Sélection Incrémentale d'un Schéma de Fragmentation Horizontale d'un Entrepôt de Données Relationnel

Rima Bouchakri*,**, Ladjel Bellatreche*

*LIAS/ISAE-ENSMA – Université de Poitiers
Futuroscope, 86960 - France
(rima.bouchakri, bellatreche)@ensma.fr
**Ecole nationale Supérieure d'Informatique (ESI)
Alger, Algérie
r_bouchakri@esi.dz

Résumé. La fragmentation horizontale permet de réduire la complexité des requêtes décisionnelles exécutées sur un entrepôt de données relationnel. Elle se base sur le principe de réorganisation des données qui ne nécessite pas un espace de stockage supplémentaire. Cependant, sélectionner un schéma de fragmentation horizontale d'un entrepôt n'est guère une tâche facile, vu l'espace de recherche très complexe à exploiter. Les algorithmes existants sélectionnent un schéma de fragmentation lors de la phase de conception physique d'un entrepôt, afin d'optimiser une charge de requêtes préalablement connue. Ces algorithmes ne prendre pas en considération les changements au niveau des requêtes. Dans cet article, nous proposons d'effectuer *une sélection d'un schéma de fragmentation dite incrémentale* basée sur les algorithmes génétiques. Notre approche permet l'optimisation de l'exécution de la charge de requêtes décisionnelles et l'adaptation du schéma de fragmentation aux changements de la charge. Nous réalisons une étude expérimentale qui montre l'intérêt de la sélection incrémentale d'un schéma de fragmentation.

1 Introduction

Les entrepôts de données ($\mathcal{ED}$) permettent le stockage et la consolidation, en une seule localité, d'une quantité gigantesque d'information issues de plusieurs sources hétérogènes. Les entrepôts de données sont souvent modélisés par des schémas relationnels (ex. schéma en étoile). Un entrepôt de données est souvent interrogé par l'intermédiaire de requêtes décisionnelles complexes. Elles renferment des opérations de sélections, d'agrégations et des opérations de jointures dites en étoiles entre la table de faits et une ou plusieurs tables de dimensions. Afin d'optimiser ses opérations, il devient primordial d'employer des techniques d'optimisations, comme la fragmentation horizontale ($\mathcal{FH}$) qui a déjà fait ses preuves dans le contexte des bases de données traditionnelles et d'entreposage de données. Brièvement, la fragmentation horizontale permet de répartir les instances d'une table, d'un index ou d'une vue matérialisée en un ensemble de partitions disjointes appelés *fragments horizontaux* [10].

Initialement, la fragmentation horizontale a été utilisée lors de la conception logique des bases de données relationnelles et orientées objet [4]. Actuellement, elle est considérée comme une structure d'optimisation utilisée au niveau physique. Deux points importants caractérisent la fragmentation horizontale : (1) elle est considérée comme une structure d'optimisation non redondante car elle ne nécessite pas un espace de stockage supplémentaire [6] et (2) elle est appliquée lors de la création de l'entrepôt de données. Deux types de fragmentation horizontale existent : (a) la fragmentation horizontale primaire $(\mathcal{FHP})$ et (b) la fragmentation horizontale dérivée $(\mathcal{FHD})$ [9]. La fragmentation horizontale primaire permet de répartir les tuples d'une table suivent la conjonction de prédicats de sélection définis sur les attributs de la même table. Elle favorise les opérations de sélection portées sur les attributs de fragmentation. La fragmentation horizontale dérivée quant à elle, permet de fragmenter une relation suivant la fragmentation horizontale primaire d'une seconde relation, à condition de l'existence de relation père-fils entre les deux tables. Ainsi, une requête qui contient un prédicat de sélection sur un attribut de fragmentation est optimisée par la fragmentation horizontale primaire car l'optimiseur ne charge que les fragments concerné par le prédicat de sélection. La fragmentation horizontale dérivée optimise à la fois les opérations de sélection de jointure. Plusieurs travaux se sont intéresses à la fragmentation horizontale des entrepôts de données. Ils peuvent être classés en deux catégories selon le type d'algorithmes employés : les algorithmes à base d'affinité et de minimalité et complétude de prédicats et les algorithmes à base de modèle de coût. *(1) Les algorithmes à base d'affinité et de minimalité et complétude de prédicats* : Les auteurs dans [2] adaptent les travaux de [11] et de [8] sur la fragmentation horizontale des bases de données traditionnelles, dans le contexte des entrepôts de données. Ils proposent d'employer l'algorithme d'affinité à base de regroupement binaire pour regrouper les prédicats de sélection par affinité tel que chaque groupe de prédicats représente un fragment horizontal. Un fragment résiduel est défini en construisant la négation de la disjonction des différents prédicats de fragmentation, afin d'éviter la perte de données lors de la fragmentation. Les auteurs adaptent également les travaux de [8] basés sur la complétude et la minimalité des prédicats afin de spécifier les différents fragments horizontaux. Le problème avec les algorithmes à base d'affinité et de minimalité est qu'ils ne permettent pas d'évaluer le schéma de fragmentation sélectionné. De plus, il n'est pas possible de contrôler le nombre de fragments générés. Quand ce nombre est important, l'administration de l'entrepôt devient une tâche difficile, d'où la nécessité d'utiliser un modèle de coût afin d'évaluer les schémas de fragmentation horizontale qui doivent être générés sous une contrainte sur le nombre maximum de fragments fait. *(2) Algorithmes basés sur le modèle de coût* : un modèle de coût est employé afin d'estimer le taux de réduction de la complexité des requêtes décisionnelles par le schéma de fragmentation horizontale de l'entrepôt de données. La fragmentation horizontale est réalisée par une $\mathcal{FHP}$ des tables dimensions suivie par une $\mathcal{FHD}$ de la table de faits selon le schéma de fragmentation des dimensions. Les travaux de cette catégorie se basent sur la formalisation du problème de sélection d'un schéma de fragmentation horizontale des dimensions sous forme d'un problème d'optimisation qui est prouvé NP-Complet [2]. Les auteurs dans [2] proposent de réduire la complexité du processus de fragmentation en réduisant le nombre de table de dimensions à fragmenter. Ils formalisent ainsi un problème de sélection des tables de dimension à fragmenter. La sélection est effectuée par un algorithme glouton guidé par modèle de coût. Les auteurs dans [2] formalise le problème en un problème de sélection d'un schéma de fragmentation des tables dimensions sous une contrainte d'optimisation qui représente le nombre maximum de fragments fait à générer

