الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Mémoire de MASTER

Domaine : Informatique
Filière : Informatiques
Option : System d'information décisionnelle

Par:
SahliAmina

THEME

L'algorithme PSO pour la fragmentation verticale

Soutenu publiquement le 12-06-2017 devant le jury composé de:

M. Atika SAHLAOUI	*M.A.A*	*Président*
Mr. Benameur ZIANI	*M.C.B*	*Examinateur*
Mr.Lardaj CHELLAMA	*M.A. A*	*Encadreur*

Année Universitaire 2016/2011

Remercîment

Je tiens à exprimer toute ma reconnaissance à mon encadreur mémoire Messieurs LARADJ CHELLAM. Je Le remercie de m'avoir encadré, orienté, aidé et conseillé. Ainsi toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet.

Je remercie mes très chers parents, ABDELI KADER et RABIA qui ont toujours été là pour moi.

Je remercie mes frères, et ma sœur KHADIDJA pour leur encouragement. Ainsi que mes grands parents, mes oncles et mes tentes et toute ma famille (SAHLI, TAHARI)

Je remercie très spécialement ABLA et MAROUA qui ont toujours été là pour moi et pour leur amitié, leur soutien inconditionnel et leur encouragement.

Enfin, je remercie tous mes Ami(e)s que j'aime tant, et sons oublier mes collègues avec qui j'ai partagé c'est années d'études

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

A mes très chers parents

Résumé

L'optimisation est une branche des mathématiques, elle aide à résoudre des problèmes en déterminant le meilleur élément d'un ensemble selon certains critères prédéfinis. De ce fait, l'optimisation est omniprésente dans tous les domaines et évolue sans cesse depuis longtemps.

La fragmentation consiste à partitionner une base de données en plusieurs sous schémas ou bien fragment dans le but de les allouer dans des sites adéquats afin d'accélérer la réponse à des requêtes utilisateurs locaux.

Plusieurs techniques ont été proposées dans la littérature pour optimiser cette tâche, soit pour la fragmentation horizontale ou bien verticale.

Dans cette optique s'inscrit notre travail qui vise à appliquer une technique innovante inspirée de monde des oiseaux et basée sur les essaims des particules (PSO), elle consiste à fournir un schéma de fragmentation verticale plus optimal qui va nous aider à améliorer la performance des requêtes réparties.

Nous évaluons notre application pour avoir des fragments plus optimaux à travers d'expérimentations réalisées à l'aide du langage Java sous l'environnement NetBeans.

Mots clé : Fragmentation verticale, PSO, requêtes repartiés..

Abstract

Optimization is a branch of mathematics, it helps to solve problems by determining the best element of a set according to certain predefined criteria. As a result, optimization is omnipresent in all domains and evolves constantly since a long time.

Fragmentation involves partitioning a database into several sub-schemas or fragmented for the purpose of allocating them in suitable sites in order to speed up the response to local user requests.

Several techniques have been proposed in the literature to optimize this task, either for horizontal or vertical fragmentation.

Our work aims to apply an innovative technique inspired by the world of birds and based on the swarms of particles (PSO), it consists in providing a more optimal vertical fragmentation scheme which will help us to improve the performance of distributed queries.

We evaluate our application to have more optimal fragments through experiments carried out using the Java language under the NetBeans environment.

Keywords: Vertical fragmentation, PSO, queries.

الملخص

الإستمثال هو فرع من الرياضيات، فهو يساعد على حل المشاكل من خلال تحديد أفضل عنصر في المجموعة وفقا لمعايير محددة مسبقا,ولهذا نجد الإستمثال في جميع المجالات كما انه في تطور مستمر منذ القدم.

تجزئة هي عملية تقسيم قاعدة البيانات إلى عدة أنماط فرعية أو شظية من أجل تخصيص المواقع المناسبة و تسريع الرد على استفسارات المستخدمين المحليين.

عدة تقنيات تم اقترحها عدة تقنيات منذ القدم من اجل تحسين هذه العملية أما في التجزئة الأفقية أو عمودية.

في هذا السياق يأتي عملنا الذي يهدف لتطبيق تكنولوجيا مبتكرة مستوحاة من عالم الطيور، و التي تستند على أسراب الجسيمات (PSO),تعمل على توفير أحسن نمط للانقسام العمودي التي من شأنها ان تساعدنا على تحسين أداء الاستعلامات الموزعة.

نقيم عملنا الذي ينص على الحصول على أحسن و امثل تقسيم من خلال التجارب التي أجريت باستخدام لغة الجافا في بيئة نتبينس .

.

Sommaire

Introduction 1

Chapitre 1 : la fragmentation verticale

1.1 Introduction 3

1.2 Définition de fragmentation 3

1.2.1 Le partitionnement verticale 5

1.2.2 Le partitionnement horizontale 6

1.2.3 Le partitionnement hybride 7

1.3 Fragmentation verticale 8

1.3.1 Problème de la fragmentation verticale 9

1.3.1 Traveux existants 9

1.3.3 Techniques de fragmentation 11

1.4 Conclusion 13

Chapitre 2 : Particle swarm optimization

2.1 Introduction 15

2.2 Historique 15

2.3 PSO pour la répartition des données 16

2.4 Les éléments de la P.S.O. 16

2.5 Principe fondamental 18

2.6 PSO pour optimisation combinatoire 19

2.6.1 Définition d'une particule 19

2.6.2 Vitesse (velocity) 20

2.6.3 Construction d'une solution de particules 21

2.7 Fonctions objectives pour la partition verticale 22

2.8 Conclusion 23

Chapitre 3 : Implémentation

3.1 Introduction 25

3.2 Algorithme implémenté 25

3.3 Environnent expérimentale 26

3.4 Conclusion 30

conclution et perspective 32

Bibography 33

Liste de figure

Figure 1.1 Evolution de la fragmentation de données (1) 4
Figure 1.2 Exemple d'une fragmentation verticale (1).....5
Figure 1.3 Exemple de la fragmentation horizontale (1).....6
Figure 1.2 Fagmentation hybride 8
Figure 2.1Schéma de principe du déplacement d'une particule. (12) 18
Figure 3.1 l'ajout des table dans l'application 27
Figure 3.2 l'ajout des attributs dans l'application 28
Figure 3.3 la matrice d'usage 28
Figure 3.4 la matrice de fréquence d'accès 29
Figure 3.5 le resultat 29

Introduction Générale

Introduction

La fragmentation est une technique de conception physique introduite dans les bases de données réparties. Elle consiste à partitionner une table verticalement / horizontalement de manière à réduire le nombre d'accès nécessaires pour le traitement de certaines requêtes. Dans ce cadre, nous nous sommes intéressés à la fragmentation verticale qui répond au problème de réduction du temps d'exécution des requêtes réparties.

Notre objectif visé par ce travail, consiste à fournir un schéma de fragmentation optimal qui permet d'optimiser la performance des requêtes. Nous proposons l'utilisation d'une nouvelle technique récemment cité dans la littérature de l'optimisation.

L'algorithme proposé, Particles Swarm Optimization, a pour but la production des fragments cohérents et performants.

A cet effet, notre mémoire est organisé comme suit :

Le premier chapitre s'articule autour des bases de données réparties ainsi que les différents type de fragmentations en particulier la fragmentation verticales et les travaux existants traitants cet axe.

Le second chapitre traite les notions de base de la technique PSO et en détaillants l'algorithme avec tous ces paramètres.

Le troisième chapitre présente notre démarche d'application de l'algorithme PSO pour répondre à la fragmentation d'une table donnée, aussi nous exposons les étapes à suivre pour avoir les fragments optimaux. Enfin, les résultats de notre expérimentation s'avèrent prometteuses.

En conclusion, une synthèse de notre travail sera présentée ainsi qu'une éventuelle perspective.

Chapitre 1 :

Fragmentation vertical

Chapitre 1 : La fragmentation vertical

1.1 Introduction

La fragmentation (le partitionnement) est une fonction du serveur de base de données qui permet de contrôler l'emplacement de stockage des données au niveau de la table. La fragmentation vous permet de définir des groupes de lignes ou des clés d'index dans une table, conformément à un algorithme ou un schéma. Donc La fragmentation est le processus de décomposition d'une base de données en un ensemble de sous-bases de données. Cette décomposition doit être sans perte d'information. Le partitionnement de tables est généralement effectué pour améliorer la gestion, la performance ou la disponibilité. Chaque partition se retrouve sur des serveurs ou des disques différents. Cela permet également d'obtenir une capacité de base de données supérieure à la taille maximum des disques durs ou d'effectuer des requêtes en parallèle sur plusieurs partitions, donc pour avoir les sous ensemble (les fragments) on a plusieurs type de fragmentation, l'un de c'est type est la fragmentation verticale qui est la plus compliqué parmi toutes les autre types.

Dans ce qui suit, nous présentons les types de fragmentation. Puis nous décrivons les principaux travaux effectués dans la fragmentation verticale vus que c'est le but domaine d'application de notre mémoire.

