Une Sélection Multiple des Structures d'Optimisation Dirigée par la Méthode de Classification K-means

Rym Bouchakri *, Ladjel Bellatreche **, Kamel Boukhalfa***

* Ecole nationale Supérieure d'Informatique, Oued-Smar Alger - Algérie
r_bouchakri@esi.dz
** LISI/ENSMA - Université de Poitiers, Futuroscope 86960 France
bellatreche@ensma.fr
*** USTHB, Alger, Algérie
boukhalk@ensma.fr

Résumé. Le volume d'information contenu dans un entrepôt de données s'accroît sans cesse, augmentant de ce fait le temps d'exécution des requêtes décisionnelles. Pour y remédier, l'administrateur doit, durant la phase de conception physique de l'entrepôt, effectuer une sélection de structures d'optimisation (index, vues matérialisées ou fragmentation), puis assurer leur gestion et maintenance. Pour optimiser un nombre maximum de requêtes, il est indispensable d'opter pour une sélection multiple de structures ayant une forte similarité. Dans la littérature, deux principales similarités entre les structures d'optimisation ont été identifiées : une entre les vues et les index et l'autre entre la fragmentation horizontale dérivée et les index de jointure binaire. Dans ce travail, nous proposons une approche de sélection multiple des index de jointure binaire et de fragmentation. Vue la complexité de la sélection multiple, nous proposons une nouvelle approche permettant d'abord de partager l'ensemble des attributs extraits des requêtes entre les deux structures, ensuite sélectionner chaque structure avec un algorithme. Pour réaliser ce partage, nous proposons d'utiliser la méthode K-means. Une étude expérimentale et des tests comparatifs sur un entrepôt de données réel sous le SGBD Oracle 11g sont proposés illustrant l'intérêt de notre approche.

1 Introduction

Dans les bases de données traditionnelles, la tâche d'un administrateur était principalement concentrée sur la gestion des utilisateurs et d'un nombre restreint de structures d'optimisation comme les index mono table ou les différentes implémentations de l'opération de jointure (boucles imbriquées, tri fusion, hash, etc.). Dans les applications décisionnelles, la conception physique est devenue un enjeu important, comme l'indique Surajit Chaudhuri dans son papier intitulé *Self-Tuning Database Systems : A Decade of Progress*, à qui on a décerné le prix de 10 *Year Best Paper Award* à la conférence Very Large Databases (VLDB'2007). Cette amplification est due aux caractéristiques suivantes liées aux entrepôts de données : (1) le

volume de données, (2) la complexité des requêtes et les (3) exigences des décideurs sur le temps de réponse de requêtes. Pour offrir une meilleure utilisation d'un entrepôt de données, la conception physique est devenue un enjeu important (Chaudhuri et Narasayya, 2007). Durant cette conception, l'administrateur doit effectuer entre autres quatre tâches principales : (1) choix des structures d'optimisation, (2) choix de leur mode de sélection (isolée et multiple), (3) développement des algorithmes de sélection et (4) validation et déploiement des solutions d'optimisation. Une large panoplie des structures d'optimisation a été proposée et la plupart supportée par les SGBD commerciaux. Certaines structures sont l'héritage des bases de données traditionnelles ; comme la fragmentation, l'allocation, certains types d'index, le regroupement, etc. Notons que chaque structure a ses avantages, ses limites et certaines ont de fortes similarités. Afin de mieux les cerner, nous les avons classées en deux principales catégories (Bellatreche et al., 2007, 2008) : les structures redondantes et les structures non redondantes. Les structures redondantes optimisent les requêtes, mais exigent des coûts de stockage et de maintenance. Les vues matérialisées (Gupta, 1997), les index (Gupta et al., 1997), la fragmentation verticale[1] (Navathe et Ra, 1989; Sanjay et al., 2004) sont trois principaux exemples de cette catégorie. Les structures non redondantes ne nécessitent ni coût de stockage, ni coût de maintenance. Deux exemples de cette catégorie sont la fragmentation horizontale (Sanjay et al., 2004; Bellatreche et al., 2006) et le traitement parallèle (dans le cas où l'allocation est faite sans réplication) (Stöhr et al., 2000).

La sélection et la gestion des structures d'optimisation sont au coeur de la conception physique. La sélection d'une structure d'optimisation donne un schéma contenant une ou plusieurs instances de cette structure. Pour satisfaire sa charge de requêtes, l'administrateur peut choisir une ou plusieurs structures. Dans le cas, où il choisit une seule structure, sa sélection est appelée *sélection isolée*. Dans le cas, où il choisit plusieurs structures, leur sélection est appelée sélection multiple. La sélection isolée n'est cependant pas toujours suffisante pour optimiser toute la charge de requêtes, chaque structure d'optimisation étant bien adaptée pour un profil particulier de requêtes, d'où le recours à la *sélection multiple* (Stöhr et al., 2000; Bellatreche et al., 2007). Comme la sélection isolée, elle peut concerner des structures redondantes, non redondantes ou les deux. Elle est plus complexe que la sélection isolée compte tenu de la complexité de son espace de recherche englobant ceux des structures concernées. En plus de cette complexité, une autre s'y ajoute concernant la gestion des similarités (ou dépendances) entre certaines structures. Prenons l'exemple de deux structures d'optimisation redondantes qui sont les vues matérialisées et les index. Elles génèrent deux schémas d'optimisation redondants, partageant la même ressource qui est l'espace de stockage et nécessitent des mises à jour régulières. Une vue matérialisée relationnelle peut être indexée afin d'augmenter ses performances. En conséquence, la présence d'un index peut rendre une vue matérialisée plus avantageuse. Cette similarité pourrait influencer la sélection de leurs schémas (Sanjay et al., 2000; Zilio et al., 2004). Récemment, une autre similarité entre les index de jointure binaire (structure redondante) et la fragmentation horizontale dérivée ou référentielle (structure non redondante) a été identifiée. La fragmentation horizontale primaire des tables de dimension et la dérivée de la table des faits optimisent les sélections et les jointures, respectivement. D'un autre côté, les index de jointure binaire optimisent également les jointures et les sélections. Les deux structures sont *concurrentes sur la même ressource* représentant *l'ensemble des attributs*

[1] Dans la fragmentation verticale, la clé de la table fragmentée est dupliquée sur tous les fragments. Pour cela, elle est considérée comme une structure redondante

de sélection définis sur les tables de dimension (EAS). Deux caractéristiques les différencient : (a) la fragmentation horizontale peut être définie sur n'importe quel(s) attribut(s) de EAS, tandis qu'un IJB est défini sur un ou plusieurs attributs de EAS, mais de faible cardinalité. (b) Les IJB sont contraints par l'espace de stockage (ou coût de maintenance) tandis que la fragmentation est contrainte par le nombre de fragments.

