

Administration et Tuning des Entrepôts de Données

Optimisation par Index de Jointure Binaires et Fragmentation Horizontale

Rym Bouchakri
Ecole Doctorale STIC
ESI
Alger, Algérie
r_bouchakri@esi.dz

Ladjet Bellatreche
ENSMA
Université de Poitiers
France
bellatreche@ensma.fr

Kamel Boukhalfa
USTHB
Alger, Algérie
boukhalk@ensma.fr

Abstract : Le volume d'information contenu dans un entrepôt de données est destiné à s'accroître sans cesse, augmentant ainsi la complexité des requêtes décisionnelles. Pour y remédier, l'administrateur doit, durant la phase de conception physique et de tuning de l'entrepôt, effectuer une sélection de structures d'optimisation (index, vues matérialisées ou fragmentation), puis assurer leur gestion et maintenance, afin d'améliorer les performances d'exécution des requêtes. Pour satisfaire un maximum de requêtes, il est préférable d'opter pour une sélection combinée de plusieurs techniques, en exploitant leur similarité. Dans notre travail, nous présentons un état de l'art sur les entrepôts de données et les techniques d'optimisation. Nous mettons l'accent sur la fragmentation horizontale et les index de jointure binaires qui partagent une même ressource, à savoir les attributs de sélection extraits des requêtes décisionnelles. Nous proposons une approche de partage de ces attributs, entre index et fragmentation, en se basant sur une classification par algorithme « k-means ». Nous terminons par une étude expérimentale et des tests comparatifs sur un entrepôt de données réel qui montrent l'intérêt de notre approche.

Mots clés: Tuning des Entrepôts de Données, Optimisation des Requêtes, Fragmentation Horizontale, Index de jointure binaires.

I. INTRODUCTION

Dans l'informatique décisionnelle, les entreprises exploitent de grands volumes de données stockés dans des entrepôts de données. Ces données sont issues de différentes sources hétérogènes et leurs volumes sont destinés à augmenter sans cesse. L'exemple type est le domaine des télécommunications (opérateurs téléphoniques) où l'entreposage des données constitue un axe vital autour duquel tourne toute la stratégie de l'entreprise de télécom.

Le concept d'entrepôt de données (ED) a été formalisé en 1990 par Bill Inmon comme « une base de données orientée sujet, intégrée et contenant des informations historisées, non volatiles destinées aux processus d'aide à la décision » [1]. Les données dans l'ED sont organisées en *cube multidimensionnel* où chaque *dimension* est un axe d'analyse et chaque cellule est le *fait* analysé [2]. Pour le stockage physique, un modèle relationnel est introduit, sous le nom de ROLAP (Relational On-Line Analytical Processing), où chaque fait est stocké dans une *table de fait*, très volumineuse (Giga ou Terra Octets), liée par clés étrangères à plusieurs *tables de dimension* formant ainsi un *schéma en étoile* (figure1).

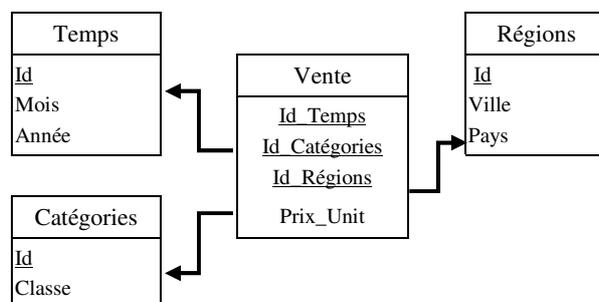


Figure 1: Modèle relationnel pour le stockage d'un ED

Les requêtes décisionnelles exécutées sur les EDs sont très complexes. Elles contiennent des jointures, des sélections et des agrégations. Ces jointures sont appelées *jointures en étoiles* car elles passent toutes par la table de fait. Quand aux opérations de sélections, elles sont appelées *prédicats de sélection*, et sont effectuées sur des *attributs dimensions* appelés *attributs de sélections* (AS).

Exemple 1: Soit la requête suivante exécutée sur un entrepôt, dont le schéma est donné par la figure 1 :

```
SELECT Id_Temps, Sum(Prix_Unit)
FROM Vente V, Régions R, Catégories C
WHERE V.Id_Catégories=C.Id AND V.Id_Régions=R.Id
AND R.Ville='Alger' AND C.Classe='A'
GROUP BY Id_Temps
```

La requête contient deux AS : « Ville » et « Classe » et donc deux prédicats de sélection « Ville='Alger' » et « Classe='A' ».

Devant la complexité et un temps de réponse long (heures, jours) de ce type de requête, la tâche de l'administrateur des ED devient très difficile. En effet, l'administrateur requière une bonne connaissance des structures d'optimisation (index, vues matérialisées, etc.), des méthodes de conceptions logiques et physiques afin de choisir la politique de conception optimale.