1.2 Définition de fragmentation

La fragmentation est le processus de décomposer des objets d'accès (tables, vues matérialisées, index) en un ensemble de partitions disjointes. Elle a été introduite à la fin des années 70 et au début des années 80 comme une technique de conception logique de bases de données traditionnelles, distribuées et parallèles. Avec le développement des entrepôts de données, la fragmentation est devenue une des structures d'optimisation la plus importante de la phase de conception physique. Actuellement, les éditeurs de bases de données commerciaux et académiques proposent des modes de partitionnement de tables

en utilisant des modes simples (Hash, List et Range) et des modes composés, toute combinaison deux à deux de modes simples (ex. Range-Range, Hash-List, etc.)[1]. La figure suivante montre l'évolution de la fragmentation dans les travaux de recherche menés par la communauté des bases de données.

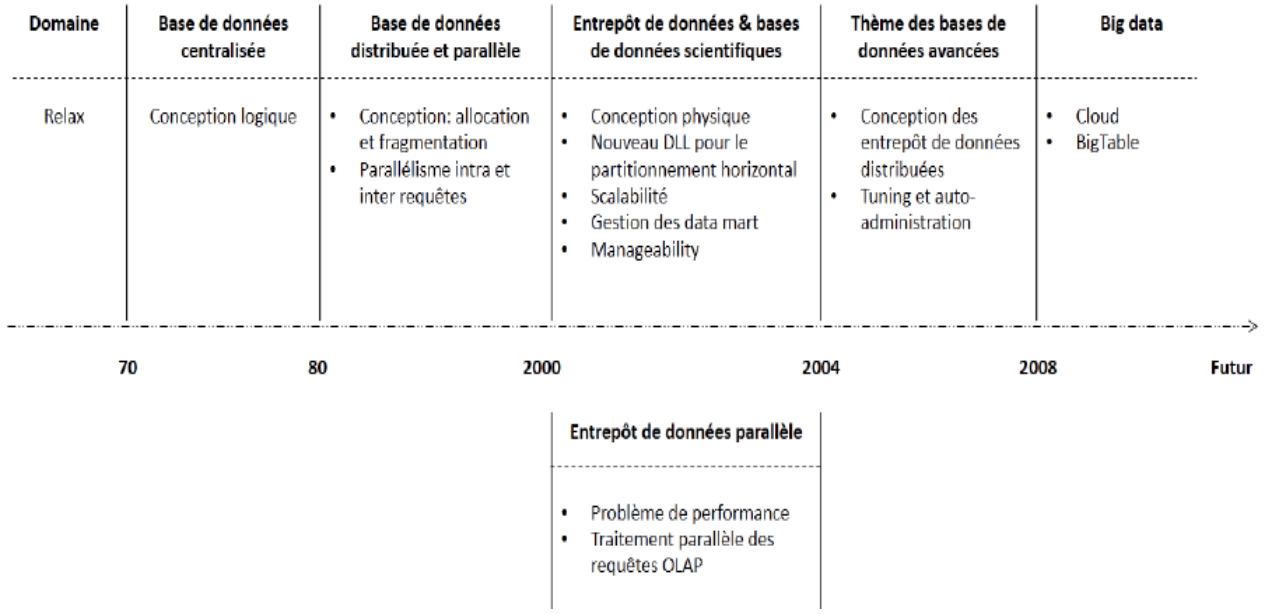

Figure 1 .1 Evolution de la fragmentation de données (1)

Comme toute méthode a des points positifs et des points négatifs donc aussi La fragmentation présente diverse avantages et en contre partie des inconvénients, on en cite les suivants :

L'efficacité du système de base de données est augmentée due à l'emplacement du stockage des données qui est près du site d'utilisation. De plus, les techniques d'optimisation de requêtes locales sont suffisantes pour la plupart des requêtes puisque les données sont disponibles localement. Étant donné la non-disponibilité des données non pertinentes sur les sites, la sécurité et la confidentialité du système de base de données peuvent être maintenues. En revanche ; les vitesses d'accès peuvent être très élevées si une nécessité de différents fragments s'impose. En cas de fragmentation récursive, le travail de reconstruction nécessitera des techniques coûteuses. Pour finir, Le manque de copies de sauvegarde de données dans différents sites peut rendre la base de données inefficace en cas d'échec d'un site.

Comme on a déjà dit que la fragmentation a défirent type chaque type donne des résultats déférente et utilise des techniques déférentes des c'est types sont la fragmentation verticale, la fragmentation horizontale et la fragmentation mixte.

1.2.1 Le partitionnement vertical

La fragmentation verticale consiste à diviser une relation R en sous relations appelées fragments verticaux résultant de l'application de l'opération de projection [1]. Elle favorise naturellement le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de la fragmentation, en limitant le nombre de fragments aux quels accéder. Mais elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments.

Exemple : Soit la table $Client\ (idClient\ Nom, Ville, Sexe)$ partitionnée comme suit

Clients1: $\prod_{\text{idClient,Sexe}}\ (Client)$

Clients2: $\prod_{\text{idClient,Nom,V ille}}\ (Client)$

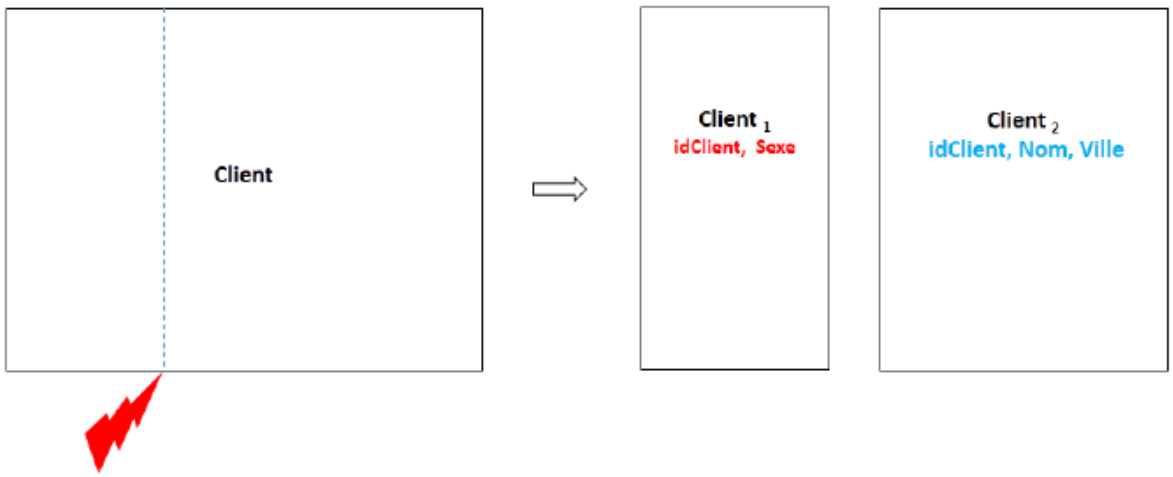

Figure 1.2 Exemple d'une fragmentation verticale (1)

1.2.2 Le partitionnement horizontal

La fragmentation horizontale consiste à diviser un objet ϑ $(table, index, vues)$ de la base de données en sous-ensembles de lignes appelés fragments horizontaux résultant de l'application de l'opération de restriction. La reconstruction de l'objet ϑ à partir de ces fragments horizontaux est obtenue par l'opération d'union de ces fragments. Il existe deux versions de la fragmentation horizontale. La fragmentation primaire s'effectue grâce à des prédicats de sélection définis sur la relation. Par contre, la fragmentation dérivée se fait avec des prédicats de sélection définis sur une autre relation. Concrètement, la fragmentation dérivée d'une table S n'est possible que lorsqu'elle est liée avec une table T par sa clé étrangère. Une fois la table T fragmentée par fragmentation primaire, les fragments de S sont générés par une opération de semi-jointure entre S et chaque fragment de la table T . Les deux tables seront équi_partitionnées grâce au lien père-fils. A partir de ces deux définitions, nous constatons que la fragmentation primaire pourrait accélérer les opérations de sélection tandis que la fragmentation dérivée accélérerait les opérations de jointure [1].

