clause « where » que nous décrivons dans ce qui suit.

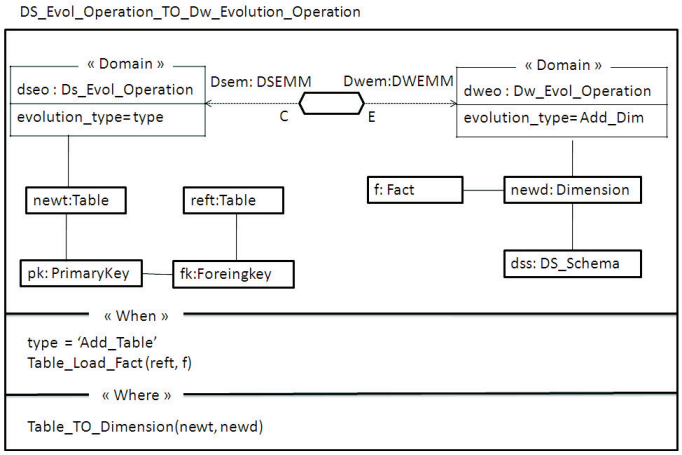

FIG. 7 – *Relation pour l'ajout d'une table.*

DS_Evolution_Operation_TO_Dw_Evolution_Operation. La figure 7 décrit cette relation qui traite l'évolution de l'ED multidimensionnel suite à l'ajout d'une table « Add_Table » dans la source. La clause « When » spécifie la condition à vérifier pour la réalisation de la relation et signifie : Si la table ajoutée « newt » est référencée par une table « reft » qui alimente un fait « f » via la relation « Table_Load_Fact (reft,f) » alors « newt » sera transformée en une nouvelle dimension « newd » via la relation « Table_TO_Dimenson (newt, newd) » spécifiée dans la clause « Where » ; nous expliquons cette relation.

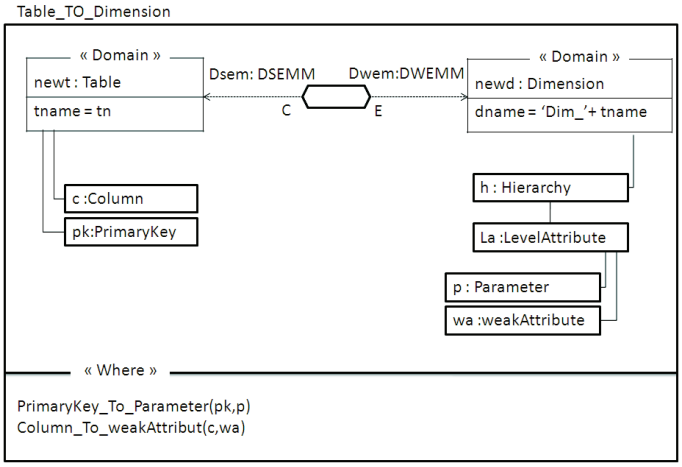

FIG. 8 – *Relation Table_TO_Dimension.*

Relation Table_TO_Dimension. Une table « newt » se transforme en une dimension dont le nom est la concaténation du suffixe « Dim_» et du nom de la table. La clé primaire de « newt » se transforme en un paramètre via la relation « PrimaryKeyToparameter » et les colonnes en attributs faibles associés à ce paramètre via la relation « ColumnToWeakattribute ».

5 Implantation

Pour valider notre démarche, nous développons actuellement un prototype nommé *DWE* (« Data Warehouse Evolution »). *DWE* est implanté sur la plateforme Eclipse Modeling Framework (EMF) qui présente un environnement complet pour la mise en œuvre d'une démarche MDA. La figure 9 décrit l'architecture globale de *DWE*.Le point de départ du processus d'évolution de l'ED est l'identification des opérations d'évolution affectant sa source d'alimentation. Ces opérations peuvent être directement repérées à partir du SGBD s'il supporte les triggers ou spécifiées par l'administrateur. *DWE* se base sur un référentiel qui effectue le stockage et la récupération des informations utiles pour les règles de transformation. Ces informations comprennent le modèle relationnel de la SD, le modèle multidimensionnel de l'ED ainsi que les correspondances entre ces deux modèles. Le modèle de la SD avec les opérations d'évolution forment ensemble le modèle d'évolution source de données.

