Évolution de schémas dans les entrepôts de données : mise à jour de hiérarchies de dimension pour la personnalisation des analyses

Cécile Favre

Université Lumière Lyon 2
École Doctorale Informatique et Information pour la Société

THÈSE

pour obtenir le grade de

Docteur

en

Informatique

présentée et soutenue publiquement par

Cécile Favre

le 12 Décembre 2007

Évolution de schémas dans les entrepôts de données : mise à jour de hiérarchies de dimension pour la personnalisation des analyses

préparée au sein du laboratoire ERIC
sous la direction de
M. Omar Boussaid et Mme Fadila Bentayeb

COMPOSITION DU JURY

Mme Rokia MISSAOUI	Rapportrice	(Professeur, Université du Québec en Outaouais)
M. Gilles ZURFLUH	Rapporteur	(Professeur, Université Toulouse 1)
Mme Corine CAUVET	Examinatrice	(Professeur, Université Aix-Marseille 3)
M. Djamal BENSLIMANE	Examinateur	(Professeur, Université Lyon 1)
M. Abdelkader Djamel ZIGHED	Examinateur	(Professeur, Université Lyon 2)
Mme Fadila BENTAYEB	Co-directrice de thèse	(Maître de Conférences, Université Lyon 2)
M. Omar BOUSSAID	Directeur de thèse	(Maître de Conférences-HDR, Université Lyon 2)
M. Michel ROUGIÉ	Invité	(LCL, Direction Rhône-Alpes Auvergne)

Hommage à Nicolas

Je dédie cette thèse au Professeur Nicolas Nicoloyannis, décédé le 16 juin 2007. Je l'ai connu durant l'année universitaire 2000-2001. Sept ans déjà... mais c'est finalement si court au contact d'une telle personne. Après avoir été un de mes enseignants, il fut mon directeur de thèse. Il n'a pas participé à l'encadrement scientifique de ce travail. Mais il a tenu avec simplicité, humour et générosité ce rôle de directeur, en étant tout simplement là pour moi... Le son de sa voix si agréable qui aurait dû chanter dans le jury nous manque tant.

Quelques jours après sa disparition, j'ai écrit ces quelques strophes qui, j'espère, seront à la hauteur de l'enseignant, du chercheur, du directeur, de l'Homme qu'il était et exprimeront bien toute l'affection que j'avais pour lui...

Un très grand Homme brusquement s'en est allé
Les mots sont bien trop difficiles à trouver
Ton absence sera bien dure à surmonter
Pour nous tous aujourd'hui qui sommes là restés

C'est l'enseignant que j'ai tout d'abord rencontré
Dans ce rôle tu faisais l'unanimité
Ta pédagogie, ta disponibilité
Et ta façon d'enseigner nous ont tous marqués

La théorie des graphes un vrai jeu devenait
Tout paraissait si simple quand tu expliquais
Le recuit simulé comme du café sucré
C'est de cette façon que tu l'as enseigné !

C'est le chercheur que j'ai ensuite rencontré
Le directeur de thèse que tu as été
Tu as su, très humainement, mes pas guider
Tu as été rassurant pour m'encourager

Même si nous n'avons pas ensemble cherché
C'est l'homme tout entier que j'ai pu apprécier
Honnêteté, modestie, joie de vivre, gaieté
Simplicité, gentillesse, générosité

Ton grand cœur a cessé de battre de bonne heure
Grâce à ces valeurs que tu as su incarner
Nous trouverons la force de continuer
Tu restes près de nous, à jamais dans nos cœurs

Remerciements

En tout premier lieu, je remercie Nicolas Nicoloyannis de m'avoir accueillie au sein du laboratoire ERIC pour réaliser cette thèse et d'avoir accepté d'en être son directeur. Il nous a malheureusement quitté trop tôt pour pouvoir faire partie du jury mais son souvenir reste à jamais présent dans mon cœur.

Je tiens à remercier Fadila Bentayeb et Omar Boussaid d'avoir assuré l'encadrement scientifique de cette thèse, qui n'a pas toujours été de tout repos. Je remercie Fadila pour nos séances de travail agréables et fructueuses, ses remarques pertinentes, mais aussi pour son écoute et son discours bienveillants. Merci également à Omar. J'ai pu apprécier la confiance qu'il m'a accordée et qui m'a permis de tracer mon propre chemin et de prendre de l'assurance. Je les remercie tous deux pour le soutien, les encouragements et la confiance qu'ils m'ont insufflée sur la dernière ligne droite ...

Je tiens à mentionner le plaisir et l'honneur que m'ont fait Madame Rokia Missaoui et Monsieur Gilles Zurfluh en acceptant de rapporter ce travail. Un grand merci à eux pour leurs critiques constructives qui m'ont permis d'améliorer ce mémoire. Je remercie également Madame Corine Cauvet, Messieurs Djamal Benslimane et Djamel Zighed d'avoir accepté d'être membres de mon jury de thèse.

Je remercie également ma seconde famille dans laquelle je pense avoir réussi à trouver ma place... Il s'agit du laboratoire ERIC et du département Informatique et Statistique. Permanents ou de passage, chercheurs-enseignants ou secrétaires, encore là ou partis... toutes ces personnes que j'ai côtoyées et qui, d'un point de vue scientifique ou relationnel, de près ou de loin, m'ont apporté leur aide ou leur soutien. Je fais le choix de ne pas les citer de peur d'en oublier. Mais je tiens tout de même à remercier nominativement Valérie Gabriele, pour son dévouement professionnel et ses qualités relationnelles.

Ils font partie de cette seconde famille mais je tiens à remercier tous les doctorants et jeunes docteurs avec qui nous avons su instaurer un climat de travail agréable et fructueux. Merci surtout à ceux qui ont successivement partagé mon bureau : Amandine, Nora, Elie, Hadj, Jean-Christian, Kamel, Riadh. Merci à Ahmad, mon compagnon de nuits studieuses

au laboratoire durant la rédaction. Merci à Hakim pour son soutien de tous les instants.

Cette thèse a été menée dans le cadre d'une convention CIFRE avec la banque LCL-Le Crédit Lyonnais. Je tiens à adresser un remerciement chaleureux à Michel Rougié, sans qui rien n'aurait été possible ; il a suffisamment cru en moi et en ce projet pour faire le nécessaire... Je dis alors merci au chef, au binôme, mais également à l'ami. Un grand merci à Jean-Marc Cros et Maurice Azaïs qui ont initialement soutenu ce projet, et ce avec ferveur. Merci à ceux qui leurs ont succédé : Claude De Bono et Philippe Chouaba. Merci à l'ensemble du Développement Commercial et en particulier à ceux qui m'ont offert un soutien au delà de la relation de collègue à collègue. Elles et ils se reconnaîtront.

Ainsi, cette thèse a pu être menée dans un contexte à la fois scientifique et industriel. Ce fut une suite logique de mon cursus. C'est en suivant la formation professionnalisante du département Informatique et Statistique que j'ai croisé le chemin du Laboratoire ERIC, j'en remercie vivement Stéphane Lallich. Je tiens également à remercier Omar Boussaid et Djamel Zighed qui m'ont mise sur le chemin de la recherche en m'offrant la possibilité d'effectuer un double diplôme me permettant de suivre une formation à la fois professionnalisante et scientifique. Merci alors à Fadila Bentayeb de m'avoir donné la main pour faire mes premiers pas sur ce chemin et m'avoir donné goût à ce travail en jouant avec des index bitmap et des arbres de décision.

Merci à Rahee qui a été près de moi ces dernières années même si aujourd'hui nos chemins se sont éloignés.

Merci à mes parents. Je leur suis très reconnaissante de m'avoir assuré de leurs encouragements et de leur soutien sans borne au cours de ce si long cursus universitaire. J'ai également une pensée affectueuse pour ma sœur, mon frère et ma future belle-sœur.

Merci à Véronique pour son aide un peu particulière ...

Merci à l'ensemble de ma famille pour tout ce qu'elle m'a apporté. Une pensée affectueuse pour mon grand-père paternel dont je suivrai peut-être les traces professionnelles, non pas sur les bancs, mais sur les chaires universitaires. Un grand merci plein d'affection à ma mamie qui m'a accompagnée dans les premiers pas de ma nouvelle vie d'étudiante en m'accueillant sous son toit et qui est toujours là pour moi.

Je remercie tous mes amies et amis pour leurs encouragements même si la fréquence des moments passés ensemble a énormément diminué durant cette période de thèse. Qu'ils se reconnaissent afin que je n'en oublie point.

Et pour n'oublier personne, j'utiliserai la formule d'un ami qui m'est très cher :

« Merci à . . . »

Résumé

Cette thèse a été réalisée dans le cadre d'une Convention Industrielle de Formation par la REcherche (CIFRE) en collaboration avec l'établissement bancaire LCL-Le Crédit Lyonnais. Elle s'inscrit dans le domaine des entrepôts de données. Ces derniers constituent un élément incontournable de l'architecture décisionnelle, sur lesquels reposent alors des outils permettant l'analyse en ligne des données (OLAP : On Line Analytical Processing) pour l'aide à la décision. Le schéma de l'entrepôt, qui détermine les possibilités d'analyse, est conçu en fonction des sources de données disponibles d'une part et des besoins d'analyse d'autre part.

Or, 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. En outre, de nouveaux besoins individuels peuvent émerger. L'émergence de nouveaux besoins d'analyse individuels fait alors apparaître la nécessité d'une personnalisation des analyses, qui placerait l'utilisateur au cœur du processus décisionnel.

Dans cette thèse, nous proposons une solution à la personnalisation des analyses dans les entrepôts de données. Cette solution se base sur une évolution du schéma de l'entrepôt guidée par les utilisateurs. Il s'agit en effet de recueillir les connaissances de l'utilisateur et de les intégrer dans l'entrepôt de données afin de créer de nouveaux axes d'analyse. Afin de développer cette solution, nous avons proposé quatre contributions majeures :

1) Notre première contribution consiste en la définition d'un modèle formel d'entrepôt de données évolutif, basé sur des règles «si-alors», que nous appelons règles d'agrégation. 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 de dimension qui sont mises à jour. Pour assurer la généricité de notre approche, nous proposons également un méta-modèle qui permet de décrire tout entrepôt de données évolutif.
2) Notre modèle d'entrepôt évolutif est soutenu par une architecture qui permet de modéliser le processus de personnalisation. Cette architecture comprend quatre modules :
   – un module d'acquisition des connaissances utilisateurs sous forme de règles d'agrégation ;

- un module d'intégration des règles d'agrégation dans l'entrepôt de données ;
- un module d'évolution du schéma ;
- un module d'analyse permettant à l'utilisateur de réaliser des analyses sur le nouveau schéma.

3) Pour mettre en œuvre cette architecture globale, nous proposons un modèle d'exécution avec l'approche relationnelle, qui vise à gérer l'ensemble des processus liés à l'architecture globale. Il est fondé sur la transformation des règles d'agrégation en une table relationnelle de mapping qui permet le stockage, la vérification des règles, la création des niveaux de hiérarchie.

4) Nous nous sommes par ailleurs intéressés à l'évaluation de la performance de notre modèle d'entrepôt de données évolutif. Or, l'évaluation de la performance des modèles est généralement basée sur une charge (ensemble de requêtes utilisateurs). Lorsqu'un changement au niveau du schéma de l'entrepôt de données se produit, la charge doit être mise à jour. Dans ce contexte, nous proposons ici une méthode de mise à jour incrémentale de la charge.

Pour valider nos différentes contributions, nous avons développé la plateforme WEDriK (data Warehouse Evolution Driven by Knowledge), qui permet la personnalisation des analyses. Elle se base sur un entrepôt de données évolutif stocké dans le SGBD relationnel Oracle d'une part et sur une interface Web programmée en PHP d'autre part.

Les problèmes posés dans ce mémoire sont directement issus de la réalité de l'entreprise LCL avec laquelle nous avons collaboré. LCL a constitué un véritable terrain d'application pour mettre en œuvre nos solutions de personnalisation. Nous nous sommes également intéressés à la personnalisation dans sa définition plus classique, dans le cadre de la gestion des interfaces et de la recherche d'information, au travers du travail d'ingénierie que nous avons réalisé pour cette entreprise durant le développement d'une plateforme pour la gestion des demandes de marketing local : la plateforme MARKLOC.

Mots-clés : entrepôt de données, évolution de schéma, hiérarchie de dimension, mise à jour, personnalisation, utilisateur, règles d'agrégation, analyse en ligne, performance, évolution de charge.

Table des matières

Table des figures

Liste des tableaux

Table des algorithmes

Introduction générale

Sommaire

Chapitre 1

Introduction générale

1.1 Contexte

Nos travaux de thèse ont été réalisés dans le cadre d'une Convention Industrielle de Formation par la REcherche (CIFRE). L'objectif d'une convention CIFRE est d'associer, autour d'un projet de recherche, trois partenaires : une entreprise, un laboratoire et un(e) doctorant(e). L'Association Nationale de la Recherche Technique (ANRT) a accepté les partenaires suivants pour la convention portant l'identifiant 735/2003 : l'établissement bancaire LCL-Le Crédit Lyonnais (noté LCL par la suite), le laboratoire ERIC (Équipe de Recherche en Ingénierie des Connaissances) et moi-même dans le rôle de doctorante. Le projet de recherche a été défini en collaboration avec les différents partenaires et avait pour but de pouvoir apporter des solutions, issues d'un travail de recherche, à des besoins réels. Ces besoins, pour LCL, se situe dans le contexte du marketing local.

LCL, qui constitue la première banque nationale de proximité en France, dispose d'une double politique marketing. Sur un plan national, des actions multi-canaux (courrier, téléphone, courriel) sur des produits identifiés sont menées auprès des clients. En complément de ces actions nationales, des actions au niveau local peuvent être menées pour répondre à des besoins spécifiques émergeant au niveau local (rattrapage de retard sur des objectifs commerciaux, création d'une nouvelle agence, etc.). Ces actions sont proposées par les responsables commerciaux eux-mêmes, à différents niveaux hiérarchiques.

En effet, l'organisation commerciale de LCL correspond à une structure hiérarchique pyramidale. Au début du projet, cette hiérarchie se présentait sous la forme de la pyramide de la Figure 1.1 qui inclut également à sa base un niveau client. Ce niveau ne fait pas partie à proprement parler de la structure organisationnelle

hiérarchique, mais il permet de situer les clients, qui ont bien évidemment un rôle prépondérant pour l'établissement.

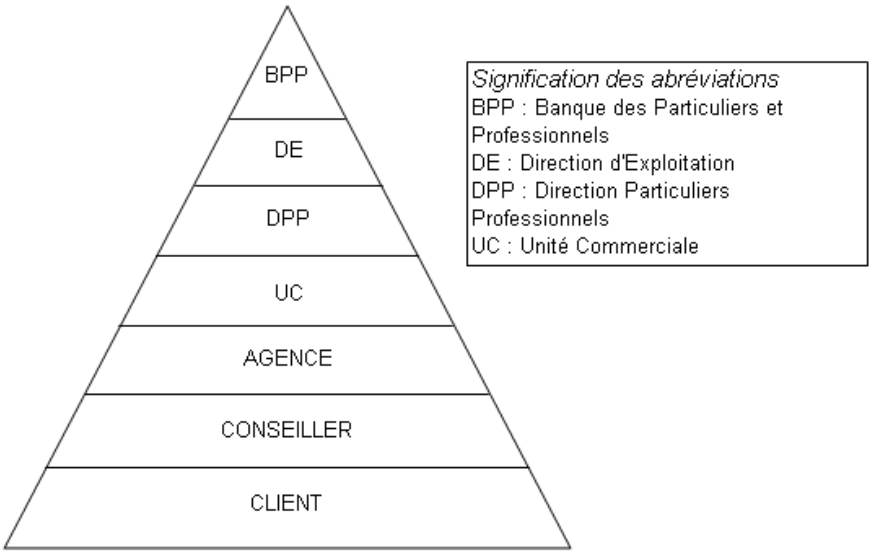


FIG. 1.1 – Structure hiérarchique de LCL au début du projet

Au sommet de cette pyramide, on trouvait la banque des particuliers et des professionnels (`BPP`), elle était composée d'un ensemble de huit directions d'exploitation (`DE`) qui correspondaient à un découpage géographique (Figure 1.2) : le Nord Ouest (`DENO`), Paris (`DEP`), le Bassin Parisien Sud (`DEBPS`), l'Ouest (`DEO`), l'Est (`DEE`), le Sud-Ouest (`DESO`), la Méditerranée (`DEM`) et Rhône-Alpes Auvergne (`DERAA`). Dans le cadre de cette thèse, nous avons travaillé au niveau d'une `DE`, en l'occurrence, celle de la région Rhône-Alpes Auvergne, au service à la fois de cette direction, mais également de toutes les unités des niveaux hiérarchiques qui dépendent de son périmètre géographique. En effet, chaque `DE` est ensuite organisée selon des niveaux hiérarchiques successifs, découpés selon des périmètres géographiques dirigés par des responsables commerciaux. Ainsi, chaque DE était composée d'un ensemble de directions particuliers-professionnels (`DPP`), chaque `DPP` était composée d'un ensemble d'unités commerciales (`UC`), chaque `UC` comprenait un ensemble d'agences. Chaque agence dispose d'un ensemble de conseillers qui sont en contact direct avec les clients. La `DERAA`, avec laquelle nous avons collaboré, comprenait une dizaine de `DPP` (citons par exemple la `DPP DRÔME ARDÈCHE HAUTES-ALPES`), une soixantaine d'`UC` (une des `UC` de la `DPP` citée en exemple est l'`UC ANNONAY`) et environ 300 agences bancaires telles que nous les connaissons.

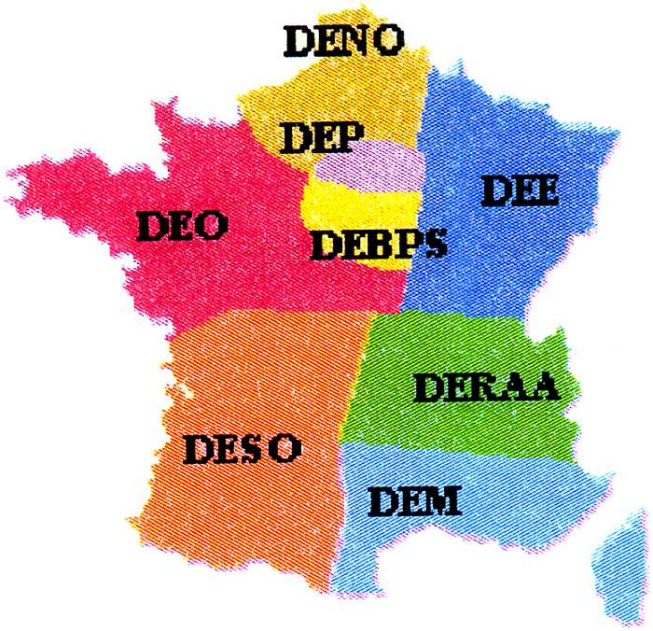


FIG. 1.2 – Périmètre géographique des directions d'exploitation

En observant cette pyramide, nous pouvons constater le rôle primordial des conseillers commerciaux qui sont au contact des clients et, par conséquent, réalisent les ventes. Il s'agit alors, pour l'ensemble de la hiérarchie, de tout mettre en œuvre pour aider et optimiser leur travail. Ainsi, pour compléter la politique marketing nationale et réaliser ainsi des ventes additionnelles, les responsables commerciaux en région Rhône-Alpes sont amenés à faire des demandes de marketing local afin de mener des actions.

Une demande de marketing local est la formulation d'une demande de ciblage de clients pour une action marketing ponctuelle, qui n'est autre qu'une opération spécifique à un produit ou à un événement particulier. Elle se traduit par l'extraction d'une liste de clients répondant à certains critères, autrement dit elle cible certains profils de clients. Cette liste permet aux conseillers commerciaux de contacter leurs clients pour un motif précis, en l'occurrence celui lié à l'action marketing en question. Par exemple, un responsable peut faire une demande de marketing afin de mener une action pour vendre des plans d'épargne logement dans l'agence Annonay suite à un retard sur les objectifs à atteindre pour ce produit.

Auparavant, la gestion des demandes de marketing local se faisait de façon manuelle, via des fiches de liaison papier, transmises par fax pour la validation de niveau hiérarchique en niveau hiérarchique. Une fois les accords successifs des différents ni-

veaux hiérarchiques d'un point de vue commercial, un accord de faisabilité devait être donné par le Pôle Outils et Méthodes de la direction d'exploitation, qui réalisait le ciblage si c'était possible.

Ce fonctionnement posait un certain nombre de problèmes, au-delà du fait qu'il n'était pas pratique. Tout d'abord, il y avait parfois un problème au niveau des informations données pour la demande. En effet, le modèle de fiche de liaison «officielle» avait évolué, mais il subsistait différentes versions utilisées dans le réseau commercial, versions non conformes aux attentes pour la réalisation du ciblage. De plus, comme ces demandes étaient formulées sur un support papier, la recherche d'informations était trop coûteuse en temps et en ressources humaines donc inefficace. En outre, il n'y avait pas de suivi des résultats commerciaux liés à cette demande. Ainsi, si la demande n'était pas relayée au niveau des conseillers, on ne pouvait que constater après la fin de l'action que celle-ci n'avait pas donné de bons résultats commerciaux. Enfin, ne disposant pas d'un système informatisé, il était très difficile, voire impossible, d'analyser les demandes elles-mêmes et donc de capitaliser les connaissances acquises lors des précédentes demandes.

Cette collaboration industrielle et les problèmes concrets que nous avons été amenés à résoudre ont fait émerger deux types d'objectifs : des objectifs liés à un travail d'ingénierie et d'autres relevant d'un travail de recherche.

En effet, il était nécessaire d'apporter une solution pour la gestion des demandes marketing en disposant d'un système informatisé qui permette de gérer l'ensemble du processus lié à celles-ci : de l'acquisition de la demande, à l'analyse des résultats, en passant par la validation hiérarchique. Préalablement, nous avons réalisé une étude conséquente de l'existant et des besoins utilisateurs en exploitant différents moyens (observation, interview et enquête), afin de déterminer le cahier des charges à satisfaire. En faisant émerger les besoins réels et en fournissant une réponse à ces besoins, nous nous assurions de l'adhésion future des utilisateurs au système qui serait développé. L'objectif en terme d'ingénierie a été atteint dans la mesure où LCL dispose aujourd'hui d'une plateforme dédiée, que nous avons développée, nommée MARKLOC. Cette plateforme est supportée par une base des demandes marketing que nous avons conçue et créée, ainsi qu'un processus de flux de travail que nous avons implémenté pour assurer le suivi de la chaîne de validation et de réalisation. Cette plateforme fait partie intégrante de l'intranet de la DE Rhône-Alpes Auvergne et est utilisée quotidiennement par les responsables commerciaux des différents niveaux hiérarchiques pour différentes raisons : émission de demandes, validation de demandes, suivi de résultats commerciaux liés à une demande, etc. Nous reviendrons plus en détails sur les différents aspects techniques de cette plateforme dans

le chapitre 8.

Par la suite, notre objectif était de proposer un système décisionnel qui aide LCL à analyser efficacement ces demandes de marketing local, pour permettre une capitalisation des connaissances. Pour atteindre cet objectif, différentes sources de données hétérogènes et autonomes potentiellement intéressantes étaient à notre disposition, dont la base des demandes marketing que nous avons conçue. Il était donc nécessaire d'avoir recours à un processus d'intégration de données dédié à l'analyse. Nous avons donc conçu un entrepôt de données en fonction des sources dont nous disposions et des besoins d'analyse que nous avions identifiés.

Une des caractéristiques phares de la modélisation des entrepôts est l'historisation des données. Les schémas multidimensionnels classiques permettent de garder effectivement trace de l'historique des faits, de leur évolution, grâce à une dimension Temps. Paradoxalement, ils s'apprêtent mal à prendre en compte les changements pouvant être opérés sur les dimensions, aussi bien du point de vue de leurs données, que de leur structure. Ceci est dû à l'hypothèse d'orthogonalité des dimensions qui est généralement faite [EK01]. Ainsi faire l'hypothèse que chacune des dimensions est orthogonale à la dimension Temps signifie faire l'hypothèse que chacune des dimensions est temporellement invariante. Néanmoins, cette hypothèse n'est pas vérifiée dans la pratique puisque l'évolution de la structure et des données des dimensions est bien réelle. En effet, des produits apparaissent ou disparaissent, les structures organisationnelles se modifient, de nouveaux clients arrivent, certains partent et pour d'autres clients, leurs caractéristiques peuvent être modifiées (statut familial, pouvoir d'achat, etc.).

Cette évolution des dimensions est due à l'évolution des sources de données d'une part et à celle des besoins d'analyse d'autre part, comme c'est le cas chez LCL. En premier lieu, les sources de LCL évoluent. En effet, LCL est un établissement au sein duquel se produisent de nombreux changements, parmi lesquels des changements organisationnels. Par exemple, en juin 2003 s'opérait le rachat par le groupe Crédit Agricole SA. S'en suivait quelques années plus tard une réorganisation complète de la structure commerciale que l'on retrouvait dans les sources de données de LCL. En second lieu, les besoins d'analyse peuvent eux-même évoluer. De nouveaux besoins peuvent émerger, en réaction à l'évolution des données par exemple, ou tout simplement parce que les utilisateurs ont des besoins qui n'avaient pas été recensés lors de la conception de l'entrepôt de données et parce qu'il est difficile de prédire les besoins à venir. LCL est un établissement regroupant des employés exerçant divers métiers et ayant donc des besoins d'analyses variés. Ainsi, de nouveaux besoins individuels peuvent émerger.

Les sources de données, tout comme les besoins d'analyse ne sont pas figés dans le temps. Étant donné que le schéma de l'entrepôt est conçu en fonction des sources de données disponibles et des besoins d'analyse, qu'advient-il de ce schéma lors de leur évolution ? C'est dans ce contexte que nous nous sommes intéressés à la problématique de l'évolution de schémas dans les entrepôts de données.

1.2 Problématique

Les entrepôts de donnée sont nés au sein des entreprises pour répondre à un besoin crucial d'analyse pour l'aide à la décision. Un entrepôt de données peut être vu comme une grosse base de données modélisée pour accueillir, après nettoyage et homogénéisation, les informations en provenance des différents systèmes de production de l'entreprise. L'un des points-clés de la réussite de l'entreposage réside alors dans la conception du schéma de l'entrepôt qui doit permettre de répondre aux besoins d'analyse pour l'aide à la décision.

Il est vrai que la création de l'entrepôt n'est pas un objectif final. L'important, c'est l'exploitation qui s'en suivra. La vocation de l'entrepôt de données est de supporter l'analyse des données. Ainsi, l'entrepôt de données constitue un support pour les outils d'analyse en ligne issus de la technologie OLAP («On-Line Analytical Processing»). Ce sont ces outils qui permettent d'accompagner les décideurs d'une entreprise en leur fournissant une possibilité de naviguer facilement dans les données, et de construire ainsi des informations opératoires.

Ainsi l'entrepôt de données permet d'offrir des contextes d'analyses aux utilisateurs pour les aider dans leur prise de décision. Il s'avère alors que les utilisateurs peuvent avoir besoin de contextes d'analyses spécifiques, répondant à des besoins particuliers voire individuels. L'émergence de nouveaux besoins d'analyse individuels fait alors apparaître la nécessité d'une personnalisation des analyses, qui placerait l'utilisateur au cœur du processus décisionnel. En effet, le processus d'entreposage de données est souvent associé à une idée de technologie centrée utilisateur. Cependant, cette place centrale se limite au fait que l'utilisateur mène son analyse en naviguant dans les données et ce, selon le schéma de l'entrepôt prédéfini. Il est donc intéressant de redonner tout son sens à l'expression «centrée utilisateur». C'est l'un des objectifs de cette thèse.

La personnalisation n'est pas un principe nouveau, elle fait l'objet de nombreux travaux dans des domaines tels que la recherche d'informations, les bases de données. Cependant, elle constitue un axe de recherche récent dans le domaine des entrepôts

de données, alors même que les caractéristiques de ces derniers lui sont favorables. En effet, la volumétrie des données connue pour être importante dans les entrepôts de données et le rôle central que joue l'utilisateur dans le processus décisionnel, au niveau de l'analyse en ligne, sont deux éléments qui justifient pleinement le recours à la personnalisation. Il est vrai que le principe général de la personnalisation, dans un contexte de recherche d'informations par exemple, est de fournir, parmi une multitude de résultats possibles, la réponse (qui peut être un ensemble de résultats) qui sera la plus pertinente pour l'utilisateur. Cette personnalisation doit se baser sur des éléments concernant l'utilisateur lui-même : ses préférences, ses habitudes, etc.

La conception de magasins de données vise à répondre aux objectifs métiers mais elle ne constitue pas une réponse réelle au besoin de personnalisation. Or, les décideurs qui exploitent les entrepôts de données ont d'importantes connaissances métiers et d'autres connaissances qu'il nous semble intéressant d'exploiter.

Ainsi, la question que nous nous posons alors est comment intégrer la connaissance utilisateur dans l'entrepôt de données pour la personnalisation des analyses.

1.3 Contributions

L'objectif général de cette thèse est de proposer une approche qui met à la disposition des analystes des moyens pour intégrer leurs connaissances dans l'entrepôt de données, à des fins d'analyses spécifiques futures.

Compte tenu de l'émergence de nouveaux besoins d'analyse, la conception et la construction d'un entrepôt de données dont le schéma est fixe ne satisfait plus les attentes des décideurs. En effet, la construction des axes d'analyse doit évoluer en même temps que les besoins. Précisons que par axes d'analyse, nous entendons, dans l'ensemble de ce mémoire, les possibilités d'analyses. Ainsi, notre première contribution dans le cadre de la personnalisation des analyses consiste en la définition d'un modèle formel d'entrepôt de données évolutif basé sur des règles de type «si-alors», que nous appelons règles d'agrégation. Nous l'avons nommé modèle *R-DW* pour «Rule-based Data Warehouse». Il 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 de l'ensemble des hiérarchies, définies lors de la conception du schéma de l'entrepôt, qui peuvent subir des mises à jour. En effet, des niveaux de hiérarchie peuvent être supprimés, d'autres peuvent être créés par l'application des règles d'agrégation. Ainsi, le modèle d'entrepôt de données que nous proposons a un schéma flexible

qui permet de prendre en compte l'évolution des besoins d'analyse. Pour assurer la généricité de notre proposition, nous définissons un méta-modèle qui permet de décrire tout entrepôt de données évolutif.

Par ailleurs, afin d'impliquer l'utilisateur dans le processus d'évolution de schéma de l'entrepôt de données, nous proposons une démarche basée sur une architecture globale qui nous permet de modéliser le processus de personnalisation.

Notons que le terme «utilisateur» désigne à la fois l'utilisateur final mais aussi l'administrateur. Par exemple, lors de la modification de la structure interne de l'entreprise, l'administrateur peut faire évoluer le schéma de l'entrepôt en exprimant des connaissances pour l'ensemble des utilisateurs finaux.

Ainsi, notre deuxième contribution est la proposition d'une architecture globale qui vise à soutenir notre modèle d'entrepôt évolutif. Cette architecture globale comprend quatre modules :

- un module d'acquisition des connaissances utilisateurs sous forme de règles d'agrégation ;
- un module d'intégration des règles d'agrégation dans l'entrepôt de données ;
- un module d'évolution de schéma de l'entrepôt qui permet la mise à jour des hiérarchies de dimension
- un module d'analyse permettant à l'utilisateur d'avoir de nouvelles analyses OLAP basées sur le nouveau schéma.

Notre troisième contribution consiste à mettre en œuvre notre démarche. En effet, nous proposons un modèle d'exécution avec l'approche relationnelle (ROLAP : Relational OLAP), qui gère l'ensemble des processus liés à l'architecture globale. Ce modèle d'exécution est constitué des étapes suivantes. Premièrement, les règles d'agrégation sont représentées dans une table relationnelle, que nous appelons table de mapping. Deuxièmement, le contenu de cette table est contrôlé pour vérifier la cohérence des règles d'agrégation. Troisièmement, si les règles d'agrégation sont cohérentes, le nouveau niveau de granularité est créé ; il prend la forme d'une table relationnelle liée à une ou plusieurs autres tables selon que le niveau a été ajouté en fin de hiérarchie ou inséré entre deux niveaux existants de la hiérarchie.

Par ailleurs, nous nous sommes intéressés à l'évaluation de la performance de notre modèle d'entrepôt de données évolutif. Or, l'évaluation de la performance dans les entrepôts de données est généralement basée sur une charge (ensemble de requêtes utilisateurs). Lorsqu'un changement se produit au niveau du schéma de l'entrepôt de données (suppression d'un niveau de hiérarchie par exemple), la charge doit être mise à jour. Ainsi notre quatrième contribution réside dans la proposition

d'une méthode de mise à jour incrémentale de la charge.

Pour valider nos différentes contributions, nous avons développé une plateforme appelée WEDriK[1] (data Warehouse Evolution Driven by Knowledge). Cette plateforme se base d'une part sur un entrepôt de données évolutif stocké dans le SGBD relationnel Oracle et d'autre part sur une interface Web qui permet l'interaction avec les utilisateurs.

Les problèmes posés dans ce mémoire sont directement issus de la réalité de l'entreprise LCL avec laquelle nous avons collaboré. LCL a constitué un véritable terrain d'application pour mettre en œuvre nos solutions de personnalisation. De plus, nous nous sommes également intéressés à la personnalisation dans sa définition plus classique, dans le cadre de la gestion des interfaces et de la recherche d'information, au travers du travail d'ingénierie que nous avons réalisé pour cette entreprise durant le développement de la plateforme dédiée à la gestion des demandes de marketing local MARKLOC, que nous présentons également dans ce mémoire.

1.4 Organisation du mémoire

Ce mémoire est organisé comme suit.

Dans la première partie, nous évoquons le contexte de nos travaux sur deux chapitres.

Le chapitre 2 présente différents aspects relatifs à la modélisation, à la conception des entrepôts de données et à leur mise en œuvre.

Le chapitre 3 est consacré à un état de l'art sur l'évolution de schéma d'une part et sur la personnalisation d'autre part. Nous y détaillons les principaux travaux proposés qui ont traité de la problématique de l'évolution de schéma et plus généralement, de l'évolution de modèle dans les entrepôts de données. Nous proposons également une classification de ces travaux selon divers critères que nous jugeons pertinents. Nous évoquons également le thème de la personnalisation à travers différents domaines que sont la recherche d'informations, l'interaction homme-machine, les bases de données. Nous présentons également les travaux émergents sur la personnalisation dans les entrepôts de données.

Dans la deuxième partie, nous présentons nos contributions scientifiques qui s'articulent selon trois chapitres.

Le chapitre 4 présente notre approche d'évolution de schéma guidée par les utilisateurs. Cette approche vise une personnalisation des analyses en prenant en compte

[1] http ://eric.univ-lyon2.fr/∼cfavre/wedrik/

les connaissances des utilisateurs eux-mêmes pour étendre les hiérarchies de dimension et par conséquent, enrichir les possibilités d'analyse de l'entrepôt. Nous y détaillons la formalisation de notre modèle *R-DW* d'entrepôt de données évolutif, le méta-modèle qui permet d'assurer la généricité de notre approche, ainsi que l'architecture globale qui permet d'impliquer l'utilisateur dans le processus d'évolution de schéma.

Vient ensuite le chapitre 5 qui vise à présenter la mise à jour des hiérarchies de dimension pour la personnalisation des analyses telle que nous la concevons. Nous y détaillons le processus de mise à jour et de sa propagation dans le schéma. Par ailleurs, nous proposons un modèle d'exécution dans le contexte relationnel qui permet de mettre en œuvre notre démarche de personnalisation jusqu'à la mise à jour des hiérarchies de dimension.

Puis, dans le chapitre 6, nous posons le problème de l'évaluation de notre modèle évolutif et nous proposons une méthode de mise à jour incrémentale de la charge en réponse à l'évolution subie par le schéma de l'entrepôt.

Dans la troisième partie, nous détaillons les développements que nous avons réalisés à la fois dans le contexte industriel et dans le contexte scientifique pour valider nos contributions.

D'un point de vue chronologique, c'est le développement et l'exploitation de la plateforme MARKLOC pour le compte de LCL qui ont finalement suscité la problématique de recherche sur la personnalisation des analyses que nous traitons dans ce mémoire. Pour des raisons de cohérence, nous abordons d'abord le développement réalisé pour la validation de nos propositions, avant d'évoquer le développement industriel.

Ainsi, le chapitre 7 présente la validation de nos propositions à travers la réalisation d'un prototype et l'utilisation des données réelles de LCL pour appliquer notre démarche. Préalablement, nous y évoquons donc la construction d'un entrepôt de données test (LCL-DW) avec les sources de données de LCL.

Le chapitre 8 aborde nos contributions en terme d'ingénierie au niveau du projet fait en collaboration avec l'entreprise LCL sur les demandes de marketing local. Ainsi, nous y détaillons les éléments constitutifs de la plateforme MARKLOC et nous nous attachons à décrire comment nous avons mis en œuvre le concept de personnalisation dans son développement.

Enfin, le chapitre 9 conclut ce mémoire. D'une part nous dressons le bilan de cette thèse et de nos contributions. D'autre part, nous présentons nos perspectives de recherche.

Première partie

Contexte de nos travaux de recherche

La première partie de cette thèse vise à situer le contexte de nos travaux de recherche. Ce contexte est présenté selon deux chapitres distincts.

Dans le chapitre 2, nous revenons d'abord sur les concepts généraux qui caractérisent les entrepôts de données, leur objectif, leur modélisation, leur exploitation, leur mise en œuvre, etc.

Une fois ces concepts généraux présentés, nous posons alors le problème de l'évolution de modèle dans les entrepôts de données et présentons un état de l'art sur le sujet dans le chapitre 3. Nous y dressons également un panorama des travaux sur la personnalisation.

Entrepôt de données : conception et exploitation

Résumé

Dans ce chapitre, nous nous attachons à présenter les différents concepts liés aux entrepôts, que ce soit au niveau de la modélisation, de la stratégie de conception, de l'exploitation, etc.

Sommaire

Chapitre 2

Entrepôt de données : conception et exploitation

2.1 Introduction

Cette thèse s'inscrit dans le domaine des entrepôts de données et traite en particulier de l'évolution de schéma. Avant de s'intéresser justement au problème de l'évolution de schéma dans les entrepôts de données, nous nous devons d'évoquer certaines généralités sur ces derniers afin d'éclairer nos propos futurs. Il s'agit en effet de revenir sur ce que sont les entrepôts de données, comment ils sont conçus, quelles sont leurs caractéristiques, etc.

Ainsi, ce chapitre est organisé de la façon suivante. Tout d'abord, nous évoquons dans la section 2.2 la vocation des entrepôts de données, en abordant tout d'abord le rôle de système d'intégration qu'ils ont, puis en précisant le support qu'ils constituent pour l'analyse en ligne et enfin, en évoquant leur place au sein de l'architecture décisionnelle. Puis, dans la section 2.3, nous présentons les stratégies de conception, les principaux éléments relatifs à leur modélisation, avant d'évoquer brièvement les récents travaux sur ce point et de revenir sur la notion de hiérarchie de dimension qui est cruciale pour nos travaux. Dans la section 2.4, nous nous focalisons sur l'exploitation des entrepôts de données, autrement dit l'analyse. Par la suite nous évoquons les stratégies de mise en œuvre (mode de stockage et d'analyse) dans la section 2.5. Enfin, dans la section 2.6, nous concluons ce chapitre en introduisant le problème de l'évolution dans le contexte des entrepôts de données.

2.2 Vocation des entrepôts de données

L'entrepôt de données permet avant tout d'intégrer des sources de données. Mais son objectif final n'est autre que de supporter un processus d'analyse en ligne. Par rapport aux aspects d'intégration et d'analyse, l'entrepôt se trouve finalement être au cœur de l'architecture décisionnelle dont l'objectif est de construire de l'information utile à l'aide à la décision.

2.2.1 Intégration de sources de données

Un entrepôt de données constitue avant tout une alternative pour l'intégration de diverses sources de données. Un système d'intégration a pour objectif d'assurer à un utilisateur un accès à des sources multiples, réparties et hétérogènes, à travers une interface unique. En effet, l'avantage d'un tel système est que l'utilisateur se préoccupe davantage de ce qu'il veut obtenir plutôt que comment l'obtenir, l'objectif étant l'obtention d'informations. Ainsi, cela le dispense de tâches telles que chercher et trouver les sources de données adéquates, interroger chacune des sources de données en utilisant sa propre interface et combiner les différents résultats obtenus pour finalement disposer des informations recherchées. Différentes solutions ont été proposées face au problème de l'hétérogénéité des sources réparties de données. En effet, pour faire des recherches sur l'ensemble de ces sources, une intégration de celles-ci est nécessaire. Deux approches sont alors envisageables : migrer les requêtes vers les sources de données ou migrer les données pour les centraliser dans une source cible. Ceci consiste à suivre respectivement une approche «non matérialisée», appelée aussi approche virtuelle ou une approche «matérialisée», appelée aussi approche d'entreposage [BBDR03].

Dans l'approche «non matérialisée», les données restent au niveau des sources. Différentes solutions suivent l'approche non matérialisée. Parmi elles, on retrouve différents systèmes tels que les portails, les APIs ("Application Programming Interfaces", ou encore la médiation. L'approche de médiation constitue une des approches les plus sophistiquées, mais aussi peut-être les plus puissantes. Elle consiste à définir une interface entre l'agent (humain ou logiciel) qui pose une requête et l'ensemble des sources. Cette interface donne l'impression d'interroger un système centralisé et homogène alors que les sources interrogées sont réparties, autonomes et hétérogènes.

L'approche médiateur présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leurs sources d'origine. Ainsi, il n'y a pas d'intégration des données, autre-

ment dit pas de centralisation de ces dernières. L'utilisateur n'a pas à se soucier du rafraîchissement des sources de données.

Un médiateur comprend un schéma global, qui a nécessité une intégration de schémas. C'est dans les termes du schéma global que l'utilisateur est amené à formuler sa requête qui doit ensuite être traitée pour que les résultats soient obtenus à partir des différentes sources. L'interrogation des sources susceptibles de contenir des informations pertinentes pour la construction de la réponse est effectuée par des adaptateurs.

Dans l'approche d'entreposage, les données sont extraites des différentes sources et combinées pour être stockées dans un schéma global. Il s'agit ici d'une intégration des données. Toutes les données provenant des différents systèmes sont organisées, coordonnées, intégrées pour être finalement stockées de manière centralisée. Cette centralisation physique des données permet à l'utilisateur d'avoir une vue globale des différentes sources en interrogeant l'entrepôt de données.

L'ensemble du processus d'intégration de données dans un entrepôt peut être décomposée en trois étapes : (1) extraction des données à partir de leurs sources ; (2) transformation des données (structurelle et sémantique) ; (3) chargement des données intégrées. On parle de façon usuelle de processus ETL (Extracting, Transforming and Loading) [CD97]. Ce processus nécessite la mise en œuvre d'une stratégie pour le rafraîchissement des données qui doit être réalisé périodiquement. Ce rafraîchissement consiste en l'alimentation de l'entrepôt qui s'enrichit de nouvelles données.