Dans (Boukhalfa et al., 2008), les auteurs ont proposé une résolution séquentielle au problème de la sélection multiple de schémas de fragmentation et d'index de jointure binaire en prenant en compte les similarités. Cette sélection commence par la fragmentation en utilisant tous les attributs de sélection définis sur toutes les requêtes de départ, ensuite l'indexation en prenant en compte les attributs non utilisés par la fragmentation et les requêtes non bénéficiaires de la fragmentation. Une requête est dite bénéficiaire si elle est optimisée par la fragmentation, dans le cas contraire, elle est dite non bénéficiaire. Ces travaux utilisent la fragmentation horizontale pour choisir les attributs candidats à l'indexation et en conséquence réduire l'espace de recherche du problème de sélection des index. Ce choix est souvent guidé par un modèle de coût qui ne prend pas en considération la présence des index de jointure binaire lors de l'évaluation des requêtes. En conséquence, le processus de fragmentation risque d'utiliser certains attributs de sélection même s'ils sont plus performants pour l'indexation et vice versa. Ce constat nous a motivé pour considérer un nouveau problème lié à la sélection multiple des deux structures d'optimisation qui est *le partage des attributs de sélection entre elles afin de maximiser la performance de l'ensemble de requêtes*.

Dans cet article, nous proposons une solution pour partager les ensembles des attributs de sélection entre la fragmentation horizontale et les index de jointure en utilisant une méthode de classification K-means. Une fois réalisée, nous sélectionnons le schéma de chaque structure avec ses propres attributs.

Ce travail rentre dans le cadre de la préparation du magistère au sein de l'Ecole nationale Supérieure d'Informatique (ESI), Alger. Il s'organise en quatre sections. La section 2 présente les concepts de base liés à la fragmentation et l'indexation et montre l'intérêt du problème de partage des attributs de sélection entre les deux structures d'optimisation. La section 3 décrit notre approche de sélection multiple par la méthode K-means. La section 4 présente l'étude expérimentale, sous le SGBD Oracle, qui montre l'apport de la classification au coût d'exécution des requêtes sur l'entrepôt de données de banc d'essai APB1. La section 5 conclut le papier en récapitulant les résultats principaux et en suggérant des travaux futurs.

2 Concepts de Base & Exemple de Motivation

Dans cette section, nous rappelons les concepts de base liés à la fragmentation horizontale et aux IJBs. Ensuite, nous donnons un exemple concret motivant la nécessité du problème de partage des attributs de sélection entre la fragmentation et les index de jointure binaire.

2.1 Fragmentation Horizontale Primaire et Dérivée

Initialement proposée comme une technique de conception logique des bases de données réparties dans les années 80, la fragmentation horizontale consiste à partitionner une table en fonction de ses n-uplets de façon à réduire le nombre d'accès non nécessaire pour le traitement de certaines requêtes. Deux types de fragmentation horizontale existent : *primaire* et *dérivée*

(Özsu et Valduriez, 1999). La fragmentation horizontale primaire d'une table se base sur des attributs définis sur cette table. La fragmentation horizontale dérivée consiste à propager la fragmentation d'une table sur une autre table. Cette propagation n'est possible que si un lien *père-fils* existe entre deux tables. Par conséquent, la fragmentation horizontale dérivée d'une table se base sur *des attributs de sélection* définis sur une ou plusieurs autres tables de dimension (Eadon et al., 2008). Les fragmentations primaire et dérivée[2] sont supportées par la plupart des SGBD commerciaux (Oracle, SQL Server, DB2, etc.) et non commerciaux (Postgress, MySQL, etc.). Notons que le problème de sélection d'un schéma de fragmentation d'un entrepôt de données est NP-difficile (Bellatreche et al., 2009).

2.2 Index de Jointure Binaire

Un IJB pré-calcule la jointure entre la table des faits et une ou plusieurs tables de dimension en utilisant un ou plusieurs attributs de sélection. Cette jointure est matérialisée à travers un ensemble de vecteurs de bits construits sur la table des faits en se basant sur un ou plusieurs attributs de dimension de faible cardinalité. Un vecteur de bits représentant les n-uplets de la table de faits est créé pour chaque valeur distincte de l'attribut de la table de dimension sur lequel l'index est construit. Le ième bit du bitmap est égal à 1, si le n-uplet correspondant à la valeur de l'attribut indexé peut être joint avec le n-uplet de rang i de la table de faits. Dans le cas contraire, le ième bit est à zéro. Les IJB sont efficaces pour les requêtes de type COUNT, AND, OR, NOT, d'où leur implémentation dans les SGBD commerciaux. Les IJB sont définis sur des attributs de faible cardinalité. La taille des index binaires est proportionnelle à la cardinalité des attributs indexés. Notons que le problème de sélection d'un schéma de d'indexation est NP-difficile (Chaudhuri, 2004).

2.3 Partage des Attributs de Sélection entre la Fragmentation et les IJB

Lors de la conception physique, si l'administrateur opte pour la fragmentation horizontale et les index de jointure binaire pour satisfaire sa charge de requêtes ; deux approches sont possibles pour identifier les attributs de fragmentation[3] et les attributs indexables[4] : (i) *manuelle* et (ii) *automatique*.