A. Techniques d'optimisation des requêtes

Afin d'adopter une politique d'optimisation d'un ED, il faut choisir la technique d'optimisation, la nature de sélection et l'algorithme de sélection [3], ceci afin d'assurer une réduction de la complexité des requêtes et du *coût d'exécution*¹.

Il existe deux classes de structures d'optimisation [1] : (1) redondantes dont l'utilisation produit une duplication des données : les index et les vues matérialisées (2) non redondantes ne dupliquent pas les données mais réorganisent leur représentation physique comme la fragmentation horizontale.

¹: le coût d'exécution représente le nombre d'entrées/sorties nécessaires pour charger les données afin de calculer les résultats de la requête.

Afin de choisir les techniques d'optimisation à implémentée sur un ED, deux types de sélection existent. (1) la sélection isolée qui consiste à implémenter une technique d'optimisation à la fois. (2) la sélection combinée qui permet de réaliser l'implémentation conjointe de deux ou plusieurs techniques [3].

Pour sélectionner les techniques d'optimisation, deux catégories d'algorithmes existent : (1) les algorithmes simples ou gloutons. (2) les algorithmes complexes, comme les heuristiques et les algorithmes génétiques [3]. Ces algorithmes sont munis d'un modèle de coût qui permet d'estimer le coût d'exécution des requêtes. Il peut être calculé soit par l'optimiseur du SGBD soit par une fonction mathématique estimée.

Les index : Les index optimisent les requêtes en minimisant le volume de données à exploiter dans les calculs. Ils ont été largement étudié dans le contexte de bases de données [4, 5, 6, 7, 8]. Pour les ED, l'opération la plus coûteuse est la jointure en étoile. Ainsi, les index les plus efficaces sont les index de jointures en étoiles ou les index de jointures binaires IJB [9,1] (cf. Section II.1). Deux types d'algorithmes ont été utilisés pour sélectionner les IJB: les algorithmes gloutons [10] et ceux basés sur la fouille de données [9, 11].

La fragmentation : permet de répartir les données d'un ED en plusieurs partitions pouvant être accédées séparément. Elle peut être soit verticale (FV) ou horizontale (FH) et est calculée sur les attributs de sélection. La FV répartie les attributs d'une relation afin de générer plusieurs tables [12, 13]. Elle optimise l'exécution des requêtes de projection mais nécessite des jointures pour retrouver la relation initiale, c'est pour cela que la FH est la plus appropriée pour les ED. Elle répartie les tuples de données en plusieurs fragments définis chacun par un ensemble de prédicats de sélection. Dans les ED, le problème de sélection d'un schéma de fragmentation (SF) vise à choisir un SF qui minimise le coût d'exécution et le nombre de fragments faits générés. Etant un problème NP-Complet [11], des travaux comme [14, 15] adoptent le Recuit simulé, le Hill Climbing. D'autres utilisent la fouille de données afin de définir les fragments des tables [16].

B. Approche d'optimisation adoptée dans cet article

Nous abordons dans ce travail le concept d'administration et *Tuning des ED*¹, plus précisément l'optimisation, par sélection combinée de FH et d'IJB, de requêtes décisionnelles les plus fréquemment exécutées sur un ED.

Nous proposons une nouvelle approche de sélection combinée basée sur la classification des attributs de sélection. La classification est réalisée à l'aide d'une technique de Datamining à savoir l'algorithme de classification « k-means ». Une fois l'approche validée, nous présentons une étude expérimentale sur un ED réel, qui montre l'optimisation d'une charge de requêtes par IJB et FH et l'intérêt de la classification.

Ce document est rédigé dans le cadre de préparation de la thèse magistère au sein de l'Ecole Nationale Supérieure d'Informatique (ESI). Il est organisé en quatre sections. Dans la section II, nous présentons la démarche de sélection combinée des IJB et FH qui existe dans la littérature. La section III, est consacrée à la description de notre approche de sélection combinée par classification. Enfin, la section IV présente l'étude expérimentale qui montre l'apport de la classification au coût d'exécution des requêtes sur un ED réel.

II. SELECTION COMBINEE DES INDEX DE JOINTURE BINAIRES ET DE LA FRAGMENTATION HORIZONTALE

Dans nos travaux, nous présentons une démarche de sélection combinée d'IJB et FH. Nous avons fait ce choix car : (1) les IJB et la FH sont les plus approprié dans le contexte d'ED, ils permettent d'optimiser les jointures en étoiles avec prédicats de sélection sur les dimensions. (2) les deux techniques présentent des similarités que nous allons montrer dans ce qui suit.

A. Les index de jointures binaires

Les IJB sont des index Bitmaps calculés sur la table de fait. Ils sont définis sur les attributs de sélection extraits à partir de la charge de requête à optimiser.

