Deuxième partie

Personnalisation des analyses

Après avoir présenté le contexte de nos travaux dans la première partie de ce mémoire, la deuxième partie est consacrée à nos contributions sur la personnalisation des analyses dans les entrepôts de données. Nous développons ces contributions selon trois chapitres.

Le chapitre 4 est dédié à la présentation d'une architecture globale qui permet l'implication de l'utilisateur dans l'évolution du schéma de l'entrepôt. Cette architecture est rendue possible grâce à un modèle d'entrepôt de données évolutif dont nous proposons une modélisation formelle. Le caractère évolutif de ce modèle est dû à l'existence d'une partie évolutive qui contient les hiérarchies de dimension de l'entrepôt qui sont susceptibles d'évoluer.

Dans le chapitre 5, nous évoquons alors en détail la mise à jour des hiérarchies de dimension. Nous y proposons également le modèle d'exécution de notre approche dans un contexte relationnel.

Enfin, dans le chapitre 6, nous nous intéressons au problème de l'évaluation de notre modèle évolutif. Nous apportons une première réponse en proposant une méthode de mise à jour incrémentale de la charge.

Modèle d'entrepôt de données à base de règles

Résumé

Dans ce chapitre, nous présentons notre approche de personnalisation des analyses en ligne. Cette personnalisation consiste en l'évolution du schéma de l'entrepôt réalisée par les utilisateurs eux-mêmes. Notre approche se base donc sur une architecture qui permet de gérer tout le processus décisionnel : acquisition des connaissances utilisateurs sous forme de règles, intégration de ces règles dans l'entrepôt, évolution du schéma de l'entrepôt et enfin l'analyse en ligne. L'évolution de schéma consiste plus précisément en l'ajout de nouveaux niveaux de granularité dans les hiérarchies de dimension existantes. Afin de soutenir cette architecture, nous proposons un modèle d'entrepôt de données évolutif à base de règles d'agrégation qui permettent l'expression des connaissances utilisateurs, ainsi que la mise à jour des hiérarchies de dimension. Par ailleurs, nous proposons un méta-modèle qui permet de représenter tout entrepôt de données évolutif, assurant ainsi la généricité de notre approche.

Sommaire

Chapitre 4

Modèle d'entrepôt de données à base de règles

4.1 Introduction

Les résultats d'une enquête menée en 2005 par le magazine CIO (magazine destiné aux décideurs informatiques) auprès de 140 grandes entreprises [IDG05] ont révélé une volonté des entreprises de «disposer d'outils souples, plus près des objectifs métiers et des usages que peuvent en faire les opérationnels». En effet, parmi les facteurs clés de succès de la mise en place d'un projet décisionnel identifiés par les entreprises interrogées, l'adéquation aux objectifs métiers et l'adhésion des utilisateurs arrivent en tête. En outre, cette enquête a également révélé qu'un tiers des entreprises envisage une extension du parc d'utilisateurs. Face à ces constats concernant à la fois le nombre d'utilisateurs et la réponse à leurs besoins, la personnalisation des possibilités d'analyse trouve un grand intérêt.

La conception de magasins de données a pour objectif de répondre aux objectifs métiers. Mais, comme il est souligné dans [RTZ07], compte tenu de la complexité de mise en œuvre des magasins de données (conception, alimentation, rafraîchissement, maintenance), il n'est pas envisageable de déployer un magasin de données pour chaque décideur.

L'adhésion des utilisateurs est donc cruciale et nécessite que leurs besoins d'analyse puissent y trouver une réponse. Mais ceci est loin d'être évident et ce, pour différentes raisons. En effet, il est difficile d'être exhaustif dans le recensement des besoins d'analyse des utilisateurs au moment de la conception du schéma de l'entrepôt. De plus, la prise en compte des besoins d'analyse lors de la phase de conception n'est pas évidente ; ceci est en partie dû à l'absence de standard pour la concep-

tion des entrepôts de données [RALT06]. En outre, il est difficile de prévoir des besoins d'analyse futurs. Or, de nouveaux besoins individuels peuvent émerger dans la mesure où les utilisateurs peuvent s'intéresser à de nouveaux objectifs d'analyse.

L'objectif général de cette thèse est donc de fournir une solution pour répondre à la personnalisation des analyses à partir d'un entrepôt de données existant et d'assurer ainsi une certaine flexibilité en terme d'évolution des possibilités d'analyse de l'entrepôt. Cette personnalisation se traduit par le fait que les utilisateurs peuvent définir de nouveaux axes d'analyse, basés sur leurs propres connaissances métier.

Afin d'atteindre cet objectif, nous proposons une démarche basée sur une architecture globale qui permet de gérer le processus décisionnel dédié à la partie analyse. Il s'agit, dans un premier temps d'acquérir les connaissances utilisateurs, puis d'intégrer ces dernières dans l'entrepôt, avant de procéder à l'évolution du schéma de l'entrepôt, permettant ainsi de nouvelles analyses.

Dans notre cas, l'évolution de schéma consiste plus précisément en la mise à jour des hiérarchies de dimension. Ainsi, le schéma de l'entrepôt n'est pas fixé lors de la phase de conception mais il évolue en fonction des nouveaux objectifs d'analyse des utilisateurs. De ce fait, pour soutenir cette architecture, et en particulier l'évolution de schéma, nous définissons un modèle formel à base de règles de type «si-alors» que nous appelons règles d'agrégation. Ce modèle, nommé modèle *R-DW* (Rule-based Data Warehouse), permet de modéliser un entrepôt de données évolutif flexible, pour la prise en compte de nouveaux besoins d'analyse.

Dans ce contexte, nous proposons également un méta-modèle qui permet de représenter le schéma de tous les entrepôts de données évolutifs, assurant ainsi la généricité de notre approche. En effet, cela rendra possible la sélection de l'entrepôt que l'on veut faire évoluer.

Ce chapitre est organisé comme suit. Tout d'abord, dans la section 4.2, nous présentons un exemple introductif, basé sur le cas réel de LCL, qui illustre l'objectif de notre approche de personnalisation. Ensuite, dans la section 4.3, nous exposons notre architecture pour la personnalisation des analyses. La section 4.4 est consacrée à la présentation de notre modèle d'entrepôt de données évolutif *R-DW*. Par la suite, nous développons, dans la section 4.5, une discussion qui a pour but de positionner notre approche par rapport à différents travaux existants et de dégager un certain nombre de remarques. Enfin, nous concluons dans la section 4.6.