Dans l'approche manuelle, l'administrateur exploite son expérience pour partager l'ensemble des attributs de sélection EAS entre la fragmentation et les IJB. Dans le contexte d'entrepôts de données, le nombre d'attributs de fragmentation et indexables peut être important (Simon, 2008). En conséquence, l'approche manuelle ne peut pas être appliquée dans les projets décisionnels réels impliquant un nombre important de tables de dimension et le nombre d'attributs non clés par table de dimension. Une autre limite est l'absence d'une métrique quantifiant la qualité de la solution finale de ce partage.

Dans l'approche automatique, à notre connaissance seulement deux travaux existent (Stöhr et al., 2000) et (Boukhalfa et al., 2008). Dans (Stöhr et al., 2000), les auteurs proposent de fragmenter et indexer l'entrepôt de données indépendamment. La fragmentation horizontale

[2] Également connue sous le nom de referential partitioning dans Oracle11G

[3] Un attribut de fragmentation est tout attribut d'une table de dimension référencé par un prédicat de sélection et utilisé pour partitionner l'entrepôt.

[4] Un attribut d'une table de dimension est dit indexable s'il est de faible cardinalité, référencé par un prédicat de sélection et utilisé pour indexer l'entrepôt.

de chaque table de dimension repose sur une fragmentation à un point (*one point fragmentation*). Chaque fragment de la table de dimension est associé à une seule valeur d'un attribut de fragmentation. Par exemple, si la fragmentation se fait sur l'attribut *Ville* et que ce dernier possède 100 valeurs différentes, alors 100 fragments de la table *Client* seront considérés pour partitionner la table des faits. Les auteurs imposent aussi que chaque attribut de fragmentation doit appartenir à un niveau de hiérarchie de sa table de dimension. Quant aux index de jointure binaire, ils sont sélectionnés avec un codage particulier sur tous les attributs d'hiérarchies importantes. En analysant chaque requête, une comparaison est effectuée afin d'identifier l'effet de la fragmentation et l'indexation sur cette requête. Si un index n'est pas utilisé pour répondre efficacement à la requête, il est automatiquement éliminé. Dans (Boukhalfa et al., 2008), la fragmentation est utilisée sur la totalité de l'ensemble EAS. Étant donnée que la fragmentation contrôle le nombre de fragments de la table des faits, certains attributs de AES ne peuvent être utilisés, en conséquence, ils seront candidats à l'indexation. Le fait de commencer par la fragmentation peut présenter un risque pour certains attributs de AES d'être utilisés même s'ils sont bien placés pour l'indexation.

En se basant sur cette analyse, il est recommandé de proposer aux administrateurs durant la conception physique une méthode automatique affectant des attributs pertinents pour la fragmentation et l'indexation avant le lancement de leurs algorithmes de sélection.

3 Problème de Partage des Attributs de Sélection entre la Fragmentation et les IJB

Rappelons que la sélection des schémas de fragmentation et d'IJB s'effectue à l'aide des attributs de sélection extraits de la charge de requêtes. Souvent la fragmentation horizontale et les IJB sont concurrentes sur un sous ensemble important de AES de faible cardinalité (Stöhr et al., 2000). Partager cet ensemble entre la fragmentation et les IJB doit satisfaire le problème global de la sélection multiple des deux structures qui est formalisé comme suit :

- Une charge de requêtes $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_m\}$, où chaque requête Q_j possède une fréquence d'accès f_j ;
- Un ensemble d'attributs de sélection AS extraits de l'ensemble $\mathcal{Q}$;
- Une capacité de stockage réservée aux IJB ;
- Un seuil W représentant le nombre de fragments de la table des faits.

Le problème de la sélection multiple (PSM) consiste à sélectionner des schémas de fragmentation et d'indexation afin de réduire le coût d'exécution de la charge de requêtes $\mathcal{Q}$ en présence de ces derniers et satisfaire les contraintes définies. La résolution de ce problème nécessite une exploitation de l'espace de recherche englobant les deux sous espaces correspondants à la fragmentation et à l'indexation. Soient Ins_F et Ins_I le nombre d'instances du problème de la fragmentation et d'indexation respectivement. L'espace de recherche global est $2^{Ins_F + Ins_I}$, du fait que les différentes instances peuvent interagir (Zilio et al., 2004).

Vue cette complexité, le PSM doit se résoudre d'une manière séquentielle en prenant en compte les similarités entre les deux structures. En conséquence, une démarche de résolution du PSM consiste d'abord à partager les attributs entre les deux structures, ensuite à lancer les algorithmes de sélection avec leurs attributs respectifs. Un partage a priori des attributs de

sélection entre les structures réduit la complexité des deux problèmes (problème de la sélection de schéma de fragmentation et problème de sélection de schémas d'indexation).

Le nombre de classifications possibles (CP) de l'ensemble EAS en deux classes correspondantes à la fragmentation et l'indexation est donné par l'équation suivante :

$$CP = 2^{||EAS||} \tag{1}$$

où $||EAS||$ représente le cardinal de l'ensemble EAS. Cependant, énumérer tous les cas possibles de classification et pour chaque cas exécuter deux algorithmes de sélection reste impossible vue la complexité identifiée par l'équation 1. En conséquence, nous développons une approche de partage moins coûteuse prenant en considération les caractéristiques de chaque problème et chaque structure d'optimisation.

