Construction et Maintenance des Entrepôts de Données Hétérogènes

Sana Hamdoun

LIPN Université Paris 13, 99 avenue Jean Baptiste Clément, 93430 Villetaneuse, France sh@lipn.univ-paris13.fr

Faouzi Boufarès

LIPN Université Paris 13, 99 avenue Jean Baptiste Clément, 93430 Villetaneuse, France boufares@lipn.univ-paris13.fr

Mohamed Badri

CRIP5 Université Paris 5, 45 rue des Saints Pères, 75270 Paris Cedex 06, France badri@univ-paris5.fr

Résumé

La construction de gros volumes de données afin de disposer, à tout moment, d'outils d'aide à la décision est devenue, de nos jours, un sujet très important dans le monde industriel. L'abondance des informations et leur hétérogénéité complique la tâche d'intégration de données et leur maintenance au moindre coût. Nous proposons dans cet article un outil d'intégration de données hétérogènes structurées et semi-structurées. Notre méthode est basée sur les liens de synonymie et d'inclusion qui peuvent exister entre les informations à travers les bases de données concernées. Nous proposons, par ailleurs, une procédure de maintenance pour pallier l'insuffisance d'Oracle.

Abstract

This work describes the construction of a data warehouse by the integration of heterogeneous relational and object-relational data. In fact, developing intelligent tools for the integration of information extracted from multiple heterogeneous sources is a challenging issue to effectively exploit the numerous sources available in global information systems. Due to the heterogeneity of the sources, various languages of interrogation and different data models are used for the warehouses. Thus the construction of the latter can be made in several manners. Our work is based on the extraction of the inter-schema relationships between the sources. Related to this, a global schema is generated and the views of the data warehouse are constructed. A maintenance procedure is also presented to mitigate the insufficiency of Oracle.

Mots-clés

sources de données hétérogènes, intégration, maintenance, bases de données, entrepôts de données, relationnel, objet

Keywords

database, data warehouse, heterogeneous structures and data, integration

1. Introduction

L'environnement informationnel actuel se caractérise par des données fortement distribuées. Ces données surabondantes sont généralement éparpillées, puisqu'il existe souvent de multiples systèmes conçus chacun pour être efficace pour les fonctions pour lesquelles il est spécialisé. Ces données sont également **hétérogènes**. En effet, avec l'apparition de l'Internet et le développement des différentes représentations et formats des documents, les données peuvent être classées en plusieurs catégories : **structurées** (données relationnelles, données objet), **semi-structurées** (HTML, XML, graphes) ou même **non structurées** (texte, images, son). Dans un tel contexte, le besoin d'intégration se fait de plus en plus sentir. Cependant, pour répondre à ce besoin, le développement des applications d'intégration (telles que pour un traitement élaboré de données, pour la construction des entrepôts de données ou des systèmes d'aide à la décision) se voit contraint de composer avec la répartition des sources, l'hétérogénéité de leurs structures et la complexité des données.

Afin d'alimenter les processus d'aide à la décision, notre travail consiste, d'une part, à intégrer des données hétérogènes pour la construction des entrepôts de données (ED) et, d'autre part, à les maintenir.

Plusieurs travaux sont menés par plusieurs équipes afin de construire des entrepôts XML (Nassis, Rajugan et al., 2004), (Byung-Kwon Park, Hyoil Han et Il-Yeol Song, 2005), (Boussaid, Ben Messaoud et Choquet, 2006).

La définition de l'hétérogénéité est assez ambiguë dans la littérature. En effet, certains travaux qualifient les données de différentes catégories d'hétérogènes (Kim et Park, 2003), (Beneventano, Bergamaschi et al., 2002), (Maibaum, Zamboulis et al., 2005). Alors que d'autres, traitant des données de même catégorie mais avec des modélisations différentes utilisent aussi le terme d'hétérogénéité (Saccol et Heuser, 2002). On trouve même des travaux, désignant des données de même catégorie avec la même modélisation qui parlent de données hétérogènes (da Silva, Evangelista Filha et al., 2002). Ceci est dû au fait qu'il n'y a pas de modèle de représentation unique pour les données à intégrer. Plusieurs modèles, selon les caractéristiques des sources et les manipulations à effectuer peuvent être retenus. Il n'existe pas, par ailleurs, un langage universel d'interrogation de données hétérogènes mais plusieurs langages peuvent être utilisés. Le traitement de données complètement hétérogènes structurées, semi-structurées et non structurées s'avère donc un volet de recherche récent et assez peu exploré.

