Conception et optimisation d'un entrepôt de données médicales

Maria Trinidad Serna Encinas, Michel Adiba

LSR-IMAG Laboratory, BP 72 38402 Saint-Martin d'Hères, France
http ://www-lsr.imag.fr/Les.Groupes/STORM/Storm2002/Francais/index.html.
{*Maria-Trinidad.Serna, Michel.Adiba*}*@imag.fr*

Résumé. Un entrepôt de données intègre des informations provenant des sources de données internes mais aussi externes à l'entreprise. L'ensemble des données est utilisé pour l'aide à la décision, ainsi la conception du modèle multidimensionnel et la sélection des vues à matérialiser représentent un processus complexe et délicat. Dans cet article, nous proposons la définition d'un modèle multidimensionnel qui se compose de trois classes : cube, dimension et hiérarchie. Nous proposons également un algorithme pour la sélection de l'ensemble optimal des vues à matérialiser. Notre algorithme utilise les paramètres de fréquence d'utilisation, de coût de calcul et de fréquence de mises à jour des relations de base. Nous avons eu l'opportunité de travailler dans le cadre d'un projet médical, ce qui nous a permis de vérifier et de valider notre proposition sur des données réelles.

1 Introduction

Les entrepôts de données sont apparus au début des années 1990 en réponse à la nécessité de rassembler toutes les informations de l'entreprise en une base de données unique destinée aux analystes et aux gestionnaires [Doucet et Gangarski, 2001]. Un processus de transformation et de nettoyage doit être appliqué aux données avant leur stockage dans l'entrepôt. L'ensemble des données, y compris leur historique, est utilisé pour l'aide à la décision [Kimball, 1996, Kimball et Ross, 2003].

La conception et la mise en oeuvre d'un entrepôt de données est une tâche complexe et délicate. Elle se compose de divers processus communément appelés extraction-intégration, organisation et interrogation. Pour l'extraction, nous devons analyser l'ensemble des sources de données internes et externes. Cette analyse sert aussi bien à la sélection de l'ensemble de données à stocker dans l'entrepôt, qu'à la sélection des outils requis pour l'extraction et la transformation de ces données avant leur stockage. Le deuxième processus consiste à organiser ces données à l'intérieur de l'entrepôt. Pour ce faire, nous devons concevoir le modèle multidimensionnel à utiliser ainsi que définir l'ensemble optimal de vues à matérialiser. Finalement, le dernier processus consiste à déterminer les outils nécessaires pour la visualisation de l'ensemble des données.

Nous avons participé au projet ADELEM (Aide à la DEcision Logistique Et Médicale). Ce projet nous a permis de valider notre proposition. Ainsi, dans ce travail, nous présentons une expérimentation sur les données médicales du projet.

Cet article se focalise sur le processus d'organisation. D'abord, la section 2 décrit le projet ADELEM et les sources de données. La section 3 présente la définition d'un modèle multidimensionnel. La section 4 montre la matérialisation de l'hypercube et un algorithme proposé pour la sélection de l'ensemble optimal de vues à matérialiser. Dans la section 5, nous analysons les travaux existants et finalement, la section 6 présente nos conclusions.

Ce travail est réalisé dans l'équipe NODS (Networked Open Database Services) `http ://www-lsr.imag.fr/Les.Groupes/STORM/Storm2002/Francais/index.html`.

2 Le projet ADELEM

Ce projet consiste en la mise au point d'outils logiciels nécessaires à l'aide à la décision logistique et médicale, dans le cadre des SROS (Schémas d'Organisation Sanitaire et Sociale) gérés par les ARH (Agences Régionales de l'Hospitalisation). Dans le système de soins d'une région, ces agences sont chargées de répartir de manière optimale les moyens médicaux (l'"offre") en fonction des besoins sanitaires (la "demande"). A partir des observations de santé constituant l'information primaire, essentiellement contenues dans les données PMSI (Programme de Médicalisation du Système d'Information des hôpitaux), il est nécessaire de mettre au point des outils logiciels d'analyse et de visualisation.

Ce projet a été abordé par une équipe interdisciplinaire : le personnel médical (Laboratoire TIMC IMAG, UMR CNRS 5525), le Laboratoire de Biométrie et Biologie évolutive, UMR CNRS 5558, qui fournit l'analyse statistique des données médicales et l'Organisation Mondiale de la Santé (basée en Suisse) qui a développé le logiciel HealthMapper pour la création et la manipulation des données géographiques. Notre équipe (Laboratoire LSR IMAG, UMR CNRS 5526) fournit le support sur les bases de données et les entrepôts.

Dans le cadre du projet ADELEM, nous nous proposons d'adapter notre savoir-faire au problème de la gestion de données médicales qui constituent un cadre applicatif particulièrement intéressant. En effet, ces données se trouvent réparties dans plusieurs sources qu'il faut, dans un premier temps, fédérer pour constituer un entrepôt de données pertinentes pour l'application visée. Cette étape est importante car elle doit non seulement identifier les sources, mais aussi déterminer comment extraire de ces sources les données désirées. Nous devons déterminer si les données doivent être extraites telles quelles ou bien s'il faut les traiter au préalable en leur appliquant des fonctions spécifiques comme des agrégats, des sommes, ... En plus, nous devons établir un mécanisme pour la gestion de l'évolution. Dans ce cas, il faut déterminer l'adaptation au niveau : de l'application d'extraction, des agrégats et de l'application d'analyse.

Cette problématique est générale à la constitution de tout entrepôt mais nous devons ici tenir compte de la nature particulière des données sur lesquelles portent l'étude : type, format, sémantique, confidentialité, degré de fiabilité et de confiance, informations manquantes ou incomplètes, ... En résumé, il s'agit de constituer un entrepôt qui contient des données pertinentes et de qualité sur lesquelles sera basé l'outil d'interrogation et le processus d'aide à la décision.

Les sources du projet ADELEM sont distribuées et hétérogènes, elles contiennent les données publiques concernant la santé (RSA "Résumés de Sortie Anonymes", RHA "Résumés Hebdomadaire Anonymes", CIM10 "Classification International des Maladies Version 10" et FINESS "Fichier National des Établissements Sanitaires et Sociaux") et les données démographiques (RP99 "Recensement Population") [Belot, 2002]. Ces dernières proviennent de l'Institut National de la Statistique et des Etudes Economiques. Le tableau 1 décrit les différentes propriétés des sources.

Source	**Propriétaire**	**Type de fichier**	**Mode d'obtention**	**Taille**	**Année**	**Mise à jour**
RSA	ARH	Texte (ASCII fixe) ou Access	Gratuit	11 Go	2000	Annuelle
RHA	ARH	Texte (ASCII fixe) ou Access	Gratuit	3 Go	2000	Annuelle
FINESS	MCASS	Texte (ASCII fixe) ou Access	Gratuit	2.24 Mo	2000	Annuelle
CIM10	OMS	Excel	-	1.34 Mo	1995	-
RP90/99	INSEE	DBF	Achat (625,04)	50.8 Mo	1999	Tous les 9 ans

Tab. 1 – Description des sources de données

Le tableau précédent indique des caractéristiques essentielles à prendre en compte lors des premières phases de la conception d'un entrepôt. Néanmoins, si nous reprenons les deux dernières propriétés, l'année et la mise à jour, pour les représenter dans le tableau 2 qui aura des données sur 10 ans, nous avons les caractéristiques des données historisées.