3.1 Critères de Partage

Dans cette section, nous analysons les attributs de sélection sur chaque structure d'optimisation afin de proposer une méthode de partage. Cette analyse est basée sur deux études, une théorique de chaque structure d'optimisation (Aouiche et al., 2005; Boukhalfa, 2009; Mahboubi et Darmont, 2008) et une autre menée sur un entrepôt de données issu de banc d'essai (Council, 1998) et implanté sous ORACLE11g. Ces études nous ont permis d'identifier trois critères importants :

1. *Fréquence d'un attribut de sélection* : ce critère représente le nombre de fois qu'un attribut de sélection est utilisé par les requêtes. Si la fragmentation est utilisée sur un ensemble d'attributs fréquents elle donne de meilleures performances que les index. Dans certains cas, les index sont meilleurs, mais on privilégie la fragmentation vue sa caractéristique de non redondance.

2. *Cardinalité des attributs :* plus la cardinalité d'un attribut est grande, plus la taille de chaque fragment est réduite, car la fragmentation est basée sur la décomposition de domaine de chaque attribut en plusieurs sous domaines. En conséquence, une requête donnée ne peut charger qu'une petite partie de la table des faits. Mais ce cas n'est pas toujours vrai car la fragmentation est contrainte par un seuil représentant le nombre de fragments de la table des faits qu'un administrateur souhaite avoir. Dans ce cas, certains domaines doivent être fusionnés ce qui a été démontré dans (Bellatreche et al., 2009). D'un autre coté, les index définis sur des attributs à forte cardinalité sont volumineux et peuvent violer la contrainte de l'espace de stockage maximum. En se basant sur cette étude, nous avons adopté le scénario qui suit partiellement la logique des travaux de (Stöhr et al., 2000) (sur la partie indexation). Les attributs de forte cardinalité sont adaptés à la fragmentation tandis que les attributs de faible cardinalité sont candidats à l'indexation :

3. *Facteur de sélectivité des prédicats définis sur les attributs de sélection* : soit un attribut A utilisé par $k(k \geq 0)$ prédicats de sélection $\{P_1, ..., P_k\}$. Les expérimentations ont montré qu'un IJB sur A est bénéfique pour les requêtes si les facteurs de sélectivité de ses prédicats sont faibles.

Nous définissons un facteur de sélectivité d'un attribut A ($FS(A)$) par la moyenne des facteurs de sélectivité de ses prédicats.

$$FS(A) = \frac{\sum_{i=1}^{k} Sel(P_i)}{k} \quad (2)$$

3.2 Notre Démarche de la Sélection Multiple

Comme nous l'avons indiqué, notre démarche pour résoudre le problème de la sélection multiple consiste d'abord à partager l'ensemble des attributs entre les deux structures d'optimisation ensuite à lancer les algorithmes de sélection. Les étapes de notre démarche sont :

1. Extraire, à partir de la charge de requêtes, les attributs de sélection.
2. Classifier ces attributs entre la fragmentation horizontale et les IJB
3. Définir un schéma de fragmentation sur sa classe d'attributs.
4. Définir les IJB sur leur classe d'attributs et les attributs non sélectionnés par le processus de fragmentation.

La Figure 1 illustre ces différentes étapes.

3.3 Démarche de classification par "k-means"

Notre problème de partage des attributs de sélection entre la fragmentation et l'IJB a une forte similitude avec le problème de classification d'un ensemble en k partitions largement étudié dans la littérature par la communauté de fouille de données. Il existe deux types de classification : supervisée qui est basée sur l'apprentissage à partir d'exemples et non supervisée ou data clustering qui ne fait aucune hypothèse sur les données à classifier (Robardet, 2005). Cette dernière peut être, à son tour, hiérarchique ou classification par partitionnement qui vise à classifier un ensemble de données, représentées dans l'espace $\Re^n$, en plusieurs classes qui regroupent chacune les données les plus proches.

La classification des attributs de sélection que nous visons à établir est une classification non supervisée par partitionnement ou les partitions ne se chevauchent pas entres elles. Parmi les algorithmes employés pour réaliser cette classification, on peut citer le "k-means" (MacQueen, 1967). Généralement, il permet de partitionner m données en k classes $\{C_1, ..., C_k\}$ en positionnant dans l'espace des données, $\Re^n$, k centroïdes $\{ct_1, ..., ct_k\}$ généralement choisis de manière aléatoire parmi les données elles même. Chaque donnée est alors affectée à une classe dont le centroïde est le plus proche de cette donnée en utilisant la *distance euclidienne*. Afin de vérifier la stabilité des classes, k-means effectue la réaffectation des données à chaque classe sur plusieurs itérations. Notre choix s'est porté sur "k-means" car cet algorithme est simple à utiliser et il s'adapte parfaitement à notre problème. De plus, il a été utilisé pour fragmenter horizontalement des entrepôts de données XML (Mahboubi et Darmont, 2008).

Afin d'adapter l'algorithme k-means à notre besoin de classification des attributs de sélection, nous considérons les adaptations suivantes :

- Les données à classifier sont les attributs de sélection EAS
- Les attributs sont représentés dans l'espace $\Re^2$ par des coordonnées (x, y). Ces coordonnées sont calculées à partir du poids de classification qui suit :
  nous définissons un poids de classification qui permet de décider, pour chaque attribut,

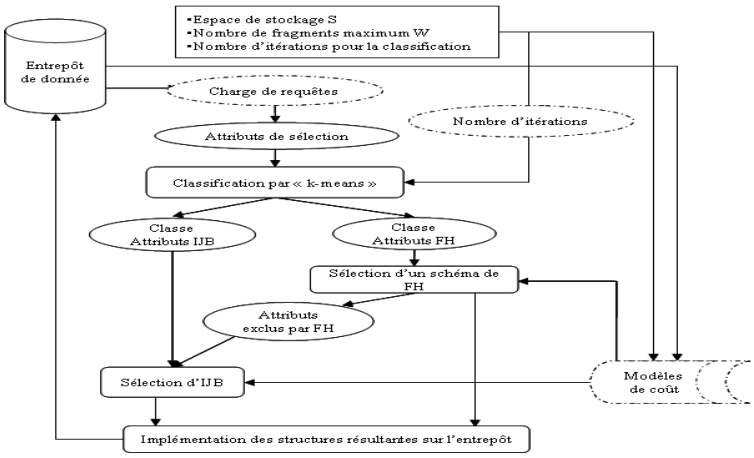

FIG. 1 – *Les étapes de notre démarche de sélection*

s'il est candidat à la fragmentation ou à l'indexation. Le calcul d'un point d'un attribut A_i est basé sur trois facteurs cités ci dessous.