Une des spécificités majeures de l'approche d'entreposage réside dans le schéma global. En effet, ce schéma global dans lequel sont centralisées les données de l'entrepôt présente une modélisation spécifique dédiée à l'analyse. La spécificité de l'entreposage des données par rapport aux approches d'intégration de sources de données réside sans aucun doute dans cet aspect d'analyse. Cette spécificité repose sur une autre distinction entre les deux approches : l'historisation des données. En effet, la modélisation dédiée à l'analyse a pour but de prendre en compte l'évolution des données dans le temps.

Nous revenons plus en détails sur les différents aspects de cette approche d'entreposage puisqu'elle constitue le cadre de travail de nos recherches.

2.2.2 Réponse aux règles de l'OLAP

En 1993, Codd, fondateur des bases de données relationnelles, définit le concept OLAP (On Line Analytical Processing) dans un document technique sous le titre de

«Providing OLAP (On-Line Analytical Processing) to User-Analysts : An IT Mandate» [Cod93]. Il prescrit 12 règles de conception pour ce qu'on appelle le «modèle OLAP».

- Règle 1 - Multidimensionnalité : le modèle OLAP est multidimensionnel par nature
- Règle 2 - Transparence : l'emplacement physique du serveur OLAP est transparent pour l'utilisateur
- Règle 3 - Accessibilité : l'utilisateur OLAP dispose de l'accessibilité à toutes les données nécessaires à ses analyses
- Règle 4 - Stabilité : la performance des interrogations reste stable indépendamment du nombre de dimensions
- Règle 5 - Client-Serveur : le serveur OLAP s'intègre dans une architecture client serveur
- Règle 6 - Dimensionnement : le dimensionnement est générique afin de ne pas fausser les analyses
- Règle 7 - Gestion complète : le serveur OLAP assure la gestion des données clairsemées
- Règle 8 - Multi-Utilisateurs : le serveur OLAP offre un support multi-utilisateurs (gestion des mises à jour, intégrité, sécurité)
- Règle 9 - Inter Dimension : le serveur OLAP permet la réalisation d'opérations inter-dimensions sans restriction
- Règle 10 - Intuitif : le serveur OLAP permet une manipulation intuitive des données
- Règle 11 - Flexibilité : la flexibilité (ou souplesse) de l'édition des rapports est intrinsèque au modèle
- Règle 12 - Analyse sans limites : le nombre de dimensions et de niveaux d'agrégation possibles est suffisant pour autoriser les analyses les plus poussées.

S'en sont suivies, en 1995, 6 règles supplémentaires moins connues, ainsi qu'une restructuration de ces règles en «features», que l'on peut traduire par dispositifs.

Étant donné que ce document a été commandité par une compagnie privée, les règles OLAP telles qu'elles sont définies par Codd ont été controversées. D'ailleurs, Nigel Pendse, l'initiateur de l'Olap Report[1], évoque que ce terme OLAP n'est pas explicite, il ne fournit pas une définition et ne permet pas de savoir si un outil relève ou non de cette technologie. Par ailleurs, 12 règles ou 18 dispositifs, c'est, selon lui, une quantité trop importante pour être facilement retenue. Nigel Pendse suggère alors d'assimiler le terme OLAP à une définition comprenant cinq termes :

[1] http ://www.olapreport.com/

Fast Analysis of Shared Multidimensional Information (le modèle FASMI)[2]. Cette définition correspond finalement aux critères retenus pour simplifier les règles de Codd et faciliter l'évaluation des outils OLAP. Elle a été traduite en français par «Analyse Rapide d'Information Multidimensionnelle Partagée»[3].

Ainsi, on parle généralement de système OLAP, sous-entendant ainsi un système d'informations répondant aux règles évoquées par Codd ou aux critères FASMI. L'aspect analyse y est crucial. C'est d'ailleurs cet aspect qui est mis en avant lorsque l'on oppose le terme OLAP à celui de OLTP («On Line Transaction Processing»). Dans ce cas, ce sont les caractéristiques des applications qui sont comparées. Les applications OLTP ont une architecture permettant de gérer des transactions en temps réel. Elles peuvent être vues comme des applications de production permettant d'effectuer des traitements factuels tels que l'ajout, la suppression et la modification de clients par exemple. Les applications OLAP sont, quant à elles, des applications d'aide à la décision permettant d'effectuer des traitements ensemblistes qui ont pour objectif de résumer l'information en réduisant une population à une valeur, un comportement. Dans ce cas, l'aspect «en ligne» s'applique à l'analyse, c'est le temps de réponse qui doit être quasi instantané. Les deux types d'applications peuvent alors être comparés selon différents aspects. Nous présentons le tableau comparatif inspiré de [Tes00], qui dresse les comparaisons d'un point de vue données et d'un point de vue utilisateurs (tableau 2.1).

Le focus sur l'analyse peut également être envisagé dans le contexte du terme analyse OLAP, i.e. analyse en ligne. Dans ce cas, il s'agit d'un type d'analyse qui se distingue des analyses statistiques ou de la fouille de données par exemple, qui sont présentes en fin de l'architecture décisionnelle. Nous reviendrons par la suite sur ce concept d'analyse en ligne.

2.2.3 Au cœur de l'architecture décisionnelle

L'entrepôt de données est le support nécessaire à la réalisation d'une architecture qualifiée de décisionnelle, qui va permettre, comme l'indique son nom, l'aide à la décision grâce au processus d'analyse qu'elle offre.

Cette architecture décisionnelle, peut être représentée classiquement selon le schéma de la figure 2.1. On peut y distinguer la partie sources, la partie gestion des données et enfin la partie analyse. On parle généralement d'architecture n-tiers en raison des différentes couches possibles pour gérer les données (entrepôt, magasin,

[2] www.olapreport.com/fasmi.htm
[3] http ://www.linux-france.org/prj/jargonf/F/FASMI.html

	OLTP	OLAP
Données	Exhaustives, détaillées	Agrégées, résumées
	Courantes	Historiques
	Mises à jour	Recalculées
	Dynamiques	Statiques
	Orientées applications	Orientées sujets d'analyse
	De l'ordre des gigaoctets	De l'ordre des téraoctets
Utilisateurs	Agents opérationnels	Décideurs
	Nombreux	Peu nombreux
	Concurrents	Non concurrents
	Mises à jour et interrogations	Interrogations
	Requêtes prédéfinies	Requêtes imprévisibles
	Réponses immédiates	Réponses moins rapides
	Accès à peu d'informations	Accès à de nombreuses informations

TAB. 2.1 – OLTP versus OLAP

etc.) et réaliser des analyses.

Cette architecture met en avant deux phases caractéristiques que sont l'intégration des données et l'analyse. Ces phases sont basées sur cinq éléments essentiels qui composent le système décisionnel :

- sources de données : ce sont les bases de production (relevant d'un système OLTP), les fichiers... qui correspondent à des sources internes ; mais elles peuvent également être d'origine externe (Internet, bases de partenaires, etc.) ;
- entrepôt de données : c'est le lieu de stockage massif et centralisé des informations utiles pour les décideurs ; il est associé à un ensemble de méta-données (informations sur les données) qui forme en quelque sorte un référentiel de données contenant différentes informations telles que les règles de transformation et de nettoyage des données permettant d'assurer différentes tâches telles que la maintenance de l'entrepôt.
- magasins de données : ce sont des extraits de l'entrepôt orientés métiers, activités (on parle de données verticalisées), contenant un volume moindre de données, permettant alors des analyses plus rapides ;
- cubes de données : ce sont des contextes d'analyse multidimensionnels ;
- outils d'analyse : ils permettent de manipuler les données et de les analyser.

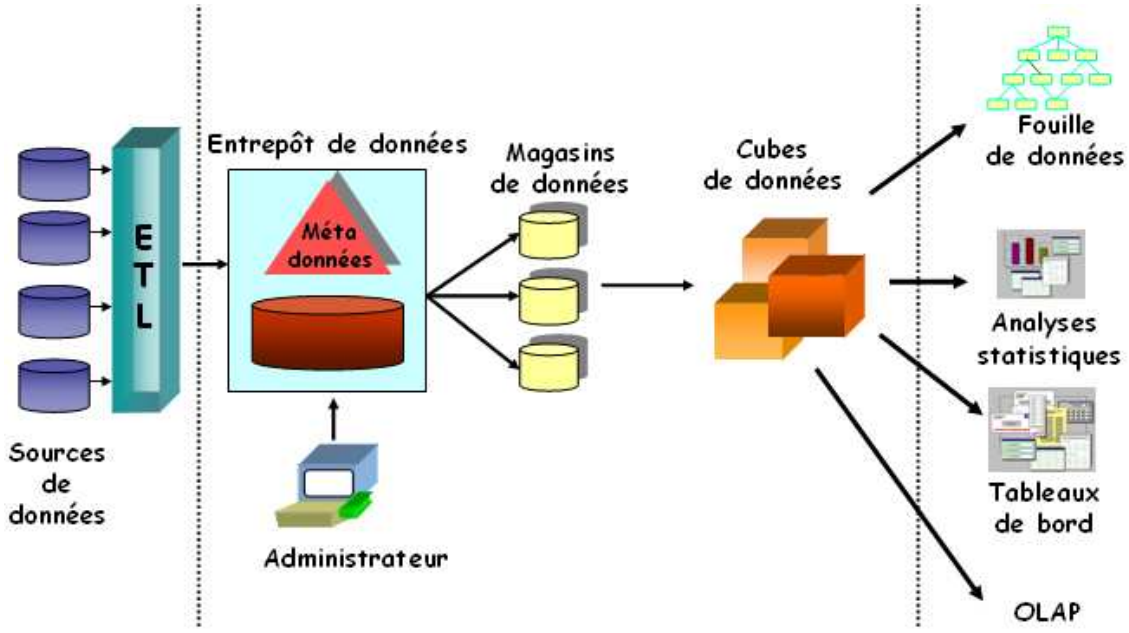


FIG. 2.1 – Architecture décisionnelle

Ainsi, on retrouve la phase d'intégration de données évoqué précédemment avec la présence de différentes sources de données (les bases de production) et le processus ETL qui permet d'intégrer ces données dans l'entrepôt qui centralise les données.

On trouve ensuite la phase d'analyse qui peut exploiter aussi bien des analyses statistiques, des tableaux de bord, de la fouille de données, de l'analyse en ligne, le couplage des deux derniers [MJBN07], etc.

Notons que ces processus d'analyse peuvent s'opérer aussi bien sur l'entrepôt de données, que sur les magasins de données, ou encore sur les cubes de données.

Nous revenons plus en détail sur ces deux phases en abordant tout d'abord la modélisation de l'entrepôt de données qui constitue le résultat de l'intégration, puis en abordant le processus d'analyse de façon plus détaillée.

2.3 Stratégie de conception et modélisation des entrepôts de données

En 1996, Bill Inmon [Inm96] définit un entrepôt de données comme étant une «collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support du processus d'aide à la décision». Les données sont «orientées sujet» dans la mesure où elles sont organisées par thème, l'entrepôt de données

est organisé autour des sujets majeurs et des métiers de l'entreprise. Il permet une vision transversale des différentes activités de l'entreprise. Le fait que les données soient «intégrées» exprime leur provenance de sources différentes. Cette intégration nécessite une bonne connaissance des sources de données, des règles de gestion, de la sémantique des données, etc. En outre, les données sont «historisées» afin de rendre possible la réalisation d'analyses au cours du temps, nécessitant un recours à un référentiel temporel associé aux données. De plus, les données sont dites «non volatiles». Cela signifie que les données stockées au sein de l'entrepôt de données ne peuvent pas être supprimées. Une requête émise sur les mêmes données à différents intervalles de temps doit donner le même résultat. Cela doit permettre de conserver la traçabilité des informations et des décisions prises. Enfin, les données sont «organisées pour le support du processus d'aide à la décision» ; il s'agit en l'occurrence d'une organisation multidimensionnelle. Cette organisation est en effet propice à l'analyse et, en particulier, à l'agrégation, comme nous le montrerons par la suite.

2.3.1 Stratégie de conception

L'un des points clés de l'entreposage réside dans la conception du schéma. En effet, les possibilités d'analyse sont conditionnées par ce dernier. C'est pourquoi différents travaux ont proposé des méthodes pour déterminer le schéma de l'entrepôt [KRRT00, GMR98a, MK00, TPGS01]. La construction du schéma de l'entrepôt n'étant pas une tâche facile, plusieurs travaux ont proposé l'automatisation partielle [SFG05], ou complète de cette tâche [KLLC03, PIR03, PD02].

Par exemple, dans [PIR03], des règles symbolisent la connaissance que l'on a sur la construction d'un entrepôt de données. Les règles définies sont par exemple, la combinaison de deux tables pour en fournir une troisième, la suppression de données ne vérifiant pas une condition, ... Chacune des règles est spécifiée grâce à une description, des structures cibles, un état de départ, des conditions d'application et un état final. C'est un algorithme qui gère l'ordre d'exécution des différentes règles pour la génération du schéma final. Cette génération consiste en une succession de transformations sur le schéma source, chaque règle déterminant quelle transformation doit être appliquée en tenant compte des différentes conditions de conception fournies par le schéma transformé, de la base de données source et des relations entre les deux. Autrement dit, lorsqu'une certaine condition de conception est remplie, la règle applique une certaine transformation.

Un autre exemple d'automatisation de la construction du schéma est proposée dans [KLLC03]. Dans le contexte de la CRM (Customer Relationship Management),

des règles de type «si-alors» permettent de représenter les campagnes marketing à analyser. La clause «si» présente les caractéristiques déterminant la population cible de la campagne marketing (exemple : clients âgés entre 20 et 30 ans et dont les achats pour les deux derniers mois sont supérieurs à 100 euros), et la clause «alors» comprend les caractéristiques de la campagne marketing elle-même (par exemple, support de la campagne : envoi d'un e-mail, indicateurs à suivre : nombre d'e-mails lus). À partir de cette règle, le schéma de l'entrepôt (table des faits et tables de dimension) peut être généré grâce à un algorithme qui extrait les mesures, les dimensions dans les clauses de la règle.

Du point de vue de la conception du schéma de l'entrepôt, nous distinguons dans la littérature trois grandes approches : celle guidée par les données, qualifiée également d'ascendante ; celle guidée par les besoins d'analyse, dénommée également descendante et l'approche mixte qui combine les deux précédentes [SFG05].

L'approche orientée données ignore les besoins d'analyse a priori. Elle concerne en particulier les travaux sur l'automatisation de la conception de schéma. En effet, cette approche consiste à construire le schéma de l'entrepôt à partir de ceux des sources de données et suppose que le schéma qui sera construit pourra répondre à tous les besoins d'analyse. Par exemple, dans [GMR98a], les auteurs proposent une méthodologie semi-automatique pour construire un schéma d'entrepôt de données à partir des schémas entité-relation qui représentent les bases de données sources. Dans [PIR03], les auteurs représentent les connaissances sur la construction de l'entrepôt de données sous forme de règles. Un algorithme gère alors l'ordre d'exécution des règles qui permettent une succession de transformations sur le schéma source pour obtenir le schéma logique final de l'entrepôt.

Les approches orientées besoins d'analyse, quant à elles, proposent de définir le schéma de l'entrepôt en fonction des besoins d'analyse et supposent que les données disponibles permettront la mise en œuvre d'un tel schéma. Parmi les approches orientées besoins d'analyse, certains auteurs proposent une distinction entre les approches guidées par les buts et les approches guidées par les utilisateurs [LBMS02].

L'approche orientée buts suppose que le schéma de l'entrepôt est défini selon les objectifs d'analyse de l'entreprise [MB00]. Ainsi, on suppose que tous les employés de l'entreprise ont des besoins d'analyses similaires vis-à-vis de l'exploitation de l'entrepôt de données. Autrement dit, tous les employés ont la même vision analytique de l'entrepôt de données.

Dans l'approche orientée utilisateurs, ces derniers sont interrogés afin de collecter l'ensemble de leurs besoins d'analyse avant de construire l'entrepôt de données, ce

qui permet de garantir l'acceptation du système par les utilisateurs. Cependant la durée de vie de ce schéma peut être courte, étant donné que le schéma dépend beaucoup des besoins des personnes impliquées dans le processus de développement de l'entrepôt. Pour y remédier, dans [Poe96], l'auteur propose une méthodologie pour conduire les entretiens avec les utilisateurs pour la collecte des besoins. Il est alors recommandé d'interroger différents groupes d'utilisateurs pour avoir la vision la plus complète possible des besoins des utilisateurs. Mais reconnaissons qu'il est non seulement difficile de déterminer de façon exhaustive les besoins d'analyse pour l'ensemble des utilisateurs à un instant donné, mais qu'il est encore moins facile de déterminer leurs besoins à venir.

Dans ces deux approches, nous considérons tout simplement qu'il s'agit de besoins d'analyse qui sont exprimés, certains sont plus globaux, au sens ou l'ensemble des utilisateurs partage ce besoin, d'autres plus spécifiques, sans pour autant que l'aspect personnalisation ne soit abordé d'ailleurs.

Enfin, l'approche mixte considère à la fois les besoins d'analyse et les données pour la construction du schéma. Cette approche est celle qui fait l'objet de plus d'investigations aujourd'hui. L'idée générale est de construire des schémas candidats à partir des données (démarche ascendante) et de les confronter aux schémas définis selon les besoins (démarche descendante) [BCC+01, PD02, SFG05]. Ainsi, le schéma construit constitue une réponse aux besoins réels d'analyse et il est également possible de le mettre en œuvre avec les sources de données.

2.3.2 Concepts de base et modélisation des entrepôts de données

La modélisation des entrepôts de données se base sur deux concepts fondamentaux : le concept de fait et le concept de dimension. Un fait représente un sujet d'analyse, caractérisé par une ou plusieurs mesures, qui ne sont autres que des indicateurs décrivant le sujet d'analyse. Ce fait est analysé selon des axes d'observation qui constituent également ses descripteurs.

Un entrepôt de données présente alors une modélisation dite «multidimensionnelle» puisqu'elle répond à l'objectif d'analyser des faits en fonction de dimensions qui constituent les différents axes d'observation des mesures. Ces dimensions peuvent présenter des hiérarchies qui offrent la possibilité de réaliser des analyses à différents niveaux de granularité (niveaux de détail).

Ces concepts de base ont permis de définir trois schémas classiques reconnus comme relevant d'un niveau logique de conception, en raison du recours à la notion de table (table de faits, table de dimension).

Le premier schéma est le schéma en étoile. Il se compose d'une table de faits centrale et d'un ensemble de tables de dimension. Le deuxième schéma est le schéma en flocon de neige. Il correspond à un schéma en étoile dans lequel les dimensions ont été normalisées, faisant ainsi apparaître des hiérarchies de dimension de façon explicite. La normalisation permet un gain d'espace de stockage en évitant la redondance de données, mais engendre une dégradation des performances, dans la mesure où elle multiplie le nombre de jointures à effectuer pour l'analyse. Le troisième et dernier schéma est le schéma en constellation, aussi appelé flocon de faits. Ce schéma fait coexister plusieurs tables de faits qui partagent ou pas des dimensions communes (hiérarchisées ou non).

Au-delà de ces schémas logiques qui sont à la base des représentations multidimensionnelles, un travail a été réalisé afin de revenir sur une modélisation conceptuelle. Il n'existe à ce jour aucun consensus sur la méthodologie de conception de l'entrepôt, comme cela peut être le cas avec la méthode MERISE pour la conception des bases de données relationnelles [TRC83]. Il n'existe pas non plus de consensus au niveau de la modélisation de l'entrepôt. Différentes pistes ont été proposées ; elles se basent sur l'utilisation de paradigmes variés tels que le paradigme entité-association comme dans [TBC99, FS99] par exemple, le paradigme objet grâce à la modélisation UML (Unified Modeling Language) comme dans [LMTS06, ASS06], ou des paradigmes plus spécifiques dédiés tels que ceux présentés dans [GMR98b]. Une bonne description et comparaison de ces modèles est donnée dans [Ann07].

2.3.3 Importance des hiérarchies de dimension

Si un des objectifs de l'analyse en ligne est bien entendu la rapidité des temps de réponse, la richesse des possibilités d'analyse a également son importance. Cette richesse dépend du schéma de l'entrepôt et, plus particulièrement, des dimensions et de leur(s) hiérarchie(s). En effet, la navigation dans les données est conditionnée par cette organisation dimensionnelle des données.

Cette navigation se base entre autres sur l'agrégation des données. Celle-ci est soutenue par le concept de hiérarchie. En effet, dans les entrepôts de données, les hiérarchies vont permettre de représenter comment doivent être agrégées les données. La hiérarchisation des données dans les modèles multidimensionnels permet des analyses à différents niveaux de détail. Classiquement, les hiérarchies sont représentées par des concepts qui sont reliés par des relations un à plusieurs. Autrement dit, une instance d'un niveau inférieur correspond à une seule instance du niveau supérieur et une instance du niveau supérieur correspond à plusieurs instances du

niveau inférieur. Par exemple, dans le cas de LCL, une `UC` (unité commerciale) appartient à une `DPP` (direction particuliers-professionnels), une `DPP` contient plusieurs `UC`. Ainsi le niveau `UC` constitue le niveau inférieur et la `DPP` le niveau supérieur dans la hiérarchie représentant la structure commerciale de LCL.

D'une façon générale, les hiérarchies correspondent à une réalité des données. Elles peuvent ainsi être définies soit grâce à l'expression des besoins d'analyse des utilisateurs qui connaissent le domaine, soit au niveau des sources de données même puisque ces dernières renferment la réalité de ces données.

Ainsi, lors de la conception de ces hiérarchies, l'approche naïve consiste à les faire émerger en fonction des besoins d'analyse et des sources de données qui sont à disposition. Pour rendre l'approche moins naïve, il a été proposé de définir des hiérarchies à un niveau conceptuel, puis logique, en les déterminant en fonction des relations de généralisation et d'agrégation de la modélisation UML (Unified Modeling Language) des besoins [ACWP01].

Dans [PR99], les auteurs proposent une caractérisation des hiérarchies de dimension. Selon eux, il s'agit de relations exprimées entre les domaines de valeurs. Autrement dit, lorsque l'on va vers un niveau de granularité plus grossier, on effectue un mapping d'un domaine de définition à un autre plus petit. Dans ce cas, on constate deux types de mapping : soit ce mapping est complet, soit il est incomplet.

En effet, différents types de hiérarchies plus complexes sont apparues pour permettre la modélisation de situations réelles. Dans [MZ04], les auteurs proposent une classification des différents types de hiérarchies, en se basant sur des situations réelles. Ils proposent des notations graphiques basées sur les modèles Entités/Relation. Cette représentation conceptuelle permet au concepteur de bien représenter les besoins des utilisateurs. Les auteurs mettent en relief le fait que les outils OLAP classiques prennent en compte un nombre limité de types de hiérarchie, en comparaison du nombre qu'on retrouve dans les situations réelles. Seules les hiérarchies qui ont des propriétés d'additivité sont en général prises en compte. Cette problématique d'additivité a déjà fait l'objet de travaux [HM01].

2.4 Exploitation de l'entrepôt

L'entrepôt a pour objectif final l'analyse des données en vue de la prise de décision. Différents types d'analyse peuvent être réalisés : analyses statistiques, fouille de données, etc. Mais on peut également parler d'analyse en ligne OLAP. Il s'agit en l'occurrence d'une navigation dans les données. Cette analyse peut être qualifiée

d'exploratoire. Le principe général est d'arriver au cours de la navigation à détecter des points intéressants qu'on essaye de décrire, d'expliquer en naviguant, par exemple en allant chercher davantage de détails. Le rôle de l'utilisateur est ici central puisque c'est lui qui réalise la navigation ; celle-ci nécessite une connaissance du domaine afin d'être en mesure de savoir si les valeurs des mesures sont intéressantes ou non. En l'occurrence, elles peuvent être intéressantes lorsqu'elles sont aberrantes. Certains travaux s'intéressent aujourd'hui à la détection automatique de ces points aberrants [SAM98, LB03, PLT07].

Afin de réaliser la navigation, différents opérateurs s'appliquent au niveau d'un cube de données. Ils peuvent être classés en deux catégories : les opérateurs liés à la structure et les opérateurs liés à la granularité. D'une manière générale, les opérateurs liés à la structure permettent la manipulation et la visualisation du cube. Quant aux opérations liés à la granularité, il s'agit d'agréger les données pour obtenir des données résumées et inversement. Nous présentons dans le Tableau 2.2 les opérateurs liés à la structure. Les opérateurs liés à la granularité sont au nombre de deux (Tableau 2.3). Ils se basent sur la hiérarchie de navigation entre les différents niveaux.

Pour enrichir ces opérateurs de navigation, certains auteurs se sont focalisés sur la construction d'opérateurs plus puissants, qui intègrent des méthodes d'extraction de connaissances, telles que la découverte de règles d'association [MRMB07] par exemple.

2.5 Mise en œuvre

Afin de mettre en œuvre l'entreposage des données et l'exploitation de ces dernières, différentes alternatives sont envisageables. Ces alternatives se nomment ROLAP (Relational OLAP), MOLAP (Multidimentional OLAP), HOLAP (Hybrid OLAP), DOLAP (Desktop OLAP). Elles sont liées à l'environnement physique d'implantation.

Afin de classifier une solution OLAP parmi les alternatives évoquées, il existe deux critères : la technologie de stockage des données d'une part et les techniques de traitement de ces données d'autre part.

Concernant la technologie de stockage, les trois possibilités sont les suivantes :

- Base de données relationnelle : les données sont stockées dans un SGBD relationnel. Il permet un stockage presque infini des données (ROLAP).
- Base de données multidimensionnelle (Cube) : les données sont stockées dans

Rotate	Pivot	Consiste à faire effectuer à un cube une rotation autour d'un de ses trois axes passant par le centre de deux faces opposées, de manière à présenter un ensemble de faces différent. C'est en quelque sorte une sélection de faces et non des membres.
Switch	Permutation	Consiste à inter-changer la position des membres d'une dimension
Split	Division	Consiste à présenter chaque tranche du cube et à passer d'une présentation tridimensionnelle d'un cube à sa présentation sous la forme d'un ensemble de tables. Sa généralisation permet de découper un hypercube de dimension 4 en cubes.
Nest	Emboîtement	Permet d'imbriquer des membres à partir du cube. L'intérêt de cette opération est qu'elle permet de grouper sur une même représentation bi-dimensionnelle toutes les informations (mesures et membres) d'un cube, quel que soit le nombre de ses dimensions.
Push	Enfoncement	Consiste à combiner les membres d'une dimension aux mesures du cube, i.e. de faire passer des membres comme contenu de cellules.

TAB. 2.2 – Opérations OLAP liées à la structure

une base de données multidimensionnelle le plus souvent propriétaire. Dans ce cas, il y'a des limitations quant à la quantité des données traitées (MOLAP).
– Fichiers sur le poste client : une petite quantité de données (mini-base multidimensionnelle ou extraction de cube) est stockée directement sur le poste client de l'utilisateur (DOLAP).

Roll-up	Forage vers le haut	Consiste à représenter les données du cube à un niveau de granularité supérieur conformément à la hiérarchie définie sur la dimension. Une fonction d'agrégation (somme, moyenne, etc.) en paramètre de l'opération indique comment sont calculées les valeurs du niveau supérieur à partir de celles du niveau inférieur.
Drill-down	Forage vers le bas	Consiste à représenter les données du cube à un niveau de granularité de niveau inférieur, donc sous une forme plus détaillée.

TAB. 2.3 – Opérations OLAP liées à la granularité

En ce qui concerne les techniques de traitements des données, on distingue trois solutions :

- SQL : SQL est utilisé pour effectuer les différents traitements sur les données. On réalise les opérations de traitement (forage vers le haut, vers le bas, etc.) en utilisant des requêtes en général très complexes et très exigeantes en terme de ressources et de temps d'exécution. Il s'agit de l'alternative relationnelle (ROLAP).
- Serveur de traitement OLAP : il s'agit de l'approche la plus adaptée aux traitements de données. Un serveur, conjointement avec la base de données, est alors dédié à effectuer les différents traitements de données. Dans ce cas, les performances sont généralement très bonnes. Il s'agit de l'alternative multidimensionnelle «pure» (MOLAP).
- Client de traitement OLAP : un nombre limité de traitements OLAP se font sur le poste client de l'utilisateur. Il s'agit de l'alternative «bureau», en local (DOLAP).

En se basant sur ces deux critères, on peut alors différencier aisément les alternatives en combinant une technologie de stockage et une technique de traitement. Les combinaisons sont regroupées dans le Tableau 2.4.

Ainsi, la stratégie du bureau (DOLAP) reste un peu en marge pour un traitement individuel d'une faible quantité de données. Des outils ont été proposés par Cognos [4] ou Business Object[5]. La stratégie relationnelle (ROLAP), proposée par Cognos (Cognos Report Net), Business Object, Microstrategy[6], Hyperion[7]..., quant à elle, est propice à de fortes volumétries de données, néanmoins les performances ne sont pas optimales. Les solutions ROLAP résident dans un environnement relationnel où des tables d'agrégation sont créées dans le même espace que l'entrepôt de données et les magasins de données qui servent de sources pour les cubes ROLAP. La stratégie multidimensionnelle «pure» (MOLAP), proposé par Cognos[8], Hyperion[9], permet d'obtenir de bonnes performances mais se trouve être limitée quant à la volumétrie des données prises en charge. Les données sont pré-agrégées dans un environnement séparé et remplacent les tables d'agrégation relationnelles de la solution ROLAP. Du fait que les données sont bien organisées et indexées, les utilisateurs passent plus de temps à analyser les données dans le cube MOLAP et quand le besoin d'accéder aux

[4] http ://www.cognos.com
[5] www.france.businessobjects.com
[6] http ://www.microstrategy.com
[7] www.hyperion.com
[8] http ://www.cognos.com
[9] www.hyperion.com

Stratégie	Stockage des données	Traitement des données
MOLAP	Base de données dimensionnelle	Serveur de traitement OLAP
ROLAP	Base de données relationnelle	SQL avancé
DOLAP	Fichier sur le poste client	Client de traitement OLAP
HOLAP	MOLAP pour données sommaires & ROLAP pour données détaillées	

TAB. 2.4 – Alternatives de mise en œuvre OLAP

données détaillées se fait sentir, des accès à la base de données relationnelle contenant ces données devient inévitable. Dans ce cas, la stratégie hybride HOLAP permet de combiner les avantages de MOLAP et ROLAP. Il s'agit d'exploiter une technologie multidimensionnelle pour les données agrégées (agrégats) et une approche relationnelle pour le détail des données, comme le propose Cognos. Les éditeurs proposent ainsi les différentes formes de stratégie.

Il y a quelques années, Microstrategy affirme qu'une mise en œuvre relationnelle est flexible et permet de répondre à davantage de besoins alors qu'une mise en œuvre en multidimensionnel «pur» constitue une solution particulière adaptée pour de faibles volumes de données, avec une dimensionnalité pas trop importante [Mic95].

Ainsi, nous venons de préciser différentes alternatives de mise en œuvre. Cette «nomenclature» se base sur le mode de stockage et de traitement des données. Le terme de DOLAP est apparu assez récemment. Classiquement, on opposait le relationnel au multidimensionnel, avec finalement une proposition de solution hybride. Le vocable exploitant l'acronyme OLAP ne cesse de croître aujourd'hui (peut-être parfois parce qu'il est commercialement porteur). Il ne faut pas alors chercher à positionner ces termes les uns par rapport aux autres puisqu'ils référencent des concepts incomparables. Ainsi, on voit aujourd'hui apparaître un terme comme JOLAP[10] (Java OLAP) qui constitue en fait une API (Application Programming Interface) Java qui permet de se connecter à des applications et des serveurs OLAP, tentant de normaliser l'accès aux bases de données multidimensionnelles. On parle de SOLAP (Spatial OLAP) lorsqu'il s'agit de traiter des données spatiales [BTM07]. D'ailleurs, le SOLAP constitue à présent un domaine à part entière de recherche. On parle également de OOLAP (Object OLAP), faisant référence à l'utilisation du paradigme objet. Néanmoins, à notre connaissance, cette technologie n'apparaît pas dans les solutions commerciales. Elle fait cependant l'objet de travaux de recherche [JAC04].

[10] http ://java.sun.com/products/jmi/pres/JOLAPOverview.pdf

2.6 Conclusion

Dans ce chapitre, nous avons évoqué les principaux concepts liés à la conception et à l'exploitation des entrepôts de données. Cette présentation nous a permis de positionner le contexte général des travaux qui seront présentés. La réussite du processus d'entreposage repose entre autres sur une bonne conception du schéma, puisque c'est ce dernier qui va déterminer les possibilités d'analyse de l'entrepôt. Ainsi, de nombreux travaux de recherche ont été menés sur la conception de schéma des entrepôts de données. Ces travaux témoignent aujourd'hui de la nécessité de prendre en compte à la fois les sources de données et les besoins d'analyse [NSF+05], plutôt que d'avoir recours, par exemple, à une approche uniquement guidée par les données telle que celle proposée par [GMR98a].

Mais, une fois l'entrepôt de données construit, ces sources de données et ces besoins d'analyse peuvent subir des changements. Ainsi, lorsque les sources de données ou les besoins évoluent, le schéma de l'entrepôt peut être amené à évoluer, tout comme les données qu'il contient, on parle alors d'évolution du modèle de l'entrepôt. Nous abordons alors dans le chapitre 3 les travaux traitant de cette problématique d'évolution, qui constitue notre contexte de travail.

État de l'art

Résumé

Dans ce chapitre, nous présentons un état de l'art sur l'évolution de modèle dans les entrepôts de données d'une part, et la personnalisation d'autre part. Concernant l'évolution de modèle, après avoir décrit les travaux relatifs à cette problématique, nous dressons une étude comparative. Au sujet de la personnalisation, nous évoquons cette problématique au travers de différents domaines qui s'y intéressent, avant d'aborder ceux relevant du domaine des entrepôts de données.

Sommaire

Chapitre 3

État de l'art

3.1 Introduction

Dans le cadre de notre proposition de personnalisation des analyses dans les entrepôts de données basée sur une évolution du schéma de ce dernier, nous présentons dans ce chapitre un état de l'art qui porte sur deux thématiques : il s'agit de l'évolution de modèle dans les entrepôts de données d'une part et de la personnalisation d'autre part. Ces deux thématiques sont présentées dans ce chapitre de façon indépendante. Elles seront liées par la suite, lors de la présentation de nos contributions.

Ainsi, nous présentons dans la section 3.2 un état de l'art sur les travaux traitant de la problématique de l'évolution de modèle. Cet état de l'art est précédé d'un positionnement de cette problématique. Puis, dans la section 3.3, après avoir dressé un panorama des travaux en matière de personnalisation dans différents domaines que sont la recherche d'informations (RI), les bases de données (BD) et l'interaction homme-machine (IHM) nous évoquons les quelques travaux émergents réalisés en matière de personnalisation dans les entrepôts de données. Enfin, nous concluons ce chapitre dans la section 3.4.

3.2 Évolution de modèle dans les entrepôts de données

3.2.1 Introduction

Les modèles multidimensionnels classiques [CT98, Kim96, Leh98] considèrent les faits comme la partie dynamique des entrepôts de données et les dimensions comme des entités statiques. L'historisation des données est assurée par la dimension *Temps*.

Les autres dimensions sont supposées temporellement invariantes, compte tenu de l'hypothèse selon laquelle les dimensions sont supposées être orthogonales les unes par rapport aux autres et donc orthogonales par rapport à la dimension *Temps*. Cependant, en pratique, des changements peuvent se produire dans le schéma des dimensions et plus généralement sur l'ensemble du schéma de l'entrepôt. En effet, comme nous l'avons souligné précédemment, le schéma peut être amené à évoluer suite à l'évolution des sources de données ou des besoins d'analyse.

La technologie d'entreposage de données s'est inspirée et s'inspire encore aujourd'hui des travaux réalisés dans le domaine des bases de données. Par exemple, les travaux sur les vues [Han87], sur les index [Val87], etc. ont été adaptés pour être appliqués aux spécificités des entrepôts. En ce qui concerne le domaine d'intérêt de ce chapitre, la mise à jour des bases de données [Rod92], les bases de données temporelles [SA86] et les bases de données multiversions [TG89] ont nourri les travaux sur l'évolution des entrepôts de données.

Ainsi, on retrouve aujourd'hui, dans la littérature, différents travaux sur la mise à jour de schéma dans les entrepôts de données, le versionnement de ces derniers pour prendre en compte l'évolution des dimensions, etc. Comme nous l'avons présenté dans [FBB07f], nous proposons ici de classer les différents travaux selon deux familles que nous baptisons respectivement : «modélisation temporelle» et «mise à jour de schéma». Ces deux familles se distinguent respectivement par la conservation ou non de la trace des évolutions subies par le schéma. Chacune de ces familles présente différentes approches que nous nous proposons d'étudier et de comparer.

La suite de cette section est organisée de la façon suivante. Tout d'abord, nous évoquons, sur un exemple issu du cas LCL, les évolutions que peuvent subir un modèle et l'impact qu'elles ont en terme de cohérence des analyses. Ensuite, nous présentons les travaux existants s'intéressant à cette problématique. Puis, nous nous proposons de discuter ces travaux.

3.2.2 Évolution de modèle dans les entrepôts : un exemple illustratif

Dans cette section, nous nous attachons à décrire les évolutions possibles d'un modèle d'entrepôt de données. Nous classons ces évolutions selon deux types : les évolutions sur le schéma d'une part et les évolutions sur les données d'autre part. Pour illustrer ces évolutions et leurs impacts, nous nous proposons de baser notre discours sur un modèle implémenté en relationnel, issu du cas bancaire LCL. Pour illustrer notre propos, nous parlons de table, de clé, etc. En particulier, nous parlerons de table de faits par opposition aux autres tables : les tables de dimension. Ces

dernières représentent donc à la fois les dimensions elles-mêmes, qui sont caractérisées par les tables directement liées à la table de faits et les niveaux de granularité qui composent leurs hiérarchies.

Le schéma multidimensionnel de la figure 3.1 permet d'analyser la mesure `NBI` (Net Banking Income). Le NBI correspond à ce que rapporte un client à l'établissement bancaire. Cette mesure est analysée selon les dimensions `CUSTOMER` (client), `AGENCY` (agence) et `YEAR` (année). La dimension `AGENCY` présente une hiérarchie. Il est ainsi possible d'agréger les données selon le niveau de granularité `COMMERCIAL_UNIT` (unité commerciale) qui est un regroupement d'agences par rapport à leur localisation géographique. Ces unités commerciales sont elles-mêmes regroupées selon un deuxième niveau de granularité : le niveau `DIRECTION`. Ce schéma constitue notre schéma initial. À chaque fois que nous explicitons une évolution, elle est réalisée sur ce schéma de base.

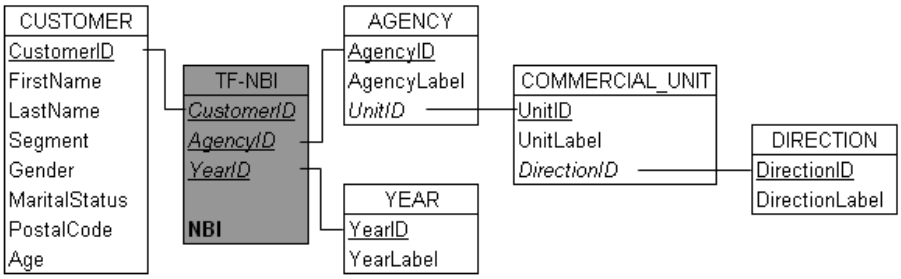


Fig. 3.1 – Schéma multidimensionnel pour observer le NBI

3.2.2.1 Évolution du schéma multidimensionnel

Un schéma multidimensionnel peut subir des évolutions qui peuvent remettre en cause le schéma existant en ayant des impacts d'importance variable sur les données. Par exemple, les évolutions de schéma impactant la table des faits ont en général des conséquences importantes sur les données entreposées. En effet, la volumétrie de l'entrepôt dépend généralement de celle de la table des faits. Ainsi l'impact sur les données de la table des faits peut être considérable. Nous évoquons les évolutions de schéma dans l'ordre décroissant selon l'importance de l'impact sur l'entrepôt.

Tout d'abord, une évolution possible est l'ajout d'une dimension. Cela équivaut à augmenter le niveau de détail de la table des faits (les faits y seront plus détaillés), puisque les mesures présentes dans la table des faits seront décrites par une dimension supplémentaire et présenteront ainsi davantage de descripteurs. Par exemple, il s'agirait d'ajouter une dimension `PRODUCT`, identifiant ainsi un NBI pour une année,

un client, une agence et un produit donnés. L'impact sur les données est considérable puisqu'il s'agit non seulement d'ajouter une table de dimension mais également de recalculer l'ensemble des données de la table des faits lorsque les données sources nous permettent de recalculer les mesures pour les anciens faits.

Une autre évolution possible est la suppression d'une dimension, qui permet de diminuer le niveau de détail de la table des faits (les faits y seront moins détaillées). Par exemple, il est possible de supprimer la dimension agence. Ainsi le NBI serait identifié pour une année et un client donnés. Là encore il faut recalculer les agrégats de la table de faits `TF_NBI` puisque l'identifiant `idAgency` serait supprimé.

Une autre modification qui touche la table des faits est l'ajout d'une mesure. Cette mesure peut être dérivée à partir d'une mesure existante. L'impact est moindre que lorsque l'on touche à une dimension, puisque cela ne remet pas en cause les données existantes de la table des faits. Cependant, cela nécessite de calculer pour chaque fait cette nouvelle mesure. Par exemple, on peut vouloir ajouter dans la table des faits `TF_NBI` une mesure telle que `OVERHEADS` correspondant aux frais de gestion d'un client, par agence et par année. Dans ce cas, cette mesure doit être calculée (à partir des sources de données) pour chacune des lignes de la table des faits. Cette dernière doit partager exactement les mêmes dimensions que les autres mesures de la table des faits. Si ce n'est pas le cas, il faut envisager la création d'une autre table des faits qui pourra partager une partie des dimensions existantes.

La suppression d'une mesure, quant à elle, touche également la table des faits. Néanmoins, aucun recalcul de la table des faits n'est nécessaire, puisqu'il s'agit seulement de «supprimer une colonne». Notons, que cette modification structurelle n'est possible que s'il existait plusieurs mesures pour analyser les faits.

Ensuite, viennent les modifications touchant aux hiérarchies de dimension, telles que l'ajout de nouveaux niveaux de granularité enrichissant une hiérarchie existante ou définissant une nouvelle hiérarchie. Par exemple, il est possible d'ajouter un niveau de granularité `PCS`, représentant les catégories socio-professionnelles, pour créer une hiérarchie sur la dimension `CUSTOMER`. La table `PCS` doit être créée et alimentée, la table de dimension `CUSTOMER` doit être enrichie par un attribut la reliant à `PCS`.

Dans le cas de la suppression de niveau de hiérarchie, l'importance de l'impact dépend de la localisation de ce niveau dans la hiérarchie. En effet, si le niveau est dans une position intermédiaire dans la hiérarchie, il faut assurer la cohérence en maintenant les liens nécessaires dans la hiérarchie. Par exemple, si le niveau `COMMERCIAL_UNIT` est supprimé, il faut assurer le lien entre les niveaux `AGENCY` et

DIRECTION. Si, par contre, c'est le niveau DIRECTION qui est supprimé, il n'y a pas de maintenance particulière à assurer, si ce n'est la suppression elle-même du niveau et celle du lien entre les niveaux COMMERCIAL_UNIT et DIRECTION.