4.2 Exemple introductif

Afin de bien comprendre les éléments de solutions que nous apportons au problème de la personnalisation, nous présentons ici un exemple illustratif. Cet exemple est issu du cas réel de LCL.

Nous disposons ici d'un extrait de l'entrepôt LCL-DW que nous avons conçu et construit avec les données bancaires de LCL. Cet extrait permet d'étudier le NBI (Net Banking Income), ce qui correspond à ce que rapporte un client à l'établissement bancaire. Le schéma multidimensionnel considéré est fourni dans la figure 4.1. Il permet d'analyser la mesure `NBI` selon les dimensions `CUSTOMER` (client), `AGENCY` (agence) et `YEAR` (année). La dimension `AGENCY` présente une hiérarchie.

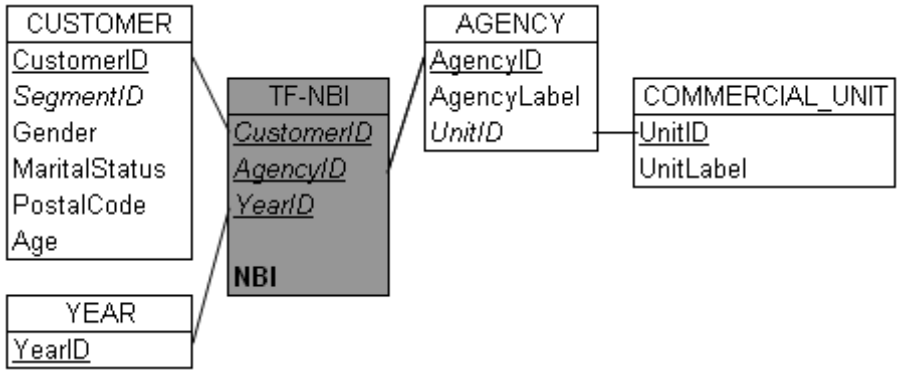

Fig. 4.1 – Schéma multidimensionnel pour observer le NBI

Il est ainsi possible d'agréger les données selon le niveau `COMMERCIAL_UNIT` (unité commerciale) qui est un regroupement d'agences par rapport à leur localisation géographique, tel que le montre le schéma de la dimension `AGENCY` de la figure 4.2a.

Supposons qu'un utilisateur veuille analyser le NBI selon le type d'agence ; il sait qu'il en existe trois : type «étudiant» pour les agences ne comportant que des étudiants, type «non résident» lorsque les agences ne gèrent que des clients ne résidant pas en France et le type «classique» pour les agences ne présentant pas de particularité. Ces informations n'étant pas présentes dans l'entrepôt, il est impossible pour lui d'obtenir cette analyse.

Nous proposons alors à l'utilisateur d'intégrer sa propre connaissance sur les types d'agence afin de créer un niveau `AGENCY_TYPE` (type d'agence). Cela passe par la mise à jour (création) de la hiérarchie de la dimension agence en ajoutant le niveau `AGENCY_TYPE` au-dessus du niveau `AGENCY` selon le schéma de la figure 4.2b. L'utilisateur pourra ainsi réaliser une analyse du NBI en fonction du type d'agence.

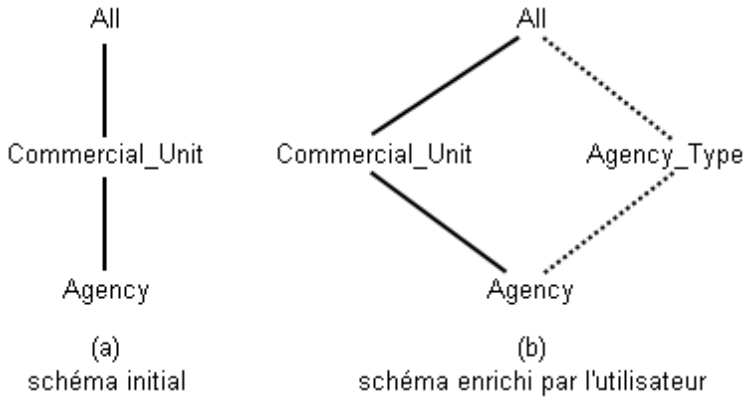

FIG. 4.2 – Schémas de la dimension `AGENCY`

4.3 Une architecture d'entrepôt pour la personnalisation des analyses

Le principe général de notre approche est de permettre une personnalisation des analyses en intégrant les connaissances utilisateurs dans l'entrepôt de données. Cette intégration se traduit par l'évolution des hiérarchies de dimension, via la création de nouveaux niveaux de granularité. Ainsi, l'architecture que nous présentons à présent implique l'utilisateur dans le processus d'évolution en lui permettant d'exprimer ses connaissances.

L'architecture globale de l'entrepôt de données évolutif guidé par les utilisateurs est présentée dans la figure 4.3. Comme nous l'avons proposé dans [FBB07e], elle se décompose en quatre modules. Le premier module est l'*acquisition* des connaissances utilisateurs. Il permet aux utilisateurs d'exprimer leurs connaissances sous la forme de règles de type «si-alors». Dans le deuxième module, il s'agit de l'*intégration* des règles dans l'entrepôt de données. Ensuite, le module d'*évolution* du schéma permet de supprimer un niveau existant ou de créer le nouveau niveau de granularité grâce aux règles, en étendant une hiérarchie de dimension existante, ou en en créant une nouvelle. Enfin, le module d'*analyse* permet de réaliser des analyses en ligne, en se basant sur le nouveau schéma de l'entrepôt.

Ainsi, il s'agit d'un processus d'évolution incrémentale dans la mesure où les nouveaux besoins exprimés par les utilisateurs font évoluer au fur et à mesure le schéma courant de l'entrepôt.

Notre architecture est centrée utilisateur, mais il est primordial que l'implication des utilisateurs ne compromette pas le schéma initial de l'entrepôt qui répond

à des besoins d'analyse globaux communs à l'ensemble des utilisateurs. Ainsi, les nouveaux besoins d'analyse ne doivent modifier ni la table des faits, ni les niveaux de granularité directement liés à celle-ci (tables de dimension). Ceci justifie le fait que nous proposons un modèle d'entrepôt de données évolutif contenant une partie fixe (modèle *R-DW*).

