Processus de transformation MDA d'un schéma conceptuel de données en un schéma logique NoSQL

Fatma ABDELHEDI [(1) et (3)], Amal AIT BRAHIM [(1)], Faten ATIGUI [(2)], Gilles ZURFLUH [(1)]

1. IRIT - Université Toulouse Capitole - France
<prenom.nom>@irit.fr
2. CEDRIC – CNAM Paris – France
<prenom.nom>@cnam.fr
3. CBI² - Sté TRIMANE Saint Germain en Laye – France
<prenom.nom>@gmail.com

RESUME : *La transformation digitale des entreprises et plus largement celle de la société, entraine une évolution des bases de données vers le Big Data. Nos travaux s'inscrivent dans cette mutation et concernent plus particulièrement les mécanismes d'implantation d'une base de données sur une plateforme NoSQL. Pour automatiser ce processus d'implantation, nous avons utilisé l'architecture MDA qui offre un cadre formalisé aux mécanismes de transformation des schémas. A partir d'un schéma conceptuel décrivant une base d'objets complexes, nous proposons des règles de dérivation pour générer, in fine, un schéma d'implantation destine à une plateforme NoSQL orientée colonnes. Nous introduisons un schéma intermédiaire de niveau logique afin de limiter les impacts liés aux évolutions techniques des plateformes NoSQL. Une expérimentation du processus de transformation a été réalisée sur une application médicale.*

ABSRACT: Recent years have seen a real explosion of volume of data available in business and on the web. In this paper, we consider the automatic transformation of Big Data conceptual schema within NoSQL systems. For this, we use the Model Driven Architecture (MDA) that provides a framework for models automatic transformation. Starting from a conceptual model that describes a set of complex objects, we propose transformation rules to generate, ultimately, two NoSQL models: columns-oriented model and documents-oriented model. To ensure efficient automatic transformation, we use a logical model that limits the impacts related to technical developments of NoSQL platforms. We provide experiments of the QVT model transformations in the context of health area.

Mots-clés : Big Data, systèmes NoSQL, transformation de schémas, architecture MDA.

Keywords: Big Data, NoSQL systems, schema transformation, MDA.

1. Introduction

Pendant trois décennies, les systèmes relationnels ont représenté l'outil majeur pour l'exploitation des bases de données. Mais le modèle relationnel connaît des limites face aux nouvelles applications qui apparaissent sur le Web. Ces dernières années ont connu une véritable explosion du volume des données disponibles dans les entreprises et sur le Web. Ces nouvelles problématiques sont désignées par l'expression « Big Data » [5] et caractérisées par la règle dite des « 3V » [10]. Il s'agit du Volume (des masses considérables des données à gérer), de la Variété (des données complexes) et de la Vélocité (en référence à la collecte et au traitement en temps-réel de ces données). Les approches classiques basées principalement sur le paradigme relationnel, ne peuvent pas répondre à ces objectifs et exigent de nouvelles approches de stockage et de manipulation des données. Regroupées sous le terme NoSQL [8], ces approches permettent une plus grande adaptabilité dans des contextes fortement distribués, ainsi qu'une gestion performante des données complexes [7]. Les développeurs d'applications Big data se trouvent notamment confrontés à la problématique du stockage des données avec des systèmes NoSQL. L'objectif de nos travaux est donc de faciliter le processus d'implantation de bases d'objets sur des plateformes NoSQL. En raison de la complexité des schémas des bases de données à implanter et de la spécificité des modèles NoSQL, nous proposons un processus de transformation des modèles basé sur l'approche de l'Ingénierie Dirigée par les Modèles [3].

Dans la section 2 suivante, nous présentons l'application médicale qui justifie l'intérêt de nos travaux. La section 3 décrit le contexte de notre étude ainsi que notre problématique de recherche visant à implanter des données sur une plateforme NoSQL. La section 4 présente notre contribution qui consiste à formaliser avec MDA le processus de transformation d'un schéma conceptuel de données en un schéma logique NoSQL. La section 5 décrit une expérimentation du processus proposé à partir de notre application médicale. Enfin, la section 6 positionne nos travaux par rapport à l'état de l'art.

2. Cas d'étude

Pour illustrer nos travaux, nous utilisons un cas extrait d'une application médicale dont la base de données est représentée dans le formalisme UML. Cet exemple nous permettra de montrer comment transformer un diagramme UML en un schéma NoSQL. Il s'agit de la mise en place de programmes nationaux ou internationaux pour le suivi de cohortes de patients atteints de pathologies graves. L'objectif majeur d'un tel programme est de collecter des données sur l'évolution temporelle d'une pathologie particulière, d'étudier les interactions de la pathologie avec des maladies opportunes et d'évaluer l'influence des traitements et médications à court et moyen termes. La durée d'un programme est décidée lors de son lancement et peut atteindre trois ans. Les données collectées par plusieurs établissements dans le cadre d'un programme pluriannuel, présentent les caractéristiques généralement admises pour le Big Data (les 3 V). En effet, le volume des données médicales recueillies quotidiennement auprès des patients, peut atteindre, pour l'ensemble des établissements et sur trois années, plusieurs téraoctets. D'autre part, la nature des données saisies (mesures, radiographie, scintigraphies, etc.) est diversifiée et peut varier d'un patient à un autre selon son état de santé. Enfin, certaines données sont produites en flux continu par des capteurs ; elles doivent être traitées quasiment en temps réel car elles peuvent s'intégrer dans des processus sensibles au temps (mesures franchissant un seuil qui

impliqueraient l'intervention d'un praticien en urgence par exemple). Le suivi des patients exige le stockage de données variées telles que l'enregistrement des consultations effectuées par les praticiens, des résultats d'examens, des prescriptions de médicaments et de traitements spécifiques. L'extrait de diagramme UML de la figure 1 montre quelques classes pour un programme médical associé au suivi des patients atteints d'une pathologie particulière.