$$Poids(A_i) = Freq(A_i) + Sel(A_i) + Card(A_i) \quad (3)$$

Lors de l'élaboration des poids des attributs, nous avons remarqué que les critères utilisés (fréquence, cardinalité et facteur de sélectivité) ont une échelle différente. Pour que le poids soit cohérent, nous avons effectué une normalisation des données issues de chaque critère. La normalisation vise à centrer et réduire un échantillon de données suivant sa moyenne et son écart type (Raffestin, 2002). Une fois le poids calculé, les coordonnées dans $\Re^2$ de chaque attribut de sélection A_i sont spécifiées comme suit : $(x, y) = (numrodel'attributi, poids(A_i)$.

Exemple 1 *Soit l'ensemble des attributs de sélection suivant :*
$EAS = \{Mois, Anne, Ville, Pays, Classe\}$. *Le domaine de chaque attribut est :*
$domaine(Genre) = \{$*'F', 'M'*$\}$ *;*
$domaine(Ville) = \{$*'Alger', 'Annaba', 'Oran', 'Msila'*$\}$ *;*
$domaine(Classe) = \{$*'A', 'B', 'C'*$\}$ *et*
$domaine(Mois) = \{$*'Janvier', 'Février', 'Mars'*$\}$.
Table 1 donne le calcul du poids pour chaque attribut. Ces coordonnées sont résumées dans la Table 2.

– Les attributs sont classifiés en deux ensembles : Classe_IJB et Classe_FH. Ainsi, $k = 2$

Figure 2 montre une répartition des attributs en deux ensembles délimités par les cercles rouges. Les attributs 'Année', 'Mois' et 'Ville' constituent la Classe_FH. Les attributs 'Pays' et 'Classe' constituent la Classe_IJB. Cette classification est ainsi, car nous favorisons, pour la fragmentation, les attributs dont la fréquence d'utilisation, le facteur de sélectivité et la cardinalité sont grands.

TAB. 1 – *Calcul du poids des attributs de sélection*

Attribut	Nbr	FS	Card	Nbr Normalisé	FS Normalisé	Card Normalisé	Poids
Année	11	0.5	23	1.14	0.53	0.01	1.70
Mois	5	0.33	12	0.26	1.41	-0.3	1.37
Ville	6	0.1	55	0.41	-0.13	0.94	1.22
Pays	9	0.09	20	0.85	-0.2	-0.07	0.57
Classe	3	0.02	62	0.02	-0.67	1.14	0.44

TAB. 2 – *Coordonnées des attributs de dimension*

Attribut	Année	Mois	Ville	Pays	Classe
Coordonnées	[1, 1.70]	[2, 1.37]	[3, 1.22]	[4, 0.57]	[5, 0.44]

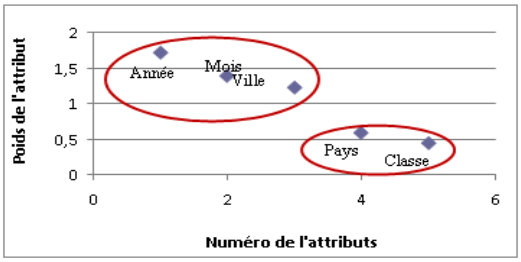

FIG. 2 – *Résultat de la classification*

4 Etude de Performance

Nous avons effectué une étude expérimentale réalisée sur une charge de 47 requêtes, exécutées sur un entrepôt de données issu du banc d'essai APB1 (Council, 1998) sous Oracle 11g, avec une machine Intel Core2Duo et une mémoire 2GO. Le schéma en étoile de notre entrepôt de données est constitué d'une table de faits *Actvars* (24 786 000 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 intégrant deux API JAVA, l'une permet d'implémenter l'algorithme de classification "k-means", la seconde, nommée *JGAP* (Java Genetic Algorithms Package), est un Framework qui implémente les deux algorithmes génétiques que nous avons utilisés pour sélectionner les schémas de fragmentation et d'indexation. La sélection de ces derniers est d'abord réalisée en utilisant deux modèles de coût, un pour la fragmentation et l'autre pour l'indexation (Boukhalfa, 2009), ainsi les résultats obtenus sont ensuite appliqués sur l'entrepôt de données sous Oracle11G. Les 47 requêtes sont alors exécutées sur l'entrepôt avec et sans structures d'optimisation.

4.1 Démarche d'Optimisation avec Classification (OAC)

Les étapes de l'optimisation avec classification sont :

1. *Classification des attributs* : pour un nombre d'itération = 50, *k-means* classifie les attributs dimensions en deux classes :
   $Classe\_FH = \{Gender, Month, Year, All, Quarter, Group\}$ et
   $Classe\_IJB = \{Family, Division, Class, City, Retailer\}$
2. *Sélection des structures d'optimisation* : pour la fragmentation, nous utilisons l'algorithme génétique développé dans (Boukhalfa, 2009), avec un seuil $W = 70$ représentant le nombre de fragments maximaux de la table des faits. Ce dernier retourne un schéma de fragmentation composé de 64 fragments de faits. En ce qui concerne l'indexation, nous avons développé un nouveau algorithme génétique qui nous donne 4 IJB définis sur les attributs suivants : *Family, Division, City* et *Retailer*. Cette sélection est contrainte par l'espace de stockage de 500Mo.
3. Les deux schémas sont directement implémentés en utilisant des scripts sur l'entrepôt de données réel.