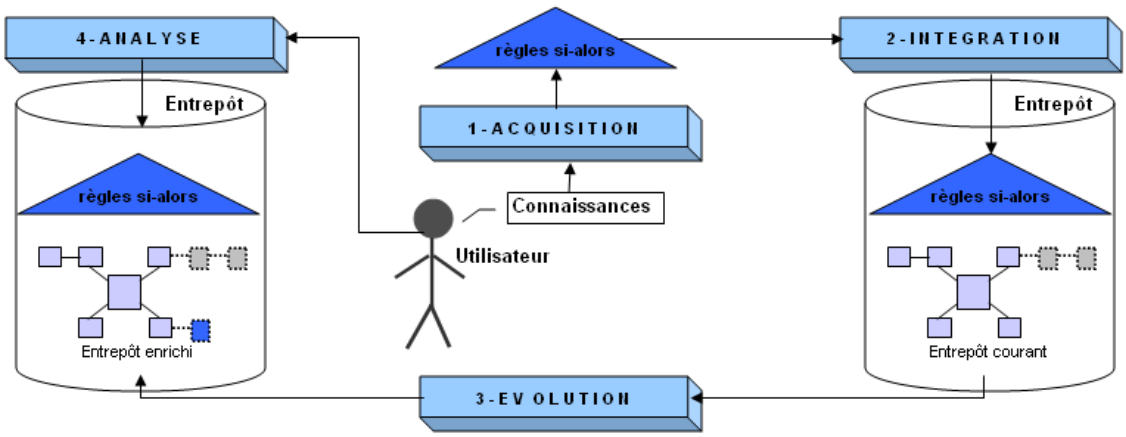

FIG. 4.3 – Architecture générale d'entrepôt de données évolutif guidé par les utilisateurs

4.4 Le modèle *R-DW*

Pour supporter l'architecture qui vise à la personnalisation des analyses, il est nécessaire de prendre en compte le fait que le schéma de l'entrepôt évolue. Il est alors crucial de disposer d'un modèle d'entrepôt évolutif, par conséquent flexible. Afin de permettre cette flexibilité, nous utilisons des règles pour exprimer la connaissance des utilisateurs. En effet, les règles et, plus particulièrement, les langages à base de règles permettent d'introduire une certaine flexibilité dans les systèmes qui les utilisent. Par exemple, Espil et al. [EV01] ont défini un langage à base de règles pour la gestion des exceptions dans le processus d'agrégation qui permet de rendre l'analyse plus flexible en redéfinissant, au niveau des instances, des chemins d'agrégation dans les hiérarchies de dimension.

Nous représentons les connaissances utilisateurs sous la forme de règles de type «si-alors». Ces règles sont très compréhensibles pour les utilisateurs puisqu'elles permettent de modéliser les connaissances de façon simple et explicite [HHNT86]. Elles constituent d'ailleurs un élément important de la plupart des théories de représentation de la connaissance en sciences cognitives. Notons qu'elles sont utilisées pour représenter les connaissances extraites avec les algorithmes d'arbres de décision, qui

font partie des méthodes d'apprentissage supervisé les plus appréciées, précisément en raison de la simplicité de compréhension de la restitution des résultats.

4.4.1 Principe du modèle *R-DW*

Dans notre approche de personnalisation, nous introduisons une flexibilité au niveau des analyses. Cette flexibilité consiste en la création d'axes d'analyse supplémentaires qui se traduit par l'ajout de nouveaux niveaux de granularité pour définir de nouvelles hiérarchies de dimension ou modifier les hiérarchies de dimension existantes. Comme nous l'avons proposé dans [FBB06c], cette flexibilité est introduite au moyen de règles de type «si-alors». Nous qualifions ces règles de règles d'agrégation. En effet, la connaissance utilisateur qu'elles vont exprimer va définir la façon d'agréger des données d'un niveau existant vers un nouveau niveau qui sera créé.

Notre modèle *R-DW* à base de règles est composé d'une partie fixe et d'une partie évolutive [FBB06b, FBB06d] (figure 4.4). La partie fixe comprend la table des faits et les dimensions qui lui sont directement reliées. La partie évolutive comprend l'ensemble des hiérarchies qui sont définies lors de la conception du schéma initial ou lors de la personnalisation des analyses.

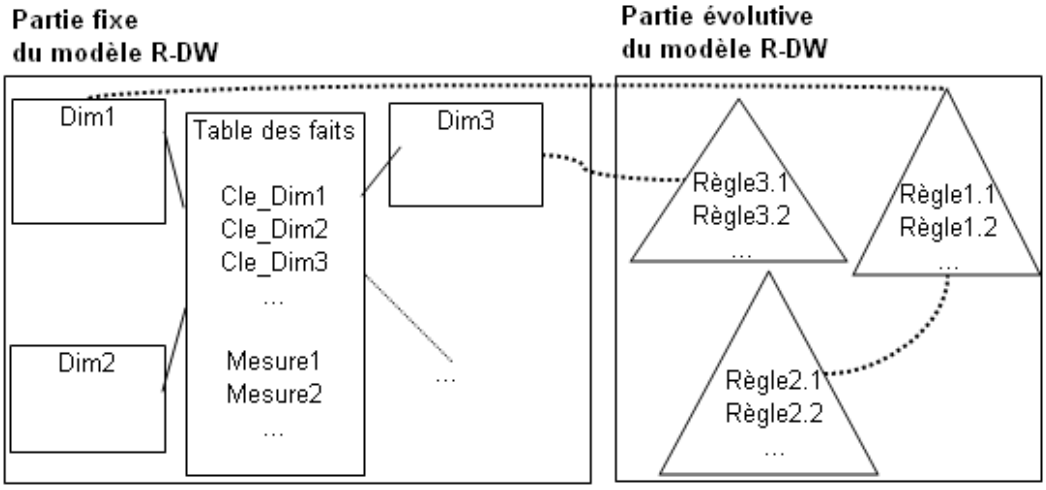

FIG. 4.4 – Principe du modèle *R-DW*

Le modèle *R-DW* ne permet pas l'ajout d'un niveau directement relié à la table des faits, qui constituerait ainsi une nouvelle dimension. Ce choix est motivé par deux aspects. Premièrement, il assure une cohérence des données stockées dans l'entrepôt. Si l'on envisage la création de dimension par les utilisateurs, cela engendrerait une

évolution dans le processus d'alimentation de l'entrepôt. En effet, une telle évolution nécessite une modification du processus ETL qui ne peut être réalisée par l'utilisateur. Deuxièmement, la partie fixe du modèle *R-DW* constitue une réponse à des besoins d'analyse initiaux, pouvant être communs aux différents utilisateurs, définis lors de la conception de l'entrepôt. La modélisation initiale fournit ainsi un schéma de l'entrepôt pouvant servir à l'ensemble des utilisateurs pour l'analyse.