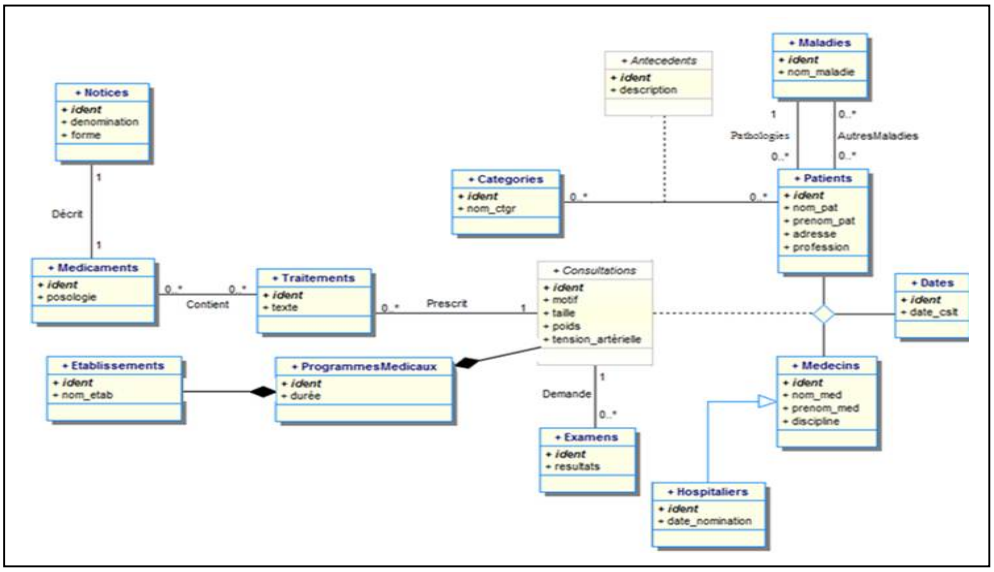


Figure 1. Extrait d'un schéma des données

3. Contexte et Problématique

Notre objectif est de partir d'un schéma de bases de données tel que celui de la figure 1 et de le transformer en un schéma NoSQL. Nous présentons dans cette section le contexte de notre étude en abordant les principes généraux de l'approche IDM. Nous décrivons ensuite la problématique de recherche traitée dans le cadre de cet article.

3.1. Ingénierie Dirigée par les Modèles (IDM)

Pour faire face à la complexité des applications informatiques, l'IDM [3] consiste à adopter les modèles comme éléments centraux dans le processus de développement d'une application et à automatiser la transformation de ces modèles afin d'aboutir au code source. L'Architecture Dirigé par les Modèles (MDA pour Model Driven Architecture) [9] proposée par l'Object Management Group[1] (OMG) est un mécanisme dérivé de l'IDM. MDA propose de décrire séparément les spécifications fonctionnelles et les spécifications

d'implantation d'une application sur une plateforme donnée. Parmi les modèles proposés, nous retenons (1) le modèle d'analyse et de conception (Platform Independent Model – PIM) qui décrit les données en faisant abstraction des aspects techniques liés aux systèmes informatiques et (2) le modèle de code (Platform Specific Model – PSM) qui représente les données en tenant compte des caractéristiques d'une plateforme de stockage particulière.

Dans la suite de l'article, nous considérons deux sortes de PIM : le PIM conceptuel qui décrit les données sous ses seuls aspects métier et le PIM logique qui tient compte en plus du type d'organisation des données choisi. Le passage entre deux modèles MDA se fait via une succession de transformations. Une transformation correspond à l'application d'un ensemble de règles qui décrit comment dériver un modèle cible à partir d'un modèle source. Pour la transformation de modèles, l'OMG a défini le standard QVT[2] (Query/View/Transformation) qui propose des langages d'expression de règles.

3.2. Problématique

Notre problématique consiste à définir des mécanismes permettant d'implanter une base de données de type Big data sur une plateforme NoSQL. A partir d'un diagramme de classes UML décrivant des données complexes, nous devons spécifier les transformations nécessaires à l'élaboration d'un schéma NoSQL. Parmi les principaux types de systèmes NoSQL (Clé-valeur, Colonne, Document, Graphe) [1], nous avons choisi d'implanter les données sur un système orienté colonnes. Ce choix a été dicté par les besoins de nos applications basés sur des requêtes multicritères faisant intervenir simultanément plusieurs attributs. Or les systèmes orientés colonnes offrent des techniques de stockage qui sérialisent les valeurs des colonnes et permettent ainsi d'accélérer l'accès aux données. Le problème consiste donc à passer d'un schéma conceptuel de base de données (DCL – Diagramme de Classes d'UML) vers un schéma physique NoSQL qui fera l'objet d'une implantation. Mais plusieurs systèmes NoSQL orientés colonne coexistent ; les plus connus sont BigTable [6], HBase[3], Cassandra[4] et Accumulo[5]. Ils présentent des spécificités techniques propres qui relèvent essentiellement des techniques d'implantation. Pour faire abstraction de ces spécificités, nous intègrerons le niveau logique dans le processus de transformation des schémas. Autrement dit, nous considérerons les transformations successives : Conceptuel $\longrightarrow$ Logique puis Logique $\longrightarrow$ Physique. Au niveau logique, le schéma décrit l'implantation des données en faisant abstraction de considérations techniques propres à tel ou tel système NoSQL.

3.3. Etat de l'art

Une base de données de type Big data contient des données variées, c'est-à-dire des données de types non standard qualifiés généralement d'objets complexes : textes, graphiques, documents, séquences vidéo. Aujourd'hui, le modèle de données d'UML

[2] http://www.omg.org/spec/QVT/1.2/PDF/.
[3] https://hbase.apache.org/.
[4] http://cassandra.apache.org/.
[5] https://accumulo.apache.org/.

représente une sorte de référence en matière de représentation de schémas de bases de données complexes [2]. Ce modèle conceptuel, permettant de décrire la sémantique des objets métiers dans une application, peut donc être appliqué à la description des bases de données de type Big data.