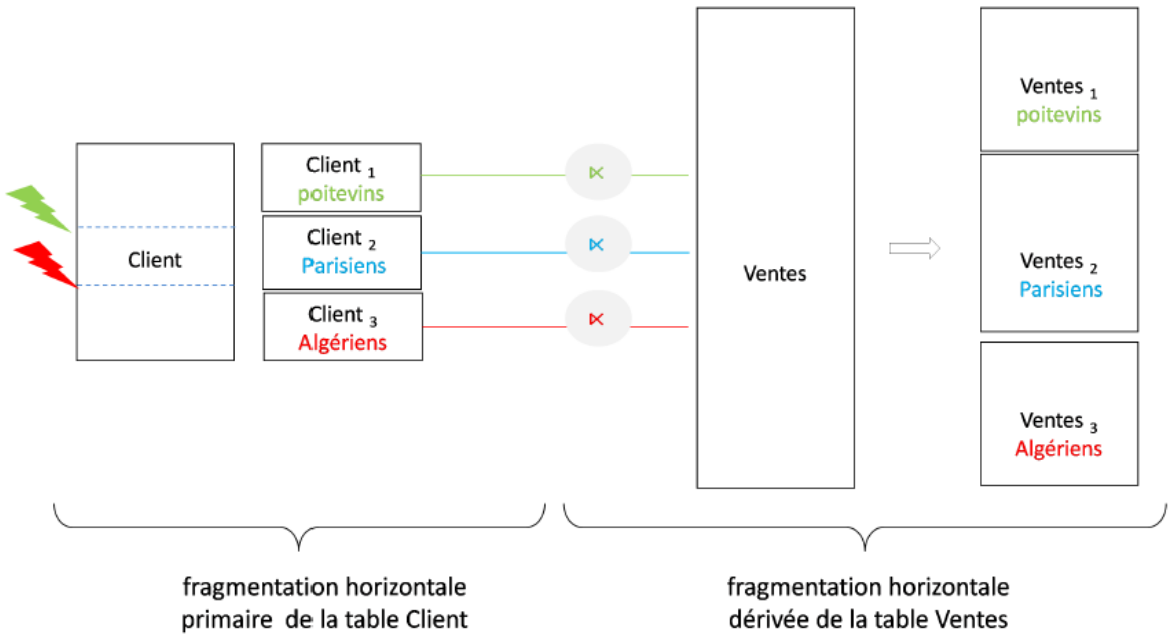

Figure 1.2 Exemple de la fragmentation horizontale (1)

Exemple : Soit un schéma d'une base de données constitué d'une table $Client$ et d'une table $Ventes$ Liée à la table $Client$ par une relation de clé étrangère. La table $Client$ est fragmentée en trois fragments,$Client1$,$Client2$ est $Client3$. Chaque fragment est défini par un prédicat de sélection sur l'attribut Ville de cette table:

$Clients_1$: $\sigma_{V\ ill\ =Poitierse}\ (Client)$

$Clients_2$: $\sigma_{V\ ille=Paris}\ (Client)$

$Clients_3$: $\sigma_{V\ ille=Alger}\ (Client)$

A partir du schéma de fragmentation de la table $Client$, la table $Ventes$ est fragmentée en trois fragments en fonction des trois fragments de la table$Client$. Chaque fragment de la table $Ventes$ est généré à l'aide d'une opération de semi-jointure ($\propto$) entre un fragment de la table $Client$ et la table des faits comme suit :

$Ventes_1 = Ventes \propto Clients_1$

$Ventes_2 = Ventes \propto Clients_2$

$Ventes_3 = Ventes \propto Clients_3$

1.2.3 Le partitionnement hybride

La fragmentation mixte résulte de la combinaison des deux sortes de fragmentation citées ci-dessus; pour atteindre les avantages de chaque type de fragmentation. Elle consiste à partitionner une relation en sous ensemble d'ensemble, cette dernière étant définie par la fragmentation verticale et les sous ensembles par la fragmentation horizontale [1].

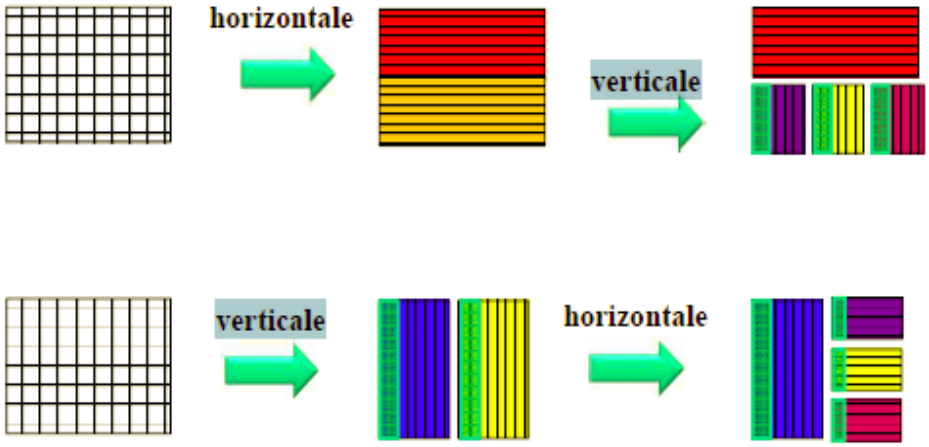

Figure 1.3 Exemple de la fragmentation hybride

1.3 Fragmentation verticale

Les fragments sont construits à partir de quelques attributs d'une relation [2]:

• Ce sont les attributs (avec leurs occurrences) d'une même relation qui peuvent être répartis dans des fragments différents.

• Toutes les valeurs des occurrences pour un même attribut se trouvent dans le même fragment.

• La répartition des attributs dans différents fragments ne peut être correcte que si l'identifiant (ou identité d'objet) est dupliqué dans chaque fragment :

1. Le découpage d'une relation en sous tables est effectuée utilisant la projection sur les colonnes composant chaque fragment.

2. La reconstruction est effectuée par jointure des différents fragments.

1.3.1 Problème de la fragmentation verticale

Le problème de fragmentation verticale est fondamentalement plus complexe que le problème de fragmentation horizontale, cela en raison de la taille de son espace de recherche [3].

En effet, une relation de m attributs peut être fragmentée en mm fragments. Le problème de la fragmentation verticale consiste à déterminer comment partitionner une relation en fragments, dans le but de maximiser la performance du système. La sélection des fragments optimaux d'une relation minimise le nombre des entrées/sorties et favorise le parallélisme [4].

1.3.2 Travaux existants

Travaux de Navathe et Ra

Les auteurs ont proposé un algorithme de partitionnement binaire[5] et une approche de partitionnement graphique[6]. L'approche est basée sur le concept d'affinité qui désigne la fréquence des requêtes. La méthode exploite des matrices spécifiques (matrice d'usage et matrice des affinités) pour regrouper les prédicats de sélection selon les fréquences des requêtes qui les utilisent. Un groupe est identifié par un cycle (ensemble de prédicats) et désigne un fragment de dimension.

L'algorithme d'affinité part d'une table T $\{PK,A_1,A_2,\ldots,A_N\}$ et d'un ensemble de requêtes $Q = \{Q_1, Q_2, \ldots, Q_L\}$ les plus fréquemment utilisées et leurs fréquences d'accès. L'ensemble des requêtes et de leurs fréquences d'accès est déterminé par l'administrateur.

La première approche suppose une relation ayant m attributs à fragmenter et un ensemble de n requêtes les plus fréquentes. Elle s'exécute en deux phases. Pendant la première phase, trois matrices sont construites.

1. **Une matrice d'usage des attributs** dans laquelle la valeur de l'élément de la ligne i et la colonne j a pour valeur 1 si l'attribut j est accédé par la requête i

(sinon cette valeur est nulle), est complétée par une colonne supplémentaire qui sauvegarde la fréquence d'accès pour chacune des requêtes.

2. **Une matrice d'affinités d'attributs** contient m lignes et m colonnes correspondant aux attributs de la relation à fragmenter. La valeur de la cellule (i, j) affiche les valeurs d'affinités définies entre les attributs Cette affinité correspond à la somme des fréquences d'accès des requêtes accédant simultanément aux deux attributs.

3. **Une matrice d'affinité ordonnée** est obtenue par l'application de l'algorithme de B.E.A (Bond Energy Algorithm) sur la matrice des affinités d'attributs. Par permutation des lignes et des colonnes, l'algorithme B.E.A regroupe les attributs qui sont utilisés simultanément en fournissant une matrice sous la forme d'un semi-bloc diagonal.

La deuxième phase consiste à effectuer un partitionnement binaire récursif de la matrice d'affinités ordonné à la recherche des fragments verticaux qui optimisent une fonction objectif.

Pour la seconde approche, Navathe et Al[6] ont présenté une méthode de regroupement des attributs basés sur les graphes. Cet algorithme construit à partir de la matrice des affinités d'attributs un graphe complet. Les nœuds de ce graphe représentent les attributs de la matrice d'affinités, et une arête, entre deux attributs, représenté la valeur d'affinité. Il cherche à générer des cycles dont les arêtes possèdent des grandes valeurs d'affinités.

Travaux de Gorla et Betty

Les auteurs exploitent les règles d'association pour la fragmentation verticale d'une base de données relationnelle[7]. Ils affirment que les performances du système peuvent être optimisées en réduisant le nombre d'accès aux données utiles pour une requête. Ils valident leur proposition avec deux bases de données réelles en faisant varier les seuils de support et de confiance.

Travaux de Bellatreche et Al

Les auteurs proposent une approche de fragmentation verticale pour partitionner une base de données objet distribuée [8]. Bellatreche et ses collègues proposent une approche topdown où l'entité de fragmentation est la classe. Ils présentent un algorithme de fragmentation verticale dans un modèle constitué par des attributs complexes et des méthodes complexes. Ce type de fragmentation facilite la décomposition de la requête, l'optimisation, et le traitement parallèle pour les systèmes de bases de données orientées objet distribués.