(contrainte définie par l'administrateur de l'entrepôt de données). Les auteurs emploient des heuristiques (Hill Climbing, Recuit Simulé) ainsi que les algorithmes génétiques afin de sélectionner le schéma de fragmentation horizontale optimal (quasi-optimal). Les auteurs dans [7] proposent une approches de fragmentation horizontale d'un entrepôt de données XML en effectuant une classification des prédicats de sélection à travers des techniques de fouilles de données, ainsi, chaque classe représente un fragment. Lors de l'analyse de ces travaux, nous avons constaté que les différentes approches de sélection d'un schéma de fragmentation pour l'optimisation d'une charge de requêtes exécutées sur un entrepôt de données, se base sur *une sélection statique* et supposent la connaissance préalable de ces requêtes. Par conséquent, rien ne garantit l'efficacité du schéma de $\mathcal{FH}$ de l'entrepôt après l'évolution de la charge de requêtes (ajout de nouvelle requête, changement de la fréquence d'exécution d'une ou plusieurs requêtes). En effet, si un attribut "Genre" n'est plus souvent employé pour interroger l'entrepôt, pourquoi maintenir un schéma de fragmentation défini sur cet attribut, particulièrement si une contrainte sur le nombre maximum de fragments fait à générer a été définie par l'administrateur. Il serait plus judicieux de fusionner les fragments définis sur l'attribut "Genre" et de re-fragmenter l'$\mathcal{ED}$ suivant un autre attribut plus fréquemment employé par les requêtes décisionnelles. Ainsi, nous proposons dans cet article *une sélection d'un schéma de $\mathcal{FH}$ dites incrémentale*. Elle se base sur la réadaptation du schéma de $\mathcal{FH}$ des tables dimensions selon les changements survenus sur la charge de requêtes, cela en employant les algorithmes génétiques. Cet article est structuré comme suit : la section 2 est dédiée au problème de sélection d'un schéma de $\mathcal{FH}$ des entrepôts de données. La section 3 aborde la sélection statique d'un schéma de $\mathcal{FH}$ par algorithmes génétiques. Dans la section 4, nous présentons notre approche de sélection incrémentale et présentons l'architecture et l'algorithme génétique mis au point. Enfin, nous terminons par une étude expérimentale qui montre l'intérêt de notre approche pour faire face à l'évolution de la charge de requêtes. La section 6 conclut l'article.

2 Problème de sélection d'un schéma de $\mathcal{FH}$

Les requêtes décisionnelles exécutées sur l'$\mathcal{ED}$ sont dites requêtes de jointures en étoiles car elles contiennent des opérations de jointures entre la table de faits volumineuse et plusieurs tables dimensions, et des sélections sur les dimensions. Dans leurs travaux, les auteurs dans [2] montrent que le meilleur scénario de $\mathcal{FH}$ d'un $\mathcal{ED}$ est la $\mathcal{FHP}$ des dimensions suivie d'une $\mathcal{FHD}$ de la table de faits selon le schéma de $\mathcal{FHP}$ des dimensions, ce qui optimise le coût d'exécution des requêtes de jointures en étoiles. Soit un $\mathcal{ED}$ modélisé en étoile ayant d tables de dimension $\mathcal{D} = \{D_1, D_2, ..., D_d\}$ et une table des faits $\mathcal{F}$. Le nombre de fragments faits $N = \prod_{i=1}^{g} m_i$, où m_i et g représentent respectivement le nombre de fragments de la table D_i, et le nombre de tables dimension participant dans le processus de fragmentation. Par conséquent, la fragmentation d'un $\mathcal{ED}$ donne lieu à la création de N sous schémas en étoile où chaque sous schéma contient un fragment faits et les fragments des tables dimensions pouvant être joints avec ce fragment. De ce fait, l'administrateur de l'$\mathcal{ED}$ aura pour tâche de gérer N sous schémas en étoiles, ce qui devient difficile à administrer. Il est impératif donc de contrôler le nombre de fragments faits générés. Le problème de sélection d'un schéma de $\mathcal{FHP}$ des dimensions peut être formalisé comme suit : Étant donné : (1) un $\mathcal{ED}$ modélisé par un schéma en étoile ayant d tables de dimension $\mathcal{D} = \{D_1, D_2, ..., D_d\}$ et une table des faits $\mathcal{F}$, (2) une charge de requêtes $Q = \{Q_1, Q_2, ..., Q_t\}$ à partir de laquelle est définie les attributs de

fragmentation $AS = \{A_1, \cdots, A_n\}$ [1] et (3) un seuil W représentant le nombre maximum de fragments de la table faits dans le schéma final de $\mathcal{FH}$. Le problème consiste à générer un schéma de $\mathcal{FH}$ primaire des tables dimensions défini sur l'ensemble AS tel que : (a) une fois la table de faits fragmentée par une $\mathcal{FHD}$ suivant le schéma de $\mathcal{FHP}$ des tables dimensions, le nombre de fragments faits ne dépasse pas W et (b) le coût total d'exécution des requêtes sur le schéma fragmenté est réduit. Formalisé ainsi, ce problème est NP-Complet, de ce fait les travaux [2] proposent d'employer l'algorithme génétique pour la sélection d'un schéma de $\mathcal{FH}$ des dimensions. A la base de ces travaux, nous avons défini la sélection statique d'un schéma de $\mathcal{FHP}$ que nous détaillons dans la section suivante.

