N°d'ordre : 04/2015-D/INF

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE
FACULTE ELECTRONIQUE ET INFORMATIQUE

THESE

Présentée pour l'obtention du grade de DOCTEUR EN SCIENCES
En : INFORMATIQUE
Spécialité : Intelligence artificielle et bases de données avancées

Par : Hacene DERRAR

Sujet

OPTIMISATION DES PERFORMANCES DANS LES ENTREPOTS DE DONNEES : TRAITEMENT DES PROBLEMES DE CONCEPTION DES SCHEMAS DE FRAGMENTATION

Soutenue publiquement le 18 avril 2015 devant le jury composé de :

Mme Zaia ALIMAZIGHI	Professeur à l'USTHB	**Présidente**
M. Mohamed AHMED-NACER	Professeur à l'USTHB	**Directeur de thèse**
M.Omar BOUSSAID	Professeur, Université Lumière, Lyon2	**Co-Directeur de thèse**
M. Djamel Eddine ZEGOUR	Professeur à l'ESI	**Examinateur**
Mme Nadjia BENBLIDIA	Professeur à l'université de Blida	**Examinateur**
M. Kamel BOUKHELFA	Maitre de conférence à l'USTHB	**Examinateur**

Résumé : L'évolution du modèle et de la charge de travail, causées particulièrement par un volume de données sans cesse croissant et aux caractéristiques des requêtes OLAP, entravent l'adaptation des techniques de fragmentation aux entrepôts de données. Dans cette thèse, nous avons traité trois principales problématiques. Notre première proposition traite le problème de la NP-complétude de la conception d'un schéma de fragmentation. Dans ce contexte, nous avons proposé l'utilisation d'une métaheuristique, en l'occurrence l'Optimisation par Essaim Particulaire (OEP), pour déterminer une stratégie de fragmentation optimale. Notre deuxième contribution présente une approche d'évaluation, qui mesure la pertinence du schéma de fragmentation implémenté sur la base des fréquences des accès aux données des différents fragments. Nous avons utilisé le calcul de l'erreur quadratique, une fonction d'estimation statistique, pour évaluer les fragments selon la pertinence de leurs attributs. Et enfin, notre troisième proposition traite le problème de l'impact de l'évolution de la charge de travail sur le schéma de fragmentation déjà implémenté. Nous avons proposé, à cet effet, une approche de fragmentation dynamique des données. Pour pouvoir intégrer l'aspect « dynamicité » nous avons utilisé les histogrammes pour la sauvegarde de l'historique des accès afin d'estimer la sélectivité des données des différents fragments.

Abstract : The evolution of the model and the workload, particularly due to the growing of data volume and characteristics of OLAP queries, impairs the application of fragmentation techniques in data warehouses. In this thesis we addressed three main issues. Our first proposal addresses the problem of NP- completeness of the design of a fragmentation schema. In this context, we proposed the use of a metaheuristic, namely Particle Swarm Optimization (PSO) to determine a strategy of optimal fragmentation. Our second contribution proposes a formal approach which measures the relevance of an implemented fragmentation schema on the basis of data access frequencies to the different fragments. We used the calculation of the squared error, a statistical estimation function, to evaluate the relevance of the fragments according to the relevance of their attributes. And finally, our third proposal addresses the problem of the impact of workload changes on the fragmentation schema already implemented. To achieve this problem, we propose a dynamic data fragmentation approach. To integrate the "dynamicity" aspect we used histograms to backup the access history in order to estimate the selectivity of data fragments.

Dédicaces

A mes parents ;

A ma femme ;

A mes enfants ;

A toute ma famille ;

A tous ceux qui me sont chers .

Remerciements

Je remercie tout d'abord mes directeurs de thèse M. le Professeur Mohamed AHMED NACER et M. le Professeur Omar BOUSSAID pour leur soutien tout au long de ce travail. Je suis heureux de pouvoir leur exprimer mes plus vifs remerciements et ma très sincère reconnaissance.

Je remercie Mme Zaia ALIMAZIGHI, Professeur à l'Université des Sciences et de la Technologie Houari Boumediene (USTHB), de m'avoir fait l'honneur de présider ce jury.

Je tiens à remercier également M. Djamel Eddine ZEGOUR, Maître de Conférences – Classe -A- à l'Ecole Supérieure de l'Informatique (ESI), Mme Nadjia BENBLIDIA, Maître de Conférences Classe -A- à l'université de Blida et M. Kamel BOUKHELA, Maître de Conférences Classe -A- à l'Université des Sciences et de la Technologie Houari Boumediene (USTHB), qui m'ont fait l'honneur d'avoir accepté de faire partie du jury.

Je souhaite également adresser mes remerciements à tous les amis et collègues que j'ai côtoyés durant mes séjours au Laboratoire ERIC de l'Université Lumière Lyon 2, notamment, Merouane, Cécile, Nora, Hadj, Emna, Elie, avec qui j'ai partagé des moments de travail inoubliable. Je leurs souhaite beaucoup de réussite.

Mes remerciements vont naturellement aux membres de l'équipe BDD, Mme Fadila BENTAYEB, Mme Nouria HARBI, Mme Sabine LOUDCHER RABASEDA et Jérôme DARMONT. J'ai eu un immense plaisir de travailler avec eux.

Je remercie mes amis et collègues du Bureau Informatique. Je leur exprime ma profonde sympathie et je leur souhaite une bonne continuation.

Je remercie chaleureusement mon père, ma mère et ma femme qui n'ont cessé de m'encourager et de me soutenir tout ou long de ce travail.

Finalement, je tiens à remercier ceux avec qui j'ai collaboré à un moment ou un autre, et ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Table des matières

PARTIE II : Contributions

PARTIE III : Expérimentation et conclusion

Liste des figures

Liste des tableaux

Glossaire

DVF	Domaine de la Valeur du Fragment
FH	Fragmentation horizontale
FHD	Fragmentation horizontale dérivée
FHP	Fragmentation horizontale primaire
FV	Fragmentation verticale
FM	Fragmentation mixte
MAA	Matrice d'Affinité des Attributs
MOLAP	Multidimensional On-Line Analytical Processing
MUA	Matrice d'Usage des Attributs
OLAP	On-Line Analytical Processing
HOLAP	Hybrid On-Line Analytical Processing
OEP	Optimisation par Essaim de Particules
OEP-FV	Notre approche de fragmentation verticale basée sur l'utilisation de l'OEP
OEP-FHD	Notre approche de fragmentation horizontale dérivée basée sur l'utilisation de l'OEP
OEP-FHP	Notre approche de fragmentation horizontale primaire basée sur l'utilisation de l'OEP
PSV	Problème de sélection des vues matérialisées
PMV	Problème de la maintenance des vues
RANGE	Mode de fragmentation horizontale permettant de partitionner une table selon des intervalles de valeurs
ROLAP	Relational On-Line Analytical Processing
SF	Schéma de fragmentation

Chapitre 1
Introduction général

1.1 Contexte

L'émergence d'une concurrence ardue, accentuée par de nouveaux enjeux économiques, a obligé les entreprises d'être plus compétitives et plus réactives. Pour faire face à ces nouvelles exigences, les entreprises doivent privilégier l'anticipation en disposant de solutions et systèmes permettant une manipulation et un accès quasi instantané aux informations dont elles disposent. Ces informations, continues dans des données gérées par des systèmes opérationnels et/ou acquises à partir de sources externes, constituent un réservoir de connaissance qui doit être analysé et exploré à des fins d'aide à la décision.

Cependant, les données sont généralement surabondantes, non organisées et éparpillées dans de multiples systèmes hétérogènes. Il devient nécessaire, pour une perspective décisionnelle, de les rassembler, les intégrer et les homogénéiser en une forme cohérente. Pour répondre à ce besoin, l'entreposage de données ou son équivalent anglais le "Data warehousing", a été la réponse parfaite. L'objectif de ce « concept », est de collecter et regrouper, en un seul annuaire, des données décrites de manière multidimensionnelle afin de les mettre à la disposition des décideurs à des fins d'analyse. Cette analyse fait appel à des traitements OLAP (On-Line Analytical Processing), qui se distinguent des processus OLTP (On-Line Transactional Processing) par de nouvelles perspectives offertes aux utilisateurs en matière d'exécution des requêtes complexes et génération des rapports à travers une interrogation intégrative des données.

Toutefois, malgré l'engouement, sans cesse croissant, par de nombreuses entreprises de tous secteurs confondus : santé, services, transport, industrie,.., pour la mise en œuvre de solutions d'analyse et d'exploration des données, l'entreposage des données demeure confronter à un problème majeur, celui de la gestion et l'interrogation efficace de gros volumes de données.

