OptAssist : outil d'assistance pour l'optimisation des entrepôts de données relationnels

Kamel Boukhalfa[1], Ladjel Bellatreche[2], and Zaia Alimazighi[1]

[1] USTHB University - Algiers - Algeria
boukhalk@ensma.fr, alimazighi@wissal.dz
[2] LISI/ENSMA - Poitiers University - France
bellatreche@ensma.fr

Résumé Pour optimiser son entrepôt, l'administrateur est amené à effectuer une conception physique. Durant cette phase, il doit effectuer de multiples choix comme les techniques d'optimisation à sélectionner, leurs algorithmes de sélection et les valeurs des paramètres de ces algorithmes ainsi que les attributs et les tables utilisés par certaines de ces techniques. Nous montrons dans cet article la nature des difficultés rencontrées par l'administrateur durant la conception physique. Nous présentons par la suite un outil d'aide qui permet à l'administrateur d'effectuer les bons choix d'optimisation. Nous montrons l'utilisation interactive de cet outil sur un entrepôt de données issu du Benchmark APB-1.

Key words: Optimisation, Entrepôts de données, Conception physique, Fragmentation horizontale, Index de jointure binaires

1 Introduction

Les principales caractéristiques des entrepôts de données sont leur grande taille et la complexité des requêtes décisionnelles dues aux opérations de sélection, de jointure et d'agrégration. Ces caractéristiques ont rendu la tâche d'administration de plus en plus complexe. Traditionnellement, dans les applications de gestion de bases de données de type OLTP (On-Line Transaction Processing), la tâche d'un administrateur était principalement concentrée sur la gestion des utilisateurs et l'utilisation d'un nombre restreint de techniques d'optimisation comme les index et les vues.

L'optimisation du temps d'exécution de requêtes constitue une exigence primordiale des utilisateurs de l'entrepôt. Pour satisfaire cette exigence, l'administrateur de l'entrepôt de données (AED) doit effectuer une conception physique qui devient cruciale pour garantir une bonne performance. La conception physique doit déterminer comment une requête doit être exécutée efficacement sur l'entrepôt de données. Pour cela, l'AED dispose d'un ensemble de techniques d'optimisation comme la fragmentation verticale, horizontale, les index, etc. Il pourra utiliser une seule technique ou en combiner plusieurs afin d'avoir une meilleure performance. Plusieurs algorithmes de sélection sont disponibles pour chaque technique choisie, chacun caractérisé par un ensemble de paramètres à

régler. Pour certaines techniques, plusieurs objets de l'entrepôt sont candidats (généralement des tables et des attributs). Pour bien mener sa conception physique, l'AED est confronté à effectuer plusieurs choix liés à : (1) les techniques d'optimisation, (2) la nature de sélection, (3) les algorithmes de sélection et leurs paramètres et (4) les tables et attributs candidats (voir figure 1).

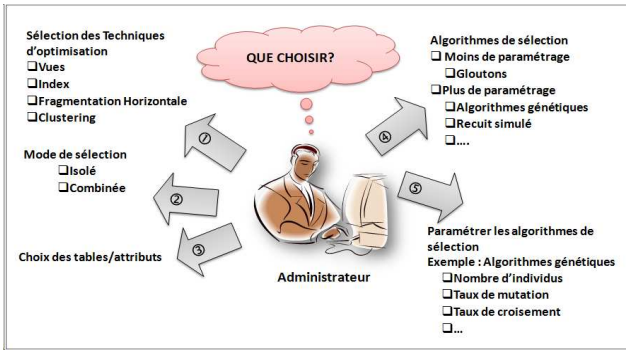

Figure 1. Choix effectués par l'administrateur

Choix des techniques d'optimisation : Si nous explorons la littérature et les éditeurs de gestion de bases de données commerciaux, nous trouvons une large panoplie de techniques d'optimisation qui peuvent être redondantes ou non. Les *techniques redondantes* nécessitent un espace de stockage et un coût de maintenance (les vues matérialisées, les index avancés, la fragmentation verticale, etc.) par contre les *techniques non redondantes* ne nécessitant ni espace de stockage ni coût de maintenance (la fragmentation horizontale, le traitement parallèle, etc.). Pour optimiser les requêtes définies sur l'entrepôt, l'AED peut choisir une ou plusieurs techniques parmi ces deux catégories. Ce choix est souvent difficile dû au fait que certaines techniques sont bénéfiques pour certaines requêtes et non pas pour d'autres.