Exemple 2 : Soit un ED représenté par le schéma de la figure 1. Un IJB défini sur l'attribut « Pays » est représenté par la figure

Table Vente	IJB_Vente_Ville			
	Algérie	Tunisie	Syrie	Liban
Vente1	1	0	0	0
Vente2	0	0	1	0
Vente3	1	0	0	0

Figure 2: IJB sur la table de fait Vente

Le problème de sélection d'un ensemble d'IJB est NP complet et dans la littérature, on adopte des heuristiques pour trouver la configuration d'index optimale [2, 11]. Ce problème est formalisé comme suit : Soit :

- Un ED ayant une fait F et p dimensions $D = \{D1...Dp\}$
- $Q = \{Q_1 ... Q_m\}$ une charge de requêtes
- $AS = \{A_1 ... A_n\}$ les attributs de sélection à partir de Q .
- S espace de stockage des index.

Il faut trouver une combinaison d'IJB sur A tel que : (1) le coût d'exécution de la charge de requêtes est minimisé, (2) l'espace alloué pour les index ne dépasse pas S .

B. La fragmentation horizontale

Deux types de FH existent : (1) la FH primaire répartie les tuples de données d'une table suivant ses attributs. (2) la FH dérivée répartie les données d'une table selon les attributs d'une autre table [15]. Les travaux sur la fragmentation des ED proposent d'effectuer une FH primaire sur les dimensions, puis de fragmenter la table de fait par une FH dérivée [14, 15]. Chaque fragment « fait » muni de ses fragments dimension forment *un sous schéma en étoile*. La formulation du problème de sélection d'un SF est la suivante : Soit :

- Un ED ayant une fait F et p dimensions $D = \{D1...Dp\}$
- $Q = \{Q_1 ... Q_m\}$ une charge de requêtes
- $AS = \{A_1 ... A_n\}$ les attributs de sélection à partir de Q .
- W le nombre de fragment maximum

Il faut trouver un SF sur A tel que : (1) le coût d'exécution de la charge de requêtes est minimisé, (2) le nombre de fragment générés ne dépasse pas W .

Les auteurs dans [14, 15] proposent une sélection d'un SF basée sur le découpage des domaines d'attributs en sous domaines. Chaque attribut muni d'un sous domaine donne un prédicat de sélection qui caractérise un fragment. Pour ce faire, les auteurs utilisent des heuristiques comme le Recuit simulé ou Hill Climbing.

¹ : Le Tuning des ED vise à sélectionner des structures d'optimisation, de maintenir leur cohérence et de prévoir les besoins d'optimisation futurs.

Exemple 3: Soit un ED de la figure 1. Soit l'attribut de FH « Pays » = {Algérie, Tunisie, Syrie}. Le découpage en sous domaine « Pays » = {Algérie, Tunisie} et {Syrie}. La FH donne trois fragments fait (trois sous schémas en étoile), présenté par la figure 3 (les ventes de l'Algérie et la Tunisie).

Régions1			Vente1	
Id	Ville	Pays	Id_Régions	...
Reg06	Alger	Algérie	Reg06	
Reg08	Oran	Tunisie	Reg06	
Reg16	Bejaia	Algérie	Reg16	

Figure 3: Exemple d'un sous schéma en étoile après fragmentation

C. Similarité entre IJB et FH

La FH et les IJB permettent de répartir les données d'une table de fait suivant la répartition d'une dimension, afin de réduire le coût d'exécution des jointures en étoile [10]. De plus, ces techniques partagent le même AS. Ils sont donc souvent sélectionnés de manière combinée.

D. Sélection combinée d'IJB et de FH

La sélection combinée de la FH et des IJB permet de trouver un compromis entre les deux contraintes définies sur les deux techniques d'optimisation, à savoir : nombre maximum de fragment fait et espace maximum de stockage des index. Le problème de sélection combinée peut être formulé comme suit :

Soit :

- Un ED ayant une fait F et p dimensions $D = \{D1...Dp\}$
- $Q = \{Q_1 ... Q_m\}$ une charge de requête
- $A = \{A_1 ... A_n\}$ les attributs de sélection à partir de Q .
- S espace de stockage des index.
- W le nombre de fragment maximum

Il faut trouver une combinaison de FH et d'IJB sur A tel que :

- (1) Le coût d'exécution de la charge de requêtes est minimisé.
- (2) L'espace alloué pour les index ne dépasse pas S .
- (3) Le nombre de fragment générés ne dépasse pas W .

Les travaux de Bellatreche et al [3, 10] proposent d'effectuer une sélection combinée entre FH et IJB. Ils présentent alors dans [10] un outil d'assistance à l'administration et au Tuning appelé ParAdmin qui gère la FHP, la FHD et les IJB.