Source	**Taille**	**Optimisation du stockage**	**Rafraîchissement**	**Requêtes complexes**	**Agrégats**	**Evolution du schéma**
RSA	110 Go	Oui	Annuel	Oui	Oui	Oui
RHA	30 Go	Oui	Annuel	Oui	Oui	Oui
FINESS	22.4 Mo	Non	Annuel	Non	Non	Oui
CIM10	13.4 Mo	Non	-	Non	Non	Oui
RP90/99	508 Mo	Oui	Annuel	Non	Oui	Oui

Tab. 2 – Caractéristiques des sources de données historiques

L'analyse du tableau précédent, nous permet de remarquer les sources RSA et RHA. Leurs tailles nous obligent à rechercher des mécanismes d'optimisation pour le stockage (partitionnement, parallélisme, agrégats) et pour traiter des requêtes complexes. Nous proposons un algorithme pour la selection de l'ensemble optimal des vues à matérialiser en considérant leur coût de calcul et leur coût de stockage.

Une autre partie à gérer est l'évolution du schéma, pour prendre en compte des changements dans la structure logique de la source et qui peuvent avoir des répercussions sur la structure logique de notre entrepôt, voire l'affecter. Ceci nous oblige à proposer une conception adaptable de l'entrepôt de données pour mieux intégrer ces changements à notre schéma.

Par rapport à la gestion des aspects temporels de l'entrepôt de données, nous pouvons utiliser l'évolution de schémas ou les versions de schémas. Nous devons choisir la

technique à utiliser par rapport aux caractéristiques de l'application cible.

3 Conception d'un modèle multidimensionnel

Nous avons dit précédemment que la conception d'un modèle est une tâche complexe et délicate. Ainsi, nous considérons essentiel d'avoir un fondement formel qui nous permet d'aboutir à la conception d'un schéma multidimensionnel. Notre proposition repose sur la définition formelle d'un modèle multidimensionnel. Pour celui-ci, nous définissons une base de données multidimensionnelle, un cube, une dimension et une hiérarchie. Ceci nous a permis d'aboutir à la conception d'un schéma en constellation pour le projet ADELEM.

3.1 Définition d'un modèle multidimensionnel

Nous sommes partis de travaux existants [Benitez, 2002, Hurtado *et al.*, 1999a, Hurtado *et al.*, 1999b, Li et Wang, 1996, Quix, 1999] pour la définition d'un schéma et d'une instance. La plupart des travaux proposent un ensemble d'opérateurs pour l'évolution d'un système multidimensionnel. Dans [Grandi et Mandreoli, 2002], l'auteur propose un modèle formel des versions de schémas temporelles pour des bases de données à objets.

Nous faisons la différence entre une dimension et une hiérarchie. Ainsi, notre modèle se compose de quatre éléments essentiels : une base multidimensionnelle, un cube, une dimension et une hiérarchie. Nous donnons pour chacun la définition de schéma et d'instance. Nous supposons l'existence de :
$DOM = \text{dom(char)} \bigcup \text{dom(integer)} \bigcup \text{dom(float)} \bigcup \text{dom(decimal)} \bigcup \text{dom(date)} \bigcup \{t\}$ contenant le domaine de chaque type atomique plus la valeur distinguée t.
C = Ensemble de noms de cubes.
D = Ensemble de noms de dimensions.
H = Ensemble de noms de hiérarchies.
M = Ensemble de noms de mesures.
P = Ensemble de noms de propriétés.
L = Ensemble de noms de niveaux.
R = Ensemble de contraintes.

Nous associons à chaque cube $c \in C$ un ensemble de valeurs $\{a_1, ..., a_n\}$ tel que $\forall_{i=1,...n} a_i \in DOM$. De même pour chaque dimension $d \in D$, nous associons un ensemble de valeurs $\{b_1, ..., b_n\}$ tel que $\forall_{i=1,...n} b_i \in DOM$. Finalement, pour chaque niveau $l \in L$, nous associons un ensemble de valeurs $\{f_1, ..., f_n\}$ tel que $\forall_{i=1,...n} f_i \in DOM$.

Nous supposons l'existence des fonctions suivantes :
measuredom : $M \longrightarrow E(DOM)$[1], qui retourne l'ensemble des valeurs associées à une mesure.
propertydom : $P \longrightarrow E(DOM)$, qui retourne l'ensemble des valeurs associées à une propriété.
levelset : $H \longrightarrow E(DOM)$, qui retourne l'ensemble des membres associés à un niveau.

[1] L'ensemble $E(A)$ est l'ensemble des parties de A. $E(A) = \{X | X \subset A\}$.

Définition 3.1 : Schéma de base de données et d'instance multidimensionnelle.

Le schéma d'une base de données multidimensionnelle est un n-uplet SM = (C_s, D_s, H_s, R), où C_s est un ensemble de schémas de cubes, D_s est un ensemble de schémas de dimensions, H_s est un ensemble de schémas de hiérarchies et R est un ensemble de contraintes.

L'instance d'une base multidimensionnelle est un n-uplet IM = (C_i, D_i, H_i, R_i), où C_i est un ensemble d'instances de cubes tel que pour chaque instance $c_i \in C_i$, il existe un schéma $c_s \in C_s$ avec c_i conforme à c_s. D_i est un ensemble d'instances de dimensions tel que pour chaque instance $d_i \in D_i$, il existe un schéma $d_s \in D_s$ avec d_i conforme à d_s. H_i est un ensemble d'instances de hiérarchies tel que pour chaque instance $h_i \in H_i$, il existe un schéma $h_s \in H_s$ avec h_i conforme à h_s. Finalement, R_i est un ensemble d'instances de contraintes tel que chaque instance $r_i \in R_i$.

Définition 3.2 : Schéma et instance du cube.

Un schéma de cube est un n-uplet C_s = (c_n, M, D), où $c_n \in C$ est le nom du cube, M est un ensemble de mesures et D est un ensemble de dimensions.

Une instance de cube est un ensemble de cellules. Une cellule est un n-uplet I_c = (c_n, $\{v_1,...,v_k\}$, $\{d\}$), où c_n est le nom du cube, $\{v_1,...,v_k\}$ est l'ensemble de mesures tel que $\forall_{i=1,...n}\, v_i \in measuredom(m_i)$ avec $m_i \in M$ et $d \in D$ est l'ensemble de dimensions.

Par exemple, la figure 1 représente le schéma et l'instance du cube Ventes.

