

SISRO^{M2C} : Un Outil de Modélisation Conceptuelle à base Ontologique d'un entrepôt de données

Selma Khouri*, Ladjel Bellatreche**, Chimène Fankam**

*Ecole Nationale Supérieure en Informatique, Alger – Algérie
s_khouri@esi.dz

*LISI/ENSMA – Université de Poitiers, Futuroscope 86960 France
(fankamc, bellatreche)@ensma.fr

Résumé. Concevoir un entrepôt de données passe souvent par trois niveaux de modélisation: conceptuel, logique et physique. Contrairement aux deux derniers niveaux, le niveau conceptuel n'a pas eu le même intérêt de la part de la communauté des entrepôts de données, et peu de méthodes de conception d'un entrepôt ont été proposées. Délaisser cette modélisation a fortement contribué à l'échec de plusieurs projets d'entrepôt. Dans cet article, nous proposons une méthode, appelée SISROM2C et un outil de conception multidimensionnelle d'un entrepôt de données qui suppose l'existence d'une ontologie de domaine décrite en OWL (Web Ontology Language) couvrant toutes les sources participant dans le processus de construction de l'entrepôt de données. Elle confronte les besoins et les sources à priori, au niveau ontologique, et offre plus de présence des concepteurs dans la génération du schéma conceptuel.

1 Introduction

Un entrepôt intègre des données hétérogènes provenant de sources réparties. Avant de les charger, les données doivent être extraites, nettoyées, transformées selon le format cible du schéma de l'entrepôt. La plupart des travaux sur les entrepôts de données se concentrent principalement sur les niveaux logique (dé-normalisation) et physique (la sélection des techniques d'optimisation). Générer directement à partir d'un cahier des charges des schémas logique et physique sans passer par une modélisation conceptuelle pose plusieurs problèmes : (i) les schémas logique et physique n'offrent pas un niveau d'abstraction indépendant de toute implémentation, (ii) ils ne facilitent pas une communication entre les concepteurs et les utilisateurs qui est primordiale dans la conduite des projets, et est reconnue comme un risque important du rejet du projet final (Gam et Salinesi, 2006) et (iii) ils omettent la détection des éventuelles erreurs de conception qui doivent être décelées et corrigées dès le commencement du projet.

Pour donner plus d'importance à la modélisation conceptuelle dans la construction des projets d'entrepôt de données, trois approches de conception existent : *ascendantes*, *descendantes* et *mixtes*. Les approches *ascendantes* (Golfarelli et al., 1998), (Moody et Kortnik, 2000) ont été les premières à être proposées. Elles découlent de la définition d'Inmon (Inmon, 1992), qui indique que le développement des entrepôts repose sur les données des sources par opposition au développement des bases de données traditionnelles

Méthode de modélisation d'un entrepôt de données à base d'ontologie

(de type OLTP (OnLine Transaction Processing)) qui reposent sur une spécification des besoins (Bonifati et al. 2001). En conséquence, ces dernières ignorent les besoins utilisateurs. Les approches *descendantes* (Gam et Salinesi, 2006) prennent en compte les besoins utilisateurs, mais ignorent les sources de données qui sont le cœur de la construction de l'entrepôt de données. Les méthodes *mixtes* réconcilient les deux approches, i.e., elles prennent en compte les sources et les besoins, mais elles nécessitent une phase de *confrontation entre le schéma* généré à partir des sources et celui obtenu à partir des besoins. La confrontation entre les sources et les besoins est souvent faite sur la seule base terminologique. Elle est souvent effectuée à *posteriori*, ce qui induit la génération d'un ensemble de schémas *inutiles* causant une perte de temps (Annoni et al., 2006). Quelques travaux comme (Annoni et al., 2006) confrontent les besoins aux schémas des sources dès le début de la conception.

En analysant les différents travaux de conception existants, nous dégageons certaines limites : (i) la difficulté de conception, (ii) l'hétérogénéité des sources, (iii) l'automatisation de la démarche de construction et (iv) la difficulté d'utilisation ultérieure de l'entrepôt.

- 1) *Difficulté de conception* : Dans les méthodes orientées sources, les sources doivent être détaillées, présentées dans des schémas de données suffisamment clairs et compréhensibles, spécialement si la méthode de génération du schéma est manuelle. Il sera dans le cas contraire très difficile d'effectuer une analyse détaillée. Les étapes des méthodes orientées besoins se présentent généralement de manière déstructurée. Dans les méthodes mixtes, les correspondances entre les sources et les besoins sont généralement faites manuellement par le concepteur. Certains travaux suggèrent d'utiliser un *thésaurus* afin de les guider mais sans détailler davantage.
- 2) *Hétérogénéité des sources de données* : un entrepôt de données, par définition, doit intégrer des données provenant de différentes sources d'informations réparties et hétérogènes. Une méthode de conception doit donc gérer cette hétérogénéité qui peut être de nature structurelle ou sémantique.
- 3) *Automatisation de la démarche de construction*: La plupart des méthodes proposées (orientées besoins, sources ou mixtes) sont généralement manuelles, quelques unes sont semi-automatiques. Apporter de l'automatisation dans une démarche de conception est primordiale pour les entrepôts alimentés par un grand nombre de sources. Une autre conséquence de cette automatisation est de faciliter l'intégration de nouvelles sources de données.
- 4) *Difficulté d'utilisation ultérieure de l'entrepôt de données généré* : malgré les problèmes d'hétérogénéité soulignés précédemment, le modèle conceptuel d'un entrepôt doit permettre d'exprimer à la fois les besoins applicatifs et la connaissance du domaine sous une forme intelligible pour un utilisateur ultérieur. Malheureusement, c'est le modèle logique (issu de diverses règles de transformations comme la normalisation des tables de dimensions) qui est exploité, et qui est en général très différent du modèle conceptuel. Ainsi, l'absence de représentation du modèle conceptuel dans l'entrepôt de données rend l'interrogation de celle-ci très difficile même pour les utilisateurs ayant une bonne connaissance du domaine d'application.

Pour remédier à certains problèmes cités ci-dessus, (Romero et Abello, 2007) présentent une méthode semi automatique de conception *dirigée par les sources* basée sur une ontologie de domaine. Ce travail est le premier et le seul travail à avoir utilisé des ontologies dans la

conception des entrepôts. L'utilisation des ontologies permet de résoudre le problème d'hétérogénéité des sources de données. En plus des inconvénients liés aux méthodes dirigées par les sources, elle repose sur les multiplicités entre les concepts de l'ontologie, ce qui ne donne pas toujours des résultats fiables.