Ces différentes modifications d'ordre structurel enrichissent (ajout de dimension, de mesure, de niveau de hiérarchie) ou appauvrissent (suppression de dimension, de mesure, de niveau de granularité) les analyses. Néanmoins, ces modifications ne remettent pas en cause la véracité, la cohérence des analyses. Le problème de cohérence des analyses se pose lors de l'évolution des données, comme nous allons le montrer dans ce qui suit.

3.2.2.2 Évolution des données

Quelle que soit l'évolution opérée sur le schéma, il est nécessaire de répercuter cette évolution au niveau des données elles-mêmes. Cette évolution nécessite parfois de disposer de données pour alimenter l'entrepôt, en particulier dans le cas d'ajout de mesure, d'ajout de niveau de hiérarchie, etc. Cela modifie entre autres le processus de chargement des données.

Il est plus difficile de définir de façon exhaustive les évolutions possibles des données que celles du schéma. De façon générale, les évolutions de données peuvent être de l'ordre de l'insertion, la suppression et la modification. Dans le contexte des entrepôts de données, ces trois opérations de base ont des conséquences différentes selon le concept sur lequel elles sont appliquées.

Envisageons tout d'abord le cas de la table des faits. L'insertion de données (de faits) correspond à la phase classique d'alimentation. Compte tenu de la non-volatilité et de l'historisation des données [Inm02], les faits ne sont pas amenés à être supprimés. En d'autres termes, il ne s'agit pas de supprimer de la table de faits un enregistrement complet. De la même façon, les données de la table des faits ne devraient pas être modifiées. Néanmoins, [RG06] évoquent la nécessité de mettre à jour les valeurs des mesures, lorsqu'elles ont fait l'objet d'erreurs ou lorsque les événements qu'elles traduisent évoluent et ce, contrairement au principe de non volatilité des données auquel répondent les entrepôts de données.

Concernant les tables de dimension, l'insertion de données (instances de dimension) correspond également à la phase classique d'alimentation. La suppression dans une table de dimension ne peut avoir lieu que si les données précédemment récoltées n'y font plus référence. Ainsi, l'ensemble du problème de cohérence des données, et donc des analyses, peut être ramené la plupart du temps au problème de la modification des données sur les instances de dimension. Nous revenons sur ce point par

la suite.

Kimball a évoqué ce problème en introduisant trois types de «Slowly Changing Dimensions» ou «dimensions changeantes à évolution lente» qui constituent en fait trois possibilités pour gérer les changements dans les structures multidimensionnelles [Kim96]. L'hypothèse de départ est de dire que l'identifiant de la dimension ne change pas, ce sont ses descripteurs qui évoluent. Par exemple, l'identifiant d'un client ne change pas, mais il peut changer d'adresse. La première solution est d'écraser l'enregistrement avec la nouvelle valeur. Cette solution engendre la perte de l'historique. Ainsi, cette solution est intéressante lorsque l'ancienne valeur de l'attribut n'a plus de sens ou qu'elle peut disparaître. La deuxième solution consiste à créer un enregistrement supplémentaire. Chaque enregistrement correspond alors à une description unique valide pendant une période donnée. Il s'agit en effet de conserver toutes les versions des membres de la dimension. Cependant, dans une telle représentation, les comparaisons des données le long des évolutions sont rendues difficiles, puisque les liens entre elles ne sont pas conservés, bien que les évolutions le soient. La troisième solution est de créer un champ conservant l'ancienne valeur de l'attribut dans le même enregistrement. Cependant des limitations existent pour cette solution, si par exemple il y a une succession de changements à prendre en compte, puisque des recouvrements entre les versions peuvent apparaître mais ne peuvent être traités.

Kimball a ainsi défini les bases de solutions permettant de gérer l'évolution des dimensions, en insistant sur le fait qu'il est important de conserver l'historisation des données telle que Inmon l'évoque dans sa définition d'un entrepôt de données [Inm02]. Mais dans quelle mesure cette historisation de données garantit une cohérence des analyses ? C'est ce que nous abordons dans ce qui suit.

3.2.2.3 Cohérence des analyses

Au-delà de pouvoir réaliser des analyses en concevant un entrepôt de données, l'objectif réel est de disposer d'un entrepôt de données qui assure la cohérence des analyses. L'atteinte de cet objectif est conditionnée largement par le fait que l'entrepôt de données soit un miroir de la réalité. De notre point de vue, le problème de l'évolution du modèle dans les entrepôts de données ne doit pas être dissocié du problème de cohérence des analyses. Ainsi, il faut savoir reconnaître les cas où la cohérence des analyses n'est pas mise en danger, même si l'historisation des données n'est pas assurée. Nous parlons de problème de cohérence des analyses lorsque l'évolution subie impacte les analyses en modifiant leurs résultats. Il s'agit d'un problème considérable puisque, par définition, les résultats des analyses sont utilisés

pour prendre des décisions.

Comme nous avons pu le remarquer précédemment, le problème de cohérence des analyses se pose essentiellement dans l'évolution des dimensions et de leurs hiérarchies. En effet, l'hypothèse classique d'indépendance (on parle aussi d'orthogonalité) des dimensions entre elles sous-entend l'indépendance des dimensions avec la dimension temps. Ceci implique que ces dernières sont temporellement invariantes. Or ce n'est pas le cas dans la réalité.

Si un attribut référence un autre niveau de la hiérarchie, nous parlons dans ce cas de «descripteur hiérarchique», la perte de l'historisation des données sur cet attribut induit forcément une incohérence des analyses, dans la mesure où le lien d'agrégation est modifié. Par exemple, dans le schéma de la figure 3.1, la modification de la valeur de l'attribut `UnitID` de la table `AGENCY` entraîne des changements considérables du point de vue de l'analyse [BMBT02]. En effet, on peut considérer que l'on veut obtenir une analyse en prenant en compte un «temps consistant», correspondant au fait que l'on considère les faits selon la période où ils sont valides : avant une certaine date ce NBI a été réalisé par une certaine unité commerciale, puis il est rattaché ensuite à une autre unité commerciale. Il est également possible de considérer que l'agence appartient encore à l'ancienne unité commerciale. Enfin, il peut être intéressant de considérer que l'agence a toujours appartenu à l'unité commerciale dans laquelle elle a été affectée nouvellement. On observe la même problématique de cohérence d'analyse lorsqu'un attribut d'un niveau est un «descripteur direct», c'est-à-dire un descripteur du niveau lui-même, tel que l'attribut `MaritalStatus` qui représente la situation familiale (marié, célibataire, etc.) et qui peut intervenir dans l'analyse pour réaliser un regroupement. Par contre, lorsqu'un descripteur direct n'intervient pas dans l'analyse, tel que l'attribut `FirstName`, l'historisation n'est pas nécessaire et le problème de cohérence des analyses ne se pose pas.

Après avoir présenté les évolutions que peut subir un modèle et avoir montré l'interaction entre l'historisation des données et la cohérence des analyses, nous nous attachons, par la suite, à évoquer les différents travaux qui permettent de gérer ces évolutions.

3.2.3 Évolution de modèle dans les entrepôts : l'existant

3.2.3.1 Mise à jour de modèle dans les entrepôts de données

Les travaux proposant la mise à jour de modèle sont caractérisés par le fait qu'ils ne présentent qu'un modèle. Les évolutions sont donc appliquées pour constituer

un nouveau modèle. Ainsi la traçabilité des différentes évolutions n'est pas assurée. Nous avons classé ces travaux en trois courants.

Un premier courant est la proposition d'opérateurs pour faire évoluer le modèle. Dans [HMV99], les auteurs proposent un modèle formel pour la mise à jour des dimensions et de leur hiérarchie, en définissant des opérateurs qui répondent non seulement à une évolution des instances des dimensions, mais également à une évolution structurelle des dimensions, telle que l'ajout d'un niveau de granularité en fin de hiérarchie. Ils étudient également l'effet de ces mises à jour sur les vues matérialisées et proposent également un algorithme pour réaliser leur maintenance de façon efficace.

Dans [BSH99], les auteurs proposent, non seulement des évolutions au niveau des dimensions, mais également au niveau des faits. L'évolution qu'ils proposent est réalisée à un niveau conceptuel, indépendant de l'implémentation. Ils proposent ainsi une algèbre comprenant quatorze opérateurs d'évolution qui peuvent être combinés pour réaliser des opérations d'évolution complexes. Par exemple, il est proposé d'ajouter un niveau et ce, à n'importe quel endroit dans la hiérarchie de dimension, contrairement à ce qui est possible dans l'approche proposée par [HMV99]. Ce travail est étendu dans [Bla00] en proposant également la propagation de ces changements du niveau conceptuel vers le niveau logique.

Ces travaux ont été exploités dans [BG02] afin de proposer un gestionnaire d'entrepôts qui permet de gérer la création et l'évolution du schéma de l'entrepôt et ce, de façon indépendante du mode de stockage des données (relationnel, etc.).

Un deuxième courant s'est s'inspiré des travaux sur les opérateurs d'évolutions en se focalisant sur la création de nouveaux niveaux dans les hiérarchies de dimension. L'objectif est de s'intéresser à comment créer ces niveaux, non pas à comment représenter cette opération. Ainsi, il s'agit d'une mise à jour du modèle de l'entrepôt qui ne remet pas en cause la cohérence de l'analyse des données existantes puisqu'il s'agit d'un enrichissement du modèle. On peut citer le travail proposé dans [MT06] qui permet d'enrichir des hiérarchies de dimension à la fois au niveau de la structure et des données et ce, de façon automatique. En partant du principe qu'une hiérarchie de dimension représente des relations sémantiques entre des valeurs, ils proposent d'exploiter les relations d'hypéronymie («*is-a-kind-of*») et de méronymie («*is-a-part-of*») de WordNet[1]. Les niveaux de granularité sont créés en fin de hiérarchie.

Le troisième courant se base sur l'hypothèse qu'un entrepôt de données est un

[1] http ://wordnet.princeton.edu/

ensemble de vues matérialisées construites à partir des sources de données [Bel02]. Dans [HMV99], les auteurs se sont intéressés à la maintenance de vues pour propager l'évolution du modèle sur les cubes de données, représentés par des vues. Dans [Bel02], il s'agit de s'intéresser à la maintenance de vues matérialisées induite directement par une évolution des sources de données. Ainsi, il s'agit de ramener le problème de l'évolution des sources de données à celui de la maintenance des vues. La prise en compte d'évolution suite à des besoins est proposée à travers l'ajout d'attributs dans les vues et la modification de domaine de définition des attributs, tous deux réalisés par l'administrateur. Nous renvoyons le lecteur, pour de plus amples détails sur la maintenance de vues matérialisées dans ce contexte, vers l'état de l'art proposé par [BBDG05] sur la maintenance de vues matérialisées issues de sources de données hétérogènes.

Nous avons présenté trois courants s'inscrivant dans une mise à jour du modèle de l'entrepôt. Dans le premier courant, les opérateurs permettent de faire évoluer le modèle ; l'évolution des données est succinctement évoquée dans ces travaux. Le deuxième courant s'est intéressé précisément à comment réaliser une évolution telle que la création de nouveaux niveaux de granularité dans les hiérarchies de dimension. Enfin le troisième courant permet une évolution du modèle induite directement par l'évolution des sources de données, en posant l'hypothèse qu'un entrepôt est un ensemble de vues matérialisées. Ces trois courants répondent à des problématiques différentes. Le premier permet de proposer une évolution du modèle de l'entrepôt pour répondre à un besoin d'évolution traité par l'administrateur. Le deuxième propose une solution pour trouver les données nécessaires afin de réaliser une évolution spécifique dans le but de répondre à des besoins d'évolution liés davantage à l'analyse. Enfin, le troisième courant répond plus particulièrement à un besoin d'évolution en réponse à l'évolution des sources de données.

3.2.3.2 Modélisation temporelle des entrepôts de données

Les travaux proposant une modélisation temporelle de l'entrepôt s'opposent à ceux présentant une mise à jour de modèle sur le plan de l'historisation des changements. En effet, les approches se distinguent sur la traçabilité des évolutions subies par le modèle. Pour assurer cette traçabilité, des extensions temporelles sont nécessaires pour enrichir le modèle. Nous distinguons alors trois courants qui utilisent des étiquettes temporelles à différents niveaux. En effet, ces étiquettes sont apposées soit au niveau des instances, soit au niveau des liens d'agrégation, ou encore au niveau des versions du schéma. Nous détaillons ces différentes approches dans ce qui suit.

Le premier courant propose ainsi de gérer la temporalité des instances de dimensions [BSSJ98]. Inspiré des travaux sur les bases de données temporelles [SA86], un schéma en étoile temporel est proposé pour représenter le fait que les informations dans un entrepôt de données sont valides sur une durée donnée. Il s'agit donc de représenter les données en «temps consistant». Le principe est d'omettre la dimension temps qui permet habituellement l'historisation des données et d'ajouter une étiquette temporelle au niveau de chacune des instances des tables de dimension et des faits de l'entrepôt.

Le deuxième courant propose, quant à lui, de gérer la temporalité des liens d'agrégation [MV00]. Il s'agit de pouvoir gérer des dimensions temporelles pour lesquelles les hiérarchies ne sont pas fixes au niveau des instances. Ainsi le chemin d'agrégation défini pour une instance le long d'une hiérarchie peut évoluer au cours du temps. Pour interroger ce modèle, les auteurs proposent un langage de requêtes nommé TOLAP.

Le dernier courant et non le moindre, est la gestion de la temporalité au niveau de versions du modèle. En effet, la gestion des versions constitue une voie de recherche très explorée et prometteuse. Cela consiste à gérer différentes versions du modèle de l'entrepôt, chaque version étant valide pendant une durée donnée. De nombreux travaux s'inscrivent dans cette alternative. Nous en présentons ici un échantillon représentatif.

Le modèle proposé dans [EK00] présente des fonctions de mise en correspondance qui permettent la conversion entre des versions de structures. Ces fonctions sont basées sur la connaissance des évolutions opérées. Dans [BMBT02, BMBT03], les auteurs proposent une approche qui permet à l'utilisateur d'obtenir des analyses en fonction de différentes situations. En effet, le modèle proposé permet de choisir dans quelle version analyser les données (en temps consistant, dans une version antérieure, ou dans une nouvelle version). Dans [RTZ06], les auteurs proposent un modèle multidimensionnel en temps consistant se caractérisant par le fait qu'il permet des évolutions sur un modèle en constellation. Le versionnement permet également de répondre à des «*what-if analysis*», en créant des versions alternatives, en plus des versions temporelles, pour simuler des changements de la réalité [BEK+04]. Différents travaux se sont ensuite focalisés sur la possibilité de réaliser des analyses en prenant en compte différentes versions [MW04, GLRV06].

Ainsi, la modélisation temporelle constitue aujourd'hui une alternative en pleine expansion qui suscite de nouveaux problèmes qu'il faut résoudre. Dans ces différents courants, les évolutions du modèle sont donc bien conservées et assurent la cohérence des analyses. Ce type de solutions implique une ré-implémentation des outils de

chargement de données, d'analyse, avec la nécessité d'étendre les langages de requêtes afin de gérer les particularités de ces modèles. Il est donc nécessaire, dans ce cas, de prévoir au moment de la conception comment vont être gérées les évolutions à venir.

3.2.4 Discussion

Dans cette section, nous présentons un ensemble de critères que nous avons proposés dans [FBB07b] et que nous avons jugé pertinents pour évaluer les travaux sur l'évolution de modèle dans les entrepôts de données. Nous les comparons ensuite selon les critères sélectionnés.

3.2.4.1 Critères de comparaison

Nous avons déterminé trois groupes de critères que nous avons jugés pertinents compte tenu des objectifs des entrepôts de données. Ils concernent d'une part les caractéristiques des approches, ensuite la mise en place de ces approches et enfin, leur performance.

Les critères sur les caractéristiques des approches sont :
- historisation des dimensions ;
- cohérence des analyses ;
- approche orientée utilisateurs.

Tout d'abord, il s'agit de savoir si l'historisation des dimensions est assurée. En effet, ce critère permet de déterminer si oui ou non les dimensions sont considérées comme temporellement invariantes. Ensuite, l'idée est de mesurer la cohérence des analyses lors de l'application de l'approche. Enfin, il s'agit de déterminer si l'approche se focalise sur le besoin utilisateur qui doit être au centre du processus décisionnel.

Les critères sur la mise en place des approches sont :
- nécessité d'implémenter la solution lors de la conception ;
- complexité de la mise en œuvre (analyse, chargement).

Il est ainsi intéressant d'étudier comment sont mises en œuvre les approches : d'une part si elles doivent être choisies dès le moment de la conception de l'entrepôt, d'autre part si elles sont complexes à mettre en œuvre (par exemple, en mesurant la nécessité d'adapter des outils).

Enfin, les critères sur les performances liées aux approches sont :
- stockage ;
- temps de réponse aux analyses.

Compte tenu de l'objectif lié aux entrepôts qui est l'analyse «en ligne», donc souhaitée rapide, les performances constituent un aspect crucial, non seulement au niveau des analyses, mais également au niveau de la capacité de stockage, étant donné que par définition, la volumétrie des entrepôts de données est d'emblée importante.

3.2.4.2 Comparaison des travaux

La comparaison des travaux selon les deux principales familles (mise à jour de modèle et modélisation temporelle) est récapitulée dans le Tableau 3.1, où un + (resp. -) signifie que l'approche a une influence positive (resp. négative) sur le critère précisé en en-tête de ligne.

		Mise à jour de modèles			Modélisation temporelle		
		Opérateurs d'évolution	Enrichissement hiérarchies	Maintenance de vues	Instances	Liens d'agrégation	Versions
Caractéristiques	historisation des dimensions	-	-	-	+	+	+
	cohérence des analyses	-	+	-	+	+	+
	approche orientée utilisateurs	-	+	-	-	-	-/+
Mise en place	mise en œuvre dès la conception	+	+	+	-	-	-
	complexité	+	+	+	-	-	-
Performances	stockage	+	+	+	-	-	-
	temps de réponse aux analyses	+	+	+	-	-	-

TAB. 3.1 – Comparatif des travaux sur l'évolution de modèle.

Les approches s'inscrivant dans une modélisation temporelle permettent d'assurer l'historisation des dimensions. Concernant les approches de mise à jour de modèle,

cette historisation n'est pas assurée. Néanmoins, des mises à jour telles que l'ajout de niveaux de granularité ne remettent pas en cause la cohérence des analyses, même si l'historisation des modifications subies par le modèle ne sont pas conservées. De ce fait, les travaux sur l'enrichissement de hiérarchies de dimension ne posent pas de problème de cohérence des analyses, tout comme les approches suivant une modélisation temporelle.

Concernant la place des utilisateurs dans le processus de gestion des évolutions, cette dernière est variable selon les approches. Pour répondre à l'évolution des besoins d'analyse, permettant une implication des utilisateurs, il s'avère qu'on peut imaginer qu'elle peut être indirecte. Il s'agit de récolter au fur et à mesure ces besoins et de mettre en œuvre les solutions pour faire évoluer le modèle de l'entrepôt en fonction de ces besoins. Les approches temporelles qui permettent un choix de la version dans laquelle les utilisateurs veulent réaliser leur analyse est positive également de ce point de vue. Cet aspect sur la place des utilisateurs est important, d'autant plus que la personnalisation dans les entrepôts de données devient un enjeu crucial, ce que nous montrons dans ce qui suit.

Concernant la mise en place des approches, les modélisations temporelles nécessitent d'être prévues dès la conception de l'entrepôt et nécessitent la conception d'outils spécifiques pour l'alimentation et l'analyse de l'entrepôt de données. Ces approches peuvent donc être complexes à mettre en œuvre. En effet, la lourdeur de la mise en œuvre d'un entrepôt de données «classique» est reconnue, on imagine donc aisément la difficulté accrue lorsqu'il s'agit d'une modélisation temporelle.

Enfin, concernant la performance, il faut savoir que la modélisation temporelle nécessite de plus grands espaces de stockage, au niveau du stockage d'étiquettes temporelles, de versions, de méta-données, etc. Par ailleurs, les temps de réponse dans les approches temporelles sont également plus longs pour prendre en compte les spécificités du modèle. La réécriture des requêtes est souvent nécessaire pour prendre en compte par exemple les différentes versions.

Pour conclure cette discussion, nous souhaitons mettre en avant que même si la modélisation temporelle fait l'objet de nombreux travaux de recherche (sur le versionnement en particulier), son utilisation n'est pas encore généralisée dans la pratique. Par exemple, SAP-BW permet à l'utilisateur de choisir quelle version des hiérarchies il souhaite utiliser pour l'analyse [ASA00]. Cependant, le versionnement de schéma n'a pas été complètement exploré et aucun outil commercial dédié n'est disponible, à notre connaissance, pour la conception et l'administration.

Par ailleurs, étant donné que ces approches nécessitent d'être prises en compte

dès la conception de l'entrepôt, celles-ci ne pourront être mises en œuvre facilement pour les entreprises qui utilisent d'ores et déjà une architecture décisionnelle basée sur un entrepôt de données «classique». Les entreprises exploitent des entrepôts dont les données sont mises à jour. C'est le cas de l'entreprise avec laquelle nous collaborons. L'ensemble de la structure commerciale de LCL a changé. Il n'y aura plus de trace de l'ancienne structure, les analyses se feront comme si la structure actuelle avait toujours été. Il s'agit finalement d'un arbitrage entre complexité (pour assurer l'exactitude des analyses) et simplicité (en ayant des analyses pouvant être erronées). Bien entendu, le coût (de conception, de maintenance, etc.) est proportionnel à la complexité de l'approche.

3.3 Personnalisation

Dans cette section, nous traitons des travaux consacrés à la personnalisation dans des domaines aussi variés que l'interaction homme-machine (IHM), les bases de données (BD), la recherche d'informations (RI), mais également des travaux plus récents dans le contexte des entrepôts de données.

3.3.1 Personnalisation en IHM, BD et RI

Dès lors que l'on souhaite répondre à des besoins utilisateurs peut se poser la question de la personnalisation vis-à-vis de ces derniers. Ainsi, l'idée même de permettre une personnalisation de l'information n'est pas nouvelle et a été abordée par différentes communautés scientifiques telles que celle de l'interaction homme-machine, des bases de données et de la recherche d'informations.

Ce besoin de personnalisation est en partie dû à la profusion des données parmi lesquelles chaque utilisateur cherche des réponses particulières (que ce soit dans une base de données ou grâce à un moteur de recherche sur Internet). Cette profusion s'explique par différentes raisons parmi lesquelles : l'augmentation des capacités de stockage, la baisse de leur coût, les progrès faits en matière de partage et de distribution des données et l'avènement d'Internet. L'accès à une information pertinente devient alors un enjeu crucial pour l'utilisateur. Il est alors nécessaire d'éviter une surcharge d'informations.

La personnalisation de l'information peut être définie comme étant un ensemble de préférences individuelles, pouvant être représentées de différentes manières, qui vont être utilisées pour fournir des réponses à l'utilisateur les plus pertinentes possibles. Généralement, la personnalisation est basée sur la notion de profil utilisateur

[BK05]. Le contenu de ce profil varie selon les approches.

Dans le domaine de l'IHM, le profil va contenir des informations qui vont permettre au système d'adapter l'affichage des résultats selon les préférences de l'utilisateur. C'est le cas de l'environnement Yahoo ! qui recueille dans le profil un certain nombre d'informations personnelles et adapte la page d'accueil en fonction des centres d'intérêt de l'internaute. Dans le domaine de la RI, le profil utilisateur peut être représenté de différentes manières dont nous évoquons ici quelques exemples. Dans certains cas, le profil utilisateur peut être confondu avec la requête elle-même de l'utilisateur. Dans ce cas, le profil est alors défini par un vecteur de mots-clés, avec éventuellement un poids associé à chaque mot-clé [PG99]. Un profil utilisateur peut également contenir les statistiques d'actions avec le système (nombre de clicks, temps de lecture, etc.) [BRS00]. Ceci permet par la suite d'inférer sur les préférences en connaissant davantage son comportement. Une autre alternative consiste à stocker dans le profil utilisateur des fonctions d'utilités sur un domaine d'intérêt, qui permettent d'exprimer l'importance relative des sujets de ce domaine, les uns par rapport aux autres [CGFZ03]. Dans le domaine des BD, le profil utilisateur peut contenir par exemple les habitudes d'interrogation de celui-ci, en l'occurrence les prédicats souvent utilisés dans ses requêtes ou des ordres dans ces prédicats [KI04].

La notion de profil utilisateur apparaît comme étant à la base de la personnalisation, mais elle est loin d'être définie de façon standard. Ainsi, dans [BK05], les auteurs tentent de classifier les différents types d'informations pouvant être contenus dans un profil et de définir un modèle de profil générique et flexible pouvant s'adapter à différents scénarios de personnalisation.

Ces profils sont ensuite utilisés dans le processus de traitement du système. Le contenu du profil peut être utilisé de différentes façons. Il peut remplacer la requête, permettre de l'enrichir (ajout de critères de sélection, de nouveaux mots-clés) ou être utilisé pour adapter les résultats, dans leur contenu (filtrage) ou dans leur forme de présentation.

3.3.2 Personnalisation dans les entrepôts de données

Si la personnalisation n'est pas une idée nouvelle dans les domaines précédemment évoqués, elle constitue un axe de recherche émergent dans le domaine des entrepôts de données. L'intérêt de cet axe de recherche peut être motivé à la fois vis-à-vis de la volumétrie des données connue pour être importante dans les entrepôts de données et du rôle central que joue l'utilisateur dans le processus décisionnel. En effet, il est en interaction directe avec le système au niveau de l'analyse des données,

en particulier dans le contexte de la navigation. Différentes pistes ont d'ores et déjà été initiées.

La première proposition s'inspire largement du domaine de la recherche d'informations (en RI ou en BD). En effet, il s'agit d'affiner la requête de l'utilisateur pour mieux répondre à ses besoins [BGMM06, BGM+05]. Dans ce cas, le concept de profil est utilisé. Il s'agit d'exprimer des préférences et de satisfaire des contraintes de visualisation. Ce travail trouve un intérêt particulier dans la mesure où l'aspect visualisation est primordial dans le contexte de l'analyse en ligne.

La seconde voie se focalise davantage sur l'utilisation du système et se rapproche davantage de ce qui se fait en IHM. En effet, dans [RTZ07], la personnalisation s'effectue au niveau de la navigation. Il s'agit de représenter les habitudes d'analyse de l'utilisateur, sous forme de coefficient de préférences, pour faciliter sa navigation.

Un autre type de travail est abordé dans [CGL+07]. Il s'agit de considérer une analyse en ligne comme une session interactive durant laquelle l'utilisateur lance des requêtes. Ainsi, il est intéressant que chaque utilisateur dispose de son propre espace de travail. L'objectif étant d'organiser les requêtes, de faciliter leur réutilisation et voire de partager ces requêtes avec d'autres utilisateurs.

Nous pouvons également citer des travaux qui se sont intéressés à rendre les entrepôts de données «actifs». Il s'agit de les munir de règles d'analyse devant être définies par les décideurs. Si la motivation mise en avant dans ces travaux n'est pas la personnalisation, il nous semble intéressant d'évoquer ces travaux qui permettent tout de même de placer les utilisateurs (autrement dit les analystes) au cœur du système. En effet, ces travaux ont pour objectif de reproduire le travail de l'analyste afin d'automatiser certaines tâches d'analyse récurrentes et éventuellement impacter les données sources de l'entrepôt en fonction des résultats de ces analyses [TSM01, TS02]. Par exemple, il s'agit de diminuer le prix de vente d'un produit dans la base de production, à la suite de l'exécution d'une règle d'analyse sur l'entrepôt. Ainsi, d'une certaine façon, ces travaux permettent de personnaliser l'utilisation de l'entrepôt de données. Ces travaux sont basés sur l'utilisation de règles «événement, condition, action» (ECA). Ainsi, l'exécution d'analyses dans les entrepôts de données est rendue plus flexible.

Enfin, mentionnons qu'afin de pouvoir rendre l'analyse plus flexible, un langage à base de règles a été développé dans [EV01] pour la gestion des exceptions dans le processus d'agrégation. Le langage IRAH (Intensional Redefinition of Aggregation Hierarchies) permet de redéfinir des chemins d'agrégation pour exprimer des exceptions dans les hiérarchies de dimensions prévues par le modèle. Tout comme pour

les entrepôts de données actifs, la motivation évoquée n'est pas la personnalisation. Néanmoins, ce travail permet aux utilisateurs d'exprimer eux-mêmes les exceptions dans le processus d'agrégation. En effet, afin de prendre en compte ces exceptions, les utilisateurs définissent et exécutent un programme IRAH, produisant ainsi une révision des chemins d'agrégation. L'exemple considéré est l'étude des prêts d'une compagnie de crédit en fonction de la dimension emprunteur qui est hiérarchisée. La catégorie de l'emprunteur est définie en fonction de son revenu. Mais les auteurs expliquent qu'il est possible que l'analyste veuille ré-affecter un emprunteur dans une autre catégorie en voulant tenir compte d'autres paramètres que le revenu. Dans ce cas, le processus d'agrégation doit tenir compte de cette «exception». Dans ces travaux les auteurs proposent alors un langage à base de règles qui permet de définir des analyses révisées qui tiennent compte de ce type d'exception. Ces travaux ont été élargis afin de proposer la maintenance des cubes de données dans ce contexte d'analyse «révisée» dans [EVT02]. L'objectif est d'éviter la reconstruction totale du cube, en ne recalculant que les cellules sujettes à une modification induite par la révision du cube. Si ce langage constitue une alternative à la rigidité du schéma multidimensionnel dans le processus d'agrégation pour les utilisateurs, il ne fait qu'en modifier les chemins, sans pour autant permettre la création de nouveaux axes d'analyse.

La motivation avancée par les auteurs des deux derniers travaux (entrepôts de données actifs et gestion d'exceptions dans le processus d'agrégation) n'est pas la personnalisation elle-même. Néanmoins, il nous a paru intéressant de les évoquer dans la mesure où les solutions proposées placent l'analyste au cœur du système en lui offrant la possibilité de transcrire ses propres règles d'analyse (entrepôts de données actifs), ou d'exprimer sa propre manière d'agréger les données au niveau des instances (gestion d'exception dans le processus d'agrégation)

3.4 Conclusion

Dans ce chapitre, nous avons tenté de fournir une vision globale de la problématique de l'évolution du modèle (schéma et données) dans les entrepôts de données et des solutions qui ont pu être proposées depuis quelques années. Ces solutions sont nécessaires pour faire face à la fois à l'évolution des besoins d'analyse et des sources de données, même si les spécificités liées à l'évolution des besoins d'analyse telles que la place de l'utilisateur dans ce processus d'évolution n'ont pas été pris en compte. Nous avons mené une étude comparative de ces travaux selon différents critères. Nous avons montré que la modélisation temporelle permet d'assurer une cohérence des analyses, mais que cette solution a un coût.

Nous tenons à exprimer l'idée que, pour nous, la problématique de l'évolution de modèle est bien différente de celle du rafraîchissement de l'entrepôt. Le rafraîchissement correspond à un processus représenté par la phase ETL qui consiste essentiellement à l'ajout de données provenant des sources, sans remettre en cause les données présentes dans l'entrepôt. Dans le cadre de l'évolution de modèle, on sous-entend des évolutions de schéma et des données qui traduisent une réalité mais pouvant aller à l'encontre du principe de non-volatilité des données.

Nous avons dressé une étude comparative des travaux portant sur l'évolution de modèle dans les entrepôts de données. Suite à cette étude, il apparaît qu'il y a un manque de lien entre l'évolution de l'entrepôt et l'origine de cette évolution. Comme nous l'avons évoqué précédemment, pour concevoir un modèle d'entrepôt de données, il est nécessaire de prendre en compte non seulement les sources de données, mais également les besoins d'analyse. Ainsi, lorsque les sources de données ou les besoins d'analyse évoluent, une évolution du modèle de l'entrepôt est peut-être nécessaire. Les différents travaux que nous avons présentés apportent des solutions à certains aspects de cette problématique. Par exemple, la maintenance de vues matérialisées permet d'assurer une certaine propagation de l'évolution des sources de données. Malheureusement, aucune solution ne prend réellement en considération l'émergence de besoins d'analyse.

Ainsi, l'enjeu réside dans le fait d'accorder à l'utilisateur une réelle place dans le processus décisionnel, au-delà de l'exploration des analyses possibles de l'entrepôt. Il s'agit ainsi de pouvoir répondre à des besoins de façon personnalisée, faisant face ainsi à l'émergence de besoins d'analyse qui n'est que trop peu considérée bien qu'elle soit réelle.

En effet, nous avons également montré l'importance de la personnalisation, qui a émergé dans le domaine de la recherche d'information (que ce soit dans les bases de données ou plus généralement sur Internet) et qui émerge de nos jours dans le contexte des entrepôts de données. Cette personnalisation peut s'apparenter à apporter au système davantage de flexibilité. Cette flexibilité peut être assurée en ayant recours à l'expression de règles, comme c'est le cas dans les entrepôts de données actifs. C'est ainsi que nous avons basé notre approche de personnalisation des analyses sur un modèle à base de règles. C'est ce que nous présentons dans le chapitre suivant.

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.

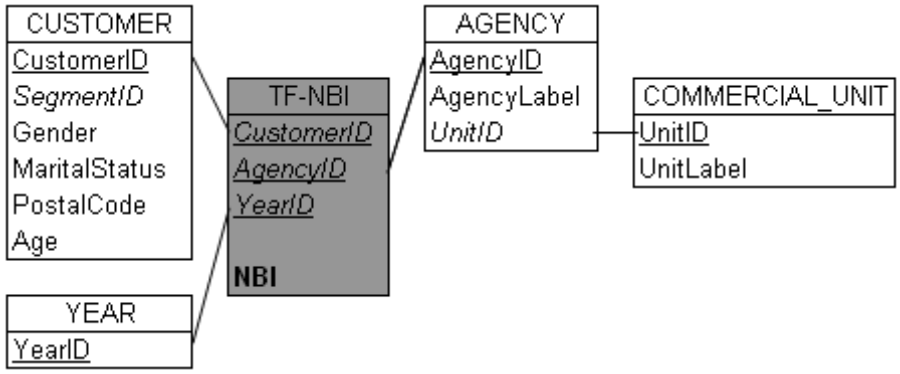


Fig. 4.1 – Schéma multidimensionnel pour observer le NBI

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.

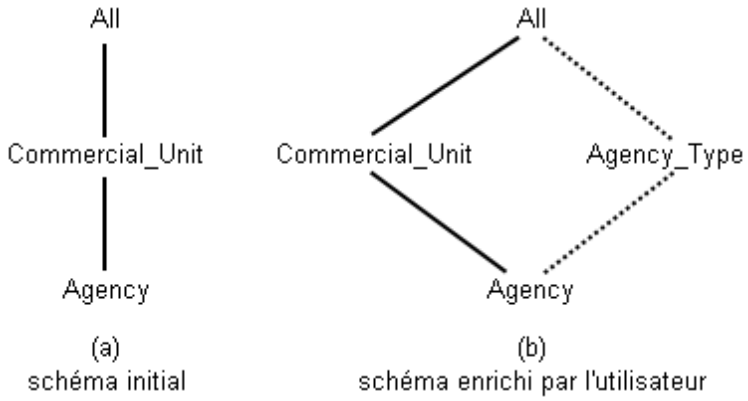


FIG. 4.2 – Schémas de la dimension `AGENCY`

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*).

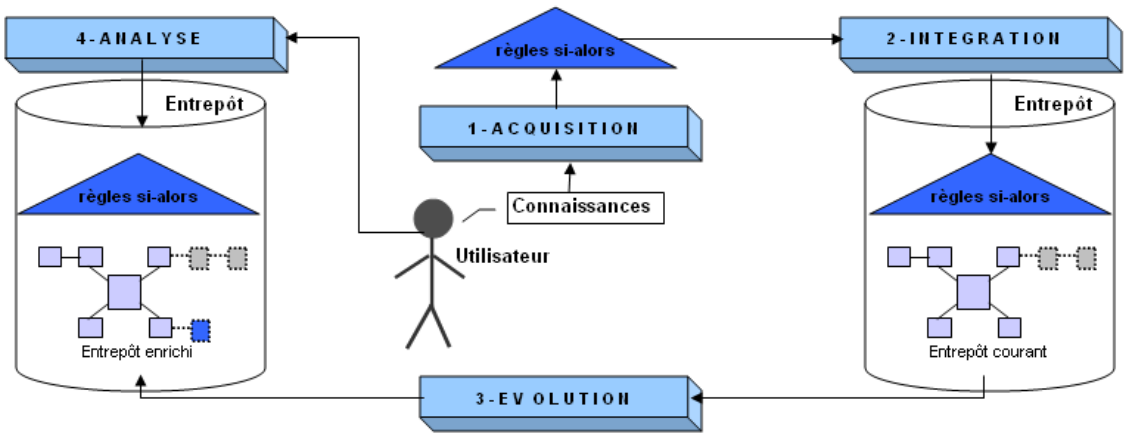


FIG. 4.3 – Architecture générale d'entrepôt de données évolutif guidé par les utilisateurs

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.

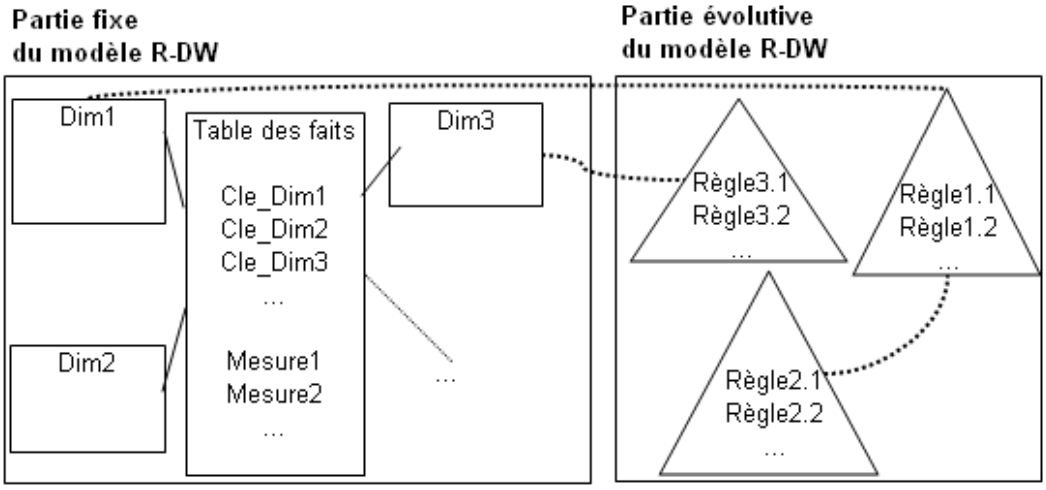


FIG. 4.4 – Principe du modèle *R-DW*

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.

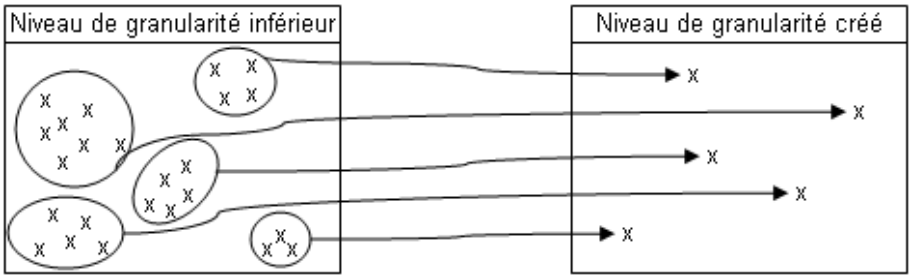


FIG. 4.5 – Définition des liens d'agrégation

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.

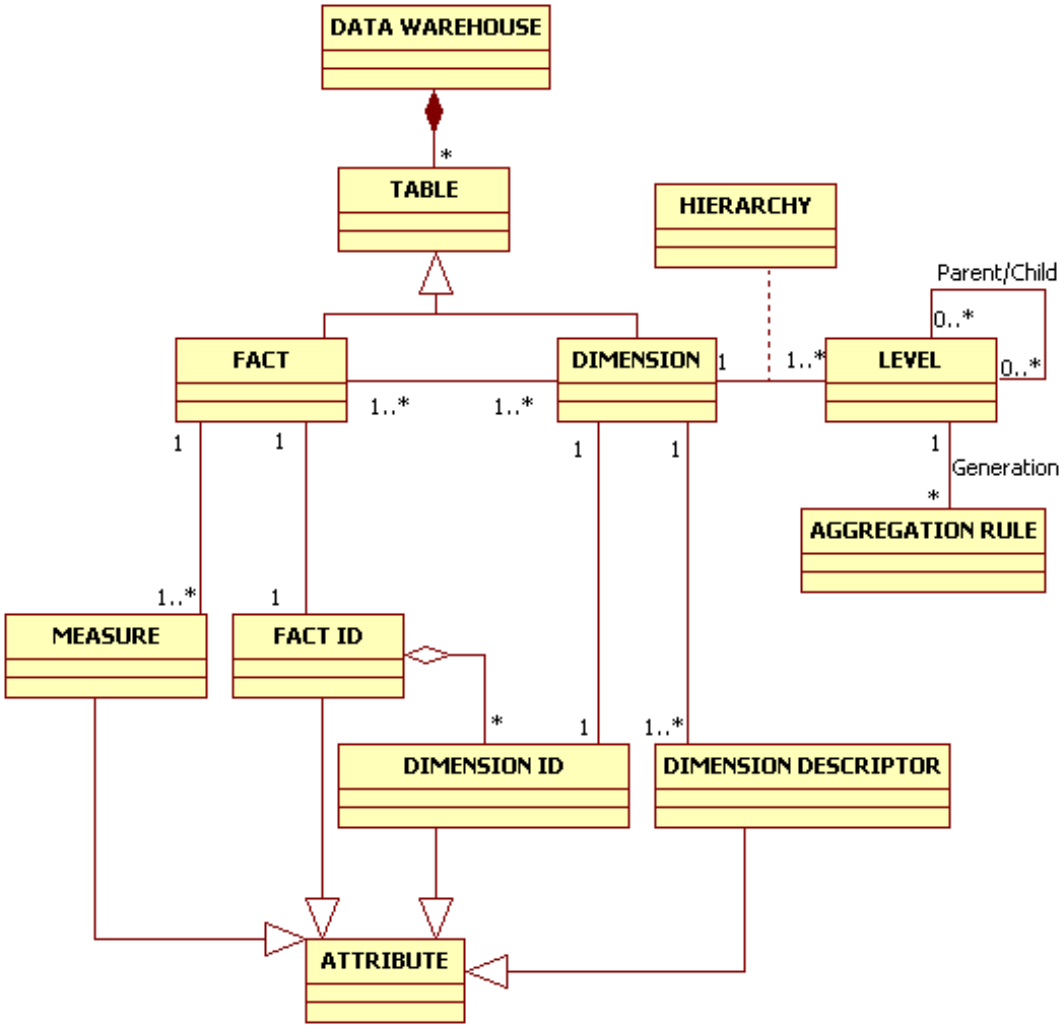


FIG. 4.6 – Méta-modèle pour gérer l'évolution du schéma de l'entrepôt

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

Chapitre 5

Mise à jour des hiérarchies de dimension

5.1 Introduction

Dans les bases de données, l'évolution de schéma peut être vue comme la mise en œuvre d'opérations élémentaires d'ajout, de suppression et de modification. Par exemple, dans le modèle objet, ces opérations sont appliquées soit au niveau des classes, soit au niveau des propriétés des classes. Dans le modèle relationnel, ces opérations sont appliquées au niveau des tables ou des attributs. Ainsi les concepts subissant les opérations ont tous le même rôle. Or ce n'est pas le cas dans la modélisation des entrepôts de données.