Un entrepôt de données, qu'il soit homogène ou hétérogène, nécessite d'être maintenu. Il doit aussi évoluer en fonction de l'évolution des données sources aussi bien au niveau des structures que celui des données**.**

Le problème de la maintenance est également très complexe. Les algorithmes proposés dans la littérature traitent essentiellement de données sources (DS) homogènes (Zhuge, Garcia-Molina et al., 1995), (Zhuge, Garcia-Molina et Wiener, 1996), (O'Gorman, Agrawal et El Abbadi, 1999), (Laurent, Lenchtenboer-Ger et al., 2001).

Cet article est structuré comme suit. Le paragraphe 2 traite de la construction des entrepôts de données hétérogènes. La concrétisation de la création des vues matérialisées de l'entrepôt est présentée dans le paragraphe 3. Nous abordons ensuite les problèmes de maintenance des vues matérialisées dans la section 4. L'accent est mis, là aussi, sur les limites de ce SGBD. Nos travaux futurs sont donnés en guise de conclusion.

2. Construction d'entrepôts de données hétérogènes

Notre objectif est l'intégration de données provenant de sources différentes et hétérogènes afin de construire des entrepôts de données (cf. figure 1). Dans notre travail, les DS sont dites hétérogènes si elles vérifient l'une des deux propriétés suivantes :

1. Elles appartiennent à la **même catégorie** (structurées, semi-structurées et non-structurées) de données mais elles ont des **modélisations différentes**. Ainsi le traitement d'une base de données relationnelles et d'une base de données objet-relationnelles revient à traiter des données hétérogènes.
2. Elles appartiennent à des **catégories de données différentes**. Ainsi le traitement d'une base de données relationnelles et d'une base de données XML entre dans le cadre du traitement de données hétérogènes.

Nous ne traitons dans cet article que l'intégration de données hétérogènes structurées relationnelles et objet-relationnelles (avec des types définis par l'utilisateur ou d'autres types tels que le type texte ou le type XML).

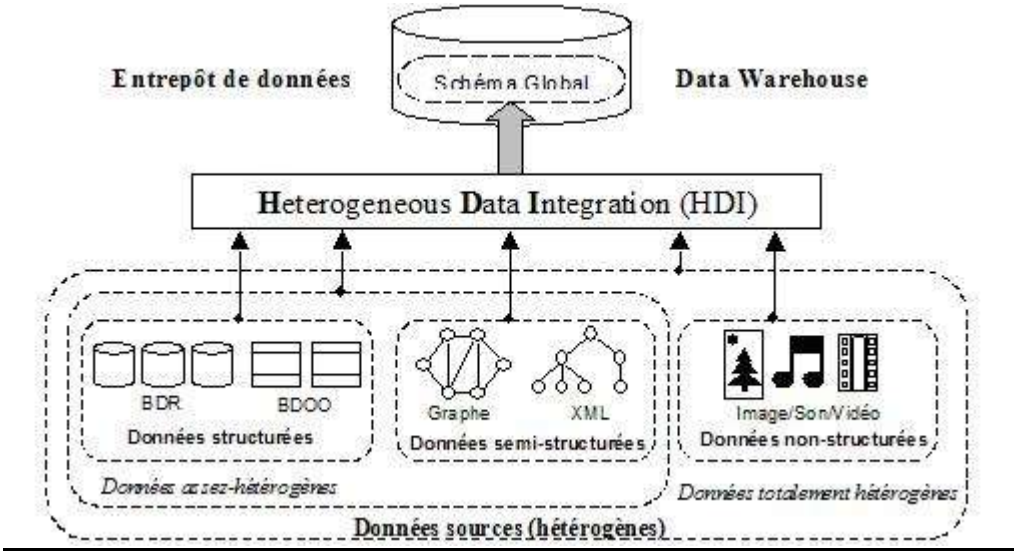


Figure 1. L'hétérogénéité des sources d'un entrepôt de données

Soit un ensemble E de bases de données à intégrer afin de former un entrepôt de données DW, {E= DB_k, k∈ Ω }. Dans la suite, nous noterons le schéma de la base de données DB_k par S_k. On note S_k =(R_k, ATT_k, DOM_k).

Rappelons que pour le modèle relationnel ou objet-relationnel, un schéma S, sans contrainte, repose sur l'existence d'un ensemble R_S de noms de *relations*, d'un ensemble ATT_S de noms d*'attributs* et d'un ensemble DOM_S de noms de *domaines*. Ces ensembles sont considérés comme des données du niveau conceptuel. Un schéma de base de données relationnelles ou objet-relationnelles peut donc être défini comme S = (R_S, ATT_S, DOM_S) **;** tel que :

- Chaque nom d'attribut A ∈ ATT_S est associé à un seul nom de domaine $dom_S(A)$ ∈ DOM_S. Cette association définit un attribut, noté A : $dom_S(A)$ ou simplement A, s'il n'y a pas risque de confusion.
- Chaque nom de domaine D ∈ DOM_S dénote un ensemble de valeurs atomiques (types de base) ou bien des types composés. Dans le cas du relationnel, les types de base sont ceux proposés par le système de gestion de bases de données (tels que Oracle, DB2).
- Chaque nom de relation R ∈ R_S est associé à un ensemble fini et non vide d'attributs $att_S(R)$ ⊆ ATT_S. Cette association définit un schéma de relation de *nom* R et de *structure* (A_1:$dom_S(A_1)$,…,A_n:$dom_S(A_n)$) noté par R (A_1 : $dom_S(A_1)$, …, A_n : $dom_S(A_n)$) où n est la cardinalité de $att_S(R)$.