En ce qui concerne les processus permettant d'implanter des bases de données sur des systèmes NoSQL, plusieurs études ont porté sur la transformation des schémas. Ainsi, dans le contexte des entrepôts de données, les travaux de Chevalier et al. [11] ont défini des règles pour traduire un modèle multidimensionnel en étoile, en deux modèles physiques NoSQL, un modèle orienté colonnes et un modèle orienté documents. Les liens entre faits et dimensions ont été traduits sous la forme d'imbrications. L'article de Li [4] a étudié l'implantation d'une base de données relationnelle dans le système HBase. La méthode proposée est basée sur des règles permettant la transformation d'un schéma relationnel en un schéma HBase ; les relations entre les tables (clés étrangères) sont traduites par l'ajout des familles de colonnes contenant des références. D'autres travaux ont étudié la transformation d'un diagramme de classes UML en un schéma de données HBase avec l'approche MDA [14]. L'idée de base est de construire des méta-modèles correspondant au diagramme de classes UML et au modèle de données orienté colonne HBase puis de proposer des règles de transformation entre les éléments des deux méta-modèles construits. Ces règles permettent de transformer un DCL directement en un schéma d'implantation spécifique au système HBase. Cet état de l'art montre que peu de travaux ont étudié la transformation d'un modèle conceptuel de données complexes vers un modèle NoSQL. Dans l'étude [14] la plus proche de notre problématique, les règles de transformation de schémas qui ont été adoptées, ne sont pas indépendantes d'une plateforme technique.

4. Transformation des schémas

Nos travaux visent à transformer un schéma conceptuel décrivant une source Big data en un modèle NoSQL orienté colonnes. Pour ce faire, nous proposons une approche dirigée par les modèles (MDA) qui fournit des métamodèles et des règles de transformation permettant le passage automatique du niveau conceptuel au niveau physique. La figure 2 montre un aperçu de notre contribution. D'une manière générale, une approche MDA repose sur les modèles CIM, PIM et PSM ainsi que les transformations automatiques entre ces modèles. Cependant, chaque approche présente ses propres caractéristiques, notamment le nombre et le type des modèles et des transformations [13]. Notre approche repose donc sur les deux niveaux PIM et PSM.

- le niveau PIM comporte deux niveaux différents ; le premier niveau présente le modèle conceptuel (Diagramme de classes UML) et le second présente le modèle logique (Modèle NoSQL orienté colonnes).

- le niveau PSM décrit les modèles physiques correspondants aux plateformes HBase et Cassandra.

Le passage d'un niveau à un autre se fait automatiquement grâce aux transformations M2M (Model-To-Model) formalisées en QVT[6] comme le montre la figure 2.

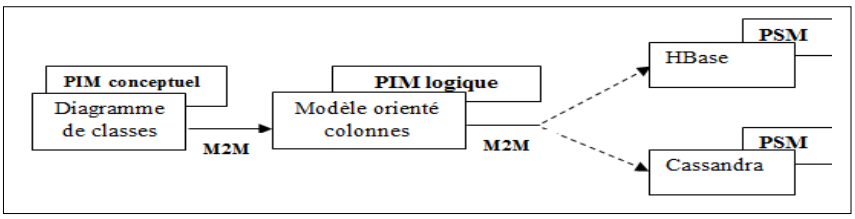


Figure 2. Les niveaux de modélisation

4.1. PIM conceptuel

UML étant le modèle reconnu par la communauté des bases de données pour représenter des objets complexes, nous décrivons le PIM conceptuel sous la forme d'un DCL. Avant d'assurer un passage automatique d'un DCL vers un modèle logique, nous devons préalablement formaliser les concepts présents dans le modèle de données d'UML. Un DCL contient un ensemble de classes. Chaque classe représente des objets ayant une sémantique et des propriétés communes ; elle est définie par son nom, ses attributs et ses opérations (dans cet article, nous prenons en compte uniquement la partie structurelle à l'exclusion des opérations). On distingue principalement quatre types de liens entre les classes : l'association, l'agrégation, la composition et l'héritage. A partir du métamodèle d'un DCL défini par l'OMG [15] et adapté à notre PIM conceptuel, nous présentons ces différents concepts à travers un métamodèle (Figure 3) que nous avons implanté sur la plateforme Eclipse Modeling Framewok[7] (EMF).

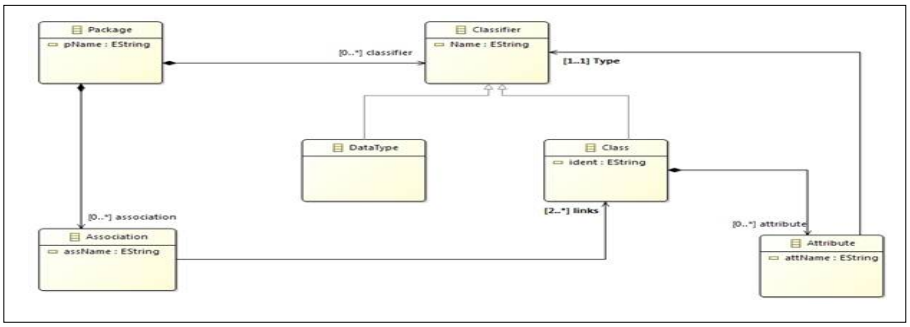


Figure 3. Méta-modèle source

4.2. Transformation automatique

4.2.1. PIM logique

4.2.1.1. Définition d'un modèle orienté colonnes

[6] http://www.omg.org/spec/QVT/1.2/PDF/.
[7] http://www.eclipse.org/modeling/emf/

Dans le niveau logique de description d'une base de données, les choix d'implantation ne sont pas complètement spécifiés. Les principes d'organisation des données sont précisés mais il est fait abstraction du SGBD utilisé pour implanter la base (ce choix se fait au niveau physique) ; seul le type du SGBD est pris en compte. Nous avons retenu un système NoSQL de type orienté colonnes. Selon ce modèle, une base de données (BD) est constituée d'un ensemble de tables. Une table permet de regrouper des objets de taille variable sous forme de lignes ; chacune d'elles est identifiée par un identificateur unique dont le type est noté clé-ligne. Généralement, on regroupe dans une table les objets fortement liés ; par exemple les employés, les services auxquels ils appartiennent et les projets auxquels ils participent. Par défaut, nous stockerons la base de données dans une table unique. Cette table, notée T, est associée à un ensemble de familles de colonnes F : $\{f_1, \dots, f_p\}$. Une famille regroupe un nombre variable de colonnes f : $\{cl_1, \dots, cl_q\}$ chacune d'elles est composée d'un nom, d'un type, d'une valeur et d'un horodatage (Timestamp) pour stocker plusieurs versions de la même donnée. Dans cet article, nous ne considérons pas l'horodatage des données. Nous présentons les concepts du PIM logique orienté colonnes à travers le métamodèle de la figure 4.