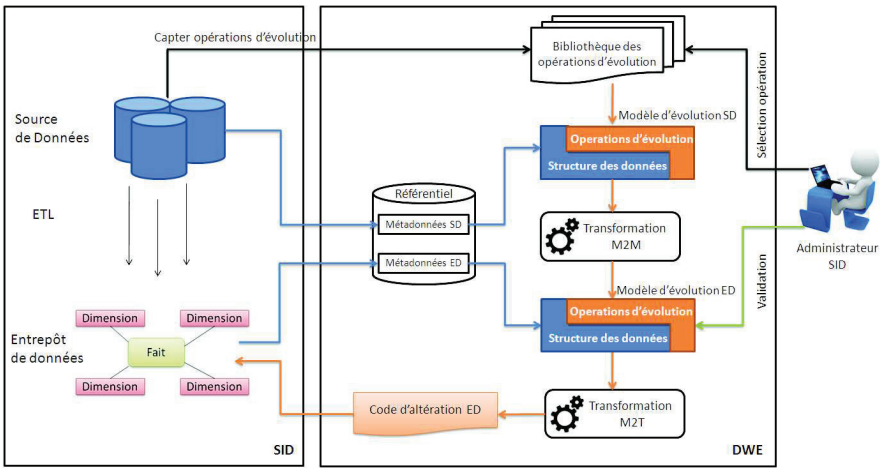

FIG. 9 – *Architecture du prototype DWE.*

Le processus d'altération d'ED se base principalement sur deux transformations successives de modèles à deux niveaux d'abstraction :

- La première est de type « Model-To-Model » et permet de transformer le modèle d'évolution SD en modèle d'évolution ED ;
- La deuxième transformation est de type « Model-To-Text » et consiste à traduire le modèle d'évolution ED en un script exécutable sur l'ED.

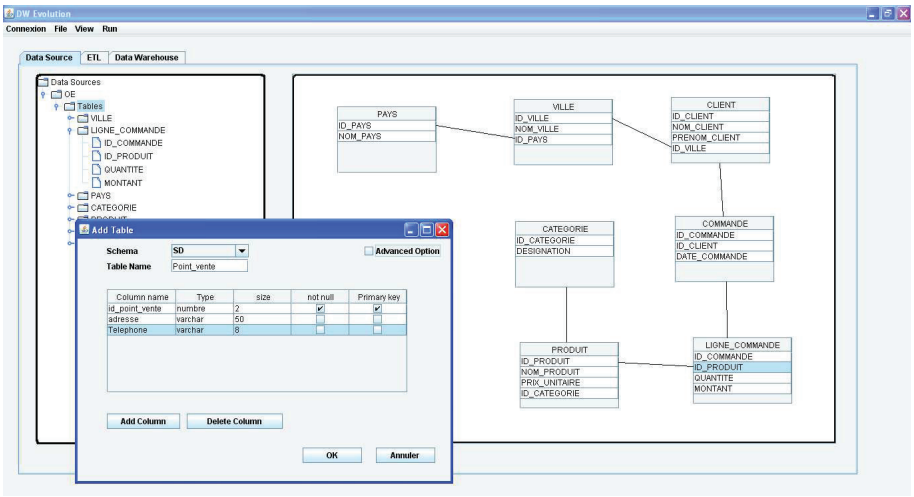

FIG. 10 – *Interface DWE/onglet source de données.*

Nous illustrons, à travers des interfaces de *DWE*, un scénario d'évolution du schéma d'une

source de données. *DWE* permet un affichage graphique et interactif des schémas de la source et de l'entrepôt. La figure 10 est un exemple de schéma relationnel d'une SD sur lequel on applique l'opération d'évolution *Add_Table* qui ajoute la table *Point_vente (id_point_vente, adresse, telephone)*.