Notons qu'un ensemble C_s de contraintes peut accompagner la définition d'un schéma S.
Notre approche d'intégration de bases de données peut être décrite par plusieurs étapes. La première se résume par le choix des différents composants de l'entrepôt de données à construire. Un ensemble de liens est défini ensuite entre les différents composants de sources de données. Cet ensemble est utilisé dans l'étape suivante qui consiste à filtrer les composants de l'entrepôt. Le schéma global de l'entrepôt est ensuite défini, cette étape est suivie par celle de construction qui consiste à alimenter les vues de l'entrepôt par les données des sources.

2.1. Choix des composants de l'entrepôt

Dans cette étape, l'ensemble L des composants de l'entrepôt est **sélectionné**. L constitue l'ensemble des attributs (de type prédéfini ou de type utilisateur) ou des éléments (XML) des sources de données qui doivent figurer dans l'entrepôt : $L \subseteq \cup_{k \in \Omega} (ATT_k)$.
L'ensemble des composants de l'entrepôt est donc un sous-ensemble de tous les composants des différentes bases hétérogènes constituant les sources de données.

2.2. Définition des liens entre les composants des sources

Plusieurs relations peuvent être définies entre les différents composants des sources (synonymie, inclusion, disjonction, incompatibilité…) (Beneventano, Bergamaschi et al., 2000). C'est le concepteur (administrateur) de l'entrepôt qui donne ces liens.
Dans le cadre de ce travail, nous définissons deux types de liens qui peuvent exister entre les différents composants des bases de données concernées : la **SYNonymie** et l'**INClusion**.
Le lien de SYNonymie est alors défini en tant qu'une relation d'équivalence **SYN** sur L. Celui d'INClusion est défini en tant qu'une relation d'ordre stricte **INC** sur L.
Nous définissons, par ailleurs, la **compatibilité** entre domaines de la manière suivante :
Définition 3 (compatibilité entre domaines)
Etant donnés deux composants A et A', Dom(A) et Dom(A') sont compatibles :

$$\frac{\exists \text{ un ensemble } \Delta \text{ tel que : } Dom(A) \subseteq \Delta \text{ et } Dom(A') \subseteq \Delta}{\text{On note } Dom(A) \approx Dom(A')}$$

Dans le cas des bases de données relationnelles ou objet-relationnelles les deux liens considérés sont définis ci-dessous.
Définition 4 (Synonymie) : Etant donnés deux attributs A ($A \in R$, $R \in DB_1$) et A' ($A' \in R'$, $R' \in DB_2$), A SYN A' Si Dom(A) ≈ Dom(A'), et si l'ensemble C1 des contraintes de domaine pour A est logiquement équivalent à l'ensemble C2 des contraintes de domaines pour A'.Définition 5 (Inclusion) : Etant donnés deux attributs A ($A \in R$, $R \in DB_1$) et A' ($A' \in R'$, $R' \in DB_2$), A INC A' Si Dom(A) ≈ Dom(A'), et si C1 et C2 ne sont pas logiquement équivalents et toute contrainte de l'ensemble C2 est impliquée par C1.Dans le cas des attributs composés ces deux liens sont définis dans ce qui suit.
Définition 6 (Synonymie, attributs composés) : Soient les deux attributs composés A et A'où $A=\{c_1,\ldots,c_r\}$ et $A'=\{c'_1,\ldots,c'_{r'}\}$ avec $c_1,\ldots,c_r$ et $c'_1,\ldots,c'_{r'}$ sont des attributs atomiques. A **SYN** A' si $r = r'$ et c_s **SYN** c'_s pour tout s compris entre 1 et r.
Définition 7 (Inclusion, attributs composés)
Soient les deux attributs composés A et A': $A=\{c_1,\ldots,c_r\}$ et $A'=\{c'_1,\ldots,c'_{r'}\}$ avec $c_1,\ldots,c_r$ et $c'_1,\ldots,c'_{r'}$ sont des attributs atomiques. A **INC** A' si $r \geq r'$ et c_s **SYN** c'_s pour tout s compris entre 1 et r.