3 Sélection statique d'un schéma de $\mathcal{FH}$ par Algorithmes Génétiques

Dans cette section, nous présentons une démarche de sélection statique d'un schéma de $\mathcal{FH}$ en utilisant les algorithmes génétiques (AGs). Les AGs sont des algorithmes itératifs, évolutionnaires qui s'inspirent de la théorie de l'évolution pour résoudre des problèmes d'optimisation [1]. Ils font évoluer un ensemble de solutions à un problème donné, dans le but de trouver la solution optimale. A chaque itération du AG, appelée aussi génération, une population de chromosomes (individus) est manipulée. Sur cette population, des opérateurs génétiques (croisement, mutation et sélection) sont appliqués afin de créer une nouvelle population qui optimise d'avantage une fonction objectif. Au fur et à mesure des générations, les individus vont tendre vers l'optimum de la fonction objectif. Chaque chromosome représente une solution potentielle au problème d'optimisation. Dans notre cas, un chromosome est un schéma de $\mathcal{FHP}$ des tables dimension. Par conséquent, La difficulté réside dans la définition d'un codage du schéma de $\mathcal{FH}$ qui permet de caractériser le chromosome de l'algorithme génétique. Dans un schéma de fragmentation, chaque fragment horizontal est caractérisé par une conjonction de prédicats définis sur les attributs de fragmentation (attributs de sélections issus des tables dimensions). Chaque attribut de fragmentation possède un domaine de valeurs. Les clauses des fragments horizontaux définissent le découpage du domaine de chaque attribut en plusieurs sous domaines. Par exemple, pour l'attribut Genre, le domaine peut être découpé en deux sous domaines comme suit : Dom(Genre)='M','F'. Ainsi, un schéma de $\mathcal{FH}$ peut être représenté sous forme d'un tableau de vecteurs, où chaque vecteur caractérise un attribut et où les cellules d'un vecteur représentent les différents sous domaine de l'attribut. Ce codage permet, ainsi, de représenté un schéma de $\mathcal{FH}$ sous forme d'un chromosome.

Exemple 1 *Soit un entrepôt de données composé d'une table de faits Vente et les tables dimensions Client et Produit. Considérons trois attributs de fragmentation : Genre et Ville de la dimension Client, et Produit de la table Produit. Les sous domaines de chaque attribut, représentés dans le tableau 1.a, sont donnés comme suit : Dom(Genre)={'M','F'},*
Dom(Produit)={'P1','P2','P3'} et Dom(Ville)={'Alger','Oran', 'Blida', 'Kala'}. Le codage d'un chromosome est donné par le tableau 1.b. Ce codage permet d'attribuer à chaque sous domaine un numéro, les sous domaines qui possèdent le même numéro sont fusionnés. Ainsi, pour l'attribut Ville par exemple, le sous domaine Alger possède la valeur 1, la valeur 2 est

[1] Un attribut de fragmentation figure dans la clause WHERE, dans les prédicats de sélection des requêtes

(a)

Genre	M	F			
Produit	P 1	P2	Reste		
Ville	Alger	Oran	Blida	Kala	Reste

(b)

Genre	1	2			
Produit	1	2	2	2	
Ville	1	2	3	3	3

TAB. 1 – *Exemple du codage d'un chromosome*

attribuée au sous domaine Oran et la valeur 3 permet de regrouper en un seul sous domaine les valeurs : " Blida ", " Kala " et " Reste ". Le sous domaine nommé $Reste$ *est ajouté afin de prendre en compte les valeurs d'attributs existant dans l'entrepôt mais ne figurant pas dans les requêtes, ce qui permet d'assurer la complétude des fragments générés. Le chromosome, donné par le tableau 1.b, donne un schéma de* $\mathcal{FH}$ *avec 6 fragments de la table Client (2 fragments sur Genre et 3 sur Ville) et 2 fragments sur Produits, soit un total de 12 (*6×2*) fragments faits.*

Dans le but de réaliser la sélection du schéma de $\mathcal{FH}$ par algorithme génétique, il faut définir la fonction objectif qui permet d'évaluer les chromosomes exploités par l'algorithme. L'évaluation d'un chromosome Ch_i est basée sur le calcul du coût d'exécution des requêtes sur l'entrepôt fragmenté selon Ch_i. Nous définissons le modèle de coût en s'inspirant du modèle avancé dans [2]. Ce modèle permet de calculer le nombre d'E/S nécessaires pour exécuter une charge de requêtes sur un schéma partitionné par $\mathcal{FH}$. Chaque requête comporte des prédicats de sélection et des jointures en étoile entre table de faits et tables de dimension. Nous considérons un entrepôt de données modélisé en étoile avec une table de fait $\mathcal{F}$ et d table de dimensions $D = \{D_1, D_2, ..., D_d\}$. La fragmentation de l'entrepôt selon le schéma $\mathcal{SF}$ donne N sous schéma en étoiles $\mathcal{SF} = S_1, ..., S_N$. Soit une charge de t requêtes $Q = \{Q_1, Q_2, ..., Q_t\}$ à exécuter sur l'entrepôt. Le modèle de coût présenté calcule le nombre de pages qu'il faut charger pour l'exécution de la requête Q_k sur le schéma $\mathcal{SF}$. Ce coût représente la somme des coût d'exécution de Q_k sur chaque sous schéma S_i. Dans un sous schéma en étoile S_i, un fragment fait est spécifié par un ensemble de M_i prédicats de sélection PF_j. Un fragment de la dimension D_s est également spécifié par L_s prédicats de sélection PM_j^s. On définit le coût de chargement d'un fragment fait et un fragment dimension par les formules suivantes : pour un fragment fait : $\prod_{j=1}^{M_i} Sel(PF_j) \times |F|$, pour un fragment de la dimension D_s :$\prod_{j=1}^{L_s} Sel(PM_j^s) \times |D_s|$, où $|R|$ et $Sel(P)$ représentent respectivement le nombre de pages nécessaires pour stocker la table R et le facteur de sélectivité du prédicat de sélection P. On suppose que les facteurs de sélectivité sont définis à travers une répartition uniforme (RU). Le coût d'exécution d'une requête Q_k sur un sous schéma S_i estime le coût de chargement du fragment fait puis la jointure entre ce fragment et les fragments dimensions. On considère le coût de jointure par hachage donné par l'équation suivante :

$$Cost(Q_k, S_i) = [3 \times [\prod_{j=1}^{M_i} Sel(PF_j) \times |F| + \sum_{k=1}^{d} \prod_{j=1}^{L_s} Sel(PM_j^s) \times |D_s|]] \quad (1)$$