Dans ce papier, nous présentons une approche *mixte* dirigée par des ontologies, appelée SISRO^{M2C} (*Spécialisation des classes, Importation Sélective des propriétés, Représentation des Ontologies et génération du Modèle Multidimensionnel Conceptuel*), où la confrontation entre les besoins et les sources est faite a priori. Notons que cette méthode est, à notre connaissance, la première méthode permettant d'effectuer le rapprochement entre les sources et les besoins d'un entrepôt de données au niveau ontologique.

De plus en plus d'ontologies sont développées dans de nombreux domaines. Il est intéressant d'utiliser une ontologie (dans le cas où elle existe) pour faciliter le processus de conception d'un entrepôt de données. On verra tout au long de cet article, comment une méthode de conception à base d'ontologies peut contribuer à résoudre les problèmes identifiés ci-dessus.

Ce papier est divisé en six sections : la section 2 introduit notre approche SISRO^{M2C} offrant une modélisation conceptuelle d'un entrepôt de données. La section 3 décrit en détail les étapes de notre approche. La section 4 décrit la mise en œuvre de SISRO^{M2C}. La section 5 conclut le papier en récapitulant les résultats principaux, en présentant les apports de notre approche et en suggérant des travaux futurs.

2 Conception Ontologique des Entrepôts de Données

Une ontologie permet de définir de façon formelle, explicite, référencable et consensuelle, l'ensemble des concepts partagés d'un domaine sous forme de classes, de propriétés et de relations qui les lient (Gruber, 1993; Fankam et al., 2008). Les ontologies ont largement contribué dans la conception des systèmes d'intégration (Skoutas et Simitis, 2006 ; Wache et al., 2001). Un entrepôt de données peut être vu comme un système d'intégration matérialisé avec des concepts multidimensionnels. La présence des ontologies permet de résoudre les différents conflits (de nommage, de mesure, de contexte, etc.) rencontrés lors du processus d'intégration (Goh et al., 1999). Quelques systèmes ont été développés autour de cette hypothèse comme le projet COIN pour échanger les données financières (Goh et al., 1999) et le projet OntoDaWa pour intégrer les données techniques dans le domaine de l'ingénierie (Bellatreche et al., 2006). Ces études nous ont permis de faire le constat suivant : une automatisation complète de construction d'un système d'intégration n'est possible qu'à deux conditions : (1) chaque source doit représenter explicitement la *signification de ses propres données* ; c'est la notion d'ontologie locale qui doit exister dans chaque source ; et (2) il doit exister une ontologie partagée (ontologie de domaine) et chaque ontologie locale doit référencer *a priori* explicitement l'ontologie partagée pour définir *les relations sémantiques existant entre les concepts des ontologies locales et globale (articulation)*.

Dans (Fankam et al., 2008), nous avons développé une méthode de conception de bases de données à partir d'ontologies, appelée SISRO (*Spécialisation des classes, Importation Sélective des propriétés et Représentation des Ontologies*). Elle se base sur la définition d'une ontologie conceptuelle locale définie à partir d'une ontologie de domaine par spécialisation de ces concepts et propriétés, ainsi que l'extension de cette ontologie locale par de nouveaux

Méthode de modélisation d'un entrepôt de données à base d'ontologie

concepts et propriétés éventuelles jugés nécessaires. Le modèle conceptuel de la base de données à construire est défini comme l'ensemble ou un fragment de cette ontologie locale. Cette base de données une fois développée, contiendra une représentation de cette ontologie locale et son articulation avec l'ontologie partagée. Aucun outil n'a été développé dans le cadre de SISRO. La démarche de conception SISRO peut être étendue afin de prendre en compte les concepts multidimensionnels et les besoins utilisateurs pour générer le modèle conceptuel.

3 La méthode SISRO^{M2C}

La méthode SISROM2C est basée sur une ontologie partagée de domaine supposée *préexistante*. Les hypothèses de SISRO^{M2C} sont : (1) l'existence d'une ontologie de domaine qui est référencée par toutes les sources participant dans le processus de construction d'un entrepôt et (2) un ensemble de besoins utilisateurs exprimés sur l'ontologie de domaine. En sortie, SISRO^{M2C} génèrera un modèle conceptuel décrivant les différentes entités et les associations en soulignant les concepts multidimensionnels (faits et dimensions).

3.1 Les étapes de SISRO^{M2C}

Les étapes principales de SISRO^{M2C} sont décrites dans la Fig. 1 : (i) la sélection d'une ontologie de domaine, (ii) la génération de l'ontologie locale, (iii) la génération du modèle conceptuel et (iv) la validation du modèle conceptuel. Une fois le modèle conceptuel validé, le passage du schéma conceptuel au schéma logique s'effectue par les règles de traduction habituelles en des schémas en étoile, en flocon de neige ou des schémas en constellation.