En effet, ce volume croît à des cadences énormes, au point atteignent le téra-octet (10^{12} octets), voire le peta-octet (10^{15} octets) pour les entrepôts données relationnels. Une enquête de Gartner Group a révélé que «les entrepôts de données vont subir une croissance moyenne annuelle de 33% en ce qui concerne leur volume de données, voire même 50% minimum lorsqu'ils sont confrontés à de fortes exigences en matière de données clients, de chaîne logistique, d'e-commerce ou de gestion de conformité des données ».

Cette croissance a été également accentuée par l'émergence du Web 2.0 et la vulgarisation des réseaux sociaux, qui ont conduit les entrepôts de données à évoluer vers d'autres technologies avec de nouvelles appellations, tels que : les entrepôts de données complexes, le Web warehouse, et tout récemment le Big Data. Et ce, afin de permettre la prise en charge de nouveaux besoins d'analyse induits par l'émergence des données non structurées (image, son, texte…).

De ce fait, les besoins d'avoir des temps de traitement rapide des requêtes décisionnelles et une facilité de gestion de cette masse importante de données, ont été toujours considérés comme des objectifs stratégiques. Ces problèmes ont fait l'objet d'un nombre important de travaux de recherche qui ont proposé des solutions fondées sur : les vues matérialisées, les techniques d'indexation, la réutilisation des résultats partiels, l'application des techniques d'apprentissage pour l'optimisation des performances et, plus particulièrement, le recours aux techniques de fragmentation, ou partitionnement, des données pour favoriser le traitement parallèles des requêtes.

Le principe de la conception d'un schéma de fragmentation est de pouvoir constituer des ensembles de données « homogènes » pour accroitre les performances, augmenter la fiabilité et étendre la disponibilité des informations. L'application des techniques de partitionnement aux entrepôts données a été d'un apport considérable en permettant, outre l'amélioration des performances et la capacité de traitement des requêtes décisionnelles, de faciliter les tâches relatives à l'administration des données (chargement, sauvegarde, restauration, nettoyage et archivage).

Néanmoins, l'évolution du modèle et de la charge de travail entravent l'application de cette technique d'optimisation dans les entrepôts de données. Dans le cadre de cette thèse nous nous intéressons aux problèmes d'adaptation des techniques de fragmentation et nous apportant des propositions en matière de conception et d'évaluation des schémas de fragmentation de données.

1.2 Problématiques

Il existe trois approches de fragmentation, l'approche horizontale [CES82], verticale [NAS84] et mixte [SAG86], [ZHY94], [CHC02].

Le partitionnement horizontal, consiste à diviser les objets d'une base de données (table, vue, index) en partitions de même schéma selon des critères de restriction. Quant au partitionnement vertical, il consiste à concevoir des partitions de schémas différents, par projection en dupliquant la clé. En ce qui concerne l'approche mixte, elle consiste à combiner la fragmentation horizontale et verticale [GAG05].

Les algorithmes, sur lesquels se basent les travaux relatifs à la conception d'un schéma de fragmentation optimal, se déclinent en trois catégories : 1) les algorithmes basés sur la minimalité et la complétudes des prédicats [NOA99] [CES82], [OZM91] [OZM91] ; 2) les algorithmes dirigés par l'affinité [NAS84] ; et (3) les algorithmes basés sur un modèle de coût [BEL00], [NAB89].

Le principe de base de ces algorithmes est de générer des fragments en procédant au regroupement (clustering) des prédicats ou attributs selon les fréquences d'accès des requêtes les plus fréquentes. Le regroupement est considéré comme étant un problème combinatoire qui nécessite, pour sa résolution, à faire appel à des heuristiques [BEL00].

En effet, la conception d'un schéma de fragmentation optimal fait partie des problèmes NP-complet [BOK09]. Si on considère *n* simple prédicats pour concevoir une approche de fragmentation horizontale primaire, 2^n est le nombre de fragments horizontaux utilisant des prédicats minterms. Pour un système distribué constitué de *k* nœuds, la complexité pour le placement de ces fragments horizontaux est $O(k2^n)$. Dans le cas d'une fragmentation verticale, une relation de *m* attributs peut être fragmentée en *B(m)* manière [OZM91]. *B(m)* étant le nombre de Bell. Pour un grand nombre d'attributs *B(m)* est approximativement m^m.

Cependant, les algorithmes, de conception d'un schéma de fragmentation optimal, sont statiques. Ils se basent dans leurs entrées sur des informations quantitatives (la sélection des prédicats et la fréquence d'accès des requêtes) et qualitatives (le schéma globale de la base de données) préalablement définies. Si un changement intervient dans les entrées de ces algorithmes, ces derniers doivent être réexécutés afin de déterminer un nouveau schéma de fragmentation.

Ainsi, nous pouvons constater qu'en cas d'évolution des modèles et/ou des changements de la charge de travail ces algorithmes deviennent très complexes et peuvent générer des schémas de fragmentation qui ne seront pas toujours les plus optimaux.

Le problème, qui peut être également relevé pour ces algorithmes, c'est l'utilisation des fréquences d'accès des requêtes comme seul critère de regroupement des prédicats ou des attributs pour générer, respectivement, des fragments horizontaux ou verticaux. Or, dans les entrepôts de données, il existe d'autres paramètres qui sont également très importants et méritent d'être considérés, à savoir : la taille des tables à fragmenter, la taille des vues matérialisées, la tailles des index ainsi que des paramètres physiques, taille de la mémoire centrale par exemple.

De plus, une conception d'un schéma de fragmentation selon les fréquences d'accès des requêtes se trouve parfois inadaptée aux entrepôts de données en raison des caractéristiques spécifiques des requêtes OLAP. Ces requêtes sont longues, complexes et nécessitent parfois un grand nombre d'opérations de sélection, de jointure et d'agrégation.

Elles peuvent manipuler des centaines voire des milliers de tuples. Les requêtes analytiques sont extrêmement variables, elles sont généralement composées d'une manière interactive et peuvent être exécutées une ou plusieurs fois.

Dans le cadre des bases de données relationnelles ou objets et quelque soit l'environnement (centralisé, parallèle, réparti) une grande partie de la littérature a traité cette problématique. Les travaux se sont focalisés sur les techniques de redistribution des données ou de réallocation des fragments en cas de détérioration des performances. Dans ce contexte, on a considéré que la solution réside au niveau physique en appliquant des stratégies d'équilibrage de charge des traitements et des données entres les sites. Le volet logique, à savoir la conception de l'approche de fragmentation, demeure adapté étant donné que la charge de travail est pratiquement stable.

Dans le même contexte, les SGBDs actuels (Oracle, SqlServer, DB2) fournissent uniquement une fragmentation statique des données. Si une table monte en échelle, l'administrateur de la base de données doit redéfinir manuellement le schéma de fragmentation et exécute des utilitaires de redistribution des données. Ceci est devenu une préoccupation croissante, particulièrement, pour les utilisateurs des bases de données parallèles [GAG05].

Par ailleurs, en matière d'estimation de la qualité du schéma de fragmentation, les travaux qui ont abordé cette problématique mesurent la pertinence d'un algorithme par rapport à un autre algorithme, de sélection d'un schéma de fragmentation, selon une fonction de coût à minimiser. Cette fonction porte généralement sur le coût d'exécution des requêtes. Il n'existe pas à priori une approche qui évalue et mesure la pertinence du schéma de fragmentation déjà implémenté et selon de nouvelles informations d'exploitation de la base de données.

Il en ressort, que l'adaptation des approches de fragmentation aux entrepôts de données nécessite une attention particulière. Pour une exploitation efficiente de cette technique d'optimisation, il ne s'agit pas seulement d'analyser les fréquences d'accès aux données pour concevoir un schéma de fragmentation optimal, mais de rendre ce processus dynamique et adapté aux changements de la charge de travail.

1.3 Objectifs et contributions

L'objectif visé dans cette thèse consiste à proposer des solutions aux problèmes de conception des schémas de fragmentation dans les entrepôts de données. Nous nous s'intéressons, plus particulièrement aux aspects liés à l'évaluation et la conception d'un schéma de fragmentation optimal.

Les contributions, développées dans le cadre de ce travail, s'articulent autour de trois propositions :

1. La recherche d'un schéma de fragmentation optimal a un coût exponentiel en temps de calcul et en espace mémoire. Comme indiquée, ci-dessus, cela fait partie des problèmes NP-complet. Pour résoudre ce problème, nous proposons une approche basée sur une métaheuristique, en l'occurrence l'Optimisation par Essaim Particulaire (OEP), pour la conception d'un schéma de fragmentation optimal.