Le choix du mode de sélection : En présence de plusieurs techniques d'optimisation, l'AED dispose de deux modes de sélection : la *sélection isolée* et la *sélection multiple*. Dans la sélection isolée, il choisit une seule technique qui pourra être redondante s'il dispose d'assez d'espace et peu de mises à jour par exemple, sinon il pourra choisir une technique non redondante. La sélection isolée a été largement étudiée [1,2,3,4,5], mais elle est souvent insuffisante pour une meilleure optimisation de l'entrepôt. La sélection multiple permet de sélectionner plusieurs techniques à la fois. Elle est principalement motivée par les fortes similarités entre les techniques d'optimisation. Les travaux majeurs dans cette catégorie sont principalement concentrés sur la sélection des vues matérialisées et des index [6,7,8].

Choix et paramétrage des algorithmes de sélection : Une fois que les techniques d'optimisation utilisées sont choisies, l'AED est confronté au problème de choix de leurs algorithmes de sélection. Pour chaque mode de sélection, isolée ou multiple, un large choix d'algorithmes est possible. Ces algorithmes sont de natures diverses qui vont des simples algorithmes comme les algorithmes gloutons aux algorithmes plus complexes comme les algorithmes basés sur la programmation linéaire, les algorithmes génétiques, les colonies de fourmis, etc. Certains algorithmes possèdent peu de paramètres comme les algorithmes gloutons. Par contre d'autres possèdent plusieurs paramètres qu'il faut régler et configurer pour avoir une bonne performance.

Choix des attributs et tables candidats : Les entrepôts de données relationnels sont généralement modélisés par un schéma en étoile composé d'une table de faits et un ensemble de tables de dimension. Pour certaines techniques comme la fragmentation horizontale (FH), verticale et les index, plusieurs tables et attributs sont candidats pour être utilisés par ces techniques. L'AED est confronté dans certains cas à choisir un sous-ensemble de tables et d'attributs candidats parmi l'ensemble initial. Ce choix est souvent effectué pour réduire la complexité du problème de sélection.

Comme nous venons de voir, la tâche de l'AED devient de plus en plus complexe vu le nombre important de choix à effectuer, d'où la nécessité de développement d'outils d'aide. Ces outils doivent assister l'AED pour effectuer les bons choix d'administration. Le présent article est organisé en 5 sections. La section 2 présente un état de l'art sur les outils d'aide développés pour assister l'administrateur dans sa tâche d'optimisation. La section 3 présente l'architecture générale de l'outil que nous proposons. La section 4 est consacrée à la présentation des fonctionnalités de l'outil appliquées sur un entrepôt de données. Enfin, la section 5 conclut le papier et présente quelques perspectives.

2 Etat de l'art

Pour assister l'AED dans sa tâche d'optimisation de la couche physique, certains outils ont été développés. La plupart des outils existant ont été proposés par les éditeurs des grands SGBD commerciaux dans le cadre de l'auto-administration des bases de données. Parmi ces outils, nous pouvons citer *Oracle SQL Acces Advisor* [9], *DB2 Design Advisor* [10] et *Microsoft Database Tuning Advisor* [11].

[9] propose l'outil *SQL Access Advisor* qui offre un ensemble complet de conseils sur la manière d'optimiser la conception d'un schéma pour maximiser les performances d'une application. Cet outil est un assistant automatisant certains aspects de la conception physique et de tuning [3] pratiquées manuellement sur les bases de données Oracle. L'outil analyse la charge de requêtes et propose des recommandations pour créer de nouveaux index si nécessaire, de supprimer

3. Le tuning est un ensemble d'activités utilisées pour optimiser les performances d'une base ou entrepôt de données suite à des évolutions de ces derniers

les index inutilisés, de créer de nouvelles vues matérialisées, etc. Les recommandations générées sont accompagnées par une évaluation quantifiée des gains de performance attendus ainsi que des scripts nécessaires pour les implémenter. [10] propose l'outil *DB2 Design Advisor* qui fait partie de *DB2 V8.2* et constitue une amélioration de l'outil *DB2 Index Advisor Tool* [12] qui permet de sélectionner un ensemble d'index. L'outil permet d'optimiser un ensemble de requêtes en proposant un ensemble de recommandations. Ces recommandations concernent quatre techniques d'optimisation : les index, les vues matérialisées, la FH et le clustering. L'outil *Microsoft Database Tuning Advisor* (DTA) est issu du projet *Microsoft Research AutoAdmin*. *DTA* peut fournir des recommandations intégrées pour les index définis sur les tables de base, les vues, les index définis sur les vues et le partitionnement horizontal. Il prend comme entrées un ensemble de BD sur un serveur, une charge de requêtes, les techniques d'optimisation à sélectionner ainsi qu'un ensemble de contraintes, comme l'espace de stockage des techniques redondantes. Il donne en sortie un ensemble de recommandations sur les index, les vues ainsi que la FH.