Si les travaux de recherche dans le domaine des entrepôts de données s'inspirent des travaux réalisés dans celui des bases de données, il n'en demeure pas moins que les entrepôts de données ont des spécificités qu'il faut prendre en compte. C'est le cas de l'évolution de schéma, comme le soulignent les différents travaux de recherche relatifs à cette problématique (chapitre 3). La sémantique portée par le schéma de l'entrepôt induit des considérations différentes sur ces concepts et sur leur évolution a fortiori. De ce fait, lorsque l'on parle d'évolution de schéma dans un entrepôt de données, on ne peut pas se baser sur les distinctions adoptées dans les bases de données. Il nous faut distinguer la mise à jour des tables de faits, de celle des tables de dimension, ou encore de celle des hiérarchies de dimension.

Une hiérarchie étant composée d'un ensemble de niveaux de granularité, la mise à jour concerne donc directement les niveaux de granularité. Ainsi, dans ce chapitre, nous proposons différents algorithmes pour réaliser la mise à jour des hiérarchies. Nous proposons un algorithme de création et de suppression de niveaux de granu-

larité. Nous proposons également un algorithme qui permet de gérer la propagation des mises à jour dans la hiérarchie. En effet, lorsqu'un niveau est créé ou supprimé en milieu de hiérarchie, il est nécessaire d'effectuer une propagation qui consiste à définir les liens d'agrégation requis.

Ensuite, nous proposons un modèle d'exécution qui gère les modules nécessaires à la réalisation des mises à jour des hiérarchies de dimension. Il permet donc de gérer les modules d'acquisition, d'intégration et d'évolution. Ce modèle d'exécution s'inscrit dans une approche relationnelle, il s'agit d'exploiter des structures relationnelles dans l'exécution des différents processus. Il comprend un ensemble d'algorithmes qui se basent sur le formalisme que nous avons proposé dans [FBB07c] et qui permet de représenter les différents concepts manipulés dans la mise en œuvre de notre approche.

Ce chapitre est organisé comme suit. Tout d'abord, la section 5.2 est consacrée à la présentation du processus de mises à jour de hiérarchies de dimension et de leur propagation dans le schéma de l'entrepôt. Puis, dans la section 5.3, nous déployons notre approche dans un contexte relationnel. La section 5.4 est consacrée à une discussion. Enfin, nous concluons dans la section 5.5.

5.2 Création et suppression de niveaux de hiérarchie

5.2.1 Création d'un nouveau niveau de hiérarchie

La création d'un niveau de granularité dans une hiérarchie consiste à créer ce niveau et le lien qu'il a avec le niveau inférieur sur lequel il est basé selon les règles d'agrégation exprimées. Cette création doit bien évidemment satisfaire les contraintes présentées dans le modèle *R-DW*, en l'occurrence : les contraintes liées aux concepts de partition d'une part et de bijection d'autre part (section 4.4).

Un niveau peut être créé au milieu d'une hiérarchie ou à la fin d'une hiérarchie. Lorsque nous parlons de création en fin de hiérarchie, cela correspond en fait soit à la création au-dessus du dernier niveau d'une hiérarchie existante, soit à la création au-dessus de la dimension elle-même, définissant ainsi une nouvelle hiérarchie.

Pour créer un niveau au milieu d'une hiérarchie, il faut que cela ait un sens d'agréger les données du niveau que l'on crée vers un niveau supérieur déjà existant, autrement dit les données sont agrégeables sémantiquement du niveau créé vers le niveau existant supérieur.

Pour illustrer cette création, nous prenons le cas de la hiérarchie de la dimension `AGENCY` issue de l'exemple LCL. Considérons le schéma de dimension de la

figure 5.1(a). Une unité commerciale correspond à un regroupement d'agences, selon leur localisation géographique. Il est alors possible de créer, entre les niveaux `AGENCY` et `COMMERCIAL UNIT`, un niveau qui correspondrait à un regroupement d'agences par quartier (`AGENCIES GROUP`), où une unité commerciale correspondrait donc à un regroupement de ces groupes d'agences, comme c'est le cas dans la figure 5.1(b). Dans ce cas, il faut définir le lien d'agrégation entre `AGENCIES GROUP` et `COMMERCIAL UNIT`.

En revanche, si l'on crée un nouveau niveau qui permet d'agréger les données des agences selon leur taille (petite, moyenne et grande), ce niveau `AGENCY SIZE` sera créé au-dessus de la dimension `AGENCY`, définissant ici une nouvelle hiérarchie pour cette dimension comme c'est illustré dans la figure 5.1(c).

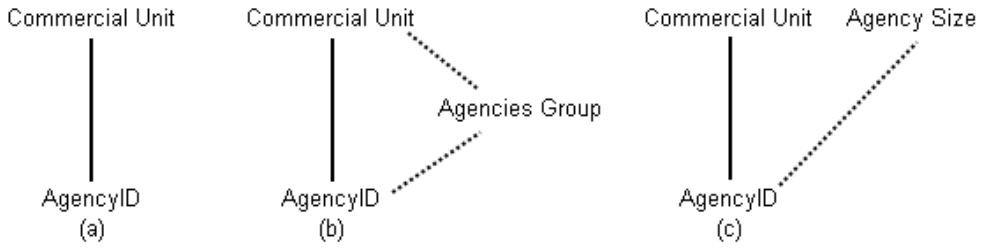


FIG. 5.1 – Schémas de la dimension `AGENCY` pour illustrer la création de niveaux

5.2.2 Suppression d'un niveau de hiérarchie existant

La suppression d'un niveau de granularité implique la suppression du niveau lui-même et du lien qu'il a avec le niveau au-dessus duquel il est construit. Cette suppression peut être réalisée sur un niveau placé au milieu ou à la fin d'une hiérarchie.

Par exemple, considérons le schéma de dimension de la figure 5.2(a). Une unité commerciale (`COMMERCIAL UNIT`) correspond à un regroupement d'agences par quartier (`AGENCIES GROUP`). Il est possible de supprimer le niveau `AGENCIES GROUP`. Dans ce cas, cette suppression implique de redéfinir le lien d'agrégation entre `AGENCY` et `COMMERCIAL UNIT` comme le montre la figure 5.2(b). Il est également possible de supprimer le niveau `COMMERCIAL UNIT`, auquel cas il suffit de supprimer le niveau et le lien qui le relie avec le niveau `AGENCIES GROUP` tel que le montre la figure 5.2(c).

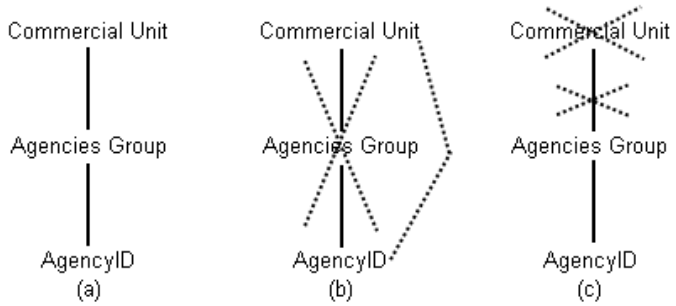


FIG. 5.2 – Schémas de la dimension `AGENCY` pour illustrer la suppression de niveaux

Ainsi, si un niveau est supprimé et que ce niveau est situé entre deux autres niveaux, il faut assurer le lien entre les deux niveaux en question. Ceci est réalisé lors de la propagation des mises à jour.

5.2.3 Propagation des mises à jour

Lorsqu'un niveau est supprimé ou créé et que ce niveau ne se situe pas en fin de hiérarchie, il est nécessaire de propager cette mise à jour dans la hiérarchie. Cette propagation consiste à assurer les liens requis entre les niveaux concernés de la hiérarchie mise à jour.

Dans le cas de la création d'un niveau de granularité au milieu d'une hiérarchie, la propagation consiste à définir le lien d'agrégation entre le niveau créé et le niveau supérieur. Par exemple, considérons le schéma de dimension de la figure 5.1(a). Lors de la création du niveau `AGENCIES GROUP` entre `AGENCY` et `COMMERCIAL UNIT`, la propagation consiste à définir le lien entre `AGENCIES GROUP` et `COMMERCIAL UNIT` tel que c'est illustré dans la figure 5.1(b).

Dans le cas de la suppression d'un niveau de granularité au milieu d'une hiérarchie, la propagation consiste à définir le lien d'agrégation entre le niveau inférieur et le niveau supérieur du niveau supprimé. Par exemple, considérons le schéma de dimension de la figure 5.2(a). Lors de la suppression du niveau `AGENCIES GROUP`, la propagation consiste à définir le lien entre `AGENCY` et `COMMERCIAL UNIT` tel que c'est illustré dans la figure 5.2(b).

Cette propagation, qui correspond à la définition de nouveaux liens entre niveaux, est réalisée de façon automatique, en déterminant ce lien à partir des liens existants. La détermination du lien se base sur les données. Ainsi, une fois que la structure prend en compte le lien, ce dernier est établi au niveau des instances.

Dans le cas de la suppression, le lien entre le niveau inférieur et le niveau supérieur du niveau à supprimer est établi avant que la suppression n'ait lieu. Dans le cas de la création, le lien entre le niveau créé et le niveau supérieur est déterminé après la création du nouveau niveau en se basant sur le lien existant entre le niveau inférieur et le niveau supérieur du niveau créé.

5.2.4 Algorithmes

Afin de clarifier la terminologie utilisée pour les niveaux de granularité dans les algorithmes, nous avons représenté les termes employés dans la figure 5.3. Le niveau courant correspond à celui qui subit la mise à jour, i.e. celui qui est créé ou supprimé.

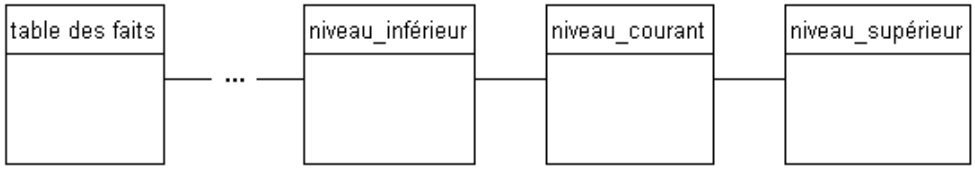


FIG. 5.3 – Terminologie des niveaux de granularité

L'algorithme 1 montre les étapes de suppression d'un niveau. Il s'agit d'abord de supprimer le lien entre le niveau inférieur et le niveau supprimé (ligne 1), avant de pouvoir supprimer le niveau lui-même (ligne 2). Ensuite, l'appel à la propagation est lancé si le niveau supprimé est placé au milieu de la hiérarchie (lignes 3 à 5).

Algorithme 1 Suppression d'un niveau de hiérarchie

Entrée: niveau_courant le niveau qui doit être supprimé, place_fin un booléen (=vrai si niveau_courant est en fin de hiérarchie)

1: Suppression dans niveau_inférieur de *id* (l'identifiant de niveau_courant dans niveau_inférieur)
2: Suppression de niveau_courant
3: **si** place_fin=faux **alors**
4: Propagation des mises à jour dans la hiérarchie (niveau_courant,suppression)
5: **fin si**

L'algorithme 2 montre les étapes de création d'un niveau. Il s'agit de créer la structure du nouveau niveau et d'y insérer les données (lignes 1 et 2). Puis le lien entre le niveau inférieur et le niveau créé doit être établi, au niveau de la structure (ligne 3), puis des données (lignes 4 à 6). Ensuite, l'appel à la propagation est lancé si le niveau n'est pas créé en fin de hiérarchie (lignes 7 à 9).

Algorithme 2 Création d'un niveau de hiérarchie

Entrée: place_fin un booléen (=vrai si niveau_courant est en fin de hiérarchie)

1: Création de la structure de niveau_courant : identifiant *id* et autres descripteurs
2: Insertion des données dans niveau_courant
3: Mise à jour de la structure de niveau_inférieur en ajoutant *id*
4: **pour tout** instance de niveau_inférieur **faire**
5: Mise à jour de la valeur de *id* dans niveau_inférieur
6: **fin pour**
7: **si** place_fin=faux **alors**
8: Propagation des mises à jour dans la hiérarchie (niveau_courant,création)
9: **fin si**

L'algorithme 3 permet de réaliser la propagation, que ce soit lors d'une création ou d'une suppression de niveau de granularité. Lors d'une création, il s'agit d'établir le lien entre le niveau créé et le niveau supérieur du point de vue de la structure d'une part (lignes 2 et 3), du point de vue des données d'autre part (lignes 4 à 7). Quand il s'agit d'une suppression, il faut créer le lien entre le niveau inférieur et le niveau supérieur du niveau supprimé sur le plan structurel d'une part (lignes 10 et 11) et sur le plan des données d'autre part (lignes 12 à 15).

Algorithme 3 Propagation des mises à jour dans la hiérarchie

Entrée: niveau_courant le niveau qui est créé ou supprimé, type_opération le type de l'opération subie par niveau_courant

{Création d'un niveau}
1: **si** type_opération=création **alors**
2: Détermination de *id* l'élément identifiant de niveau_supérieur
3: Création dans niveau_courant de *id*
4: **pour tout** instance de niveau_courant **faire**
5: Récupération de la valeur de *id* dans niveau_inférieur
6: Mise à jour de la valeur de *id* dans niveau_courant
7: **fin pour**
8: **fin si**
{Suppression d'un niveau}
9: **si** type_opération=suppression **alors**
10: Détermination dans niveau_courant de *id* l'élément identifiant niveau_supérieur
11: Création dans niveau_inférieur de *id*
12: **pour tout** instance de niveau_inférieur **faire**
13: Récupération de la valeur de *id* dans niveau_courant
14: Mise à jour de la valeur de *id* dans niveau_inférieur
15: **fin pour**
16: **fin si**

5.3 Déploiement de notre démarche dans un contexte relationnel

5.3.1 Principe général

Nous présentons ici la mise en œuvre de notre approche dans un contexte relationnel (ROLAP), en se focalisant sur la création de niveaux de granularité puisque c'est elle qui nécessite l'expression de règles utilisateurs pour répondre à leurs besoins d'analyse, contrairement à la suppression de niveau.

Afin de déployer notre approche dans un contexte ROLAP, nous présentons le modèle d'exécution que nous avons proposé dans [FBB07a]. Ce modèle d'exécution permet de mettre en œuvre les différents modules de notre architecture globale grâce à un enchaînement d'algorithmes (figure 5.4). Nous nous intéressons ici aux différentes étapes qui mènent jusqu'à la mise à jour des hiérarchies.

L'entrepôt de données évolutif étant stocké dans un SGBD relationnel (SGBDR), nous exploitons alors les structures relationnelles de ce SGBDR pour déployer notre approche. L'idée clé consiste alors à transformer les règles qui définissent un nouveau niveau de granularité en une table relationnelle, que nous appelons table de mapping.

Le point de départ de la séquence d'algorithmes est bien évidemment l'ensemble des règles exprimées par l'utilisateur grâce au module d'acquisition. Le module d'intégration permet alors de transformer les règles en une table de mapping. Cette table de mapping permet de stocker les règles sous une forme relationnelle et de vérifier la validité des règles vis-à-vis des contraintes que nous avons posées dans le modèle *R-DW*. Si les règles ne sont pas valides, au sens de ces contraintes, l'utilisateur doit les modifier. Lorsque celles-ci le sont, le module d'évolution permet de créer le niveau de granularité en fin de hiérarchie ou entre deux niveaux existants, nous parlons respectivement d'ajout ou d'insertion de niveau (i.e. création sans ou avec propagation).

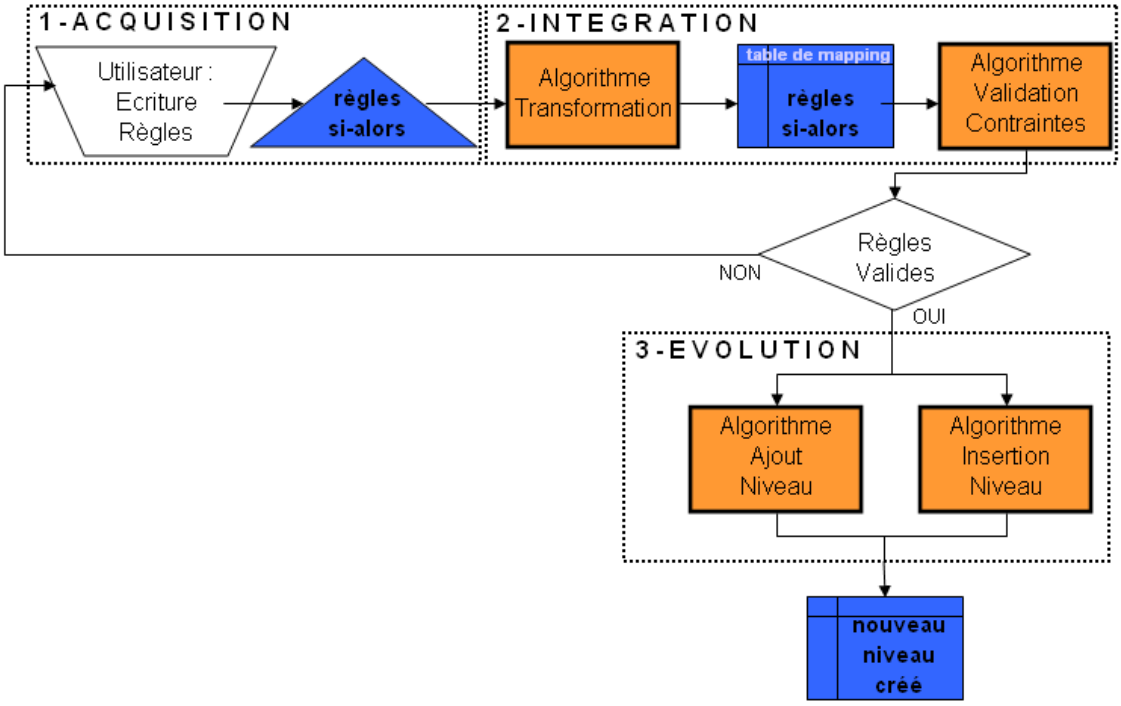


FIG. 5.4 – Modèle d'exécution pour l'évolution des hiérarchies de dimension

5.3.2 Règles et méta-règles

1. Principe

L'acquisition des règles se déroule en deux étapes :

1) *Définition d'une méta-règle d'agrégation qui précise la structure du lien d'agrégation entre deux niveaux.* En effet, la méta-règle permet à l'utilisateur de définir, le niveau de granularité qu'il veut créer, les attributs caractérisant ce nouveau niveau, le niveau de granularité existant à partir duquel il crée le nouveau niveau et les attributs du niveau existant utilisés pour créer le lien d'agrégation.
2) *Définition des règles d'agrégation qui déterminent le lien d'agrégation au niveau des instances.* Ces règles correspondent aux règles d'agrégation utilisées dans le modèle *R-DW*. Pour les définir, l'utilisateur instancie la méta-règle en exprimant une règle par instance du niveau généré afin d'y associer les instances du niveau existant.

2. Définitions

1) *Définition d'une méta-règle*

Soit EL le niveau existant.
Soit $\{EA_i,\ i = 1..m\}$ le sous-ensemble des m attributs pris parmi les m' attributs existants de EL, sur lesquels vont porter les conditions.
Soit GL le niveau généré et $\{GA_j,\ j = 1..n\}$ l'ensemble des n attributs générés de GL.
La méta-règle MR est définie de la façon suivante :

$$MR : if\, SelectionOn(EL, \{EA_i,\ i = 1..m\})\ then\, Generate(GL, \{GA_j,\ j = 1..n\})$$

2) *Définition de l'instanciation de la méta-règle*
Dans la formalisation du modèle *R-DW*, nous avons défini ce qu'est une règle d'agrégation. Une règle d'agrégation est basée sur un ensemble $\mathcal{T}$ de z termes de règles, notés RT_x, tel que :

$$\mathcal{T} = \{RT_x, 1 \leq x \leq z\} = \{EA_x\ op_x\ \{ens|val\}_x\}$$

où EA_x est un attribut du niveau inférieur, op_x est un opérateur soit relationnel ($=$, $<$, $>$, $\leq$, $\geq$, $\neq$, ...), soit ensembliste ($\in$, $\notin$, ...) et $\{ens|val\}_x$ correspond respectivement soit à un ensemble de valeurs, soit à une valeur finie. Ces valeurs appartiennent au domaine de définition de EA_x.

Nous notons ici $c_x = op_x\ \{ens|val\}_x$ la condition portée sur l'attribut EA_x incluant l'opérateur et la valeur ou l'ensemble de valeurs.

Soit q le nombre d'instances du niveau généré GL, un ensemble $R = \{r_d, d = 1..q\}$ de q règles d'agrégation doit être défini.

Ainsi, la condition se rapportant à l'attribut EA_i dans la règle r_d est notée c_{di}. Par ailleurs, nous notons v_{dj} la valeur attribuée à l'attribut généré GA_j dans la règle r_d.

Une règle r_d qui instancie la méta-règle MR est notée :

$$r_d : if\ RT_1\ AND\ ...\ AND\ RT_x\ AND\ ...\ AND\ RT_z$$

$$then\ GA_1 = v_{d1}\ AND\ ...\ AND\ GA_j = v_{dj}\ AND\ ...\ AND\ GA_n = v_{dn}$$

Nous avons donc, en utilisant le concept de condition :

$$r_d : if\ EA_1\ c_{d1}\ AND\ ...\ AND\ EA_i\ c_{di}\ AND\ ...\ AND\ EA_m\ c_{dm}$$

$$then\ GA_1 = v_{d1}\ AND\ ...\ AND\ GA_j = v_{dj}\ AND\ ...\ AND\ GA_n = v_{dn}$$

Cette définition est illustrée dans la figure 5.5.

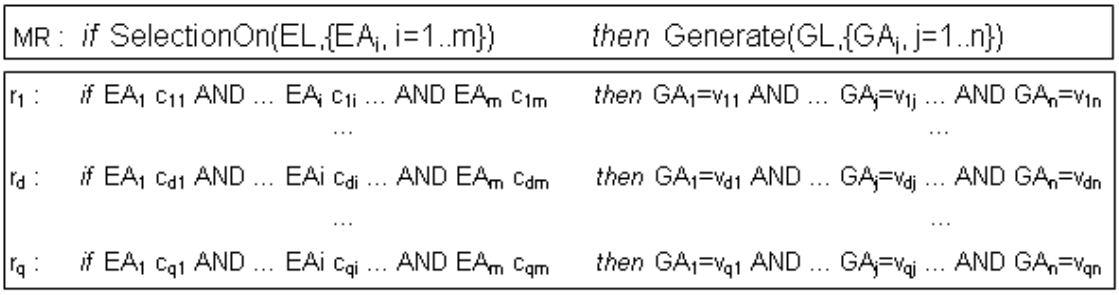


FIG. 5.5 – Illustration de la méta-règle et des règles d'agrégation

3. Exemple

Dans l'exemple de LCL, lors de la phase d'*acquisition*, l'utilisateur définit la méta-règle d'agrégation MR pour spécifier la structure du lien d'agrégation pour le type d'agence. Elle exprime donc le fait que le niveau `AGENCY_TYPE` va être caractérisé par l'attribut `AgencyTypeLabel` et qu'il sera créé au-dessus du niveau `AGENCY` ; les regroupements des instances de `AGENCY` se baseront sur des conditions exprimées sur l'attribut `AgencyID`.

MR : *if* SelectionOn(`AGENCY`,{`AgencyID`})
then Generate(`AGENCY_TYPE`{`AgencyTypeLabel`})

Puis l'utilisateur définit les règles d'agrégation qui instancient la méta-règle pour créer les différents types d'agence. Ainsi, il définit une règle pour chaque type d'agence, en exprimant à chaque fois la condition sur l'attribut `AgencyID` et en affectant à l'attribut généré `AgencyTypeLabel` sa valeur correspondante :

(R1) *if* `AgencyID` $\in$ {'01903','01905','02256'} *then* `AgencyTypeLabel`='student'
(R2) *if* `AgencyID` = '01929' *then* `AgencyTypeLabel`='foreigner'
(R3) *if* `AgencyID` $\notin$ {'01903','01905','02256','01929'} *then* `AgencyTypeLabel`='classical'

5.3.3 Table de mapping et méta-table de mapping

1. Principe

L'intégration des règles dans l'entrepôt se décompose en deux étapes :

1) **Transformation des règles en une table de mapping.** Il s'agit de transformer les règles de type «si-alors» en une table de mapping et de la stocker dans le SGBDR au moyen d'une structure relationnelle. Pour un ensemble donné de règles qui définit un nouveau niveau de hiérarchie, nous y associons une table de mapping. Pour réaliser cette transformation, nous exploitons dans un premier temps la méta-règle d'agrégation pour définir la structure de la table de mapping. Dans un deuxième temps, nous utilisons les règles d'agrégation elles-mêmes pour insérer les enregistrements correspondants dans la table de mapping. Cette table de mapping reprend les attributs sur lesquels portent des conditions, ainsi que les attributs générés caractérisant le niveau créé.
2) **Stockage des informations sur la table de mapping dans une méta-table de mapping.** Nous définissons une méta-table de mapping pour rassembler les informations sur les différentes tables de mapping. En effet, leur structure peut différer en fonction du nombre d'attributs supportant des conditions et du nombre d'attributs générés caractérisant le nouveau niveau. Elle contient également le rôle des attributs dans le processus.

2. Définitions

1) *Définition d'une table de mapping*

Soit ET (resp. GT) la table existante (resp. générée), correspondant à EL (resp. GL).
Soient $\{EA_i,\ i = 1..m\}$ l'ensemble des m attributs de ET et $\{GA_j,\ j = 1..n\}$ l'ensemble des n attributs générés de GT.
Soit GA_{key} l'attribut ajouté automatiquement qui sera la clé primaire de GT.

La table de mapping MT_GT construite à partir de la méta-règle MR correspond à la relation suivante :

$$MT\_GT(EA_1,\ ...,\ EA_i,\ ...,\ EA_m,\ GA_1,\ ...,\ GA_j,\ ...,\ GA_n,\ GA_{key})$$

Les règles d'agrégation permettent d'insérer dans cette table de mapping les conditions sur les attributs $\{EA_i,\ i = 1..m\}$ et les valeurs des attributs générés correspondants $GA_j,\ j = 1..n$. Pour l'attribut GA_{key}, la valeur est ajoutée de

façon incrémentale. Ainsi, pour chaque enregistrement d de la table de mapping, on a les conditions appliquées aux attributs ($c_{di},\ i = 1..m$) définissant quelles instances de la table ET sont concernées, à quelle instance elles correspondent dans GT. Cette dernière est caractérisée par les valeurs des attributs de la table GT ($v_{dj},\ j = 1..n$).
Cette définition est illustrée dans la figure 5.6.

MT_GT										
EA_1	...	EA_i	...	EA_m	GA_1	...	GA_j	...	GA_n	GA_{key}
c_{11}	...	c_{1i}	...	c_{1m}	v_{11}	...	v_{1j}	...	v_{1n}	key_1
...	...	...	...	...	...	...	...	...	...	...
c_{d1}	...	c_{di}	...	c_{dm}	v_{d1}	...	v_{dj}	...	v_{dn}	key_d
...	...	...	...	...	...	...	...	...	...	...
c_{q1}	...	c_{qi}	...	c_{qm}	v_{q1}	...	v_{qj}	...	v_{qn}	key_q

FIG. 5.6 – Illustration de la table de mapping

2) *Définition d'une méta-table de mapping*

La structure de la méta-table de mapping est représentée par la relation suivante :

$$MAPPING\_META\_TABLE(Mapping\_Table\_ID,\ Mapping\_Table\_Name,\ Table\_Name,\ Attribute\_Name,\ Attribute\_Type)$$

où $Mapping\_Table\_ID$ et $Mapping\_Table\_Name$ correspondent respectivement à l'identifiant et au nom de la table de mapping ; $Attribute\_Name$ et $Table\_Name$ caractérisent l'attribut impliqué dans le processus d'évolution et sa table d'appartenance ; $Attribute\_Type$ fournit le rôle de cet attribut.

Plus précisément, ce dernier attribut a trois modalités : «*conditioned*» s'il apparaît dans la clause «si» et «*generated descriptor*» ou «*generated key*» s'il est dans la clause «alors» selon qu'il s'agit d'un attribut clé ou non. Notons que même si un attribut a le rôle de «*generated descriptor*» ou «*generated key*» dans une table de mapping, il peut avoir le rôle «*conditioned*» dans une autre table de mapping pour servir à la construction d'un autre niveau. Ainsi, il est possible de construire deux niveaux successifs dans une hiérarchie.
Cette définition est illustrée dans la figure 5.7.

MAPPING_META_TABLE				
Mapping_Table_ID	**Mapping_Table_Name**	**Table_Name**	**Attribute_Name**	**Attribute_Type**
...	...	...	...	...
y	MT_GT	ET	EA_i	conditioned
y	MT_GT	...	...	...
y	MT_GT	GT	GA_j	generated descriptor
y	MT_GT	...	...	...
y	MT_GT	GT	GA_{key}	generated key
...	...	...	...	...

FIG. 5.7 – Illustration de la méta-table de mapping

3. Exemple

Dans l'exemple LCL, la phase d'*intégration* exploite la méta-règle et les règles d'agrégation R1, R2 et R3 pour générer la table de mapping MT_AGENCY_TYPE (figure 5.8) et les informations concernant cette table sont insérées dans la méta-table $MAPPING\_META\_TABLE$ (figure 5.9). Ainsi la table de mapping contient les attributs `AgencyTypeLabel` et `AgencyID` respectivement, que l'on retrouve dans la méta-table $MAPPING\_META\_TABLE$ et dont les rôles sont respectivement «*generated descriptor*» et «*conditioned*». La clé `AgencyTypeID` est rajoutée automatiquement et figure donc dans la méta-table avec le rôle «*generated key*».

MT_AGENCY_TYPE		
AgencyID	***AgencyTypeLabel***	***AgencyTypeID***
IN ('01903', '01905', '02256')	student	1
= '01929'	foreigner	2
NOT IN ('01903', '01905', '02256', '01929')	classical	3

FIG. 5.8 – Table de mapping pour le niveau `AGENCY_TYPE`

MAPPING_META_TABLE				
Mapping_Table_ID	***Mapping_Table_Name***	***Attribute_Table***	***Attribute_Name***	***Attribute_Type***
1	MT_AGENCY_TYPE	AGENCY	AgencyID	conditioned
1	MT_AGENCY_TYPE	AGENCY_TYPE	AgencyTypeLabel	generated descriptor
1	MT_AGENCY_TYPE	AGENCY_TYPE	AgencyTypeID	generated key

FIG. 5.9 – Méta-table de mapping prenant en compte le niveau `AGENCY_TYPE`

5.3.4 Création du niveau

3. Principe

Le processus d'évolution est réalisé à partir de la table de mapping dédiée à la création du niveau et des instances qui lui sont associées dans la méta-table. Quel que soit le type d'évolution appliqué (ajout ou insertion), la structure de la table GT est donc créée selon les attributs de la méta-table $MAPPING\_META\_TABLE$ qui comportent la mention «*generated key*», i.e. GA_{key}, pour la clé de la table et «*generated descriptor*» pour les autres attributs $\{GA_j,\ j = 1..n\}$.

Les instances de cette table sont insérées selon le contenu de la table de mapping MT_GT, en l'occurrence les valeurs de la clé $\{key_d,\ d = 1..q\}$ et les valeurs des autres attributs $\{v_{dj},\ d = 1..q,\ j = 1..n\}$.

La structure de la table ET est mise à jour avec l'ajout de l'attribut GA_{key} qui constitue une clé étrangère référençant la table GT (coloration en gris clair dans la figure 5.10). La valeur de cet attribut est mise à jour pour chacune des instances de la table en fonction des conditions exprimées sur les attributs de ET dans la table de mapping.

Dans le cas où GT est inséré entre ET et $ET2$, la table GT comporte également un attribut L qui constituera la clé étrangère référençant l'attribut L de $ET2$ pour établir le lien (liaison en pointillé sur la figure 5.10). Cet attribut est alimenté grâce à la valeur qu'il a dans la table ET (coloration en gris foncé dans la figure 5.10). Ainsi, dans le cas d'une insertion de niveau, le lien entre le niveau créé et le niveau supérieur est défini de façon automatique.

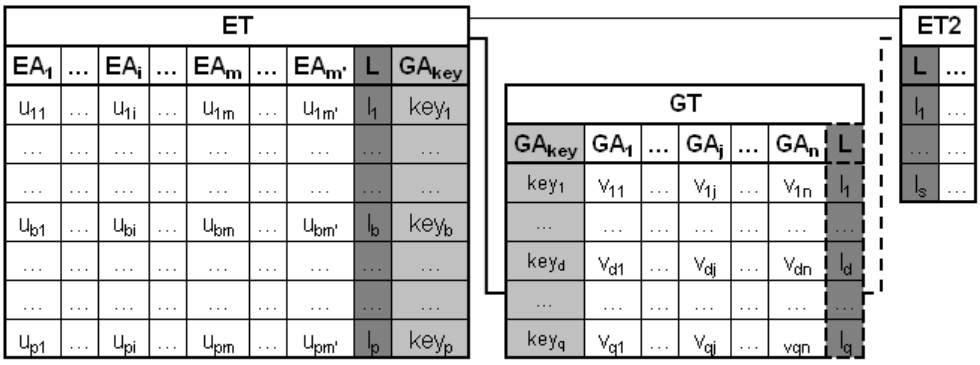
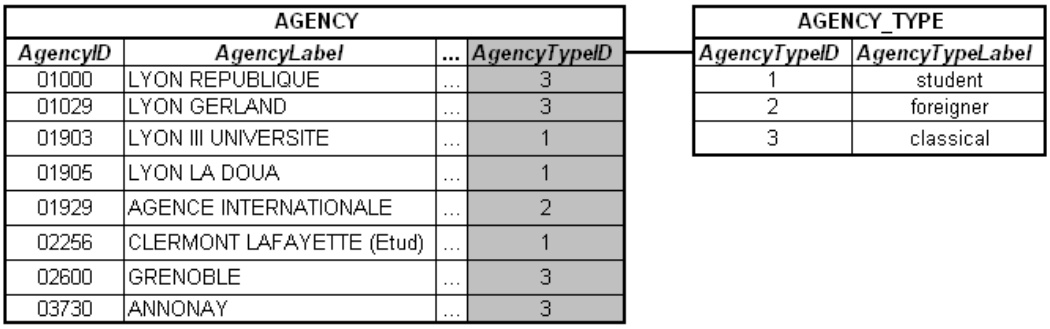


FIG. 5.10 – Évolution de la hiérarchie de dimension

2. Exemple

Dans notre exemple, l'évolution choisie par l'utilisateur correspond à un ajout.

En effet, il n'y a pas de lien possible d'agrégation entre le type d'agence et l'unité commerciale qui correspond à un regroupement géographique des agences. La phase d'*évolution* qui suit permet donc d'une part de créer et d'alimenter la table `AGENCY_TYPE` ; et d'autre part de mettre à jour la table `AGENCY` pour la relier à la table `AGENCY_TYPE`, avec l'ajout de l'attribut `AgencyTypeID` et la mise à jour de ses valeurs (figure 5.11).



AGENCY

AgencyID	*AgencyLabel*	...	*AgencyTypeID*
01000	LYON REPUBLIQUE	...	3
01029	LYON GERLAND	...	3
01903	LYON III UNIVERSITE	...	1
01905	LYON LA DOUA	...	1
01929	AGENCE INTERNATIONALE	...	2
02256	CLERMONT LAFAYETTE (Etud)	...	1
02600	GRENOBLE	...	3
03730	ANNONAY	...	3

AGENCY_TYPE

AgencyTypeID	*AgencyTypeLabel*
1	student
2	foreigner
3	classical

FIG. 5.11 – La table `AGENCY_TYPE` créée et la table `AGENCY` mise à jour

5.3.5 Algorithmes

Le modèle d'exécution exploite différents algorithmes afin de mettre en œuvre les modules de notre architecture. Ces algorithmes sont basés sur les concepts et les définitions que nous venons de présenter.

Concernant le module d'intégration, les règles sont donc transformées en une table de mapping (algorithme 4). Puis la validité de ces règles est vérifiée en testant le contenu de la table de mapping (algorithme 5). Le module d'évolution permet la création du niveau : soit le niveau de granularité est ajouté à la fin d'une hiérarchie, comme niveau le plus grossier de la hiérarchie (algorithme 6), soit il est inséré entre deux niveaux existants (algorithme 7).

5.3.5.1 Transformation des règles en une table de mapping

L'algorithme 4 permet de transformer les règles en une table de mapping. Plus précisément, pour une méta-règle MR et l'ensemble des règles d'agrégation qui lui sont associées R, nous construisons une table de mapping MT_GT (ligne 1). La structure de la table de mapping est définie à l'aide de la méta-règle. En effet, comme nous l'avons vu précédemment, la table de mapping contient à la fois les attributs du niveau inférieur sur lesquels porteront les conditions définissant le regroupement des instances et les attributs créés pour le nouveau niveau. La clé primaire de ce

nouveau niveau est également ajoutée de façon automatique. Ensuite, le contenu (les instances) de la table de mapping est déterminé par rapport à l'ensemble des règles d'agrégation (lignes 2 à 4). Chacune des règles d'agrégation correspond à une instance dans la table de mapping : les conditions portant sur les attributs du niveau inférieur et les valeurs des attributs du nouveau niveau sont donc reportées dans la table de mapping.

En outre, diverses informations sur la table de mapping sont insérées dans la méta-table $MAPPING\_META\_TABLE$ (lignes 5 à 10). Cela permet de connaître, à terme, les différentes tables de mapping et leur structure. Rappelons en effet que chaque table de mapping a sa propre structure puisque le nombre d'attributs sur lesquels portent des conditions et le nombre d'attributs dans le niveau créé sont variables.

Algorithme 4 Pseudo-code pour la transformation des règles en une table de mapping

Entrée: (1) méta-règle MR : *if* SelectionOn(EL,{EA_i}) *then* Generate(GL, {GA_j}) où EL est le niveau existant, {EA_i, $i = 1..m$} est l'ensemble des m attributs de EL, GL est le niveau à créer et {GA_j, $j = 1..n$} est l'ensemble des n attributs générés de GL ; (2) l'ensemble des règles d'agrégation R ; (3) la méta-table de mapping $MAPPING\_META\_TABLE$
Sortie: la table de mapping MT_GT
{Création de la structure de MT_GT}
1: CREATE TABLE MT_GT({EA_i},{GA_j})
{Insertion des valeurs dans MT_GT}
2: **pour tout** règle $\in R$ **faire**
3: INSERT INTO MT_GT VALUES (SelectionOn(EL,{EA_i}) , Generate(GL, {GA_j}))
4: **fin pour**
{Insertion des valeurs dans $MAPPING\_META\_TABLE$}
5: **pour tout** $EA_i \in \{EA_i\}$ **faire**
6: INSERT INTO $MAPPING\_META\_TABLE$ VALUES(MT_GT, EL, EA_i, 'conditioned')
7: **fin pour**
8: **pour tout** $GA_j \in \{GA_j\}$ **faire**
9: INSERT INTO $MAPPING\_META\_TABLE$ VALUES(MT_GT, GL, GA_i, 'generated')
10: **fin pour**
11: **return** MT_GT

5.3.5.2 Vérification des contraintes

L'algorithme 5 vise à vérifier que les règles satisfont les contraintes liées à la partition et à la bijection. Cette vérification se fait par rapport aux données contenues dans l'entrepôt suivant cette méthode :

1) Pour chaque enregistrement de MT_GT (ce qui correspond à une règle), nous formulons la requête correspondante afin de construire la vue qui contient l'ensemble des instances concernées de ET (lignes 2 à 4).
2) Vérification de la contrainte liée au concept de partition

- Vérifier qu'aucune des vues n'est vide (lignes 5 à 10)
- Vérifier que l'intersection des vues prises deux à deux est vide (lignes 11 à 16)
- Vérifier que l'union des instances de l'ensemble des vues correspond à l'ensemble des instances de la table ET (lignes 17 à 23).

3) Vérification de la contrainte liée au concept de bijection. Cette contrainte est vérifiée si les contraintes liées au concept de partition sont satisfaites. Compte tenu de la forme de la table de mapping et de son contenu, il suffit alors de vérifier que le nombre de valeurs distinctes (non nulles) correspondant au nombre d'instances du niveau créé est égal au nombre de ligne de la table MT_GT, cela représente l'idée qu'une instance du niveau créé n'est pas mise en correspondance avec plus d'un sous-ensemble d'instances du niveau inférieur et que chaque sous-ensemble a son instance correspondante (lignes 24 à 30). Notons que nous ne pouvons nous baser sur la valeur de la clé GA_{key} puisque celle-ci est générée automatiquement lors de la création de la table de mapping.

Algorithme 5 Pseudo-code pour vérifier les contraintes

Entrée: table existante ET, table de mapping MT_GT
Sortie: NonEmpty_checked, Intersection_checked, Union_checked et Bijection_checked quatre booléens
{Initialisation des booléens}
1: NonEmpty_checked=true ; Intersection_checked=true ; Union_checked=true ; Bijection_checked=true
{Création des vues}
2: **pour tout** tuple $t \in MT\_GT$ **faire**
3: CREATE VIEW v_t AS SELECT * FROM ET WHERE Conjunction($t(\{EA\})$)
4: **fin pour**
{Vérification de la contrainte de partition}
{Vérification qu'aucune des vues n'est vide}
5: **pour** $x = 1$àt_{max} **faire**
6: nb=SELECT COUNT(*) FROM v_x
7: **si** $nb \neq 0$ **alors**
8: NonEmpty_checked=false
9: **fin si**
10: **fin pour**
{Vérification que l'intersection des vues prises deux à deux est vide}
11: **pour** $x = 1$ à $(t_{max} - 1)$ **faire**
12: I=SELECT * FROM v_x INTERSECT SELECT * FROM $v\_x + 1$
13: **si** $I \neq \emptyset$ **alors**
14: Intersection_checked=false
15: **fin si**
16: **fin pour**
{Vérification que l'union de toutes les vues correspond à la table ET}
17: U=SELECT * FROM v_1
18: **pour** $x = 2$ à t_{max} **faire**
19: U=U UNION SELECT * FROM v_x
20: **fin pour**
21: **si** U $\neq$ SELECT * FROM ET **alors**
22: Union_checked=false
23: **fin si**
{Vérification de la bijection}
24: **si** (NonEmpty_checked **et** Intersection_checked **et** Union_checked) **alors**
25: nb'=SELECT COUNT(DISTINCT GA_j) FROM MT_GT
26: **si** $nb' \neq$ SELECT COUNT(*) FROM MT_GT **alors**
27: Bijection_checked=false
28: **fin si**
29: **return** Bijection_checked
30: **fin si**
31: **return** NonEmpty_checked, Intersection_checked, Union_checked

5.3.5.3 Algorithmes de création d'un niveau de hiérarchie

L'algorithme 6 permet d'ajouter (fin de hiérarchie) une nouvelle table GT, à partir d'une table existante ET, en exploitant une table de mapping MT. L'opération s'effectue en trois étapes : (1) créer la table GT (ligne 1) ; (2) modifier la structure de la table ET pour la lier à GT (ligne 2) ; (3) insérer dans GT les valeurs nécessaires en fonction de la table de mapping (lignes 3 à 6) .