Travaux de Datta et al

Les auteurs ont proposé une nouvelle stratégie de traitement parallèle des requêtes de jointure [9]. Ce traitement parallèle est basé sur la fragmentation verticale. La fragmentation verticale a été utilisée comme une structure d'indexation. Chaque fragment représente une structure spéciale nommé Data Index (BDI). Pour accélérer les opérations de jointure, une extension du BDI nommée JDI est proposée. JDI est composé de BDI et une liste indiquant les enregistrements correspondants dans la table de dimension correspondante.

1.3.3 Techniques de fragmentation

- **Groupement**: Chaque attribut forme un fragment, jointure des fragments tant qu'un critère de performance est satisfait.

- **Splitting**: on débute par la relation globale, on génère ensuite des partitions en se basant sur le comportement des applications (requêtes).

Principe de l'algorithme

Un fragment vertical contient des attributs fréquemment utilisés. Les informations dont on aura besoin sont les suivantes :

- $Q = \{q_1, q_2, ..., q_m\}$: ensemble de requêtes fréquentes.
- $R\ (A_1, A_2, \ldots, A_n)$: relation R avec n attributs, et l sites.
- Matrice d'usage $m \times n$: U_{ij} : 1 si attribut A_j est référencé par q_i sinon0 .
- Matrice d'accès $m \times l$: acc_{ik}: fréquence d'accès de la requête i sur le site k .
- Matrice de référence $m \times l$: ref_{ik}: le nombre d'accès aux attributs pour chaque exécution de la requête q_i sur le site k .

Exemple : Soit la relation$PROJ(PNO, PNAME, BUDGET, LOC)$ .Sachant qu'on a 3 sites, 4 requêtes SQL :

q1: SELECT BUDGET FROM PROJ WHERE PNO = Val;

q2: SELECT PNAME, BUDGET FROM PROJ;

q3: SELECT PNAME FROM PROJ WHERE LOC = Val;

q4: SELECT SUM (BUDGET) FROM PROJ WHERE LOC=Val;

$$U = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{vmatrix} \qquad acc = \begin{vmatrix} 15 & 20 & 10 \\ 5 & 0 & 0 \\ 25 & 25 & 25 \\ 3 & 0 & 0 \end{vmatrix}$$

Après avoir la matrice d'usage ainsi que la matrice d'accès, on calcule alors Mesure d'affinité entre deux attributsA_i et A_j($\boldsymbol{aff_{ij}}$) d'une relation R qui est définie comme: $\boldsymbol{aff_{ij}} = \Sigma\{ k | Uk_i = 1 \wedge Uk_j = 1\} \Sigma l\ acclk$

$$U = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{vmatrix} \quad acc = \begin{vmatrix} 15 & 20 & 10 \\ 5 & 0 & 0 \\ 25 & 25 & 25 \\ 3 & 0 & 0 \end{vmatrix}$$

$$aff = \begin{vmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{vmatrix}$$

Matrice d'affinité et son rôle pour la FV

- La matrice d'affinité est utilisée pour guider la génération des fragments verticaux.
- les attributs ayant une grande affinité seront groupés pour former un fragment vertical.

1.4 Conclusion

Nous avons vu dans ce chapitre que veut dire la fragmentation d'une base de données. Elle perme d'optimiser les requêtes et de faciliter la gestion des données. Cette fragmentation a été déployée sur trois divers types, les 3 type de fragmentation on été étudiée et on a montre leur différente approches puis on a concentré sur la fragmentation verticale ce qui est notre thème de cette mémoire on a montré les problèmes de la fragmentation verticale ainsi que les travaille est les algorithmes qui sont déjà été étudier et approuvé par des certains chercheur, le chapitre suivant est consacré à l'étude de la technique CPSO pour une fragmentation optimale.

Chapitre 2 :

Particle Swarm Optimisation

Chapitre 3 : Particle Swarm Optimization

2.1 Introduction

Malgré le développement et la création d'ordinateurs qui sont de plus en plus performants mais y'a toujours des nombreux problèmes qui n'ont pas souvent des solutions déterministe en un temps raisonnable .Pour résoudre ce problème, on a opté pour des méthodes heuristiques, des méthodes qui ramène une solution approchée. Toutefois, il faut reproduire le processus sur plusieurs itérations pour tendre vers une solution acceptable.

Parmi ces heuristiques, certains algorithmes qui possèdent un principe générique adaptable et qui s'applique donc à plusieurs problèmes d'optimisation. Et parmi c'est méthode on trouve L'optimisation par essaim particulaire, qui dérive de la descente stochastique (commence solution initiale, on la compare à tous ses voisins en conservant à chaque fois le meilleur résultat). Elle s'inspire fortement des relations grégaires des oiseaux migrateurs qui doivent parcourir des longues distances et qui doivent donc optimiser leurs déplacements en termes d'énergie dépensée, dans se chapitre on va montre les technique de cette méthodes ainsi l'algorithme avec c'est paramètre.

2.2 Historique

L'optimisation par essaim particulaire (OEP) ou bien (PSO Particle swarm optimization), a été introduite par Kennedy et Eberhart (1995) [10]. Qu'on des racines dans la simulation des comportements sociaux à l'aide d'outils et d'idées tirés de la recherche en informatique et en psychologie sociale.Dans le domaine de l'infographie, les premiers antécédents de l'optimisation des essaims de particules remontent au travail de Reeves (1983), qui a proposé des systèmes de particules pour modéliser des objets dynamiques et ne pouvant être facilement représentés par des polygones ou des surfaces. Des exemples de ces objets sont le feu, la fumée, l'eau et les nuages.

Dans ces systèmes, les particules sont indépendantes l'une de l'autre et leurs mouvements sont régis par un ensemble de règles. Quelques années plus tard, Reynolds utilisait un système de particules pour simuler le comportement collectif d'un troupeau d'oiseaux. Dans une simulation similaire, Heppner et Grenander comprenaient un roost qui était attrayant pour les oiseaux simulés. Les deux modèles ont inspiré l'ensemble des règles qui ont été utilisées plus tard dans l'algorithme original d'optimisation des essaims de particules.

2.3 PSO pour la répartition des données

Les algorithmes à essaim de particules ont été utilisés pour réaliser différentes tâches. Dans le cadre de la sélection d'attributs, Agrafiotis et Cedeno en 2002 proposent un algorithme basé sur les essaims de particules pour l'étude de la relation quantitative entre la structure et l'activité de composant chimique.

Omran et al, aussi en 2002 utilisent les essaims de particules pour effectuer une classification d'images. Dans le cadre de l'extraction de règles de classification, les OEP ont été comparés aux algorithmes génétiques et à l'algorithme C4.5 et ont été appliqués à la génération de règles pour définir le profil des utilisateurs d'un site web et à l'extraction de règles à partir d'un réseau de neurones. Aussi, Xiao et al, en 2003 utilisent une méthode hybride basée sur les essaims de particules et l'algorithme de clustering "Self-Organizing Maps" pour réaliser un partitionnement des gènes. D'autres travaux ont porté sur l'extension de cette heuristique en vue d'élargir son champ d'adaptation.[12]

2.4 Les éléments de la P.S.O.

Si on veut appliquer la PSO il faut d'abord qu'on défini un espace de recherche constitué de particules et une fonction objective à optimiser. Le principe c'est de déplacer ces particules pour trouvent l'optimum.

Chacune de ces particules a :

1. Une position.

2. Une vitesse : permet la particule de se déplacer. (Au cours des Itérations, chaque particule change de position. Elle évolue en fonction de son meilleur voisin, de sa meilleure position, et de sa position précédente). C'est cette évolution qui permet de tomber sur une particule optimale.

3. Un voisinage : un ensemble de particules qui interagissent directement sur la particule, en particulier celle qui a le meilleur critère.

Chaque particule connait :

1. Sa meilleure position visitée.

2. La position du meilleur voisin de l'essaim qui correspond à l'ordonnancement optimal.

3. La valeur qu'elle donne à la fonction objective (à chaque itération il faut une comparaison entre la valeur du critère donnée par la particule courante et la valeur optimale).

Donc on conclu ce que Maurice Clerc et Patrick Siarry [11] on déjà dit, que l'évolution d'une particule est finalement une combinaison de trois types de comportements : égoïste (suivre sa voie suivant sa vitesse actuelle), conservateur (revenir en arrière en prenant en compte sa meilleure performance) et panurgien (suivre aveuglement le meilleur de tous en considérant sa performance).