4.2 Démarche d'Optimisation avec Requêtes non Bénéficiaires (ORB)

La première expérimentation présente une comparaison entre notre proposition et celle proposée dans (Boukhalfa et al., 2008). Le principe de base de cette sélection multiple est :

1. Sélectionner un schéma de fragmentation sur l'ensemble des attributs de sélection.
2. Déterminer les requêtes non bénéficiaires de la fragmentation horizontale.
3. Extraire, à partir des requêtes non bénéficiaires de la fragmentation, les attributs candidats à l'indexation.
4. Sélectionner un schéma d'indexation sur ces attributs.

4.3 Démarche d'Optimisation sans Classification

L'optimisation sans classification (dite *optimisation simple* (OS)) consiste à effectuer les deux dernières étapes de la section 4.1 (sans la classification).

4.4 Tests et Résultats

Nous avons effectué plusieurs tests, et pour chaque test, nous avons comparé entre OAC, ORB et OS. Les coûts des requêtes, après chaque optimisation, sont calculés par l'optimiseur d'Oracle11g. Nous avons réalisé l'optimisation de l'entrepôt de données selon quatre différents modes : Sans structure d'optimisation, IJB Seul, FH Seule et $FH\&IJB$ avec $S = 500Mo$ et $W = 70$. Pour chaque mode, nous effectuons l'optimisation par les trois démarches OAC, ORB et OS afin de comparer les résultats. Nous présentons, dans la Figure 3 les coûts d'exécution de la charge de requêtes pour chaque démarche sous chaque mode. La figure 4 représente le taux des requêtes optimisées. Nous remarquons que le meilleur coût d'exécution est obtenu pour le mode mixte $FH\&IJB$ et pour la démarche avec classification. En effet, le coût passe de 28.4 millions à 12.5 millions d'entrées sorties (E/S), soit une réduction de 56% du coût total et 91% des requêtes ont été optimisées). Cela montre plusieurs résultats essentiels :

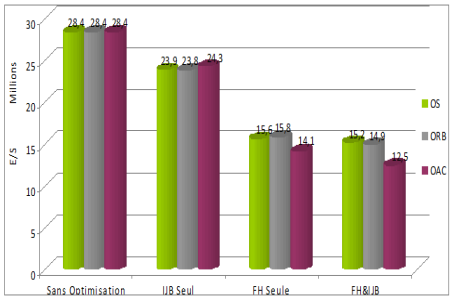

FIG. 3 – *Coût d'exécution (en nombre d'ES) de différents modes sous Oracle*

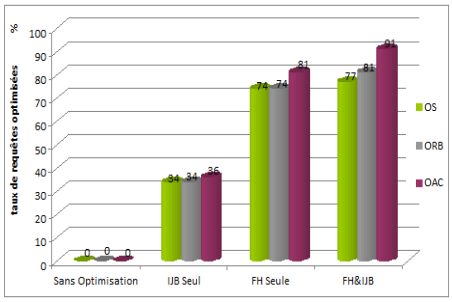

FIG. 4 – *Taux de requêtes optimisées*

- L'optimisation par sélection multiple apporte les meilleurs résultats.
- La fragmentation sur la classe d'attributs choisis par k-means donne un meilleur coût que la fragmentation sur tout l'ensemble d'attributs. En effet, pour l'approche OAC, les attributs les plus appropriés sont choisi pour la fragmentation.
- La sélection ORB apporte une amélioration par rapport à OS uniquement dans le mode $FH\&IJB$. En effet la démarche sélectionne les index uniquement sur les requêtes non bénéficiaires, ce qui réduit le nombre d'index candidats. Par contre, la classification apporte un bénéfice quel que soit le mode d'exécution.
- Notre démarche OAC permet une meilleure optimisation, elle permet de choisir pour la fragmentation et les IJB les attributs les plus adéquats.
- Notons que pour l'indexation seule, la méthode simple présente un meilleur résultat. En effet, la méthode simple définis les IJB sur tous les attributs de sélection, par contre notre démarche OAC défini l'indexation uniquement sur la classe IJB.

Afin d'étudier l'influence de la contrainte W (nombre de fragments maximum) sur l'optimisation de la charge de requêtes, nous avons fait varier W avec $S = 500Mo$. Pour chaque valeur de W, nous effectuons une optimisation par OS, ORB et OAC. Nous relevons le coût d'exécution et le taux de requêtes optimisées (Figures 5, 6). Les résultats qu'illustre la figure 5 montrent que la meilleure optimisation est obtenue avec notre démarche OAC surtout pour $W > 100$. En effet, le coût est réduit de 58% à 61% avec 91% des requêtes optimisées. Nous avons voulu étudier l'influence de la contrainte d'espace de stockage des index S sur l'optimisation de la charge de requêtes. Nous avons fait varier S avec $W = 20$ (cette valeur de W laisse pour l'indexation un grand nombre d'attributs). Pour chaque valeur, nous effectuons une optimisation par les trois démarches. Nous avons relevé le coût d'exécution de la charge de requêtes (Figure 7). Nous remarquons que, pour $S < 900Mo$, l'approche ORB donne de meilleurs résultats que la démarche OS. En effet, pour ces valeurs de S, choisir les attributs candidats aux index à partir d'un sous ensemble de requêtes (requêtes non bénéficiaires) réduit la complexité du problème de sélection des index. Mais plus l'espace de stockage augmente plus les deux démarches deviennent linéaires. Par contre, quelle que soit la valeur de S, notre démarche OAC apporte les meilleurs résultats car les attributs choisis pour indexation sont ceux qui donnent les IJBs les plus bénéfiques. En effet, le coût de la charge de requêtes, pour OAC, est réduit de 19.1 millions E/S pour $S = 50Mo$ jusqu'à 15.8 Millions E/S pour