Algorithme 6 Pseudo-code pour ajouter un nouveau niveau de hiérarchie

Entrée: (1) table existante ET, (2) table de mapping MT, (3) $\{EA_i,\ i = 1..m\}$ l'ensemble des m attributs de MT qui contiennent les conditions sur les attributs, (4) $\{GA_j,\ j = 1..n\}$ l'ensemble des n attributs générés de MT qui contiennent les valeurs des attributs générés, (5) GA_{key} l'attribut clé primaire de GT
Sortie: table générée GT

```
    {Création de la table GT}
 1: CREATE TABLE GT (GA_key, {GA_j});
    {Mise à jour de la structure de ET pour la lier à GT}
 2: ALTER TABLE ET ADD (GA_key);
    {Alimentation de GT, liaison entre ET et GT}
 3: pour tout (tuple t of MT) faire
 4:    INSERT INTO GT VALUES (GA_key,{GA_j});
 5:    UPDATE ET SET ET.GA_key = GT.GA_key WHERE {EA_i};
 6: fin pour
 7: return GT
```

L'algorithme 7 permet d'insérer une nouvelle table GT entre deux tables existantes (ET et $ET2$), en prenant en compte une table de mapping MT. Cette table de mapping ne contient que le lien entre la table inférieure ET et la table générée GT. Les étapes sont les mêmes que pour l'ajout d'un niveau. Ensuite, le lien d'agrégation est déterminé de façon automatique entre GT et $ET2$, en le déterminant à partir du lien existant entre ET et $ET2$ (ligne 6).

Algorithme 7 Pseudo-code pour insérer un nouveau niveau de hiérarchie

Entrée: table existante inférieure ET, table existante supérieure $ET2$, table de mapping MT, $\{EA_i,\ i = 1..m\}$ l'ensemble des m attributs de MT qui contiennent les conditions sur les attributs, $\{GA_j,\ j = 1..n\}$ l'ensemble des n attributs générés de MT qui contiennent les valeurs des attributs générés, GA_{key} l'attribut clé primaire de GT, L l'attribut liant ET à $ET2$
Sortie: table générée GT

```
    {Création de la table GT}
 1: CREATE TABLE GT (GA_key, {GA_j}, L);
    {Mise à jour de la structure de ET pour la lier à GT}
 2: ALTER TABLE ET ADD (GA_key);
    {Alimentation de GT, liaison entre ET et GT, liaison entre GT et ET2}
 3: pour tout (tuple of MT) faire
 4:    INSERT INTO GT VALUES (GA_key,{GA_j});
 5:    UPDATE ET SET ET.GA_key = GT.GA_key WHERE {EA_i};
 6:    UPDATE GT SET L=(SELECT DISTINCT L FROM ET WHERE ET.GA_key=GT.GA_key AND {EA_i});
 7: fin pour
 8: return GT
```

5.3.5.4 Complexité des algorithmes de mises à jour

Pour étudier la complexité des algorithmes de création de niveau (ajout, insertion), nous nous focalisons sur les opérations d'écriture et de lecture. Nous supposons ici que la création de la structure d'une table ou que la modification d'une structure de table existante par l'ajout d'un attribut sont négligeables. La complexité est me-

surée en terme de nombre d'opérations de lecture ou d'écriture d'un attribut pour un enregistrement donné.

Pour l'ajout d'un niveau de hiérarchie, il y a deux types d'opérations d'écriture : les insertions dans la nouvelle table GT et les mises à jour dans la table existante ET pour créer le lien entre ces deux tables. La complexité de ces opérations est de :

$$\omega((p + n + 1) \times q + p)$$

En effet, dans la table GT, il y a $n + 1$ attributs et q enregistrements. Donc il y a $(n + 1) \times q$ opérations d'écriture (insertion des valeurs). Pour faire le lien entre ET et GT, la valeur de l'attribut représentant la clé étrangère est mise à jour pour chaque enregistrement de la table ET, soit concernant le nombre d'opérations d'écriture : p. Ainsi, le total des opérations d'écriture est de : $p + (n + 1) \times q$. Pour les opérations de lecture, afin de mettre à jour la valeur de la clé étrangère, pour chacun des q enregistrements de la méta-table, les p enregistrements de la table ET sont parcourus, ainsi on obtient pour les opérations de lecture : $p \times q$.

L'insertion d'un niveau de hiérarchie entre deux niveaux existants comprend, en plus des opérations décrites précédemment dans le cas de l'ajout, une opération de mise à jour pour lier la table générée GT avec la table existante de niveau supérieur $ET2$. La complexité totale de ces opérations est de :

$$\omega((p + n + 2) \times q + p)$$

En effet, pour mettre à jour l'attribut représentant la clé étrangère dans le niveau créé GT, il y a q opérations. L'opération de lecture qui y est associée est assimilable à celle qui est réalisée lors de l'ajout et n'augmente donc pas la complexité.

Ainsi notre approche est plus pertinente lorsque q (nombre d'enregistrements dans la table générée GT) et p (nombre d'enregistrements dans la table existante ET) sont petits du point de vue de la complexité, ce qui est le cas pour des niveaux de granularité élevé. C'est également le cas d'un point de vue pratique, puisque l'utilisateur doit définir une règle d'agrégation par instance du niveau créé. Ainsi, si q est élevé, l'utilisateur doit définir un grand nombre de règles, ce qui rend l'approche très fastidieuse. Notre approche trouve donc son utilité pour créer de nouveaux agrégats lorsque les niveaux d'agrégation créés comportent peu d'instances de regroupement.

5.4 Discussion

5.4.1 Extension des possibilités d'analyse

Différents logiciels décisionnels de restitution de données tels que Business Object[1] fournissent aux utilisateurs une interface qui leur permet de manipuler les données comme ils le souhaitent. Une des fonctionnalités qu'ils offrent est la création de variables, avec la possibilité de regrouper des instances de dimension. Il s'agit donc également de personnaliser les analyses. Néanmoins cette fonctionnalité diffère réellement de notre approche sur les points suivants.

Tout d'abord, au niveau de l'expressivité du regroupement, mentionnons que les instances doivent être pointées, il n'y a pas de «règles» exprimables de regroupement. Cela rend la tâche d'autant plus fastidieuse lorsqu'il y a un certain nombre d'instances à regrouper et cela peut être source d'erreurs.

Ensuite, cette création de variable se fait au niveau des rapports (état d'analyse) directement. La création de ces regroupements n'est donc pas persistante, ni vis-à-vis d'autres états d'analyse, ni par rapport aux autres utilisateurs ; autrement dit la nouvelle possibilité d'analyse générée n'est pas partageable avec d'autres utilisateurs.

Ainsi notre approche permet cette création de variable en allant bien au-delà. En effet, il s'agit de créer un nouveau niveau avec la possibilité de créer plusieurs descripteurs pour ce niveau. En outre, grâce à l'usage de règles, nous permettons une plus grande expressivité dans les conditions de regroupement, avec une source d'erreurs moindre. Enfin, en proposant une évolution de modèle, nous rendons persistantes les nouvelles possibilités d'analyse créées individuellement, que ce soit pour le créateur lui-même, ou pour d'autres utilisateurs.

5.4.2 Évolution des règles

Jusque là, nous avons traité d'évolution de schéma grâce à l'utilisation de règles. Mais une fois ces règles exprimées, il est possible qu'elles-mêmes évoluent et ce, pour deux raisons. Le premier cas se produit si l'utilisateur souhaite les modifier, autrement dit, changer la définition du niveau créé. Le second cas se produit lorsque le rafraîchissement des données dans l'entrepôt rendent les règles obsolètes. Il est possible par exemple que les conditions de partition ne soient plus satisfaites suite à l'ajout d'instances dans un niveau inférieur. Notons d'ailleurs que, dans le cas de mises à jour sur les données (ajout, suppression, modification), les contraintes

[1] http ://www.france.businessobjects.com/

doivent être re-vérifiées. Selon les fonctionnalités du SGBDR, on peut envisager le recours à des déclencheurs (triggers) paramétrés pour réeffectuer les vérifications de façon automatique.

Si les règles définissant un niveau sont modifiées, la table de mapping doit subir également les modifications, ces dernières devront être répercutées sur le niveau de hiérarchie impacté. Ainsi, le problème de l'évolution des règles s'apparente finalement à celui de l'évolution de modèle telle que nous l'avons définie dans l'état de l'art. On peut alors réaliser soit une mise à jour des règles, soit un versionnement de celles-ci, qui conduiront respectivement à une mise à jour ou à un versionnement des hiérarchies de dimension.

Se posent alors différentes questions. Doit-on adopter une solution de façon systématique ? En effet, n'est-t-il pas préférable d'adopter des solutions différentes en fonction de cas différents, comme l'évoquait Kimball à travers ses trois types de «dimensions changeantes à évolution lente» [Kim96]. S'il y a une distinction à faire selon les cas, qui doit la faire ? Est-ce le rôle de l'utilisateur qui définit le niveau de granularité en question ? Est-ce que sa tâche ne s'en trouve pas trop complexifiée ? Autant de questions qui méritent d'être soulevées et qui doivent être approfondies.

5.5 Conclusion

Dans ce chapitre, nous avons présenté la mise à jour des hiérarchies de dimension. Cette mise à jour est basée sur la création et la suppression de niveaux de granularité. La création permet quant à elle la réponse à des besoins d'analyse émergents et nécessitent l'expression des règles d'agrégation. Ces actions de création et de suppression de niveaux de granularité peuvent nécessiter une propagation dans la hiérarchie de dimension lorsque les niveaux en question ne se situent pas en fin de hiérarchie.

Notre modèle d'exécution, proposé dans un contexte relationnel, gère l'ensemble des processus liés aux modules de notre architecture globale. Il se base ainsi sur des structures relationnelles en gérant la transformation des règles en une table de mapping qui est exploitée durant toute la démarche. Ainsi, ce modèle d'exécution permet de mettre en œuvre l'évolution du schéma de l'entrepôt, qui est rendue possible grâce au modèle d'entrepôt évolutif que nous avons proposé.

Nous abordons alors dans le chapitre suivant la problématique de l'évaluation de ce modèle évolutif.

Vers une évaluation des modèles d'entrepôt de données évolutifs

Résumé

Afin d'évaluer les performances de notre approche, nous nous sommes intéressés à l'évolution incrémentale de la charge (ensemble de requêtes) en fonction des changements subis par le schéma de l'entrepôt. Dans ce chapitre, nous présentons notre approche d'évolution de charge qui a pour but de permettre l'évaluation de modèles d'entrepôt évolutifs. L'objectif est double : maintenir la cohérence des requêtes existantes de la charge vis-à-vis de l'évolution du schéma, générer de nouvelles requêtes pour traduire les besoins d'analyse potentiels lorsque l'évolution du schéma de l'entrepôt engendre un enrichissement des possibilités d'analyse.

Sommaire

Chapitre 6

Vers une évaluation des modèles d'entrepôt de données évolutifs

6.1 Introduction

L'évaluation des modèles d'entrepôts de données se base généralement sur une charge (ensemble de requêtes). En effet, cette charge permet de tester la performance de l'exploitation d'un modèle.

L'exploitation efficace d'un entrepôt de données nécessite la mise en place de structures d'optimisation telles que des index et/ou des vues matérialisées. La sélection de ces structures, basée sur la charge qui représente les requêtes des utilisateurs de l'entrepôt, a pour but de fonder l'optimisation sur les besoins d'exploitation des utilisateurs eux-mêmes. L'évolution du schéma de l'entrepôt (en l'occurrence sa mise à jour) nécessite alors la maintenance de ces structures d'optimisation redondantes, mais au-delà de la maintenance, il peut être nécessaire de redéployer la stratégie de sélection.

Dans ce chapitre, nous présentons la méthode que nous avons proposée dans [FBB07d] pour faire évoluer la charge. Il s'agit d'une part de maintenir la cohérence des requêtes existantes vis-à-vis du nouveau schéma de l'entrepôt. D'autre part, nous proposons la création de nouvelles requêtes en réponse à certaines évolutions pour prendre en compte de nouveaux niveaux de hiérarchie créés par les utilisateurs. Notre objectif est en effet de permettre une pro-activité par rapport à l'évolution du schéma de l'entrepôt, en évitant d'attendre des requêtes posées sur l'entrepôt évolué pour appliquer la stratégie d'optimisation.

Pour atteindre cet objectif, nous proposons une typologie des changements pouvant être opérés sur le schéma de l'entrepôt avec les conséquences sur la charge.

Nous présentons le processus général de mise en œuvre de notre méthode. Nous fournissons un algorithme, basé sur la typologie, afin de permettre la modification des requêtes existantes de la charge et la création d'autres.

Ce chapitre est organisé de la façon suivante. Tout d'abord, nous proposons dans la section 6.2 un état de l'art qui se décompose en deux parties. La première partie traite de l'optimisation de performances dans les entrepôts de données, en particulier, du choix de la stratégie d'optimisation. La seconde partie présente l'évolution des requêtes comme étant un problème à part entière. Par la suite, nous dressons, dans la section 6.3, notre typologie des changements. Dans la section 6.4, nous présentons notre approche pour adapter les requêtes d'une charge donnée, afin de supporter l'évolution de la stratégie d'optimisation. Nous y précisons le cadre général d'application, ainsi que la mise en œuvre au moyen d'un algorithme d'évolution de charge. Puis nous présentons un exemple illustrant notre méthode dans la section 6.5. Nous développons ensuite une discussion de notre approche dans la section 6.6. Enfin, nous concluons et indiquons les perspectives de recherche de ce travail dans la section 6.7.

6.2 État de l'art

6.2.1 Optimisation des performances dans les entrepôts de données

La modélisation multidimensionnelle fournit une vue consolidée des données utilisées pour supporter le processus de décision. Cette vue est généralement basée sur un ensemble de sources hétérogènes. Le schéma multidimensionnel constitue alors un des piliers de l'architecture décisionnelle. Ainsi un des points-clés du succès d'un processus d'entreposage de données est la conception de ce schéma en fonction des sources de données disponibles et des besoins d'analyse.

Ces sources de données sont amenées à évoluer, tout comme les besoins d'analyse. Il est alors nécessaire de faire évoluer en conséquence le schéma. Rappelons que dans la littérature, il existe deux approches pour prendre en compte cette évolution : la mise à jour et le versionnement.

Les entrepôts de données contiennent un large volume de données. Afin de répondre aux requêtes de manière efficace, il est nécessaire d'avoir recours à des méthodes d'accès aux données efficaces. Une alternative possible est d'utiliser des structures redondantes. En effet, parmi les techniques issues des implémentations relationnelles des entrepôts de données pour améliorer le temps de réponse des requêtes décisionnelles, la matérialisation de vues et l'indexation sont très efficaces [RS03]. Un des aspects les plus importants dans la conception physique de l'entrepôt est

justement de sélectionner un ensemble approprié de vues à matérialiser et d'index, qui permet de minimiser le temps de réponse des requêtes, compte tenu d'un espace de stockage limité [Bel00].

Un choix judicieux pour la sélection d'index et de vues doit être basé sur le coût et guidé par les requêtes posées par les utilisateurs. En effet, il est crucial d'adapter la performance du système en fonction de son utilisation [Gal02]. Dans cette perspective, la charge de l'entrepôt doit correspondre à un ensemble de requêtes posées par les utilisateurs.

Les approches les plus récentes analysent syntaxiquement les requêtes de la charge pour en extraire les candidats pertinents (index et vues) [ACN00]. Par exemple, dans [AJD06], les auteurs proposent une solution pour la sélection de vues matérialisées qui exploite une technique de fouille de données (clustering) déterminant des ensembles de requêtes similaires à partir de l'analyse des requêtes de la charge. La sélection en elle-même se base sur des modèles de coût qui évaluent le coût d'accès aux données en utilisant les vues et le coût de stockage de ces vues.

La charge est supposée représenter les requêtes des utilisateurs sur l'entrepôt de données en question. La plupart des approches proposées dans la littérature sont basées sur l'idée qu'il existe une charge de référence qui représente les besoins cibles pour l'optimisation [TB00]. Cependant, dans [GS03], les auteurs mettent en avant le fait que les charges de requêtes réelles sont beaucoup plus volumineuses que celles qui peuvent être prises en compte par ces techniques. Ainsi, le succès de la matérialisation de vues et de l'indexation dépend en pratique de l'expérience du concepteur. Dans ce contexte, les auteurs proposent une approche pour définir une charge représentative de la charge originale, en se basant sur le concept de clustering, permettant ainsi l'application des algorithmes de sélection de vues à matérialiser et d'index.

La stratégie d'optimisation qui consiste à sélectionner des index et des vues à matérialiser est ainsi définie en fonction de l'utilisation de l'entrepôt de données. Or, l'évolution du schéma de l'entrepôt nécessite non seulement une évolution des structures redondantes dans le cas de la maintenance de vues matérialisées par exemple [HMV99], mais aussi une évolution de la stratégie d'optimisation elle-même : la configuration d'index et de vues choisie peut être amenée à évoluer.

En matière de sélection de vues, différents travaux se sont intéressés à la sélection dynamique [KR99, LRC06, TS00] qui constitue une réponse à l'évolution de la charge. Cependant, l'évolution de la charge peut être considérée selon différents point de vues. Dans [LRC06], les auteurs considèrent que la sélection de vues doit

être réalisée à des intervalles de maintenance réguliers et se basent sur un changement observé ou attendu des fréquences de requêtes. Dans [KR99, TS00], les auteurs supposent que des requêtes sont ajoutées à la charge initiale. Ainsi, ces différents travaux supposent que les requêtes de la charge initiale sont toujours correctes. Or, un changement dans le schéma de l'entrepôt peut rendre les requêtes incohérentes.

Notre idée clé consiste alors à fournir une solution pour faire évoluer la charge, afin de soutenir le travail de l'administrateur. L'objectif est de maintenir les requêtes cohérentes au travers de l'évolution du schéma de l'entrepôt et d'en créer de nouvelles. En effet, dans un contexte où le temps de réponse aux requêtes est un aspect crucial (contexte d'analyse en ligne), il nous paraît intéressant d'adopter une démarche pro-active en exprimant des requêtes qui tiennent compte des nouvelles possibilités d'analyse induites par l'évolution du schéma. Cette démarche pro-active permet d'éviter d'attendre l'utilisation du schéma mis à jour pour obtenir une nouvelle charge cohérente avec le schéma courant.

Notre objectif étant l'optimisation des temps de réponse des analyses, nous considérons que la charge contient uniquement des requêtes décisionnelles (alors que les charges contiennent parfois des requêtes d'alimentation par exemple, comme c'est le cas dans les bancs d'essais [1]). Nous ne traitons pas ici de l'évolution de la configuration d'optimisation. Ainsi, nous nous focalisons ici uniquement sur l'évolution des requêtes de la charge, qui constitue un problème en soi.

6.2.2 Évolution de requêtes

Le problème de l'évolution des requêtes a été traité, dans les entrepôts de données, de façon indirecte. Nous avons évoqué en introduction le problème de la maintenance des vues matérialisées. Il faut considérer la dualité des vues qui sont à la fois des ensembles d'enregistrements si l'on considère la définition en extension et des requêtes d'après leur définition en intention. En effet, par définition, une vue est le résultat d'une requête. Ainsi le problème de l'évolution des vues peut être traité en se focalisant sur le problème de l'évolution des requêtes. Cette alternative est suivie dans [Bel02]. Dans ce travail, le problème de la maintenance des vues est décomposé en deux sous-problèmes : (1) propager l'évolution de schéma sur le schéma de la vue, (2) adapter l'extension de la vue. Pour ce faire, l'impact des changements sur le schéma est examiné sur les clauses SELECT, WHERE et FROM de la requête représentant la vue. L'idée est de formuler l'expression de la nouvelle vue en termes de sous-requêtes qui peuvent être subsumées par des vues matérialisées existantes

[1] http ://www.tpc.org/tpcd/default.asp

afin d'éviter un recalcul complet du contenu de la vue.

Ainsi, dans le domaine des entrepôts de données, la problématique de l'évolution de requêtes n'a pas encore été traitée comme un problème à part entière. Or dans ce domaine, cet enjeu apparaît comme crucial puisque les requêtes ne servent pas uniquement à la définition de vues. En effet, elles constituent un élément important du processus décisionnel. D'une part, elles permettent d'assurer la phase d'analyse qui est la raison même de l'existence des entrepôts de données. D'autre part, elles permettent de tester la performance du système décisionnel pour la définition de stratégies d'optimisation (dans le cas des charges de requêtes par exemple). Or cette performance est également un enjeu des systèmes décisionnels pour assurer la rapidité des analyses et leur donner le caractère «en ligne» qu'elles doivent satisfaire.

Le problème de la maintenance de requêtes a été évoqué dans le domaine des bases de données, en particulier dans la modélisation de celles-ci. Certains auteurs ont argumenté que la maintenance des requêtes est cruciale dans la mesure où celles-ci peuvent être impliquées au niveau des applications exploitant la base de données [VPVS07]. Or les techniques classiques de modélisation de base de données considèrent qu'une base de données est un tout. Elles omettent le fait qu'elles sont liées à une large variété d'applications et qu'elles sont liées à des outils tels que des rapports, des formulaires, etc. Dans ce cas, un petit changement tel que la suppression d'un attribut dans la base de données en question peut engendrer d'importants dysfonctionnements (des applications, des formulaires, etc.).

Ainsi, dans [PKVV05], les auteurs introduisent un modèle basé sur une représentation de graphe qui permet de décrire à la fois les relations, les vues, les contraintes et les requêtes. Par la suite, dans [PVV06], ces auteurs étendent leur précédent travail en proposant de formuler un ensemble de règles qui permettent l'identification de l'impact d'un changement sur les relations, les attributs et les contraintes. Ils proposent une méthode semi-automatique pour répondre à ces changements. L'impact inclut la base elle-même, ses requêtes, les procédures stockées, les triggers, etc. Cet impact se traduit par une annotation sur le graphe, ce qui nécessite un important travail de la part de l'administrateur en amont.

Dans l'approche qui vient d'être présentée, la gestion de l'évolution passe par une annotation sur le schéma de la base lui-même, demandant un important travail préalable. Dans notre approche, nous proposons un traitement généralisé des changements, applicable sur n'importe quel schéma. Ceci est rendu possible grâce à la particularité des schémas multidimensionnels adoptés pour les entrepôts de données, qui contiennent des concepts porteurs de sémantique (dimension, hiérarchie de dimension, etc.), ayant des rôles particuliers et qui se retrouvent dans toute modé-

lisation d'entrepôt de données.

6.3 Évolution de schéma : conséquences sur la charge

Afin de propager les évolutions de schéma sur la charge, nous proposons une typologie des changements opérés sur le schéma et des impacts que ces changements peuvent avoir au niveau de la charge. Notre typologie est présentée dans le tableau de la figure 6.1.

Nous déterminons quel «élément» est modifié, quel est le type de la modification et sa conséquence sur la charge. Concernant la charge, trois conséquences sont possibles : (1) modification de requête(s), (2) suppression de requête(s), (3) création de requête(s).

Rôle dans le schéma	Type	Opération	Requête à modifier	Requête à supprimer	Requête à créer
Dimension et Niveau	Table	Création			OUI
Dimension et Niveau	Table	Suppression	OUI	OUI	
Dimension et Niveau	Table	Mise à jour (Renommage)	OUI		
Fait	Table	Modification (Renommage)	OUI		
Mesure	Attribut	Ajout			OUI
Mesure	Attribut	Suppression	OUI	OUI	
Mesure	Attribut	Modification (Renommage)	OUI		
Descripteur Dimension	Attribut	Ajout			OUI
Descripteur Dimension	Attribut	Suppression	OUI	OUI	
Descripteur Dimension	Attribut	Modification (Renommage)	OUI		

FIG. 6.1 – Évolutions de schéma et conséquences sur la charge

Pour une évolution donnée, le tableau indique quelle conséquence elle a sur la charge (marqué par un OUI) : modification, suppression ou création de requête. Dans le cas où il y a deux OUI pour une évolution donnée (en l'occurrence modification et suppression), il s'agit d'essayer de mettre à jour les requêtes en fonction de l'évolution de schéma et de les supprimer si la modification n'est pas possible. Concernant la modification des requêtes, cela consiste à mettre à jour la syntaxe sur des requêtes concernées.

6.4 Évolution de charge

Dans cette section, nous détaillons les différents aspects de notre proposition. Tout d'abord, nous fournissons le cadre général de notre approche. Ensuite, nous présentons l'algorithme pour l'évolution de la charge.

6.4.1 Cadre général

Notre proposition d'évolution de charge se place dans un cadre général que nous exposons dans la figure 6.2.

À partir de l'utilisation de l'entrepôt de données, une charge initiale contenant les requêtes des utilisateurs est définie. Une sélection de la stratégie d'optimisation peut être réalisée sur la base de cette charge (choix d'index et de vues à matérialiser par exemple). Lorsque l'administrateur fait évoluer le schéma de l'entrepôt de données, les requêtes peuvent devenir incohérentes, i.e. elles peuvent ne plus être applicables sur le schéma courant de l'entrepôt de données.

En prenant en compte à la fois la charge initiale et les changements subis par le schéma de l'entrepôt de données, le processus d'évolution de la charge est appliqué. Cela permet de disposer d'une charge mise à jour qui peut être appliquée sur l'entrepôt mis à jour. Cette charge contient à la fois les requêtes de la charge initiale qui ont été mises à jour pour rester cohérentes par rapport au schéma courant de l'entrepôt, mais également des nouvelles requêtes dans le cas où les changements opérés ont généré de nouvelles possibilités d'analyse dans l'entrepôt de données.

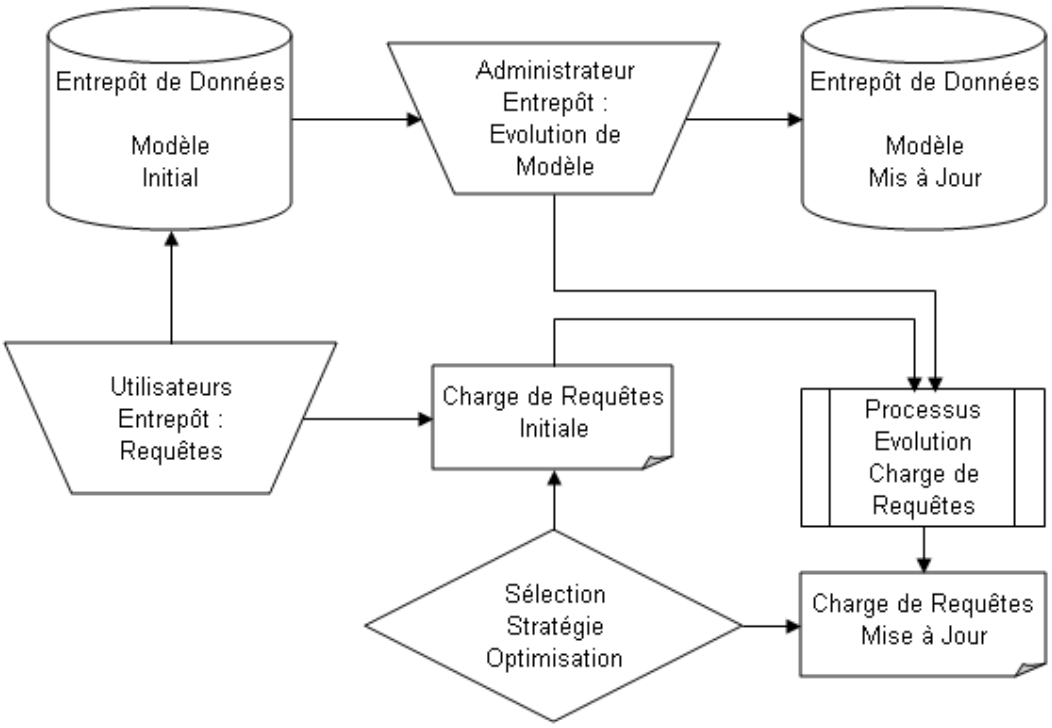


FIG. 6.2 – Processus pour l'évolution de la charge

6.4.2 Mise en œuvre

Afin de mettre en œuvre l'évolution de la charge, nous proposons un algorithme. Cette mise en œuvre s'effectue dans un contexte relationnel.

Classiquement, dans un environnement d'analyse en ligne, les requêtes nécessitent le calcul d'agrégats selon les différentes dimensions. En effet, à partir d'un schéma donné, le processus d'analyse consiste à résumer les données en utilisant (1) des opérateurs d'agrégation appliqués sur la mesure, tel que l'opérateur SUM ou AVG ; (2) des clauses GROUP BY permettant le regroupement.

Dans le cas où nous générons de nouvelles requêtes, nous utiliserons l'opérateur AVG sur une mesure de la table des faits. Ce choix est motivé par le fait que l'opérateur de la moyenne est plus généralement utilisable. Par exemple, il est possible de réaliser une moyenne de températures, de notes alors qu'il n'est pas possible de réaliser des additions (opérateur SUM) sur ces mesures.

Pour réaliser l'évolution de la charge, nous proposons un algorithme qui prend en paramètres d'entrée les évolutions de schéma de l'entrepôt et la charge initiale (algorithme 8). L'algorithme est fondé sur la typologie présentée précédemment.

Le principe général est le suivant : pour chaque évolution de schéma, on propage son impact sur les requêtes concernées.

- Concernant le processus de réécriture après le renommage d'un élément, il s'agit d'analyser chaque clause de chaque requête dans la charge (SELECT, FROM, WHERE, GROUP BY) afin de détecter les anciens noms et de les remplacer par les nouveaux.
- En cas d'opération de suppression dans le schéma (dimension ou descripteur de dimension ou mesure), si le processus de mise à jour échoue pour une requête, la requête correspondante est détruite. Par exemple, si une dimension est supprimée du schéma et que la requête existante correspond à une analyse selon cette dimension uniquement et aucune autre, il faut la détruire.
- Concernant la création d'un élément (dimension ou descripteur de dimension), nous définissons une requête décisionnelle prenant en compte ce nouvel attribut ou ce nouveau niveau.

Algorithme 8 Processus d'évolution de la charge

Entrée: charge W, ensemble des évolutions de schéma E
Sortie: charge mise à jour W'

Copie des requêtes de W dans W'
pour tout évolution $e \in E$ **faire**
  si e="renommage table de dimension, niveau " OU "renommage table des faits" OU "renommage mesure" OU "renommage descripteur de dimension" **alors**
    Réécrire les requêtes décisionnelles concernées en fonction des nouveaux noms
  fin si
  si e="création niveau de granularité, dimension" OU "création descripteur de dimension" **alors**
    Chercher le lien relationnel entre la table des faits et la table concernée
    Écrire une requête décisionnelle en fonction de ce nouveau niveau : (attribut créé ou clé de la table dans la clause SELECT et GROUP BY)
    Ajout de la requête dans la charge W'
  fin si
  si e="création d'une mesure" **alors**
    Écrire une requête décisionnelle utilisant cette nouvelle mesure
  fin si
  si e="suppression d'un niveau de granularité ou d'une dimension" **alors**
    Chercher un autre niveau dans la même dimension ou une autre dimension
    Réécrire les requêtes décisionnelles concernées en fonction du niveau ou de la dimension de remplacement, détruire les requêtes pour lesquelles la réécriture est impossible
  fin si
  si e="suppression d'une mesure" **alors**
    Chercher une autre mesure dans la même table des faits
      {Nous supposons qu'il y a une autre mesure ; dans le cas contraire, il s'agit de la suppression de la table des faits et pas seulement de la mesure}
    Réécrire les requêtes décisionnelles concernées en fonction de la nouvelle mesure
  fin si
  si e="suppression d'un descripteur de dimension" **alors**
    Chercher un autre descripteur de dimension dans le même niveau (clé de la table sinon)
      {Réécrire les requêtes décisionnelles concernéesen fonction du nouvel attribut}
  fin si
fin pour
return charge W'

6.5 Exemple

Pour illustrer notre approche, nous nous basons sur l'exemple de LCL. Le schéma initial que nous considérons est représenté dans la figure 6.3. Le NBI est observé par rapport aux dimensions suivantes : `CUSTOMER` (client), `AGENCY` (agence) and `YEAR` (année). La dimension `AGENCY` présente une hiérarchie de dimension avec le niveau `COMMERCIAL_DIRECTION` (direction commerciale), qui correspond à un regroupement géographique d'agences.

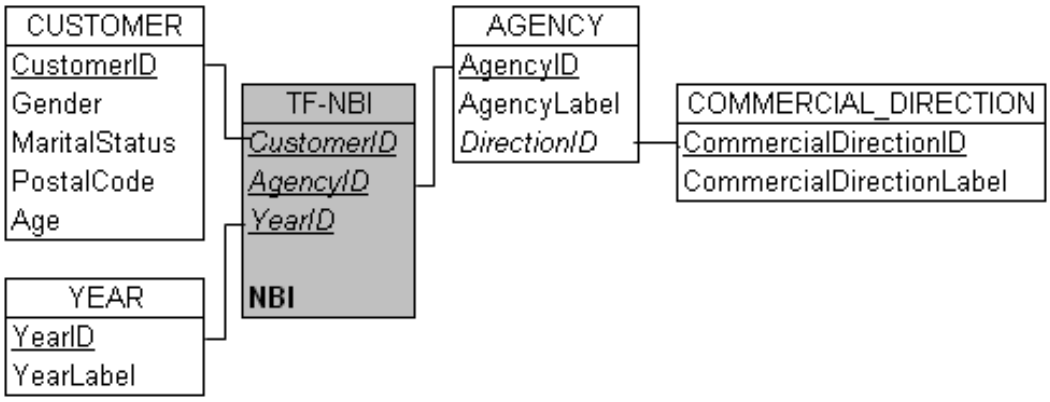


FIG. 6.3 – Schéma initial de l'entrepôt pour l'analyse du NBI

Dans un but pédagogique, nous considérons ici une charge comprenant un très faible nombre de requêtes, soit cinq requêtes qui permettent d'analyser la somme et la moyenne du NBI en fonction des différentes dimensions, à différents niveaux de détail (figure 6.4).

Q1: **SELECT** AgencyLabel, YearLabel, AVG(NBI) **FROM** AGENCY, YEAR, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND YEAR.YearID=TF-NBI.YearID **GROUP BY** AgencyLabel, YearLabel ;

Q2: **SELECT** CommercialDirectionLabel, YearLabel, AVG(NBI) **FROM** AGENCY, COMMERCIAL_DIRECTION, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND COMMERCIAL_DIRECTION.CommercialDirectionID= AGENCY.DirectionID **GROUP BY** CommercialDirectionLabel, YearLabel ;

Q3: **SELECT** MaritalStatus, YearLabel, SUM(NBI) **FROM** CUSTOMER, TF-NBI WHERE CUSTOMER.CustomerID=TF-NBI.CustomerID **GROUP BY** MaritalStatus, YearLabel ;

Q4: **SELECT** CommercialDirectionLabel, YearLabel, AVG(NBI) **FROM** AGENCY, COMMERCIAL_DIRECTION, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND COMMERCIAL_DIRECTION.CommercialDirectionID=AGENCY.DirectionID AND YearLabel='2000' **GROUP BY** CommercialDirectionLabel, YearLabel ;

Q5: **SELECT** AgencyLabel, SUM(NBI) **FROM** AGENCY, COMMERCIAL_DIRECTION, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND COMMERCIAL_DIRECTION.CommercialDirectionID=AGENCY.DirectionID AND YearLabel='2000' AND CommercialDirectionLabel='Lyon' **GROUP BY** AgencyLabel ;

FIG. 6.4 – Charge initiale pour l'analyse du NBI

En raison des évolutions des sources de données et des besoins d'analyse, les changements suivants sont opérés sur le schéma de l'entrepôt :

- ajout de l'attribut `Segment` dans la dimension `CUSTOMER` ;
- renommage du niveau `COMMERCIAL_DIRECTION` en `DIRECTION` ;
- renommage des attributs `CommercialDirectionID` et `CommercialDirectionLabel` respectivement en `DirectionID` et `DirectionLabel` ;
- insertion du niveau `COMMERCIAL_UNIT` entre les niveaux `AGENCY` et `DIRECTION`.

Les mises à jour une fois effectuées, nous disposons du schéma de la figure 6.5.

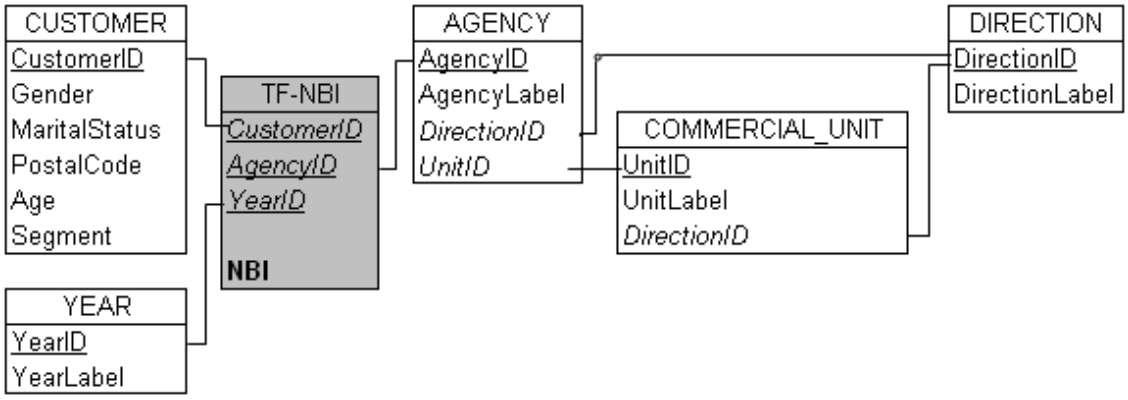


FIG. 6.5 – Schéma mis à jour de l'entrepôt pour l'analyse du NBI

Une fois l'évolution effectuée, notre algorithme est appliqué et la mise à jour de la charge est réalisée, fournissant une charge mise à jour (figure 6.6).

Ainsi, les requêtes Q1 et Q3 n'ont pas subi de changement. Les requêtes Q2, Q4 et Q5 ont été mises à jour pour prendre en compte les renommages de niveau (`COMMERCIAL_DIRECTION` en `DIRECTION`) et d'attributs (`CommercialDirectionID` et `CommercialDirectionLabel` en `DirectionID` et `DirectionLabel`). La requête Q6 a été ajoutée pour prendre en compte le niveau créé `COMMERCIAL_UNIT`.

Ainsi la syntaxe des requêtes a été mise à jour afin de rester cohérente avec le schéma de l'entrepôt. De plus, une requête a été ajoutée pour prendre en compte un besoin d'analyse potentiel. Les évolutions du schéma ont ainsi été propagées au niveau de la charge.

Q1: **SELECT** AgencyLabel, YearLabel, AVG(NBI) **FROM** AGENCY, YEAR, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND YEAR.YearID=TF-NBI.YearID **GROUP BY** AgencyLabel, YearLabel ;

Q2: **SELECT** DirectionLabel, YearLabel, AVG(NBI) **FROM** AGENCY, DIRECTION, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND DIRECTION.DirectionID=AGENCY.DirectionID **GROUP BY** DirectionLabel, YearLabel ;

Q3: **SELECT** MaritalStatus, YearLabel, SUM(NBI) **FROM** CUSTOMER, TF-NBI WHERE CUSTOMER.CustomerID=TF-NBI.CustomerID **GROUP BY** MaritalStatus, YearLabel ;

Q4: **SELECT** DirectionLabel, YearLabel, SUM(NBI) **FROM** AGENCY, DIRECTION, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND DIRECTION.DirectionID=AGENCY.DirectionID AND YearLabel='2000' **GROUP BY** DirectionLabel, YearLabel ;

Q5: **SELECT** AgencyLabel, AVG(NBI) **FROM** AGENCY, DIRECTION, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND DIRECTION.DirectionID=AGENCY.DirectionID AND YearLabel='2000' AND DirectionLabel='Lyon' **GROUP BY** AgencyLabel ;

Q6: **SELECT** UnitLabel, AVG(NBI) **FROM** AGENCY, COMMERCIAL_UNIT, TF-NBI WHERE AGENCY.AgencyID=TF-NBI.AgencyID AND COMMERCIAL_UNIT.UnitID=AGENCY.UnitID **GROUP BY** UnitLabel ;

FIG. 6.6 – Charge mise à jour pour l'analyse du NBI

6.6 Discussion

Dans cette section, nous souhaitons apporter une discussion sur notre approche d'évolution de charge.

Tout d'abord, nous avons fait des choix en terme des évolutions de schéma que nous avons traitées. En effet, nous avons estimé qu'en cas de changement trop important du schéma, d'une part il était difficile de maintenir la charge, d'autre part il n'était pas possible d'envisager de façon automatique la définition de nouvelles requêtes. Ainsi, nous n'avons pris en compte ni la création, ni la suppression d'une table de faits. Prenons par exemple le cas de la suppression d'une table de faits. Si c'était l'unique table des faits, l'entrepôt n'a plus aucun sens. Dans le cas contraire, toutes les requêtes se rapportant à cette table devraient néanmoins être détruites. Devant ce changement considérable, nous supposons que la charge initiale est a priori complètement caduque. Considérons à présent le cas où le schéma est enrichi d'une nouvelle table de faits avec diverses dimensions. Comment savoir quelles seront les dimensions pertinentes pour les utilisateurs ? Par ailleurs, nous n'avons pas considéré des évolutions telles que la modification du domaine de définition. En effet, nous avons considéré uniquement les évolutions de schéma qui avaient un impact sur la syntaxe des requêtes.

Un autre aspect que nous souhaitons discuter ici est le côté automatique de notre approche. Ceci peut être pertinent, par exemple dans un contexte d'auto-administration. Mais nous reconnaissons que l'automatisation complète nécessite des choix arbitraires qui pourraient être relaxés en ayant recours à un processus semi-automatique. C'est par exemple le cas par rapport aux choix concernant la création de nouvelles requêtes pour traduire des besoins d'analyse potentiel : choix de l'opérateur appliqué sur la mesure, choix de l'attribut pour le regroupement, etc.

Notons que lorsqu'une évolution de schéma génère une nouvelle possibilité d'analyse, nous créons une requête qui est censée représenter cette nouvelle possibilité. Nous devons évoquer le cas où nous créons une requête traduisant une demande d'analyse potentielle de l'utilisateur, alors que a posteriori, on pourrait dire que cela ne correspond pas à une requête d'utilisateur pertinente. Notons que la création d'une requête n'est possible que lorsqu'il y a une évolution de schéma génératrice d'une nouvelle possibilité d'analyse, en l'occurrence dans le cas d'un ajout de descripteur de dimension, de niveau de granularité de façon plus générale ou d'ajout de mesure. Or, notons, que l'évolution du schéma peut être engendrée soit par une évolution des besoins d'analyse eux-mêmes, soit par une évolution des sources de données. Dans le cas où l'évolution est faite suite à une évolution des besoins d'ana-

lyse, comme c'est le cas avec notre approche de personnalisation, il est pertinent de créer une requête qui traduise cette nouvelle possibilité d'analyse, puisqu'elle correspond à un besoin réel d'un utilisateur. Si, par contre, l'évolution est due à une évolution des sources et qu'il s'agit d'une évolution génératrice de nouvelles possibilités d'analyse, cela sous-entend que l'administrateur a jugé utile de répercuter cette évolution au niveau du schéma de l'entrepôt. Autrement dit, il a jugé qu'elle avait un intérêt pour l'analyse des utilisateurs. Ainsi, il est pertinent d'ajouter une requête à la charge pour traduire cette nouvelle possibilité d'analyse.