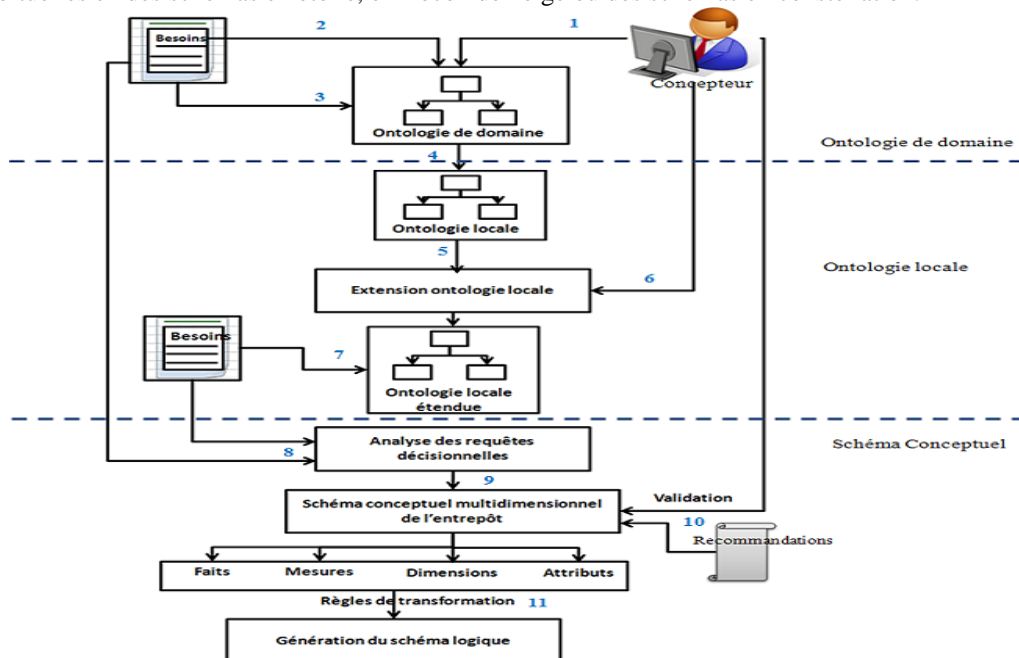


FIG. 1 – Démarche globale de notre méthode de conception.

3.1.1 Sélection d'une ontologie de domaine

L'ontologie à sélectionner doit porter sur le domaine d'intérêt de l'entrepôt à développer. Le concepteur n'a donc pas à maîtriser le domaine couvert par l'ontologie ce qui présente un gain de temps considérable. FIG. 2 présente un exemple d'ontologie de domaine Cyc¹ décrite en OWL. Nous avons adapté une partie des concepts et propriétés de Cyc de façon à ce qu'elle représente le domaine des transactions commerciales dans une organisation. Cette ontologie est utilisée tout au long de cet article afin d'illustrer les différentes étapes de notre approche. Nous rappelons que la méthode SISROM2C n'est pas spécifique à un langage particulier (OWL ou autre). Certaines classes (en pointillé) sont dupliquées pour des raisons de lisibilité.

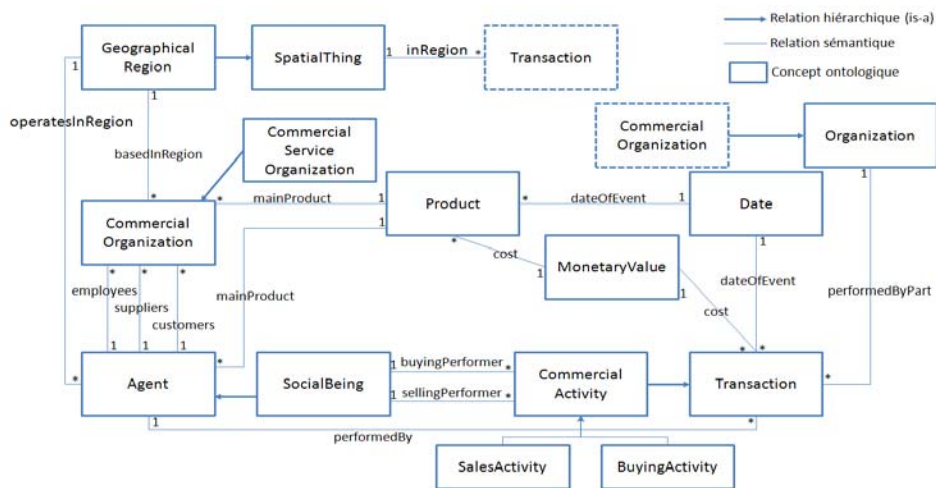


FIG. 2 – Ontologie de domaine des transactions commerciales.

3.1.2 Génération de l'ontologie locale

Le concepteur doit pouvoir concevoir le modèle conceptuel de l'entrepôt selon les données et objectifs propres à l'entreprise. Ces données sont donc plus spécifiques que celles définies au niveau de l'ontologie de domaine. On doit ainsi permettre au concepteur de construire sa propre ontologie locale à partir de l'ontologie globale, en important et en spécialisant les concepts, attributs, et relations de l'ontologie globale. Ceci permet également d'offrir une autonomie du modèle conceptuel par rapport à l'ontologie de domaine (qui peut évoluer sans incidence sur le modèle conceptuel). Cette ontologie locale est considérée comme le schéma conceptuel initial (qui n'est pas encore annoté par les éléments multidimensionnels). La définition de l'ontologie locale passe par quatre sous étapes : (a) l'expression des besoins décisionnels sur l'ontologie de domaine, (b) la construction de

¹ Cyc (<http://www.cyc.com/cyc>) est une ontologie représentant une base de connaissances générale, contenant des milliers de concepts et d'assertions sur ces concepts. OpenCyc (<http://www.cyc.com/cyc/opencyc>) est la version open source de Cyc. La version OWL de OpenCyc (2008) contient 54932 concepts, 11676 propriétés et 600203 instances.

Méthode de modélisation d'un entrepôt de données à base d'ontologie

l'ontologie locale, (c) l'extension de l'ontologie locale et (d) l'expression des besoins décisionnels sur l'ontologie locale.

Exprimer les besoins décisionnels sur l'ontologie de domaine : Les besoins décisionnels sont formulés par des requêtes SQL, en utilisant les concepts et propriétés ontologiques, comme suit :

```
SELECT Propriétés_des_classes_ontologiques, Fonction_d'agrégation (Propriétés ontologiques)
FROM Classes de l'ontologie
WHERE Condition_de_sélection (appliqués sur les propriétés ontologiques)
GROUP BY Propriétés_de_groupement_des_classes_ontologiques
```

L'utilisation du langage SQL sert uniquement à structurer et à formaliser les besoins exprimés par les décideurs. Chaque besoin est saisi par le concepteur en utilisant une interface lui permettant de choisir uniquement les concepts et propriétés représentant le besoin. Il n'a donc pas à maîtriser le langage SQL. Des contrôles sont effectués concernant les propriétés appliquées aux différentes clauses.