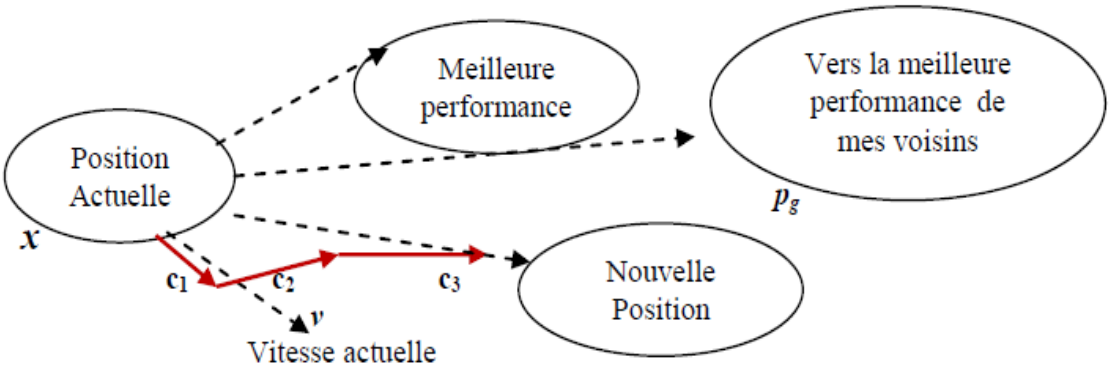

Figure 2.1 Schéma de principe du déplacement d'une particule. (12)

2.5 Principe fondamental

L'algorithme de base de la P.S.O. travaille sur une population appelée essaim de solutions possibles, elles-mêmes appelées particules, qui sont placées aléatoirement dans l'espace de recherche de la fonction objectif.

A chaque itération, les particules se déplacent en prenant en compte leur meilleure position (déplacement égoïste) mais aussi la meilleure position de son voisinage (déplacement panurgien).

Soit m la taille de l'essaim. Chaque particule i peut être représentée comme un objet avec plusieurs caractéristiques. Supposons que l'espace de recherche soit un espace n-dimensionnel, alors la $i^{\text{ème}}$ particule peut être représentée par un n-dimensionnelle vecteur, $X_i\{x_{i1}, x_{i2}, \dots, x_{in}\}$, et la vitesse, $V_i\{v_{i1}, v_{i2}, \dots, v_{in}\}$, ou i= 1,2, $\dots m$.

Dans PSO, la particule i se souvient de la meilleure position qu'elle a visitée jusqu'ici, appelé $P_i\{p_{i1}, p_{i2}, \dots, p_{in}\}$, et la meilleure position de la meilleure particule dans l'essaim, appelé $G_i\{G_1, G_2, \dots, G_n\}$.

PSO est similaire a un algorithme de calcul évolutif et dans chaque itération t le particule i ajuste sa vitesse sa vitesse v_{ij}^t et sa position x_{ij}^t pour chaque dimension $j$en lui référant.

Dans les faits, on calcule la nouvelle vitesse à partir de la formule suivante :

$V_{ij}^{t} = v_{ij}^{t-1} + C_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + C_2 r_2 (G_i^{t-1} - x_{ij}^{t-1})$), ……. (1)

est

$x_{ij}^{t} = x_{ij}^{t-1} - v_{ij}^{t}$, ……. (2)

$C_1, C_2,$ Sont des constant représente l'accélération et $r_1$est r_2Sont des nombres réels prend des valeurs entre$[0,1]$. La particule vole à travers des solutions potentielles vers p_i^t et G^t Tout en explorant de nouveaux domaines. Un tel mécanisme stochastique peut permettre d'échapper à l'optimum local. Comme il n'y avait aucun mécanisme réel pour contrôler la vitesse d'une particule, il fallait imposer une valeur maximale $V_{\max}$ et de n'a pas dépassait ce seuil, elle était Qui contrôle la distance de déplacement maximale à chaque itération, afin d'éviter qu'une particule dépasse de bonnes solutions. L'algorithme PSO se termine âpre un nombre un d'itérations maximum ou la meilleure position de la particule de l'ensemble de l'essaim ne peut être améliorée après se nombre.

Le problème mentionné ci-dessus a été abordé en incorporant un paramètre de poids dans la vitesse précédente, Eqs (2) et (3) Sont transformés comme suivants :

$v_{ij}^{t} = \chi(\mathrm{w} v_{ij}^{t-1} + C_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + C_2 r_2 (G_{ij}^{t-1} - x_{ij}^{t-1}))$, ……. (3)

$x_{ij}^{t} = x_{ij}^{t-1} - v_{ij}^{t}$, ……. (4)

w est appelé poids d'inertie et est utilisé pour contrôler l'impact de l'historique précédent des vitesses sur le courant .En conséquence, le paramètre w régule le compromis entre les capacités d'exploration globales et locales de l'essaim .Une valeur appropriée Pour le poids d'inertie x fournit généralement un équilibre entre les capacités d'exploration globales et locales et aboutit par conséquent à une réduction du nombre d'itérations requises pour localiser la solution optimale.χ est un facteur de constriction, qui sert à limiter la vitesse.

L'algorithme PSO a montré sa robustesse et son efficacité pour résoudre les problèmes d'optimisation de la valeur de la fonction dans les espaces numériques réels.

2.6 PSO pour optimisation combinatoire

Dans se mémoire, nous proposons une extension de l'algorithme PSO visant à résoudre le problème d'optimisation combinatoire. La PSO combinatoire diffère essentiellement de la PSO originale (ou continue) dans certaines caractéristiques.

1.6.1 Définition d'une particule

Note par $Y_i^t\{y_{i1}^t, y_{i2}^t, \dots y_{in}^t\}$ Le vecteur n-dimensionnel associé à la solution $X_i^t\{x_{i1}^t, x_{i2}^t, \dots x_{in}^t\}$ qui prend des valeurs entre $\{-1, 0, 1\}$ Selon l'état de solution de la i ème particule à l'itération$t$.

Y_i^t est une variable utilisée pour permettre la transition de l'état combinatoire à l'état continu et le contraire.

$$y_{ij}^t = \begin{cases} 1 & if\ x_{ij}^t = G_j^t \\ 1 & if\ x_{ij}^t = P_{ij}^t \\ -1 \ \ ou\ bien \ \ 1\ aléatoirement & si\ \left(x_{ij}^t = G_j^t = P_{ij}^t\right), \\ 0 & si\ non\ . \end{cases} \quad , \dots\dots\dots\dots(5)$$

1.6.2 Vitesse (velocity)

On a $d_1 = -1 - y_{ij}^{t-1}$ es la distance entre x_{ij}^{t-1} et la meilleure solution obtenue par la i ème particule.

On a $d_2 = 1 - y_{ij}^{t-1}$ es la distance entre la solution actuelle x_{ij}^{t-1} et la meilleure solution obtenue dans l'essaim .Alors la nouvelle équation de la vitesse dans le CPSO est :

$$v_{ij}^t = w * v_{ij}^{t-1} + C_1 r_1 d_1 + C_2 r_2 d_2, \dots\dots\dots (6)$$

$v_{ij}^{t} = \mathrm{w}v_{ij}^{t-1} + C_1 r_1(-1 - y_{ij}^{t-1}) + C_2 r_2(1 - y_{ij}^{t-1})$, ……. (7)

Avec cette fonction, le changement de la vitesse v_{ij}^{t} dépend du résultat dey_{ij}^{t-1}.

Si $x_{ij}^{t-1} = G_{ij}^{t-1}$ alors y_{ij}^{t-1}=1. Donc d_2 se tourne vers '2' , et d_1 prend la valeur '0' , Ce qui impose que la vitesse changer au sens positif.

Dans le cas ou $x_{ij}^{t-1} \neq G_{ij}^{t-1}$ et ou $x_{ij}^{t-1} \neq p_{ij}^{t-1}$,alors y_{ij}^{t-1} se tourne vers '0' , $d_2 = 1$ et $d_1 = -1$, donc les paramètre r_1 , r_2 , c_1 et c_2 vont déterminé le sens du changement de la vitesse .

Dans le cas ou $x_{ij}^{t-1} = G_{ij}^{t-1}$ et $x_{ij}^{t-1} = p_{ij}^{t-1}$, alors y_{ij}^{t-1} prend une valeur entre$\{-1 , 1\}$, Ce qui impose que la vitesse changer au sens du signe de y_{ij}^{t} .

1.6.3 Construction d'une solution de particules

Le changement du résultat est calcule avec y_{ij}^{t} :

$\lambda_{ij}^{t} = y_{ij}^{t-1} + v_{ij}^{t}$, …….(8)

La valeur de y_{ij}^{t} est ajusté selon la fonction suivante :

$$y_{ij}^{t} = \begin{cases} 1 & if\ \lambda_{ij}^{t} > \alpha \\ -1 & if \lambda_{ij}^{t} < -\alpha \\ 0 & si\ non \end{cases} \text{ ,……. (9)}$$

La nouvelle solution est :

$$\begin{cases} G_j^{t-1} & if\ y_{ij}^{t} = 1 \\ P_{ij}^{t-1} & if y_{ij}^{t} = -1 \\ a\ nombre\ aleatoire & si\ non\ . \end{cases} \text{ , ……. (10)}$$

Le choix précédent atteint pour l'affection d'une valeur aléatoire dans $\{-1,1\}$ pour y_{ij}^{t-1} dans le cas de l'agilité entrep_{ij}^{t-1} et G_{ij}^{t-1} Permet d'assurer que la variable y_{ij}^{t} prend une valeur0, Et pour permettre une variation de la valeur du x_{ij}^{t} Nous définissons un paramètre αpour l'intensification et la diversification des raccords. Pour une petite valeur deα, x_{ij}^{t} prend une de deux valeurs p_{ij}^{t-1} etG_{ij}^{t-1} .