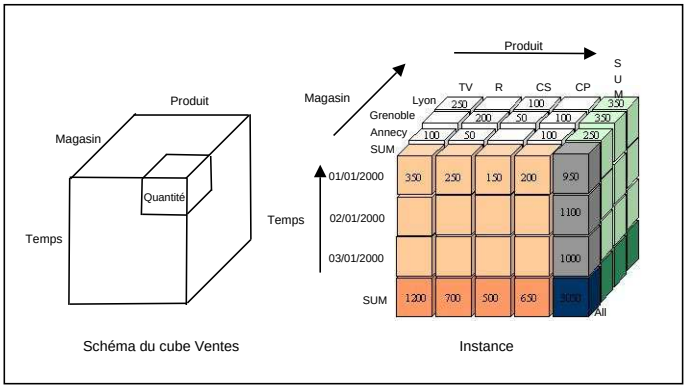

FIG. 1 – Schéma et une possible instance du Cube

La partie gauche de la figure 1 montre le schéma du cube de nom Ventes. L'ensemble {Temps, Magasin, Produit} sont les axes du cube et la mesure est Quantité. La partie droite montre une instance de ce cube. Par exemple, nous avons 100 Téléviseurs vendus dans le magasin d'Annecy le 1er janvier 2000.

Définition 3.3 : Schéma et instance de dimension.

Un schéma de dimension est un n-uplet D_s = (d_n, P, H), où $d_n \in D$ est le nom de la dimension, P est l'ensemble de propriétés de la dimension d_n et H est l'ensemble de hiérarchies.

Une instance de dimension est un n-uplet $I_d = (d_n, \{(v_1,...,v_i)\}, \{h\})$*, où* $d_n \in D$ *est le nom de la dimension,* $(v_1,...,v_i)$ *est un ensemble de propriétés tel que* $\forall_{j=1,...n} v_j \in$ *propertydom*(p_j) *avec* $p_j \in P$*. Enfin,* $h \in H$ *est l'ensemble de hiérarchies qui appartiennent à* d_n*.*

Par exemple, le schéma de la dimension Magasin est défini de la manière suivante :

d_n = Magasin

$\{p_1,...p_i\}$ = {Cle_Magasin, Libelle_Magasin }

h_n = H_Geo (*cf.* Définition 3.4)

Schéma et instance de hiérarchie.

Pour nous un schéma et une instance de hiérarchie peuvent prendre la forme d'un graphe séquentiel ou d'un graphe acyclique dirigé représentant les hiérarchies de niveaux. Chaque noeud du graphe contient un niveau et un arc entre deux noeuds représente une association entre les niveaux. Nous supposons l'existence de deux noeuds, nommés *base* et *sommet*, à partir desquels tous les autres noeuds peuvent être atteints directement ou par transitivité. Le noeud *sommet* contient le niveau $\top$ de la hiérarchie. La figure 2 présente un exemple de graphe séquentiel. Comme exemple de graphe acyclique dirigé, nous pouvons considérer le schéma de hiérarchie suivant : H_Temps = ({(Jour, Mois), (Mois, Semestre), (Semestre, Annee)}, {(Jour, Semaine), (Semaine, Annee)}), où la *base*=Jour et le *Sommet*=Annee. Dans notre modèle, nous considérons les caractéristiques suivantes :

1. Nous distinguons les termes de dimension et de hiérarchie. Ainsi, une dimension peut relier 0, une ou plusieurs hiérarchies. Pour une hiérarchie donnée, il y a une liaison directe avec la dimension associée dans le cube. Par exemple, si le cube `Ventes` comporte la dimension `Magasin` reliée à une commune, ce cube sera alors indirectement lié aux niveaux supérieurs à commune (Département, Région, $\top$). Ainsi, il sera possible d'effectuer des agrégats sur ces niveaux supérieurs.
2. Nous pouvons insérer un niveau de hiérarchie soit à côté de la hiérarchie déjà définie soit à l'intérieur. Pour faire cela, nous avons besoin de spécifier trois niveaux, le premier est le niveau à insérer, les deux derniers représentent les niveaux inférieur et supérieur à partir desquels le nouveau niveau sera placé.
3. Dans la définition du schéma de hiérarchie, nous considérons le cas où $l=\top$, c'est à dire quand nous avons seulement les niveaux *base* et *sommet*.

Nous donnons dans la suite la définition de schéma et d'instance de hiérarchie.

Définition 3.4 : Schéma et instance de hiérarchie.

Un schéma de hiérarchie est un n-uplet $H_s = (h_n, L, \prec)$*, où* $h_n \in H$ *est le nom de la hiérarchie, L est l'ensemble de niveaux à l'intérieur de la hiérarchie* h_n *et le symbole* $\prec$ *est une relation de dépendance entre les niveaux telle que sa fermeture transitive et réflexive* $\preceq^*$ *représente un ordre partiel dans L. Il existe un seul niveau* $l_\perp$ *tel que (i) si* $l \neq \top$*,* $\forall_{l \in L}\ l_\perp \preceq^* l \wedge \forall_{l \in L}\ l \preceq^* \top$ *et (ii) si* $l=\top$*,* $l_\perp \preceq^* \top$*. Un niveau* $l_j \in L$ *est un niveau immédiatement supérieur (relation de dépendance) de* $l_i \in L$ *si* $l_i \prec l_j$*.*

Une instance de hiérarchie organise les propriétés participant aux hiérarchies d'agrégation. Pour chaque arc dans un schéma de dimension, il existe une fonction de *rollup* associée.

Une instance de hiérarchie est un n-uplet $I_h = (\bigcup_{l_i \in L} levelset(l_i), RUP)$*, où RUP est un ensemble de fonctions* $rup_{l_i}^{l_j}$ *tel que (i)* $\forall (l_1, l_2) \in L$ *tels que* $l_1 \prec l_2$*, il existe une fonction* $rup_{l_1}^{l_2} : levelset(l_1) \longrightarrow levelset(l_2)$ *et (ii)* $\forall (l_1, l_2, l_3) \in L$ *tels que* $l_1 \prec l_2$ *et* $l_2 \prec l_3$*,* $ran(rup_{l_1}^{l_2}) \subseteq dom(rup_{l_2}^{l_3})$*.*

Par exemple, dans la figure 2, la partie gauche présente le schéma de la hiérarchie H_Geo. Il est défini de la manière suivante : h_n est le nom de la hiérarchie H_Geo, l'ensemble de propriétés (niveaux) à l'intérieur de la hiérarchie est {Commune, Département, Région, $\top$} et la relation $\prec$ est représentée par {(Commune, Département), (Département, Région), (Région, $\top$)}.