2.3. Filtrage de l'ensemble des composants de l'entrepôt

L'ensemble des composants de l'entrepôt est raffiné en utilisant l'ensemble des liens définis dans l'étape précédente. En effet, deux étapes de filtrage sont effectuées, sur l'ensemble L, selon les liens établis.

Etape 1 : Filtrage en utilisant la relation d'équivalence SYN

L est filtré en laissant *un seul représentant de chaque classe d'équivalence* correspondante au lien SYN. Le choix du représentant est aléatoire et il n'intervient pas dans le processus d'intégration. Le domaine associé au représentant choisi est égal à l'*union* des différents domaines de tous les composants de la classe d'équivalence.

Etape 2 : Filtrage en utilisant la relation d'ordre INC

La deuxième opération de filtrage consiste à laisser *le "plus grand" élément correspondant à la relation d'ordre* INC. Le domaine associé au représentant choisi est égal à l'*union* des différents domaines de tous les éléments inclus ou égale (INC) à ce dernier.

L'ensemble L' résultat du filtrage est donc formé : L'$\subseteq$ L. L' peut être écrit de la manière suivante :

$$L' = \cup_{k \in \Omega} (L'_k) \text{ avec } L'_k = L' \cap (ATT_k)$$

2.4. Génération du schéma global de l'entrepôt

La génération du schéma global comporte les étapes suivantes :

1- Pour chaque L'_k *un graphe non orienté* G_k est construit. Chaque nœud correspond à une relation R$\in$ R_k et chaque arc indique l'existence d'une contrainte de clé étrangère entre deux relations. Soit d' le nombre de ces graphes.

2- On détecte les *sous-graphes connexes* pour chaque graphe G_k. Rappelons qu'un graphe est dit connexe s'il existe au moins un chemin entre chaque couple de noeuds du graphe (Lellahi et Zamulin, 2001). Le graphe G_k est donc l'union de ses sous graphes connexes.

3- Soit [k l'ensemble formé de ces sous graphes, à savoir, [k={G_{ki} / i$\in T_k$} où T_k désigne une indexation pour [k

4- Soit $T = \Pi_{k \in \Omega} T_k$. Pour tout u = ($u_1$,…,$u_d$) $\in$ T. On considère le graphe C_u formé de l'union de tous les C_{ui} (i=1,…,d), et on dénote par C l'ensemble des graphes ainsi obtenu : C={ C_u / u $\in$ T}

5- Un *schéma de vue* V_u est associé à chaque *sous graphe* C_u telles que les relations intervenant dans V_u sont celles représentées par des nœuds dans le graphe C_u et les attributs de V_u sont ceux de L' qui appartiennent aux relations intervenant dans V_u.

2.5. Construction des vues de l'entrepôt

L'étape précédente de l'approche a permis d'établir le nombre de vues (α) de l'entrepôt et leurs structures (DW={V_i / i $\in$ [1.. α]}). Un algorithme de construction ConstruitVue( ) sera appliqué pour chacune d'entre elles.

Pour toute vue V à construire, l'algorithme construitVue( ) consiste à intégrer des données de différentes bases afin d'alimenter la vue. Cet algorithme est présenté dans la suite pour une vue V. La procédure Créer_Vue(Q) consiste à exécuter une requête de sélection, des différents composants figurant dans Q. Dans notre cas, la sélection consiste à interroger la base concernée avec le langage SQL.

```
ConstruitEntrepôt_ETL ()
Début
    Pour toute V de l'entrepôt faire
        ConstruitVue(V)
    Fin pour
Fin
```

Début
  Pour tout $k \in \Omega$ **faire**
    Q :=vue_vide /* initialisation vue vide */
    $V0 := (ATT_k) \cap V$
    $V' := V - V0$
    $y := |V'|$ /* y est la cardinalité de V' */
    Pour tout $v' \in V'$ **faire**
      $SI_{v'} := (S(skl(v'), v', k) \cup I(skl(v'), v', k))$
    Fin pour
    Pour tout $(a_1, \ldots, a_y) \in \square \prod_{v' \in V'} (SI_{v'})$ **faire**
      $Q := V0 \cup \{a_1, \ldots, a_y\}$
      Q := Q (union) Créer_Vue(Q)
      * Q contient le résultat partiel de l'interrogation de la base DB_k */
    Fin pour
    V := V (union) Q
  Fin pour
Fin

2.6. Bases de données exemple

L'exemple sur lequel nous travaillons porte sur les systèmes d'information de santé : SIM. En effet, le dossier médical informatisé d'un patient, qui peut être suivi pour plusieurs pathologies, est un cas très complet d'intégration de données hétérogènes (web, informations spécialisées, images numériques, …) et de leurs mises à jour régulières.