La plupart des outils que nous venons de présenter ont été proposés dans le cadre de l'auto-administration des bases de données et sont généralement spécifiques à un SGBD donné. Ils sont caractérisés aussi par l'utilisation de l'optimiseur de requêtes pour évaluer la qualité des techniques sélectionnées. Cela représente une tâche supplémentaire de l'optimiseur et pourra engendrer une détérioration du temps de réponses des requêtes.

En voulant automatiser l'administration et le tuning des bases et entrepôts de données, les auteurs de ces outils visaient à décharger l'administrateur et à l'éloigner de ces deux tâches. [13] montre qu'une conception physique élaborée sans l'intervention de l'administrateur pose un problème de robustesse. Les techniques d'optimisation générées peuvent détériorer les performances au lieu de les améliorer. Les algorithmes utilisés par ces outils pour la sélection des techniques d'optimisation sont figés et non accessibles pour l'administrateur. Il est intéressant d'enrichir cette panoplie d'outils par d'autres outils d'aide permettant plus d'interactivité avec l'administrateur. Ces outils doivent donner la possibilité à l'administrateur de personnaliser sa conception physique et d'utiliser son expérience afin d'améliorer la qualité des techniques d'optimisation sélectionnées.

L'outil *OptAssist* que nous proposons permet d'aider l'administrateur dans sa tâche d'optimisation d'entrepôts de données. Il lui permet de choisir les techniques d'optimisation, le mode de leur sélection, les algorithmes utilisés, les paramètres relatifs à chaque algorithme ainsi que les tables et les attributs pris en compte pour la génération des recommandations. *OptAssist* permet de recommander une fragmentation primaire et dérivée[4], au contraire de la plupart des outils qui proposent uniquement la fragmentation primaire. Il permet aussi une sélection multiple de la FH et des index de jointure binaires (IJB) pour mieux optimiser l'entrepôt. L'outil utilise un modèle de coût que nous avons proposé

4. La FH primaire consiste à fragmenter une table en utilisant les attributs de cette table. La FH dérivée consiste à fragmenter une table selon les fragments d'une autre table

dans [14]. Il supporte plusieurs SGBD grâce à l'exploitation de la méta-base pour collecter toutes les informations et statistiques nécessaires pour l'optimisation.

3 Architecture de l'outil

OptAssist accepte comme entrée un schéma d'entrepôt, une charge de requêtes Q et un ensemble de contraintes (le nombre de fragments maximum W et le quota d'espace réservé aux index S), il permet en sortie de fragmenter horizontalement l'entrepôt, de l'indexer ou les deux en même temps. Le choix d'utiliser la FH et les IJB est motivée par plusieurs similarités que nous avons identifiées entre ces deux techniques [14]. L'outil est composé d'un ensemble de modules permettant d'aider l'AED à effectuer ses différents choix d'optimisation de l'entrepôt (voir figure 2) : (1) module d'interrogation de la méta-base, (2) module de gestion des requêtes, (3) module de sélection d'un schéma de FH, (4) module de sélection de configuration d'IJB, (5) module de FH, (6) module d'indexation et (7) module de réécriture des requêtes.