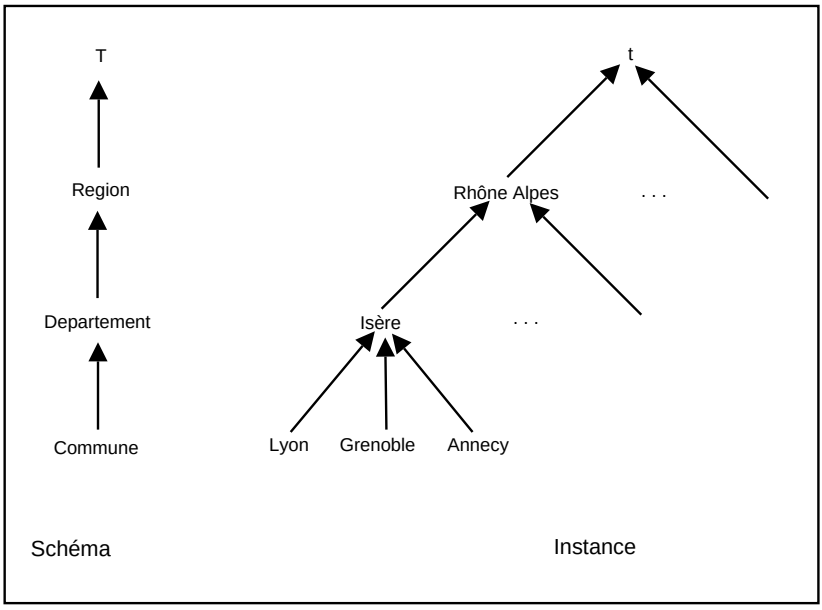

FIG. 2 – Schéma et une instance possible de la hiérarchie H_Geo

La partie droite de la figure 2 montre une instance possible de ce schéma. La fonction $rup_{Commune}^{Departement}$ met en relation les propriétés `Lyon, Grenoble` et `Annecy` du niveau `Commune` avec la propriété `Isère` du niveau `Departement`. Les propriétés du niveau `Region` sont mises en relation avec la propriété unique t du niveau $\top$.

3.2 Schéma en constellation pour ADELEM

La figure 3 montre le schéma en constellation avec trois relations de faits et leurs dimensions. La relation de faits `Prise_MCO` (Médecine-Chirurgie-Obstretique) a été conçue pour la gestion des séjours hospitaliers effectués dans la partie court séjour d'un établissement. La relation de faits `Population` manipule des informations démogra-

phiques. La troisième `Prise_SSR` (Soins de Suite et de Réadaptation) à différence de `Prise_MCO` enregistre des données correspondantes à des longs séjours.

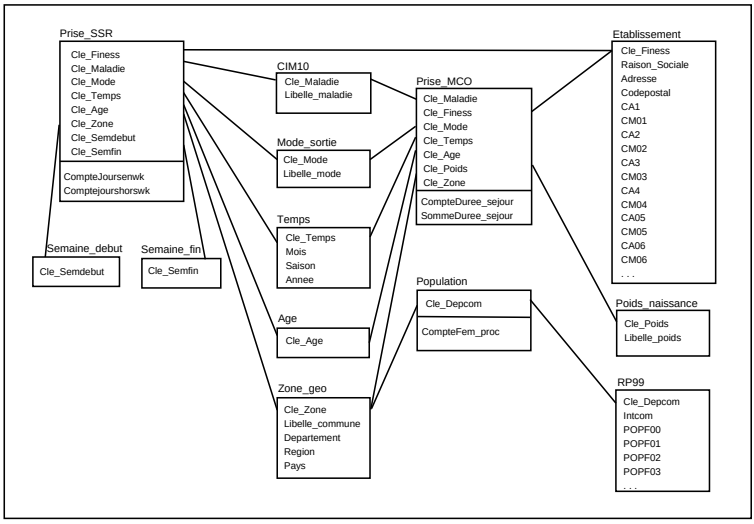

FIG. 3 – Schéma en Constellation pour ADELEM

Dans la suite, nous décrivons la première relation de faits `Prise_MCO` du schéma en constellation conçu :

Le schéma d'une base de données multidimensionnelle est un n-uplet SM = (C_s, D_s, H_s, R), où

C_s = {Prise_MCO, Population, Prise_SSR}

D_s = {Etablissement, CIM10, Mode_sortie, Temps, Age, Zone_geo, Poids_naissance, RP99, Semaine_debut, Semaine_fin}

H_s = {H_Geo, H_Temps}

R = {C_Cube, D_Dimension}

Le schéma d'un cube est un n-uplet C_s = (c_n, M, D), où

c_n = Prise_MCO

M = {CompteDuree_sejour, SommeDuree_sejour}

D = {Etablissement, CIM10, Mode_sortie, Temps, Age, Zone_geo, Poids_naissance}

Le schéma d'une dimension est un n-uplet D_s = (d_n, P, H), où

d_n = Etablissement

P = {Cle_Finess, Raison_Sociale, Adresse, Codepostal, CA1 .. CA7, CMO1 .. CMO7, Commune, Departement, Region, Pays}

H = {H_Geo}

Le schéma d'une hiérarchie est un n-uplet H_s = (h_n, L, $\prec$), où

h_n = H_Geo

L = {Commune, Departement, Region, Pays}

$\prec$ = {(Commune, Departement), (Departement, Region), (Region, Pays)}

4 Vues matérialisées

Pour notre cas expérimental, nous avons repris l'étoile `Prise_MCO` et seulement les dimensions : `Etablissement`, `CIM10`, `Temps`, `Mode_sortie`. En utilisant ce schéma nous avons construit un entrepôt comportant un échantillon de 10% des données réelles. Nous avons utilisé le système décisionnel d'Oracle9$_i$ Entreprise Edition Release 9.2.0.1.0 pour la création du schéma et pour la génération des vues matérialisées. Ceci nous a permis de connaître le coût de stockage (représenté par le nombre de n-uplets du résultat) de chaque vue et de pouvoir facilement déterminer leur coût de calcul (produit des cardinalités approximatives des relations de base).

4.1 Matérialisation de l'Hypercube

Dans les systèmes décisionnels, les utilisateurs sont intéressés par des requêtes du type : "le nombre de séjours de l'hôpital E1 par la maladie C1". Dans ce cas, la cellule (E1, C1, ALL[2], ALL, m1), contient la valeur de : "le nombre de séjours de l'hôpital E1 pour la maladie C1 pour tous les mois (ALL) et pour tous les modes de sortie (ALL)".

Nous pouvons calculer la valeur de la cellule (E1, C1, ALL, ALL, m1) comme la somme des valeurs des cellules de (E1, C1, T_1, M_1, m1), ..., (E1, C1, T_{Ntemps}, M_{Nmode}, m1), où T_{Ntemps} et M_{Nmode} représentent respectivement l'ensemble des mois et l'ensemble des modes de sortie. Toutes les cellules contenant la valeur ALL dans un de ses axes sont des cellules dépendantes. Ainsi, pour la sélection des vues à matérialiser, le problème se réduit à déterminer l'ensemble des cellules dépendantes à matérialiser. Etant donné que le coût de stockage[3] pour une matérialisation de toutes les vues est de 206781 n-uplets ($\sum_{i=1,...,n} \mathrm{CS}(\mathrm{v}_i)$, où $\mathrm{CS}(\mathrm{v}_i)$ est le coût de stockage des vues du tableau 3), nous avons un pourcentage de 74% ((206781-53799)*100/206781) de cellules dépendantes pour le schéma `Prise_MCO` construit.

Dans la suite, nous définissons l'ensemble des vues possibles à matérialiser. Le tableau 3 contient les 15 vues pour une matérialisation complète ainsi que leur coût de stockage et de calcul. La figure 4 représente l'hypercube sur 4 dimensions du schéma `Prise_MCO`, nous utilisons la notation : M pour million et K pour mille.

La figure 4 représente le treillis pour l'hypercube et elle contient le coût de stockage à droite de chaque noeud (vue) et le coût de calcul à gauche.