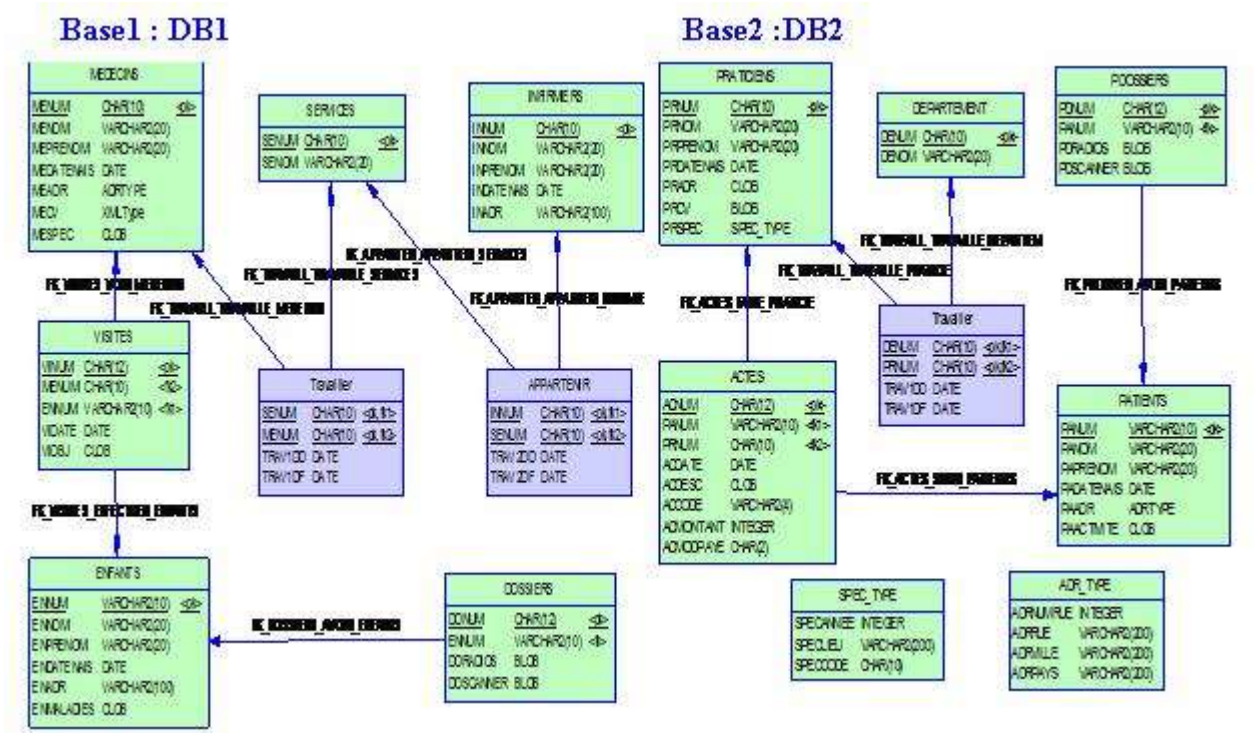


Figure 2. Structure des deux bases SIM

Les différentes étapes de construction de l'entrepôt de données sont les suivantes :

-1- Liste des composants de l'entrepôt

L={DB1.MEDECINS.MENUM; DB1.MEDECINS.MENOM ; DB1.VISITES.VINUM ; DB1.VISITES.VIDATE ; DB2.PRATICIENS.PRNUM ; DB2.PRATICIENS.PRNOM ; DB2.ACTES.ACNUM ; DB2.ACTES.ACDATE ; DB2.ACTES.ACCODE}

-2- Etablissement des liens de Synonymie et d'Inclusion

DB1.VISITES.VINUM **SYN** DB2.ACTES.ACNUM
DB1.VISITES.VIDATE **SYN** DB2.ACTES.ACDATE
DB1.MEDECINS.MENUM **INC** DB2.PRATICIENS.PRNUM
DB1.MEDECINS.MENOM **INC** DB2.PRATICIENS.PRNOM

-3- Filtrage en utilisant la relation d'équivalence SYN et la relation d'ordre INC

Utilisation de la SYNonymie

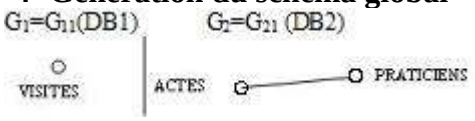
L={DB1.MEDECINS.MENUM; DB1.MEDECINS.MENOM ; DB1.VISITES.VINUM ; DB1.VISITES.VIDATE ; DB2.PRATICIENS.PRNUM ; DB2.PRATICIENS.PRNOM ; ~~DB2.ACTES.ACNUM~~ ; ~~DB2.ACTES.ACDATE~~ ; DB2.ACTES.ACCODE}