Pour une évolution de schéma qui correspond à une nouvelle possibilité d'analyse, nous créons une requête nouvelle. Ceci peut poser problème selon l'algorithme utilisé par la suite pour définir la configuration d'optimisation. En effet, s'il y a un prétraitement de la charge, avec un choix par rapport à la fréquence d'apparition des requêtes, cela peut engendrer l'inefficacité de notre approche. En effet, en créant une unique requête, cette dernière pourrait passer inaperçue dans la stratégie de sélection. Cela dépend entre autres de la taille initiale de la charge.

Nous n'avons pas apporté ici d'éléments de réponse numériques quant à l'intérêt de notre approche. En effet, on peut imaginer que celle-ci n'est pertinente que lorsqu'un grand nombre de requêtes composent la charge que l'on considère pour la sélection de la configuration d'optimisation. On peut penser que si le nombre de requêtes est faible, un changement manuel est peut-être davantage indiqué. Néanmoins, nous avons basé notre approche sur l'hypothèse selon laquelle les charges de requêtes réelles sont volumineuses [GS03].

Deux autres paramètres pouvant influencer l'intérêt de notre approche sont l'importance et la fréquence des évolutions opérées sur le schéma de l'entrepôt. Ces paramètres, ainsi que leur impact ne sont pas faciles à évaluer. Le seul élément de réponse que nous pouvons donner à ce jour est de l'ordre de notre propre expérience. Depuis notre arrivée chez LCL, il y a eu trois modifications complètes de la structure commerciale, ce qui signifie des modifications de l'ensemble de la hiérarchie de dimension concernant la structure commerciale, avec à la fois des renommages de niveaux et des ajouts de niveaux. Mais compte tenu du nombre d'utilisateurs et de la nécessité d'obtenir des résultats rapidement, la pro-activité de notre approche devient un élément prépondérant. En effet, il n'est pas acceptable que lors du changement de structure commerciale, tous les utilisateurs soient pénalisés dans l'obtention des analyses qu'ils avaient l'habitude d'avoir en un temps donné et que ce temps augmente sans raison du point de vue des utilisateurs.

6.7 Conclusion

Dans ce chapitre, nous avons abordé le problème de l'évaluation de modèle d'entrepôt de données évolutif en s'intéressant à celui de l'évolution de requêtes. À notre connaissance, ce problème n'a jamais été traité dans le domaine des entrepôts de données, si ce n'est de manière indirecte à travers la maintenance des vues. Plus particulièrement, nous nous sommes focalisés sur le problème de l'évolution de charge. Pour ce faire, nous avons déterminé dans un premier temps une typologie des changements possibles du schéma de l'entrepôt et de l'impact que ces changements ont sur une charge. Nous avons proposé un cadre global pour cette évolution de charge, ainsi qu'un algorithme pour réaliser l'évolution elle-même. Nous avons illustré notre approche par un exemple. Nous évoquerons l'implémentation que nous avons réalisée dans le chapitre 7.

Le principal avantage de notre approche est de répercuter l'évolution de schéma directement sur la charge, évitant ainsi d'attendre une nouvelle charge pour mettre en œuvre une stratégie d'optimisation. Ceci permet l'évaluation de modèle d'entrepôt de données évolutif. De plus, l'administrateur est aidé dans une démarche pro-active qui assure la cohérence syntaxique de requêtes par rapport à l'évolution subie par l'entrepôt de données et qui propose dans certains cas de nouvelles requêtes qui traduisent les besoins d'analyse susceptibles d'apparaître. Ainsi nous proposons un support pour l'évolution des stratégies d'optimisation. Cette approche permet également une auto-administration des performances dans un contexte évolutif.

Troisième partie

Développement : validation de nos contributions et réalisation industrielle

La troisième et dernière partie de ce mémoire est consacrée au travail de développement. Ce travail s'inscrit dans deux contextes différents : industriel d'une part et scientifique d'autre part.

Tout d'abord, nous avons été amenés à réaliser pour le compte de LCL une plateforme logicielle permettant la gestion du processus des demandes de marketing local. Ainsi, dans le chapitre 8, nous donnons les différents éléments concernant la conception et le développement de cette plateforme que nous avons baptisée MARKLOC.

À partir de ce travail d'ingénierie, nous avons conçu un entrepôt de données test nommé LCL-DW sur lequel se sont posés les problèmes scientifiques que nous avons traités dans ce mémoire en apportant des contributions. Pour valider ces dernières, nous avons donc développé la plateforme WEDriK que nous présentons dans le chapitre 7.

La plateforme WEDriK

Résumé

Ce chapitre a pour but de présenter les développements informatiques que nous avons réalisés dans le but de valider nos contributions en terme de personnalisation des analyses. Ainsi, nous avons implémenté un applicatif, baptisé WEDriK pour data Warehouse Evolution Driven by Knowledge. Dans ce chapitre, nous nous attachons à décrire cet applicatif, comment il fonctionne, quelles sont ses fonctionnalités, etc.

Sommaire

Chapitre 7

La plateforme WEDriK

7.1 Introduction

Pour valider notre approche de personnalisation des analyses, nous avons développé un prototype nommé WEDriK[1] (data Warehouse Evolution Driven by Knowledge) selon une configuration client/serveur. Comme nous l'avons évoqué dans [FBB06e], ce prototype a pour but de mettre en œuvre nos contributions sur la personnalisation.

Le prototype WEDriK est implémenté dans un environnement relationnel. Ainsi, il correspond à une couche applicative qui permet la personnalisation des analyses dans un entrepôt de données stocké dans le SGBD relationnel Oracle 10g[2]. Pour mettre en œuvre la personnalisation, l'utilisateur interagit avec le système via une interface web. Des scripts PHP[3] sont utilisés. L'interaction entre l'interface et le SGBD se fait grâce à des fonctions OCI (Oracle Call Interface) qui sont amenées, parfois, à lancer des scripts PL/SQL.

L'objectif de WEDriK est de permettre l'évolution de schéma de l'entrepôt de données selon les besoins exprimés par l'utilisateur et ce, quel que soit l'entrepôt de données considéré. Pour ce faire, nous avons implémenté le méta-modèle d'entrepôt qui permet de décrire différents schémas d'entrepôt dans une base de données dans le SGBDR Oracle 10g. Nous présentons ici les fonctionnalités de la plateforme WEDriK elle-même. Par ailleurs, nous avons intégré dans WEDriK un module assurant l'évolution incrémentale de la charge.

[1] http ://eric.univ-lyon2.fr/~cfavre/wedrik
[2] http ://www.oracle.com
[3] http ://www.phpfrance.com

7.2 Fonctionnalités

WEDriK offre différentes fonctionnalités qui permettent la réalisation de notre approche de personnalisation selon son architecture globale. La figure 7.1 montre les fonctionnalités de la plateforme WEDriK pour l'utilisateur : visualisation du schéma courant de l'entrepôt, expression des connaissances sous forme de règles, construction des requêtes d'analyse.

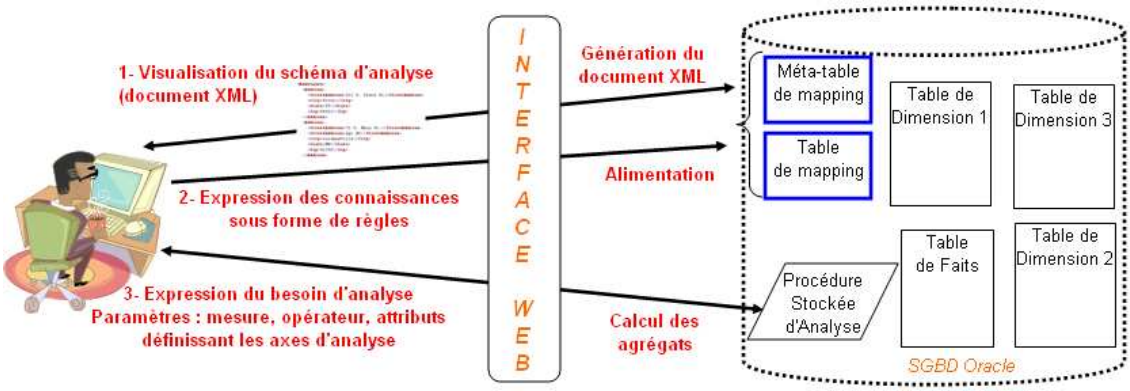


FIG. 7.1 – Fonctionnement de WEDriK

7.2.1 Schéma de l'entrepôt de données

Avant d'envisager d'étendre les possibilités d'analyse de l'entrepôt, il faut que l'utilisateur soit en mesure de connaître les possibilités d'analyse actuelles. En effet, étant donné que les utilisateurs font évoluer l'entrepôt de façon incrémentale, il est nécessaire de pouvoir visualiser le schéma que l'on peut qualifier de courant (photographie du schéma au moment où on l'observe).

Pour ce faire, un document XML (eXtensible Markup Language : langage de balisage extensible) est généré à partir de l'interrogation du méta-modèle. L'avantage de XML est de permettre aux utilisateurs de naviguer à travers les hiérarchies du modèle et de décrire correctement les possibilités d'analyse. Ainsi, ce document va permettre aux utilisateurs de les aider dans leur choix d'analyse. Cette fonctionnalité exploite donc le standard XML, évitant le recours à un outil de visualisation utilisant un format propriétaire. Ce document XML est généré grâce à l'interrogation de la base implémentant le méta-modèle. En effet, grâce à des requêtes sur cette base, il est possible de déterminer les faits, les dimensions, les hiérarchies, et l'ensemble des attributs qui les décrivent.

7.2.2 Saisie des règles d'agrégation

Notre objectif pour permettre la saisie des règles était de faire en sorte que la démarche soit conviviale et de rendre transparents les concepts théoriques sous-jacents de notre méthode.

La saisie des règles est rendue facile pour les utilisateurs parce qu'elle est guidée. En effet, l'exploitation du méta-modèle permet de fonctionner avec des listes de valeurs prédéfinies comme nous l'indiquons dans ce qui suit.

Les différentes étapes de la saisie sont les suivantes :

1) Choix du niveau à partir duquel est créé le nouveau niveau : ce niveau est sélectionné dans une liste ;
2) Choix des attributs qui seront utilisés dans les conditions des règles : les attributs du niveau choisi figurent dans une liste ;
3) Saisie des conditions (figure 7.3) : les attributs sélectionnés apparaissent, l'utilisateur choisit l'opérateur qui est utilisé dans la règle (inférieur, supérieur, égal, dans la liste, etc.) et saisit la ou les valeurs par rapport à cet opérateur, il demande la saisie d'une nouvelle condition pour passer à une autre règle ;
4) Définition de la structure du nouveau niveau : l'utilisateur donne un nom au nouveau niveau et précise le nombre d'attributs qu'il contiendra (la clé est gérée automatiquement, elle n'est pas explicitée par l'utilisateur) ;
5) Saisie des attributs du nouveau niveau (figure 7.2) : l'utilisateur saisit un nom pour chacun des attributs et lui affecte un type de données (chaîne de caractères, entier, etc.) ;
6) Association des conditions aux attributs du nouveau niveau : l'utilisateur considère les conditions des différentes règles et, pour chacune, détermine la valeur des attributs du nouveau niveau.

Ainsi les étapes 1 et 2, permettent de définir la clause «si» de la méta-règle, les étapes 4 et 5 définissent la clause «alors» de cette dernière. L'étape 3 permet d'instancier la clause «si» de la méta-règle en définissant la clause «si» des différentes règles d'agrégation. L'étape 6 permet d'instancier la clause «alors» de la méta-règle en définissant la clause «alors» des différentes règles d'agrégation.

FIG. 7.2 – Saisie de la méta-règle d'agrégation

FIG. 7.3 – Saisie des règles d'agrégation

Après validation, la table de mapping est générée automatiquement. La vérification des contraintes liées au concept de partition a lieu selon l'algorithme que nous

avons présenté dans la section 5.3. Dans le cas où les règles ne sont pas valides, elles doivent être ressaisies par l'utilisateur.

7.2.3 Analyse

Concernant l'analyse, compte tenu du mode de stockage relationnel utilisé, cette fonctionnalité permet la création conviviale de requêtes SQL. Il s'agit en l'occurrence de choisir sur l'interface la mesure, l'opérateur à appliquer, ainsi que les attributs représentant les axes d'analyse.

La création de la requête est facilitée dans la mesure où, encore une fois, l'exploitation du méta-modèle permet de guider le choix de l'utilisateur sur l'interface (figure 7.4).

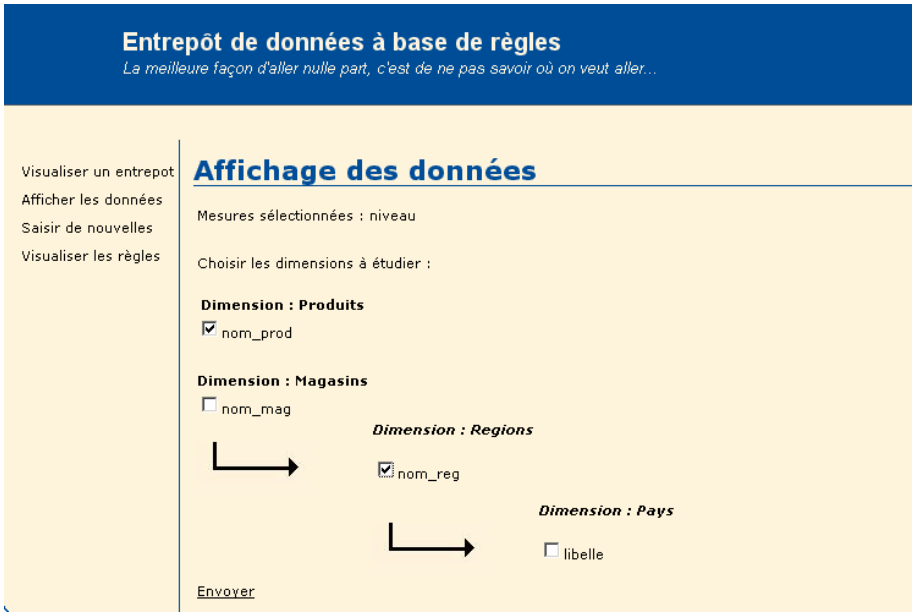


FIG. 7.4 – Choix des paramètres de l'analyse

7.3 Module d'évolution de charge

Pour mettre en œuvre le module dédié à l'évolution de la charge, nous sommes restés dans une configuration client/serveur. Le processus d'évolution se fait au moyen de scripts PHP lancés à partir d'un client. Le fonctionnement général de ce module est décrit dans la figure 7.5.

La charge est stockée, comme l'entrepôt de données, dans le SGBD Oracle 10g.

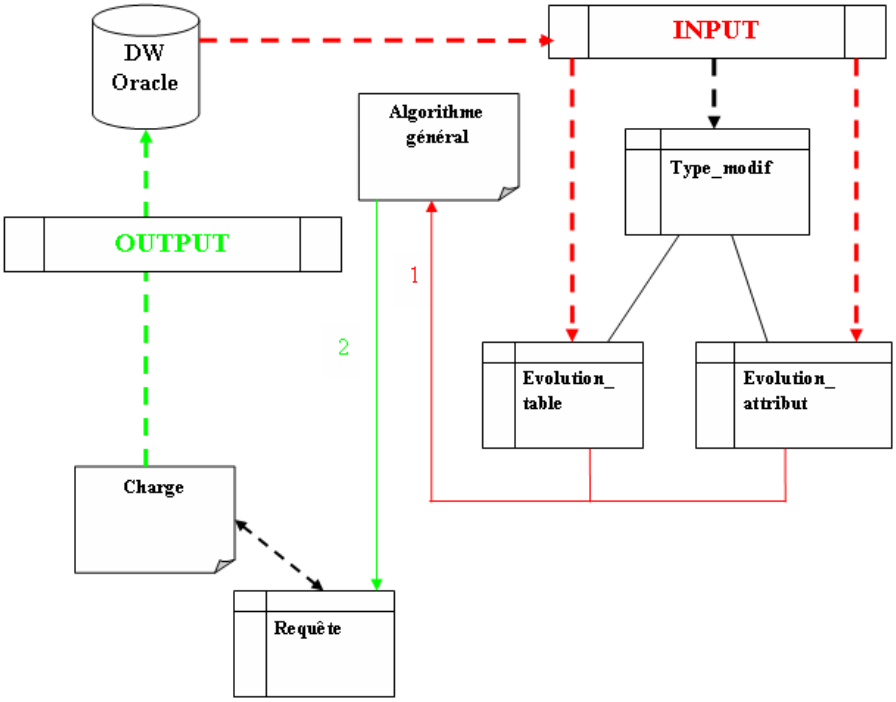


FIG. 7.5 – Fonctionnement du module d'évolution de charge

L'hypothèse de base que nous adoptons est qu'un outil nous fournit l'ensemble des modifications subies par le modèle, que ces modifications soient faites par l'administrateur de l'entrepôt ou qu'elles soient engendrées par WEDriK dans le cadre de la personnalisation des analyses. En effet, ce module a pour but de supporter l'évaluation de performances de modèle évolutif, sans se restreindre au seul cas de l'évolution induite par la personnalisation.

Dans le contexte relationnel, nous supposons que ces changements sont stockés dans deux tables relationnelles : une pour les changements effectués sur les tables, une pour les changements effectués sur les attributs. En effet, en entrée du processus, nous disposons des tables «evolution_table» et «evolution_attribut». Ces tables présentent différentes informations utiles telles que le type de changement opéré, le nom de l'élément qui a subi le changement, etc. Nous émettons l'hypothèse qu'elles sont mises à jour dès lors qu'une modification sur l'entrepôt se produit. En sortie du processus, nous obtenons le fichier SQL «charge».

En effet, les requêtes de la charge sont préparées pour être transformées en une représentation relationnelle dans une table nommée «requete» et ce, afin de faciliter leur traitement. Cette table contient les attributs stockant l'identifiant de la requête, la clause SELECT, la clause FROM, la clause WHERE et la clause GROUP BY.

L'algorithme est ensuite appliqué et, une fois les modifications réalisées, le fichier «charge» est reconstruit à partir de cette table.

Lors de l'exécution de l'algorithme, les tables evolution_table et evolution_attribut sont parcourues, chaque ligne correspondant à une modification du schéma. Pour chaque enregistrement, la valeur de l'attribut nommé «id_type_modif» va permettre d'indiquer le type de la modification. Suivant la valeur de cet attribut, le traitement adéquat est réalisé selon l'algorithme présenté précédemment.

Notons que pour ce traitement, nous exploitons également le méta-modèle de l'entrepôt présenté dans la section précédente.

L'interrogation de celui -ci est nécessaire pour déterminer, par exemple, quelles sont les hiérarchies de dimension, les liens entre les tables, etc. afin d'écrire les nouvelles requêtes.

7.4 Étude de cas : le LCL

7.4.1 Construction de l'entrepôt de données test LCL-DW

Afin d'appliquer nos propositions, nous avons conçu et développé un entrepôt de données test que nous avons appelé LCL-DW. Cet entrepôt porte sur l'analyse du NBI qui, rappelons-le, correspond à ce que rapporte un client à l'établissement bancaire.

Pour construire cet entrepôt, nous avons d'une part recensé les sources de données exploitables et avons recueilli d'autre part les besoins d'analyse.

Nous avons retenu les sources de données suivantes :

1) une source sous un format propriétaire appelée *SIAM* contenant les informations marketing sur les clients dont leur NBI annuel
2) une base de données sous Access appelée *Structures* qui fournit la structure organisationnelle de la direction d'exploitation Rhône-Alpes Auvergne pour laquelle nous disposons de données.

Ainsi, le schéma de LCL-DW que nous avons conçu est présenté dans la figure 7.6. Le NBI est analysé en fonction des dimensions `AGENCY` (agence), `YEAR` (année) et `CUSTOMER` (client). Deux de ces dimensions sont hiérarchisées : `AGENCY` et `CUSTOMER`. La dimension `AGENCY` présente une hiérarchie avec les niveaux `COMMERCIAL_UNIT` (unité commerciale) et `DIRECTION` (direction). La dimension `CUSTOMER` présente deux hiérarchies : la première contient le niveau `POTENTIAL_SCORING` (score potentiel) qui correspond au potentiel du client, la seconde contient le niveau `REAL_SCORING` (score

réel) qui correspond en quelque sorte à la «valeur» réelle du client.

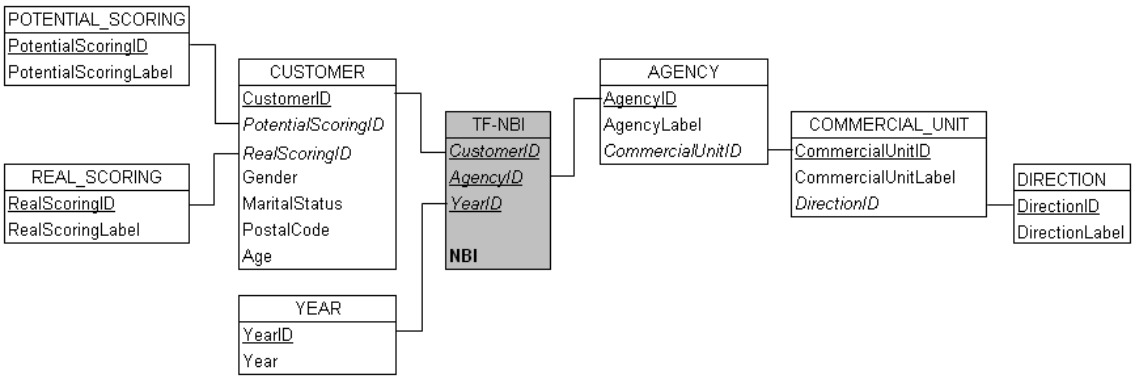


FIG. 7.6 – Schéma de LCL-DW

Nous avons créé la structure de LCL-DW sous Oracle 10g. Nous avons ensuite procédé à la phase ETL en développant nos propres scripts PHP.

Notons que concernant la source SIAM, les données sont uniquement accessibles via un requêteur. Ainsi, nous avons bâti une requête afin d'extraire les données qui nous intéressaient et les avons récupérées dans un fichier texte.

7.4.2 Application de la personnalisation

Nous nous basons sur l'entrepôt LCL-DW.

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 clients ne résident pas en France et le type «classique» pour les agences ne présentant pas de particularité. Ces informations n'étant pas 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 mettre à jour la hiérarchie de la dimension agence en ajoutant le niveau `AGENCY_TYPE` au-dessus du niveau `AGENCY`. Afin de réaliser une analyse du NBI en fonction du type d'agence, notre approche est utilisée.

Considérons les extraits de la table de dimension `AGENCY` et de la table de faits `TF-NBI` de la figure 7.7.

AGENCY		
AgencyID	*AgencyLabel*	...
01000	LYON REPUBLIQUE	...
01029	LYON GERLAND	...
01903	LYON III UNIVERSITE	...
01905	LYON LA DOUA	...
01929	AGENCE INTERNATIONALE	...
02256	CLERMONT LAFAYETTE (Etud)	...
02600	GRENOBLE	...
03730	ANNONAY	...

TF-NBI			
AgencyID	*CustomerID*	*YearID*	*NBI (€)*
01000	1	2006	2000
01903	2	2006	1000
01000	3	2006	4000
03730	4	2006	2000
01929	5	2006	5000
01000	6	2006	2000
01029	7	2006	1000
02600	8	2006	1000
01905	9	2006	2000
01929	10	2006	3000

FIG. 7.7 – Extraits des tables pour l'exemple du NBI

7.4.2.1 Phase d'acquisition.

Lors de la phase d'*acquisition*, l'utilisateur définit la méta-règle d'agrégation MR pour spécifier la structure du lien d'agrégation pour le type d'agence. Elle exprime donc le fait que le niveau `AGENCY_TYPE` va être caractérisé par l'attribut `AgencyTypeLabel` et il sera créé au-dessus du niveau `AGENCY` ; les regroupements des instances de `AGENCY` se baseront sur des conditions exprimées sur l'attribut `AgencyID`.

MR : *if* SelectionOn(`AGENCY`,{`AgencyID`})
then Generate(`AGENCY_TYPE`,{`AgencyTypeLabel`})

Puis l'utilisateur définit les règles d'agrégation qui instancient la méta-règle pour créer les différents types d'agence. Ainsi, il définit une règle pour chaque type d'agence, en exprimant à chaque fois la condition sur l'attribut `AgencyID` et en affectant à l'attribut généré `AgencyTypeLabel` sa valeur correspondante :

(R1) *if* `AgencyID` ∈ {'01903','01905','02256'} *then* `AgencyTypeLabel`='student'
(R2) *if* `AgencyID` = '01929' *then* `AgencyTypeLabel`='foreigner'
(R3) *if* `AgencyID` ∉ {'01903','01905','02256','01929'} *then* `AgencyTypeLabel`='classical'

En effet, les trois types d'agences sont «student», «foreigner» et «classical». Il exprime les différents types en fonction des identifiants des agences «AgencyID».

7.4.2.2 Phase d'intégration.

La phase d'*intégration* exploite la méta-règle et les règles d'agrégation R1, R2 et R3 pour générer la table de mapping MT_AGENCY_TYPE et les informations

concernant cette table sont insérées dans la méta-table $MAPPING\_META\_TABLE$ (figure 7.8). Ainsi la table de mapping contient les attributs `AgencyID`, `AgencyTypeLabel` respectivement, que l'on retrouve dans la méta-table $MAPPING\_META\_TABLE$ et dont les rôles sont respectivement «*conditioned*» et «*generated descriptor*». La clé `AgencyTypeID` est rajoutée automatiquement et figure donc dans la méta-table avec le rôle «*generated key*».

MT_AGENCY_TYPE		
AgencyID	***AgencyTypeLabel***	***AgencyTypeID***
IN ('01903', '01905', '02256')	student	1
= '01929'	foreigner	2
NOT IN ('01903', '01905', '02256', '01929')	classical	3

FIG. 7.8 – Table de mapping pour le niveau `AGENCY_TYPE`

MAPPING_META_TABLE				
Mapping_Table_ID	***Mapping_Table_Name***	***Attribute_Table***	***Attribute_Name***	***Attribute_Type***
1	MT_AGENCY_TYPE	AGENCY	AgencyID	conditioned
1	MT_AGENCY_TYPE	AGENCY_TYPE	AgencyTypeLabel	generated descriptor
1	MT_AGENCY_TYPE	AGENCY_TYPE	AgencyTypeID	generated key

FIG. 7.9 – Méta-table de mapping pour le niveau `AGENCY_TYPE`

7.4.2.3 Phase d'évolution.

L'évolution choisie par l'utilisateur correspond à un ajout. En effet, il n'y a pas de lien possible d'agrégation entre le type d'agence et l'unité commerciale qui correspond à un regroupement géographique des agences. La phase d'*évolution* qui suit permet donc d'une part de créer et d'alimenter la table `AGENCY_TYPE` ; et d'autre part de mettre à jour la table `AGENCY` pour la relier à la table `AGENCY_TYPE`, avec l'ajout de l'attribut `AgencyTypeID` et la mise à jour de ses valeurs (figure 7.10).

AGENCY			
AgencyID	*AgencyLabel*	...	*AgencyTypeID*
01000	LYON REPUBLIQUE	...	3
01029	LYON GERLAND	...	3
01903	LYON III UNIVERSITE	...	1
01905	LYON LA DOUA	...	1
01929	AGENCE INTERNATIONALE	...	2
02256	CLERMONT LAFAYETTE (Etud)	...	1
02600	GRENOBLE	...	3
03730	ANNONAY	...	3

AGENCY_TYPE	
AgencyTypeID	*AgencyTypeLabel*
1	student
2	foreigner
3	classical

FIG. 7.10 – La table `AGENCY_TYPE` créée et la table `AGENCY` mise à jour

7.4.2.4 Phase d'analyse.

Finalement, la phase d'*analyse* permet d'exploiter le schéma enrichi d'un nouvel axe d'analyse. Classiquement, dans un environnement d'analyse en ligne dans un contexte relationnel, les requêtes décisionnelles consistent en la création d'agrégats en fonction des niveaux de granularité dans les dimensions. En effet, étant donné un schéma, le processus d'analyse permet de résumer les données en exploitant (1) des opérateurs d'agrégation tels que SUM et (2) des clauses de regroupement telles que GROUP BY. Dans notre cas, l'utilisateur souhaitait une analyse sur la somme du NBI en fonction des types d'agence qu'il avait définis. La requête correspondante et le résultat de cette requête sont présentés dans la figure 7.11.

```
SELECT AgencyTypeLabel, SUM(NBI)
FROM TF-NBI, AGENCY, AGENCY_TYPE
WHERE TF-NBI.AgencyID=AGENCY.AgencyID
AND AGENCY.AgencyTypeID=AGENCY_TYPE.AgencyTypeID
GROUP BY AgencyTypeLabel ;
```

AgenyTypeLabel	NBI (€)
student	3000
foreigner	8000
classical	12000

FIG. 7.11 – Analyse du NBI en fonction du type d'agence

7.5 Conclusion

Le développement du prototype WEDriK a permis de mettre en œuvre notre approche de personnalisation des analyses dans les entrepôts de données. En effet, il permet aux utilisateurs de définir de nouveaux niveaux de granularité, avant de les exploiter pour obtenir des analyses personnalisées.

Grâce à la mise en place d'un méta-modèle définissant la structure de l'entrepôt de données évolutif, notre prototype permet non seulement de faciliter la saisie des utilisateurs, mais également de s'adapter à n'importe quel entrepôt. En d'autres termes, WEDriK est une plateforme générique indépendante de l'entrepôt de données qu'elle permet de faire évoluer.

Notons que notre objectif était de valider notre approche d'évolution de schéma par l'exploitation de règles définies par l'utilisateur. L'interface conviviale permet alors une saisie facile des règles, sans avoir connaissance de l'approche théorique qui est derrière (notion de méta-règle, de partie évolutive, etc.). Ceci permet une personnalisation dans la phase d'analyse en ligne. Par ailleurs, le module dédié à l'évolution de charge permet de tenir compte à la fois des évolutions du schéma induites par WEDriK et des évolutions réalisées par l'administrateur.

Développement industriel : la plateforme MARKLOC

Résumé

Ce chapitre a pour but de présenter la plateforme logicielle que nous avons réalisée au cours de cette thèse dans le cadre de la convention CIFRE. Cette plateforme, baptisée MARKLOC, permet de gérer l'ensemble du processus lié aux demandes de marketing local. Nous nous attachons à donner des éléments sur le logiciel, les ressources développées pour son fonctionnement, le bilan quant à la gestion de projet et à l'utilisation du logiciel. Nous montrons également comment nous avons mis en œuvre le concept de personnalisation dans cette plateforme.

Sommaire

Chapitre 8

Développement industriel : la plateforme MARKLOC

8.1 Introduction

Dans le cadre de cette thèse, nous avons été amenés à nous impliquer dans deux projets de développement qui avaient chacun des objectifs bien différents.

Le premier était de fournir aux collaborateurs de LCL de la direction Rhône-Alpes Auvergne une plateforme pour la gestion du processus des demandes de marketing local. Nous avons baptisé cette plateforme MARKLOC pour «MARKeting LOCal». Ainsi l'objectif était non seulement d'informatiser le processus de gestion des demandes marketing, mais d'aller au-delà en proposant des fonctionnalités pour améliorer ce processus, à la fois au niveau de l'émission des demandes, mais également dans leur suivi et dans leur analyse.

Dans le cadre de cette thèse, nous avons mené un projet de développement conséquent qui consistait à concevoir et implémenter une plateforme pour la gestion des demandes de marketing local. Nous le qualifions de conséquent pour deux raisons principales. Tout d'abord, cette plateforme devait être utilisée quotidiennement par un ensemble de collaborateurs de toute la direction Rhône-Alpes Auvergne, soit environ un millier de personnes. Ensuite, il faut noter que tout était à faire : du recueil des besoins, au développement informatique, en passant par la conception. Au-delà de ces aspects plutôt techniques, une conduite de projet était nécessaire avec une planification d'un échéancier, des réunions avec les personnes impliquées, etc.

Afin de mener ce projet à bien, nous avons préalablement réalisé une étude qui nous a permis de prendre connaissance de l'existant, des besoins par rapport à ce projet. Il est apparu que l'émission des demandes marketing n'était pas informatisée,

des fiches de liaison papier étaient utilisées. De ce fait, il n'y avait pas de stockage informatique exploitable des précédentes demandes émises. De plus, il y avait un fort besoin de normalisation de cette demande pour éviter la multiplication de versions différentes qui ne correspondent finalement pas aux informations requises pour ensuite traiter la demande.

Ensuite, un problème de suivi est apparu. Il se situe à deux niveaux. D'une part, le besoin d'un tableau de bord pour mesurer l'activité liée aux demandes marketing s'est fait ressentir. En effet, il est nécessaire d'avoir une vision des quantités de demandes qui sont faites, puisqu'elles se traduisent par la réalisation d'un ciblage de clients au niveau du Pôle Outils et Méthode de la direction d'exploitation. D'autre part, l'étude a révélé un manque au niveau de la mesure des résultats liés aux demandes marketing. En effet, lorsqu'une demande marketing est émise, elle a pour but de générer des ventes additionnelles. Il est intéressant de pouvoir quantifier ces ventes. Auparavant, cette mesure était effectuée un mois et demi après la fin de l'action, ce qui empêchait toute action corrective de la part des managers. En d'autres termes, les managers n'étaient pas en mesure de réagir durant l'action si les résultats n'étaient pas bons, par exemple dans le cas où l'action n'est pas bien relayée au niveau des conseillers commerciaux.

Enfin, il est apparu qu'un protocole devait être suivi avant l'émission de la demande de marketing. Il s'agit par exemple de vérifier s'il n'existe pas d'action marketing à venir au niveau national pour le produit en question, d'exploiter, s'ils existent, des critères pour définir les profils de clients ciblés lorsque le produit fait l'objet de ce qui est appelé kit marketing. Or le suivi de ce protocole n'était pas facilité avec l'ancien fonctionnement.

Ainsi, l'objectif était donc de concevoir une plateforme permettant de gérer l'ensemble du processus des demandes marketing : de leur émission, au suivi des résultats commerciaux faisant suite à ces demandes, en passant par leur validation et leur réalisation, tout en facilitant le respect du protocole.

Ce chapitre est organisé de la façon suivante. Tout d'abord, nous présentons le fonctionnement général de l'application, ainsi que les ressources qu'elle mobilise dans la section 8.2. Par la suite, nous évoquons dans la section 8.3 la gestion des habilitations des utilisateurs. Puis, dans la section 8.4, nous présentons la base des demandes de marketing. Vient ensuite la présentation des fonctionnalités offertes par la plateforme MARKLOC dans la section 8.5 en les illustrant par quelques interfaces du logiciel. Nous revenons finalement sur la prise en compte du concept de personnalisation, qui est également présent dans MARKLOC, dans la section 8.6. Enfin, nous concluons ce chapitre dans la section 8.7.

8.2 Fonctionnement général

8.2.1 Enchaînement du processus

L'application est accessible à partir d'un portail qui nécessite une connexion avec identifiant et mot de passe. Les applications pour lesquelles l'utilisateur a une habilitation apparaissent, parmi lesquelles MARKLOC peut figurer.

Normalement, avant la saisie d'une demande, le responsable commercial doit s'assurer qu'il n'y a pas d'action marketing nationale en cours ou à venir sur le produit sur lequel il veut faire une demande d'action. Si sa demande ne fait pas double emploi avec une action nationale, il doit vérifier si le produit en question ne fait pas l'objet d'un kit marketing. En effet, le service marketing national a mis en place des «kits marketing» qui correspondent à des préconisations de critères pour cibler les clients potentiellement intéressés par le produit. Ainsi, ces préconisations peuvent servir de base comme critères de ciblage dans la demande de marketing locale.

Afin de saisir la demande, différents renseignements sont demandés tels que le bénéficiaire de la demande, les critères de ciblage, la période durant laquelle l'action marketing aura lieu, etc.

Cette demande doit ensuite être validée par l'ensemble de la hiérarchie du responsable commercial. Ainsi, selon le niveau hiérarchique du responsable, cela peut nécessiter plusieurs validations successives. Les responsables hiérarchiques sont automatiquement prévenus par mail de la validation qu'ils doivent faire. Précisons que le résultat de la validation peut aussi être un refus.

Si la demande a été validée par toute la hiérarchie, elle est transmise au responsable du Pôle Outils et Méthodes qui valide (ou non) sa faisabilité et affecte la réalisation du ciblage à une de ses collaboratrices.

Durant l'action marketing, des résultats commerciaux sont consultables sur le(s) produit(s) concerné(s) par la demande.

La mise en œuvre de cette plateforme mobilise un certain nombre de ressources qui existaient préalablement dans le système d'informations de LCL, ou que nous avons dû également concevoir et développer.

8.2.2 Ressources exploitées

La plateforme MARKLOC s'appuie sur un ensemble de ressources. Pour davantage de clarté, nous avons représenté l'architecture globale de l'Intranet de la direc-

tion d'exploitation Rhône-Alpes Auvergne (RAA) nommé RAAnet avec l'ensemble des ressources mobilisées dans la figure 8.1.

Certaines ressources faisaient d'ores et déjà partie du système d'informations de LCL. Pour d'autres, nous les avons conçues et développées. Dans ce cas, elles prennent généralement la forme de bases de données relationnelles, avec pour certaines, une couche applicative qui permet l'alimentation par des personnes.

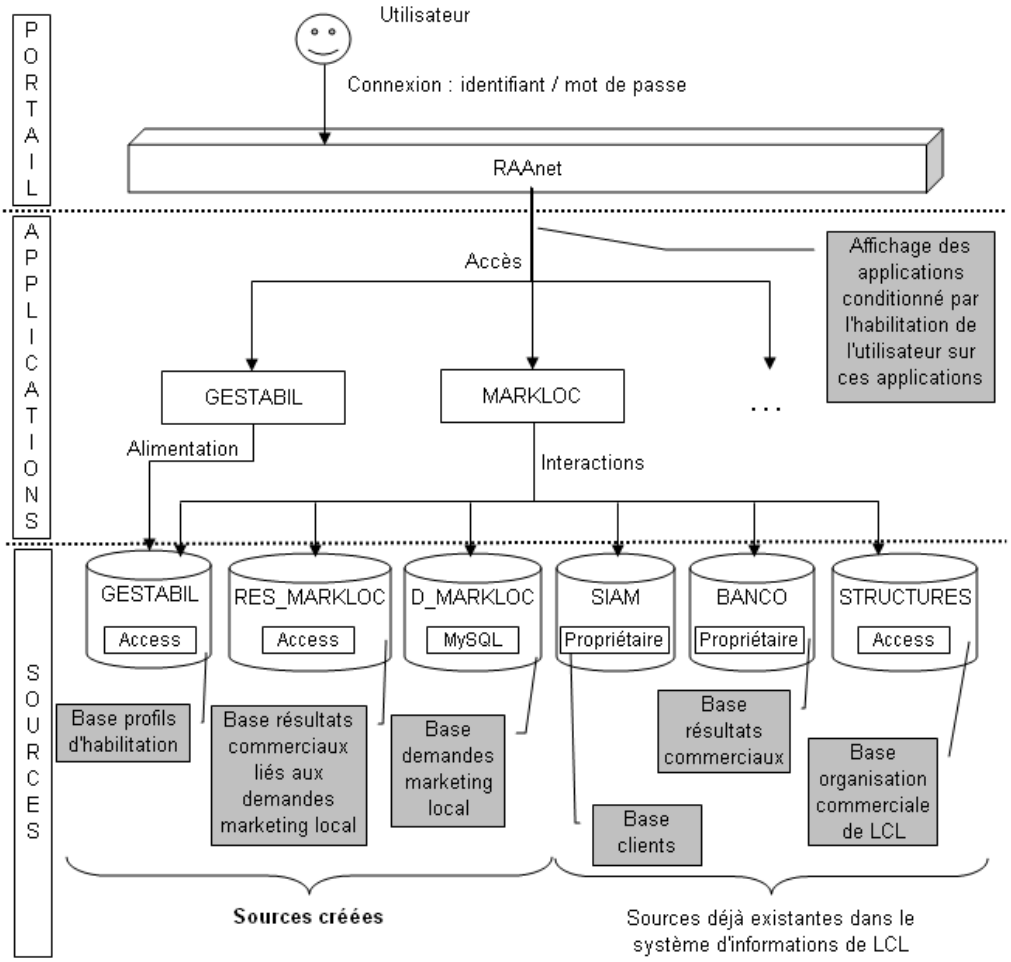


FIG. 8.1 – Ressources de la plateforme MARKLOC

L'accès à la plateforme MARKLOC se fait par l'intermédiaire du portail d'application que nous avons conçu. Ce portail s'appuie sur un système d'habilitations que nous avons développé et baptisé GESTABIL. Il comprend une base de données pour stocker ces habilitations et une interface d'alimentation, accessible d'ailleurs via le portail d'application pour les personnes travaillant au Pôle Outils et Méthodes. En effet, d'après les données de GESTABIL, ces personnes ont l'habilitation pour ac-

céder à l'application GESTABIL. Nous revenons par la suite plus en détails sur ce système d'habilitations.

Ce système d'habilitation est également exploité dans l'application MARKLOC. En effet, conformément à la structure hiérarchique présente chez LCL, les employés, selon leur position dans la pyramide, ont des rôles à jouer différents dans l'application MARKLOC : émission de demande, validation de demande, consultation de résultats, etc.

L'application MARKLOC est basée bien évidemment sur une base de données qui contient l'ensemble des demandes (D_MARKLOC). Mais elle exploite également des ressources variées telles qu'une base contenant les résultats commerciaux liés aux demandes (la base RES_MARKLOC) qui est alimentée par des données traitées provenant de BANCO.

L'application va également chercher certaines informations telles que les actions marketing nationales dans la base SIAM qui est une base de clients dédiée au marketing.

MARKLOC exploite de façon transversale une base STRUCTURES qui contient l'organisation hiérarchique de LCL.

8.3 Gestion des habilitations : l'application GESTABIL

L'objectif est de pouvoir gérer les habilitations des collaborateurs pour les applications de l'Intranet. Pour ce faire, la notion de profil est utilisée. En effet, il s'agit de pouvoir affecter à un collaborateur un profil utilisateur en fonction de son poste et de disposer des différents renseignements concernant celui-ci comme son unité d'affectation par exemple. Une interface Web permet de gérer l'affectation des profils aux collaborateurs. Le modèle conceptuel de la base des habilitations est présenté dans la figure 8.2.