Afin d'estimer le coût d'exécution de Q_k sur un schéma fragmenté, il faut identifier les sous schéma valides pour la requête. Un sous schéma valide est un sous schéma où le fragment fait est accédé par la requête sur au moins un tuple. Soit NS_k le nombre de sous schémas valides

pour Q_k. Le coût total d'exécution de Q_k sur tout l'entrepôt fragmenté est : $Cost(Q_k, \mathcal{SF}) = \sum_{j=1}^{NS_k} Cost(Q_k, S_j)$. Ainsi, le coût d'exécution de toute la charge de requête (t requêtes) sur l'entrepôt fragmenté est donné comme suit :

$$Cost(Q, SF) = \sum_{k=1}^{t} Cost(Q_k, SF) \tag{2}$$

Une fois le modèle de coût présenté, nous pouvons annoncer la fonction objectif. L'algorithme génétique exploite une population de chromosomes (schéma de $\mathcal{FH}$) dans chaque étape d'exécution. Supposant l'espace de m solutions ou schémas de fragmentations possibles générés par l'algorithme génétique $SF_1, ..., SF_m$. Etant donné une contrainte sur le nombre de fragments fait maximum W, l'algorithme génétique peut générer des solutions SF_i dont le nombre de fragments dépasse W. Par conséquent, ces schémas doivent être pénalisés. La fonction pénalité d'un schéma de $\mathcal{FH}$ est égale à :$Pen(SF_i) = \frac{N_i}{W}$, où N_i est le nombre de sous schémas de SF_i, sachant que le nombre de sous schéma représente également le nombre de fragments de la tables de faits. Enfin, l'algorithme génétique doit sélectionner un schéma de $\mathcal{FH}$ qui minimise la fonction objectif suivante :

$$F(SF_i) = \{^{Cost(Q,SF_i) \times Pen(SF_i), si Pen(SF_i) > 1}_{Cost(Q,SF_i), sinon} \tag{3}$$

Afin d'exploiter l'algorithme génétique, nous avons utilisé une API JAVA nommé JGAP *JGAP* [2] (Java Genetic Algorithms Package) qui permet d'implémenter cet algorithme. Le déroulement du processus de sélection d'un schéma de $\mathcal{FH}$ par AG est le suivant : (1) Codage du schéma de fragmentation, (2) définition de la fonction objectif et (3) sélection du schéma de $\mathcal{FH}$ optimal (quasi-optimal) par AG : L'algorithme génère une population initiale de chromosome, puis applique dessus les opérations génétiques (croisement, mutation et sélection) afin de générer les nouvelles populations. Chaque chromosome est évalué par la fonction objectif afin d'estimer le coût d'exécution des requêtes sur un $\mathcal{ED}$ fragmenté selon le schéma de $\mathcal{FH}$ décrit par le chromosome. Le meilleur schéma de fragmentation est sélectionné en fin de processus. Nous présentons l'algorithme de sélection d'un schéma de $\mathcal{FH}$ par algorithme génétiques :

[2] http ://jgap.sourceforge.ne

Algorithme de sélection d'un schéma de $\mathcal{FH}$
Entrées :
Q : charge de m requêtes
AS : ensemble de n attributs de fragmentation, $AS = \{A_1, \cdots, A_n\}$
Dom : l'ensemble des domaines des attributs de fragmentation, $Dom = \{Dom_1, \cdots, Dom_n\}$
$mathcalED$: données relatives au modèle de coût (taille des tables, page système, etc.)
W : contrainte sur le nombre maximum de fragments faits à générer
Sortie : Schéma de $\mathcal{FH}$ des tables dimensions SF. **Notations :**
$Coder\_Chromosome$: Coder le chromosome en un schéma de fragmentation
$FitnessFH$: fonction objectif pour l'AG
$JGAP$: API JAVA qui permet d'implémenter l'algorithme génétique
Début
$ChromosomeFH = Coder\_Chromosome(AS, Dom)$;
$FitnessFH = Genetic\_FitnessFonction(Q, AS, W, mathcalED)$;
$SF = JGAP(ChromosomeFH, FitnessFH)$;
Fin

4 Sélection incrémentale d'un schéma de $\mathcal{FH}$ par Algorithmes Génétiques

Les travaux qui traitent du problème de la sélection d'un schéma de $\mathcal{FH}$ se basent sur une sélection statique. Elle permet de réaliser la sélection d'une structure d'optimisation lors de la phase de conception physique d'un ED. Par conséquent, cette sélection ne permet pas de faire face aux changements qui surviennent sur l'ED principalement l'ajout de nouvelles requêtes. Afin de réaliser la sélection incrémentale d'un schéma de $\mathcal{FH}$, il faut réadapter le schéma de $\mathcal{FH}$ en cours en prenant en compte l'exécution d'une nouvelle requête Q_i (qui n'existe pas dans la charge de requêtes). L'exécution de Q_i peut provoquer l'ajout de nouveaux attributs de fragmentation ou d'éventuelle extension des domaines des attributs. Cela peut conduire à réaliser des fusions ou éclatements des fragments de l'entrepôt. Sous le SGBD Oracle, il est possible de réadapter un ED fragmenté selon un nouveau schéma de $\mathcal{FH}$ en utilisant les opérations SPLIT PARTITION ou MERGE PARTITION. L'opération MERGE PARTITION permet de combiner deux fragments en un seul et ainsi diminuer le nombre de fragments générés. L'opération SPLIT PARTITION quant à elle permet d'éclater un fragment en deux et augmenter ainsi le nombre de fragments générés.