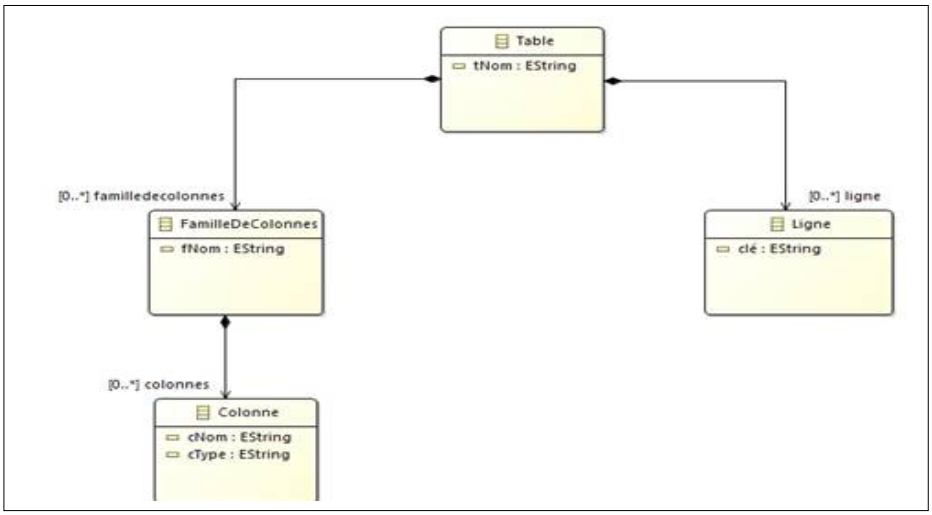


Figure 4. Méta-modèle cible

4.2.1.2. Règles de transformation

Nous proposons les règles suivantes en mettant en vis-à-vis un schéma conceptuel UML et sa traduction dans le PIM logique. Plusieurs solutions de transformation en PIM logique sont parfois possibles ; nous avons opté pour celles qui s'adaptent le mieux aux manipulations prévues dans notre application médicale.

- R1 : Package → Table, par défaut, les objets fortement liés sont regroupés dans une table ; c'est par exemple le cas des patients, des consultations médicales et des traitements prescrits par les médecins. Par analogie, la notion de Table est ici à rapprocher du concept

de paquetage dans UML qui permet de regrouper des éléments (classes, associations, composants, etc.) dans le but de constituer des ensembles d'objets sémantiquement liés.

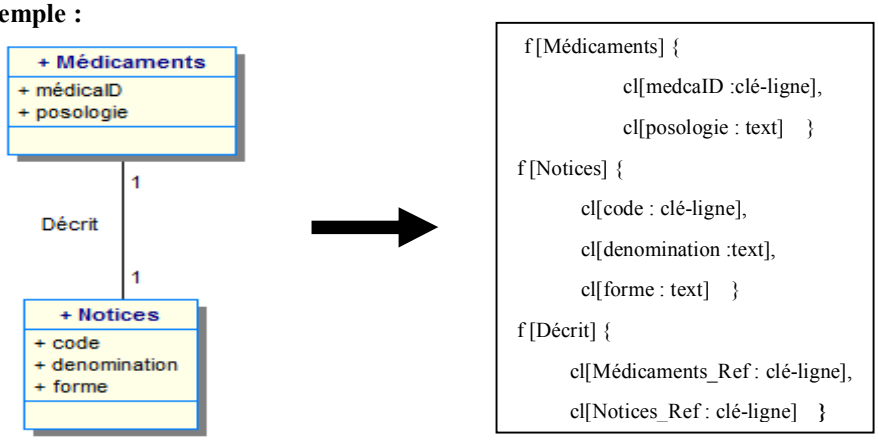
- R2 : Class → ColumnsFamily, les attributs de la même classe sont regroupés dans une seule famille ; le nom de la famille correspond au nom de la classe d'origine. Le stockage des données orienté colonnes sur le disque étant organisé par famille de colonnes, cette solution privilégie les requêtes qui utilisent simultanément les attributs de la classe.

- R3 : Oid →clé-ligne, les lignes décrites par une famille de colonnes représentent des objets (instances) de la classe correspondante. Ainsi, un identificateur de type « Oid » qui permet d'identifier les objets d'une classe est traité comme un identificateur de type « clé-ligne » qui permet d'identifier les lignes décrites par une famille de colonnes.

- R4 : Classe-Associations → ColumnsFamily, comme toute classe, une classe d'association est transformée en une famille de colonnes ; chaque colonnes correspond à un attribut de la classe soit à un attribut de type « Oid » qui référence une des classes liées.

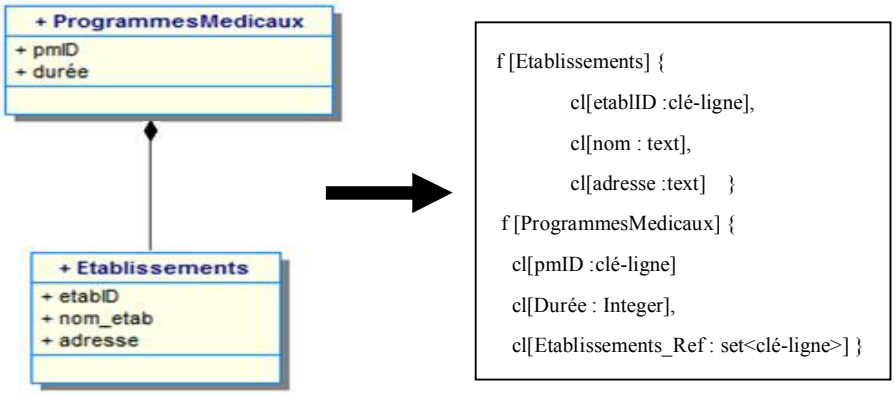
- R5 : Association →ColumnsFamily, chaque lien d'association n-aire se traduit par une nouvelle famille de n colonnes. Chaque colonne est associée à un type clé-ligne : une telle colonne référence une famille cible (classe liée). Ce principe de transformation s'applique à tout lien d'association quel que soit son degré (nombre de classes participantes) et quel que soit ses cardinalités. Nous avons eu ici le souci de généraliser le processus de traduction des liens d'association.