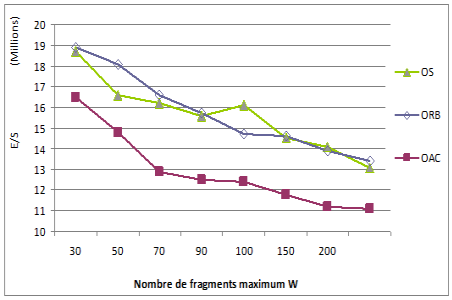

FIG. 5 – *Effet du Seuil sur le coût d'exécution de requêtes*

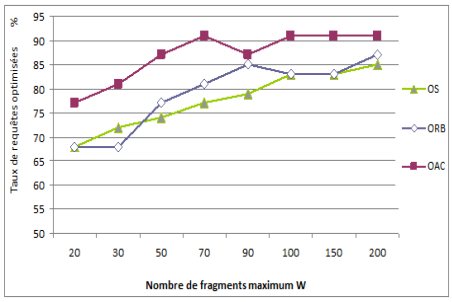

FIG. 6 – *Seuil et taux de requêtes optimisées*

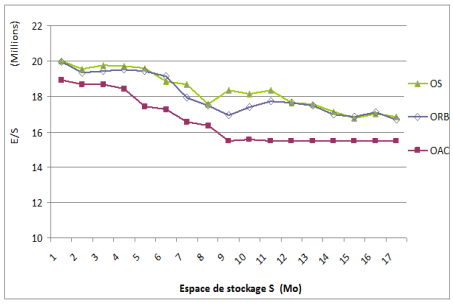

FIG. 7 – *Comparaison entre OAC, ORB et OS par variation de l'espace de stockage*

$S > 600Mo$, contrairement à OS et ORB où le coût varie entre 20.1 et 16.9 Millions E/S.

Les tests effectués jusque là se basaient sur la sélection d'un schéma de FH suivi de la sélection des IJB. Nous avons ensuite testé le changement de l'ordre de sélections (chaque technique dispose de son propre module de sélection). La figure 8 montre trois possibilités de sélections : FH puis IJB, IJB puis FH et $FH//IJB$ avec $S = 500Mo$ et $W = 70$. Concernant la démarche de sélection $FH//IJB$, elle signifie que chaque sélection est effectuée sur sa classe correspondante, sans prendre en compte les attributs non choisit par l'autre sélection. La figure 8 montre que les meilleurs résultats sont obtenus par notre approche OAC pour l'ordre FH puis IJB (12.5 M E/S), suivi de la démarche OAC avec l'ordre $FH//IJB$ (13.7 m E/S) puis ORB avec l'ordre FH puis IJB (15.1 m E/S). A partir de ces résultats, nous pouvons faire les remarques suivantes :

– L'ordre IJB puis FH donne l'optimisation la moins intéressante. En effet, les attributs non sélectionnés par IJB vont être ajoutés à la classe de la fragmentation horizontale et vont fausser le choix du schéma d'optimisation. Contrairement à la sélection d'IJB, où l'attribut est soit choisi soit écarté, la sélection d'un schéma de FH peut choisir un attribut avec un nombre de fragments réduit.
– Les résultats de OS et ORB dans le mode FH puis IJB (15.6 et 15.1 Millions E/S) sont meilleurs que OAC dans le mode IJB puis FH (17,2 Millions E/S), cela est dû

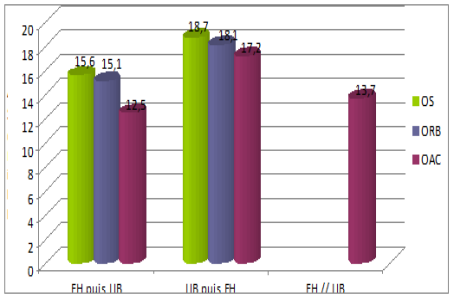

FIG. 8 – *Différents ordres d'exécution*

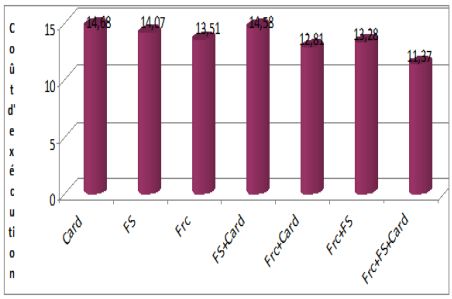

FIG. 9 – *Optimisation selon la formulation le poids de classification*

principalement à la sélection de schéma de fragmentation. En effet, la fragmentation dans OS ou ORB (FH puis IJB) s'effectue sur tous les attributs de sélection. Par contre, la fragmentation dans OAC (IJB puis FH) s'effectue sur un ensemble d'attributs qui contient des attributs non adaptés à la fragmentation horizontale (attributs choisi par k-means pour indexation et écartés par la sélection d'IJB)

– L'ordre $FH//IJB$ est moins bénéfique que fragmentation puis les IJB car la séparation des deux sélections empêche la prise en compte par IJB des attributs non choisis par fragmentation horizontale.

Le test, illustré dans la figure 8, nous conduit à déduire l'intérêt de la classification des attributs. En effet, et malgré le fait que la FH est défini sur un nombre réduit d'attributs, ces attributs constituent les meilleurs candidats pour la fragmentation. D'autre part, ceux choisit pour les index sont plus adaptés aussi. Ce qui donne une meilleure optimisation globale.

Le poids de classification comporte trois facteurs : Frc, 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. Les résultats sont présentés dans Figure 9. 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 tous les facteurs. Cela montre que les trois facteurs sont pertinents pour la classification.

5 Conclusion

Afin de satisfaire les requêtes complexes définies sur les entrepôts de données, la sélection multiple est devenue une solution incontournable. Elle est plus complexe que la sélection isolée compte tenu de la complexité de son espace de recherche englobant ceux des structures concernées. En plus de cette complexité, une autre s'y ajoute concernant la gestion des similarités (ou dépendances) entre certaines structures comme les *index de jointure binaire* et la *fragmentation horizontale*. Dans ce papier, nous nous sommes intéressés à la sélection multiple de schémas de fragmentation (considérée comme une *structure non redondante*) et d'indexation (considérée comme *une structure redondante*). Nous avons identifié que les processus de fragmentation et d'indexation sont concurrents à la même ressource représentant l'ensemble d'attributs de sélection. Pour répondre à cette sélection, nous avons proposé de classer les attributs de sélec-

tion, extraits, à partir des requêtes entre les deux structures d'optimisation avant le lancement de leurs algorithmes de sélection. Cette classification est faite par la méthode 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 ses prédicats définis ainsi que sa cardinalité. Une fois la classification effectuée, chaque structure d'optimisation est sélectionnée par un algorithme génétique dirigé par un modèle de coût. 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. Les premiers résultats ont montré l'intérêt de notre démarche de classification pour résoudre la sélection multiple.