   Notre choix de l'application, d'une version adaptative, de l'OEP par rapport à d'autres méthodes évolutionnaires se justifie principalement par le fait que cette métaheuristique :
   - met l'accent sur la coopération plutôt que sur la compétition. Il n'y a pas de sélection. L'idée étant qu'une particule même actuellement médiocre doit être conservée ;
   - permet d'effectuer un regroupement distribué donc sans contrôle ;
   - s'applique au problème d'optimisation combinatoire dynamique dans le cas ou la fonction objectif varie dans le temps ;
   - existence des versions adaptative de l'algorithme OEP ;
   - facile à programmer ;
   - permet d'utiliser des fonctions multi-objectifs.
2. Une approche d'évaluation des schémas de fragmentation. En effet, mesurer la qualité d'un schéma de fragmentation dont on ne connaît pas à priori le modèle de coût utilisé est une tâche qui n'est pas toujours évidente. De plus, la majorité des algorithmes de sélection d'un schéma de fragmentation sont dirigés par la mesure d'affinité. Dans le cas de la fragmentation verticale, le calcul des fréquences d'accès des requêtes s'effectue uniquement entre une paire d'attributs. Ce qui ne permet pas, par conséquent, de mesurer l'affinité entre tous les attributs d'une partition. Pour répondre à ce besoin, nous présentons une approche formelle basée sur la définition d'une fonction coût permettant d'évaluer et de mesurer l'affinité d'un schéma de fragmentation. Cette fonction de coût sera générique et flexible permettant éventuellement de prendre en considération divers métriques telles que : les informations de placement des données, la capacité de stockage et le coût de transfert des données entre sites.
3. Une approche de fragmentation dynamique des données basée sur l'exploitation d'informations statistiques récentes. Nous développons, à ce titre, un algorithme itératif basé sur la sélectivité des données et nous introduisons pour sa mise œuvre l'utilisation des histogrammes. Ces derniers sont une structure de données

pour la sauvegarde et la manipulation des informations statistiques. Le choix de cette structure est justifié par le fait que les histogrammes :

- peuvent contenir assez d'informations issues des modèles des accès ;
- permettent un traitement efficace des mises à jour, sachant que dans les entrepôts de données ces dernières sont très fréquentes ;
- ne nécessitent pas beaucoup d'espace mémoire ;
- permettent de manipuler avec souplesse les informations conservées ; c'est-à-dire de supprimer facilement l'ancienne historique et de garder uniquement le nouveau historique ;
- sont faciles à implémenter.

1.4 Organisation de la thèse

Le mémoire est organisé en trois parties scindées en 8 chapitres :

1. La première partie consacrée à l'état de l'art est composé de deux chapitres. Dans son premier chapitre, nous présentons un état de l'art des travaux relatifs à l'entreposage de données et en mettent l'accent sur l'insuffisance de l'optimisation des performances offerte par la modélisation multidimensionnelles et les systèmes utilisés pour l'implémentation des entrepôts de données. Nous présentons dans le second chapitre, une étude détaillée sur les techniques de fragmentation et les travaux réalisés dans le cadre des entrepôts de données. Dans la conclusion de cette partie, nous détaillons les problèmes qui entravent l'adaptation de la fragmentation aux entrepôts de données et nous présentons, à l'issue, nos différentes propositions.

2. La deuxième partie présente nos contributions, à savoir : notre approche de fragmentation basée sur l'optimisation par essaim particulaires, une approche d'évaluation du schéma de fragmentation et enfin l'utilisation des histogrammes pour la fragmentation dynamique des données. Ces propositions seront développées respectivement dans les chapitres 4, 5 et 6.

3. La dernière partie présente les résultats des expériences que nous avons menées pour valider nos propositions. Ils sont détaillés dans le chapitre 7. Le chapitre 8 conclut cette thèse. Nous y dressons le bilan de nos contributions et nous présentons nos perspectives de recherche.

PARTIE I : ENTREPOTS DE DONNEES ET TECHNIQUES D'OPTIMISATION DES PERFORMANCES

Notre travail traite de la fragmentation dans les entrepôts de données. Pour clarifier les concepts inhérents à cette problématique, nous présentons dans cette partie le domaine de l'entreposage de données en terme de modélisation et d'exploitation des données pour l'analyse (chapitre 2), et une étude sur les techniques de fragmentation des données et les différentes contributions relatives à l'adaptation de ces techniques aux entrepôts de données (Chapitre 3).

Chapitre 2
Architecture et modélisation des entrepôts de données

2.1 Concepts de l'entreposage de données (*data warehousing*)

La complexité du processus de prise de décision et le besoin avéré de le rendre de plus en plus normalisé et structuré a donné naissance, depuis les années 80, au développement d'outils d'analyse et d'aide à la décision. Pour la réalisation de ce processus, un nouveau concept de base de données a vu le jour, l'entreposage de données ou en anglais le "Data warehousing".

L'objectif principal de l'entreposage de données est d'intégrer les données, provenant de multiples sources, en un seul annuaire, à partir duquel les utilisateurs peuvent aisément trouver des réponses à des requêtes complexes, générer des rapports et effectuer des analyses.

Il existe de nombreuses définitions de l'entreposage de données, dont les plus anciennes se concentrent sur les caractéristiques des données détenues dans l'entrepôt. D'autres définitions élargissent la portée de la notion d'entreposage de données pour y inclure le traitement associé à l'accès aux données à partir des différentes sources, pour les mettre à la disposition des décideurs sous un format adapté [ANS97].

Ainsi, selon Bill Inmon, un entrepôt de données est *"une collection de données orientées sujet, intégrées, historisée et non volatiles, utilisée pour supporter le processus d'aide à la décisions*" [INB05]. Dans cette définition les données sont :

- **Orientées sujet**
  Les données sont orientées métier et donc triées par thèmes, à l'effet de refléter le besoin de disposer de données de support à la décision et non simplement de données orientées application ;
- **Intégrées**
  Les données proviennent de sources hétérogènes et souvent de formats différents. Les données doivent donc être unifiées mise en forme dans un état cohérent avant de les intégrer dans l'entrepôt pour présenter une vue unifiée;
- **Historisées**
  Dans un système de production, la donnée est mise à jour à chaque nouvelle transaction. Dans un entrepôt, la donnée ne doit jamais être mise à jour. Un référentiel temps doit être associé à la donnée afin de suivre dans le temps l'évolution des différentes valeurs des mesures à analyser ;

- **Non volatiles**
  Les données ne sont pas modifiables. Un entrepôt de données doit garantir qu'une requête lancée à différentes dates sur les mêmes données donne toujours les mêmes résultats. De plus, les données d'un entrepôt sont mise à jour périodiquement, ce ne sont donc pas des informations en temps réel.

En résumé, l'objectif à atteindre par la mise en œuvre d'n entrepôt c'est de recomposer les données disponibles pour offrir :

- une vision intégrée et transversale aux différentes fonctions de l'entreprise ;
- une vision métier au travers de différents axes d'analyse ;
- une vision agrégée ou détaillée suivant le besoin des utilisateurs.

2.2 Architecture et modélisation des entrepôts de données

Concrètement, un entrepôt de données est constitué d'un ensemble de méthodes, de matériels et de logiciels regroupés autour de trois grandes fonctions classiques dans le domaine des systèmes de gestion de bases de données, à savoir :

- ✓ l'acquisition
- ✓ le stockage
- ✓ le traitement des données.

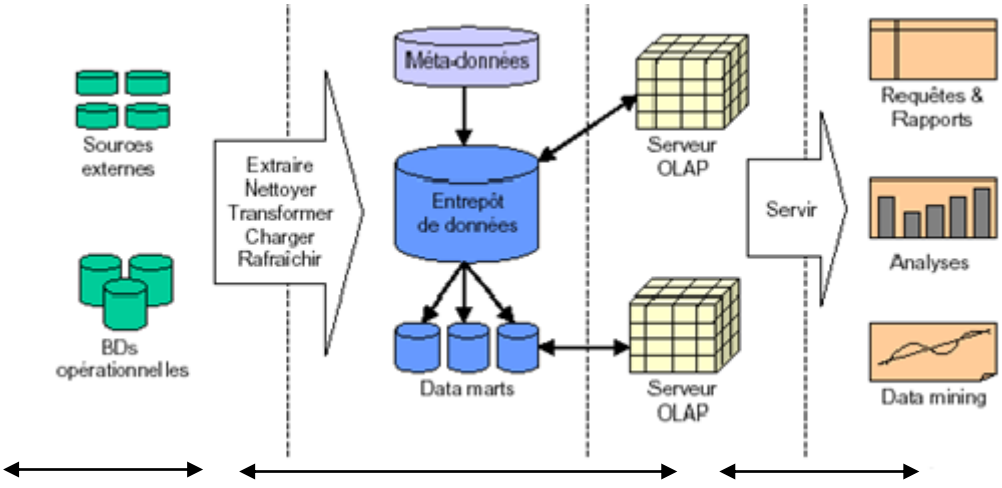


Figure 2.1- Architecture d'un entrepôt de données

2.2.1 L'analyse des données : OLAP vs OLTP

L'objectif principal de la conception des entrepôts de données est d'être un support pour les applications de traitement analytique en ligne OLAP (On Line Analytical Processing) [COE93], [KIR96], [KIR98], [KIR13]. Ce type d'application se caractérise par

une vision multidimensionnelle des données et leurs analyses au travers de l'interrogation interactive. Il s'agit essentiellement de pouvoir analyser des indicateurs (ou mesures) d'une activité selon différents axes d'analyse (ou dimension).