III. NOTRE APPROCHE DE SELECTION COMBINEE D'IJB ET FH : CLASSIFICATION DES ATTRIBUTS DE SELECTION

A. Motivation

Dans l'approche précédente, la FH est sélectionnée sur tout les AS. Ceux qui restent permettent de définir les IJB. Cela dit, ces index peuvent s'avérer inutiles. En effet, si l'index possède une faible sélectivité¹, l'optimiseur du SGBD jugera préférable d'utiliser la table directement. D'autre part, des attributs peuvent être choisis pour la FH alors que leur sélection pour un IJB donnerait un meilleur résultat. Ainsi, nous proposons de répartir les attributs entre IJB et FH, avant de sélectionner les structures, en deux ensembles: Classe_IJB et Classe_FH. Cette classification a deux avantages. (1) Réduire le nombre d'attributs à introduire dans la fragmentation. (2) Choisir, pour chaque technique d'optimisation les attributs les plus adaptés surtout pour les IJB.

¹: la faible sélectivité signifie que la sélection de données par index donne un nombre de tuples tout aussi grand que le nombre de tuple de la table.

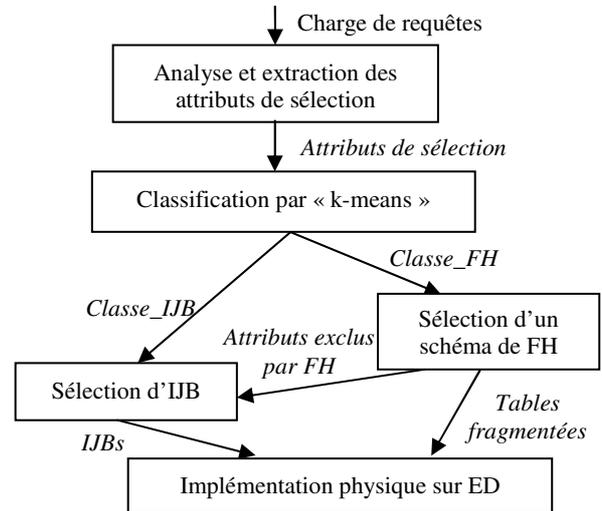


Figure 4: processus de sélection d'IJB et FH avec classification

B. Formalisation du problème

La sélection d'un schéma d'optimisation combinée (FH-IJB) avec classification des attributs est présentée comme suit. Il est à noté que le nombre de classifications possibles d'un ensemble en deux classes est donné par la formule suivante :

$$nb_{classification} = 2^n \quad (1)$$

Avec n le nombre d'attributs de sélection.

Etant donnée

- Un ED ayant une fait F et p dimensions $D = \{D1...Dp\}$
- $Q = \{Q_1 ... Q_m\}$ une charge de requête
- $AS = \{A_1 ... A_n\}$ les attributs de sélection définis de Q .
- S espace de stockage des index.
- W le nombre de fragment maximum

Il faut répartir les attributs de sélection entre FH et IJB tel que : (1) La sélection combinée des deux techniques, définies chacune sur son ensemble d'attributs, permet de réduire le coût d'exécution des requêtes. (2) L'espace réservée aux index ne dépasse pas S . (3) Le nombre de fragments ne dépasse pas W .

C. Déroulement du processus de sélection

On se basant sur les travaux de Bellatreche et al et Boukhalfa et al, [3, 10, 14, 15] pour la sélection d'un schéma d'optimisation avec FH et d'IJB, nous proposons l'approche suivante (figure 4): (1) Extraire, à partir d'une charge de requêtes, les attributs de sélection. (2) Classifier ces attributs entre FH et IJB par la méthode de classification « k-means » avec $k=2$. (3) Définir un schéma de fragmentation sur la classe FH. (4) Définir les IJB sur la classe IJB à laquelle on ajoute les attributs non sélectionnés par FH.

Notre choix s'est porté sur « k-means » car cet algorithme est simple à utiliser, n'est pas gourmand en temps d'exécution et répond bien à nos besoins de classification.

D. Démarche de classification

La classification d'AS s'effectue en deux phases : (1) Affecter un poids de classification pour chaque attribut. (2) A travers « k-means » et ce poids, classifier l'AS en deux classes. Afin d'effectuer la classification, nous avons définis des critères de décision suivants :

- *Nombre de requêtes pour un attribut Nbr*: Nous allons favoriser, pour la FH, les attributs qui figurent dans un maximum de requêtes, afin d'être sûr d'obtenir une amélioration du coût pour ces requêtes là.
- *Cardinalité de l'attribut Card*: Plus la cardinalité est grande, plus la taille de chaque fragment est réduite, ainsi la taille des données chargées est réduite lors d'exécution des requêtes.
- *Facteur de sélectivité FS* : les index à faible sélectivité peuvent ne pas être utilisés lors de l'exécution des requêtes. On définit alors le facteur de sélectivité d'un attribut $FS(A_i)$ la moyenne de la sélectivité des valeurs de A_i . Plus ce facteur est grand, plus la sélectivité est faible et plus l'attribut est candidat à la FH.