Exemple 2 *Soit un $\mathcal{ED}$ contenant une table de faits Vente et une dimension Client fragmenté suivant le schéma illustré dans la figure 1.a. Les données de la table Client ainsi que son schéma de $\mathcal{FH}$ sont représentées dans la figure 1.b : Supposant l'exécution de la requête suivante :*

```
SELECT AVG(Prix)
```

Sélection incrémentale de fragments horizontaux

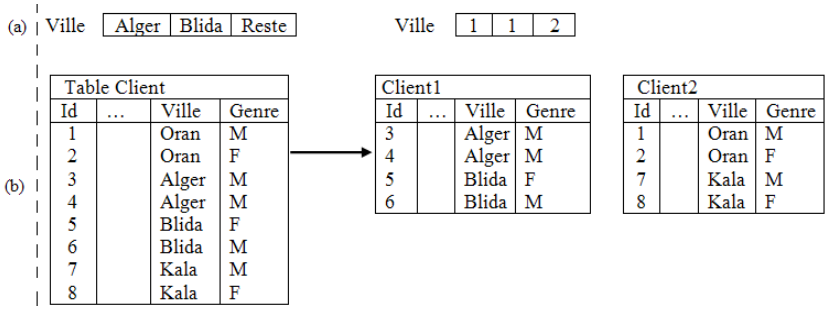

FIG. 1 – *La dimension Client fragmentée*

```
FROM Vente V, Client C
WHERE S.IdC=C.IdC AND C.Genre = 'F'
```

Afin de prendre en compte l'optimisation de cette requête, une nouvelle sélection d'un schéma de $\mathcal{FH}$ est réalisée, cette sélection donne le schéma de $\mathcal{FH}$ de la figure 2.a. L'adaptation du nouveau schéma de $\mathcal{FH}$ sur l'$\mathcal{ED}$ nécessite deux opérations MERGE sur les deux fragments de Client, le résultat de la nouvelle fragmentation de la table Client est donné sur la figure 2.b.

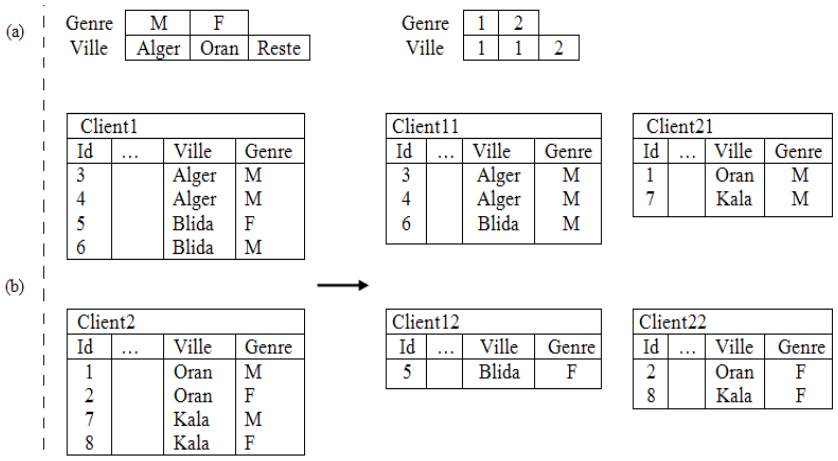

FIG. 2 – *La dimension Client fragmentée selon le nouveau schéma $\mathcal{FH}$*

Le principal challenge de la sélection incrémentale d'un schéma de $\mathcal{FH}$ est la mise en oeuvre d'une démarche de sélection qui prend en compte les changements survenus lors de l'évolution de la charge de requête. Dans un premier lieu, nous proposons de faire une sélection incrémentale dite Naïve ($FHNI$) qui se base sur des opérations de fusions et d'éclatement. Par la suite,

dans un souci de réadapter notre sélection statique d'un schéma de $\mathcal{FH}$ basée sur les AG, nous annonçons deux sélections incrémentales par algorithme génétique ($FHGA$ et $FHGA^*$).

4.1 Sélection incrémentale Naïve $FHNI$

Considérons un $\mathcal{ED}$ sur lequel un ensemble de requêtes sont exécutées successivement et continuellement. La sélection incrémentale naïve ($FHNI$) part d'un schéma de $\mathcal{FH}$ initial qui optimise la charge de requêtes en cours, représenté sous forme d'un chromosome. A l'arrivée d'une nouvelle requête Q_i, le chromosome est mis à jour suivant les attributs et valeurs contenu dans Q_i. Si la contrainte W est violée, une opération de fusion doit être réalisée afin de réduire le nombre de fragments généré. Cette fusion consiste à fusionner deux sous domaine d'un attribut donné en un seul sous domaine. Soit la table Client fragmenter en 3 fragments sur l'attribut Ville : Client1 : (Ville = 'Alger'), Client2 : (Ville = 'Oran') et Client3 : (Ville = Reste). La fusion des sous domaines Oran et Reste aboutie aux deux fragments : Client1 : (Ville = 'Alger') et Client2 : (Ville = 'Oran' ou Reste). Nous rappelons que le fragment " Reste " contient les autres villes qui ne sont pas utilisées par les requêtes mais qui existent dans la table Client, à savoir " Blida " et " Kala ". La mise en oeuvre de la démarche de sélection naïve s'inspire de la sélection statique par algorithmes génétiques. Nous utilisons le codage du chromosome annoncé précédemment pour représenter un schéma de $\mathcal{FH}$, puis lors de l'exécution d'une nouvelle requête Q_i, nous réalisons la sélection naïve à travers l'application des opérations suivantes sur le chromosome Ch :

1. Extraire à partir de Q_i les attributs de fragmentation A_j et leurs valeurs respectifs V_{jk} à partir des prédicats de sélection. Un prédicat P se présente comme suit : " Aj op V_{jk} ", où op est dans $\{=, <, >, <>, \leq, \geq\}$. Chaque valeur V_{jk} constitue un nouveau domaine de A_j. Supposant la présence de la requête suivante :

   ```
   SELECT AVG(Prix)
   FROM Vente V, Client C, Produit P
   WHERE V.IdC=C.IdC AND V.IdP=P.IdP
   AND P.Produit = 'P4'
   AND (C.Ville = 'Alger' OR C.Ville = 'Oran')
   ```

   A partir de cette requête, on extrait les attributs Produit, Ville et les valeurs P4, Alger et Oran.

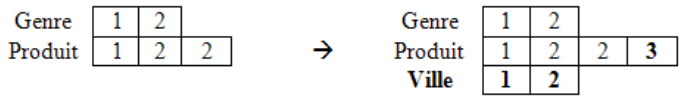

FIG. 3 – *Sélection incrémentale naïve : mise à jour du chromosome Ch*

2. Mettre à jour Ch suivant les attributs et leurs domaines obtenus dans (1). On attribue à chaque domaine un nouveau numéro. Selon la requête précédente, la mise à jour de Ch est donnée par la figure 3.