Comme son nom l'indique, le modèle *R-DW* est basé sur des règles. Celles-ci vont permettre la création de nouvelles hiérarchies par ajout de niveau de granularité audessus d'une table de dimension ou l'extension des hiérarchies existantes par ajout d'un niveau de granularité en fin de hiérarchie ou au sein de celle-ci.

Les règles utilisées dans le modèle *R-DW* sont de type «si-alors». La clause «si» permet d'exprimer les conditions sur les attributs caractérisant le niveau de granularité inférieur, c'est-à-dire le niveau à partir duquel sera généré le nouveau niveau. Dans la clause «alors» figure la définition du niveau de granularité à créer, c'est-à-dire la définition des valeurs des attributs caractérisant ce nouveau niveau de granularité. Nous qualifions ces règles de règles d'agrégation puisqu'elles établissent un lien d'agrégation entre deux niveaux de granularité dans une hiérarchie de dimension.

Comme nous l'avons évoqué dans le chapitre 2, il existe différents types de liens d'agrégation au sein d'une hiérarchie de dimension [MZ04]. Nous considérons ici le cas classique, que l'on peut qualifier de hiérarchie symétrique stricte selon la typologie présentée dans [MZ04]. Ainsi on prend en considération le cas où toutes les instances d'un niveau donné ont une et une seule instance correspondante dans le niveau supérieur. Les règles exprimées par les utilisateurs doivent satisfaire deux contraintes pour que la création du niveau se réalise selon le schéma de la figure 4.5.

La première contrainte est que les clauses «si» des règles d'agrégation définissent une partition des instances du niveau inférieur. Par définition, la partition d'un ensemble est un ensemble de parties non vides de cet ensemble, deux à deux disjointes et dont la réunion est égale à l'ensemble initial. Ainsi les sous-ensembles d'instances définis par les clauses «si» des règles doivent être non vides, deux à deux disjoints et leur union correspond à l'ensemble de départ comportant toutes les instances.

La deuxième contrainte est liée au lien d'agrégation défini entre le niveau inférieur et le niveau créé. Chaque sous-ensemble d'instances de cette partition est associé à une et une seule instance du niveau créé. Les données concernant chaque instance du niveau inférieur pourront alors être agrégées en une instance du niveau créé. Ainsi la mise en correspondance entre les deux niveaux revient à l'application d'une fonction bijective entre les sous-ensembles de la partition du niveau inférieur et les instances

du niveau créé.

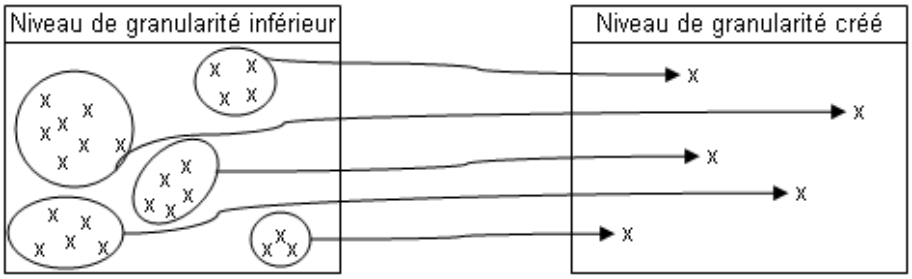

FIG. 4.5 – Définition des liens d'agrégation

Nous conservons la notation «si-alors» dans le texte, mais nous emploierons son équivalent anglais «if-then» dans les formalisations ou les exemples.

4.4.2 Formalisation du modèle *R-DW*

Dans cette section, nous présentons le formalisme du modèle *R-DW*

Nous désignons le modèle d'entrepôt évolutif à base de règles *R-DW* par le triplet suivant :

$$R\text{-}DW = (\mathcal{F}, \mathcal{E}, \mathcal{U})$$

où $\mathcal{F}$ est la partie fixe, $\mathcal{E}$ la partie évolutive et $\mathcal{U}$ l'univers de *R-DW*.

Définition 1. *Univers de l'entrepôt*
Soit $R\text{-}DW = (\mathcal{F}, \mathcal{E}, \mathcal{U})$.
L'*univers* $\mathcal{U}$ *de l'entrepôt R-DW* est un ensemble d'attributs, tel que :

$$\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$$

où $\mathcal{U}_1 = \{B_\alpha, 1 \leq \alpha \leq z\}$ est l'ensemble des z attributs existants dans le schéma initial de l'entrepôt et $\mathcal{U}_2 = \{C_\beta, \beta \geq 1\}$ est l'ensemble des attributs générés, présents dans la partie évolutive $\mathcal{E}$ de *R-DW*.

Définition 2. *Partie fixe de R-DW*
Soit $R\text{-}DW = (\mathcal{F}, \mathcal{E}, \mathcal{U})$.
La *partie fixe de R-DW* est définie par :

$$\mathcal{F} = < F, \mathcal{D} >$$

où F est une table de faits, et $\mathcal{D} = \{D_s, 1 \leq s \leq t\}$ est l'ensemble des t dimensions de premier niveau qui ont un lien direct avec F. Nous supposons que ces dimensions sont indépendantes.
Exemple. Dans la figure 4.1, < TF_NBI,{AGENCY, YEAR, CUSTOMER} > constitue la partie fixe de l'entrepôt *R-DW* pour l'analyse du NBI.