1) *Calcul du poids de classification* : Afin de classer les attributs, nous avons défini une métrique qui affecte à chaque attribut un poids, se basant sur les facteurs que nous venons de définir. Le calcul du poids est effectué comme suit :

$$poids_i = Nbr_i + FS_i + Card_i \quad (2)$$

Où Nbr_i , FS_i , $Card_i$ représentent respectivement le nombre de requêtes, le facteur de sélectivité et la cardinalité de l'attribut i .

Remarque : Pour que le poids soit cohérent, nous avons accompli la normalisation des données. L'échantillon de données de chaque facteur doit suivre la Loi Normale Centrée Réduite (1, 0) [17]

Exemple 4 : Soit un ED représenté par le schéma de la figure 1 et $AS = \{Mois, Année, Ville, Pays, Classe\}$. Soit une charge de requêtes à optimiser à partir de laquelle les facteurs Nbr , FS , $Card$ et le poids sont calculés. Le tableau résume le calcul du poids pour chaque attribut.

Attribut	Nbr	FS	Card	Nbr Normalisé	FS Normalisé	Card Normalisé	Poids
Année	11	0,2	23	1,14	0,53	0,01	1,70
Mois	5	0,33	12	0,26	1,41	-0,30	1,37
Ville	6	0,1	55	0,41	-0,13	0,94	1,22
Pays	9	0,09	20	0,85	-0,20	-0,07	0,57
Classe	3	0,02	62	-0,02	-0,67	1,14	0,44

Tableau 1 : Calcul du poids des attributs dimension

2) *Classification par « k-means »* : Afin d'exécuter l'algorithme « k-means », il faut introduire plusieurs paramètres qui sont : (1) les coordonnées dans \mathbb{R}^2 pour chaque attribut, (2) le nombre d'itération, (3) le nombre de classe (dans notre cas $k=2$). La représentation graphique de ces attributs munis de leurs poids est illustrée dans la figure 5, elle montre une répartition des attributs en deux ensembles.

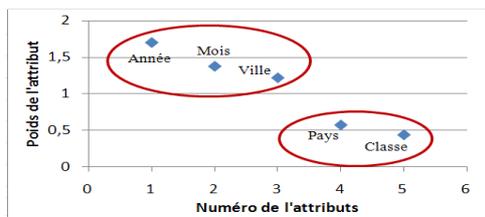


Figure 5 : Répartition des attributs en fonction de leurs poids

1 : La fonction objectif et les paramètres du génétique sont issus des travaux de Bellatreche et al (75% de taux de croisement et 5% de taux de mutation).

IV. ENVIRONNEMENT EXPERIMENTAL

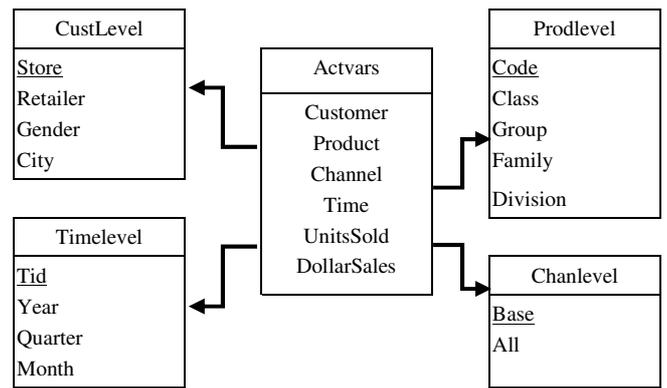


Figure 6 : Schéma en étoile de l'entrepôt expérimental

L'étude est réalisée sur une charge de **47 requêtes**, exécutées sur un ED réel issu du benchmark APB1 [18] sous **Oracle 11g**, avec une machine Intel Core2Duo et une mémoire 2go. Le schéma en étoile (figure6) est constitué d'une faits Actvars (24786000 tuples) et de quatre dimensions, Prodlevel (9000 tuples), Custlevel (900 tuples), Timelevel (24 tuples) et Chanlevel (9 tuples).

Afin d'effectuer la sélection des structures d'optimisation, nous avons implémenté une application sous l'**IDE Eclipse**. Nous avons intégré dans notre application deux **API JAVA**, l'une permet d'implémenter l'algorithme de classification « K-Means », la seconde, nommé JGAP (Java Genetic Algorithms Package), est un Framework qui implémente un algorithme génétique pour la sélection des techniques d'optimisation.

