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.

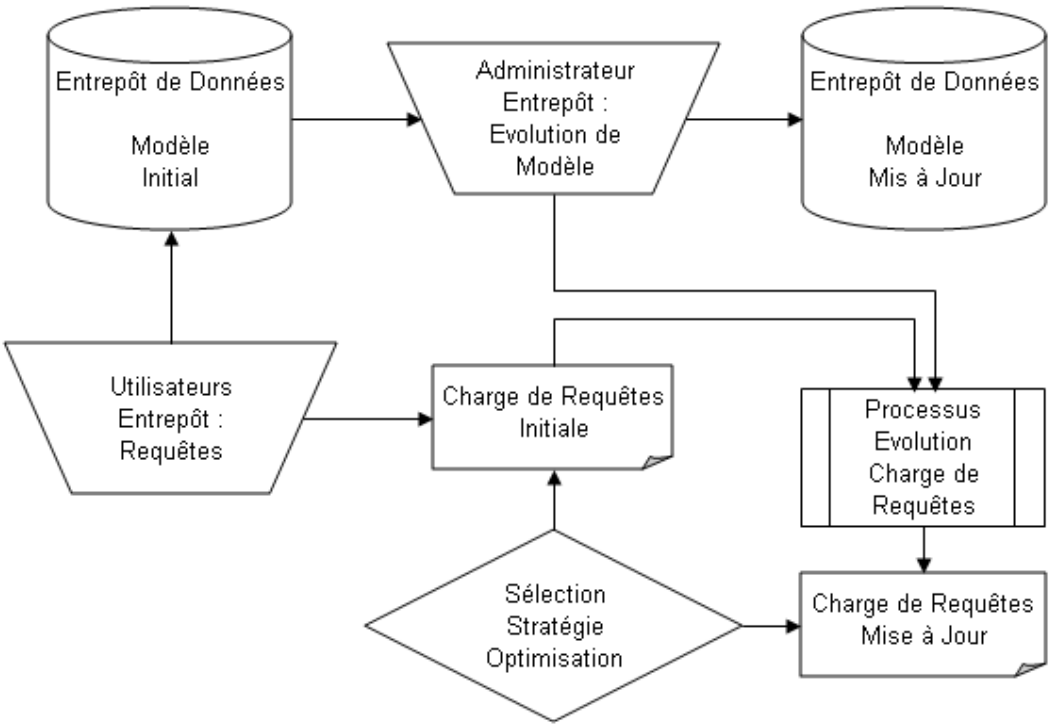

FIG. 6.2 – Processus pour l'évolution de la charge

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.

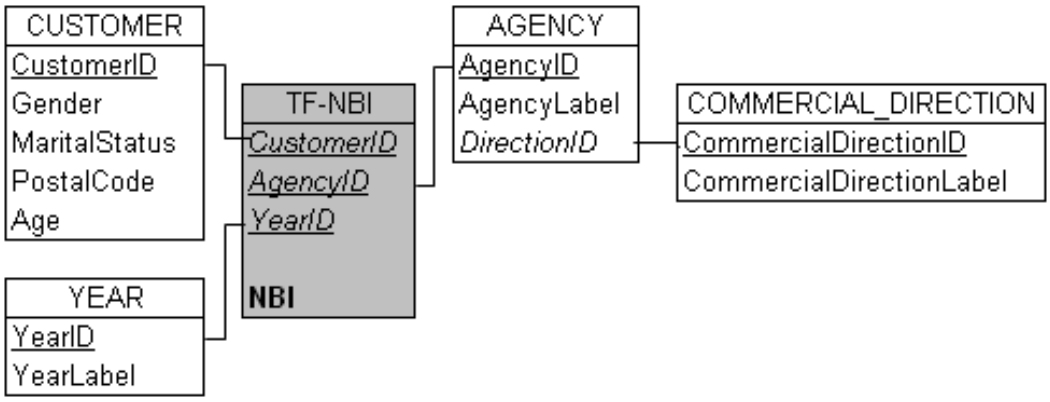

FIG. 6.3 – Schéma initial de l'entrepôt pour l'analyse du NBI

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.

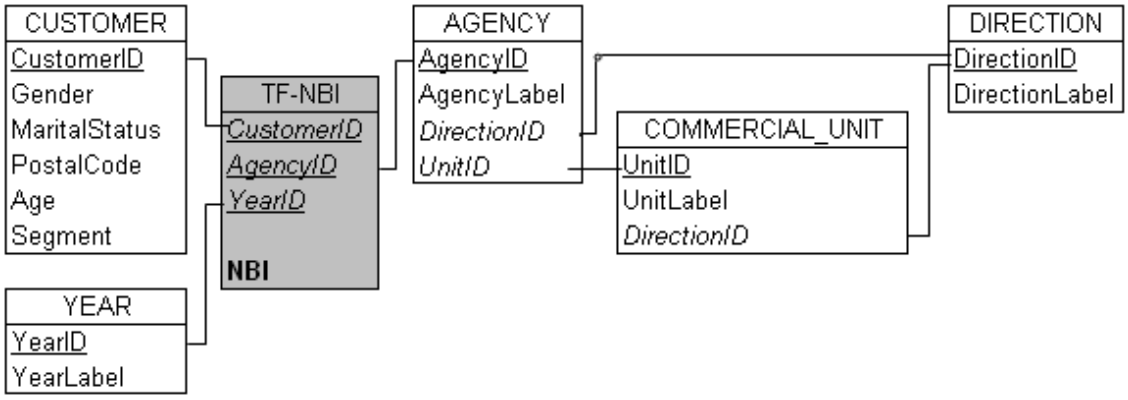

FIG. 6.5 – Schéma mis à jour de l'entrepôt pour l'analyse du NBI

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.