Exemple :



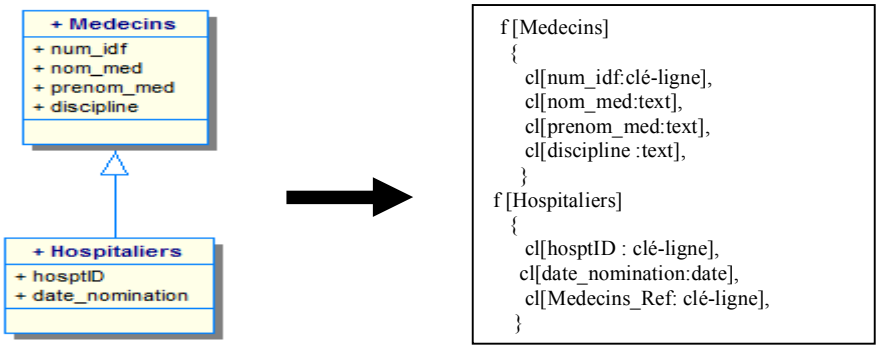
- R6 : Composition →NewColumn, une relation de composition est traduite soit par une imbrication soit par l'utilisation d'une référence. Comme le modèle orienté colonnes ne permet pas la présentation des données imbriquées (par exemple une colonne contenant d'autres colonnes), nous avons choisi l'utilisation des références. La composition/agrégation associe une classe composite et des classes composantes, tel que toute classe composante appartient à une et une seule classe composite. C'est donc une association 1..* (voire 1..1), nous la transformons alors comme suit : Tout lien de composition ou d'agrégation entre une classe composite et des classes composantes se traduit par l'ajout d'une nouvelle colonne de type « set clé-lignes » référençant les classes composantes dans la famille correspondante à la classe-composite.

Exemple :



- R7 : Héritage → NewColumn, nous proposons de transformer chaque lien d'héritage par l'ajout d'une nouvelle colonne de type « clé-ligne » dans la famille correspondante à la sous-classe ; cette colonne a pour rôle de référencer la super-classe.

Exemple :



La formalisation des règles précédentes avec QVT (Query/View/Transformation) permet un passage automatique entre le schéma conceptuel et le schéma logique de notre démarche. QVT est un langage déclaratif standardisé par l'OMG. Une transformation QVT entre deux modèles candidats est spécifiée grâce à un ensemble de relations. Chaque transformation est composée des éléments suivants : « Domains », « Relation Main », une clause « When » et une clause « Where ». Nous présentons ci-dessous les principales règles de transformation en QVT graphique.

Relation « Main ». Cette relation est le point d'entrée du processus de transformation. La partie gauche de la figure 4 montre les éléments du modèle source (uml : UML) transformés en éléments du modèle logique Orienté Colonne (OC : OrientéColonne) présenté par la partie droite de la figure. Un package est transformé en une table de même nom. La clause "where" spécifie que les classes et les associations sont transformées en familles de colonnes.

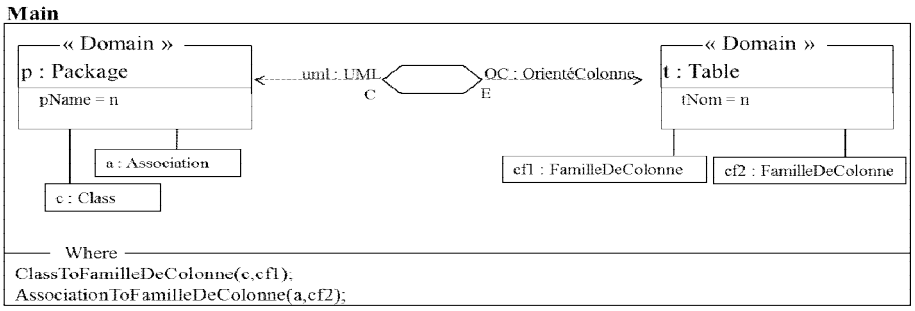


Figure 5. Relation Main de la transformation du PIM conceptuel en PIM logique

Relation « Classe en Famille de colonnes ». Une classe est transformée en une famille de colonnes dont le nom correspond au nom de la classe. Tous les attributs de la classe sont transformés en colonnes de la famille de colonnes en appliquant la relation « AttributeToColonne ». Les associations auxquelles participe la classe sont transformées en famille de colonnes.

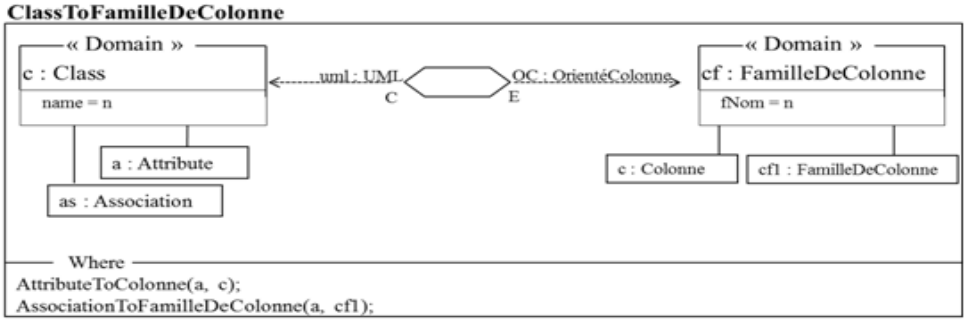


Figure 6. Relation de transformation de classe en famille de colonnes

Relation « Attribut en colonne ». Après avoir transformé les classes en familles de colonnes (relation "ClassToFamilleDeColonne" de la clause when), tous les attributs de cette classe sont transformés en colonnes de la famille ayant le même nom. Les types de ces attributs correspondent aux types des colonnes.