Relations de dépendance à l'intérieur de l'hypercube : Par exemple, V6 $\preceq$ V2 si et seulement si V6 peut être répondu par V2, donc V6 est dépendante de V2.
V2 : V6 $\preceq$ V2, V7 $\preceq$ V2, V9 $\preceq$ V2, V12 $\preceq$ V2, V13 $\preceq$ V2, V14 $\preceq$ V2, V16 $\preceq$ V2
V3 : V6 $\preceq$ V3, V8 $\preceq$ V3, V10 $\preceq$ V3, V12 $\preceq$ V3, V13 $\preceq$ V3, V15 $\preceq$ V3, V16 $\preceq$ V3
V4 : V7 $\preceq$ V4, V8 $\preceq$ V4, V11 $\preceq$ V4, V12 $\preceq$ V4, V14 $\preceq$ V4, V15 $\preceq$ V4, V16 $\preceq$ V4
V5 : V9 $\preceq$ V5, V10 $\preceq$ V5, V11 $\preceq$ V5, V13 $\preceq$ V5, V12 $\preceq$ V5, V15 $\preceq$ V5, V16 $\preceq$ V5
V6 : V12 $\preceq$ V6, V13 $\preceq$ V6, V16 $\preceq$ v6
V7 : V12 $\preceq$ V7, V14 $\preceq$ V7, V16 $\preceq$ V7
V8 : V12 $\preceq$ V8, V15 $\preceq$ V8, V16 $\preceq$ V8
V9 : V13 $\preceq$ V9, V14 $\preceq$ V9, V16 $\preceq$ V9

[2]Nous utilisons la recommandation d'ajouter la valeur ALL au domaine de la dimension T et M, présentée dans [Gray *et al.*, 1995]

[3]Le coût de stockage d'une requête est le nombre de n-uplets du résultat.

Vue	Relations	C. de Stockage	C. de Calcul
V1	Prise_MCO+Etablissement+CIM10+Temps+Mode_sortie	53799 n-uplets	90M
V2	Prise_MCO+Etablissement+CIM10+Temps	47133	90M
V3	Prise_MCO+Etablissement+CIM10+Mode_sortie	18456	90M
V4	Prise_MCO+Etablissement+Temps+Mode_sortie	603	61K
V5	Prise_MCO+CIM10+Temps+Mode_sortie	32918	346K
V6	Prise_MCO+Etablissement+CIM10	13973	90M
V7	Prise_MCO+Etablissement+Temps	184	60K
V8	Prise_MCO+Etablissement+Mode_sortie	58	25K
V9	Prise_MCO+CIM10+Temps	26058	216K
V10	Prise_MCO+CIM10+Mode_sortie	8186	90K
V11	Prise_MCO+Temps+Mode_sortie	48	60
V12	Prise_MCO+Etablissement	19	5K
V13	Prise_MCO+CIM10	5329	18K
V14	Prise_MCO+Temps	12	12
V15	Prise_MCO+Mode_sortie	4	5

TAB. 3 – Matérialisation complète de l'hypercube

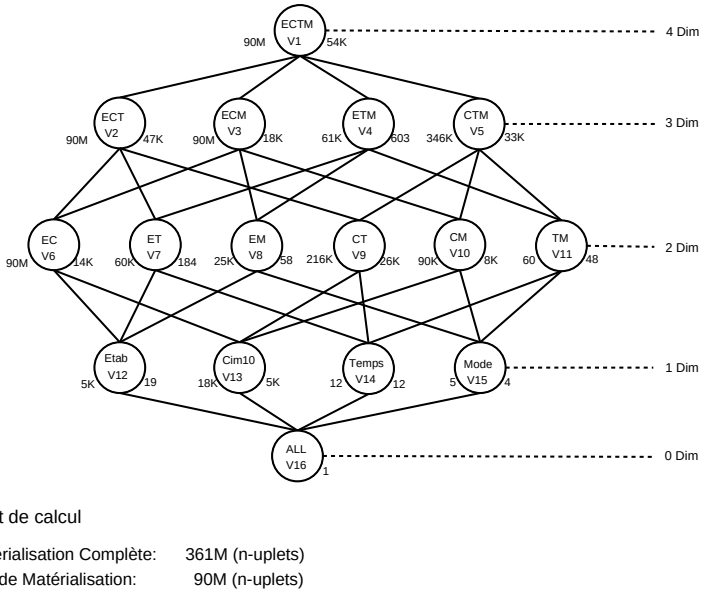

FIG. 4 – Hypercube avec le coût de calcul (à gauche) et le coût de stockage (à droite) de chaque noeud (vue)

V10 : V13 $\preceq$ V10, V15 $\preceq$ V10, V16 $\preceq$ V10
V11 : V14 $\preceq$ V11, V15 $\preceq$ V11, V16 $\preceq$ V11
V12 : V16 $\preceq$ V12
V14 : V16 $\preceq$ V14
V15 : V16 $\preceq$ V15

4.2 Sélection des vues à matérialiser

Dans la suite, nous formalisons la sélection des vues à matérialiser en utilisant l'algorithme Greedy [Harinarayan *et al.*, 1995].

Quelques notations :
n = Nombre de choix.
C(v) = Coût de stockage de la vue v (représenté par le nombre de n-uplets du résultat).
$\preceq$ = Relation de dépendance. Par exemple, Q2 $\preceq$ Q1 si et seulement si Q2 peut être répondu par Q1.
S = Ensemble de vues sélectionnées,
B(v, S) = Bénéfice de la vue v relative à S,
1) Pour chaque $w \preceq v$, définir la quantité B_w par :
a) Si u est la vue de coût inférieur dans S, tel que $w \preceq u$. Nous remarquons que l'ensemble S contient au moins la vue V1 (*top view*), et
b) Si C(v) < C(u), alors B_w = C(v) - C(u). Sinon $B_w = 0$.
2) Définir B(v, S) = $\sum_{w \preceq v} B_w$.
La figure 5 décrit l'algorithme Greedy pour une sélection de n vues à matérialiser.

```
S = {top view};
for i = 1 to n do begin
    select that view v not in S such that
        B(v, S) is maximized;
    S = S union {v};
end;
resulting S in the greedy selection;
```

FIG. 5 – Algorithme Greedy [Harinarayan *et al.*, 1995]

Nous utilisons l'hypercube de la fig 4 pour l'application de l'algorithme. La table 4 décrit cette application avec n égal à 7, pour déterminer l'ensemble de vues à matérialiser.

Pour le premier choix la valeur de u est égal à 54K (le coût de V1, car elle représente la vue avec le coût de stockage minimum dans l'ensemble S), pour le reste elle est représentée par 603 (V4), car elle est toujours, au moins jusqu'à la 7ème itération, la vue d'un coût de stockage minimale à l'intérieur de S. Dans la première itération, le choix est la vue V4, car elle représente le bénéfice le plus élevé (53K*8=424K), le bénéfice est calculé de la manière suivante : coût(V4) - coût(V1) = 53K multiplié par 8, le nombre de vues dépendantes de V4 (V4, V7, V8, V11, V12, V14, V15 et V16).