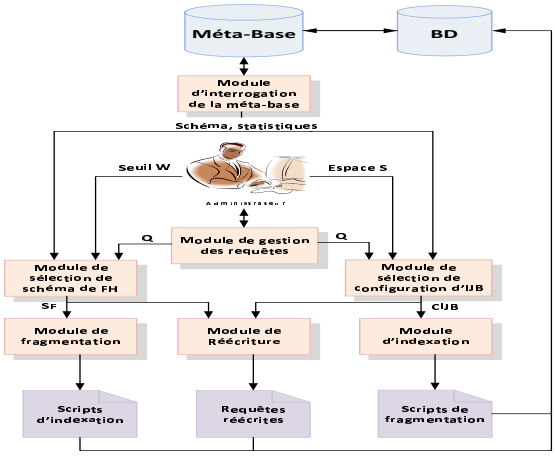

Figure 2. Architecture de l'outil

3.1 Module d'interrogation de la méta-base

Le module d'interrogation de la méta-base est un module très important qui permet à l'outil de fonctionner avec n'importe quel type de SGBD. A partir d'un type de SGBD, nom d'utilisateur et un mot de passe ce module permet de se connecter au compte correspondant et collecter un certain nombre d'informations à partir de la méta-base. Ces informations touchent deux niveaux, logique

et physique. Les informations du niveau logique regroupent la liste des tables du schéma logique que l'utilisateur gère ainsi que les attributs appartenant à ces tables. Les informations du niveau physique représentent les techniques d'optimisation utilisées ainsi qu'un ensemble de statistiques sur les tables et attributs de l'entrepôt.

3.2 Module de gestion des requêtes

Ce module permet d'aider l'AED à définir la charge de requêtes les plus fréquentes (Q) sur laquelle l'optimisation est basée. Le module permet une édition manuelle des requêtes ou une importation à partir de fichiers externes. Il permet aussi de gérer la charge en donnant la possibilité d'ajouter, de supprimer ou de modifier des requêtes. Le module intègre un *parseur* qui permet d'identifier les erreurs syntaxiques ainsi que les tables et attributs utilisés par chaque requête.

3.3 Module de sélection d'un schéma de fragmentation horizontale (MSSFH)

Le MSSFH nécessite en entrée un schéma de l'entrepôt, une charge de requêtes et un seuil W représentant le nombre de fragments maximum que l'administrateur souhaite avoir. A partir de ces données, ce module sélectionne un schéma de fragmentation (SF) permettant de minimiser le coût d'exécution des requêtes et générant un nombre de fragments ne dépassant pas W. Dans [15], nous avons effectué une étude de complexité du problème de FH dans le cadre des entrepôts de données relationnels et nous avons prouvé qu'il est NP-Complet. Pour cela, nous avons proposé trois algorithmes heuristiques pour le résoudre, un Algorithme Génétique (AG), un algorithme de Recuit Simulé (RS) et un algorithme de Hill Climbing (HC) [1,15]. Ces trois algorithmes sont supportés dans le MSSFH.

3.4 Module de sélection de configuration d'IJB

Ce module nécessite en entrée un schéma de l'entrepôt, une charge de requêtes (Q) et un espace de stockage S que l'administrateur réserve pour les index. Il sélectionne une configuration d'IJB ($CIJB$) permettant de minimiser le temps d'exécution des requêtes en entrée et occupant un espace de stockage ne dépassant pas S. Le module supporte deux algorithmes gloutons que nous avons proposé dans [14] et un algorithme basé sur une technique de data-mining (*Recherche des motifs fréquents fermés*) proposé par Aouiche et al. [6].

3.5 Module de fragmentation horizontale (MFH)

Le MFH est responsable de fragmenter physiquement l'entrepôt de données selon le schéma de FH obtenu à partir de module de sélection. A partir du schéma de FH, ce module détermine les tables de dimension à fragmenter par

la fragmentation primaire ainsi que les attributs utilisés pour effectuer cette fragmentation. Le module permet ensuite de fragmenter physiquement la table des faits par une fragmentation dérivée en utilisant les fragments des tables de dimension. Dans [14] nous avons identifié deux problèmes : (1) la plupart des SGBD ne supportent pas la FH primaire sur trois attributs ou plus et (2) la FH dérivée n'est pas supportée en cas de deux tables de dimension ou plus. Nous avons proposé une technique permettant de résoudre ces deux problèmes. Cette technique est supportée par le MFH. Ce dernier génère tous les scripts qui permettent de fragmenter les tables de dimension ainsi que la table de faits selon le schéma de fragmentation SF en entrée.

3.6 Module d'indexation

Le module d'indexation est responsable de la création des IJB sélectionnés par le module de sélection des index. Ce module génère les requêtes de création des index sur l'entrepôt de données.