Définition 3. *Hiérarchie de dimension et niveau de granularité*
Soit *R-DW* = $(< F, \mathcal{D} >, \mathcal{E}, \mathcal{U})$.
$D_s.H_k, D_s \in \mathcal{D}, k \geq 1$ est une *hiérarchie de la dimension* D_s. La hiérarchie de dimension $D_s.H_k$ est composée d'un ensemble de w niveaux de granularité ordonnés notés L_i :

$$D_s.H_k = \{L_0, L_1, \ldots, L_i, \ldots, L_w, w \geq 0\}$$

avec $L_0 \prec L_1 \prec \cdots \prec L_i \prec \cdots \prec L_w$ où $\prec$ exprime l'ordre total sur les L_i.
Le *niveau de granularité* L_i de la hiérarchie H_k de la dimension D_s est noté $D_s.H_k.L_i$ ou plus simplement L_i^{sk}.
L_0^{sk} correspond au premier niveau de la hiérarchie, il s'agit de la dimension elle-même.
Exemple. Selon le schéma de la figure 4.1, on a :

$D_{\texttt{AGENCY}}.H_1$={AGENCY, COMMERCIAL_UNIT} ;$L_2^{\texttt{AGENCY 1}}$=COMMERCIAL_UNIT
$D_{\texttt{AGENCY}}.H_2$={AGENCY, AGENCY_TYPE} ; $L_2^{\texttt{AGENCY 2}}$=AGENCY_TYPE

Définition 4. *Partie évolutive de R-DW*
Soit *R-DW* = $(\mathcal{F}, \mathcal{E}, \mathcal{U})$.
La *partie évolutive de R-DW* est l'ensemble des hiérarchies de dimension du schéma de l'entrepôt, privé des niveaux correspondant aux dimensions elles-mêmes ; autrement dit c'est donc l'ensemble des différents niveaux de granularité composant ces hiérarchies à partir du niveau 1 :

$$\mathcal{E} = \{D_s.H_k\} - \{L_0^{sk}\} = \{L_i^{sk}\},\ avec\ 1 \leq s \leq t,\ k \geq 1,\ i > 0$$

Notons que ici, i et k ne sont pas fixés a priori puisque le principe de notre modèle est justement que de nouveaux niveaux de granularité peuvent être créés dans les hiérarchies existantes et de nouvelles hiérarchies peuvent être créées également via la création de niveaux de granularité au-dessus d'une dimension. En outre, $i \geq 1$ puisque les dimensions (niveau 0) n'appartiennent pas à la partie évolutive.

La création de niveaux de granularité engendre la génération d'attributs.

Définition 5. *Attribut généré*
Soient *R-DW* $= (< F, \mathcal{D} >, \mathcal{E}, \mathcal{U})$, L_i^{sk} le niveau de granularité L_i de la hiérarchie H_k de la dimension D_s et $\mathcal{U}_2$ l'ensemble des attributs générés dans la partie évolutive $\mathcal{E}$ de *R-DW*.
Le niveau de granularité L_i^{sk} peut être créé et est alors défini par un ensemble de λ *attributs générés* qui décrivent ce niveau :

$$L_i^{sk}.A \; avec \; A = \{a_\delta, 1 \leq \delta \leq \lambda, a_\delta \in \mathcal{U}_2\}$$

Exemple. $L_2^{\texttt{AGENCY}\ 2}.A = \{\texttt{AgencyTypeLabel}\}$

Les attributs générés sont définis par des règles d'agrégation.

Définition 6. *Règle d'agrégation*
Une *règle d'agrégation* permet de définir le lien d'agrégation qui existe entre deux niveaux de granularité dans une hiérarchie de dimension.
Elle est basée sur un ensemble $\mathcal{T}$ de n termes de règles, notés RT_p, tel que :
$\mathcal{T} = \{RT_p, 1 \leq p \leq n\} = \{u \; op \; \{ens|val\}\}$
où u est un attribut de l'univers $\mathcal{U}$ de l'entrepôt ; *op* est un opérateur ($=$, $<$, $\leq$, $\geq$, $\neq$, $\in$, ...) ; *ens* est un ensemble de valeurs et *val* est une valeur finie.
Exemple. RT_1 : `AgencyID` $\in$ {'01903','01905','02256'} ; RT_2 : `Age` > 80

Une règle d'agrégation est une règle de type «si-alors». La conclusion de la règle (clause «alors») définit la valeur des attributs générés à l'aide de conjonctions d'égalité. La prémisse de la règle (clause «si») est basée sur des conjonctions de termes de règles :

$$r_{ij} \; :$$

$$if \; RT_1 \; AND \; ... \; AND \; RT_p \; AND \; ... \; AND \; RT_n$$

$$then \; L_i^{sk}.a_1 = val_1^{ij} \; AND \; ... \; AND \; L_i^{sk}.a_\delta = val_\delta^{ij} \; AND \; ... \; AND \; L_i^{sk}.a_\lambda = val_\lambda^{ij}$$

où $val_\delta^{ij} \in Dom_{L_i^{sk}.a_\delta}$ le domaine de définition de l'attribut $L_i^{sk}.a_\delta$.
Exemple. Les règles suivantes déterminent les valeurs des attributs `AgeClass` et `AgeClassDescription` qui caractérisent le niveau de granularité `AGE` construit au-

dessus de la dimension CUSTOMER :
r_{11} : if Age $<$ 18
$then$ AgeClassDescription = 'minor' AND AgeClass = 'less than 18 years old'
r_{12} : if Age $\geq$ 18
$then$ AgeClassDescription = 'major' AND AgeClass = 'more than 18 years old'

Le principe de définition du lien d'agrégation est que les règles construisent une partition des instances du niveau inférieur. Pour satisfaire cette contrainte de partition, nous définissons trois propriétés sur les règles. La propriété 1 a pour but d'exprimer le fait que les sous-ensembles d'instances du niveau inférieur définis par les clauses «si» des règles d'agrégation ne sont pas vides.
La propriété 2 a pour but d'exprimer le fait que les sous-ensembles d'instances du niveau inférieur définis par les clauses «si» des règles d'agrégation sont deux à deux disjoints, autrement dit, l'intersection des ces sous-ensembles pris deux à deux doit être vide.
La propriété 3 a pour but d'exprimer le fait que l'union des sous-ensembles d'instances du niveau inférieur définis par les clauses «si» des règles d'agrégation correspond à l'ensemble initial de toutes les instances de ce niveau.