Le second choix est la vue V3 avec 144K (36K*4 (V3, V6, V10 et V13)), les autres vues dépendantes de V3 ne sont pas prises en compte, car elles tiennent un bénéfice plus

Vue	1ère Choix	2ème	3ème	4ème	5ème	6ème	7ème
V2	7K*8=56K	7K*4=28K	7K*2=14K	7K*1=7K	7K	7K	-
V3	36K*8=288K	36K*4=144K	-	-	-	-	-
V4	53K*8=424K	-	-	-	-	-	-
V5	21K*8=168K	21K*4=84K	21K*2=42K	-	-	-	-
V6	40K*4=160K	40K*2=80K	4K*2=8K	4K*2=8K	4K	4K	4K
V7	54K*4=216K	419*4=2K	419*4=2K	419*4=2K	2K	2K	2K
V8	54K*4=216K	545*4=2K	545*4=2K	545*4=2K	2K	2K	2K
V9	28K*4=112K	28K*2=56K	28K*1=28K	7K*1=7K	7K	-	-
V10	46K*4=184K	46K*2=92K	10K*2=20K	10K*2=20K	-	-	-
V11	54K*4=216K	555*4=2K	555*4=2K	555*4=2K	2K	2K	2K
V12	54K*2=108K	584*2=1K	584*2=1K	584*2=1K	1K	1K	1K
V13	49K*2=98K	49K*1=49K	13K*1=13K	13K*1=13K	3K	3K	3K
V14	54K*2=108K	591*2=1K	591*2=1K	591*2=1K	1K	1K	1K
V15	54K*2=108K	599*2=1K	599*2=1K	599*2=1K	1K	1K	1K
V16	54K*1=54K	1K	1K	1K	1K	1K	1K
{V1}	S+=V4	S+=V3	S+=V5	S+=V10	S+=V9	S+=V2	S+=V6

TAB. 4 – Application de l'algorithme Greedy aux données ADELEM

élevé avec la vue V4. Finalement, la dernière ligne contient le résultat, elle représente l'ensemble S={V1, V4, V3, V5, V10, V9, V2, V6}.

Dans la figure 6, nous avons dessiné un graphique qui montre l'application de l'algorithme Greedy par rapport aux coûts de calcul et de stockage. Nous considérons seulement les 7 premiers résultats, même si à partir de la 6ème itération, nous n'avons aucun avantage pour la matérialisation. Par exemple, la vue V2, résultat de la 6ème itération, a pour coût de stockage 47K et un coût de calcul de 90M. Si nous faisons la comparaison de ces résultats, par rapport à la vue V1, que nous sommes censés matérialiser, nous nous apercevons que le bénéfice est presque minimum, mais que les coûts de leur matérialisation sont doublés.

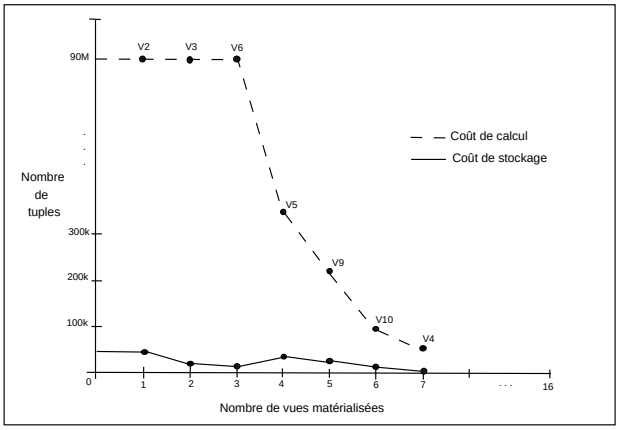

FIG. 6 – Ensemble ordonné des vues sélectionnées

Le graphique de la figure 6 contient les coûts de calcul et de stockage dans l'axe

Y et les sept premières vues de la table 4 ordonnées (par rapport à leur coût de calcul) sur l'axe X. L'algorithme Greedy est simple et efficace, néanmoins, dans notre expérimentation, à partir du 6ème choix, l'algorithme sélectionne les vues les plus coûteuses. Ceci nous motive pour proposer un mécanisme pour une sélection plus fiable. Nous proposons d'inclure les paramètres de la fréquence d'utilisation, le coût de calcul et la fréquence de mises à jour sur les relations de base. Dans la suite, nous décrivons notre proposition en détail.

4.3 Algorithme proposé pour la sélection des vues à matérialiser

Dans notre algorithme, les relations dépendantes jouent un rôle essentiel, ainsi, nous considérons que le nombre de relations dépendantes d'une vue représente un paramètre initial pour la fréquence d'utilisation, ainsi que pour le coût de calcul. Pour le premier, nous multiplions le bénéfice optimal, donné par l'algorithme Greedy, par le nombre total des relations dépendantes, car pour nous, la fréquence d'utilisation augmente en proportion directe du nombre des relations dépendantes. Pour le coût de calcul, nous considérons qu'il diminue par rapport au nombre de relations dépendantes, car ce coût est partagé entre ces vues, ainsi, nous divisons le coût de calcul par le nombre total des relations dépendantes.

Notre algorithme considère aussi la fréquence des mises à jour des relations de base. Dans ce cas, nous avons besoin du nombre d'éléments à l'intérieur de la vue qui peuvent subir des changements. Sur l'hypercube du schéma `Prise_MCO` (*cf.* figure 4), nous avons 36 éléments pouvant évoluer (32 propriétés et 4 dimensions). La relation Etablissement contient 24 propriétés, CIM10 et Mode_sortie contiennent 2 propriétés et Temps en contient 4 (*cf.* paragraphe 3.2). Ainsi, par exemple, considérons la vue V2. Elle se compose des relations Etablissement, CIM10 et Temps. Supposons que nous avons une probabilité de changement de 20%, notre calcul pour la fréquence des mises à jour de V2 est (33*100/36*.20), où 33 est le nombre d'éléments de la vue V2. Dans la suite, nous décrivons l'algorithme proposé.

Quelques notations :

CC = Coût de calcul (produit des cardinalités approximatives des relations de base) divisé par le nombre de relations dépendantes. Par exemple, le CC(v) si v=V2 (vue ECT, *cf.* Figure 6) est : (((5K*18K) + (14K*12)=90M)/8)=11M, où 8 est le nombre total des relations dépendantes de V2.

PC = la fréquence des mises à jour sur les relations de base tel que PC(v) = $|v|$ multipliée par le coût de calcul. Par exemple, si v=V2 et si la fréquence de mise à jour est de 20%, alors PC(v)=(3300/36*.20)*(11M)=2M.

V = Ensemble de vues.

S = Ensemble de vues sélectionnées.

w = Nombre de choix.

fq(v) = Fréquence d'utilisation (nombre total des relations dépendantes de la vue v).

v = Vue sélectionnée.

vo = C(vue $\top$ "*top view*").