3.7 Module de réécriture des requêtes

Une fois les structures d'optimisation créées physiquement sur l'entrepôt (FH et/ou IJB), une étape de réécriture des requêtes est nécessaire. La réécriture pour les IJB consiste à ajouter un $Hint$ dans la clause SELECT pour forcer l'utilisation des index Créés[5]. La réécriture pour la FH consiste à identifier les fragments valides pour chaque requête, de réécrire la requête sur chacun de ces fragments et enfin faire l'union des résultats obtenus.

4 Fonctionnalités de l'outil

Nous présentons dans cette section les principales fonctionnalités de l'outil à travers son utilisation sur un entrepôt de données réel issu du Benchmark Apb-1 [16]. Le schéma en étoile que nous avons dégagé à partir de ce banc d'essais est constitué d'une table de faits *Actvars* (24 786 000 n-uplets) et de quatre tables de dimension, *Prodlevel* (9 000 n-uplets), *Custlevel* (900 n-uplets), *Timelevel* (24 n-uplets) et *Chanlevel* (9 n-uplets).

Pour aider l'AED dans l'optimisation de l'entrepôt de données, l'outil assure quatre principales fonctionnalités : visualisation de l'état de l'entrepôt, la préparation de l'optimisation, la fragmentation de l'entrepôt et l'indexation de l'entrepôt (fragmenté ou non).

4.1 Visualisation de l'état de l'entrepôt

La visualisation de l'état de l'entrepôt permet à l'AED de connaître le schéma de l'entrepôt de données, les tables de dimension, la table des faits ainsi que

5. Le hint *INDEX* dans une requête permet de forcer l'utilisation d'un ou plusieurs index dans le plan d'exécution d'une requête.

certaines statistiques sur ces tables. La visualisation permet aussi d'afficher les techniques d'optimisation déjà créées sur l'entrepôt de données. Toutes ces informations sont collectées grâce au module de d'interrogation de la méta-base. Cette visualisation permet à l'administrateur d'avoir une vue globale sur son entrepôt avant de commencer un processus d'optimisation. La figure 3(a) montre un exemple de visualisation où les tables, les attributs ainsi que les techniques d'optimisation créées sont affichés. La figure 3(b) montre un ensemble de statistiques collectées sur les objets de l'entrepôt.

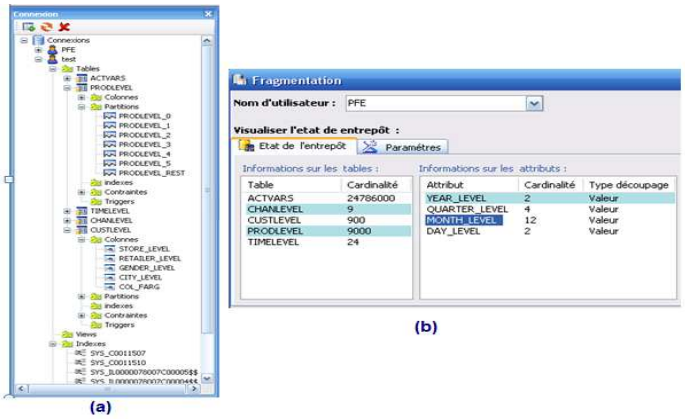

Figure 3. (a) Visualisation des objets de l'entrepôt, (b) visualisation des statistiques

4.2 Préparation de l'optimisation

La préparation de l'optimisation vise à collecter les informations nécessaires pour effectuer cette optimisation. Elle concerne la préparation de la charge de requêtes Q utilisée pour effectuer l'optimisation, le choix du mode de sélection et la définition des paramètres physique. La figure 4 montre l'interface permettant de gérer la charge des requêtes. L'outil permet aussi d'ajouter, modifier et supprimer une requête et de vérifier sa syntaxe. L'outil supporte deux modes de sélection : *isolé* et *multiple*. Dans le mode isolé l'AED peut utiliser la fragmentation seule (FHSEULE) ou l'indexation seule (IJBSEULS) pour optimiser son entrepôt. La sélection multiple consiste à utiliser les deux techniques en fragmentant l'entrepôt en un ensemble de fragments ensuite indexer ces derniers. L'outil permet à l'AED de fixer certains paramètres physiques comme la taille du buffer et la taille de la page système.