Utilisation de l'INClusion

L={~~DB1.MEDECINS.MENUM~~; ~~DB1.MEDECINS.MENOM~~ ; DB1.VISITES.VINUM ; DB1.VISITES.VIDATE ; DB2.PRATICIENS.PRNUM ; DB2.PRATICIENS.PRNOM ; ~~DB2.ACTES.ACNUM~~ ; ~~DB2.ACTES.ACDATE~~ ; DB2.ACTES.ACCODE}

L'={DB1.VISITES.VINUM ; DB1.VISITES.VIDATE ; DB2.PRATICIENS.PRNUM ; DB2.PRATICIENS.PRNOM ; DB2.ACTES.ACCODE}

-4- Génération du schéma global



On a donc une seule vue résultat pour l'entrepôt. Elle contient tous les attributs de L'.

V={DB1.VISITES.VINUM ; DB1.VISITES.VIDATE ; DB2.PRATICIENS.PRNUM ; DB2.PRATICIENS.PRNOM ; DB2.ACTES.ACCODE}

5-En applicant l'algorithme ConstruitVue à la vue V de l'entrepôt, nous aurons :

Pour k=1

Q={DB1.VISITES.VINUM ;DB1.VISITES.VIDATE ;DB1.MEDECINS.MENUM ; DB1.MEDECINS.MENOM ; null}

Pour k=2

Q={DB2.ACTES.ACNUM ;DB2.ACTES.ACDATE;DB2.PRATICIENS.PRNUM ; DB2.PRATICIENS.PRNOM ; DB2.ACTES.ACCODE}

Ainsi les composants de l'ED sont déterminés. La construction physique des vues est effectuée à l'aide des requêtes (SQL dans le cadre du relationnel-étendu, XML-query pour les données de type XML).

3. La gestion des vues matérialisées sous Oracle

Dans cette section, nous illustrons la gestion (création et maintenance) des vues matérialisées (VM) à travers le SGBD Oracle. Notre choix de l'outil se justifie essentiellement par la position qu'il occupe sur le marché et notamment par sa réputation en matière de performance et de gestion de gros volumes de données complexes.

3.1. Création et rafraîchissement de vues matérialisées

Une vue matérialisée (VM) est définie sur une ou plusieurs tables à partir d'une ou de plusieurs bases. Ces dernières peuvent être homogènes ou hétérogènes. La structure d'une vue peut être ainsi composée d'attributs de types complexes. Les données sont extraites à partir des DS.

L'entrepôt de données mentionné à construire peut être composé de plusieurs vues issues de l'intégration de plusieurs sources hétérogènes (cf. Figure 2). Nous avons effectué des tests de création de VM sous Oracle en utilisant les différentes combinaisons possibles entre type et périodicité de rafraîchissement. Chaque combinaison est testée à la fois sur des données de types simples et des données de types complexes. Les mêmes tests ont été effectués sur des VM mono-table et des VM multi-tables.
Nous constatons, à notre grand étonnement, que la seule possibilité pour intégrer des données hétérogènes serait de créer **une vue multi-tables mono-base** avec le type de **rafraîchissement** le plus coûteux vu qu'il doit être **complet et sur demande**.
Nous avons développé en PL/SQL le package *pck_constvues* qui permet la création d'entrepôts de données hétérogènes en remplaçant les vues matérialisées, version Oracle, par des tables classiques (Boufares et Hamdoun, 2005). Nous n'avons pas pu joindre, en annexe, le package par manque de place.

3.2. La Maintenance des vues matérialisées sous Oracle

La maintenance des VM revient à répercuter les changements apparus au niveau des DS. Afin de s'adapter à l'évolution des processus d'analyse, l'évolution d'un entrepôt requiert, d'une part, la maintenance de structure et, d'autre part, la maintenance de données.

3.2.1. Maintenance de la structure des VM

La maintenance structurelle a pour objectif de maintenir le schéma des vues de l'ED suite aux changements des structures des DS (ajout/suppression de DS, ajout/suppression de relation/classe, ajout/suppression d'attribut, …). L'ED peut aussi subir des changements de structure, indépendamment des DS, suite à une redéfinition de l'une ou de plusieurs de ses vues, ou suite à la matérialisation de vues non matérialisées, ou suite à une reconfiguration (ajout de vues). La maintenance structurelle peut engendrer un changement fondamental du schéma de l'ED (Badri, Boufares et al., 2005)
A travers les tests qu'on a effectué lors de la manipulation des vues matérialisées sous Oracle, nous constatons qu'il est impossible de modifier la structure des données d'une table source si l'un de ses attributs est de type complexe. Il est cependant possible de modifier la structure de certaines données de types simples qui appartiennent à des tables relationnelles, seulement ces modifications ne sont pas répercutées sur les VM.