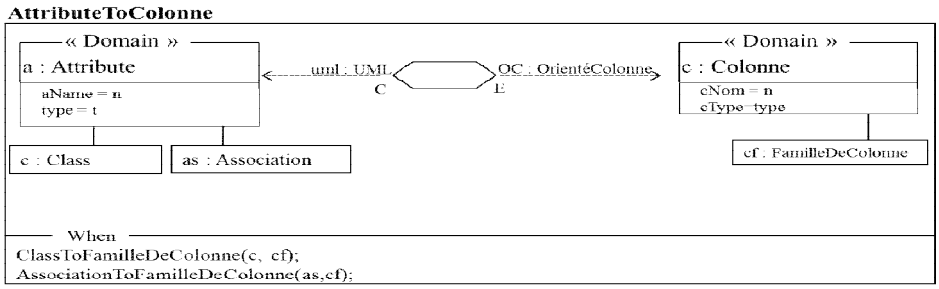


Figure 7. Relation de transformation d'attributs en colonnes

Relation « Associations en Famille de colonnes ». Une fois les classes qui participent à l'association transformées en famille de colonnes (la précondition « ClassToFamilleDeColonne » de la clause « When »), l'association est convertie en une famille de colonnes ayant le même nom. Les classes qui participent à cette association sont par la suite traduites en colonnes de la nouvelle famille de colonne correspondant à l'association (relation "ClassToColumn" de la clause "Where"). De même, les attributs de l'association sont aussi convertis en colonnes de la famille de colonnes.

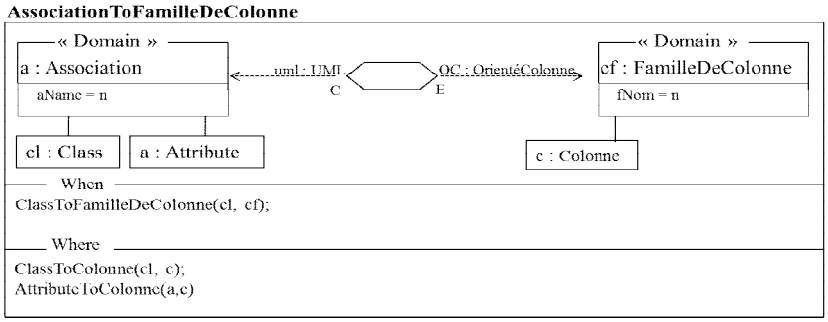


Figure 8. Relation de transformation d'association en famille de colonnes

4.2.2. Modèles physiques

Un PIM logique permet de générer plusieurs PSM associés à des plateformes distinctes ; ce principe assure l'indépendance du niveau logique face aux spécificités techniques des systèmes NoSQL ainsi qu'à leurs évolutions. Nous présentons brièvement les deux plateformes d'implantation : Cassandra et HBase, qui sont compatibles avec le PIM logique proposé. Mais, dans la mesure où cet article est consacré à la transformation du PIM conceptuel vers le PIM logique, nous ne décrivons pas le passage du PIM logique vers les PSM.

4.2.2.1. PSM HBase

HBase est un système NoSQL orienté colonnes et développé au-dessus du système de fichiers HDFS (Hadoop Distributed File System) de la plateforme Hadoop [12]. Une base

de données HBase est par défaut composée d'une seule table notée HTable (l'administrateur peut modifier ce paramètre pour créer plusieurs tables). Une HTable est associée à un nombre fixe de familles de colonnes devant être spécifiées à la création de la HTable. Seul le nom de la famille est précisé sans mention des noms de colonnes. Chaque famille est un regroupement logique de colonnes qui seront ajoutées au moment de l'insertion des données. Chaque ligne (ou enregistrement) au sein d'une HTable est identifiée par une clé notée RowKey et choisie par l'utilisateur. Au triplet (RowKey, famille de colonnes, colonne) correspond une cellule unique qui contiendra une valeur.

4.2.2.2. PSM Cassandra

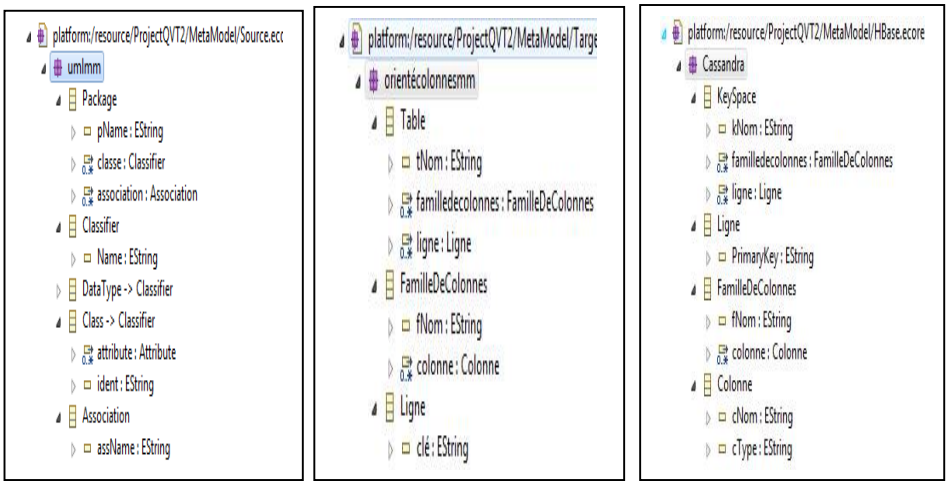
Cassandra est un SGBD NoSQL orienté colonnes, initialement basé sur le modèle BigTable de Google, mais qui emprunte également des caractéristiques au système Dynamo d'Amazon[8]. Une base de données Cassandra est par défaut composée d'un seul conteneur de données noté KeySpace. Ce dernier est associé à une ou plusieurs familles de colonnes, chacune d'elles est un regroupement logique de lignes. Une ligne est composée d'un ensemble de colonnes et est identifiée par une clé notée PrimaryKey. Chaque colonne est représentée par un quadruplet correspondant à un nom, un type, une valeur et un timestamp.