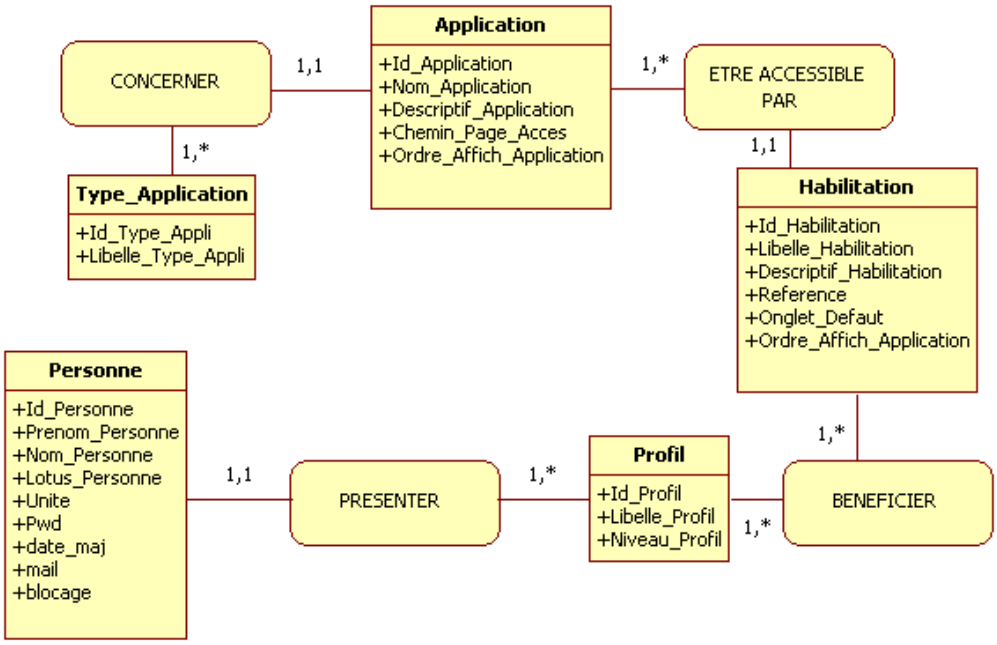


FIG. 8.2 – Schéma conceptuel E/A de la base GESTABIL

Chaque personne dispose d'un profil qui correspond finalement à une notion de métier. Ensuite, un profil peut bénéficier d'habilitations. Les applications, qui sont de différents types, peuvent présenter plusieurs habilitations. Par exemple, l'application MARKLOC présente des habilitations telles que la saisie de demande, la consultation de résultats.

8.4 Base de données des demandes marketing

8.4.1 Modélisation de la base des demandes

La base des demandes constitue le pivot de l'application MARKLOC. Comme nous l'avons précisé dans [FBBN05], lors de la conception de cette base, nous avons tenu compte du caractère générique de la demande. En effet, nous avons pris en compte le fait qu'il peut y avoir plusieurs sortes de demandes, même si la demande la plus fréquemment émise est la demande d'action marketing. Par exemple, nous avons également prévu dans MARKLOC la demande d'analyse. Le modèle UML pour la base des demandes marketing est présenté dans la figure 8.3.

Les éléments principaux de ce modèle sont les suivants. Une demande marke-

ting présente un certain type qui correspond à un ensemble de descripteurs spécifiques, ajoutés aux descripteurs communs à tous les types de demande. Elle cible une certaine unité. Elle est émise par un expéditeur et validée par plusieurs hiérarchiques. Les émetteurs des demandes et les valideurs sont des collaborateurs qui appartiennent à une unité. Par ailleurs, cette demande porte ou non sur des produits qui sont regroupés en famille de produits. Ces produits peuvent faire l'objet ou non de kit marketing.

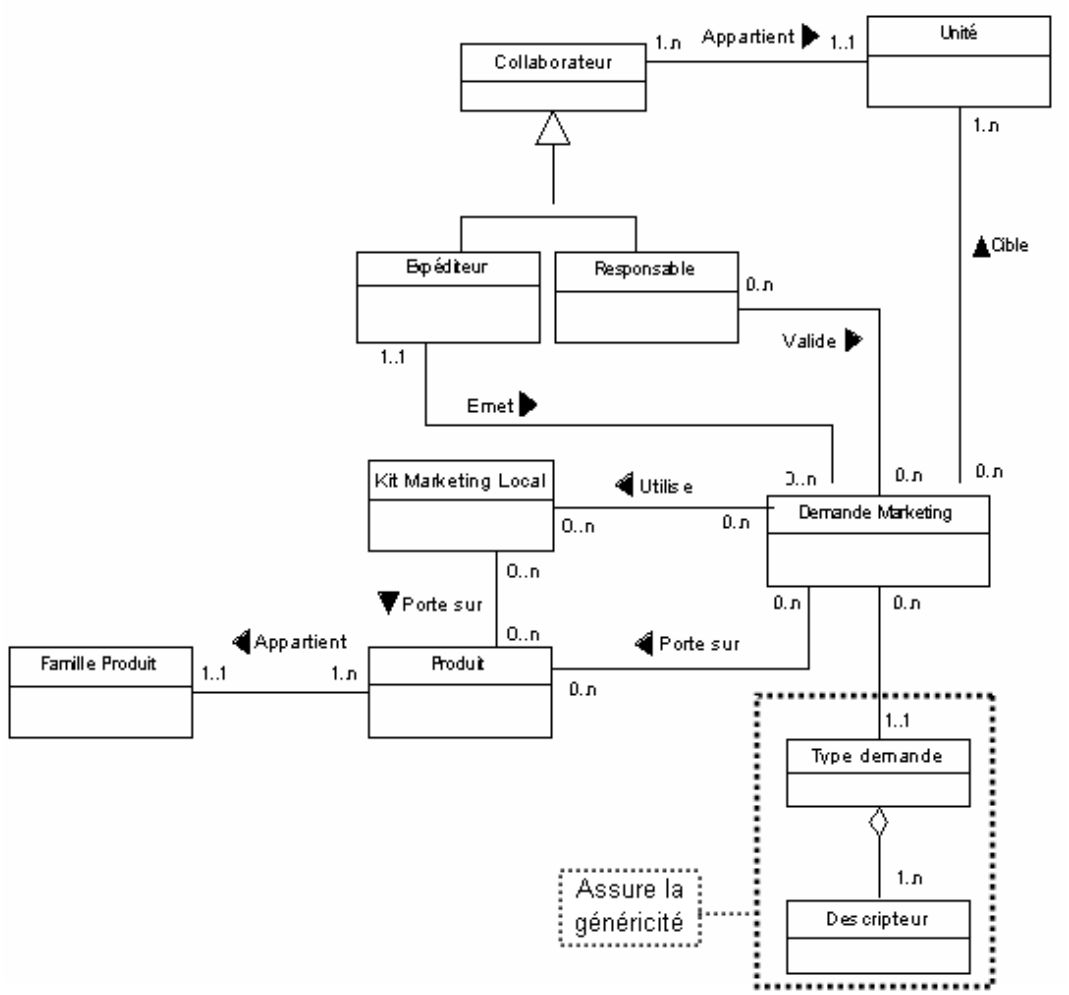


FIG. 8.3 – Modèle pour la base des demandes marketing

Dans les descripteurs de la demande, que nous n'avons pas représentés dans le modèle pour des raisons de clarté, nous retrouvons deux descripteurs qui correspondent respectivement à l'état actuel de la demande et l'état à venir. Ces deux notions sont importantes dans la mesure où finalement l'alimentation de la base des demandes,

pour une demande donnée, ne se fait pas en une seule étape. Les informations sont rajoutées dans la base au fur et à mesure du cycle de vie de celle-ci.

8.4.2 Alimentation de la base des demandes

La base des demandes a été implémentée dans MySql. L'alimentation est réalisée grâce à la saisie des demandes. Mais une demande émise va subir des validations, va être attachée à un ciblage, etc. Ainsi, l'émission de la demande marketing peut être vue comme le début d'un processus général. Le cheminement de la demande jusqu'à sa réalisation (listing ou mailing) est différent selon le niveau hiérarchique de l'expéditeur. On retrouve alors les trois diagrammes de collaboration requis dans les figures 8.4, 8.5 et 8.6. Reprécisons, pour la bonne compréhension de ces diagrammes, que hiérarchiquement la `DE` (Direction d'Exploitation) est au-dessus des `DPP` (Direction Particuliers Professionnels), qui sont elles-mêmes au-dessus des `UC` (Unité Commerciale). À chaque étape de ce cheminement des informations viennent compléter les données sur la demande.

Ainsi, le cycle de vie d'une demande de marketing local correspond à un enchaînement d'activités réalisées par différents acteurs (émission d'une demande, validation hiérarchique, etc.). Pour permettre l'alimentation de la base des demandes, il était nécessaire de considérer les différents états de celle-ci (demande soumise, demande validée, etc.) et le fait que l'état évolue suite à une intervention humaine. Ainsi, dans l'objectif d'informatisation du processus des demandes de marketing, nous nous sommes intéressés au concept de workflow (littéralement, flux de travail).

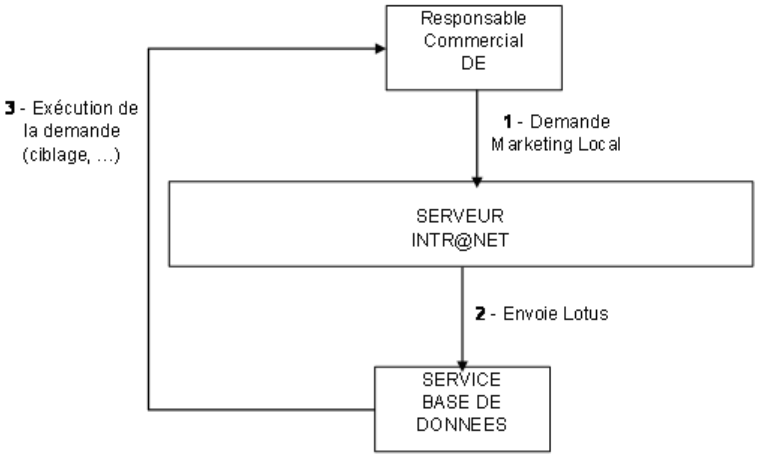


FIG. 8.4 – Diagramme de collaboration pour une demande émise au niveau DE

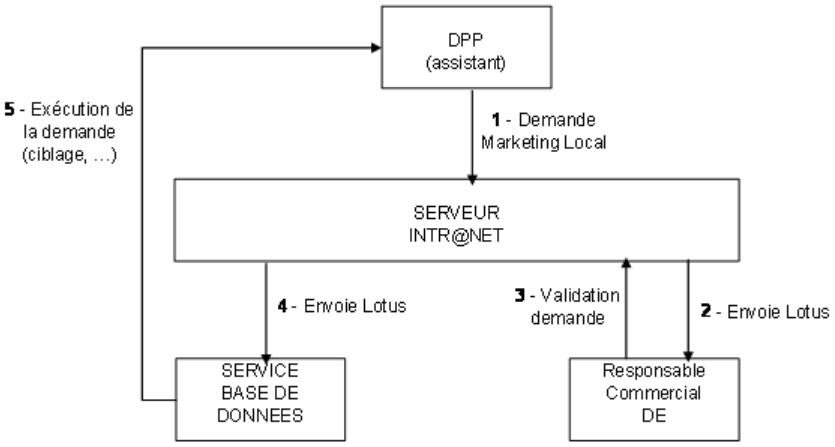


FIG. 8.5 – Diagramme de collaboration pour une demande émise au niveau DPP

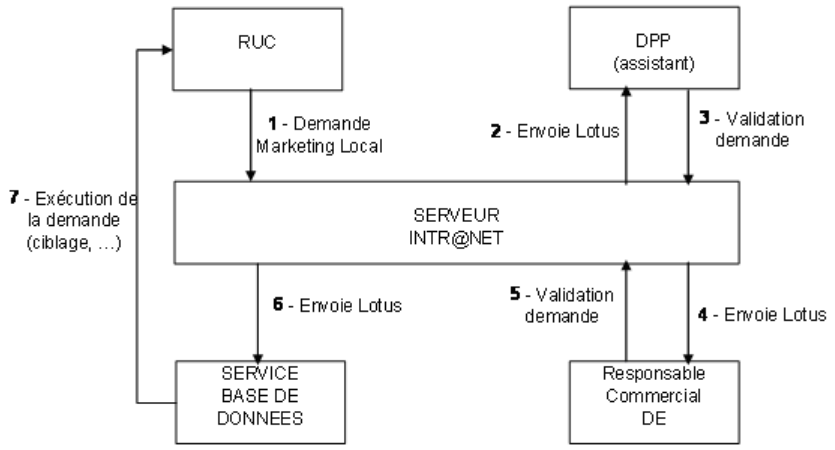


FIG. 8.6 – Diagramme de collaboration pour une demande émise au niveau UC

En effet, un workflow peut être défini comme étant une collection de tâches organisées pour l'accomplissement d'un processus [GHS95]. L'enchaînement de ces tâches est conditionné par des événements. Les workflows constituent un domaine de recherche à part entière. Nous nous sommes alors contentés d'en exploiter les concepts principaux et d'implémenter le processus des demandes marketing en le considérant comme un workflow sans aller vers une modélisation formelle de ce flux, un recours à des outils dédiés, etc.

Nous avons tenu à prendre en compte certains concepts essentiels. Par exemple,

l'adhésion des utilisateurs à une nouvelle méthodologie est d'autant plus importante que l'éloignement avec l'existant est faible [Cun01]. Ainsi notre objectif était de fournir une plateforme informatisée qui colle au plus près du fonctionnement sans outil informatique.

Dans le processus des demandes marketing, l'intervention humaine est requise à différents niveaux (validation d'une demande, réalisation d'une demande, etc.). La personne qui doit réaliser une tâche doit savoir ce qu'elle a à faire, c'est-à-dire obtenir l'information nécessaire. Concernant, l'accès aux informations, on distingue deux techniques : push ou pull. Le principe push signifie que l'information est transmise automatiquement, elle vient à l'utilisateur. Au contraire le principe pull désigne le fait que l'utilisateur doit aller rechercher l'information. La littérature souligne que l'utilisation du principe push n'induit pas une bonne adhésion des utilisateurs, c'est pourquoi ce principe est utilisé pour des processus à très grande fréquence, avec des degrés d'urgence importants. Dans la plupart des cas, il est préférable d'adopter le principe pull, qui permet aux utilisateurs une certaine forme d'indépendance. La solution idéale serait alors de permettre une utilisation des deux principes [Cun01], la difficulté étant de déterminer le mode de diffusion pour chaque information.

C'est donc vers cette solution que nous avons tendu dans la mesure où des e-mails sont envoyés automatiquement lorsqu'il y a un traitement particulier, pouvant être qualifié d'urgent, à effectuer tel qu'une validation. Et que pour d'autres tâches, c'est la personne qui va prendre l'initiative d'aller chercher l'information comme c'est le cas par exemple pour le suivi hebdomadaire des résultats.

8.5 Fonctionnalités offertes par MARKLOC

Dans cette section, nous présentons à présent les différentes fonctionnalités de l'application MARKLOC.

8.5.1 Aide au respect du protocole

Afin de mener une action de marketing local dans les meilleures conditions, il existe une certaine philosophie d'utilisation de la plateforme sur le plan commercial. Cette philosophie devait d'ores et déjà être appliquée avant l'informatisation du processus, mais ce n'était pas forcément le cas.

Notre objectif était alors de favoriser le suivi d'un protocole pour préparer une demande de marketing local : une démarche de réflexion doit précéder la saisie d'une demande sur l'outil. La démarche préconisée est la suivante :

- Déterminer l'objectif de la demande marketing en nombre de produits vendus ou en capitaux (ex : réaliser 20% de l'objectif annuel en Plan Epargne Logement sur les mois de septembre-octobre)
- Vérifier dans les actions menées au niveau national si une action similaire n'est pas prévue
- Si ce n'est pas le cas, déterminer la cible de clients. Pour cela, il est possible de se baser dans certains cas (pour les produits pour lesquels cela a été prévu) sur des critères prédéfinis, ce qui est appelé kit marketing ; ces critères peuvent être relaxés ou affinés (ex : existence d'un kit pour les PEL, critères du kit avec une restriction sur l'âge, entre 25 et 60 ans)
- Déterminer les moyens de l'action (ex : «accrochage» au niveau des guichets, contact téléphonique du conseiller, etc.)

Pour faciliter cette démarche, nous fournissons un accès direct au niveau de la plateforme aux actions marketing nationales, ainsi qu'aux critères préconisés des kits marketing existants. Concernant les actions nationales, nous avons développé un module qui permet de récupérer mensuellement et de façon automatique ces actions dans la source SIAM. Concernant les kits marketing, leur description et les critères définissant les profils cibles de clients ont été stockés dans la base D_MARKLOC.

8.5.2 Saisie d'une demande

À l'heure actuelle, deux types de demandes peuvent être saisies via l'application MARKLOC : des demandes d'action et des demandes d'analyse. Elles correspondent toutes deux à des demandes de ciblage. Le principe est de préciser le bénéficiaire de la demande (l'unité) et ses caractéristiques (critères de ciblage, etc.). La différence entre les deux réside dans la précision de la période d'exploitation du ciblage dans le cas où une action marketing est menée.

Afin de saisir une demande d'action, trois onglets doivent être complétés. Le premier est le bénéficiaire de la demande (figure 8.7). Le niveau hiérarchique doit d'abord être choisi, ensuite la ou les unités sont sélectionnées. Notons que les unités pouvant être choisies (autrement dit proposées pour la sélection) sont celles dépendant du périmètre commercial de l'utilisateur connecté. L'affichage du périmètre commercial est donc personnalisé.

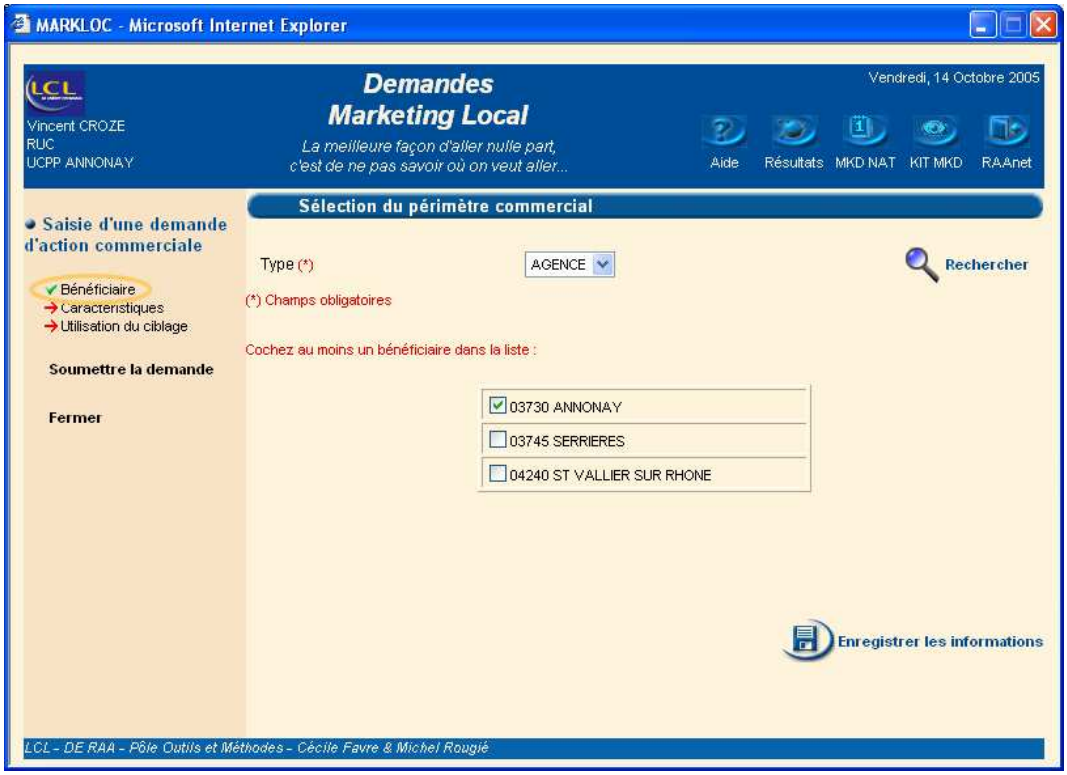


FIG. 8.7 – Sélection du bénéficiaire de la demande d'action

8.5.3 Visualisation d'une demande

Les demandes peuvent être visualisées selon deux accès.

Premièrement, il est possible de rechercher une demande via un module de recherche qui s'appuie sur les critères de bénéficiaire de la demande et de date. Concernant le bénéficiaire, il est donc possible de préciser le niveau hiérarchique de ce dernier et l'unité elle-même. Pour la date, la période de recherche doit être bornée.

Deuxièmement, lorsque la demande est en cours de traitement, elle apparaît dans l'interface principale. Son état courant apparaît de façon précise («Demande validée par le Pôle Outils et Méthodes», «Ciblage réalisé», etc.). Elle apparaît bien évidemment sur l'écran de toutes les personnes concernées par cette demande, non seulement pour l'émetteur, mais également pour le valideur par exemple.

8.5.4 Validation d'une demande

La validation des demandes est de deux types : validation hiérarchique au niveau de la hiérarchie commerciale (figure 8.8) et validation de faisabilité au niveau du

Pôle Outils et Méthodes. Ce processus de validation répond au cycle de vie présenté précédemment.

FIG. 8.8 – Validation/refus d'une demande

8.5.5 Résultats commerciaux liés à une demande

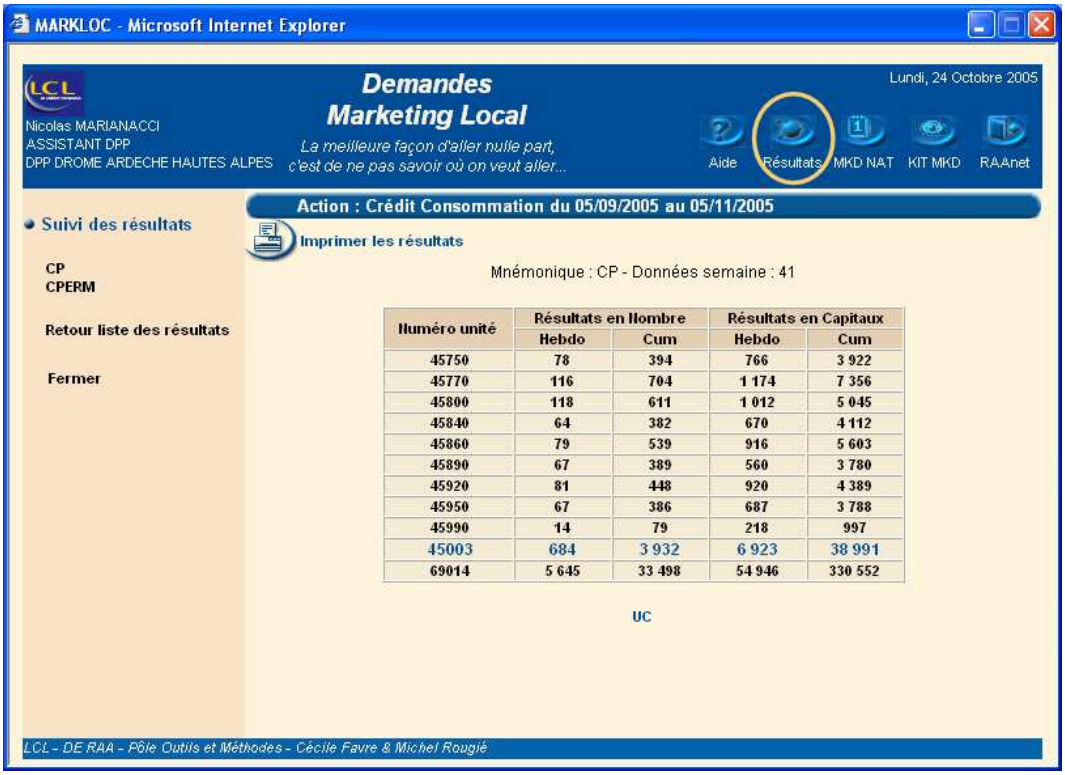
L'objectif était de pouvoir fournir des résultats commerciaux afin d'évaluer l'impact de la demande et sa réussite. Un des aspects était de ne pas attendre la fin de la demande marketing pour évaluer les ventes additives.

Or le logiciel permettant de mesurer les ventes réalisées sur les clients ciblés (SIAM) ne fournissait les résultats qu'après la fin de la demande, environ 45 jours après en raison du temps de chargement des données. Notre objectif était donc de fournir une mesure de résultats hebdomadaire, en utilisant une autre source de données qui ne traduit pas les ventes sur les clients ciblés mais les ventes déclarées par les conseillers. Ainsi, nous fournissons la possibilité de suivre de façon hebdomadaire les ventes sur des produits donnés qui font l'objet d'une action marketing. Nous partons du postulat que les responsables connaissent bien leur périmètre commercial et sont en mesure d'évaluer les ventes additionnelles sur un produit donné, pour une

semaine donnée.

Ce processus de résultats est entièrement automatisé. Les résultats commerciaux sont contenus dans une source dénommée BANCO, dont les données sont accessibles via Business Object. Ainsi des requêtes paramétrées permettent de récupérer les données concernant les actions pour lesquelles il y a un suivi de résultats. Les paramètres sont par exemple les unités sur lesquelles on récupère les résultats, les produits dont il faut suivre les ventes etc. Ces données sont ensuite stockées dans une base Access nommée RES_MARKLOC. Le choix d'Access a été fait en raison des besoins de procédures qui devaient être appliquées pour traiter les données. En effet, BANCO permet de récupérer des données hebdomadaires et quotidiennes. En fonction de la date du début de l'action, il s'agit de calculer des agrégations pour permettre un bon suivi durant l'action. L'utilisation d'un codage en VBA (Visual Basic for Application) était le plus simple. De plus, le recours à une base Access ne pose pas de problème dans la mesure où les données de cette base ne sont que consultées (pas de mise à jour, de suppression, etc.).

Pour chaque action, les résultats commerciaux sont disponibles (figure 8.9), avec une possibilité de visualiser les données selon les différents niveaux de la structure organisationnelle de LCL. Il y a un résultat hebdomadaire, ainsi qu'un cumul depuis le début de l'action. L'affichage des unités dépend du niveau de la demande, mais également du profil et du périmètre commercial de la personne connectée.



Numéro unité	Résultats en Nombre		Résultats en Capitaux	
	Hebdo	Cum	Hebdo	Cum
45750	78	394	766	3 922
45770	116	704	1 174	7 356
45800	118	611	1 012	5 045
45840	64	382	670	4 112
45860	79	539	916	5 603
45890	67	389	560	3 780
45920	81	448	920	4 389
45950	67	386	687	3 788
45990	14	79	218	997
45003	684	3 932	6 923	38 991
69014	5 645	33 498	54 946	330 552

FIG. 8.9 – Suivi des résultats commerciaux pendant l'action marketing

8.5.6 Tableaux de bord de l'activité marketing local

Pour la personne responsable du Pôle Outils et Méthodes, il est nécessaire de disposer de tableaux de bord permettant de mesurer l'activité de ciblages. Nous proposons alors une interface qui permet de générer différents tableaux de bord en précisant la période d'analyse.

Par exemple, il est possible d'obtenir le nombre de demandes réalisées par unité, par type de demande (action, analyse) ; de connaître le nombre de ciblages réalisés par personne travaillant au Pôle Outils et Méthodes (figure 8.10), permettant d'équilibrer à terme la répartition de charge de travail, etc.

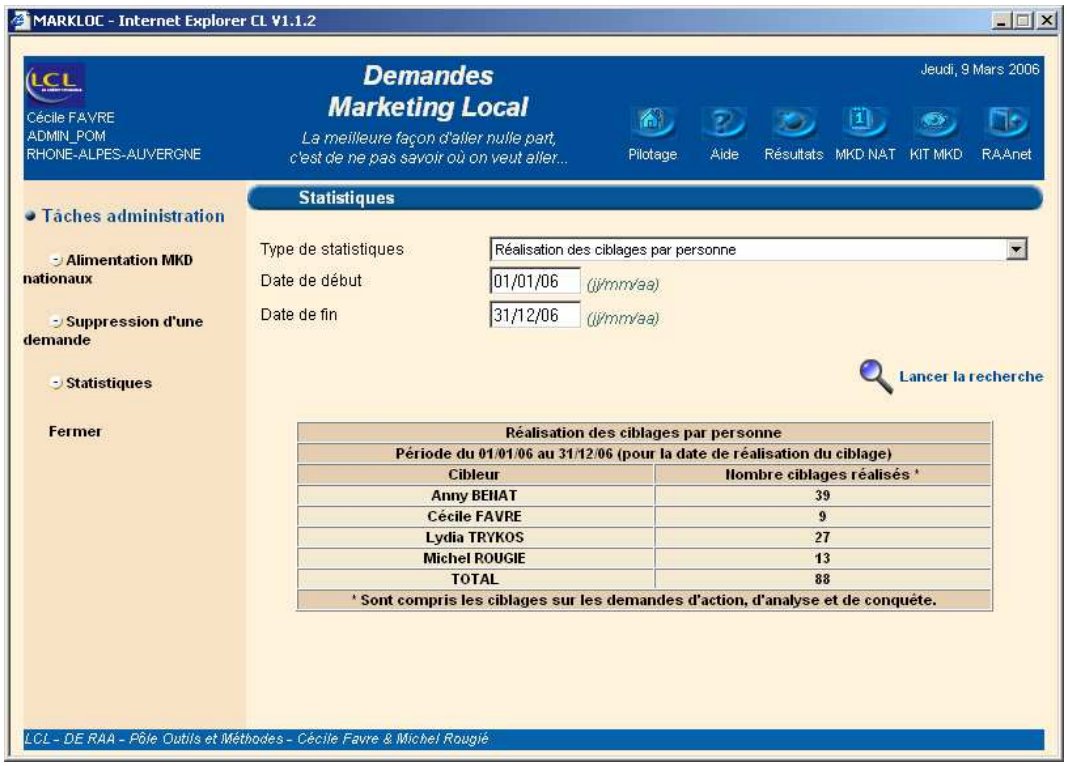


FIG. 8.10 – Tableau de bord : réalisation de ciblage par personne

8.6 Personnalisation : au cœur de MARKLOC

Si notre proposition de personnalisation permet l'extension des possibilités d'analyse, dans le cadre de MARKLOC, nous nous sommes également intéressés à la personnalisation dans le contexte plus classique de restriction. En effet, dans le contexte des interfaces homme-machine, des bases de données et de la recherche d'information, l'objectif de la personnalisation est de restreindre les informations, afin que l'utilisateur n'ait que des réponses pertinentes vis-à-vis de ses objectifs.

Comme nous l'avons indiqué dans l'état de l'art, la personnalisation peut se baser sur le concept de profil et d'éléments qui concernent l'usager. C'est cette alternative que nous avons utilisée dans le cadre de MARKLOC. En effet, nous nous appuyons sur la notion de profil, au sens du métier de l'utilisateur, ainsi que sur l'unité à laquelle il est rattachée (le périmètre commercial).

Ces deux informations sont exploitées en permanence dans MARKLOC pour personnaliser à la fois la restitution d'informations et les interfaces.

Le profil et l'unité conditionnent par exemple le résultat d'une recherche de de-

mandes. Ainsi, lorsque la requête de recherche est bâtie en fonction des critères précisés par l'utilisateur, elle est enrichie de nouveaux prédicats dans la clause de condition WHERE afin que seules les demandes la concernant directement ou indirectement apparaissent.

Il en est de même pour l'affichage des résultats commerciaux liés aux demandes d'action marketing. Selon le profil, on n'obtient des niveaux de détail de l'information différents. Quant à l'unité de rattachement, elle permet de filtrer les résultats qui concernent seulement les unités liées à elles (de niveau hiérarchiquement inférieur ou supérieur). Si nous nous basons sur l'exemple de l'`UC` d'Annonay, le responsable pourra visualiser les résultats de toutes les unités qui composent cette `UC`, mais également les résultats de la `DPP DRÔME ARDÈCHE HAUTES-ALPES`.

Le principe de personnalisation est également introduit au niveau des interfaces. C'est le cas dès l'accès au portail où n'apparaissent que les applications par lesquelles l'utilisateur est concerné. Notons que cette notion de personnalisation rejoint ainsi parfois la notion de droit d'accès.

Dans ce chapitre, nous avons présenté notre travail de développement. Ce travail a été réalisé sous deux angles. Premièrement, dans un contexte industriel, nous avons conçu et développé la plateforme MARKLOC, qui permet la gestion de l'ensemble du processus des demandes de marketing local de la direction d'exploitation Rhône-Alpes Auvergne. Deuxièmement, afin de valider nos propositions, nous avons réalisé le prototype WEDriK qui peut, moyennant quelques initialisations préalables, être utilisés avec n'importe quel entrepôt de données.

8.7 Conclusion

Dans ce chapitre, nous avons présenté le développement que nous avons réalisé dans un contexte industriel. Il s'agit d'une plateforme qui permet la gestion de l'ensemble du processus des demandes marketing.

Le développement de la plateforme MARKLOC a été précédé d'une phase relativement longue qui a permis de bien recueillir les attentes des utilisateurs, mais également de bien modéliser tous les éléments relatifs à cette plateforme. Ainsi, la phase de développement s'en est trouvé facilitée. Cette conception préalable complète, ainsi qu'un protocole de tests réalisés pour tenir compte à la fois de l'ensemble du cheminement de la demande et des différents profils ont permis une mise à disposition de l'outil sans problème. En effet, seuls quelques réglages visuels ont dû être réalisés.

Malgré une tendance à supprimer l'usage d'outils locaux et à préférer les outils mis à disposition au niveau national par LCL, la plateforme est toujours en cours d'utilisation par l'ensemble des collaborateurs de la direction d'exploitation Rhône-Alpes Auvergne, aux différents niveaux hiérarchiques.

Un travail d'enrichissement de cette plateforme est aujourd'hui mené par le responsable du Pôle Outils et Méthodes afin de doter le système de fonctionnalités supplémentaires, telles que la mise en ligne des listes de clients ciblés et le suivi de leur traitement. En effet, il est intéressant de pouvoir être en mesure de savoir combien de clients ont été effectivement contactés pour l'opération puisque la réussite de l'opération marketing en dépend en partie.

Chapitre 9

Conclusion générale

9.1 Bilan et contributions

Cette thèse a été réalisée dans le cadre d'une convention CIFRE entre le laboratoire ERIC et l'établissement bancaire LCL. Au-delà de l'aspect d'ingénierie que revêtait la mission réalisée en entreprise, LCL a su suscité des problématiques scientifiques bien réelles. En effet, le souci de la personnalisation des outils est permanent chez LCL. Ceci est sans doute dû au fait que les employés exercent des métiers différents tout en faisant partie de la même banque, ils ont donc des attentes et des besoins différents, etc.

Ainsi, nous avons proposé une solution de personnalisation qui consiste à recueillir les connaissances des utilisateurs pour créer de nouveaux axes d'analyse répondant à leurs propres besoins. Cette solution se base sur une évolution du schéma de l'entrepôt guidée par les utilisateurs qui vise plus précisément à mettre à jour les hiérarchies de dimension en créant de nouveaux niveaux de granularité. Nous avons alors tenté de redonner tout son sens à l'expression «technologie centrée utilisateur» en le plaçant au cœur du processus d'évolution, en plus du fait qu'il soit maître de l'analyse au sens de la navigation.

Notre approche de personnalisation se basant sur une évolution de schéma, elle permet de rendre les nouvelles possibilités d'analyse pérennes et partageables avec d'autres utilisateurs, ce qui n'est pas le cas avec la proposition de création de variables offerte par certains éditeurs de logiciels décisionnels.

Vis-à-vis de cette évolution de schéma, notre approche de personnalisation s'inscrit dans une alternative de mise à jour de schéma. Un des avantages est alors de pouvoir la mettre en œuvre sur n'importe quel entrepôt de données existant sans complexité particulière.

Du point de vue de cette mise à jour de schéma, le principe que nous utilisons se rapproche des travaux proposés dans [BSH99], dans lesquels un ensemble d'opérateurs de mise à jour de schéma étaient proposés. Dans notre travail, nous nous sommes concentrés sur une partie des évolutions proposées et ce, pas seulement au niveau structurel. En effet, dans notre approche, nous nous sommes intéressés également à fournir les données nécessaires pour l'évolution, en exploitant la connaissance des utilisateurs.

Notre objectif d'enrichissement des hiérarchies de dimension est commun avec celui de Mazon et al. [MT06]. Les deux approches diffèrent néanmoins sur le moyen d'y parvenir. En effet, leur approche vise à enrichir les hiérarchies de dimension de façon automatique, en exploitant WordNet. Mais dans notre cas, cet enrichissement est réalisé grâce aux utilisateurs, donnant ainsi une place centrale à l'utilisateur dans le processus d'évolution.

Cet enrichissement permet alors de créer de nouveaux chemins d'agrégation, allant au-delà de la proposition faite dans [EV01], dans laquelle les utilisateurs pouvaient seulement modifier les chemins d'agrégation existants, en exprimant des exceptions dans le processus d'agrégation au niveau des instances.

Concernant les travaux traitant de la personnalisation dans les entrepôts de données eux-mêmes, il s'avère que notre approche s'inscrit dans une perspective différente des travaux émergents dans le domaine. En effet, les principaux travaux se basent sur l'expression de préférences pour personnaliser le processus d'analyse en diminuant les réponses aux requêtes [BGMM06, BGM+05] ou en diminuant le nombre d'opérations à réaliser lors de la navigation [RTZ07]. Dans notre travail, la personnalisation n'est pas fondée sur une expression de préférences pour gérer des possibilités existantes. Il s'agit au contraire d'étendre ces possibilités en permettant la réalisation de nouvelles analyses qui soient personnalisées par rapport aux besoins des utilisateurs, en prenant en compte leurs propres connaissances.

Notre démarche se base alors sur une architecture globale comprenant quatre modules :

- un module d'acquisition des connaissances utilisateurs sous forme de règles d'agrégation de type «si-alors» ;
- un module d'intégration des règles d'agrégation dans l'entrepôt de données ;
- un module d'évolution de schéma permettant la mise à jour des hiérarchies de dimension ;
- un module d'analyse permettant à l'utilisateur d'avoir de nouvelles analyses OLAP basées sur le nouveau schéma.

Pour soutenir cette architecture, nous avons défini un modèle d'entrepôt de données évolutif à base de règles d'agrégation *R-DW*. 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. Elle constitue une réponse à des besoins d'analyse initiaux, définis lors de la conception de l'entrepôt. Ces besoins initiaux peuvent être considérés comme communs à l'ensemble des utilisateurs. La partie évolutive est composée de l'ensemble des hiérarchies de dimension pouvant être mises à jour par les utilisateurs. Nous nous assurons que ces mises à jour n'introduisent pas d'incohérences dans les analyses en les propageant au niveau du schéma. Cette partie évolutive apporte ainsi une flexibilité pour la prise en compte de nouveaux besoins d'analyse.

En outre, nous avons proposé un méta-modèle de l'entrepôt de données évolutif qui nous permet d'appliquer notre démarche sur n'importe quel entrepôt. En effet, ce méta-modèle permet de décrire n'importe quel entrepôt de données évoluant selon notre approche. Nous assurons ainsi la généricité de notre modèle évolutif.

Nous avons déployé notre approche dans un contexte relationnel en proposant un modèle d'exécution qui a pour but de gérer l'ensemble des processus liés à l'architecture, de l'acquisition des règles à l'évolution du schéma.

Par ailleurs, nous nous sommes intéressés à l'évaluation de la performance de notre modèle évolutif. Étant donné que l'évaluation de la performance dans les entrepôts de données est généralement basée sur une charge, nous avons proposé une méthode de mise à jour incrémentale de la charge en fonction des modifications subies par le schéma de l'entrepôt, afin que la charge reste cohérente vis-à-vis de ces modifications.

LCL a non seulement suscité notre problématique de personnalisation, mais a constitué par la suite un terrain d'application pour la mise en œuvre de nos propositions en matière de personnalisation. Ainsi, l'ensemble de nos propositions a fait l'objet d'une implémentation au travers de la plateforme WEDriK qui permet donc d'impliquer les utilisateurs dans l'évolution du schéma de l'entrepôt. Nous avons donc appliqué notre démarche sur l'entrepôt de données test LCL-DW construit à partir de données réelles de LCL.

Cette expérience de thèse en collaboration avec LCL a été enrichissante tant sur le plan humain que professionnel. J'ai eu l'occasion d'utiliser mes compétences, mais j'ai également beaucoup appris. Par exemple, j'ai pu acquérir de nouvelles connaissances sur les structures organisationnelles, les produits, le marketing, etc.

LCL est une entreprise nationale qui présente l'avantage d'être fortement struc-

turée tout en permettant un dynamisme dans les différents niveaux hiérarchiques. En effet, j'ai pu réellement apprécier le fait que la direction d'exploitation Rhône-Alpes Auvergne soit très active en matière de développement d'outils en local pour faciliter le travail de ses employés. D'ailleurs, ces outils sont parfois repris au niveau national pour en faire profiter les autres directions d'exploitation. Un tel fonctionnement me paraît pertinent dans la mesure où les personnes développant ces outils au niveau local sont plus à même de déterminer les besoins réels des employés de part leur proximité. Partir des besoins est en effet essentiel.

C'est ce que nous avons fait dans le cadre de la conception et du développement de la plateforme MARKLOC. En effet, nous avons réalisé une importante étude préalable des besoins, de l'existant, etc. Dans le cadre de cette étude pour le développement de MARKLOC et d'autres outils, j'ai pu constater la difficulté pour des opérationnels à définir un cahier des charges. En effet, il s'agit de travailler avec des personnes qui ne sont pas informaticiennes, qui ne peuvent donc mesurer le travail nécessaire au développement et n'ont pas conscience de l'importance d'être précis dans l'expression des besoins.

J'ai pu alors développer ma capacité à être une véritable interface entre les utilisateurs et la définition d'un cahier des charges. Il s'agit réellement de passer du langage naturel à un langage technique, une formalisation des problèmes et des solutions que l'on peut apporter au niveau informatique.

Ce travail est d'autant moins évident, qu'il est très difficile pour les utilisateurs d'exprimer de façon exhaustive les besoins, les procédures existantes, etc. Cela implique, lors de la récolte des informations, d'être capable de creuser par des questions, pour être sûr qu'aucun détail ne nous a échappé. Dans ce contexte, les validations successives sont nécessaires. Elles permettent de faire émerger des points qui n'avaient pas été abordés, des précisions qui ont leur importance.

Dans le cadre du développement de la plateforme MARKLOC, la phase de conception et la phase de tests ont été relativement longues. Mais je suis convaincue aujourd'hui qu'il s'agit de la meilleure façon de faire. En effet, il y a une forte tendance dans les entreprises à développer des outils dans l'urgence sans forcément prendre du temps pour réfléchir de façon posée au problème, à la conception de l'outil. Or, j'ai pu constater, que ce temps est bénéfique non seulement vis-à-vis du développement lui-même, mais également pour l'évolution de l'outil. En effet, en prenant le temps de bien concevoir un outil, il est généralement plus facile de le faire évoluer. De plus, le fait de mettre à disposition l'outil sans qu'aucun problème technique ne soit relevé est une réelle satisfaction. Pour permettre ce résultat, une longue phase de tests a été mise en œuvre. Nous avons dû définir un protocole qui

envisage tous les scénarios possibles : en fonction des différents utilisateurs (différents profils), des différentes interventions qu'ils peuvent réaliser (demandes validées ou refusées par exemple), etc.

La plus grande satisfaction que l'on peut avoir par rapport à un projet de développement de ce type est bien sûr de constater que le produit est réellement utilisé, d'avoir des retours sur le fait que les personnes qui l'utilisent apprécient l'outil. En effet, cette satisfaction n'est pas forcément obtenue dans le cadre du développement réalisé dans un contexte scientifique pour tester et valider des propositions qui ne seraient pas des réponses à des besoins réels (d'une entreprise par exemple).