3.2.2. Maintenance des données des VM

La maintenance des données consiste à alimenter l'entrepôt suite aux mises à jour des sources de données. Oracle assure la maintenance de données à travers le rafraîchissement. Ce dernier est déclanché de deux manières selon la périodicité du rafraîchissement choisi lors de la définition de la VM (ON COMMIT ou ON DEMAND).
Pour assurer le type de rafraîchissement incrémental (FAST), un journal de vues matérialisées doit être crée manuellement par l'utilisateur avec la syntaxe *CREATE MATERIALIZED VIEW LOG ON nom_de_la_table*. Le journal créé est une vue matérialisée qui a pour nom MLOG$_nom_de_la_table. La liste des journaux se trouve dans la méta-table USER_MVIEW_LOGS.
La maintenance des VM issues de données hétérogènes et complexes n'est possible, sous Oracle, qu'avec les options COMPLETE et ON DEMAND. Ce qui signifie que les vues sont entièrement recalculées. Afin d'éviter de les régénérer complètement nous proposons une gestion "manuelle" du rafraîchissement. Ceci nécessite la clause FOR UPDATE : *CREATE MATERIALIZED VIEW nom_de_la_MV FOR UPDATE as sous_requête*. Ce qui pourrait permettre de reporter des insertions, des modifications ou des suppressions dans la VM selon les opérations de mise à jour sur les données sources.

Nous remarquons qu'Oracle ne permet pas de créer des VM modifiables dans le cas où les données sources sont issues de plusieurs tables.
Ces manquements dans les versions actuelles 9i/10g d'Oracle < www.oracle.com> nous amènent à construire un entrepôt de données avec sources hétérogènes non sous forme de vues matérialisées mais sous forme de tables. Celles-ci sont construites à partir de plusieurs tables éventuellement définies sur des colonnes de types complexes. Ainsi, la maintenance de cet entrepôt suivra la procédure que nous proposons ci-dessous.

3.2.3. Procédure de maintenance des VM

Nous proposons la procédure MV_maintenance qui assure la maintenance *incrémentale* des VM multi-tables issues de données complexes de type objet. Notre procédure, contrairement à celle employée par Oracle (DBMS_MVIEW.REFRESH), ne recalcule pas entièrement la vue mais rajoute uniquement les lignes qui ont été mises à jour dans les tables sources.
Dans cet article, nous ne traitons que le cas *mono-base*. On donne ci-dessous notre démarche :
Etant donné, l'ensemble des relations R_k ($k \in [1..n]$) des données sources.
La relation R_k est définie sur l'ensemble de ses attributs A_k, on note $R_k(A_k)$ ou R_k, avec $A_k = \{A_{k1}, A_{k2}, \ldots A_{km}\}$.
La vue matérialisée V est définie sur le sous-ensemble B des attributs des relations concernées : $B \subseteq \bigcup_{k \in [1..n]} A_k$; On note V(B) ou V.
Les données de la vue V(B) constituent un sous-ensemble du produit ($\prod$) des relations R_k. Soit $R'_1(A_1)$ l'ensemble des mises à jour, depuis la dernière maintenance, dans une relation source donnée $R_1(A_1)$.
On définit la vue V'(B) comme étant égale au sous-ensemble du produit de R'_1 et des relations R_j $j \in [2..n]$.
Le rafraîchissement de la vue V sera égal à l'union de V et de V' :
$V' = R'_1 \times \prod_{j \in [2..n]} R_j$; $V = V \cup V'$

4. Conclusion

Nous avons entièrement terminé, en utilisant Oracle Forms et PL/SQL, le développement de l'outil HDI. Celui-ci permet d'intégrer des données hétérogènes relationnelles et objet-relationnelles (avec types complexes et XML). L'algorithme présenté prend en considération les relations qui existent entre les composants (attributs ou éléments) dans une même base et les liens qui existent (synonymie et inclusion) entre ces composants dans des bases différentes. La maintenance des vues matérialisées qui constituent l'entrepôt s'est avérée très limitée avec le SGBD Oracle. En effet, celui-ci ne permet que la création de VM multi-tables mono-base et une maintenance en recalculant toute la vue. Nous avons ainsi proposé une procédure de maintenance afin de ne répercuter que les mises à jour nécessaires.
Des mesures sont en cours de réalisation sur plusieurs outils afin d'aider à déterminer le type de VM à créer (relationnelles, objet ou XML). L'extension des liens entres les composants dans les DS (autres que la synonymie et l'inclusion) d'une part, et d'autre part, la généralisation de notre algorithme aux données non structurées constituent nos travaux futurs.