Les clauses *WHERE* et *GROUP BY* sont facultatives et peuvent ne pas être spécifiées. Les classes de la clause *FROM* sont ajoutées automatiquement en fonction des propriétés utilisées dans les autres clauses (*SELECT*, *WHERE* ou *GROUP BY*). Si un attribut est sélectionné dans une de ces clauses, le domaine de cet attribut est automatiquement ajouté dans le *FROM*. Si une relation est sélectionnée, les classes domaines et rangs de cette relation sont ajoutés dans le *FROM*. Nous rappelons que les propriétés dans une ontologie peuvent être de type *attribut* ou *relation*. Ces propriétés possèdent un domaine et un rang. Le domaine d'un attribut ou d'une relation est l'ensemble des classes pour lesquelles cet attribut ou relation est applicable. Le rang d'un attribut est le type de données de cet attribut (Chaîne, Entier, Réel, Booléen, ...). Le rang d'une relation représente la classe cible de cette relation. D'autres classes supplémentaires peuvent être ajoutées au niveau de la clause *FROM*.

Pour illustrer nos propos, on prend l'exemple des deux besoins décisionnels (a et b) suivants :

- *Liste des fournisseurs et Total (Coût transactions), par client et par région_client, de la date '12_03_2008'. (Besoin a)*
- *Total (Coût d'achat) de l'année 2008 par Fournisseur. (Besoin b)*

Ces besoins peuvent être exprimés en utilisant les concepts et propriétés de l'ontologie de domaine (FIG. 2), respectivement par les requêtes suivantes :

```
SELECT Suppliers, Sum(cost(SalesActivity))
FROM SalesActivity, MonetaryValue, CommercialOrganization, GeographicalRegion,
Agent, Date
WHERE dateOfEvent = '12_03_2008'
GROUP BY operatesInRegion, Customers
```

```
SELECT Sum(cost(BuyingActivity))
FROM BuyingActivity, MonetaryValue, Date, Agent, CommercialOrganization
WHERE dateOfEvent = '2008'
GROUP BY Suppliers
```

Construction de l'ontologie locale : la construction de l'ontologie locale se fait en analysant les différents besoins décisionnels exprimés. L'ensemble des classes et propriétés intervenant pour la définition des besoins décisionnels est importé de l'ontologie de domaine et est utilisé pour construire l'ontologie locale automatiquement. Importer une classe ou une propriété signifie que cette classe ou propriété est une spécialisation de sa classe d'origine

dans l'ontologie de domaine (FIG. 3). Il faut aussi garder une trace de la classe de référence (de l'ontologie de domaine) au sein de l'ontologie locale. Ceci se fait en faisant référence à la classe de l'ontologie de domaine (sans l'importer explicitement). Ceci peut se faire en langage OWL par exemple par le maintien de différents *espaces des noms*. L'articulation entre l'ontologie locale et les ontologies de domaine doit également être gardée. Ceci se fait en créant un nouvel attribut pour chaque classe locale enregistrant l'identifiant de sa classe de référence. Cette articulation avec l'ontologie de domaine sera utile par la suite dans le cas où l'on souhaite intégrer ou interroger sémantiquement des entrepôts de données construits par différents concepteurs, à partir de la même ontologie de domaine.

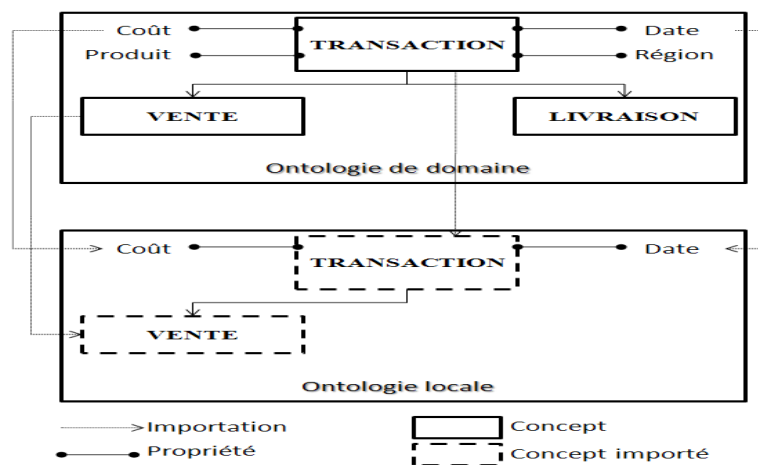


FIG. 3 – Définition de l'ontologie locale à partir de l'ontologie de domaine.

La construction du schéma de l'ontologie locale se fait en respectant les deux contraintes suivantes : (i) Pour chaque classe sélectionnée (intervenant dans la définition d'un besoin), il faut importer la classe, mais également importer ses superclasses en parcourant la hiérarchie des classes (jusqu'à la classe racine de l'ontologie). (ii) Pour chaque classe sélectionnée, on vérifie les contraintes et restrictions définies sur cette classe. Ces restrictions se présentent dans une ontologie OWL sous forme de formules logiques (implication, équivalence, ...). Si une restriction est définie sur une classe ou propriété importée dans l'ontologie locale, et si cette restriction implique une autre classe ou propriété, il faut aussi importer ces classes et propriétés au niveau de l'ontologie locale si elle n'existe pas encore dans cette ontologie.

L'importation de la hiérarchie des classes (les superclasses) dans l'ontologie locale permet d'assurer une structure cohérente de l'ontologie locale tout en sauvegardant la sémantique des concepts qu'elle définit. L'importation des classes sur lesquelles des restrictions sont définies permet de valider et de raffiner le schéma conceptuel final. Si par exemple, un concept X est inclus au niveau du schéma conceptuel, et qu'une restriction « *Concept Y équivaut au Concept X* » est définie, alors un message est fourni au concepteur, sous forme de recommandation, sur la possibilité d'inclure le concept Y au niveau du schéma conceptuel. Les restrictions que permet de définir une ontologie OWL peuvent être des *contraintes sur propriétés* (contraintes de *valeurs* (quantificateur universel, existentiel et *restriction hasvalue*), contraintes de *cardinalités* (*maximale* ou *minimale*)), ou des

Méthode de modélisation d'un entrepôt de données à base d'ontologie

contraintes sur classes (l'intersection, l'union et la complémentarité, les classes équivalentes et disjointes). Une fois le modèle conceptuel généré à partir de l'ontologie locale, un ensemble de messages ou recommandations, tirés à partir de ces restrictions, est présenté au concepteur, afin de raffiner le modèle conceptuel obtenu. Il appartient au concepteur de prendre en compte ces recommandations, d'inclure les classes et propriétés proposées au niveau du modèle conceptuel ou de les rejeter.