Dans le cas contraire, Nous imposons à l'algorithme d'attribuer une valeur nulle à y_{ij}^{t-1}, Entraînant ainsi à x_{ij}^{t-1} une valeur différente de p_{ij}^{t-1} et G_{ij}^{t-1} . c_1, c_2 Deux paramètres liés à l'importance des solutions p_{ij}^{t-1} et G_{ij}^{t-1} pour generé une nouvelle solution X_i^t , ils ont aussi un rôle dans l'intensification de la recherche.

2.7 Fonctions objectives pour la partition verticale

Les deux types de fonctions objectives utilisées pour les algorithmes de partition sont : Fonctions de modèle de coûts basées sur l'analyse d'accès aux transactions et basés sur une hypothèse empirique. La forme antérieure de la fonction objective est spécifique au SGBD sous-jacent tandis que la dernière est plus générale et plus intuitive.

Dans cette mémoire, nous utilisons une fonction d'objectif empirique, une version modifiée de l'évaluateur de partition proposé par Chakravarthy et al. Cet évaluateur de partition utilise le critère d'erreur carrée couramment appliqué aux stratégies de regroupement. Nous nommons donc l'évaluateur de partage de la partition carrée (SEPE).

Le SEPE se compose de deux facteurs de coût majeurs: Le coût d'accès aux attributs locaux non pertinent et Le coût d'accès aux attributs distants pertinents. Ils représentent le coût supplémentaire requis, Autre que le coût minimum idéal. En outre, le coût idéal est le coût lorsque les transactions n'accèdent que les attributs dans un seul fragment et n'ont aucun cas d'attributs non pertinents dans ce fragment. Les deux coûts sont calculés à l'aide du résultat d'erreur carrée, noté par E_M^2et E_R^2 respectivement.

2.8 Conclusion

L’optimisation est une branche des mathématiques qui permet de résoudre des problèmes en déterminant le meilleur élément d’un ensemble selon certains critères prédéfinis. PSO est une méthode d'optimisation relativement nouvelle, moderne et puissante qui a été démontrée empiriquement pour bien performer sur plusieurs de ces problèmes d'optimisation.

Il est largement utilisé pour trouver la solution globale optimale dans un espace de recherche complexe, notre but c’est d’adapté cette technique sur la fragmentation verticale.

Chapitre 3 :

Implémentation

Chapitre 3 :L'implémentation

1.1 Introduction

Dans les deux chapitres précédents on a vu que veut dire une fragmentation verticale ainsi que ses problèmes et les travaux qui ont déjà été effectués. Ensuite on a parlé dans le deuxième chapitre de la technique PSO pour l'optimisation, notre but dans ce mémoire, c'est d'adapter l'algorithme PSO pour la fragmentation verticale. Dans ce chapitre, on aborde le volet expérimental et la description des outils nécessaires, cela est fait via un exemple, et enfin une discussion sera détaillée pour le résultat obtenu.

1.2 Algorithme implémenté

Implémentation de l'algorithme a été réalisée avec le langage JAVA sous l'environnement NetBeans IDE 8.2. Des modifications ont été appliquées sur notre code afin de l'adapterà notre domaine d'application.Nous illustrons dans ce qui suit l'utilisation d'une stratégie de fragmentation basée sur l'algorithme PSO. [13]

Algorithme : PSO

- Entrée: matrice d'affinité
- Sortie: un ensemble de fragments optimaux

1. Etape1: Définissez la dimension des particules comme égale au nombre des attribué de la matrice.
2. Etape2 : initialisé la position de particule X_i et la vitesse v_i aléatoirement, et le nombre maximum d'itération.
3. Etape3 : pour chaque particule on calcule sa valeur de fitness

4. Etape4 :si la valeur de fitness et supérieure que Précédent p_{best} , Définissez la valeur de fitness actuelle comme le nouveaup_{best}.

5. Etape5 : âpre l'étape 3 et 4 pour toutes les particules, Sélectionner la meilleure particule commeg_{best}.

6. Etape6: pour toutes particule calculé la vitesse, et Mettre a jour sa position.

7. Etape7 : si le Critères d'arrêt ou bien le nombre maximum d'itération n'est pas satisfait on revient à Etape 3.

1.3 Environnent expérimentale :

Pour créer une nouvelle solution, nous utilisons Eq(5). Nous obtenons la valeur du vecteur y_{ij}^{t-1} le vecteur de la vitesse v_i^tCalculé avecEq(6). La nouvelle valeur de λ_{ij}^t et calculé avec λ_{ij}^t=y_{ij}^{t-1}+v_{ij}^t . y_{ij}^tet déterminé avec Eq(9) et utilisons Eq(10) pour la nouvelle solution du vecteurX_{ij}^t.

Dans notre programme les paramètres étaient comme suite :$c_1 = 0,3$ et $c_2 = 1,2$,r_1 et r_2 prend un intervalle entre $[-1, 1]$, le nombre d'itération fixé a 50.

L'exemple pris été une matrice de 10 attributs et 8 transactions,

T/A	1	2	3	4	5	6	7	8	9	10
$T1$	25	0	0	0	25	0	25	0	0	0
$T2$	0	50	50	0	0	0	0	50	50	0
$T3$	0	0	0	25	0	25	0	0	0	25
$T4$	0	35	0	0	0	0	35	35	0	0
$T5$	25	25	25	0	25	0	25	25	25	0
$T6$	25	0	0	0	25	0	0	0	0	0
$T7$	0	0	25	0	0	0	0	0	25	0
$T8$	0	0	15	15	0	15	0	0	15	15

Tableau 1 matrice d'attributs

Le résultat été sous forme d'un vecteur X_i^t , avec $k = 3$ fragments et

	X_i^t	X_i^t	X_i^t	X_i^t	X_i^t	X_i^t	X_i^t	X_i^t	X_i^t	X_i^t
X_i^t	1	2	2	3	1	3	1	2	2	3

Le vecteur nous explique que âpre 50 itération on a obtenir la solution optimale de la fragmentation avec 3 fragment $\{1 , 5 , 7\}, \{2 , 3,8,9\}, \{4,6,10\}$.