Ainsi, je suis convaincue qu'il est réellement intéressant de pouvoir développer des collaborations entre les laboratoires de recherche et les entreprises. Cela permet, pour les premiers, de s'imprégner de la réalité des problèmes des entreprises et de voir émerger de réelles problématiques scientifiques et, pour les seconds, de bénéficier de forces de proposition pour résoudre ces problématiques. En effet, les objectifs des uns et des autres ne sont pas incompatibles, il finissent par converger.

C'est ce qui s'est produit dans mon cas. La spécificité de ma thèse était d'arriver à mener un projet selon deux points de vue différents mais complémentaires : ingénierie d'une part, et scientifique d'autre part. Le fait d'être imprégnée de problèmes réels, de se nourrir de lectures, etc. a permis de proposer des solutions à des problèmes concrets, en les mettant en œuvre.

L'objectif de LCL était de disposer d'un outil qui fonctionne, facile à utiliser et répondant à certains besoins. L'objectif pour le laboratoire se situe davantage au niveau de la proposition de solutions à des problèmes et à la publication de contributions expliquant ces solutions, solutions pouvant être validées par un travail de développement. Les objectifs convergent alors lorsque le laboratoire va être à même de proposer des solutions scientifiques à des problèmes réels basées sur des outils utilisables.

La plus grande satisfaction que l'on peut obtenir par rapport à une thèse de ce type est alors d'obtenir une satisfaction finale vis-à-vis des attentes de l'entreprise et vis-à-vis des attentes du laboratoire, ce qui est mon cas aujourd'hui. En effet, la plateforme MARKLOC est aujourd'hui régulièrement utilisée et est appréciée de ses utilisateurs. Sur le plan scientifique, nous avons pu faire des propositions qui ont été validées par des publications.

9.2 Perspectives de recherche

Concernant notre proposition de personnalisation, les discussions que nous avons menées sur notre travail tout au long de ce mémoire ont fait émerger de nombreuses perspectives, directement liées à notre travail sur la personnalisation, ou de façon plus large sur l'évolution de schéma et la performance.

Tout d'abord, un des points cruciaux qu'il nous reste à explorer dans le cadre de notre proposition est la gestion de l'évolution des règles. Si ce problème peut être abordé sous l'angle de l'évolution de schéma de façon générale avec les alternatives que l'on connaît de mise à jour ou de versionnement, il n'en demeure pas moins que les particularités de notre approche doivent être prises en compte. L'une des particularités les plus notables est l'implication de l'utilisateur dans le processus de mise à jour des hiérarchies de dimension. Il s'agit alors de connaître quels sont les besoins réels au niveau de l'historisation des dimensions. Mais il s'agit également de bien prendre en compte une certaine facilité d'utilisation.

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. Cette méthode permettrait également de découvrir des regroupements intéressants pour l'analyse auxquels l'utilisateur n'aurait pas pensé. Par exemple, dans [RB07], les auteurs proposent d'utiliser la méthode des K-means pour construire les classes de regroupement d'instances pour représenter le nouveau niveau de hiérarchie à créer. Notre approche étant basée sur des règles d'agrégation de type «si-alors», il nous paraît intéressant de pouvoir générer ces règles de façon automatique par l'application d'une méthode d'apprentissage non supervisée. Ainsi nous pensons que l'algorithme KEROUAC [JN03] peut répondre à notre objectif. KEROUAC correspond à l'acronyme des termes anglais Knowledge Explicit Rapid Off-beat User-centered. Ces termes font référence aux caractéristiques de la méthode : caractérisation explicite des classes (Knowledge Explicit), coût calculatoire relativement faible (Rapid), bonne utilisabilité (User-centered). L'un des aspects qui nous intéresse particulièrement est que chacune des classes de la partition résultat est caractérisée par une règle logique ; alors que très souvent le résultat des algorithmes fournit la composition des classes uniquement. Dans un contexte relationnel, nous pensons qu'il serait intéressant d'intégrer cet algorithme au sein du SGBDR, en particulier si le nombre d'instances à classer est important. En effet, comme nous l'avons montré dans [BDFU07], il est possible d'intégrer les méthodes de fouille au cœur des SGBD, optimisant ainsi les temps de réponses face à de larges volumétries de données à

traiter, en exploitant les outils du SGBD tels que les index bitmap par exemple comme nous l'avons proposé dans [FB05b, FB05a].

Dans notre travail, la personnalisation a pour objectif de répondre à des besoins d'analyse spécifiques donnant la possibilité aux utilisateurs de créer de nouveaux niveaux de hiérarchie. Ces niveaux créés peuvent intéresser d'autres utilisateurs. Il est donc crucial qu'un utilisateur qui réalise une analyse en fonction d'un niveau créé par un autre utilisateur connaisse exactement la sémantique de ce niveau. Pour ce faire, nous pensons que le recours à un processus d'annotations, comme il a pu être proposé dans [CCRT07], peut être pertinent. En effet, dans ce travail les auteurs traitent du concept de mémoire d'expertises décisionnelles. Un des objectifs de cette mémoire est d'éviter la perte de connaissances lors du départ d'un collaborateur et de faciliter le transfert de ces connaissances entre les collaborateurs. Deux aspects ont retenu plus particulièrement notre attention dans cette proposition. Il s'agit d'une part de l'idée de préciser la sémantique au niveau des concepts dans le schéma. D'autre part, il s'agit de l'idée d'usage collectif, de partage d'expertises. Ainsi, notre approche présente ces deux idées : il est en effet crucial de pouvoir préciser la sémantique du niveau créé dans le schéma afin de pouvoir partager cette possibilité d'analyse supplémentaire avec d'autres utilisateurs. Le créateur du nouveau niveau de hiérarchie pourrait annoter celui-ci afin de lui donner une bonne description, pour que la compréhension soit facilitée pour les autres utilisateurs. C'est ce qui permettra un réel partage des nouvelles possibilités d'analyse en assurant la bonne interprétation de ces analyses. Rappelons que 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).

Dans le contexte des entrepôts de données évolutifs, nous pensons qu'une «simple» maintenance des structures d'optimisation peut s'avérer insuffisante. En effet, l'évolution du schéma et des données de l'entrepôt peut nécessiter une évolution de la configuration même des structures d'optimisation. Nous pensons alors que le travail réalisé pour l'évolution de charge pourrait être exploité de façon pertinente dans ce contexte. Les algorithmes de sélection d'index et de vues à matérialiser se basent généralement sur l'utilisation d'une charge. Ainsi, répercuter l'évolution du schéma et des données sur la charge pourrait permettre une gestion des performances de façon pro-active. Nous entendons par pro-active le fait de mettre à jour la charge pour sélectionner et donc éventuellement modifier la configuration des index et des vues à matérialiser. Dans ce contexte, afin d'évaluer la performance de ces entrepôts de données évolutifs, nous pensons qu'un recours au benchmark (banc d'essai) peut être pertinent. Dans ce contexte, il serait nécessaire de doter les bancs d'essais

d'opérateurs permettant de faire évoluer les schémas et les données des entrepôts.

Publications

Chapitre dans un ouvrage d'audience internationale avec comité de lecture

- **C. Favre**, F. Bentayeb, O. Boussaid, *A Survey of Data Warehouse Model Evolution*, Encyclopedia of Database Technologies and Applications, Second Edition, Idea Group Publishing, 2007. (to appear)

Articles dans une revue internationale avec comité de lecture

- F. Bentayeb, **C. Favre**, O. Boussaid, *A User-driven Data Warehouse Evolution Approach for Concurrent Personalized Analysis Needs*, Integrated Computer-Aided Engineering (ICAE), Special Issue, 2008, IOS Press. (to appear)
- F. Bentayeb, J. Darmont, **C. Favre**, C. Udréa, *Efficient On-Line Mining of Large Databases*, International Journal of Business Information Systems, Vol.2, N.3, 2007, 328-350.

Publications dans une conférence internationale avec comité de lecture

- **C. Favre**, F. Bentayeb, O. Boussaid, *Evolution of Data Warehouses' Optimization : a Workload Perspective*, 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 07), Regensburg, Germany, September 2007 ; LNCS, Vol. 4654, 13-22.
- **C. Favre**, F. Bentayeb, O. Boussaid, *Dimension Hierarchies Updates in Data Warehouses : a User-driven Approach*, 9th International Conference on Enterprise Information Systems (ICEIS 07) : Databases and Information Systems Integration, Funchal, Madeira, Portugal, June 2007 ; 206-211.
- **C. Favre**, F. Bentayeb, O. Boussaid, *A Knowledge-driven Data Warehouse*

Model for Analysis Evolution, 13th ISPE International Conference on Concurrent Engineering : Research and Applications (CE 06), Antibes, France, September 2006 ; Frontiers in Artificial Intelligence and Applications, Vol. 143, 271-278.

- **C. Favre**, F. Bentayeb, O. Boussaid, *A Rule-based Data Warehouse Model*, 23rd British National Conference on Databases (BNCOD 06), Belfast, Northern Ireland, July 2006 ; LNCS, Vol. 4042, 274-277.
- **C. Favre**, F. Bentayeb, *Bitmap Index-based Decision Trees*, 15th International Symposium on Methodologies for Intelligent Systems (ISMIS 05), New York, USA, May 2005 ; LNAI, Vol. 3488, 65-73.

Publications dans une conférence nationale avec comité de lecture

- **C. Favre**, F. Bentayeb, O. Boussaid, *Evolution de modèle dans les entrepôts de données : existant et perspectives*, 3èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 07), Poitiers, Juin 2007 ; Revue des Nouvelles Technologies de l'Information, Vol. B-3, 21-35.
- **C. Favre**, F. Bentayeb, O. Boussaid, *Evolution et personnalisation des analyses dans les entrepôts de données : une approche orientée utilisateur*, 25ème Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID 07), Perros-Guirec, Mai 2007 ; 308-323.
- **C. Favre**, F. Bentayeb, O. Boussaid, *Intégration des connaissances utilisateurs pour des analyses personnalisées dans les entrepôts de données évolutifs*, 7èmes Journées d'Extraction et de Gestion des Connaissances (EGC 07), Namur, Belgique, Janvier 2007 ; Revue des Nouvelles Technologies de l'Information, Vol. E-9, 217-222.
- **C. Favre**, F. Bentayeb, O. Boussaid, *Evolution de schémas dans les entrepôts de données : modèle à base de règles*, 2ème journée francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 06), Versailles, Juin 2006 ; Revue des Nouvelles Technologies de l'Information, Vol. B-2, 175-176.
- R. Ben Messaoud, K. Aouiche, **C. Favre**, *Une approche de construction d'espaces de représentation multidimensionnels dédiés à la visualisation*, 1ère journée francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 05), Lyon, France, Juin 2005 ; Revue des Nouvelles Technologies de l'Information, Vol. B-1, 34-50.
- **C. Favre**, F. Bentayeb, *Intégration efficace des arbres de décision dans les*

SGBD : utilisation des index bitmap, 5èmes Journées d'Extraction et de Gestion des Connaissances (EGC 05), Paris, Janvier 2005 ; Revue des Nouvelles Technologies de l'Information, Vol. E-3, 319-330.

Ateliers avec comité de lecture

- **C. Favre**, F. Bentayeb, O. Boussaid, *WEDriK : une plateforme pour des analyses personnalisées dans les entrepôts de données évolutifs*, Atelier Systèmes Décisionnels (ASD 06) en conjonction avec MCSEAI 06, Agadir, Maroc, Décembre 2006.
- **C. Favre**, F. Bentayeb, O. Boussaid, *Modèle d'entrepôt de données à base de règles*, 3ème atelier Fouille de Données Complexes dans un processus d'extraction des connaissances (FDC 06) en conjonction avec EGC 06, Lille, Janvier 2006, 39-50.
- **C. Favre**, F. Bentayeb, O. Boussaid, N Nicoloyannis, *Entreposage Virtuel de demandes marketing : de l'acquisition des objets complexes à la capitalisation des connaissances*, 2ème atelier Fouille de Données Complexes dans un processus d'extraction des connaissances (FDC 05) en conjonction avec EGC 05, Paris, Janvier 2005, 65-68.

Bibliographie

[ACN00] S Agrawal, S Chaudhuri, and V R Narasayya. Automated Selection of Materialized Views and Indexes in SQL Databases. In *XXVIth International Conference on Very Large Data Bases (VLDB 00), Cairo, Egypt*, pages 496–505. Morgan Kaufmann, 2000.

[ACWP01] J Akoka, I Comyn-Wattiau, and N Prat. Dimension Hierarchies Design from UML Generalizations and Aggregations. In *XXth International Conference on Conceptual Modeling (ER 01), Yokohama, Japan*, volume 2224 of *LNCS*, pages 442–455. Springer, 2001.

[AJD06] K Aouiche, P Jouve, and J Darmont. Clustering-Based Materialized View Selection in Data Warehouses. In *Xth East European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece*, volume 4152 of *LNCS*, pages 81–95. Springer, 2006.

[Ann07] E Annoni. *Éléments méthodologiques pour le développement des systèmes décisionnels dans un contexte de réutilisation.* Thèse de doctorat, Institut de Recherche en Informatique de Toulouse - Université Toulouse 1, Toulouse, France, 2007.

[ASA00] ASAP. Multi-Dimensional Modelling with BW. Technical report, SAP America Inc. and SAP AG, 2000.

[ASS06] A Abelló, J Samos, and F Saltor. YAM2 : A Multidimensional Conceptual Model Extending UML. *Information Systems*, 31(6) :541–567, 2006.

[BBDG05] M Badri, F Boufares, C F Ducateau, and F Gargouri. Etat de l'art de la maintenance des entrepôts de données issus de systèmes d'information hétérogènes. In *Vèmes journées scientifiques Génie Electrique et Informatique (GEI 05), Sousse, Tunisie*, pages 13–18, 2005.

[BBDR03] O Boussaid, F Bentayeb, J Darmont, and S Rabaséda. Vers l'entreposage des données complexes. structuration, intégration et analyse. *Ingénierie des Systèmes d'Information*, 8(5-6) :79–107, 2003.

[BCC+01] A Bonifati, F Cattaneo, S Ceri, A Fuggetta, and S Paraboschi. Designing Data Marts for Data Warehouses. *ACM Transactions on Software Engineering and Methodology*, 10(4) :452–483, 2001.

[BDFU07] F Bentayeb, J Darmont, C Favre, and C Udréa. Efficient On-Line Mining of Large Databases. *International Journal of Business Information Systems*, 2(3) :328–350, 2007.

[BEK+04] B Bebel, J Eder, C Koncilia, T Morzy, and R Wrembel. Creation and Management of Versions in Multiversion Data Warehouse. In *XIXth ACM Symposium on Applied Computing (SAC 04), Nicosia, Cyprus*, pages 717–723. ACM Press, 2004.

[Bel00] L Bellatreche. *Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique dŠun entrepôt de données*. Thèse de doctorat, Université Blaise Pascal, Clermont Ferrand, France, 2000.

[Bel02] Z Bellahsene. Schema Evolution in Data Warehouses. *Knowledge and Information Systems*, 4(3) :283–304, 2002.

[BFB08] F Bentayeb, C Favre, and O Boussaid. A User-driven Data Warehouse Evolution Approach for Concurrent Personalized Analysis Needs. *Integrated Computer-Aided Engineering (ICAE), Special Issue (to appear)*, 2008.

[BG02] E I Benitez-Guerrero. *Infrastructure adaptable pour les entrepôts de données*. Thèse de doctorat, Université Grenoble 1, Grenoble, France, 2002.

[BGM+05] L Bellatreche, A Giacometti, P Marcel, H Mouloudi, and D Laurent. A Personalization Framework for OLAP Queries. In *VIIIth ACM International Workshop on Data Warehousing and OLAP (DOLAP 05), Bremen, Germany*, pages 9–18. ACM Press, 2005.

[BGMM06] L Bellatreche, A Giacometti, P Marcel, and H Mouloudi. Personalization of MDX Queries. In *XXIIèmes journées Bases de Données Avancées (BDA 06), Lille, France*, 2006.

[BK05] M Bouzeghoub and D Kostadinov. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. In *IIème COnférence en Recherche d'Infomations et Applications (CORIA 05), Grenoble, France*, pages 201–218, 2005.

[Bla00] M Blaschka. *FIESTA : A Framework for Schema Evolution in Multidimensional Databases*. Thèse de doctorat, Institut für Informatik des Technischen - Universität München, München, Germany, 2000.

[BMBT02] M Body, M Miquel, Y Bédard, and A Tchounikine. A Multidimensional and Multiversion Structure for OLAP Applications. In *Vth ACM International Workshop on Data Warehousing and OLAP (DOLAP 02), McLean, Virginia, USA*, pages 1–6. ACM Press, 2002.

[BMBT03] M Body, M Miquel, Y Bédard, and A Tchounikine. Handling Evolutions in Multidimensional Structures. In *XIXth International Conference on Data Engineering (ICDE 03), Bangalore, India*, pages 581–591. IEEE Computer Society, 2003.

[BRS00] K Bradley, R Rafter, and B Smyth. Case-Based User Profiling for Content Personalisation. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 00), Trento, Italy*, volume 1892 of *LNCS*, pages 62–72. Springer, 2000.

[BSH99] M Blaschka, C Sapia, and G Höfling. On Schema Evolution in Multidimensional Databases. In *Ist International Conference on Data Warehousing and Knowledge Discovery (DaWaK 99), Florence, Italy*, volume 1676 of *LNCS*, pages 153–164. Springer, 1999.

[BSSJ98] R Bliujute, S Saltenis, G Slivinskas, and C Jensen. Systematic Change Management in Dimensional Data Warehousing. In *IIIrd International Baltic Workshop on Databases and Information Systems, Riga, Latvia*, pages 27–41, 1998.

[BTM07] S Bimonte, A Tchounikine, and M Miquel. Spatial OLAP : Open Issues and a Web Based Prototype. In *Xth AGILE International Conference on Geographic Information Science, Aalborg, Denmark*, 2007.

[CCRT07] G Cabanac, M Chevalier, F Ravat, and O Teste. An Annotation Management System for Multidimensional Databases. In *IXth International Conference on Data Warehousing and Knowledge Discovery (DaWaK 07), Regensburg, Germany*, volume 4654 of *LNCS*, pages 89–98. Springer, 2007.

[CD97] S Chaudhuri and U Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec.*, 26(1) :65–74, 1997.

[CGFZ03] M Cherniack, E F Galvez, M J Franklin, and S B Zdonik. Profile-Driven Cache Management. In *XIXth International Conference on Data Engi-*

neering (ICDE 03), Bangalore, India, pages 645–656. IEEE Computer Society, 2003.

[CGL+07] Y W Choong, A Giacometti, D Laurent, P Marcel, E Negre, and N Spyratos. Context-based Exploitation of Data Warehouses. In *IIIèmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 07), Poitiers, France*, volume B-3 of *Revue des Nouvelles Technologies de l'Information*, pages 131–146. Cépaduès Editions, 2007.

[Cod93] E F Codd. Providing OLAP (On-Line Analytical Processing) to User-Analysts : an IT mandate. Technical report, E.F. Codd and Associates, 1993.

[CT98] L Cabibbo and R Torlone. A Logical Approach to Multidimensional Databases. In *VIth International Conference on Extending Database Technology (EDBT 98), Valencia, Spain*, volume 1377 of *LNCS*, pages 183–197. Springer, 1998.

[Cun01] C Da Cunha. Gestion des modifications. Master's thesis, Ecole Nationale Supérieure de Génie Industriel, Institut National Polytechnique de Grenoble, Septembre 2001. Mémoire de DEA.

[EK00] J Eder and C Koncilia. Evolution of Dimension Data in Temporal Data Warehouses. Technical report, University of Klagenfurt, Austria, 2000.

[EK01] J Eder and C Koncilia. Changes of Dimension Data in Temporal Data Warehouses. In *IIIrd International Conference on Data Warehousing and Knowledge Discovery (DaWaK 01), Munich, Germany*, volume 2114 of *LNCS*, pages 284–293. Springer, 2001.

[EV01] M Minuto Espil and A A Vaisman. Efficient Intensional Redefinition of Aggregation Hierarchies in Multidimensional Databases. In *IVth ACM International Workshop on Data Warehousing and OLAP (DOLAP 01), Atlanta, Georgia, USA*, pages 1–8. ACM Press, 2001.

[EVT02] M Minuto Espil, A A Vaisman, and L Terribile. Revising Data Cubes with Exceptions : a Rule-based Perspective. In *IVth International Workshop on Design and Management of Data Warehouses (DMDW 02), Toronto, Canada*, volume 58 of *CEUR Workshop Proceedings*, pages 72–81. CEUR-WS.org, 2002.

[FB05a] C Favre and F Bentayeb. Bitmap Index-based Decision Trees. In *XVth International Symposium on Methodologies for Intelligent Systems (ISMIS 05), New York, USA*, volume 3488 of *LNAI*, pages 65–73, Heidelberg, Germany, May 2005. Springer.

[FB05b] C Favre and F Bentayeb. Intégration efficace des arbres de décision dans les SGBD : utilisation des index bitmap. In *Vèmes Journées d'Extraction et de Gestion des Connaissances (EGC 05), Paris, France*, volume E-3 of *Revue des Nouvelles Technologies de l'Information*, pages 319–330. Cépaduès Editions, 2005.

[FBB06a] C Favre, F Bentayeb, and O Boussaid. A Knowledge-driven Data Warehouse Model for Analysis Evolution. In *XIIIth ISPE International Conference on Concurrent Engineering : Research and Applications (CE 06), Antibes, France*, volume 143 of *Frontiers in Artificial Intelligence and Applications*, pages 271–278. IOS Press, 2006.

[FBB06b] C Favre, F Bentayeb, and O Boussaid. A Rule-based Data Warehouse Model. In *XXIIIrd British National Conference on Databases (BNCOD 06), Belfast, Northern Ireland, UK*, volume 4042 of *LNCS*, pages 274–277, Heidelberg, Germany, 2006. Springer.

[FBB06c] C Favre, F Bentayeb, and O Boussaid. Evolution de schémas dans les entrepôts de données : modèle à base de règles. In *IIème journée francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 06), Versailles, France*, volume B-2 of *Revue des Nouvelles Technologies de l'Information*, pages 175–176. Cépaduès Editions, 2006.

[FBB06d] C Favre, F Bentayeb, and O Boussaid. Modèle d'entrepôt de données à base de règles. In *IIIème atelier Fouille de Données Complexes dans un processus d'extraction des connaissances (FDC 06), en conjonction avec les VIèmes journées francophones Extraction et de Gestion des Connaissances (EGC 06), Lille, France*, pages 39–50, 2006.

[FBB06e] C Favre, F Bentayeb, and O Boussaid. WEDriK : une plateforme pour des analyses personnalisées dans les entrepôts de données évolutifs. In *Ier Atelier Systèmes Décisionnels (ASD 06), in conjunction with the IXth Maghrebian Conference on Information Technologies (MCSEAI 06), Agadir, Maroc*, 2006.

[FBB07a] C Favre, F Bentayeb, and O Boussaid. Dimension Hierarchies Updates in Data Warehouses : a User-driven Approach. In *IXth International Conference on Enterprise Information Systems (ICEIS 07), Funchal, Madeira, Portugal*, volume Databases and Information Systems Integration, pages 206 – 211, 2007.

[FBB07b] C Favre, F Bentayeb, and O Boussaid. Evolution de modèle dans les entrepôts de données : existant et perspectives. In *IIIèmes journées fran-*

cophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 07), Poitiers, France, volume B-3 of *Revue des Nouvelles Technologies de l'Information*, pages 21–35. Cépaduès Editions, 2007.

[FBB07c] C Favre, F Bentayeb, and O Boussaid. Evolution et personnalisation des analyses dans les entrepôts de données : une approche orientée utilisateur. In *XXVème congrès INFormatique des ORganisations et Systèmes d'Information et de Décision (INFORSID 07), Perros-Guirec, France*, pages 308 – 323, 2007.

[FBB07d] C Favre, F Bentayeb, and O Boussaid. Evolution of Data Warehouses' Optimization : a Workload Perspective. In *IXth International Conference on Data Warehousing and Knowledge Discovery (DaWaK 07), Regensburg, Germany*, volume 4654 of *LNCS*, pages 13–22. Springer, 2007.

[FBB07e] C Favre, F Bentayeb, and O Boussaid. Intégration des connaissances utilisateurs pour des analyses personnalisées dans les entrepôts de données évolutifs. In *VIIèmes journées francophones Extraction et Gestion des Connaissances (EGC 07), Namur, Belgique*, volume E-9 of *Revue des Nouvelles Technologies de l'Information*, pages 217–222. Cépaduès Editions, 2007.

[FBB07f] C Favre, F Bentayeb, and O Boussaid. *A Survey of Data Warehouse Model Evolution*. Encyclopedia of Database Technologies and Applications, Second Edition (to appear). Idea Group Publishing, 2007.

[FBBN05] C Favre, F Bentayeb, O Boussaid, and N Nicoloyannis. Entreposage virtuel de demandes marketing : de l'acquisition des objets complexes à la capitalisation des connaissances. In *IIème atelier Fouille de Données Complexes dans un processus d'extraction des connaissances (FDC 05), en conjonction avec les Vèmes Journées d'Extraction et de Gestion des Connaissances (EGC 05), Paris, France*, pages 65–68, 2005.

[FS99] E. Franconi and U. Sattler. A Data Warehouse Conceptual Data Model for Multidimensional Aggregation : a Preliminary Report. *Italian Association for Artificial Intelligence AI*IA Notizie*, 1 :9–21, 1999.

[Gal02] J Gallo. Operations and Maintenance in a Data Warehouse Environment. DM Review Magazine, http ://www.dmreview.com/article_sub.cfm ?articleId=6118, 2002.

[GHS95] D Georgakopoulos, M F Hornick, and A P Sheth. An Overview of Workflow Management : From Process Modeling to Workflow Automa-

tion Infrastructure. *Distributed and Parallel Databases*, 3(2) :119–153, 1995.

[GLRV06] M Golfarelli, J Lechtenborger, S Rizzi, and G Vossen. Schema Versioning in Data Warehouses : Enabling Cross-Version Querying via Schema Augmentation. *Data and Knowledge Engineering*, 59(2) :435–459, 2006.

[GMR98a] M Golfarelli, D Maio, and S Rizzi. Conceptual Design of Data Warehouses from E/R Schemes. In *XXXIst Annual Hawaii International Conference on System Sciences (HICSS 98), Big Island, Hawaii, USA*, volume 7, pages 334–343, 1998.

[GMR98b] M Golfarelli, D Maio, and S Rizzi. The Dimensional Fact Model : A Conceptual Model for Data Warehouses. *International Journal of Cooperative Information Systems*, 7(2-3) :215–247, 1998.

[GS03] M Golfarelli and E Saltarelli. The Workload You Have, the Workload You Would Like. In *VIth ACM International Workshop on Data Warehousing and OLAP (DOLAP 03), New Orleans, Louisiana, USA*, pages 79–85. ACM Press, 2003.

[Han87] E N Hanson. A Performance Analysis of View Materialization Strategies. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 87), San Francisco, California, USA*, pages 440–453. ACM Press, 1987.

[HHNT86] J H Holland, K J Holyoak, R E Nisbett, and P R Thagard. *Induction : Processes of Inference, Learning, and Discovery*. MIT Press, 1986.

[HM01] C A Hurtado and A O Mendelzon. Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. In *VIIIth International Conference on Database Theory (ICDT 01), London, UK*, volume 1973 of *LNCS*, pages 375–389. Springer, 2001.

[HMV99] C A Hurtado, A O Mendelzon, and A A Vaisman. Maintaining Data Cubes under Dimension Updates. In *XVth International Conference on Data Engineering (ICDE 99), Sydney, Australia*, pages 346–355. IEEE Computer Society, 1999.

[IDG05] IDG. Entreprises et décisionnel : état des lieux, objectifs et perspectives. http ://www.decisio.info/Entreprises-et-decisionnel.html, Résultats de l'enquête dans le document http ://www.decisio.info/IMG/pdf/Enquete_CIO_SAS_230605.pdf, 2005.

[Inm96] W H Inmon. *Building the Data Warehouse*. John Wiley & Sons, 1996.

[Inm02] W H Inmon. *Building the Data Warehouse*. John Wiley & Sons, Third edition, 2002.

[JAC04] T S Jung, M S Ahn, and W S Cho. An Efficient OLAP Query Processing Technique Using Measure Attribute Indexes. In *Vth International Conference on Web Information Systems Engineering (WISE 04), Brisbane, Australia*, volume 3306 of *LNCS*, pages 218–228. Springer, 2004.

[JN03] P E Jouve and N Nicoloyannis. KEROUAC : An Algorithm for Clustering Categorical Data Sets with Practical Advantages. In *International Workshop on Data Mining for Actionable Knowledge (DMAK 03), in conjunction with the VIIth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 03), Seoul, Korea*, 2003.

[KI04] G Koutrika and Y Ioannidis. Personalization of Queries in Database Systems. In *XXth International Conference on Data Engineering (ICDE 04), Boston, Massachusetts, USA*, pages 597–608. IEEE Computer Society, 2004.

[Kim96] R Kimball. *The Data Warehouse Toolkit*. John Wiley & Sons, 1996.

[KLLC03] H J Kim, T H Lee, S G Lee, and J Chun. Automated Data Warehousing for Rule-Based CRM Systems. In *XIVth Australasian Database Conference on Database Technologies*, pages 67–73. Australian Computer Society, 2003.

[KR99] Y Kotidis and N Roussopoulos. DynaMat : A Dynamic View Management System for Data Warehouses. *SIGMOD Record*, 28(2) :371–382, 1999.

[KRRT00] R Kimball, L Reeves, M Ross, and W Thornthwaite. *Concevoir et déployer un data warehouse*. Eyrolles, 2000.

[LB03] S Lin and D E Brown. Criminal Incident Data Association Using the OLAP Technology. In *Ist NSF/NIJ Symposium in Intelligence and Security Informatics (ISI 03), Tucson, Arizona, USA*, volume 2665 of *LNCS*, pages 13–26. Springer, 2003.

[LBMS02] B List, R Bruckner, K Machaczek, and J Schiefer. A Comparison of Data Warehouse Development Methodologies Case Study of the Process Warehouse. In *XIIIth International Conference on Database and Expert Systems Applications (DEXA 02), Aix-en-Provence, France*, volume 2453 of *LNCS*, pages 203–215. Springer, 2002.

[Leh98] W Lehner. Modelling Large Scale OLAP Scenarios. In *VIth International Conference on Extending Database Technology (EDBT 98), Valencia, Spain*, volume 1377 of *LNCS*, pages 153–167. Springer, 1998.

[LMTS06] S Luján-Mora, J Trujillo, and I Y Song. A UML Profile for Multidimensional Modeling in Data Warehouses. *Data and Knowledge Engineering*, 59(3) :725–769, 2006.

[LRC06] M Lawrence and A Rau-Chaplin. Dynamic View Selection for OLAP. In *VIIIth International Conference on Data Warehousing and Knowledge Discovery (DaWaK 06), Krakow, Poland*, volume 4081 of *LNCS*, pages 33–44. Springer, 2006.

[MB00] A Ulbrich vom Ende M Boehnlein. Business Process Oriented Development of Data Warehouse Structures. In *Data Warehousing 2000 - Methoden Anwendungen, Friedrichshafen, Germany*. Physica Verlag, 2000.

[McQ67] J B McQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Vth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, USA*, volume 1, pages 281–297, 1967.

[Mic95] Microstrategy. The Case for Relational OLAP. White Paper, 1995.

[MJBN07] R Missaoui, G Jatteau, A Boujenoui, and S Naouali. *Towards Integrating Data Warehousing with Data Mining Techniques*, pages 253–276. Data Warehouses and OLAP : Concepts, Architectures and Solutions. Idea Group Publishing, 2007.

[MK00] D L Moody and M A R Kortink. From Enterprise Models to Dimensional Models : a Methodology for Data Warehouse and Data Mart Design. In *Design and Management of Data Warehouses*, page 5, 2000.

[MRMB07] R Ben Messaoud, S Loudcher Rabaséda, R Missaoui, and O Boussaid. OLEMAR : an On-Line Environment for Mining Association Rules in Multidimensional Data. *Advances in Data Warehousing and Mining, IGI Global*, 2, 2007.

[MT06] J N Mazón and J Trujillo. Enriching Data Warehouse Dimension Hierarchies by Using Semantic Relations. In *XXIIIrd British National Conference on Databases (BNCOD 06), Belfast, Northern Ireland, UK*, volume 4042 of *LNCS*, pages 278–281. Springer, 2006.

[MV00] A O Mendelzon and A A Vaisman. Temporal Queries in OLAP. In *XXVIth International Conference on Very Large Data Bases (VLDB 00), Cairo, Egypt*, pages 242–253. Morgan Kaufmann, 2000.

[MW04] T Morzy and R Wrembel. On Querying Versions of Multiversion Data Warehouse. In *VIIth ACM International Workshop on Data Warehousing and OLAP (DOLAP 04), Washington, Columbia, USA*, pages 92–101. ACM Press, 2004.

[MZ04] E Malinowski and E Zimányi. OLAP Hierarchies : A Conceptual Perspective. In *XVIth International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia*, volume 3084 of *LNCS*, pages 477–491. Springer, 2004.

[NSF+05] A Nabli, A Soussi, J Feki, H Ben-Abdallah, and F Gargouri. Towards an Automatic Data Mart Design. In *VIIth International Conference on Enterprise Information Systems (ICEIS 05), Miami, Florida, USA*, pages 226–231, 2005.

[PD02] C Phipps and K C Davis. Automating Data Warehouse Conceptual Schema Design and Evaluation. In *IVth International Workshop on Design and Management of Data Warehouses (DMDW 02), Toronto, Canada*, volume 58 of *CEUR Workshop Proceedings*, pages 23–32. CEUR-WS.org, 2002.

[PG99] A Pretschner and S Gauch. Ontology Based Personalized Search. In *XIth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 99), Chicago, Illinois, USA*, pages 391–398. IEEE Computer Society, 1999.

[PIR03] V Peralta, A Illarze, and R Ruggia. On the Applicability of Rules to Automate Data Warehouse Logical Design. In *Ist International Workshop on Decision Systems Engineering (DSE 03), in conjunction with the XVth International Conference on Advanced Information Systems Engineering (CAiSE 03), Klagenfurt/Velden, Austria*, volume 75 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[PKVV05] G Papastefanatos, K Kyzirakos, P Vassiliadis, and Y Vassiliou. Hecataeus : A Framework for Representing SQL Constructs as Graphs. In *XXth International Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD 05), in conjunction with the XVIIth International Conference on Advanced Information Systems Engineering (CAiSE 05), Oporto, Portugal*. FEUP, 2005.

[PLT07] M Plantevit, A Laurent, and M Teisseire. Extraction d'outliers dans des cubes de données : une aide à la navigation. In *IIIèmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 07), Poitiers, France*, volume B-3 of *Revue des Nouvelles Technologies de l'Information*, pages 113–130. Cépaduès Editions, 2007.

[Poe96] V Poe. *Building a Data Warehouse for Decision Support*. Prentice Hall, 1996.

[PR99] E Pourabbas and M Rafanelli. Characterization of Hierarchies and Some Operators in OLAP Environment. In *IInd ACM International Workshop on Data Warehousing and OLAP (DOLAP 99), Kansas City, Missouri, USA*, pages 54–59. ACM Press, 1999.

[PVV06] G Papastefanatos, P Vassiliadis, and Y Vassiliou. Adaptive Query Formulation to Handle Database Evolution. In *XVIIIth International Conference on Advanced Information Systems Engineering (CAiSE 06), CAiSE Forum, Luxembourg, Grand-Duchy of Luxembourg*, volume 231 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[RALT06] S Rizzi, A Abelló, J Lechtenbörger, and J Trujillo. Research in Data Warehouse Modeling and Design : Dead or Alive ? In *IXth ACM International Workshop on Data Darehousing and OLAP (DOLAP 06), Arlington, Virginia, USA*, pages 3–10. ACM Press, 2006.

[RB07] O Rakotoarivelo and F Bentayeb. Evolution de schéma par classification automatique pour les entrepôts de données. In *IIIèmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 07), Poitiers, France*, volume B-3 of *Revue des Nouvelles Technologies de l'Information*, pages 99–112. Cépaduès Editions, 2007.

[RG06] S Rizzi and M Golfarelli. What Time Is It in the Data Warehouse ? In *VIIIth International Conference on Data Warehousing and Knowledge Discovery (DaWaK 06), Krakow, Poland*, volume 4081 of *LNCS*, pages 134–144. Springer, 2006.

[Rod92] J F Roddick. Schema Evolution in Database Systems : an Annotated Bibliography. *SIGMOD Rec.*, 21(4) :35–40, 1992.

[RS03] S Rizzi and E Saltarelli. View Materialization vs. Indexing : Balancing Space Constraints in Data Warehouse Design. In *XVth International Conference on Advanced Information Systems Engineering (CAiSE 03), Klagenfurt, Austria*, volume 2681 of *LNCS*, pages 502–519. Springer, 2003.

[RTZ06] F Ravat, O Teste, and G Zurfluh. A Multiversion-Based Multidimensional Model. In *VIIIth International Conference on Data Warehousing and Knowledge Discovery (DaWaK 06), Krakow, Poland*, volume 4081 of *LNCS*, pages 65–74. Springer, 2006.

[RTZ07] F Ravat, O Teste, and G Zurfluh. Personnalisation de bases de données multidimensionnelles. In *XXVème Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID 07), Perros-Guirec, France*, pages 121–136, 2007.

[SA86] R Snodgrass and I Ahn. Temporal databases. *Computer*, 19(9) :35–41, 1986.

[SAM98] S Sarawagi, R Agrawal, and N Megiddo. Discovery-Driven Exploration of OLAP Data Cubes. In *VIth International Conference on Extending Database Technology (EDBT 98), Valencia, Spain*, volume 1377 of *LNCS*, pages 168–182. Springer, 1998.

[SFG05] A Soussi, J Feki, and F Gargouri. Approche semi-automatisée de conception de schémas multidimensionnels valides. In *Ière journée sur les Entrepôts de Données et l'Analyse en ligne (EDA 05), Lyon*, volume B-1 of *Revue des Nouvelles Technologies de l'Information*, pages 71–90. Cépaduès Editions, 2005.

[TB00] D Theodoratos and M Bouzeghoub. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In *IIIrd ACM International Workshop on Data Warehousing and OLAP (DOLAP 00), Washington, Columbia, USA*, pages 1–8. ACM Press, 2000.

[TBC99] N Tryfona, F Busborg, and J G Borch Christiansen. starER : A Conceptual Model for Data Warehouse Design. In *IId ACM International Workshop on Data Warehousing and OLAP (DOLAP 99), Kansas City, Missouri, USA*, pages 3–8. ACM Press, 1999.

[Tes00] O Teste. *Modélisation et manipulation d'entrepôts de données complexes et historisées*. Thèse de doctorat, Institut de Recherche en Informatique de Toulouse - Université Toulouse 3, Toulouse, France, 2000.

[TG89] A U Tansel and L Garnett. Nested Historical Relations. In *ACM International Conference on Management of Data (SIGMOD 89), Portland, Oregon, USA*, pages 284–294. ACM Press, 1989.

[TPGS01] J Trujillo, M Palomar, J Gomez, and I-Y Song. Designing Data Warehouses with OO Conceptual Models. *Computer*, 34(12) :66–75, 2001.

[TRC83] H Tardieu, A Rochfeld, and R Coletti. *La méthode Merise*. Les Editions d'Organisation, Paris, 1983.

[TS00] D Theodoratos and T Sellis. Incremental Design of a Data Warehouse. *Journal of Intelligent Information Systems*, 15(1) :7–27, 2000.

[TS02] T Thalhammer and M Schrefl. Realizing Active Data Warehouses with Off-the-Shelf Database Technology. *Software Practice and Experience*, 32(12) :1193–1222, 2002.

[TSM01] T Thalhammer, M Schrefl, and M Mohania. Active Data Warehouses : Complementing OLAP with Analysis Rules. *Data and Knowledge Engineering*, 39(3) :241–269, 2001.

[Val87] P Valduriez. Join Indices. *ACM Transactions on Database Systems (TODS)*, 12(2) :218–246, 1987.

[VPVS07] P Vassiliadis, G Papastefanatos, Y Vassiliou, and T Sellis. Management of the Evolution of Database-Centric Information Systems. In *Ist International Workshop on Database Preservation (PresDB 07), Edinburgh, Scotland, UK*, 2007.

Résumé

Cette thèse a été réalisée en collaboration avec l'établissement bancaire LCL-Le Crédit Lyonnais. Elle s'inscrit dans le domaine des entrepôts de données. Ces derniers constituent un élément fondamental de l'architecture décisionnelle, sur lesquels reposent des outils permettant de répondre à des besoins d'analyse. Or, l'émergence de nouveaux besoins d'analyse individuels fait apparaître la nécessité d'une personnalisation des analyses. Pour permettre cette personnalisation, nous proposons une solution basée sur une évolution du schéma de l'entrepôt guidée par les utilisateurs. Il s'agit en effet de recueillir les connaissances de l'utilisateur et de les intégrer dans l'entrepôt de données afin de créer de nouveaux axes d'analyse. Cette solution s'appuie sur la définition d'un modèle formel d'entrepôt de données évolutif, basé sur des règles «si-alors», que nous appelons règles d'agrégation, qui permettent de représenter les connaissances utilisateurs. Notre modèle d'entrepôt évolutif est soutenu par une architecture qui place l'utilisateur au cœur du processus d'évolution du schéma de l'entrepôt. Nous nous sommes par ailleurs intéressés à l'évaluation de la performance de notre modèle d'entrepôt de données évolutif. L'évaluation de performances se base généralement sur une charge (ensemble de requêtes). Dans le contexte évolutif dans lequel nous nous plaçons, nous proposons alors une méthode de mise à jour incrémentale d'une charge donnée en répercutant l'évolution de schéma subie par l'entrepôt. Pour valider nos différentes contributions, nous avons développé la plateforme WEDriK (data Warehouse Evolution Driven by Knowledge).

Mots-clés : Entrepôt de données, évolution de schéma, mise à jour de hiérarchie de dimension, personnalisation, utilisateur, règles d'agrégation, analyse en ligne (OLAP), évolution de charge.

Abstract

This thesis has been done in collaboration with the bank LCL-Le Credit Lyonnais. It deals with the evolution of analysis needs in data warehouses. Indeed the latter beeing a fundamental component in decision support systems it is important to place users at the center of the evolution process to better reflect individual analysis needs. Our proposal consists in guiding data warehouse schema evolution through the use of users' knowledge. These knowledge are used to create new analysis axis. Our solution is based on the definition of a formal model describing an evolving data warehouse. The model is composed of "if-then" rules that we call aggregation rules that support the personnalisation process. This process is also supported by an architecture composed of four steps : the knowledge acquisition gathers users' knowledge in the form of aggregation rules ; integration of these aggregation rules in the data warehouse ; the data warehouse schema evolution, and on-line analysis based on the new schema. The implementation of our proposal is done through a relational execution model that manages the overall evolution process. We also interested in the performance of our evolving data warehouse model. The performance evaluation is usually based on a workload (a set of queries), and since we are in an evolving context, we proposed an incremental update of the workload. To validate our contributions, we have implemented the WEDriK plateform (data Warehouse Evolution Driven by Knowledge).

Keywords : Data warehouses, schema evolution, dimension hierarchy updating, personnalisation, user, aggregation rules, on-line analysis (OLAP), workload evolution.