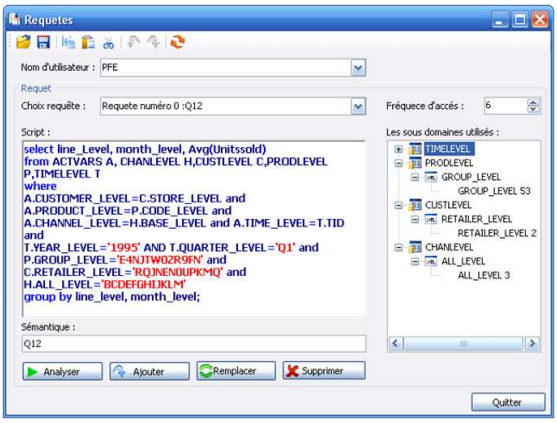

Figure 4. Interface de gestion de la charge de requêtes

4.3 Le partitionnement de l'entrepôt

Le partitionnement de l'entrepôt consiste à fragmenter les tables de dimension par la FH primaire ensuite la table des faits par la FH dérivée. L'AED commence par choisir le nombre de fragments maximum (W) ensuite il choisit s'il veut une fragmentation personnalisée ou non. S'il choisit la fragmentation non personnalisée alors *OptAssist* partitionne l'entrepôt de données en utilisant tous les attributs candidats et un algorithme de fragmentation par défaut. La fragmentation personnalisée, *offre plus de liberté* à l'AED dans le processus de sélection. Il peut choisir les tables et les attributs de dimension participant au processus de fragmentation. Il doit choisir l'algorithme de partitionnement (AG, RS ou HC) et établir ses paramètres. La figure 5 représente la zone de choix des algorithmes et de leurs paramètres. Pour chaque algorithme sélectionné, *OptAssist* active les paramètres correspondants et donne la possibilité à l'AED de les modifier. Pour illustrer la fragmentation personnalisée nous considérons le cas où l'AED choisit d'éliminer certains attributs et tables du processus de fragmentation. La figure 7 représente l'interface de personnalisation de la fragmentation. Si l'AED choisit de personnaliser sa fragmentation, alors *OptAssist* lui donne la possibilité de choisir les attributs et les tables de dimension candidats pour le processus de fragmentation. Dans cette figure, l'AED a éliminé la table *CustLevel*, un attribut de la table *TimeLevel* et trois attributs de la table *ProdLevel* du processus de fragmentation. Après avoir choisi les tables et les attributs, l'AED choisit l'algorithme de recuit simulé, fixe W à 100 et lance l'exécution. Le schéma de fragmentation obtenu avec cette personnalisation génère 72 fragments et un coût d'exécution de requêtes représentant un gain de l'ordre de 8,8% par rapport à la fragmentation non personnalisée.

Une fois le schéma de l'entrepôt fragmenté, l'AED peut visualiser une recommandation proposée par *OptAssist*. Elle contient le nombre de fragments générés, les attributs utilisés par la FH, les tables de dimension fragmentées, une estimation du nombre d'entrées-sorties nécessaires pour exécuter la charge de requêtes, le nombre de fragments de chaque table de dimension, le gain de performance obtenu par cette fragmentation (par rapport à un schéma non fragmenté), etc. La figure 8 montre l'interface affichant les attributs utilisés pour fragmenter l'entrepôt (quatre attributs parmi douze ont été utilisés : *Line_level, Year_level, Month_level* et *All_level*). Si l'AED n'est pas satisfait de cette recommandation, il peut revenir à l'étape précédente et changer les différents paramètres (rechoisir les attributs et les tables ou l'algorithme, ses paramètres, etc.). Ce retour en arrière est primordial dans la phase de conception physique. Une fois satisfait, l'AED demande à l'outil de générer les scripts de fragmentation. Ceux-ci pourront par la suite être appliqués sur l'entrepôt de données.