Dans un premier temps, nous allons présenter notre sélection d'IJB et FH avec classification d'AS. Par la suite, nous allons effectuer une série de test afin de montrer l'intérêt de notre démarche.

A. Démarche d'Optimisation Avec Classification OAC

1) *Classification des attributs* : Nous avons calculé les coordonnées des attributs et fixé le nombre d'itération à 50. Les classes obtenues sont : Classe_FH = {Gender, Month, Year, All, Quarter, Group}, Classe_IJB = {Family, Division, Class, City, Retailer}.

2) *Sélection des structures d'optimisation* : La sélection de FH et IJB est faite comme suit :

a) *Fragmentation* : suivant les travaux de Bellatreche et al [13] et pour $w=70$ fragments, l'algorithme génétique¹ sélectionne un schéma de FH avec 8 fragments sur Timelevel (Month, Year, Quarter), 2 fragments sur Chanlevel (All) et 4 fragments sur Custlevel (Gender, City), ceci donne 64 fragments Fait.

b) *Indexation* : s'il ya des attributs non choisis par la FH, ils sont ajoutés à la Classe_IJB. Sur cette dernière, le génétique sélectionne quatre IJB sur cinq qui sont : {Family, Division, City, Retailer}, ceci sous une contrainte d'espace $S=500 mo$.

3) *Implémentation de la FH et des IJB* : L'application génère les scripts et réalise l'implémentation physique des structures.

B. Démarche d'Optimisation Simple OS

Afin d'évaluer les performances de l'optimisation par classification, nous avons effectué l'optimisation de l'ED simple. Elle consiste à effectuer les étapes présentées dans la partie IV.2 sans effectuer la classification. L'exécution de la FH donne : 4 fragments sur Timelevel (Month, Quarter), 8 fragments sur Prodlevel (Group, Family, Class) et 2 fragments sur Custlevel (Retailer), ceci donne 64 fragments Fait. Les index sélectionnés par génétique concernent les attributs suivants : {Gender, Year, All, City, Division}.

C. Tests et résultats

Nous avons effectué plusieurs tests, dans chaque test, nous avons comparé entre OAC et OS afin de montrer la méthode qui donne la meilleure optimisation. Les coûts des requêtes, après chaque optimisation, sont calculés par l'optimiseur d'Oracle 11g.

1) Test 1 : Variation du mode d'optimisation :

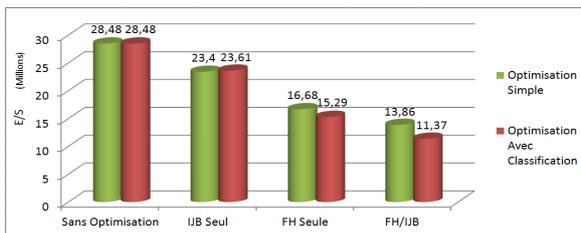


Figure 7 : Coût d'exécution avec différents modes sous Oracle 11g

Nous avons réalisé l'optimisation de l'ED selon différent mode : IJB Seul, FH Seule et FH et IJB ($S=500$ mo et $w=70$). Pour chaque mode, nous effectuons l'optimisation par OAC et OS. La figure 7 illustre les coûts d'exécution des requêtes.

Nous remarquons que le meilleur coût d'exécution est obtenu pour le mode mixte FH/IJB et pour la démarche avec classification OAC. En effet, le coût passe de 28.48 millions à 11.37 millions E/S (réduction du coût de 61%). Cela montre deux résultats essentiels :

- L'optimisation par FH et IJB apporte les meilleurs résultats.
- Notre démarche OAC permet une meilleure optimisation que la méthode simple.

Notons que pour IJB seul, la méthode simple présente un meilleur résultat. En effet, la méthode simple définit les IJB sur tout l'AS par contre notre démarche OAC définit l'indexation uniquement sur la classe IJB.

2) Test 2 : Variation de W :

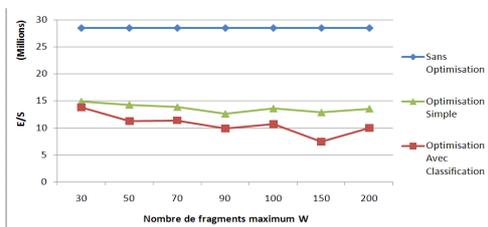


Figure 8 : Coût d'exécution par variation de W

Afin de voir l'influence de la contrainte W sur l'optimisation, nous l'avons varié et pour chaque valeur, nous effectuons une optimisation par la démarche OS et une par OAC. Nous avons relevé le coût d'exécution.

Les résultats qu'illustrent les figures 8 montrent que la meilleure optimisation est obtenue avec notre démarche OAC surtout pour $W = 90$ et 150 , car le coût est réduit de 68% et 75%.