3. Si la contrainte W est violée (le nombre de fragments générés > W) : (a) Ordonner les attributs suivant leurs fréquences d'utilisation par les requêtes, du moins utilisé au plus utilisé. (b) Pour chaque attribut, ordonner les sous domaines suivant leurs fréquences d'utilisation par les requêtes, du moins utilisé au plus utilisé. (c) Fusionner les sous domaines des attributs jusqu'à obtention d'un schéma $\mathcal{FH}$ qui ne viole pas la contrainte W. Supposant l'ordre, Ville, Produit, Genre et une contrainte $W = 4$, les fusions successives appliquées sur Ch sont données par la figure 4. Le schéma $\mathcal{FH}$ résultant possède 4 fragments, 2 sur Produit et 2 sur Genre.

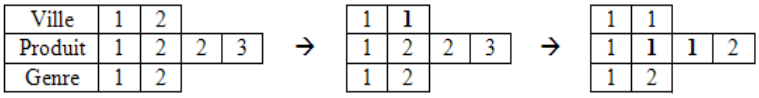

FIG. 4 – *Sélection incrémentale naïve : fusions successives appliquées sur* Ch

4.2 Sélection incrémentale par algorithme génétique $FHGA$

Nous avons exploité la sélection statique par algorithme génétique présenté précédemment. A l'exécution de chaque requête, le codage du chromosome est amélioré en prenant en compte les changements apportés par la requête. Considérons le schéma de fragmentation donné par la figure 5. Supposant l'exécution d'une nouvelle requête donc la spécification est la suivante :

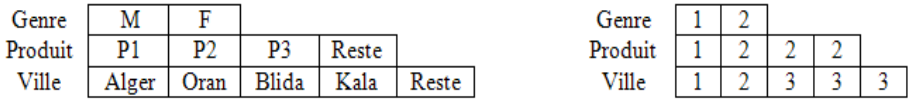

FIG. 5 – *Sélection incrémentale par* AG *: exemple d'un chromosome*

```
SELECT AVG(Prix)
FROM Vente S, Client C, Produit P
WHERE V.IdC=C.IdC AND V.IdP=P.IdP AND
P.Produit = 'P4' AND C.Pays = 'Algérie'
```

L'adaptation du chromosome est donnée par la figure 6. Après l'adaptation du codage du chro-

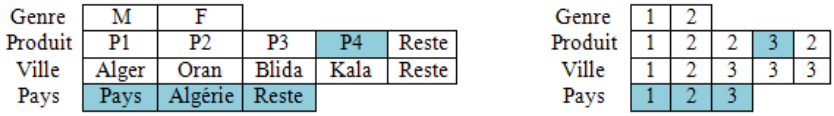

FIG. 6 – *Sélection incrémentale par* AG *: mise à jour du chromosome*

mosome aux changements survenus, la sélection d'un schéma de $\mathcal{FH}$ par AG est effectuée.

Le principal problème avec cette approche est que le nouveau schéma de $\mathcal{FH}$ est sélectionné par l'algorithme génétique sans prendre en compte le schéma actuel de fragmentation de l'entrepôt. En effet, afin d'adapter un nouveau schéma de $\mathcal{FH}$ sur un entrepôt déjà fragmenté, des opérations de fusions ou d'éclatements sont requises. Par conséquent, un nouveau schéma de $\mathcal{FH}$ sélectionné peut significativement améliorer les performances d'exécution des requêtes mais peut être très couteux du point de vu mise en oeuvre. En effet, afin d'implémenter un nouveau schéma de fragmentation sur un entrepôt déjà fragmenté, plusieurs opérations de fusions et éclatements des fragments sont requises (adaptation du schéma de fragmentation actuel de l'entrepôt afin qu'il corresponde au nouveau schéma sélectionné). Ainsi, nous proposons une nouvelle approche de sélection incrémentale par algorithmes génétique que nous annonçons dans la section suivante.

4.3 Sélection incrémentale par algorithmes génétiques améliorée $FHGA^*$

Afin de réduire le coût d'implémentation du nouveau schéma de $\mathcal{FH}$, tout en prenant en compte les modifications qui surviennent sur la charge de requête, nous proposons une nouvelle démarche de sélection incrémentale par algorithmes génétique dite améliorée ($FHGA^*$). Le principe est de réadapter la fonction objectif qui permet d'évaluer les différentes solutions générées par l'AG afin de pénaliser les solutions qui proposent des schémas de $\mathcal{FH}$ couteux du point de vu de l'implémentation effectives sur l'$\mathcal{ED}$ (nombre de fusions et éclatements importants). Ainsi, nous avons définis une fonction dite Fonction de Dissimilarité " FD " dont la signature est la suivante : $FD(SF_i)$ = nombre de fusion et éclatement nécessaires pour réadapter le schéma de fragmentation de l'entrepôt selon le schéma SF_i. Tel que SF_i représente le schéma de $\mathcal{FH}$ (chromosome) en cours d'exploitation par l'algorithme génétique durant son exécution. Dans la figure 7, nous présentons deux schémas de $\mathcal{FH}$, le schéma réel de l'entrepôt et un nouveau schéma en cours d'évaluation par l'algorithme génétique Pour cet

Schéma FH de l'ED

Genre	1	2			
Produit	1	1	2	2	2
Ville	1	2	3	3	3
Pays	1	2	3		

Schéma SF_i

Genre	1	2			
Produit	1	1	2	3	2
Ville	1	2	2	3	3
Pays	1	1	2		

FIG. 7 – *Sélection $FHGA^*$: schéma réel vs. nouveau schéma de $\mathcal{FH}$*

exemple, $FD(SF_i)$ = 1 éclatement sur Produit + 1 éclatement sur Ville + 1 Fusion sur Ville + 1 Fusion sur Pays = 4. Rappelons que la fonction objectif de l'algorithme génétique est donnée comme suit : $F(SF_i) = \{^{Cost(Q,SF_i)\times Pen(SF_i),siPen(SF_i)>1}_{Cost(Q,SF_i),sinon}$ avec $Pen(SF_i) = N_i/W$. Nous proposons de définir une nouvelle fonction de pénalité $Pen2(SF_i) = FD(SF_i)$. Ainsi, la nouvelle fonction objectif est donnée comme suit : $F'(SF_i) = F(SF_i) \times FD(SF_i)$.