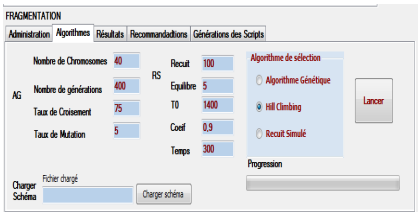

Figure 5. Choix et configuration des algorithmes

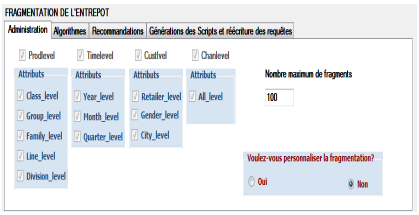

Figure 6. Fragmentation non personnalisée

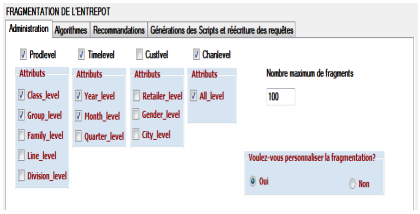

Figure 7. Personnalisation de la fragmentation

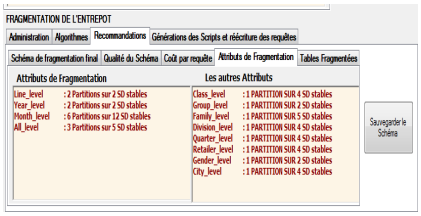

Figure 8. Attributs de fragmentation

Dans le cas où l'AED est satisfait du schéma sélectionné, il demande la génération des scripts et la réécriture des requêtes sur ce schéma. Pour fragmenter physiquement l'entrepôt, l'AED exécute les scripts générés sur l'entrepôt d'origine, ce dernier sera remplacé par l'entrepôt fragmenté.

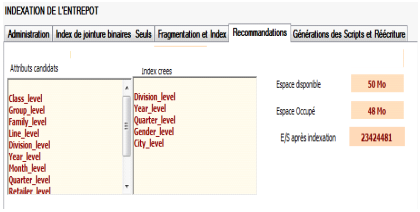

Figure 9. Recommandations d'indexation dans IJBSEULS

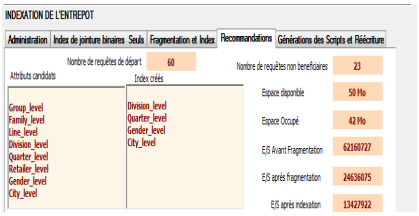

Figure 10. Recommandation d'indexation dans HP&IJB

4.4 L'indexation de l'entrepôt

La tâche d'indexation permet de sélectionner un ensemble d'IJB sur l'entrepôt de données des IJB. L'outil supporte deux manières d'indexation : isolée (IJBSEULS) et multiple (FH&IJB). Dans le cas d'une indexation de type IJBSEULS, l'AED doit d'abord choisir les attributs indexables candidats et l'espace de stockage S. Comme pour la FH, deux types d'indexation sont possibles : *l'indexation non personnalisée* et *l'indexation personnalisée*. Les recommandations générées contiennent des informations concernant le gain apporté par les index, les attributs indexés, le coût de stockage exigé par les index sélectionnés, etc. Pour illustrer ce cas, nous considérons que l'AED choisit de faire une indexation non personnalisée avec un seuil de 50 Mo. La figure 9 présente l'interface dédiée à aux recommandations générées sur les index créés (les attributs indexés, l'espace occupé, l'espace disponible, etc.). Parmi les douze attributs indexables, cinq attributs ont été utilisés pour créer cinq IJB occupant 48 Mo d'espace disque.

L'indexation dans FH&IJB consiste à indexer l'entrepôt après son fragmentation. La différence entre l'indexation dans IJBSEULS et FH&IJB réside dans le choix des attributs candidats. Au lieu de considérer les attributs candidats à partir de la configuration initiale de l'entrepôt de données, l'AED doit identifier ces attributs parmi les attributs d'indexation non utilisés par la fragmentation. Pour illustrer cette indexation, considérons que l'AED cherche à indexer l'entrepôt de données fragmenté. Il choisit le mode de sélection multiple, l'outil désactive automatiquement les attributs utilisés pour fragmenter l'entrepôt, puisqu'ils ne sont pas utilisés pour indexer l'entrepôt. L'AED choisit alors les attributs candidats à l'indexation et un seuil d'espace de stockage de 50 Mo, puis lance l'algorithme de sélection. La figure 10 montre les recommandations d'indexation après la sélection d'une configuration d'index. Nous trouvons plusieurs informations, comme le nombre de requêtes non bénéficiaires de la fragmentation, les attributs indexés, l'espace de stockage des index sélectionnés, le coût d'exécution avant et après indexation, etc. De la même façon que pour la fragmentation, s'il est satisfait des recommandations, il lance la génération des scripts de création des index binaires, sinon il peut revenir en arrière pour modifier ses choix.