3) Test 3 : Variation de l'ordre d'exécution pour FH/IJB :

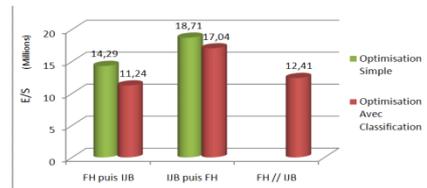


Figure 9 : différentes exécutions pour FH et IJB

Nous avons testé le changement de l'ordre d'exécution entre les deux sélections (avec $S=500$ mo et $W=50$). La figure 9 montre trois possibilités de sélections : FH puis IJB, IJB puis FH et FH//IJB. Nous remarquons que les meilleurs résultats sont pour FH puis IJB avec classification. En effet, FH//IJB empêche la prise en compte par IJB des attributs non choisis par FH. Concernant le cas IJB puis FH, les attributs sélectionnés par indexation peuvent donner lieu à des IJB non bénéfiques.

Nous constatons que la classification n'apporte pas un grand bénéfice dans le mode IJB puis FH, car les attributs non sélectionnés par l'indexation vont être ajoutés à la classe FH et vont fossé le choix du SF. Aussi, les résultats de l'optimisation simple dans le mode FH puis IJB (14,29 millions) sont meilleurs que ceux de la classification avec le mode IJB puis FH (17,04 millions). En effet, la FH de la méthode simple (FH puis IJB) s'effectue sur tout les AS, l'optimisation s'avère meilleure que de fragmenter sur un sous ensemble d'attributs.

Cela montre la force de la classification des attributs. En effet, et malgré le fait que la FH est défini sur un nombre réduit d'attributs, ces attributs constitue les meilleurs candidats pour la FH. D'autre part, ceux choisis pour les index sont plus adaptés aussi. Ce qui donne une meilleure optimisation globale.

4) Test 4 : Performance d'algorithmes de sélection $w = 70$:

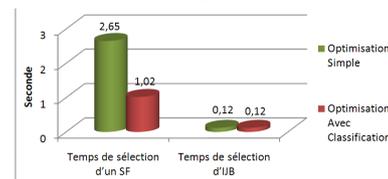


Figure 10 : Analyse des performances des algorithmes de sélection

Nous avons étudié les performances de sélection d'uns schéma d'optimisation par le génétique, ceci au point de vu temps d'exécution, avec $S=500$ mo et $w=70$. Selon la figure 10, nous notons que les performances d'exécution des algorithmes de sélection diffèrent sur la FH. En effet, la sélection pour OS est effectuée sur tous les attributs (11 attributs), pour OAC, seul les attributs de la Classe FH sont concernés (6 attributs).

5) Test 5 : Choix des Facteurs du Poids de classification :

Le poids de classification comporte trois facteurs : Nbr, FS et Card. Afin d'étudier leurs pertinences, nous avons effectué l'optimisation par OAC en faisant varier la formulation du poids. Cela donne sept combinaisons possibles (figure 11).

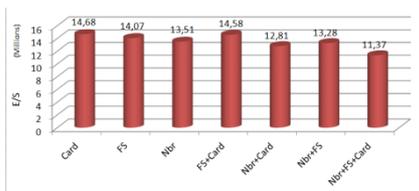


Figure 11 : Optimisation selon la formulation du poids de classification

Nous remarquons que la meilleure optimisation de la charge de requêtes est de 11,37 millions, elle est obtenue pour un poids de classification qui comporte tout les facteurs. Cela montre que les trois facteurs sont pertinents pour la classification.

D. Bilan général

Notre étude permet de faire les constatations suivantes :

1) Bilan sur la sélection combinée

a) Les techniques les plus adéquats pour optimiser une charge de requêtes exécutée sur un ED en étoile sont la FH et IJB. Ils permettent d'optimiser les jointures en étoile entre table de fait volumineuse et plusieurs dimensions.

b) La sélection combinée de structures d'optimisation apporte de meilleures améliorations, car chaque technique permet de palier aux manques de l'autre.

c) L'ordre d'exécution des deux sélections est la sélection d'un schéma de fragmentation puis la sélection d'index.

2) Bilan sur la classification

a) Par rapport à l'optimisation simple, la sélection des structures d'optimisation avec classification OAC donne une meilleure optimisation des requêtes exécutées sur ED.

b) Par classification, les attributs les mieux adaptés pour IJB sont sélectionnés. Or, dans la sélection simple, la sélection d'un SF peut prendre des attributs mieux adaptés pour IJB et laisser d'autre dont l'IJB sera inutile.

c) Cette classification permet de respecter au mieux la contrainte sur FH, car le nombre d'attributs soumis à la fragmentation est réduit.

V. CONCLUSION

Nous avons effectué une expérimentation sur l'optimisation d'une charge de requêtes par sélection combinée d'IJB et de FH. Cette sélection combinée permet d'optimiser plus efficacement les requêtes exécutées sur ED. Ce sont des requêtes multi prédicats de sélection (PS) et avec de jointures en étoiles.