Les figure suivent montre le travaille de notre application

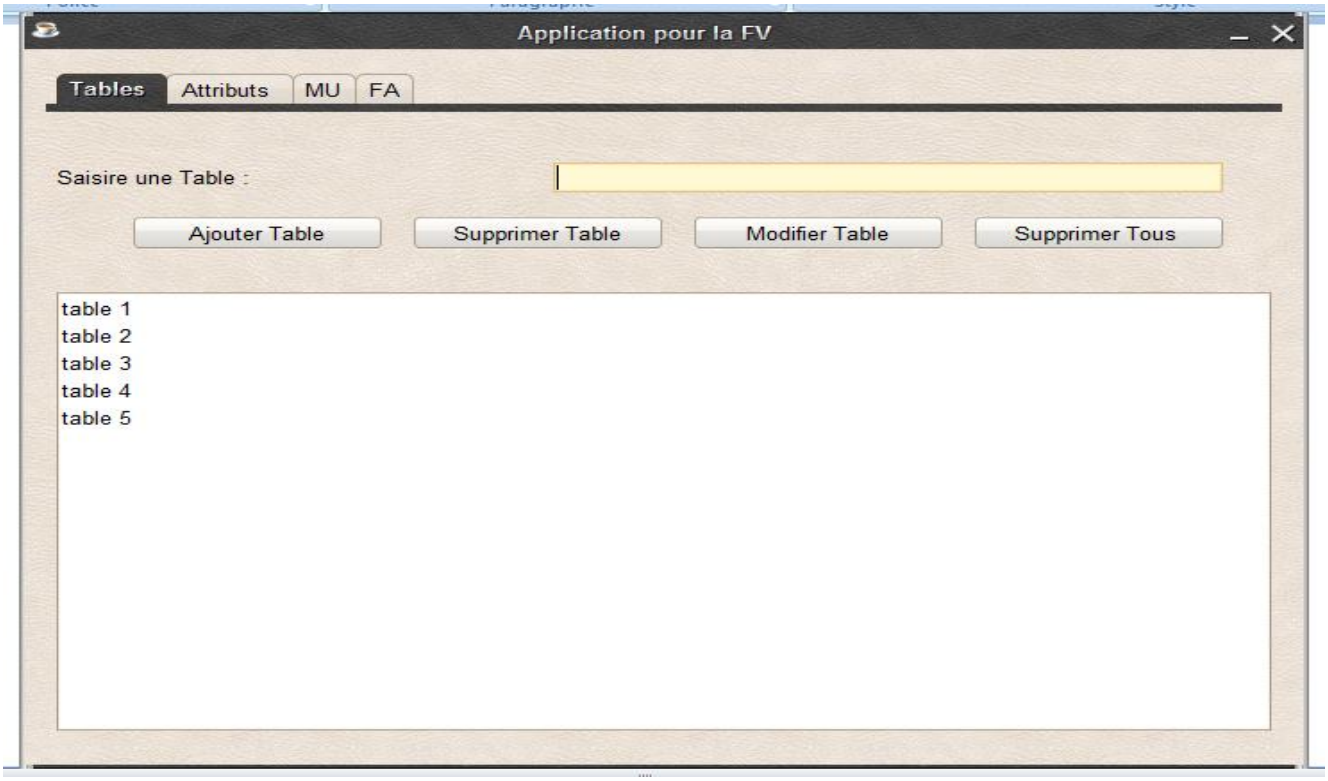

Figure 3.1 l'ajout des table dans l'application

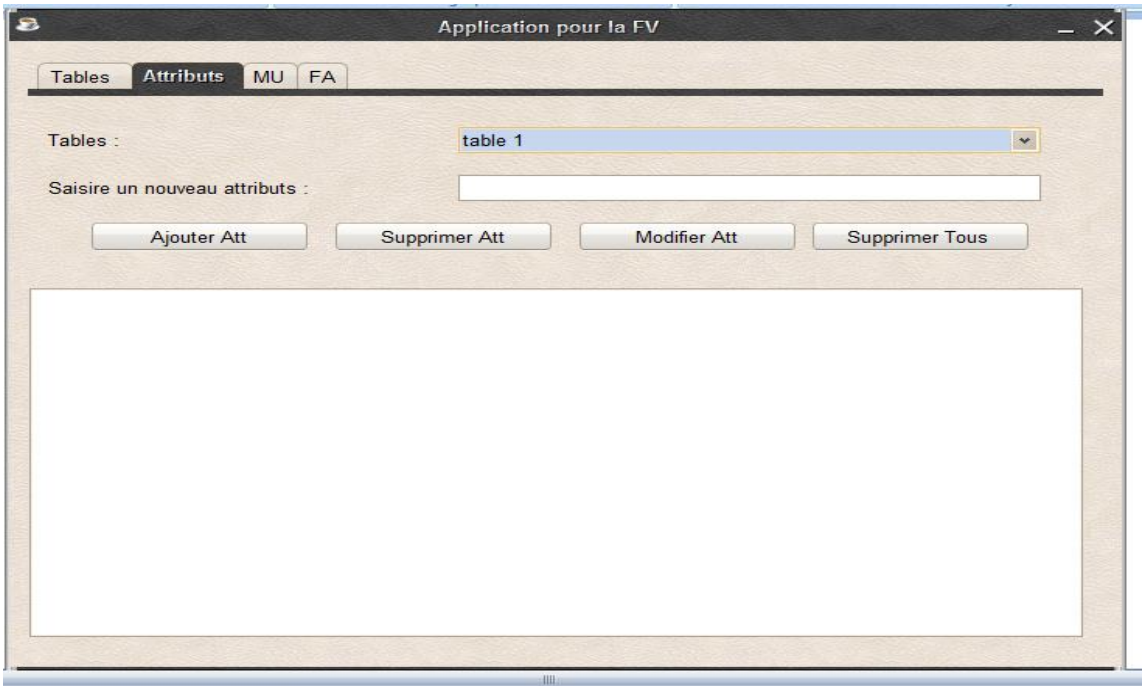

Figure 3.2 l'ajout des attributs dans l'application

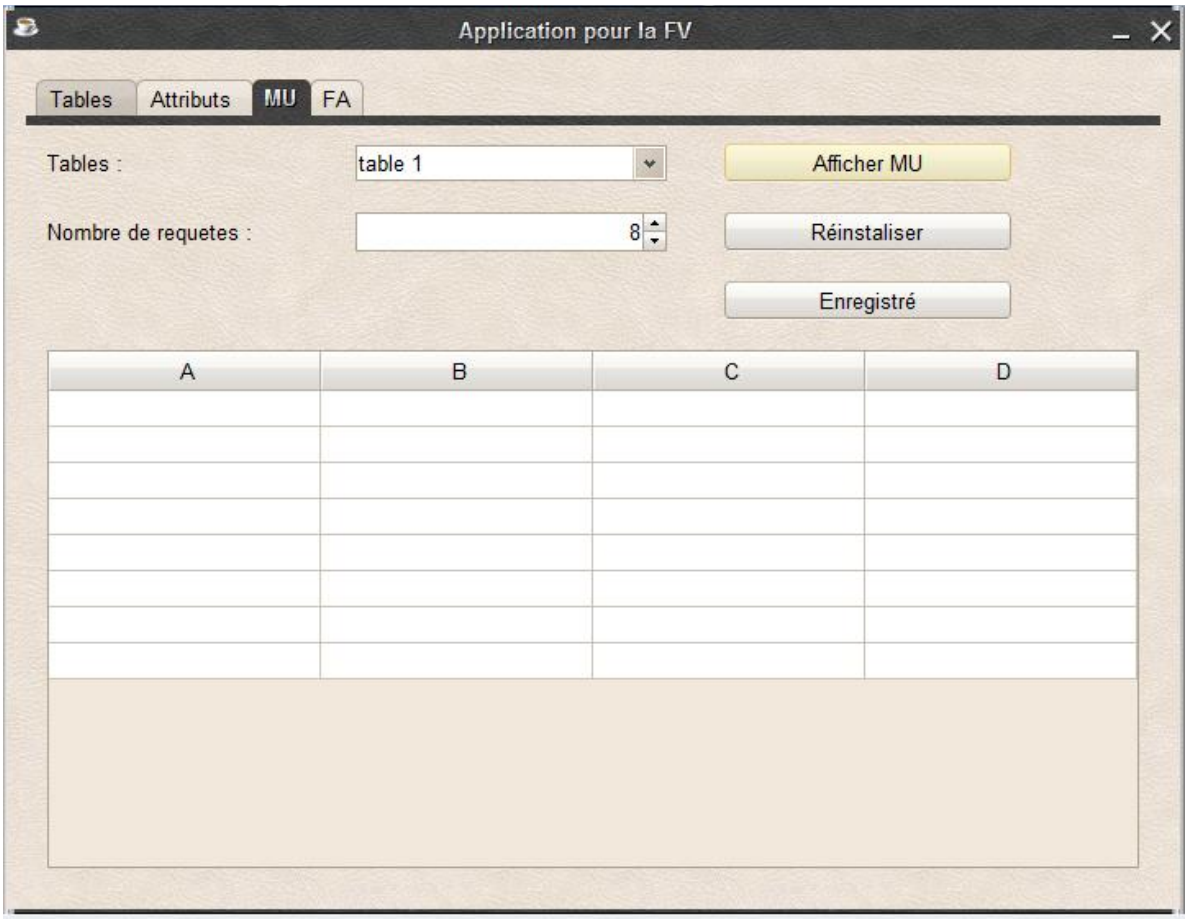

Figure 3.3 la matrice d'usage

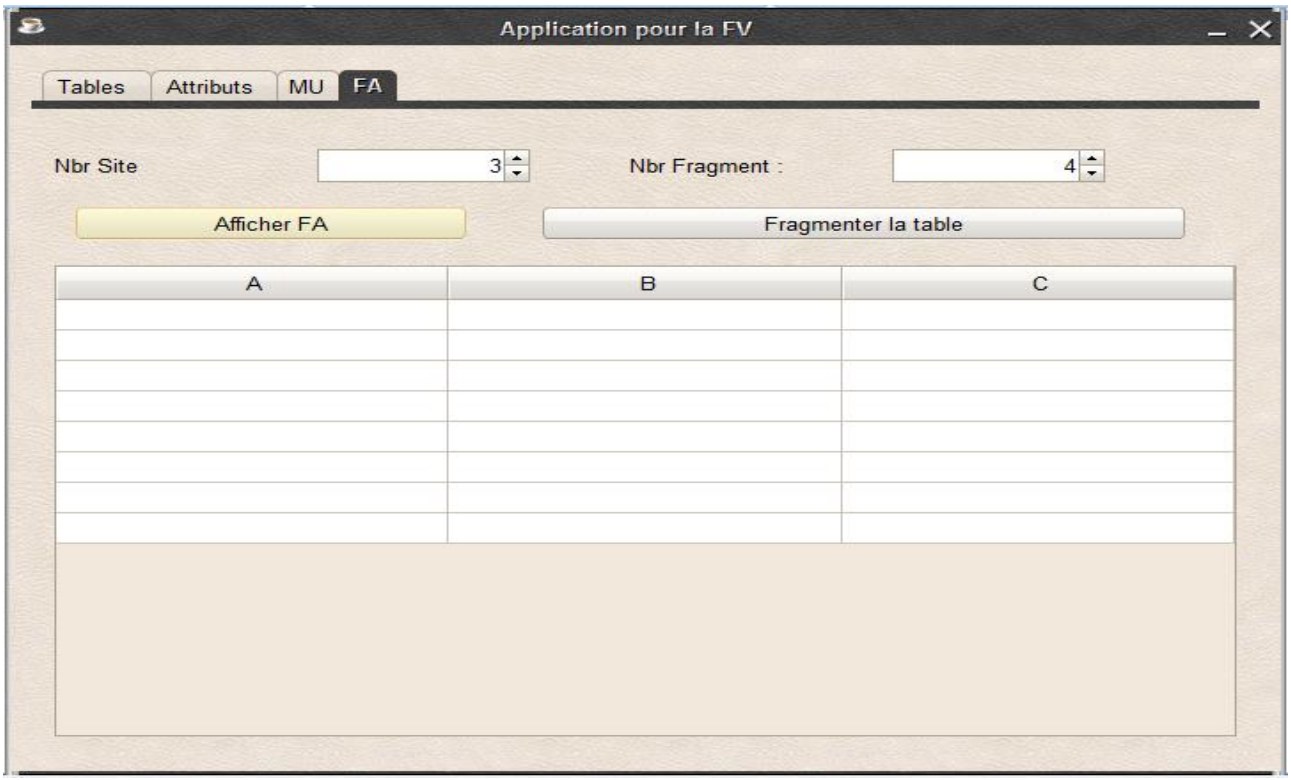

Figure 3.4 la matrice de fréquence d'accès

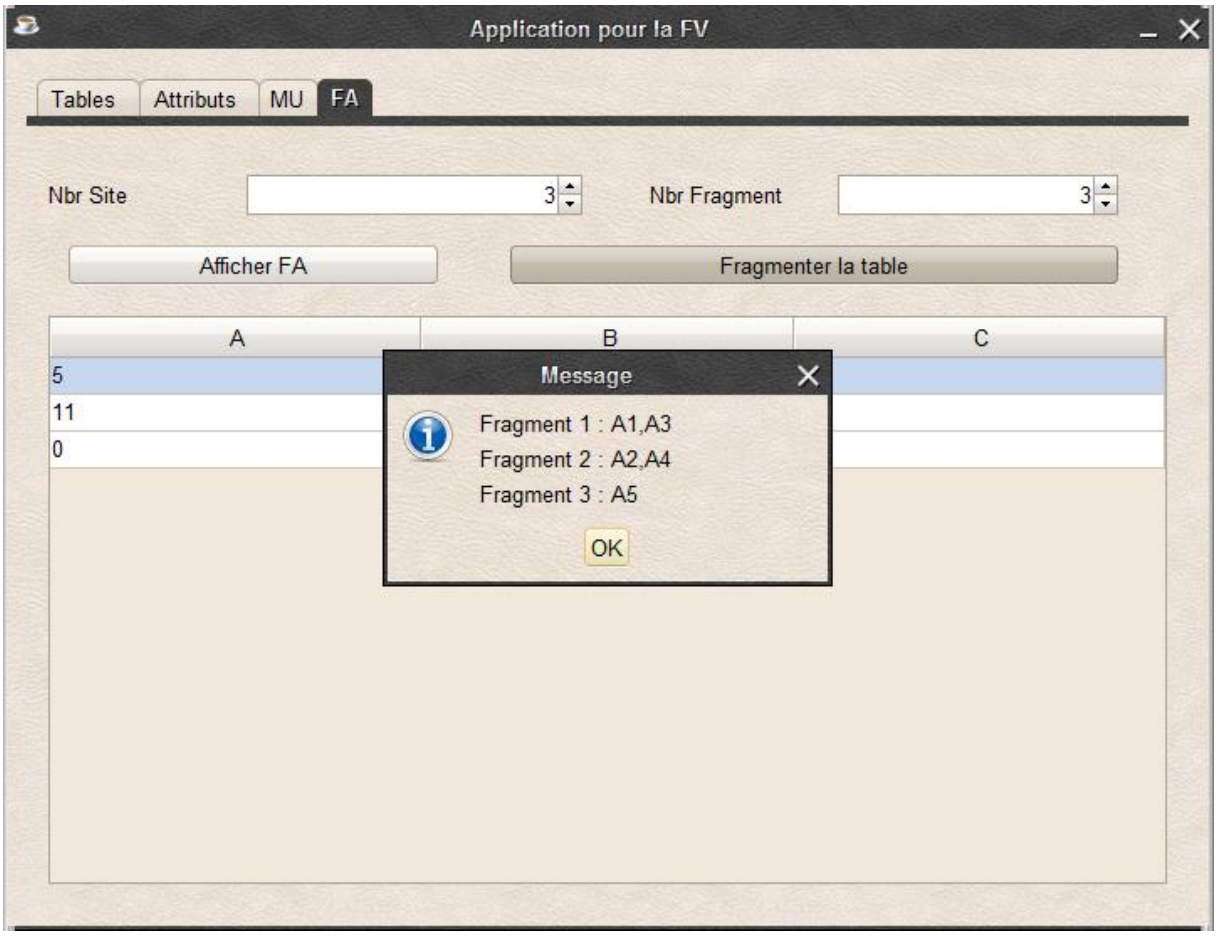

Figure 3 .5 le resultat

1.4 Conclusion

L'optimisation est fait pour résoudre des problèmes en déterminant le meilleur élément d'un ensemble, le PSO a été prouvé comme un outille efficace de l'optimisation pas les résultats qu'on trouvé.

Les résultats trouvé on respecté les 3 règles La complétude : pour toute donnée d'une relation R, il existe un fragment $R_i$de la relation R qui possède cette donnée, La reconstruction : pour toute relation décomposée en un ensemble de fragmentR_i, il existe une opération de reconstruction. Pour les fragmentations vertical, l'opération de reconstruction est l'une union. Pour les fragmentations verticales c'est la jointure el la Disjonction: assure que les fragments d'une relation sont disjoints deux à deux de la on peut dire que cette approche réussi.

Conclusion Générale et perspective

Conclusion et perspective

La partition verticale est une technique importante dans la conception de base de données utilisée pour améliorer les performances dans les systèmes de base de données. La fragmentation verticale est un problème d'optimisation combinatoire qui est NP-Hard dans la plupart des cas.

L'objectif que nous avons visé lors de ce travail été proposons une application et une d'un algorithme d'optimisation amélioré d'essaims de particules (PSO) pour le problème de la fragmentation verticale.

Ce choix justifie l'efficacité de cette nouvelle approche qui est considéré comme une des meilleures méthodes dans l'optimisation

Pour ce faire, nous avons commencé par l'introduire le PSO qui l'algorithme de base qui Souffre d'un inconvénient de la viens le CPSO pour réglé c'est inconvénient

L'efficacité l'algorithme PSO a été illustrées est prouvé par un exemple.