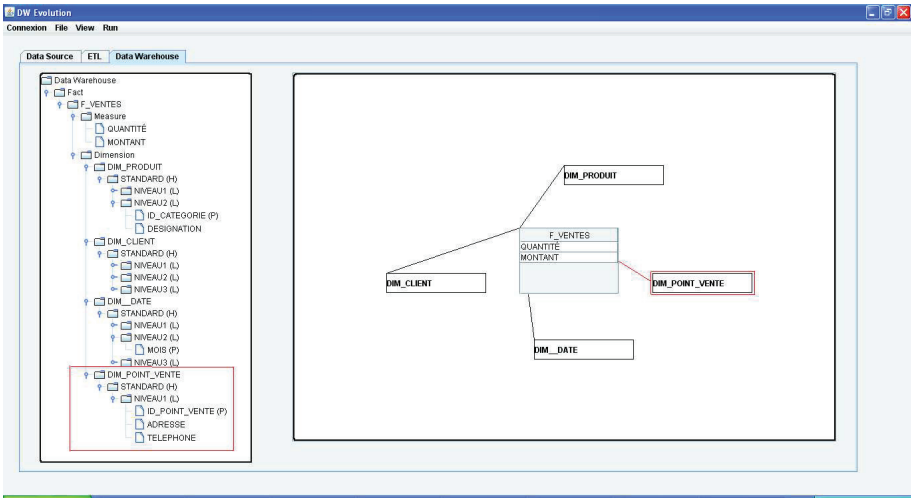

FIG. 11 – *Entrepôt de données modifié après ajout de table.*

La figure 11 donne le schéma multidimensionnel de l'ED modifié suite à l'ajout de la table *Point_vente* à la SD. Graphiquement, l'effet de cette évolution est marqué en rouge et consiste à la présence d'une nouvelle dimension nommée *Dim_point_vente* rattachée au fait *F_VENTES* conformément aux règles de transformation définies dans la section (4.2). En effet, puisque *Point_vente* est référencée par la table *Commande* et qu'elle alimente le fait *F_VENTES* alors une nouvelle dimension *Dim_point_vente* peut être ajoutée au fait *F_ventes* (cf. § 5, règle R1).

6 Conclusion

Dans cet article nous avons abordé le problème d'évolution dans le système d'information décisionnel. Nous avons étudié l'effet de l'évolution du modèle de la source de données relationnel sur le modèle multidimensionnel de l'entrepôt qu'elle alimente. Pour répondre à cette problématique, nous avons proposé une approche dirigée par les modèles pour automatiser la propagation de ces évolutions. Nous avons présenté ici la mise en œuvre de cette approche, en proposant deux modèles d'évolution décrivant à la fois les aspects structurels des données et comportementaux. Parmi les règles de transformation, nous avons défini en QVT celle d'ajout d'une table à la source de données. Plus particulièrement, nous avons illustré à travers des interfaces de *DWE* l'effet de cette règle sur l'ED multidimensionnel assurant ainsi le passage entre les deux modèles. D'autres cas d'évolutions sont actuellement supportés par le prototype (ajout de colonne normale ou clé, suppression de table, ...). A l'état actuel, nous sommes entrain de renforcer *DWE* par des fonctionnalités graphiques qui devraient offrir à l'utilisateur

le moyen de modifier les résultats des règles. Les outils de BPMN (« Business Process Model and Notation ») ou de workflow seront envisagés.

Ce travail ouvre différentes perspectives. Nous envisageons réaliser une étude de performance des règles de transformations à travers un cas réel d'étude. Également nous comptons étendre le processus d'altération de l'ED pour tenir compte des effets des évolutions sur le processus de chargement ETL. La généralisation de l'approche à de multiples sources structurellement hétérogènes nécessiterait le passage par un modèle de données pivot entre la source et le modèle d'évolution qui deviendrait générique.

Références

Atigui, F., F. Ravat, O. Teste, et G. Zurfluh (2012). Using ocl for automatically producing multidimensional models and etl processes. In *Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK'12, Berlin, Heidelberg, pp. 42–53. Springer-Verlag.

Bellahsene, Z. (2002). Schema evolution in data warehouses. *Knowl. Inf. Syst. 4*(3), 283–304.