Propriété 1.
Soit $L_i^{sk}.A$ l'ensemble des attributs générés qui caractérisent le niveau de granularité créé L_i de la hiérarchie H_k de la dimension D_s.
Soit $\mathcal{R}_{i.}^{sk} = \{r_{ij}, 1 \leq j \leq v\}$ l'ensemble des v règles d'agrégation définissant les valeurs de l'ensemble des attributs générés $L_i^{sk}.A$
Chaque clause «si» des règles de $\mathcal{R}_{i.}^{sk}$ définit un ensemble d'instances I_{ij} dans le niveau inférieur L_{i-1}^{sk}.
On a alors :

$$\forall i, \forall j, I_{ij} \neq \emptyset$$

Propriété 2.
Soit $L_i^{sk}.A$ l'ensemble des attributs générés qui caractérisent le niveau de granularité L_i de la hiérarchie H_k de la dimension D_s.
Soit $\mathcal{R}_{i.}^{sk} = \{r_{ij}, 1 \leq j \leq v\}$ l'ensemble des v règles d'agrégation définissant les valeurs de l'ensemble des attributs générés $L_i^{sk}.A$.
Chaque clause «si» des règles de $\mathcal{R}_{i.}^{sk}$ définit un ensemble d'instances I_{ij} dans le niveau inférieur L_{i-1}^{sk}.

On a alors :

$$\forall i, \forall j, q \; tels \; que \; j < q, j \in [1, v-1], q \in [2, v], \;\; I_{ij} \bigcap I_{iq} = \emptyset$$

Propriété 3.
Soit $L_i^{sk}.A$ l'ensemble des attributs générés qui caractérisent le niveau de granularité L_i de la hiérarchie H_k de la dimension D_s.
Soit $\mathcal{R}_{i.}^{sk} = \{r_{ij}, 1 \leq i \leq w, 1 \leq j \leq v\}$ l'ensemble des v règles d'agrégation définissant les valeurs de l'ensemble des attributs générés $L_i^{sk}.A$.
Chaque clause «si» des règles de $\mathcal{R}_{i.}^{sk}$ définit un ensemble d'instance I_{ij}.
Le niveau L_{i-1}^{sk} sur lequel est construit le niveau L_i^{sk} comprend un ensemble d'instances noté I_{i-1}^{ini}.
On a alors :

$$\forall i, \bigcup_{j=1}^{v} I_{ij} = I_{i-1}^{ini}$$

Chaque sous-ensemble d'instances du niveau inférieur est mis en correspondance avec une et une seule instance du niveau créé. Ainsi, nous définissons une fonction qui détermine le lien d'agrégation entre le niveau créé et le niveau inférieur sur lequel il est basé.

Définition 7. *Fonction de correspondance*
Soit $\mathcal{C}$ une fonction de correspondance de $\mathcal{I}^{inf}$ dans $\mathcal{I}^{cre}$ où $\mathcal{I}^{cre}$ désigne l'ensemble des instances du niveau créé et $\mathcal{I}^{inf}$ désigne l'ensemble des instances du niveau inférieur. Cette fonction $\mathcal{C}$ a une propriété de bijectivité au sens où nous la décrivons.
On a alors :

$$\forall \iota \in \mathcal{I}^{cre}, \exists ! \Theta \subset \mathcal{I}^{inf}, \mathcal{C}(\Theta) = \iota$$

4.4.3 Version utilisateur

Dans notre approche, nous supposons que les utilisateurs disposent d'un entrepôt de donnée initial. Nous leur offrons la possibilité de créer de nouveaux axes d'analyse. Comme nous l'avons détaillé précédemment, ils peuvent exprimer de nouveaux besoins d'analyse en fonction de leur propre connaissance : connaissance du domaine, objectif métier, etc.

Reprenons le cas de LCL. Il est possible que deux utilisateurs aient besoin d'analyser le NBI en fonction de l'âge des clients. Selon leur métier, la définition des classes d'âge n'est pas la même pour les deux utilisateurs. L'un peut vouloir se baser sur deux catégories d'âge : les plus et les moins de 60 ans, parce qu'il travaille dans le service marketing dédié aux produits d'épargne de retraite. L'autre, qui travaille dans le service dédié aux offres étudiantes, doit par exemple distinguer les mineurs des majeurs dans ses analyses.

Ainsi nous devons faire face à des besoins d'analyse identique, en l'occurrence la définition de classes d'âge, mais avec des sémantiques différentes. Ainsi, dans [FBB06a, BFB08], nous avons proposé de gérer des versions de règles différentes. Cela correspond à gérer des versions de hiérarchies différentes.

Pour cela, nous avons introduit la notion de version de niveau de granularité qui se greffe au schéma de façon parallèle. Dans le cas d'une analyse en fonction de ce niveau, il faudra donc choisir la version qui comprend la définition souhaitée. Notons qu'il s'agit ici de considérer des versions qui dépendent d'utilisateurs différents. Nous ne traitons pas des versions temporelles.

Définition 8. *Version de niveau de granularité*
Soit L_i^{sk} le niveau de granularité L_i de la hiérarchie H_k de la dimension D_s qui peut être défini par différents utilisateurs.
Une version de ce niveau est noté :

$$L_i^{sk}(v_c),\ \ c \geq 1$$

où v_c représente le numéro de la version.
Si une seule version existe, nous adoptons la première notation : L_i^{sk}.
Exemple. Supposons que $L_2^{\texttt{CUSTOMER }2}$ correspond au niveau `AGE CLASS` de la dimension `CUSTOMER`. Deux utilisateurs définissent différemment ce niveau, nous notons : $L_2^{\texttt{CUSTOMER }2}(v_1)$ et $L_2^{\texttt{CUSTOMER }2}(v_2)$.

4.4.4 Méta-modèle

Afin de pouvoir appliquer notre démarche sur n'importe quel entrepôt de données, nous avons conçu un méta-modèle qui permet de représenter le schéma logique de l'entrepôt de données évolutif (figure 4.6).

Ce méta-modèle permet d'assurer la généricité de notre modèle d'exécution. En effet, une fois mis en œuvre, il permet de gérer plusieurs entrepôts de données évolutifs et donc sélectionner l'entrepôt de données que l'on va faire évoluer.

L'interprétation de ce méta-modèle est la suivante. Un entrepôt de données évolutif est décrit comme un ensemble de tables. Ces tables sont soit des tables de dimension, soit des tables de faits. Chaque table de dimension possède un ou plusieurs niveaux de granularité qui forment ainsi une hiérarchie de dimension. Chaque niveau possède un ensemble d'attributs et une clé primaire. Chaque niveau de granularité peut être généré par un ensemble de règles d'agrégation. Donc chaque niveau correspond soit à un ensemble d'attributs explicites, soit à des attributs générés avec des règles. Parallèlement, les tables de faits présentent une ou plusieurs mesures et une clé primaire qui est une composition des clés étrangères correspondant aux clés primaires des tables de dimension qui leurs sont directement reliées.