Un SGBD, conçu initialement pour le traitement de transactions en ligne OLTP (On Line Transactionnel Processing), est généralement considéré comme inadéquat à l'entreposage de données. Le tableau 2.1 compare les principales caractéristiques des systèmes OLTP et des systèmes d'entreposage des données [SIH97].

	OLTP	OLAP
Applications	Production	Aide à la décision
Utilisateurs	Un service, professionnel IT	Transversal, non IT
Données	Normalisées non argégées	Dénormalisées, agrégées
Requêtes	Simples, nombreuses, régulières, prévisibles	Complexes, peu Nombreuses,irrégulières
Nb tuples par requêtes	Dizaines	Millions
Taille données	100 MB à 1GB	1 GB à 1 TB
Ancienneté des données	Récentes, mises à jour	Historique

Tableau 2.1- OLAP Vs OLTP

2.2.2 Modélisation multidimensionnelle

Pour permettre la gestion de cette masse de données et pour favoriser l'analyse et l'exploration, la conception des entrepôts de données s'effectue à travers d'une modélisation adaptée, en l'occurrence la modélisation multidimensionnelle [KIR 96]. Ce type de modélisation agence les données d'une façon très différente de la structure en 3FN (3ème forme normale) fréquemment utilisée par les concepteurs des systèmes OLTP. L'utilisateur peut ainsi analyser des informations selon diverses perspectives, par rapport à différents axes (par exemple les ventes par rapport aux dimensions temps, produits et magasins de la figure 2.7).

La dénormalisation du modèle des données et la redondance d'informations sont totalement envisageables dans un contexte d'entreposage de données. La non volatilité des données permet de ne pas se préoccuper, lors de l'exploitation de l'entrepôt, des problèmes d'intégrité des données ou de transaction.

Afin de répondre à ce type de besoin, tout en conservant la technologie des SGBD relationnels, il sera souvent nécessaire de modéliser les données de manière particulière,

en distinguant les différents axes et les indicateurs à analyser. On parlera alors de modèle en étoile, en flocon de neige ou en constellation (voir section 2.3.2).

Comme indiqué, ci-dessus, un entrepôt de données est orienté sujet et doit permettre d'analyser des données suivant plusieurs dimensions. Les bases de données multidimensionnelles, à la différence des bases de données relationnelles classiques, permettent d'effectuer des traitements sur des données en prenant en compte plus de deux axes : les données ne sont pas modélisées sous forme tabulaire mais sous forme d'hyper-cubes (ou « cubes de données » ou encore « data cubes »). Un exemple d'un tel cube est présenté à la figure 2.3. Ces cubes sont parfois appelés hyper-cubes s'il y a plus de 3 dimensions. Les données pourront être interrogées directement et facilement sur n'importe quelle combinaison de dimension. Passer d'une hiérarchie de dimension à une autre est réalisée facilement dans un cube de données par la technique de pivot, de rotation. Par cette technique, le cube peut être pivoté pour afficher différentes orientations des axes. Par exemple, on peut pivoter un cube pour afficher les régions en lignes, les trimestres en colonnes et les produits dans la troisième dimension. Cette technique est équivalente à avoir une table de vente par région pour chaque produit, où chaque table affiche les ventes par trimestre et par région du produit.

2.2.2.1 Notions

- **Sujet**

Un sujet est défini par un ensemble de mesures et un ensemble de dimensions. Par exemple, les mesures d'une vente peuvent être son numéro, son prix et sa quantité : ce sont les valeurs numériques que l'on veut comparer. Les dimensions seraient la date, le type de produit et la région : ce sont les points de vue depuis lesquels les mesures peuvent être observées.

- **Les dimensions :**

Une dimension est une liste d'éléments organisés de façon hiérarchique. La granularité d'une dimension est son nombre de niveaux hiérarchiques. Par exemple, pour le temps, nous pourrions avoir la hiérarchie suivante : année, semestre, trimestre, mois, semaine, jour, soit six niveaux. Les axes de dimensions doivent fournir des règles de calcul d'agrégat pour chaque mesure. Par exemple, le nombre des ventes du premier trimestre est la somme du nombre des ventes de janvier, février et mars. On aura alors une vision pyramidale des données, la base de la pyramide représentant le niveau le plus détaillé et le haut le niveau le plus global.

Une dimension détermine la manière dont on regarde les données pour les analyser. Elle est formée par un ensemble de descripteurs et chaque descripteur peut prendre différentes valeurs. On distingue des dimensions plates et des dimensions hiérarchiques.

Les *hiérarchies* associées à une dimension organisent les descripteurs de cette dimension à différents *niveaux* pour analyser les données à différents niveaux de granularité (figure 2.2). On note l'existence pour chaque dimension d'une valeur *ALL* correspondant à la granularité de plus haut niveau. Un *niveau* est un ensemble nommé de membres. En général, le niveau le plus bas est celui de l'entrepôt. Les agrégations successives sur ces données fournissent de nouveaux points de vue qui sont de moins en moins détaillés et qui constituent des niveaux de plus en plus élevés dans la hiérarchie

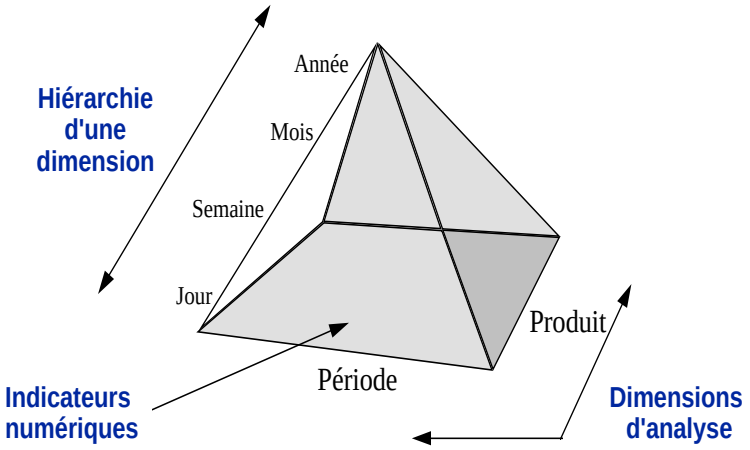


Figure 2.2- Organisation d'une dimension

- **Les faits**

Un fait représente la valeur d'une mesure calculée selon un attribut de chacune des dimensions. Par exemple, « 5000» est un fait qui exprime la valeur de la mesure « coût des travaux » pour « 2002 » de l'attribut « année » de la dimension « temps » et de la ville « Alger » de l'attribut « ville » de la dimension « région ». Les mesures sont stockées dans des tables de faits qui contiennent les valeurs des mesures et les clés primaires permettant d'établir des relations avec les tables de dimension.

- **Cube de données**

Un cube de données (ou hypercube) est un ensemble de données organisées selon une ou plusieurs *dimensions* qui déterminent une *mesure* d'intérêt. On appelle *mesure* la valeur contenue dans une cellule du cube associée aux valeurs (ou positions) prises sur les dimensions composant le cube.

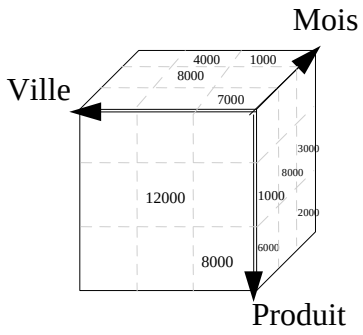


Figure 2.3. Exemple de cube

Un exemple de cube est présenté dans la figure 2.3. Il s'agit d'un cube à 3 dimensions : Produit, Ville et Temps. Dans chaque cellule du cube est stocké le total de ventes pour un produit donné, sur une ville particulière et pour un mois donné. La mesure considérée est le total de ventes, et les descripteurs de dimensions sont la ville, le produit et le mois.