Benitez-Guerrero, E., C. Collet, et M. Adiba (2004). The WHES approach to data warehouse evolution. Technical report, E-Gnosis [online],Vol. 2, Art. 11.

Blaschka, M., C. Sapia, et G. Höfling (1999). On schema evolution in multidimensional databases. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*, DaWaK '99, London, UK, UK, pp. 153–164. Springer-Verlag.

Favre, C., F. Bentayeb, et O. Boussaid (2007). Dimension hierarchies updates in data warehouses : a user-driven approach. In *Proceedings of the 9th International Conference on Enterprise Information Systems*, ICEIS 07, Madeira, Portugal, pp. 206–211.

Golfarelli, M., J. Lechtenbörger, S. Rizzi, et G. Vossen (2006). Schema versioning in data warehouses : Enabling cross-version querying via schema augmentation. *Data Knowl. Eng. 59*(2), 435–459.

Hurtado, C. A., A. O. Mendelzon, et A. A. Vaisman (1999). Maintaining data cubes under dimension updates. In *Proceedings of the 15th International Conference on Data Engineering*, ICDE '99, Washington, DC, USA, pp. 346–355. IEEE Computer Society.

Inmon, W. H. (2002). *Building the Data Warehouse,3rd Edition* (3rd ed.). New York, NY, USA : John Wiley & Sons, Inc.

Jain, H. et A. Gosain (2012). A comprehensive study of view maintenance approaches in data warehousing evolution. *SIGSOFT Softw. Eng. Notes 37*(5), 1–8.

OMG (2001). Object Management Group : Unified Modeling Language Specification (UML) 1.4. Technical report, http ://www.omg.org/cgi-bin/doc ?formal/01-09-67.

OMG (2004). Object Management Group : Model Driven Architecture (MDA). Technical report, http ://www.omg.org/cgibin/doc ?formal/03-06-01.

OMG (2009). Object Management Group : Meta Object Facility (MOF) 2.0. Technical report, http ://www.omg.org/spec/QVT/1.1/Beta2/.

Papastefanatos, G., P. Vassiliadis, A. Simitsis, T. Sellis, et Y. Vassiliou (2010). Rule-based management of schema changes at etl sources. In *Proceedings of the 13th East European*

Conference on Advances in Databases and Information Systems, ADBIS'09, Berlin, Heidelberg, pp. 55–62. Springer-Verlag.

Rundensteiner, E. A., A. Nica, et A. J. Lee (1997). On preserving views in evolving environments. In *Proceedings of the 4th International Workshop Knowledge Representation Meets Databases*, pp. 131–141.

Solodovnikova, D. (2008). The formal model for multiversion data warehouse evolution. In *Proceedings of the 8th International Baltic Conference on Databases and Information Systems*, pp. 91–102.

Taktak, S. et J. Feki (2012). Toward propagating the evolution of data warehouse on data marts. In *Proceedings of the 2Nd International Conference on Model and Data Engineering*, MEDI'12, Berlin, Heidelberg, pp. 178–185. Springer-Verlag.

Taktak, S., J. Feki, et G. Zurfluh (2014). Toward evolution models for data warehouses. In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, MODELSWARD 2014, pp. 472–479.

Talwar, K. et A. Gosain (2012). Implementing schema evolution in data warehouse through complex hierarchy semantics. *International Journal of Scientific and Engineering Research 3*, 917–922.

Wrembel, R. et B. Bebel (2007). Metadata management in a multiversion data warehouse. *Journal on Data Semantics VIII 4380*(2), 118–157.

Summary

Modeling and data warehousing have been considered, for more than a decade, as a research problem for which different approaches have been proposed. These proposals focused only on static aspects. However, the evolution of the information system can have impacts on its dependant data warehouse, and therefore may require the evolution of the multidimensional model. In this evolving context, we propose a model-driven approach to automate the propagation of the evolution from data source to the warehouse. This approach is based on two evolution models and a set of transformation rules formalized in Query/View/Transformation. We are developing a software prototype called DWE (Data Warehouse Evolution) which supports this approach.