En reprenant l'exemple précédent, les deux requêtes (besoins a et b) font intervenir les concepts et propriétés de l'ontologie de domaine (FIG. 2), qui permettront de constituer l'ontologie locale, dont le schéma est présenté ci-dessous :

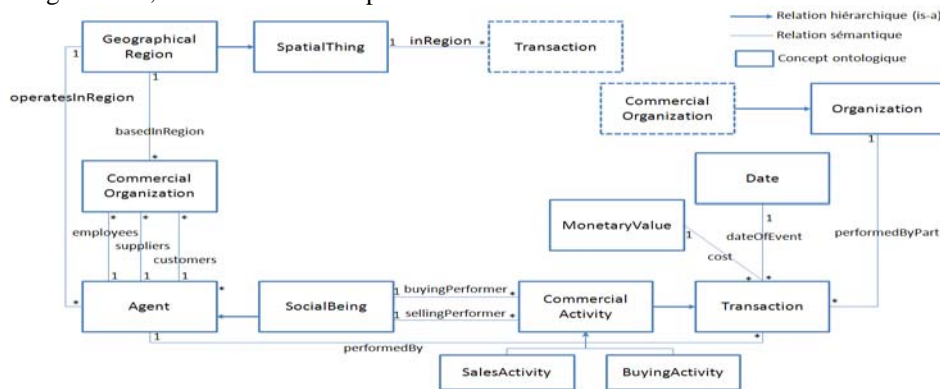


FIG. 4 – Ontologie locale définie à partir de l'ontologie de domaine.

Extension de l'ontologie locale : Il est possible que cette ontologie locale définie ne couvre pas complètement le domaine de l'application cible, il est donc indispensable de permettre d'étendre cette ontologie. Cette extension s'effectue en permettant au concepteur :

De créer de nouvelles classes au niveau de l'ontologie locale. Dans l'exemple précédent qu'on a pris (FIG. 4), la classe *Agent* étant trop générale, on peut la spécialiser par la création de deux sous classes *Customer* et *Supplier*.

De créer de nouvelles propriétés (attributs ou relations) aux classes (existantes ou ajoutées) de l'ontologie locale.

De renommer les classes et propriétés de l'ontologie locale pour refléter au mieux le domaine de l'application. Le renommage des concepts impliquent un renommage des concepts utilisés dans les requêtes décisionnelles. Ce renommage n'a aucune incidence sur l'ontologie, puisque chaque concept possède un identifiant unique. Si on prend l'exemple du cas d'une entreprise de téléphonie mobile, *dateOfEvent* peut être renommé en *DateAppel* et *Transaction* en *Appel*.

La suppression des classes ou propriétés n'est permise que sur les classes et propriétés étendant l'ontologie locale, et pas les classes et propriétés initiales, importées de l'ontologie de domaine en analysant les requêtes décisionnelles. Cette limitation est nécessaire, car ces classes et propriétés sont indispensables pour définir le schéma conceptuel de l'entrepôt.

Exprimer les besoins décisionnels sur l'ontologie locale étendue : Le concepteur peut exprimer de nouveaux besoins décisionnels sur l'ontologie locale étendue, en utilisant les concepts et propriétés ontologiques initiaux ou ajoutés pour enrichir l'ontologie locale. Ceci

se fait de la même manière que pour les besoins exprimés sur l'ontologie de domaine en suivant le même formalisme de requêtes.

En reprenant l'exemple précédent, on peut ré exprimer les deux besoins *a* et *b* sur l'ontologie locale en utilisant les classes (*Supplier* et *Customer* au lieu de la classe *Agent*), qui ont été créées pour étendre l'ontologie locale. Ceci permet d'exprimer des besoins décisionnels plus spécifiques.

3.1.3 Génération du modèle conceptuel

Une fois l'ontologie locale validée et les besoins décisionnels exprimés, l'ensemble ou uniquement une partie de cette ontologie est utilisée pour représenter un schéma conceptuel *initial* de l'entrepôt de données. La conception du modèle conceptuel de l'entrepôt s'effectue en deux étapes : (1) l'identification des concepts multidimensionnels (les faits, les mesures, les dimensions, et les attributs de dimensions) et (2) la définition du schéma conceptuel.

Identification des concepts multidimensionnels : L'identification des concepts ou éléments multidimensionnels constituant le modèle conceptuel se fait à partir de l'ontologie locale validée. Ceci s'effectue en exploitant les caractéristiques des requêtes décisionnelles en suivant les étapes décrites ci-dessous :

Pour chaque besoin exprimé :

- Les propriétés présentes dans les fonctions d'agrégation sont candidates à représenter des mesures de faits, et les propriétés de la clause Group by sont candidates à représenter des attributs de dimensions.
- Les propriétés présentes dans la clause Select (pas ceux utilisés dans les fonctions d'agrégation) et non présentes dans la clause Group by sont candidates à représenter des mesures de faits.
- Les propriétés de la clause WHERE, si elles ne sont pas encore identifiées, sont considérées comme attributs de dimension.
- A partir des propriétés identifiées comme attributs de fait ou de dimension, les classes de la clause FROM, contenant des propriétés identifiées comme mesures sont candidates à représenter des Faits et les classes contenant des propriétés identifiées comme attributs de dimensions sont candidates à représenter des Dimensions (Si la propriété est un attribut, on considère uniquement sa classe domaine. Si la propriété est une relation, on considère les classes domaine et rang.). De cette manière toutes les classes de la clause FROM sont identifiées comme concept *Fait* ou *Dimension*.
- L'algorithme a permis jusque là d'identifier un ensemble de Faits et de Dimensions et leurs attributs. Les relations entre ces classes de l'ontologie locale sont ensuite analysées afin de relier les dimensions au fait correspondant. La liaison entre un fait et ses dimensions se fait en analysant les multiplicités des relations entre les classes 'fait' et les autres classes 'dimensions'. Pour chaque classe 'Fait', s'il existe une relation (1,n) entre cette classe et une classe Dimension (ou avec une de ces superclasses), cette dernière est considérée comme dimension de la classe fait.
- Si après l'analyse des multiplicités des relations, un fait ne trouve aucune dimension à laquelle il est relié, ce schéma (contenant un seul fait) est rejeté. Si deux classes sont identifiées dans un schéma et que ces classes sont reliées par une relation (is-a), seule la classe la plus spécialisée est considérée dans le schéma. Si deux schémas,



Méthode de modélisation d'un entrepôt de données à base d'ontologie

produits par des besoins différents, partagent des faits ou des dimensions, alors ces schémas sont combinés.