D'une manière formelle, la structure d'un cube se présente comme suit [AGR 97]:

- chaque dimension *d* porte un nom D_i. A chaque dimension est associé un domaine de valeurs Dom_{Di}.
- une référence de cellule est un n-uplet $< v_1,... v_d >$ appartenant à Dom_{D1} *X* Dom_{D2} *X .....X* Dom_{Dd}. Le contenu d'une cellule du cube est soit la constante 0, soit la constante 1, soit un n-uplet $<m_1,..., m_k>$ appartenant à $Dom_{D'1}$ *X* $Dom_{D'2}$ *X .....X* $Dom_{D'd}$. Tel que Dom_{D1} *X* Dom_{D2} *X .....X* Dom_{Dd} sont des domaines discrets et $Dom_{D'1}$ *X* $Dom_{D'2}$ *X .....X* $Dom_{D'd}$ sont des domaines numériques.

Un cube ***C*** de *d* dimensions est une fonction $\boldsymbol{F}_c$ associant à chaque cellule $<v_1,...,v_n>$ l'un des éléments suivants :

- la constante 0 si la cellule de référence n'existe pas pour **C**,
- la constante 1 si la cellule de référence $<v_1,..., v_d>$ existe mais ne contient pas de mesure,
- Un n-uplet $<m_1,..., m_k>$ si la cellule de coordonnées $<v_1,..., v_d>$ contient *k* mesures.

Dans le même contexte, il s'avère également opportun d'introduire la notion de cuboïde [HAJ00]. Un cuboïde est le résultat d'une requête *Group BY* selon un ensemble *X* de dimensions.

Considérant les dimensions du cube présenté à la figure 2.3, Ville, Produit et Mois, les cuboïdes correspondants sont alors les suivants : (ville, produit, mois), (ville, produit), (ville, mois), (produit, mois), (ville), (produit), (mois), ()}, où () signifie que

l'ensemble des attributs group-by est vide. Ces attributs group-by forment un treillis de cuboïdes pour le cube de données. Le treillis de cuboïdes correspondant est présenté à la figure 2.4.

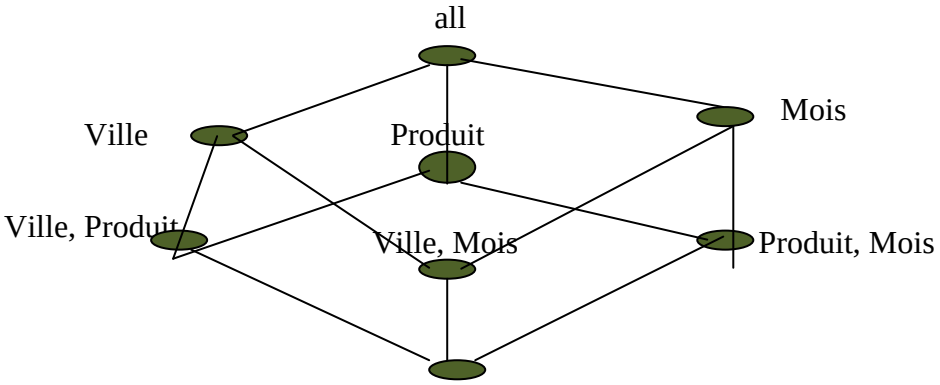


Figure 2.4- Treillis de cuboïdes

Pour chaque mesure *mj*, on associe une fonction d'agrégation $f_j : P(D'_j) \rightarrow D_j$, *où* D'_j est le domaine de valeurs pour la $j^{ème}$ mesure et l'argument en entrée de la fonction est un multi-ensemble.

A noter qu'ils ne sont considérées que les fonctions SQL *MIN, MAX, COUNT*, *SUM* qui sont additives ou qui peuvent être exprimées comme une fonction scalaire de mesures additives (c'est le cas de la fonction *AVG* qui est obtenue par *SUM* et *COUNT*).

2.2.2.2 Les opérations sur les cubes

Pour exploiter et visualiser les données contenues dans les cubes, des opérations spécifiques à cette structure multidimensionnelle ont été misent en œuvre. On distingue des opérations sur la structure du cube et des opérations sur le contenu du cube [POU00, [VAP98], [RAM03].

- **Opérations sur la structure**

- La rotation (*Rotate*) consiste à faire une rotation du cube de manière à présenter une face différente (examiner le cube selon un autre angle). Par exemple, pour un cube à trois dimensions *X*, *Y*, et *Z* et une mesure *M*, on peut positionner le cube de façon à ce que les dimensions *X* et *Y* soient visibles, ensuite les tourner pour visualiser *X* et *Z* ;
- L'inversion (Switch) consiste à interchanger la position (ou l'ordre) des membres d'une dimension ;
- La décomposition (*Split*) consiste à décomposer les données selon certaines va leurs de dimensions. Cette opération permet de réduire le nombre de dimensions ;

- L'imbrication (*Nest*) permet de grouper sur une même représentation bidimensionnelle toutes les informations (mesures et membres) d'un cube quel que soit le nombre de dimensions. L'opération (*Unnest)* reconstitue une dimension séparée à partir des membres imbriqués ;
- La concaténation (*Push*) consiste à combiner les membres d'une dimension aux mesures du cube, c'est-à-dire de faire passer des membres comme contenus de cellules. L'opération inverse (*Pull*) permet de transformer certaines mesures du cube en membres.

▪ **Opérations sur le contenu du cube**

- La généralisation (*Roll-Up*) effectue l'agrégation des mesures en allant d'un niveau particulier de la hiérarchie vers un niveau général. Cette opération consiste à faire des groupes dans les modalités d'une dimension et de calculer un agrégat sur ces groupes ;
- L'opération inverse du *Roll-Up*, la spécialisation (*Drill-Down*) consiste à représenter les données du cube à un niveau inférieur, et donc sous une forme plus détaillée ;
- La restriction (*Slice*) consiste à extraire de l'information résumée pour une certaine dimension. Il s'agit d'une sélection sur les cellules. On ne retient alors que les valeurs correspondant à un certain critère (par exemple toutes les valeurs supérieures à 10000) ;
- La projection (*Dice*) est une restriction sur les dimensions (une sélection sur les dimensions).

2.3 Les différents systèmes OLAP

Deux approches sont généralement utilisées pour l'implémentation d'un système basé sur le modèle multidimensionnel : l'approche ROLAP (Relational OLAP), dans laquelle un système de gestion de bases de données relationnel est utilisé pour stocker les données, et l'approche MOLAP (Multidimensional OLAP) [CHS97],[DOA01], [THD00], où les données sont physiquement implémentées de manière multidimensionnelle. Il existe une autre approche nommée HOLAP (Hybrid OLAP), où l'on couple les deux premières approches.

Il convient de noter que d'autres approches et outils ont été développés pour la prise en charge des entrepôts de données spécifiques. On cite particulièrement les approches SOLAP (pour *spatial OLAP)* et XOLAP (ou XML-OLAP), qui permettent, respectivement, la combinaison des outils OLAP et les capacités de représentations cartogra-

phiques pour l'exploitation des informations géographiques et l'exploitation des entrepôts de données complexes ou XML.

2.3.1 Le système MOLAP

Dans les systèmes MOLAP, les données sont stockées sous la forme d'un tableau multidimensionnel. Chaque dimension de ce tableau est associée à une dimension d'une structure de données en l'occurrence le cube.

MOLAP est conçue exclusivement pour l'analyse multidimensionnelle, avec un mode de stockage optimisé par rapport aux chemins d'accès prédéfinis. Ainsi, toute valeur d'indicateur associée à l'axe temps sera pré-calculée au chargement pour toutes ses valeurs hebdomadaires, mensuelles, etc.

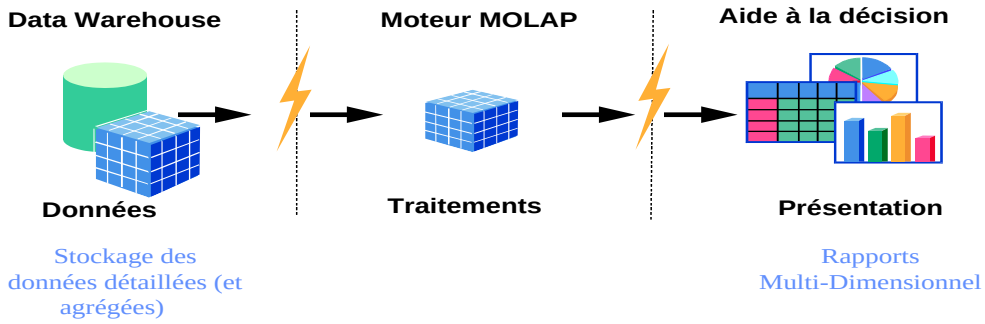


Figure 2.5- Architecture d'un système MOLAP

2.3.2 Le système ROLAP

Les systèmes ROLAP sont les plus utilisés pour l'implémentation des entrepôts de données. Dans ces systèmes, les données sont stockées dans une base de données relationnelle. Le cube n'est réellement crée qu'au moment de la phase de requête et par conséquent les temps de réponses aux requêtes seront élevés.