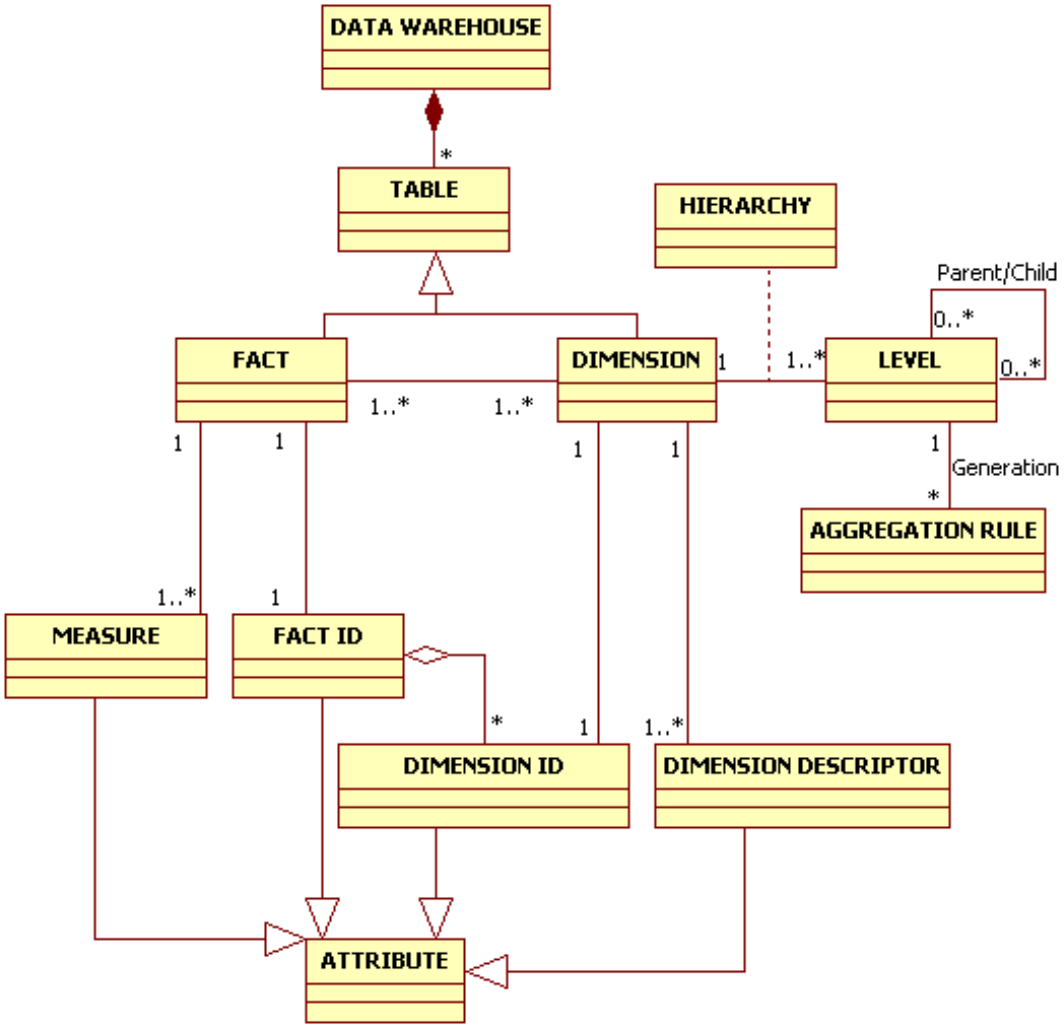

FIG. 4.6 – Méta-modèle pour gérer l'évolution du schéma de l'entrepôt

4.5 Discussion

4.5.1 Positionnement

Dans la littérature, l'évolution de schéma peut être réalisée soit par une mise à jour soit par une modélisation temporelle. Notre approche de personnalisation s'inscrit dans une alternative de mise à jour de schéma. Mais si notre approche ne gère pas une historisation des dimensions, elle ne pose néanmoins pas de problème au niveau de la cohérence des analyses.

Du point de vue de la mise à jour de schéma, elle se rapproche ainsi des travaux proposés dans [BSH99]. En effet les auteurs proposent un ensemble d'opérateurs élémentaires (ajouter un niveau, ajouter un attribut, connecter un attribut à un niveau, etc.). En combinant ces opérateurs, il est possible de réaliser l'ensemble des mises à jour du schéma. Néanmoins, ce travail consiste à représenter ces opérateurs au niveau structurel. Dans notre approche, nous nous sommes intéressés également à fournir les données nécessaires pour l'évolution, en exploitant en l'occurrence la connaissance utilisateur.

Par ailleurs, notre approche a un objectif commun avec celle de Mazon et al. [MT06] : celui d'enrichir les possibilités d'analyse des entrepôts de données en étendant les hiérarchies de dimension ; les deux approches diffèrent néanmoins sur la méthode utilisée. En effet, leur approche vise à enrichir les hiérarchies de dimension de façon automatique, en exploitant certaines relations (hypéronymie et méronymie) de WordNet. Mais dans notre cas, plutôt que d'utiliser les relations sémantiques extraites de WordNet, ces dernières sont exprimées par l'utilisateur lui-même. De ce fait, notre approche est bien orientée utilisateur, constat important dans un contexte de technologie dite centrée utilisateur.

Dans [EV01], les auteurs ont développé un langage à base de règles pour la gestion des exceptions dans le processus d'agrégation. Ce langage permet de redéfinir des chemins d'agrégation pour exprimer des exceptions dans les hiérarchies de dimension. Ainsi, on peut considérer que ce travail permet une certaine flexibilité dans le processus d'analyse, et qu'il peut répondre à une certaine forme de personnalisation. Néanmoins, si ce langage constitue une alternative à la flexibilité du schéma multidimensionnel lors du processus d'agrégation, il ne fait qu'en modifier les chemins, sans permettre la création de nouveaux chemins. C'est précisément cette limite que nous dépassons grâce à notre approche.