La figure 7 montre l'algorithme proposé incluant la fréquence d'utilisation, le coût de calcul et la probabilité de changement.

```
S = {vue T}; vo = C(vue T); fq = 1; Bg = 0; t = 0; n = |V|;
for y = 1 to w do begin                                  //nombre de choix
    for x = 2 to n do begin                              //parcours de l'hypercube pour chaque vue
        i = x + 1;
        repeat                                           //calculer la fréquence et le bénéfice de l'algorithme Greedy
            if vi dépend de vx
                if {vx} appartient à S
                  vo=C(vx)
                  else vo=C(vue T)
                endif
                if vi dépend de v tel que {v} appartient à S
                  vo=C(v)
                endif
                fq = fq + 1;
                if C(vx) < vo
                  B(vi) = C(vx) - vo
                  else B(vi) = 0
                endif
                Bg = Bg + B(vi)                          //bénéfice Greedy
            endif
            i = ++;
        until i = n;
        Bg = Bg + (C(vx) - vo);                          //bénéfice de vx
        B(vx, S) = (Bg * fq) - (CC(vx)/fq + PC(vx));     //bénéfice incluant la fréquence, le coût de calcul et la prob. chang.
        if B(vx, S) > t
          t = B(vx, S);                                  //bénéfice maximal
          v = vx;
        endif
    endfor
    t=0;
    S = S U {v};
endfor
résultat S
```

FIG. 7 – Algorithme proposé

Le tableau 5 montre les résultats de l'application de notre algorithme pour les 7 premiers choix, en considérant 20% de fréquence de mises à jour sur les relations de base.

Vue	1ère Choix	2ème	3ème	4ème	5ème	6ème	7ème
V2	(448K-(11M*1.18))=-13M	-13M	-13M	-13M	-13M	-13M	-13M
V3	(2M-13M)=-11M	-12M	-12M	-12M	-12M	-12M	-12M
V4	(3392K-9K)=3M	-	-	-	-	-	-
V5	(1344K-46K)=1298K	626K	-	-	-	-	-
V6	(640K-26M)=-25M	-26M	-26M	-26M	-26M	-26M	-26M
V7	(864K-18K)=845K	-11K	-11K	-11K	-11K	-11K	-11K
V8	(864K-7K)=857K	2K	2K	2K	2K	-	-
V9	(448K-56K)=392K	168K	0K	-28K	-28K	-28K	-28K
V10	(736K-24K)=713K	345K	177K	-	-	-	-
V11	(864K-62)=864K	9K	9K	9K	-	-	-
V12	(216K-3K)=213K	-1K	-1K	-1K	-1K	-3K	-3K
V13	(196-9K)=187K	89K	47K	-3K	-3K	-3K	-3K
V14	(54K*2*2=216K)=216K	2K	2K	2K	30	30	30
V15	(54K*2*2=216K)=216K	2K	2K	2K	41	41	-
{V1}	S+=V4	S+=V5	S+=V10	S+=V11	S+=V8	S+=V15	S+=V14

Tab. 5 – Application de notre algorithme aux données ADELEM

Pour le premier choix la valeur de *vo* est égale à 54K, le coût de V1, car elle représente la vue avec le coût de stockage minimum dans l'ensemble S. Pour le deuxième et le troisième elle est représentée par 603 (le coût de V4), car c'est la vue ayant un coût de stockage inférieur à l'intérieur de S. Dans la première itération, le choix est la vue V4, car elle représente le plus haut gain ((((53K*8)*8)=3392K)-((61K/8)*1.18=9K)=3M). Le gain est calculé de la manière suivante : coût(V4) - coût(V1) = (53K*8) représente le gain de Greedy, nous multiplions ce gain par le nombre de relations dépendantes de V4 (V4, V7, V8, V11, V12, V14, V15 et V16). Ensuite, nous le soustrayons le coût de calcul divisé par le nombre de relations dépendantes et multiplié par la fréquence des mises à jour.

Le second choix est la vue V5 avec 626K (((21K*4*8)=672K) - ((346K/8) * 1.06 = 46K)=626K), les vues dépendantes sont (V3, V6, V10 et V13), les autres vues dépendantes de V3 ne sont pas prises en compte, car elles ont un gain inférieur à celui de la vue V4. Néanmoins, nous multiplions le résultat par le nombre total des vues dépendantes et nous divisons le coût de calcul par ce nombre total. Finalement, la dernière ligne contient le résultat, elle représente l'ensemble S={V1, V4, V5, V10, V11, V8, V15, V14}.

La figure 8 montre l'application de notre algorithme par rapport aux coûts de calcul et de stockage. Nous considérons les 7 premiers résultats. Si nous prenons comme référence la figure 4, nous remarquons que la sélection est fait de la manière suivante : du deuxième niveau (3 Dim), notre algorithme sélectionne les deux vues de bénéfice optimal. Pour le troisième niveau (2 Dim), il sélectionne les trois meilleurs et finalement pour le dernier niveau (1 Dim), il sélectionne les deux premières vues optimales.

Le graphique contient les coûts de calcul et de stockage sur l'axe des Y et les sept premières vues de la table 5 ordonnées (par rapport à leur coût de calcul) sur l'axe X. Nous constatons, qu'à la différence de l'algorithme Greedy, notre proposition donne de meilleurs résultats dans notre cas expérimental.

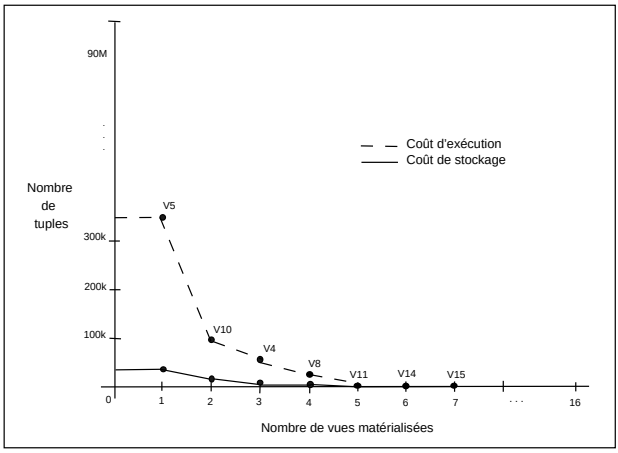

FIG. 8 – Ensemble ordonné des vues sélectionnées

5 Travaux analogues

Pour la sélection des vues à matérialiser, nous proposons un algorithme qui prend en compte la fréquence d'utilisation, le coût de calcul ainsi que la fréquence de mises à jour. Nous nous sommes inspiré des travaux de recherche [Gupta et Mumick, 1995, Levy *et al.*, 1995, Chan *et al.*, 1999, Yang *et al.*, 1997, Baril et Bellahsène, 2003] qui utilisent un modèle de coût, comme celui de Greedy, mais qui a été adapté avec l'inclusion de certains paramètres. Par exemple, dans [Zhang *et al.*, 2001], les auteurs proposent un MVPP (Multiple View Processing Plan) pour une sélection des vues à matérialiser, leur algorithme utilise la fréquence d'utilisation et la fréquence de mises à jour.