Les outils ROLAP diminuent sensiblement le coût lié à la mise en œuvre d'un serveur de base de données multidimensionnelle supplémentaire. Au travers des métadonnées, ils permettent de transformer l'analyse multidimensionnelle demandée par l'utilisateur en requêtes SQL. Pour cela, ces outils s'appuient pour la plupart sur une modélisation particulière des données, distinguant les axes d'analyse et les faits à observer. On parlera notamment de modèle en étoile et de modèle en flocon ou encore des techniques de définition physique d'agrégation. Ceci oblige à définir le modèle en fonction de l'outil à utiliser et des analyses à mener mais, en revanche, une amélioration de performance et une cohérence lors de l'utilisation de ce type de produits.

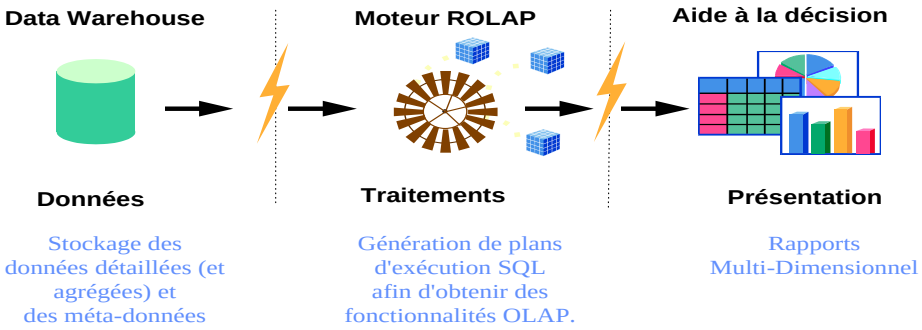


Figure 2.6- Architecture d'un système ROLAP

Trois schémas sont utilisés pour modéliser les systèmes ROLAP : (1) le schéma en étoile, (2) le schéma en flocon de neige, (3) le schéma en constellation.

2.3.2.1 Le schéma en étoile

Dans un schéma en étoile, les mesures sont stockées dans une table appelée table des faits et chaque dimension correspond à une table appelée table de dimension. La table des faits est constituée d'attributs représentant les mesures d'activité et des attributs clés étrangères de chacune des tables de dimension. Les tables de dimension contiennent les paramètres de l'analyse et une clé primaire permettant de réaliser des jointures avec la table des faits. Cette manière d'organiser les données fait que les temps de réponses pour les requêtes peuvent être élevés (du fait des jointures nombreuses entre les tables).
En fait, le modèle en étoile essaie de superposer une structure multidimensionnelle au dessus d'un modèle relationnel normalisé à deux dimensions.
Les requêtes généralement utilisées pour ce schéma sont appelées des requêtes de jointure en étoile (*star-join-queries*) qui sont caractérisées :

- Par des jointures multiples entre les tables des faits et les tables de dimension ;
- Il n'y a pas de jointure entre les tables de dimension ;
- Chaque table de dimension impliquée dans une opération de jointure a plusieurs prédicats de sélection sur ses attributs descriptifs.

La syntaxe en SQL de ce type de requête est la suivante :

```
Select <liste de projection> <liste d'agrégation>
From <nom de la table des faits> <liste de nom de tables
      de dimension>
Where <liste de prédicats de sélection et de jointure>
Group by <liste des attributs de tables de dimensions>
```

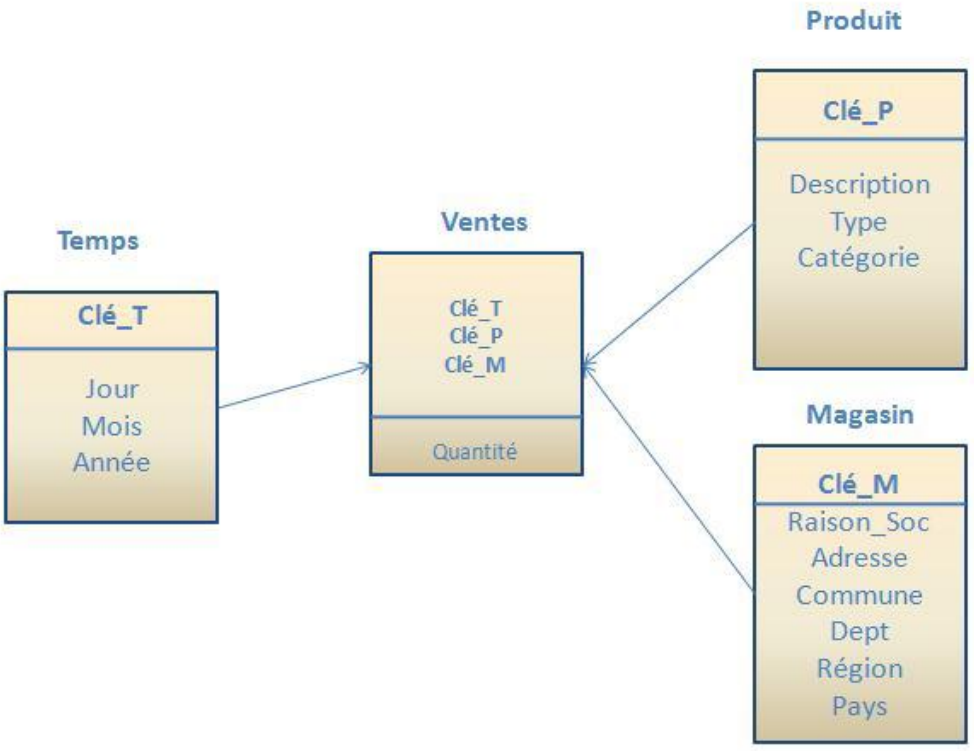


Figure 2.7- Schéma en étoile

2.3.2.2 Le schéma en flocon de neige

La modélisation en flocon est une modélisation en étoile pour laquelle on représente les dimensions selon différents niveaux de hiérarchies à travers l'éclatement des tables de dimension en sous-tables contrairement au schéma en étoile, où chaque dimension correspond à une seule table.

Dans la théorie, la différence réside dans la simple normalisation des tables de dimensions en 3ème FN. Il est donc tout simplement question de mettre les attributs de chaque niveau hiérarchique dans une table de dimension distincte pour éviter la redondance. Bien que ce modèle permet de gagner de l'espace disque et facilite l'alimentation, il implique de nombreuses jointures dans les requêtes et donc une difficulté d'écriture de ces dernières.

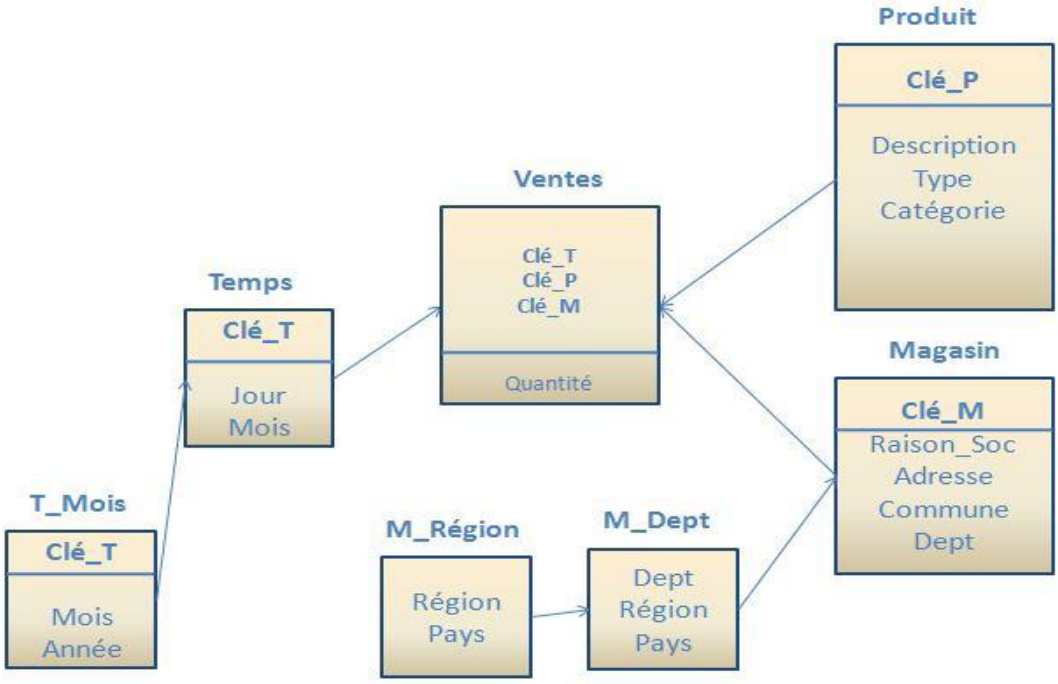


Figure 2.8- Schéma en flocon de neige

2.3.2.3 Le schéma en constellation de faits

Les schémas les plus appropriés pour la modélisation des entrepôts de données font appel à un mélange de schémas dénormalisé en étoile et un schéma normalisé en flocon. Cette combinaison de schémas en flocon et en étoile s'appelle un *schéma en constellation*. Se sont plusieurs tables de faits reliées aux tables de dimension.
Il s'agit, en fait, de satisfaire des exigences de requêtes analytiques, de fusionner plusieurs schémas en étoile qui utilisent des dimensions communes [GHF05], [SOA05].