Requête	Attributs
Q1	Group
Q2	Month, Quarter
Q3	Month, Class
Q4	City, Gender
Q5	Month, Year, City, Class
Q6	Class, Gender
Q7	City, Gender, Class
Q8	City, Gender, Group

TAB. 2 – *Description de la charge des 8 requêtes*

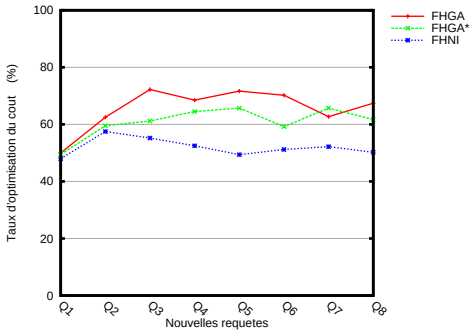

FIG. 8 – *Taux de réduction du coût d'exécution des requêtes (cas 8 requêtes)*

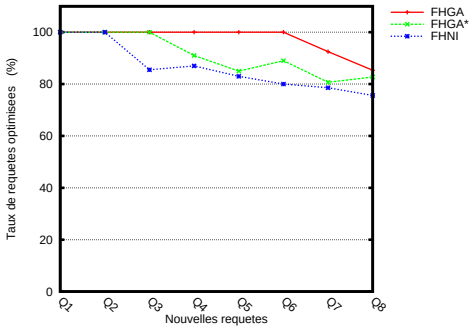

FIG. 9 – *Taux de requêtes optimisées (cas 8 requêtes)*

5 Expérimentation

Afin de comparer les différentes stratégies de sélection incrémentale d'un schéma de fragmentation, nous avons réalisé des tests de comparaison sur un ED réel issu du benchmark APB1 [3] sous le SGBD Oracle 11g. Cet entrepôt est composé d'une table de faits *Actvars*(24 786 000 tuples) et quatre tables de dimension *Prodlevel* (9000 tuples), *Custlevel* (900 tuples), *Timelevel* (24 tuples) et *Chanlevel* (9 tuples). Pour effectuer la sélection d'un schéma de fragmentation, nous avons utilisé l'API JGAP. Nos tests se déroulent sur deux phases : d'abords nous réalisons des tests à petite échelle sur une charge de 8 requêtes, par la suite nous menons des tests à grandes échelle sur une charge de 60 requêtes.

5.1 Tests à petite échelle

Afin de bien comprendre le fonctionnement de la sélection incrémentale, nous avons réalisé une première étude sur une charge de requête de jointure en étoile vide avec une contrainte $W = 40$. L'étude incrémentale est réalisée avec ajout successif de 8 nouvelles requêtes (Tableau 2). L'ajout de chaque requête déclenche le processus de sélection d'un schéma de $\mathcal{FH}$. Nous évaluons chaque schéma de $\mathcal{FH}$ sélectionné en notant le coût d'exécution des requêtes,

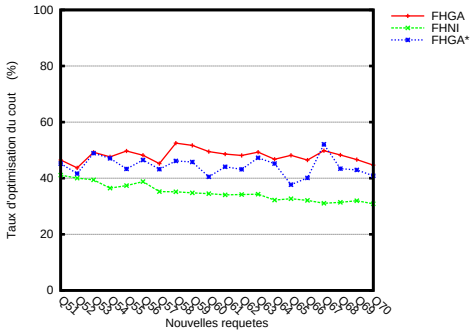

FIG. 10 – *Taux de réduction du coût d'exécution des requêtes*

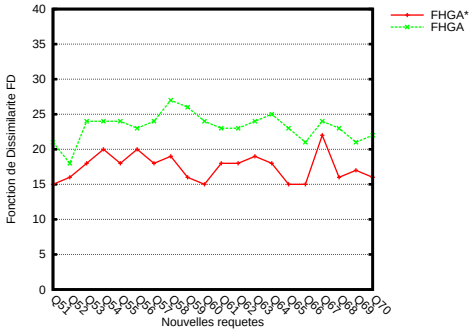

FIG. 11 – *Fonction de dissimilarité : $FHGA$ vs. $FHGA^*$*

d'où est calculé le taux d'optimisation du coût (figure 8), et le taux des requêtes optimisées par fragmentation (figure 9). Nous remarquons que les meilleurs résultats sont donnés par la démarche $FHGA$. En effet, le coût d'exécution est réduit de 70% pour 95% des requêtes optimisées. Par contre, $FHGA$ donne des schémas de $\mathcal{FH}$ qui nécessitent plusieurs opérations de fusions et éclatements pour l'implémentation sur $\mathcal{ED}$. Afin de mieux voir l'efficacité de notre stratégie incrémentale et pour une meilleure comparaison, nous avons effectué des tests à grande échelle.