- Ainsi, chaque besoin décisionnel génère un ou plusieurs schémas multidimensionnels (fait et ses dimensions). Il est possible qu'une classe se trouve être candidate à représenter un fait dans un schéma et une dimension dans un autre schéma. Dans ce cas, on analyse la fréquence d'apparition (nombre d'apparition) $f1$ des propriétés de cette classe dans les clauses Select de toutes les requêtes par rapport à sa fréquence d'apparition $f2$ dans la clause Group By. Si $f1$ est supérieure à $f2$ alors le concept est maintenu comme fait, si $f1$ est inférieure à $f2$, le concept est maintenu comme dimension, sinon si $f1$ est égale à $f2$, cette contradiction est signalée au concepteur qui doit valider les schémas.

Définition du schéma conceptuel : A partir de l'étape précédente (*Identification des concepts multidimensionnels*), on a pu identifier :

- Des classes de l'ontologie locale candidates à représenter des faits,
- Des classes de l'ontologie locale candidates à représenter des dimensions,
- Des propriétés de l'ontologie locale identifiées comme mesures de faits,
- Des propriétés de l'ontologie locale identifiées comme attributs de dimensions,
- L'association entre les classes candidates à représenter des faits et leurs Dimensions possibles (liées par des relations (1,n)).

Une fois ces éléments multidimensionnels identifiés, il est possible de construire le schéma conceptuel de l'entrepôt de données. Pour cela, nous représentons ce schéma conceptuel en utilisant le diagramme des classes de la notation UML. Dans ce diagramme de classes, on s'inspire de (Lujan-mora et Trujillo, 2003) et on distingue deux types de classes : les classes *Fait* et les classes *Dimension*. On associe l'icône  aux classes *Fait*, et l'icône  aux classes *Dimension*. Chaque classe *Fait* possède des attributs qui sont des mesures du fait qu'elle représente. Chaque classe *Dimension* possède des attributs qui sont les attributs de la dimension qu'elle représente. Les classes *Fait* et *Dimension* sont reliées entre elles par des associations représentant des relations (1,n).

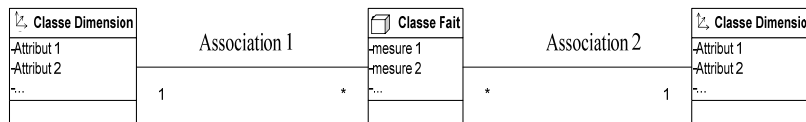


FIG. 5 – Diagramme de classes représentant les concepts multidimensionnels.

Afin de constituer le schéma conceptuel de l'entrepôt de données, nous suivons les étapes décrites ci-dessous :

- Chaque classe de l'ontologie locale identifiée comme fait devient une classe *Fait* au niveau du schéma conceptuel,
- Chaque classe de l'ontologie locale identifiée comme dimension devient une classe *Dimension* au niveau du schéma conceptuel,
- Chaque propriété de l'ontologie locale identifiée comme mesure d'un fait est associée à la classe de l'ontologie locale qui lui est applicable, et représente ainsi une mesure de la classe *Fait* correspondante au niveau du schéma conceptuel,

- Chaque propriété de l'ontologie locale identifiée comme attribut d'une dimension est associée à la classe de l'ontologie locale qui lui est applicable, et représente ainsi un attribut de la classe Dimension correspondante au niveau du schéma conceptuel,
- Chaque relation (1,n) de l'ontologie locale liant une classe fait à ses dimensions possibles, représente une association (1,n) au niveau du schéma conceptuel.

En reprenant les exemples précédents des besoins décisionnels (a et b), et en appliquant l'algorithme d'identification des concepts multidimensionnels, on obtient le schéma conceptuel multidimensionnel suivant :

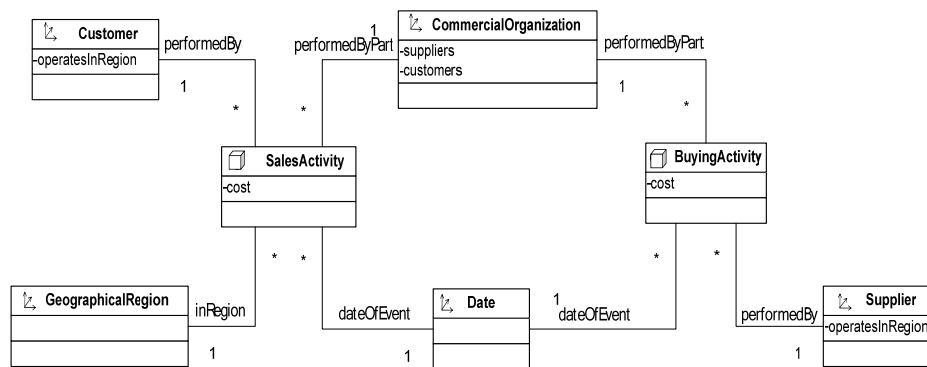


FIG.6 – Schéma multidimensionnel final obtenu (des besoins a et b).

3.1.4 Validation du modèle conceptuel

Le schéma conceptuel multidimensionnel obtenu doit être validé par le concepteur qui a le droit de le modifier. Il lui est ainsi possible de rajouter ou de changer les faits et les dimensions, et leurs attributs. Cette validation est accompagnée par un ensemble de recommandations permettant de raffiner le schéma conceptuel obtenu, tirées à partir des restrictions et contraintes définies sur les classes et propriétés de l'ontologie locale. Ces recommandations sont de la forme :

Concernant les contraintes de valeurs :

- Si une classe est définie dans le modèle conceptuel présentant le domaine d'une propriété P
- Si une contrainte de valeur de propriété ($\exists, \forall, hasValue$) est définie sur la propriété P ,

Alors proposer la classe correspondante (définie dans la restriction) pour compléter le modèle multidimensionnel.

Concernant les contraintes de cardinalités

- Si une classe est définie dans le modèle conceptuel
- Si une restriction de cardinalité ($maxCardinality, minCardinality$ ou $cardinality$) existe sur une des propriétés de cette classe

Alors proposer cette cardinalité sur la propriété correspondante

Concernant la restriction d'union sur classes

- Si une classe de dimension est définie dans le modèle conceptuel
- Si une restriction d'union est définie sur cette classe

Alors proposer les classes constituant l'union comme hiérarchies de la dimension.