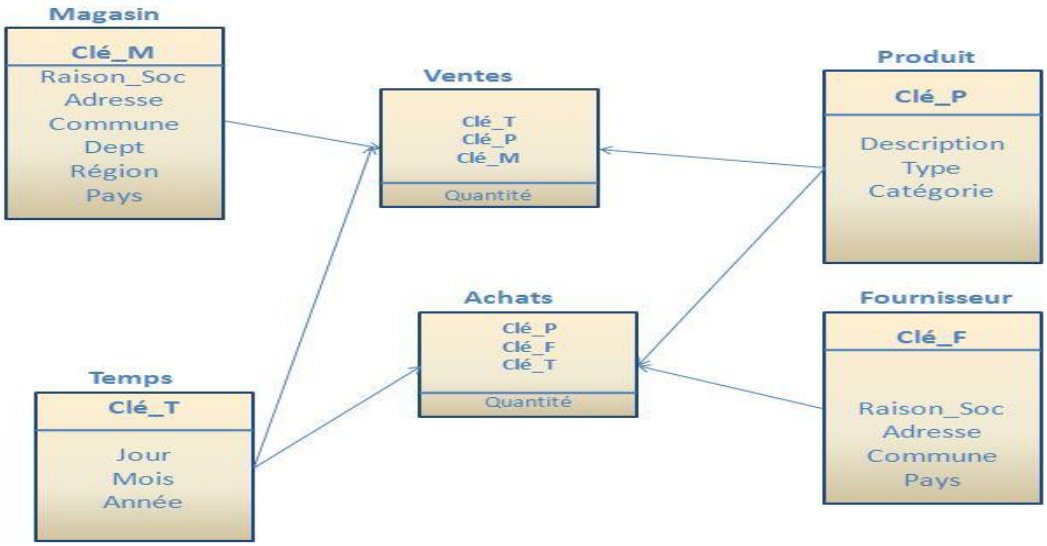


Figure 2.9- Schéma en constellation de faits

2.3.3 Le système HOLAP

Dans les systèmes HOLAP [PEN00], seule une partie du cube est stockée sous forme multidimensionnelle et le reste est laissé dans la base de données. L'avantage de cette approche est le compromis entre l'optimisation de stockage et les temps de réponses aux requêtes.

2.4 Conclusion

Les entrepôts de données intègrent des données issues de différentes sources, parfois hétérogènes et dont le nombre peut évoluer. Le chargement des données s'effectue à travers le processus ETL (*Extract-Load-Transform*) par lot et d'une manière périodique. Ceci signifie que le volume de données dans un entrepôt de données est en perpétuelle évolution, et par conséquent plusieurs problématiques sont à considérer, comme le stockage des données, l'hétérogénéité des données et plus particulièrement la complexité du traitement des requêtes décisionnelles.

Afin de palier à ces problèmes, le recours à des techniques d'optimisation, pour gérer les grands volumes de données et traiter des requêtes décisionnelles complexes, est plus qu'indispensable et ce malgré l'existence de formes physiques implémentées dans les entrepôts de données.

En effet, les outils ROLAP, considérés comme étant les mieux adaptés aux gros volumes de données, proposent le plus souvent un composant serveur, pour optimiser les performances lors de la navigation dans les données ou pour les calculs complexes.

ROLAP n'agrège rien, mais tire parti des agrégats s'ils existent. De ce fait ROLAP est plus lourd à administrer que MOLAP, puisqu'il demande de créer explicitement certains agrégats.

Par ailleurs, toujours en terme de performance, les systèmes MOLAP offrent de bon temps de réponses aux requêtes que les systèmes traditionnels. Cependant, ces systèmes réservent un emplacement de cellule même si cette cellule est vide ce qui est coûteux en terme d'espace mémoire de stockage. MOLAP agrège tout par défaut. Plus le volume de données à gérer est important, plus les principes d'agrégation implicites proposés par MOLAP sont pénalisants dans la phase de chargement de l'entrepôt, tant en terme de performance qu'en capacité de stockage. La limite fréquemment évoquée pour MOLAP étant de quelques giga octets.

S'agissant de la modélisation multidimensionnelle, le schéma en étoile simplifie le modèle logique normalisé en organisant les données de manière optimale pour les traitements des requêtes analytiques. Les schémas en étoile permettent d'augmenter les performances des requêtes grâce à la dénormalisation des données contenues dans les tables de dimensions. La dénormalisation s'avère adéquate quand un certain nombre d'entités reliées à la table de dimension reçoivent des accès fréquents, ce qui réduit la charge de joindre des tables supplémentaires pour accéder à ces attributs. La dénormalisation, par contre, engendre une complexité d'alimentation et plus de capacité de stockage à cause de la redondance des données.

Il en résulte, que malgré l'existence de ces forme d'optimisation, la mise en place d'autres techniques d'optimisation, telles que : les indexes, les vues matérialisées, la fragmentation des données et le traitement parallèles des requêtes analytique, s'avèrent nécessaire pour l'exploitation et l'administration des entrepôts de données.

La problématique de l'optimisation des performances dans les entrepôts des données sera détaillée dans la section qui suit, dans laquelle nous mettons l'accent plus particulièrement sur les techniques de fragmentation des données, objets de cette thèse.

Chapitre 3

Les techniques de fragmentation et leurs applications dans les entrepôts de données

3.1 Introduction

Le processus d'exploration et d'analyse des données s'appuie sur des requêtes complexes qui regroupent un nombre important d'opération de jointure et d'agrégat. Ce processus, qui se caractérise par l'aspect interactivité [GAG05], requiert un temps de réponse rapide. L'exigence est que le temps de réponse aux requêtes décisionnelles ne devra pas dépasser quelques secondes, voire à la limite quelques minutes. Or, sans l'implémentation des techniques d'optimisation de performance l'exploitation des entrepôts de données s'avère complexe et le traitement des requêtes peut prendre des heures voire des jours [BEL00].

Pour une meilleure exploitation des entrepôts de données, plusieurs techniques d'optimisation ont été proposées et adaptées. On peut citer les vues matérialisées [ZHC01], les index [CHS04, TAA08], la fragmentation des données [BEL00, BEL05, NOA99, SAA04], le groupement [JAH99], le traitement distribué [BEJ02] et le traitement parallèle [FUP04].

Dans ce qui suit nous présentons succinctement les techniques d'optimisation les plus appliquées au contexte des entrepôts de données, en l'occurrence les index, les vues matérialisées et le traitement parallèle, tout en se focalisant dans les chapitres qui suivent sur la fragmentation des données.

3.2 Les techniques d'optimisation dans les entrepôts de données

Les travaux concernant l'optimisation des performances des requêtes décisionnelles sont inspirés des techniques héritées des bases de données relationnelles/objets. L'indexation et les vues matérialisées sont les techniques les plus utilisés dans les entrepôts de données [BEL00, AOK05, GOM99, GOM02, BOK09].

3.2.1 Les vues matérialisées

Les vues matérialisées se basent sur le principe de matérialisation des requêtes les plus fréquentes par duplication de données. L'utilisation des vues permet de stocker physiquement le résultat retourné par l'exécution des requêtes OLAP, qui renferment des opérations de jointure et d'agrégation qui sont des opérations très coûteuses en

temps d'exécution. Ce type d'optimisation permet donc une amélioration en terme de réponse de certaines requêtes grâce à l'accès aux données de la vue dont le nombre d'instances est beaucoup moins important que celui des tables de dimensions et de faits.

Dans les entrepôts de données, deux principales possibilités sont utilisées pour la sélection d'ensemble de vues :

- **Pré-calculer toutes les vues**

Cette approche consiste à matérialiser la totalité du cube dans le cas d'une implémentation de l'entrepôt de données sur un système MOLAP. Chaque cellule du cube est considérée comme une vue potentielle. Tandis que dans un système ROLAP, elle consiste à matérialiser tous les nœuds intermédiaires des arbres algébriques représentant les requêtes. Cependant, dans le cas d'un entrepôt contenant un volume de données important, stocker et maintenir toutes les cellules ou nœuds intermédiaires est le principal inconvénient de cette approche.