Ce travail nous a permis de se familiariser avec le langage JAVA et d'approfondir nos connaissances dans le domaine de l'optimisation en particulier la fragmentation et d'acquérir des connaissances sur l'algorithme PSO.

La réussite du CPSO pour résoudre un problème de partition verticale nous permet de suggère qu'il peut être utilisé pour résoudre d'autres problèmes de regroupement ou de regroupement.

Bibliographie

1. **BENKRID, Soumia.** *Le déploiement, une phase à part entière dans le cycle de vie des entrepôts de données.* 24/06/2014. 1.

2. **Sabri, Abdelouahed.** *Base de données reparties.* 2011/2012. 2.

3. **Valduriez., M. T. Özsu and P.** *Principles of Distributed Database Systems,Third Edition.* 2011. 3.

4. Barker., C. I. Ezeife and K.*Vertical fragmentation for advanced object models in a distributed.* Conf. on Computing and Information. 1995.

5. **S. Navathe, S. Ceri, G. Wiederhold, and J. Dou.** *Vertical partitioning algorithms for database design. ACM Transaction on Database Systems.* 1984. 5.

6. *Vertical partitioning for database design : a graphical algorithm.* **Ra., S. Navathe and M.** 1989. International Conference on Management of Data. pp. 440–450,. 6.

7. **Yan., N. Gorla and B. P. W.** *Vertical fragmentation in databases using data-mining technique. In Database Technologies.* 2009. pages 2543–2563.. 7.

8. **L. Bellatreche, A. Simonet, and M. Simonet.** *An algorithm for vertical fragmentation in distributed object database systems with complex attributes and methods.* September 1996.

9. *A case for parallelism in data warehousing and olap.* **A. Datta, B. Moon, et H. Thomas.** August 1998. in the 9th International Workshop on Database and Expert Systems Applications. pp. pages 226–231,. 9.

10. **SENE, Maxime BOMBRUN Abdoulaye.** *L'optimisation par essaim particulaire pour des problèmes d'ordonnancement.* 24/03/2011. 10.

11. **SIARRY, CLERC et.** *Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires,.* 2003. 11.

12. **Hacène Derrar*, Omar Boussaid**, Mohamed Ahmed-Nacer.** approche de répartition des données d'un entrepôt basée sur l'Optimisation par Essaim Particulaire. 12.

13. **BENSLIMANE, MOHAMED EL AMINE CHERGUI et SIDI MOHAMED.** Using Combinatorial Particle Swarm Optimization to Automatic ServiceIdentification. 2013, 13.