Les concepts « Table » et « colonne de type clé-ligne » vont correspondre respectivement aux concepts HTable et RowKey sous HBase et par KeySapce et PrimaryKey sous Cassandra.

5. Implantation

Dans cette section, nous décrivons les techniques que nous avons utilisées pour mettre en œuvre la démarche présentées dans la figure 2. Etant donné que notre approche est dirigée par les modèles, nous avons utilisé un environnement technique adapté à la modélisation, la métamodèlisation et la transformation des modèles. Nous avons eu recours à la plateforme Eclipse Modeling Framework (EMF) qui utilise Ecore pour créer et manipuler les modèles. La figure 9 illustre les métamodèles Ecore utilisés par notre module de transformation : PIM conceptuel (a), PIM logique (b) et PSM Cassandra (c).

[8] https://aws.amazon.com/fr/documentation/dynamodb/



(a) ***(b)*** ***(c)***

Figure 9. Métamodèles implantés avec Ecore

Notre choix du langage de transformation a été fondé sur des critères spécifiques à notre démarche. En effet, 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 opérationnel. La figure 10 montre un extrait de code QVT assurant la génération du PIM logique et du PSM Cassandra à partir du PIM conceptuel ; les commentaires figurant dans le code précisent les règles utilisées.

```
modeltype UML uses "http://umlmm.com";
modeltype OrientéColonnes uses "http://orientécolonnesmm.com";
transformation TransformationUmlToNoSQL(in Source: UML, out Target: OrientéColonnes);

main() {
Source.rootObjects()[Package] -> map PackageToTable();
}

mapping Package::PackageToTable():Table{
tNom := self.pName;

familledecolonnes:=self.classes -> map toColumnFamilyC();

familledecolonnes:=self.association -> map toColumnFamilyA();
} -- Transformation de Package en Table

 mapping UML::Class::toColumnFamilyC():OrientéColonnes::FamilleDeColonnes{
     fNom:=self.Name;
     colonne:=self.attribute -> map toColumnC();
}
 mapping UML::Attribute::toColumnC():OrientéColonnes::Colonne{
     cNom:=self.attName;
     cType:=self.attType;
} -- Transformation de Classes en Familles de colonnes

mapping UML::Association::toColumnFamilyA():OrientéColonnes::FamilleDeColonnes{
     fNom:=self.assName;
     colonne := self.links -> toColumnA()
}
mapping UML::Class::toColumnA():OrientéColonnes::Colonne{
     cNom:=self.Name + ".Ref";
     cType:="clé-ligne";
} -- Transformation d'Associations n-aire en Familles de colonnes
```

Figure 10. Extrait du code QVT

L'étape suivante vise à instancier le métamodèle du PIM conceptuel ; un exemple de cette instance est présenté dans la figure 11.

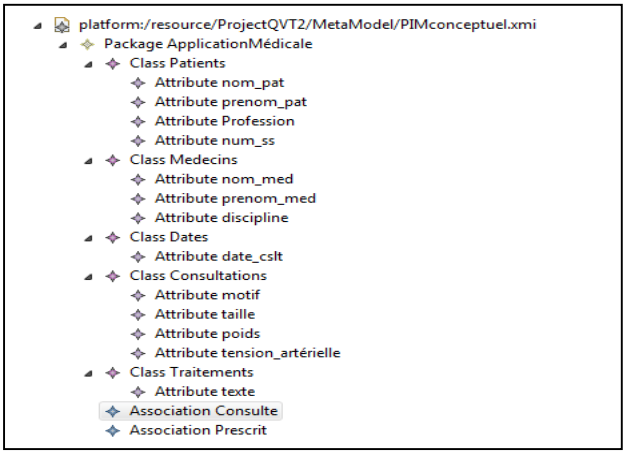
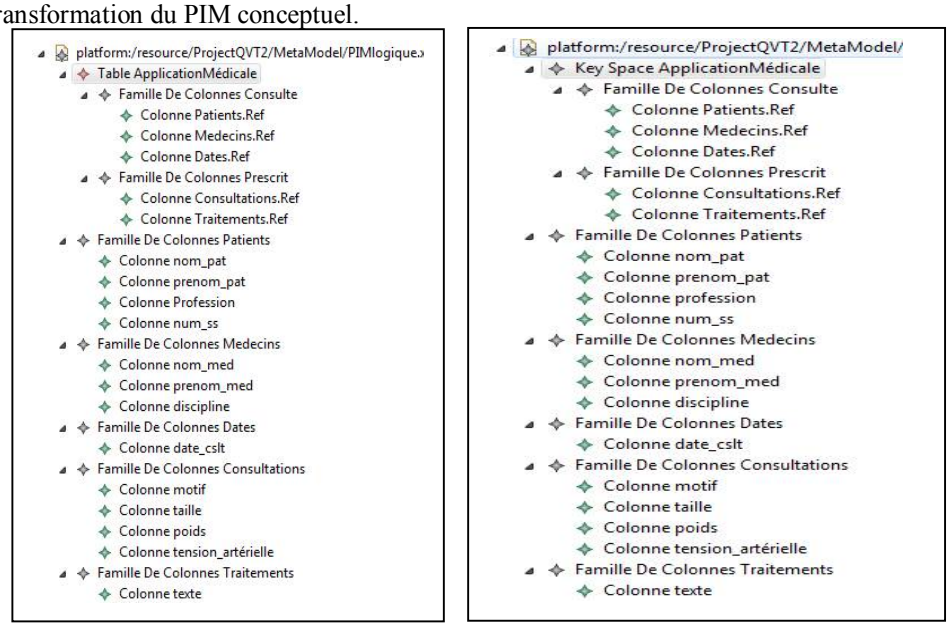


Figure 11. PIM conceptuel entré par l'utilisateur

La figure 12 montre le PIM logique (a) et le PSM Cassandra (b) obtenus par transformation du PIM conceptuel.