5.2 Tests à grande échelle

Nous avons considéré une charge de requêtes de jointures en étoile contenant **60 requêtes** qui génèrent **18 attributs** de sélection *(Line, Day, Week, Country, Depart, Type, Sort, Class, Group, Family, Division, Year, Month, Quarter, Retailer, City, Gender et All)* avec les cardinalités : 15, 31, 52, 11, 25, 25, 4, 605, 300, 75, 4, 2, 12, 4, 99, 4, 2, 3 respectivement. L'étude incrémentale est réalisée avec ajout successif de 20 nouvelles requêtes en considérant une contrainte W = 100. Afin d'avoir une bonne base de comparaison, nous supposons la charge de 40 requêtes optimisées avec un schéma $\mathcal{FH}$ préalablement sélectionné par algorithmes génétiques. Pour chaque requête nouvellement exécutée, nous réalisons trois sélections incrémentales : une sélection incrémentale naïve $FHNI$, basée sur les algorithmes génétiques $FHGA$ et basée sur les algorithmes génétiques améliorée $FHGA^*$. Pour chaque sélection ($FHNI$, $FHGA$ et $FHGA^*$), et pour chaque nouvelle requête, nous relevons le coût d'exécution de toute la charge de requêtes en cours, à partir duquel est calculé le taux d'optimisation du coût (figure 10) Nous constatons que les deux démarches de sélection par algorithmes génétiques $FHGA$ et $FHGA^*$ apportent de meilleurs résultats que la démarche $FHNI$. En effet, $FHGA$ apporte en moyenne une réduction de 45% du coût d'exécution des requêtes et $FHGA^*$ donne une amélioration de 42% en moyenne contre 37% pour $FHNI$. En comparant les deux approches $FHGA$ et $FHGA^*$, nous remarquons que $FHGA$ donne de meilleurs résultats. En effet, dans la démarche de $FHGA^*$ les solutions, dissimilaires par rapport au schéma de $\mathcal{FH}$ courant de l'entrepôt, sont pénalisées. Se sont les schémas dont l'implémentation nécessite plusieurs opérations de fusions et éclatements des fragments de l'entrepôt. Ainsi,

dans $FHGA^*$ de bonnes solutions, qui permettent d'améliorer les performances des requêtes, peuvent être écartées par le processus de sélection. Cependant, dans la démarche $FHGA$, l'algorithme génétique peut sélectionner comme solution finale un schéma de $\mathcal{FH}$ très couteux en implémentation nécessitant plusieurs opérations de fusions et éclatements. Afin d'illustrer ce problème, nous avons relevé, durant la réalisation du test précédent, la valeur de la fonction de dissimilarité FD de chaque schéma final de $\mathcal{FH}$ sélectionné par les deux démarches $FHGA$ et $FHGA^*$. Le résultat est illustré sur la figure 11. La figure 11 montre clairement que les solutions finales sélectionnées par la démarche $FHGA$ nécessitent plus de fusions et d'éclatement de l'$\mathcal{ED}$ que les solutions apportées par la démarche $FHGA*$. De ce fait, en prenant en compte les deux paramètres importants à savoir : l'optimisation de la charge de requêtes et le coût de mise en oeuvre du schéma sélectionné, notre approche $FHGA*$ permet d'apporter les meilleurs résultats par rapport à $FHNI$ et $FHGA$.

6 Conclusion

Ce travail traite de la sélection incrémentale d'un schéma de fragmentation dans le contexte d'entrepôt de données en étoile. Nous avons proposé une sélection incrémentale d'un schéma de $\mathcal{FH}$ par algorithmes génétiques qui, contrairement à la sélection statique, permet de prendre en considération l'évolution de la charge de requêtes. Afin d'utiliser ces derniers, nous avons proposé un codage du chromosome qui doit représenter un schéma de $\mathcal{FH}$. Trois stratégies de sélection ont été proposées : (1) la sélection naïve $FHNI$ qui permet, à travers des opérations simples, d'adapter le schéma de fragmentation de l'entrepôt aux changements survenus après exécution d'une nouvelle requête, (2) la sélection par algorithme génétique $FHGA$ effectue la sélection d'un nouveau schéma de fragmentation à la base du schéma courant de l'$\mathcal{ED}$ et des changements survenus, et (3) la sélection par algorithme génétique améliorée $FHGA^*$, qui pallie au problème de la sélection $FHGA$ du fait qu'elle pénalise les solutions coûteuses en implémentation effective sur l'entrepôt (nombre de fusions et éclatements nécessaires pour adapter le schéma de $\mathcal{FH}$ sélectionné). Nos stratégies de sélection de schémas incrémentaux de $\mathcal{FH}$ ont été validées par une étude expérimentale. Les résultats obtenus montrent que la sélection incrémentale par GA améliorée est un bon compromis entre la sélection naïve et la sélection par algorithme génétique. Elle donne une meilleure optimisation de la charge de requêtes tout en réduisant le coût de mise en oeuvre de la fragmentation. Ce travail prend en considération les changements au niveau requêtes. Une des perspectives est de considérer d'autres types de changements qui peuvent exister dans un environnement d'entreposage de données : le changement de la fréquence d'accès des requêtes [5], des modifications au niveau de la population de l'entrepôt de données, des modifications au niveau de la structure de l'entrepôt de données (nombre d'attributs, leurs domaines, etc.).

Références

[1] T. Bäck. *Evolutionnary algorithms in theory and practice*. Oxford University Press, New York, 1995.

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

[3] OLAP Council. Apb-1 olap benchmark, release ii. *http ://www.olapcouncil.org/research/bmarkly.htm*, 1998.

[4] K. Karlapalem and Q. Li. A framework for class partitioning in object-oriented databases. *Distributed and Parallel Databases*, 8(3) :333–366, 2000.

[5] K. Karlapalem, S. B. Navathe, and M. H. Ammar. Optimal redesign policies to support dynamic processing of applications on a distributed relational database system. *Information Systems*, 21(4) :353–367, 1996.

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

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

[8] M. T. Özsu and P. Valduriez. Distributed database systems : Where are we now ? *IEEE COMPUTER*, 24(8) :68–78, August 1991.

[9] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems : Second Edition*. Prentice Hall, 1999.

[10] A. Sanjay, V. R. Narasayya, and B. Yang. Integrating vertical and horizontal partitioning into automated physical database design. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 359–370, June 2004.

[11] Y. Zhang and M-E. Orlowska. On fragmentation for distributed database design. *Information Sciences*, 1(3) :117–132, 1994.

Summary

The horizontal data partitioning is a technique that contributes in improving the cost of executing OLAP queries defined on the top of relational data warehouses. It is considered as a non redundant optimization technique. Due to the hardness of selecting an optimal fragmentation schema for a given data warehouse, several research studies exist and propose heuristics to select near optimal solutions. Most of these studies assume the existence of a priori known set of queries. Note that in real life applications, queries may change dynamically and fragmentation heuristics need to integrate these changes. In this paper, we propose *an incremental selection of fragmentation schemes* using on genetic algorithms. Intensive experiments are conducted to validate our proposal.