(Zhuge, Garcia-Molina et al., 1995) : Zhuge, Y., Garcia-Molina, H., Hammer, J., Windom, J., (1995). View Maintenance in a Warehousing Environment. *Proc. of the ACM SIGMOD*, pp 316-327, Mai 1995 California

(Zhuge, Garcia-Molina et Wiener, 1996) : Zhuge, Y., Garcia-Molina, H., Wiener, J.L., (1996). The Strobe Algorithms for Multi-Source Warehouse Consistency. *Parallel and Distributed Information Systems*, pp 146-157 Décembre 1996.
(O'Gorman, Agrawal et El Abbadi, 1999) : O'Gorman, K., Agrawal, D., et El Abbadi, A., (1999). Posse: A Framework for Optimizing Incremental View Maintenance at Data Warehouses. *Data Warehousing and Knowledge Discovery.* pp 106-115, Italie 1999.
(Beneventano, Bergamaschi et al., 2000) : Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M., et Vincini, M., (2000). Information integration: the MOMIS project demonstration. *In International Conference on Very Large Data Bases VLDB*, pp 611-614, 2000, Le Caire Egypt.
(Lellahi et Zamulin, 2001) : S.K. Lellahi, A. Zamulin, (2001). Object-oriented database as a dynamic system with implicit state. *Proceedings of the Fifth East-European Conference on Advances in Database and Information System (ADBIS'01).* pp 239-252, Vilnius, Lithuania, Septembre 2001.
(Laurent, Lenchtenboer-Ger et al., 2001) : Laurent, D., Lenchtenboer-Ger, J., Spyratos, N., Vossen, G., (2001). Monotonic Complements for Independent Data Warehouses, *The International Journal of Very Large Data Base VLDB*, volume 10 issue 4, pp 295-315, Décembre 2001.
(Saccol et Heuser, 2002) : Saccol, D.d.B., et Heuser, C. A., (2002). Integration of XML Data, *LNCS 2590*, pp 68-80, 2002.
(Da Silva, Evangelista et al., 2002) : Da Silva, A.S., Evangelista Filha, I. M. R., Laender, A. H. F., Embley, D. W., (2002). Representing and querying semistructured Web Data Using Nested Tables With structural Variants. *LNCS-2503 : 21st International conference on conceptual modelling ER*, pp 135-151 Octobre 2002, Tampere Finland.
(Beneventano, Bergamaschi et al., 2002) : Beneventano, D., Bergamaschi, S., Castano, S., De Antonellis, V., Ferrara, A., Guerra, F., Mandreoli, F., Ornetti, G. C., et Vincini, M., (2002). Semantic Integration and query optimization of heterogeneous data sources. *LNCS 2426*, pp 154-165, Septembre 2002.
(Kim et Park, 2003) : Kim, H.H. et Park, S. S., (2003). Building a Web-enabled Multimedia Data warehouse. *LNCS 2713*, pp 594-600, Juin 2003.
(Nassis, Rajugan et al., 2004) : R. , Nassis, V., Rajugan, R. , Dillon, T. S. et Rahayu, W., (2004). Conceptual Design of XML Document Warehouses. *Data Warehousing and Knowledge Discovery: 6th International Conference, DaWaK 2004*, Zaragoza, Espagne, Septembre 1-3, 2004.
(Park, Han et Song, 2005) : Park, B.-K., Han, H. et Song, I.-Y., (2005). XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses. *Data Warehousing and Knowledge Discovery: 7th International Conference, DaWaK 2005, Copenhagen, Denmark*, Aout 22-26, 2005.
(Badri, Boufares et al., 2005) : Badri, M., Boufares, F., Ducateau, C.F., et Gargouri, F., (2005). Etat de l'art de la maintenance des entrepôts de données issus de systèmes d'information hétérogènes. *Cinquièmes Journée Scientifiques GEI*, pp 13-18, Mars 2005, Sousse Tunisie.
(Maibaum, Zamboulis et al., 2005) : Maibaum, M., Zamboulis, L., Rimon, G., Orengo, C., Martin, N. et Poulovassilis, A., (2005). Cluster based Integration of Heterogeneous Biological Databases using the AutoMed toolkit. *In Proceedings of DILS'05*, 2005.
(Boufares et Hamdoun, 2005) : Boufares F. et Hamdoun S., (2005). Integration Techniques to Build a Data Warehouse using Heterogeneous Data Sources. *Journal of Computer Science*, pp 48-55, November 2005, New York USA.
(Boussaid, Ben Messaoud et al., 2006) : Boussaid O., Ben Messaoud, R., Choquet, R., Anthoard, S., (2006). Conception et construction d'entrepôts en XML. *Dans la RNTI correnspondant à la 2ième journée francophone sur les entrepôts de données et l'analyse en ligne EDA'06 Versaille 19 juin 2006*.
< www.oracle.com> Oracle Database 10*g* Release 2.www.oracle.com.