Dans [Baril et Bellahsène, 2003], pour déterminer l'ensemble des vues à matérialiser offrant un plus grand bénéfice, les auteurs proposent également un MVMG (Multi View Materialization Graph). Ils définissent un algorithme général pour calculer le bénéfice global de la vue par rapport au coût de la requête et au coût de maintenance. Néanmoins, à la différence de MVPP, ils différencient le coût d'évaluation d'une requête par rapport au type d'opération (*Select, Project* et *Join*). Ils utilisent aussi le paramètre de fréquence de la requête ainsi que le coût de maintenance basé sur les opérateurs *Add* et *Delete*.

Notre algorithme se base sur le modèle de coût proposé par Greedy, néanmoins, dans notre cas, le nombre total des relations dépendantes joue un rôle essentiel, car nous considérons que le bénéfice d'une vue augmente en rapport avec le nombre de relations qui en dépendent. Pour une raison similaire, le coût de calcul de la vue diminue en rapport avec le nombre total de relations dépendantes d'elle. La faiblesse de notre algorithme est l'absence du coût d'évaluation d'une requête par rapport au type d'opération (*Select, Project* ou *Join*). Nous ne prenons pas en compte les restrictions éventuelles sur l'espace de stockage.

6 Conclusions

Pour la conception d'un modèle multidimensionnel, nous avons spécifié trois classes : Cube, Dimension et Hiérarchie. Nous avons donné une définition formelle pour chaque classe et leur instance, ainsi qu'une définition d'une base de données multidimensionnelle et son instance. Ceci nous a permis d'aboutir à la conception d'un schéma en constellation pour la construction d'un entrepôt multidimensionnel de données médicales. Nous avons utilisé les informations concernant l'ensemble de sources de données réelles et des indicateurs du projet ADELEM pour prouver et vérifier le modèle proposé.

Par rapport à la sélection des vues à matérialiser, nous avons implanté d'abord le schéma en étoile `Prise_MCO`. Nous avons utilisé ce schéma pour aboutir à la matérialisation de l'hypercube, ce qui nous a permis de connaître le coût de stockage des vues et de pouvoir déterminer leur coût de calcul. Nous avons utilisé l'algorithme Greedy pour la sélection de l'ensemble optimal de vues à matérialiser, néanmoins, dans notre expérimentation, à partir du 6ème choix, l'algorithme sélectionne les vues plus coûteuses. Ceci nous a motivé à proposer un mécanisme pour une sélection plus fiable. Notre algorithme considère les paramètres de fréquence d'utilisation, de coût de calcul et de fréquence de mises à jour sur les relations de base. Nous avons prouvé l'algorithme proposé sur des données médicales réelles et nous avons constaté que notre proposition donne de meilleurs résultats pour notre cas d'expérimentation.

Remerciements

Nous remercions les referees anonymes pour leurs commentaires détaillés sur la première version de l'article.

Références

[Baril et Bellahsène, 2003] X. Baril et Z. Bellahsène. Selection of Materialized Views : A Cost-Based Approach. In *CAiSE*, pages 665–680, Bethesda, Maryland, USA, 2003. J. Eder and M. Missikoff.

[Belot, 2002] A. Belot. Développement d'un logiciel de cartographie de l'offre et de la demande de soins hospitaliers en France. Technical Report Projet ADELEM-SIGT, Université Claude Bernard Lyon 1, 2002.

[Benitez, 2002] E. Benitez. *Infrastructure adaptable pour l'évolution des entrepôts de données.* PhD thesis, Université Joseph Fourier, Grenoble, France, Septembre 2002.

[Chan *et al.*, 1999] G. Chan, Q. Li, et L. Feng. Design and Selection of Materialized Views in Data Warehousing : A Case Study. In *ACM International Workshop on Data Warehousing and OLAP*, 1999.

[Doucet et Gangarski, 2001] A. Doucet et S. Gangarski. *Entrepôts de données et Bases de Données Multidimensionnelles, Chapitre 12 du livre : Bases de Données et Internet, Modèles, langages et systèmes.* A. Doucet, G. Jomier (éditeurs). Editions Hermès, 2001.

[Grandi et Mandreoli, 2002] Fabio Grandi et Federica Mandreoli. A Formal Model for Temporal Schema Versioning in Object-Oriented Databases. Technical Report A TIME CENTER, TR-68, 2002.

[Gray *et al.*, 1995] J. Gray, A. Bosworth, A. Layman, et H. Pirahesh. Data Cube : A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals. Technical Report Microsoft No. MSR-TR-95-22, 1995.

[Gupta et Mumick, 1995] A. Gupta et I. Mumick. Maintenance of Materialized Views : Problems, Techniques and Applications. *IEEE Quarterly Bulletin on Data Engineering ; Special Issue on Materialized Views and Data Warehousing*, 18(2) :3–18, 1995.

[Harinarayan *et al.*, 1995] V. Harinarayan, A. Rajaraman, et J. Ullman. Implementing Data Cubes Efficiently. A full version. Technical Report IRI-92-23406 and DAAH04-95-1-0192 and F33615-93-1-1339, 1995.

[Hurtado *et al.*, 1999a] Carlos A. Hurtado, Alberto O. Mendelzon, et Alejandro A. Vaisman. Maintaining Data Cubes under Dimension Updates. In *ICDE*, pages 346–355, 1999.

[Hurtado *et al.*, 1999b] Carlos A. Hurtado, Alberto O. Mendelzon, et Alejandro A. Vaisman. Updating OLAP dimensions. In *DOLAP '99 : Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*, pages 60–66. ACM Press, 1999.

[Kimball, 1996] R. Kimball. *Entrepôts de données, Guide pratique du concepteur de data warehouse*. John Wiley and Sons, Inc., 1996.

[Kimball et Ross, 2003] R. Kimball et M. Ross. *Entrepôts de données, Guide pratique de modélisation dimensionnelle*. Vuibert, Paris, 2003.

[Levy *et al.*, 1995] A. Levy, A. Mendelzon, Y. Sagiv, et D. Srivastava. Answering Queries Using Views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 95–104, San Jose, Calif., 1995.

[Li et Wang, 1996] Chang Li et Xiaoyang Sean Wang. A Data Model for Supporting On-Line Analytical Processing. In *CIKM*, pages 81–88, 1996.

[Quix, 1999] Christoph Quix. Repository Support for Data Warehouse Evolution. In *DMDW*, page 4, 1999.

[Yang *et al.*, 1997] J. Yang, K. Karlapalem, et Q. Li. Algorithms for Materialized View Design in Data Warehousing Environment. In *The VLDB Journal*, pages 136–145, 1997.

[Zhang *et al.*, 2001] C. Zhang, X. Yao, et J. Yang. An evolutionary approach to materialized view selection in a data warehouse environment. *IEEE Trans. Systems, Man, Cybernetics, PART C*, 31 :282–294, September 2001.

Summary

A data warehouse integrates information from several internal and external data sources in an enterprise or an organization. Data sets are used for decision making, thus multidimensional model design and materialized view selection constitute a complex

and delicate process. In this article, we designed a multidimensional model with three classes : cube, dimension and hierarchy. Also, we propose an algorithm for an optimal selection of materialized views based upon different frequencies (queries, updates on database relations). We had the opportunity to apply our proposal to medical data. This allowed us to verify and validate our approach.