Pour les travaux futurs, il serait intéressant d'inclure d'autres critères dans la méthode K-means, comme la nature de chaque requête (recherche, mise à jour), le nombre de jointures par requête, etc. Une deuxième piste consisterait à étudier cette démarche dans le contexte dynamique.

Références

Aouiche, K., O. Boussaid, et F. Bentayeb (2005). Automatic Selection of Bitmap Join Indexes in Data Warehouses. pp. 64–73.

Bellatreche, L., K. Boukhalfa, et H. I. Abdalla (2006). Saga : A combination of genetic and simulated annealing algorithms for physical data warehouse design. In *23rd British National Conference on Databases (BNCOD'06)*, pp. 212–219.

Bellatreche, L., K. Boukhalfa, et M. K. Mohania (2007). Pruning search space of physical database design. In *18th International Conference On Database and Expert Systems Applications (DEXA'07)*, pp. 479–488.

Bellatreche, L., K. Boukhalfa, et P. Richard (2009). Referential horizontal partitioning selection problem in data warehouses : Hardness study and selection algorithms. *International Journal of Data Warehousing and Mining 5*(4), 1–23.

Bellatreche, L., R. Missaoui, H. Necir, et H. Drias (2008). A data mining approach for selecting bitmap join indices. *Journal of Computing Science and Engineering 2*(1), 206–223.

Boukhalfa, K. (2009). De la conception physique aux outils d'administration et de tuning des entrepôts de données. Phd. thesis, Poitiers University, France.

Boukhalfa, K., L. Bellatreche, et Z. Alimazighi (2008). Hp&bji : A combined selection of data partitioning and join indexes for improving olap performance. *Annals of Information Systems, Special Issue on new trends in data warehousing and data analysis, Springer 3*, 179–2001.

Chaudhuri, S. (2004). Index selection for databases : A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering 16*(11), 1313–1323.

Chaudhuri, S. et V. Narasayya (2007). Self-tuning database systems : A decade of progress. In *Proceedings of the International Conference on Very Large Databases*, pp. 3–14.

Council, O. (1998). Apb-1 olap benchmark, release ii. *http ://www.olapcouncil.org/research/bmarkly.htm*.

Eadon, G., E. I. Chong, S. Shankar, A. Raghavan, J. Srinivasan, et S. Das (2008). Supporting table partitioning by reference in oracle. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1111–1122.

Gupta, H. (1997). Selection of views to materialize in a data warehouse. In *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*, pp. 98–112.

Gupta, H., V. Harinarayan, A. Rajaraman, et J. Ullman (1997). Index selection for olap. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pp. 208–219.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press*, pp. 281–297.

Mahboubi, H. et J. Darmont (2008). Data mining-based fragmentation of xml data warehouses. In *ACM 11th International Workshop on Data Warehousing and OLAP (DOLAP'08)*, pp. 9–16.

Navathe, S. et M. Ra (1989). Vertical partitioning for database design : a graphical algorithm. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 440–450.

Özsu, M. T. et P. Valduriez (1999). *Principles of Distributed Database Systems : Second Edition*. Prentice Hall.

Raffestin, M. (2002). La loi normale (ou loi de laplace-gauss ou loi de gauss). cours de probabilité - statistiques inférentielles. Techreport, Université de Pau France.

Robardet, C. (2005). Classification supervisée. Techreport, INSA-Lyon : http ://liris.cnrs.fr/celine.robardet/doc/classification.pdf.

Sanjay, A., V. R. Narasayya, et B. Yang (2004). Integrating vertical and horizontal partitioning into automated physical database design. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 359–370.

Sanjay, A., C. Surajit, et V. R. Narasayya (2000). Automated selection of materialized views and indexes in microsoft sql server. In *Proceedings of the International Conference on Very Large Databases*, pp. 496–505.

Simon, E. (2008). Reality check : a case study of an eii research prototype encountering customer needs. In *11th International Conference on Extending Database Technology (EDBT'08)*, pp. 1.

Stöhr, T., H. Märtens, et E. Rahm (2000). Multi-dimensional database allocation for parallel data warehouses. In *Proceedings of the International Conference on Very Large Databases*, pp. 273–284.

Zilio, D. C., J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, et S. Fadden (2004). Db2 design advisor : Integrated automatic physical database design. In *Proceedings of the International Conference on Very Large Databases*, pp. 1087–1097.

Summary

The volume of data contained in a data warehouse grows dramatically, thereby increasing the complexity of decision support queries. To remedy this, the administrator shall, during

Sélection multiple des structures d'optimisation dirigée par K-means

the physical design phase of the data warehouse, perform a selection of structure optimization (indexes, materialized views or fragmentation), and ensure their maintenance. In order to maximize a large number of queries, it is mandatory to select several similar optimization structures. This similarity facilitates the manageability of each structure. In the literature, two main similarities between the optimization structures have been identified: one between materialized views and indexes and the other between the derived horizontal fragmentation and bitmap join indexes. In this work, we deal with the problem of selecting both horizontal partitioning and bitmap join indexes schemes. Due to the complexity of this selection, we propose a new approach that first splits the set of selection attributes among these two structures, and then selects each structure with an algorithm. This split is done using K-means method. An intensive experimental study is conducted on a real data warehouse under the Oracle 11g DBMS.