5 Conclusion

La tâche d'optimisation de la couche physique dans les entrepôts de données est devenue un enjeu majeur. Cela est dû aux caractéristiques des entrepôts : la volumétrie, la complexité des requêtes, les exigences de temps de réponse raisonnable et la gestion de l'évolution de l'entrepôt. Dans cet environnent, nous avons mis en évidence les difficultés qu'un administrateur pourrait rencontrer durant l'optimisation de la couche physique. Ces difficultés sont multiples, car elles concernent plusieurs niveaux de conception : le choix des techniques d'optimisation pertinentes pour l'ensemble de requêtes à optimiser, le choix de la nature de sélection des techniques d'optimisation et le choix des algorithmes et leur paramètres. Vu ces difficultés, nous avons identifié le besoin de développer un outil d'assistance de l'administrateur qui permet de répondre aux besoins en termes de choix possibles. Nous avons proposé l'outil, *OptAssist*, offrant trois techniques d'optimisation : la FH primaire, FH dérivée et les IJB. Il permet à l'administrateur de choisir les différents algorithmes et leurs paramètres. Il peut alors utiliser ces techniques d'une manière isolée ou multiple. Une autre particularité de *OptAssist* est le fait de proposer des sélections personnalisées et non personnalisées des structures d'optimisation. Une des perspectives de cet outil est son extension en considérant d'autres techniques d'optimisation comme les vues matérialisées, la fragmentation verticale, traitement parallèle.

Références

1. Bellatreche, L., Boukhalfa, K., Abdalla, H.I. : Saga : A combination of genetic and simulated annealing algorithms for physical data warehouse design. in 23rd British National Conference on Databases (212-219) (July 2006)
2. Bellatreche, L., Missaoui, R., Necir, H., Drias, H. : A data mining approach for selecting bitmap join indices. Journal of Computing Science and Engineering **2**(1) (January 2008) 206–223
3. Chaudhuri, S. : Index selection for databases : A hardness study and a principled heuristic solution. IEEE Transactions on Knowledge and Data Engineering **16**(11) (November 2004) 1313–1323
4. Johnson, T. : Performance measurements of compressed bitmap indices. Proceedings of the International Conference on Very Large Databases (1999)
5. Chee-Yong, C. : Indexing techniques in decision support systems. Phd. thesis, University of Wisconsin - Madison (1999)
6. Aouiche, K., Darmont, J., Boussaid, O., Bentayeb, F. : Automatic Selection of Bitmap Join Indexes in Data Warehouses. 7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK 05) (August 2005)
7. Sanjay, A., Surajit, C., Narasayya, V.R. : Automated selection of materialized views and indexes in microsoft sql server. Proceedings of the International Conference on Very Large Databases (September 2000) 496–505
8. Talebi, Z.A., Chirkova, R., Fathi, Y., Stallmann, M. : Exact and inexact methods for selecting views and indexes for olap performance improvement. 11th International Conference on Extending Database Technology (EDBT'08) (Mars 2008)

9. Agrawal, S. : Automatic sql tuning in oracle 10g. In Proceedings of the 30th International Conference on Very Large Databases (VLDB) (2004)
10. Zilio, D.C., Rao, J., Lightstone, S., Lohman, G.M., Storm, A., Garcia-Arellano, C., Fadden, S. : Db2 design advisor : Integrated automatic physical database design. Proceedings of the International Conference on Very Large Databases (August 2004) 1087–1097
11. Agrawal, S. : Database tuning advisor for microsoft sql server 2005. In Proceedings of the 30th International Conference on Very Large Databases (VLDB) (2004)
12. Valentin, G., Zuliani, M., Zilio, D., Lohman, G., Skelley, A. : Db2 advisor : An optimizer smart enough to recommend its own indexes. In : 16th International Conference on Data Engineering (ICDE 00), San Diego, USA. (2000) 101–110
13. Gebaly, K.E., Aboulnaga, A. : Robustness in automatic physical database design. in 11th International Conference on Extending Database Technology (EDBT'08), March (2008)
14. Boukhalfa, K. : De la conception physique aux outils d'administration et de tuning des entrepôts de données. Ph.d. thesis, Ecole Nationale Supérieure de Mécanique et d'aéronautique Poitiers et Université de Poitiers (July 2009)
15. Bellatreche, L., Boukhalfa, K., Richard, P. : Horizontal partitioning in data warehouse : Hardness study, selection algorithms and validation on oracle10g. in 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008) (September 2008) 87–96
16. Council, O. : Apb-1 olap benchmark, release ii. http ://www.olapcouncil.org/ research/ resrchly.htm (1998)