(a) ***(b)***

Figure 12. Modèle générés par le système : PIM logique (a) et PSM Cassandra (b)

6. Positionnement de nos travaux

Nos travaux traitent de la transformation d'un schéma conceptuel des données, représenté par un DCL d'UML, en un schéma NoSQL orienté colonnes. Nous positionnons nos travaux au regard de trois articles de recherche dont les problématiques et/ou les solutions proposées sont proches des nôtres. L'article de Chevalier et al. [11] s'inscrit dans le contexte de l'entreposage des données puisqu'il étudie les règles de passage d'un schéma multidimensionnel en un schéma physique ; deux plateformes NoSQL ont été retenues : le système orienté colonne HBase et le système orienté document MongoDB. Bien que le point de départ du processus (un schéma multidimensionnel) se situe au niveau conceptuel, ce schéma ne présente pas les mêmes caractéristiques qu'un DCL d'UML ; notamment, il comporte exclusivement des classes Faits et Dimensions et un type de lien unique entre ces deux classes. L'article de C. Li [4] traite de la transformation d'un schéma relationnel en un schéma orienté colonne HBase. Ces travaux répondent bien aux attentes concrètes des entreprises qui, face aux évolutions récentes de l'informatique, souhaitent stocker leurs bases de données actuelles dans des systèmes NoSQL. Mais la source du processus de transformation, ici un schéma relationnel, ne présente pas la richesse sémantique que l'on peut exprimer dans un DCL (notamment grâce aux différents types de liens entre classes : agrégation, composition, héritage, …). Les travaux de Y. Li et al. présentés dans [14] ont pour objet de spécifier un processus de transformation MDA d'un schéma conceptuel (DCL) vers un schéma physique HBase. Ce processus ne propose pas un niveau intermédiaire (le niveau logique) qui permettrait de rendre le résultat indépendant d'une plateforme particulière.

7. Conclusion

Nos travaux s'inscrivent dans le cadre de l'évolution des bases de données vers les Big Data, ceci pour prendre en compte le volume, la variété et la vélocité des données présentes dans les nouvelles applications liées à la transformation digitale des entreprises. Nos études portent actuellement sur les mécanismes de stockage des données dans des systèmes NoSQL. Dans cet article, nous avons traité du processus de transformation d'un schéma conceptuel représenté par un DCL d'UML en un schéma physique NoSQL orienté colonne. Pour automatiser ce processus, nous avons utilisé l'approche MDA pour créer des transformations successives entre un DCL, un schéma logique NoSQL et un schéma physique spécifique à une plateforme NoSQL. Selon notre approche, le schéma logique constitue un niveau intermédiaire qui fait abstraction de considérations techniques propres aux plateformes d'implantation et qui apparaîtront uniquement dans le schéma physique ; ce principe permet de rendre le niveau logique indépendant des évolutions technologiques des plateformes. Le point de départ du processus de transformation MDA est un métamodèle de DCL proposé par l'OMG. Les règles de transformation basées sur ce métamodèle et qui permettent de produire un schéma logique NoSQL, ont été exprimées en langage QVT. Nous avons expérimenté notre démarche et nos modèles sur une application du domaine médical qui porte sur des programmes pluriannuels de suivi de pathologies. Nous avons automatisé le processus de transformation d'un DCL décrivant une base de

données en un schéma NoSQL orienté colonne. Ce schéma a été implanté sur les systèmes HBase et Cassandra. Actuellement, nous poursuivons nos travaux en prenant en compte les spécificités du modèle de données dans les systèmes NoSQL orientés graphes.

Bibliographie

[1] Abhinay B. Angadi, Akshata B. Angadi, Karuna C. Gull. “Growth of New Databases & Analysis of NOSQL Datastores”. International Journal of Advanced Research in Computer Science and Software Engineering. 2013.

[2] A. Tanasescu, O. Boussaïd, F. Bentayeb. “Preparing Complex Data for Warehousing”. 3rd ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 05). Cairo, Egypt, January 2005.

[3] B.Combemale. “Ingénierie Dirigée par les Modèles (IDM) - Etat del'art” .2008.

[4] C. Li. “Transforming relational database into HBase : A case study”. In International Conference on Software Engineering and Service Sciences (ICSESS), pp. 683–687. IEEE.2010.

[5] C. L. P. Chen, C. Zhang. “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data”. Inf. Sci., 275, 2014.

[6] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T, Chandra, A. Fikes, R.E. Gruber. “Bigtable: a distributed storage system for structured data,”. ACM Trans. Comput. Syst. 26(2), 2008.

[7] J. Darmont, O. Boussaid, J. Ralaivao, K. Aouiche. “An Architecture Framework for Complex Data Warehouses”. 7th International Conference on Enterprise Information Systems (ICEIS 05), Miami, USA, May 2005.

[8] J.Han, E. Haihong, G. Le, J. Du.“Survey on NoSQL Database”. Pervasive Computing and Applications (ICPCA), 6th International Conference on, 2011.

[9] J. Miller, J. Mukerji. Model Driven Architecture (MDA) 1.0.1 Guide. Object Management Group, Inc., June 2003.

[10] META Group (devenu Gartner). “3D Data Management: Controlling Data Volume, Velocity, and Variety”. February 2001.

[11] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, R. Tournier. “Entrepôts de données multidimensionnelles NoSQL”. EDA 2015, 161-176.

[12] M. Grover, T. Malaska, J. Seidman, G. Shapira. “Hadoop Application Architectures”. O'Reilly, 2015.

[13] X. Blanc, O. Salvatori "MDA En Action : Ingénierie Logicielle Guidée Par Les Modèles". Paris : Eyrolles, 2005.

[14] Yan Li, Ping Gu, Chao Zhang. “Transforming UML Class Diagrams into HBase Based on Meta-model. Information Science”. Electronics and Electrical Engineering (ISEEE), 2014.

[15] http://www.omg.org/spec/QVT/1.1/.