Enfin, concernant les travaux traitant de la personnalisation dans les entrepôts de données, il s'avère que notre approche s'inscrit dans une perspective différente

des travaux émergents dans le domaine. Deux propositions principales s'inscrivent dans cette volonté de personnalisation. D'une part, celle présentée dans [BGMM06, BGM+05] a pour objectif d'affiner la requête de l'utilisateur pour mieux répondre à ses besoins. D'autre part, celle présentée dans [RTZ07] vise à prédéterminer des analyses intéressantes pour faciliter la navigation de l'utilisateur dans les données. Ainsi ces deux travaux se basent sur l'expression de préférences pour personnaliser le processus d'analyse en diminuant les réponses aux requêtes ou en diminuant le nombre d'opérations à réaliser lors de la navigation pour obtenir un résultat.

Dans notre travail, la personnalisation n'est pas fondée sur une expression de préférences pour faire face à une multitude de possibilités en terme de résultats de requêtes ou d'opérations de manipulation des données en restreignant ces possibilités. Bien au contraire, il s'agit d'étendre ce champ pour permettre de nouvelles analyses qui soient personnalisées par rapport aux besoins des utilisateurs, en prenant en compte leurs propres connaissances.

4.5.2 Partage des analyses personnalisées

L'évolution de schéma qui vise à personnaliser les analyses permet de rendre les nouvelles possibilités d'analyse accessibles aux autres utilisateurs. Ainsi, le partage des nouvelles possibilités d'analyse constitue un des avantages de notre approche. Étant donné que ces nouvelles possibilités proviennent des utilisateurs eux-mêmes, il est crucial que les autres utilisateurs puissent avoir connaissance du contenu informationnel de ces possibilités. En effet, cette connaissance est primordiale afin de bien utiliser ces possibilités et les interprétations qui pourraient en découler lors des analyses. Cela permettrait de clarifier la sémantique du niveau d'agrégation ajouté.

Pour ce faire, nous pensons que le recours à un processus d'annotations, comme il a pu être proposé dans [CCRT07], peut être pertinent. Ainsi le créateur du nouvel axe d'analyse pourrait annoter celui-ci afin de lui donner une bonne description, pour que la compréhension soit facilitée pour les autres utilisateurs. Cette nécessité est accentuée dans le cas de versions utilisateurs différentes qui consistent à représenter un même niveau avec des règles de construction différentes (cas des classes d'âge par exemple).

4.5.3 Expression des règles par les utilisateurs

Notre approche se base sur l'expression des connaissances utilisateurs qui permettent de regrouper des instances de dimension.

L'utilisabilité de notre approche dépend alors particulièrement de la facilité d'identifier les instances dans le niveau inférieur à partir duquel sera créé le nouveau niveau et ainsi de la simplicité des règles exprimées.

La difficulté d'exprimer les règles dépend surtout du type des attributs utilisés dans les règles. En effet, l'objectif des règles «si-alors» est de regrouper des instances du niveau inférieur. Si des attributs continus sont utilisés, il sera plus simple de définir par exemple des intervalles de valeurs en exprimant des conditions (tel que sur l'attribut âge). Lorsque des attributs discrets sont utilisés, l'utilisateur doit lister chaque valeur de l'attribut de façon explicite. Si la cardinalité de l'attribut concerné est importante, la tâche peut devenir réellement fastidieuse (dans le cas d'un regroupement par rapport à l'identifiant de client par exemple). Ainsi, nous pensons qu'il est important de fournir à l'utilisateur une interface qui permette d'exprimer des règles de façon simple.

Par ailleurs, dans l'expression de conditions, nous utilisons des conjonctions de conditions. Dans un souci de simplification pour l'utilisateur, nous évitons la gestion de priorité qu'il y aurait à faire si on avait recours indifféremment à des conjonctions (opérateur AND) et des disjonctions (opérateur OR). En effet, le recours à des parenthèse pour traiter les priorités dans les opérations peut devenir très complexe à exprimer et à gérer. Il serait plus difficile pour l'utilisateur de satisfaire les contraintes liées à la définition de la partition en utilisant des disjonctions dans les règles.

4.5.4 Construction automatique de règles

Lorsque le nombre d'instances à identifier lors du regroupement devient trop important, donc la tâche trop fastidieuse pour l'utilisateur, une méthode d'apprentissage permettant un regroupement automatique des instances paraît pertinente. Par exemple, dans [RB07], les auteurs proposent d'utiliser la méthode des K-means [McQ67] pour construire les classes de regroupement d'instances pour représenter le nouveau niveau de granularité à créer.

4.6 Conclusion

Dans ce chapitre, nous avons présenté notre approche de personnalisation des analyses dans les entrepôts de données. L'architecture globale de notre approche permet d'impliquer l'utilisateur dans le processus d'évolution de schéma de l'entrepôt, évolution qui sera génératrice de nouvelles possibilités d'analyse. Cette architecture comprend des modules permettant l'acquisition des connaissances utilisateurs sous

forme de règles d'agrégation, l'intégration des règles d'agrégation dans l'entrepôt de données, l'évolution de schéma de ce dernier et enfin, l'analyse sur le nouveau schéma.

Cette architecture est soutenue par notre modèle *R-DW*, un modèle d'entrepôt de données évolutif flexible, basé sur des règles d'agrégation de type «si-alors». Ce modèle est composé d'une partie «fixe» et d'une partie «évolutive». La partie fixe est constituée de la table des faits et des tables de dimension qui lui sont directement reliées. La partie évolutive est composée d'un ensemble de hiérarchies qui peuvent exister préalablement (définies lors de la conception du modèle de l'entrepôt) et qui évoluent, ou nouvellement créées par l'application des règles d'agrégation (création de niveaux de granularité). Pour assurer la généricité de notre approche, nous avons également proposé un méta-modèle qui permet de représenter tout entrepôt de données évolutif.

Dans ce chapitre, nous avons surtout évoqué la création de niveaux de granularité qui est rendue possible par l'expression des règles d'agrégation. Mais la partie évolutive de notre modèle *R-DW* correspond de façon plus générale à une mise à jour des hiérarchies de dimension. C'est le point que nous détaillons dans le chapitre qui suit.

Mise à jour des hiérarchies de dimension

Résumé

Dans ce chapitre, nous proposons une méthode de mise à jour des hiérarchies de dimension qui permet la création et la suppression de niveaux de granularité. De plus, nous présentons une méthode de propagation de cette mise à jour lorsque le niveau de granularité concerné se situe en milieu de hiérarchie. Nous déployons par la suite notre approche dans le contexte relationnel.

Sommaire