Méthode de modélisation d'un entrepôt de données à base d'ontologie

Ces restrictions, dans le cas où elles sont définies sur des classes dimensions, peuvent être utilisées pour distinguer les hiérarchies et niveaux de dimensions. Par exemple si une classe Date est définie comme l'union des classes Jour, Mois et Année. Ceci permet de distinguer dans la dimension Date, la hiérarchie Jour, Mois et Année.

Concernant la restriction d'équivalence sur classe

- *Si une classe est définie dans le modèle conceptuel*
 - *Si sa classe équivalente est définie dans modèle conceptuel,*
- Alors proposer de remplacer une des classes par sa classe équivalente

Le passage au schéma logique permet d'explicitier les entités utilisées et les relations entre ces entités. Selon la manière dont sont stockées les données, on peut avoir une modélisation ROLAP ou MOLAP. Chaque schéma conceptuel multidimensionnel est traduit en un schéma en étoile. Si les tables de dimensions sont éclatées selon les hiérarchies de dimensions, on obtient des schémas en flocon de neige. La concaténation entre les différents schémas multidimensionnels obtenus partageant des dimensions similaires permet d'obtenir un schéma en constellation.

4 Mise en œuvre de SISRO^{M2C}

Afin de valider notre approche, nous proposons un outil permettant d'assister le concepteur dans le développement du schéma conceptuel tout en interagissant avec ce dernier. L'outil proposé est une application graphique développé avec l'IDE de développement JBuilder. Cet outil comporte toutes les étapes de notre approche (décrite dans la Section 3).

Le concepteur choisit une ontologie de domaine (son URI ou son adresse locale) qui sera à la base de la définition du schéma conceptuel. Il est possible de choisir des adresses de plusieurs ontologies. L'ensemble des ontologies seront importées automatiquement au sein d'une seule ontologie (cette importation permet uniquement l'utilisation de plusieurs ontologies simultanément, elle n'effectue pas une intégration de plusieurs ontologies en une ontologie cible). La visualisation de l'ontologie de domaine, et de l'ontologie locale une fois construite, s'effectue en parcourant les classes de l'ontologie et en les affichant sous forme d'une hiérarchie. Pour chaque classe, l'outil propose l'ensemble des informations relatives à ce concept (nom de la classe, documentation de la classe, attributs et relations applicables à cette classe, ses instances, et ses superclasses). L'accès à cette ontologie se fait à travers l'API de développement Java fournie par l'éditeur d'ontologies Protégé.

Le concepteur exprime ces besoins décisionnels sur l'ontologie de domaine et également sur l'ontologie locale une fois celle-ci construite, validée et étendue. Ces besoins sont exprimés en respectant la syntaxe de requêtes présentée plus haut, à travers une interface lui permettant de sélectionner un à un les concepts et propriétés à utiliser, ou bien en sélectionnant un fichier contenant ces requêtes décisionnelles. Dans ce cas, des contrôles sont effectués pour vérifier la bonne syntaxe de chaque requête.

L'outil propose le schéma de l'ontologie locale construite automatiquement à partir des besoins décisionnels exprimés sur l'ontologie de domaine. Une fois l'ontologie locale validée, elle est sauvegardée sous forme d'un fichier OWL à un emplacement spécifié par le concepteur.

Le schéma conceptuel est présenté sous forme d'un arbre regroupant les faits. Chaque fait lui est associé ses mesures, et ses dimensions possibles ainsi que leurs attributs (FIG. 7, Zone

a). La sélection d'un fait permet la visualisation du schéma conceptuel multidimensionnel relatif à ce fait (le fait et ses dimensions) sous forme d'un diagramme de classes (FIG. 7, Zone b) comme présenté au paragraphe 'Définition du schéma conceptuel' (voir Section 3.1.3).

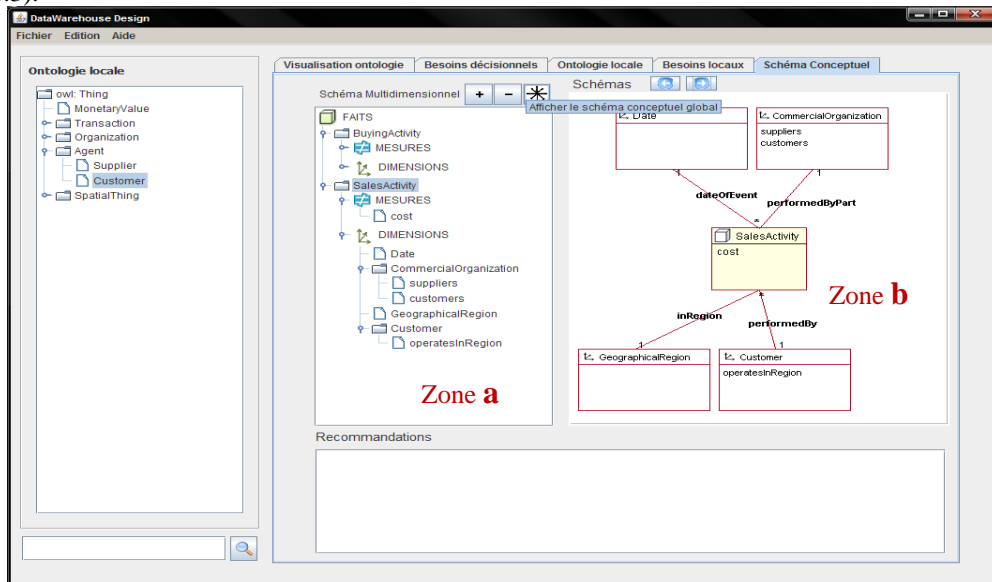


FIG.7 – Définition du schéma conceptuel de l'entrepôt de données.

Le concepteur peut valider ce schéma proposé et le modifier. Il est aidé pour cela par un ensemble de recommandations proposées par l'outil. Ainsi, le concepteur peut ajouter, modifier ou supprimer des éléments multidimensionnels s'il juge cela nécessaire (FIG. 7, Boutons + et -). Après chaque modification les schémas conceptuels multidimensionnels graphiques sont redessinés et présentés au concepteur.