Les IJB et la FH sont définis sur les attributs de sélection figurant dans les prédicats de sélections des requêtes. Ils partagent donc une même ressource. Puisque la fragmentation est réalisée en premier lieu, il n'est pas sûr que les index définis sur les attributs qui restent soient bénéfiques. Ainsi, nous avons étudié le partage de cet ensemble d'attributs entre FH et IJB par une classification en deux classes : une pour FH, l'autre pour IJB. Cette classification est effectuée par l'algorithme K-Means qui se base sur une métrique définie sur chaque attribut à partir de plusieurs paramètres : la fréquence d'apparition de l'attribut dans la charge de requêtes, le facteur de sélectivité de ces valeurs ainsi que sa cardinalité. Une fois la classification effectuée, nous avons implémenté les structures d'optimisation sur chaque classe.

Enfin, pour montrer l'intérêt de notre approche de classification, nous avons effectué une série de tests. Nous avons comparé notre démarche à la sélection et implémentation simple des structures d'optimisation. Dans ces tests, nous avons fait varier la contrainte W du nombre de fragments maximum, exécuter la sélection sous différents modes : fragmentation seule, index seule et combinaison des deux avec plusieurs ordres : fragmentation d'abord et index d'abord et même fragmentation et indexation en parallèle. Sauf pour l'indexation seule, nous avons conclu que notre approche avec classification apporte de meilleurs résultats que l'approche simple.

REFERENCES

- [1] Bellatreche L., « Utilisation des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données », Thèse de Doctorat en Informatique, Université de Clermont Ferrand, 2000.
- [2] Teste O., « Modélisation et manipulation d'entrepôts de données complexes et historisées », Thèse de Doctorat, Université Paul Sabatier, 2000.
- [3] Bellatreche L., Boukhalfa K., Caffiau S., « ParAdmin: Un Outil d'Assistance à l'Administration et Tuning d'un Entrepôt de Données », Revue des Nouvelles Technologies de l'Information (EDA'2008), edited by Editionns Cépaduès, Juin 2008.
- [4] Golfarelli M., Rizzi S., Saltarelli E., « Index selection for data warehousing », in 4th International Workshop on Design and Management of Data Warehouses (DMDW 2002), Canada, 2002.
- [5] Kratica J., Ljubic I., Tosic D., « A Genetic Algorithm for the Index Selection Problem », in Applications of Evolutionary Computing, EvoWorkshops, 2003.
- [6] Feldman Y.A., Reouven J., « A knowledge-based approach for index selection in relational databases », Expert System with Applications, 25(1):15-37. 2003.
- [7] Chaudhuri, S., Datar, M., and Narasayya V. « Index Selection for Databases: A Hardness Study and a Principled Heuristic Solution ». IEEE Transactions on Knowledge and Data Engineering, VOL. 16, NO. 11, November 2004.
- [8] G. Valentin, M. Zuliani, D. Zilio, « DB2 advisor: An optimizer smart enough to recommend its own indexes ». In Proceedings of the International Conference on Data Engineering, 2000.
- [9] Aouiche K., « Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données », Thèse de Doctorat, Université Lumière Lyon 2, 2005
- [10] Bellatreche L., Boukhalfa K., Mohania M. K., « Pruning Search Space of Physical Database Design », in 18 International Conference on Database and Expert Systems Applications (DEXA'07), 2007.
- [11] Bellatreche L., Missaoui R., Necir H., Drias H., « A Data Mining Approach for Selecting Bitmap Join Indices », Journal of Computing Science and Engineering, vol. 2, n° 1, p. 206-223, January, 2008.
- [12] Navathe, S., Ra M., « Vertical partitioning for database design : graphical algorithm », ACM SIGMOD, pages 440-450, 1989
- [13] Hammer M., Niamir B. « A heuristic approach to attribute partitioning », Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 93-101, 1979.
- [14] 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 (BNCOD'06), p. 212-219, July, 2006.
- [15] Boukhalfa K., Bellatreche L., Richard P., « Fragmentation Primaire et Dérivée: Étude de Complexité, Algorithmes de Sélection et Validation sous ORACLE10g », Rapport de stage, ENSMA, 2008.
- [16] Mahboubi H., « Optimisation de la performance des entrepôts de données XML par fragmentation et répartition », Thèse pour obtenir le grade de Docteur en Informatique, 2009.
- [17] M. Raffestin, « La loi normale », Cours de probabilités – statistiques inférentielles, <http://ufr-pluribab.univ-pau.fr/live/digitalAssets>
- [18] Council O., « APB-1 OLAP Benchmark, Release II » <http://www.olapcouncil.org/research/bmarkly.htm>, 1998.