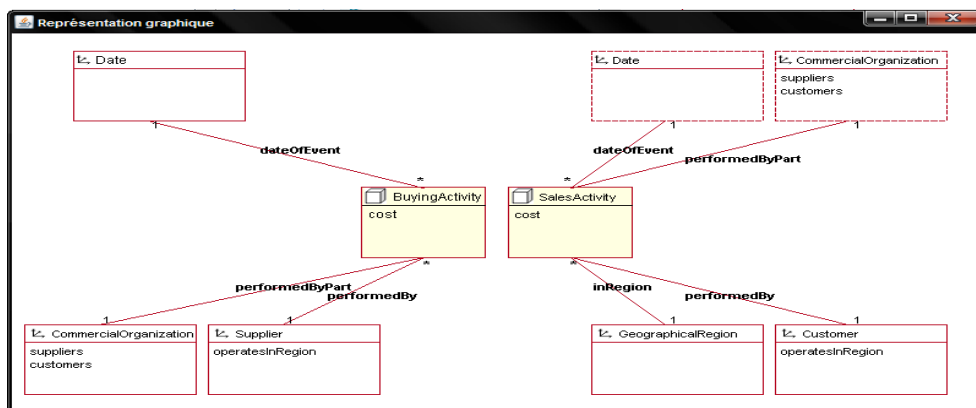


FIG. 8 – Représentation du schéma conceptuel global.

5 Conclusion & Perspectives

Les ontologies émergent dans plusieurs domaines et ont apporté leur contribution dans le processus de construction des systèmes d'intégration et des bases de données. Vue la forte similarité entre les entrepôts de données et les systèmes d'intégration, nous nous sommes inspirés de ces travaux pour proposer une démarche, baptisée SISRO^{M2C} de conception dirigée par les besoins et les sources en utilisant des ontologies de domaine. Cette démarche est conçue autour d'une hypothèse fondamentale qui est la présence d'une ontologie de domaine référencée a priori par toutes les ontologies locales des sources. Les principales étapes de SISRO^{M2C} sont : (1) la sélection d'une ontologie de domaine, (2) la génération de l'ontologie locale, (3) la génération du modèle conceptuel et (4) la validation du modèle conceptuel.

L'utilisation d'une méthode basée sur une ontologie de domaine présente plusieurs apports très intéressants. SISROM2C permet de simplifier la tâche du concepteur en lui offrant une vue consensuelle du domaine d'intérêt. Les besoins décisionnels sont exprimés au niveau ontologique et donc sémantique. Elle repose sur un schéma intégré des différentes sources hétérogènes (dans le cas où cette ontologie de domaine existe). Le caractère formel de l'ontologie facilite une automatisation partielle du processus de conception tout en offrant plus de présence aux concepteurs d'entrepôts de données lors du développement des applications d'entrepôt, où ils peuvent interagir et revoir leurs besoins. Le modèle conceptuel étant représenté au niveau connaissances (et non au niveau données), SISROM2C offre plus de sémantique aux entrepôts générés qui rend leur intégration future facile.

Dans les travaux futurs, il serait intéressant de permettre aux concepteurs d'exprimer leurs besoins en langage naturel sans passer par le langage SQL (un mapping sémantique de ces besoins sur l'ontologie de domaine permettrait de construire l'ontologie locale), de rendre l'outil plus générique en acceptant d'autres formats d'ontologie (en offrant un outil acceptant différents langages d'ontologies comme RDF ou F-logic, qui seront traduits en un langage pivot) et enfin d'étudier l'effet de cette conception sur le modèle physique.

Références

- Annoni, E., F. Ravat, O. Teste, et G. Zurfluh (2006). Méthode de Développement des Systèmes d'Information Décisionnels : Roue de Deming. *INFORSID* 657-672.
- Bellatreche, L., Xuan, D., Pierra, G. et Dehainsala, H. (2006). Contribution of Ontology-based Data Modeling to Automatic Integration of Electronic Catalogues within Engineering Databases, *Computers in Industry Journal*, 57 (8-9), pp. 711-724.
- Bonifati, A., F. Cattaneo, S. Ceri, A. Fuggetta, et S. Paraboschi (2001). Designing Data Marts for Data Warehouses. *ACM Transactions on Software Engineering and Methodology*, 10(4), pp. 452-483.
- Fankam, C., L. Bellatreche, H. Dehainsala, Y. Ait Ameer, et G. Pierra (2008). SISRO: Conception de bases de données à partir d'ontologies de domaine. A apparaître dans *Technique et science informatiques (TSI)*, pp. 1- 29.
- Gam, I. et C. Salinesi (2006). A Requirement-driven Approach for Designing Data Warehouses. *REFSQ'2006*.

- Goh, C. H., S. Bressan, E. Madnick et M.D. Siegel (1999). Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3), pp. 270–293.
- Golfarelli, M., D. Maio, et S. Rizzi (1998). Conceptual Design of Data Warehouses from E/R Schemes. in *the Proceedings of the Hawaii International Conference On System Sciences*, pp. 334-343.
- Gruber, T. (1993). A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220.
- Inmon, W.H. (1992). *Building the Data Warehouse*. John Wiley.
- Lujan-mora, S., et J. Trujillo (2003). A Comprehensive Method for Data Warehouse Design. In Proc. *DMDW*.
- Moody, D.L. et M. Kortnik (2000). From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. *DMDW'2000*, 5.
- Romero, O. et A. Abelló (2007). Automating Multidimensional Design from Ontologies. *DOLAP'07*, pp. 1–8.
- Skoutas, D. et A. Simitsis (2006). Designing ETL processes using semantic web technologies. *DOLAP 2006*, pp.67-74
- Wache, H., T. Voge, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, et S. Hubner (2001). Ontology-Based Information Integration : A Survey of Existing Approaches. Proc. Of the International Joint Conference on Artificial Intelligence Workshop: Ontologies and Information Sharing.

Summary

Designing a data warehouse usually requires three main phases: conceptual, logical and physical. The conceptual design does not get same attention compared to logical and physical design. Ignoring conceptual design when building data warehouse applications may contribute on failure of those projects. In this article, we propose a methodology, called SISRO^{M2C} and a tool for a conceptually designing a data warehouse. This approach supposes the existence of domain ontology described in OWL. Each source has its own local ontology that references the global domain ontology. Our approach belongs to hybrid category, where sources and user requirements are confronted, at the ontological level, in a priori manner.