- **Matérialiser uniquement une partie des cubes ou des nœuds**

Dans le cas de manipulation des cubes de données, il n'est pas nécessaire de matérialiser toutes les vues, et ce à cause de la dépendance entre les cellules, c'est à dire que les valeurs contenues dans certaines d'entre elles peuvent être réutilisées à partir des valeurs d'autres cellules. De même pour les systèmes ROLAP, cette dépendance existe également dans les arbres algébriques. Il est alors souhaitable de matérialiser les sous-ensembles communs (cellules ou nœuds) à plusieurs requêtes.

L'utilisation des vues matérialisées se confronte à plusieurs problématiques dont les principales sont : la sélection des vues matérialisées et la maintenance des vues matérialisées.

3.2.1.1 Problème de sélection des vue matérialisées

Le problème de sélection des vues matérialisées (PSV) consiste à identifier un sous-ensemble de vues candidates en tenant compte de certaines contraintes (espace de stockage réservé, coût de maintenance). Ainsi, il faut sélectionner un ensemble de vues afin d'optimiser le coût d'exécution des requêtes sous une contrainte d'espace de stockage ou de maintenance [BAE97, AOK05].

Un problème PSV peut être formulé de la manière suivante Soit :

- $V=\{v_1,...,v_n\}$ un ensemble de vues ;
- $Q=\{Q_1,...,Q_m\}$ un ensemble de requêtes ;
- Une contrainte de ressource S (la taille de l'espace de stockage allouée pour les vues, par exemple) ;

Le PSV consiste à trouver une configuration de vues tel que le coût d'exécution des requêtes soit minimal (ou le coût de maintenance des vues) et satisfaisant la contrainte *S*.

Le PSV est considéré comme étant un problème NP-Complet [GUH99]. Le nombre de vues à sélectionner pour la matérialisation peut être très important. Si *d* est le nombre de table de dimension, le nombre de vues sera égal à 2^d. La complexité du PSV est de $O(2^n)$ où *n* représente le nombre de vues candidates dans le schéma [BOK09]. Pour résoudre ce type de problème il faut utiliser des méthodes non exactes comme les heuristiques afin de rechercher une solution optimale ou quasi-optimale. Les principaux travaux qui ont traité cette problématique peuvent être divisés en deux catégories en fonction de l'objectif de l'optimisation recherché [BEL00] :

1. Les travaux dirigés par la contrainte d'espace, dans lesquels la taille maximale allouée pour matérialiser les vues ne devra pas dépasser la taille totale de stockage [BAX03, AOK06] ;
2. Les travaux dirigés par le temps de maintenance des vues, dans lesquels ce temps ne devra doit pas dépasser un certain seuil fixé au préalable [GUH05].

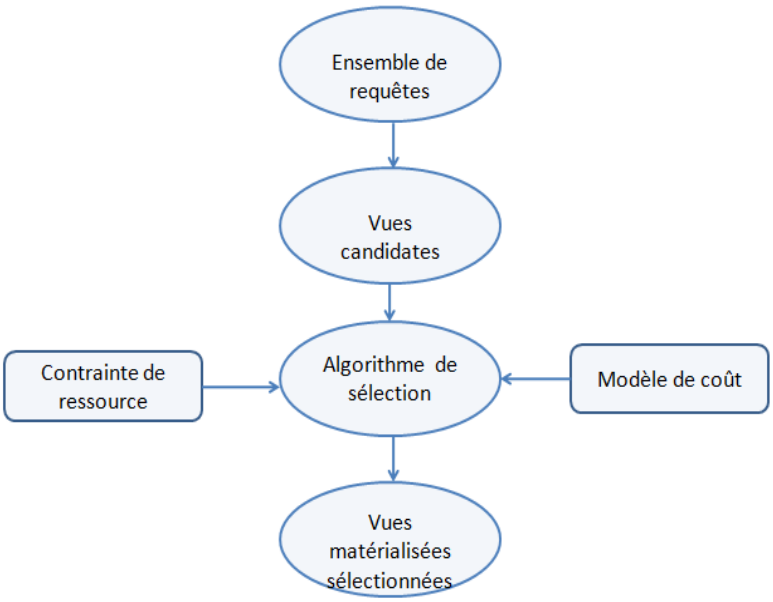


Figure 3.1- Le processus de sélection des vues matérialisées.

Dans le contexte des entrepôts de données, le problème de sélection des vues matérialisées peut être abordé sous deux angles différents en fonction du type de modèle de données adopté (figure 3.2): 1) Le PSV de type ROLAP ; 2) le PSV de type MOLAP.

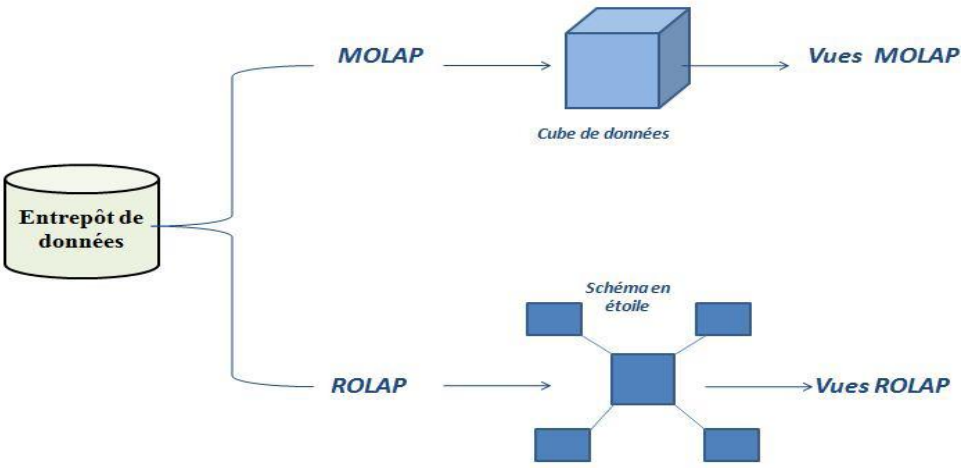


Figure 3.2- Les types de PSV dans les entrepôts de données

- Dans la présentation cubique (PSV dans MOLAP), chaque cellule du cube présente une vue potentielle. Notons que les vues matérialisées à ce niveau ne sont que des requêtes ayant des agrégations sur les relations de base ;
- Dans le PSV de type ROLAP, chaque requête est représentée par un arbre algébrique. Chaque noeud (non feuille) est considéré comme une vue potentielle. Ce type de PSV est plus général que le premier ;

3.2.1.2 Le problème de la maintenance des vues matérialisées

Le problème de la maintenance des vues matérialisées (PMV) porte sur l'aspect évolution et changements des données de l'entrepôt. Les vues matérialisées représentent le stockage physique du résultat des requêtes fréquemment exécutées sur l'entrepôt. Lorsqu'il y a des modifications sur les données, celles-ci doivent être reportées sur les vues afin de s'assurer de la cohérence de leurs contenues. La maintenance des vues matérialisées consiste à apporter les modifications survenues sur les tables de base au niveau des vues. Ce processus génère, d'une part, un coût de maintenance, qui peut surcharger le système, et d'autre part, un espace de stockage additionnel qu'il faut allouer pour les vues [AOK05].

Plusieurs travaux ont été réalisés pour traiter le problème de maintenance des vues matérialisées, et ce dans l'objectif de tenter de réduire le temps d'exécution des opérations de maintenance [GAC06, KIN07].

En résumé, la maintenance des vues matérialisées peut se faire selon trois approches : périodique, immédiate et différée [BOK09]. Dans la première, les vues sont mises à jour continuellement à des périodes précises. Dans la seconde, les vues sont mises à jour

immédiatement à la fin de chaque transaction. Enfin, dans la dernière approche, les modifications sont propagées d'une manière différée. Une vue est mise à jour uniquement au moment où elle est utilisée par une requête d'un utilisateur.

3.2.2 Les index

Les index sont utilisés pour réduire les temps de traitement des requêtes. L'utilisation des index est adaptée pour les requêtes qui nécessitent d'ordonner les tuples selon un ou plusieurs attributs comme les clauses GROUP BY et ORDER BY. Ils permettent également de réduire le nombre d'opérations d'entrée-sortie, particulièrement pour les opérations de jointures. Si l'attribut de jointure entre deux ou plusieurs tables est indexé, il suffit juste de calculer la jointure en utilisant les index sans pour autant accéder aux tables [GOM02].

Un index peut être défini sur une ou sur plusieurs colonnes d'une relation. Ce type d'index est appelé mono-index. Il existe également des index définis sur deux relations comme les index de jointure qui sont appelés multi-index. Dans les entrepôts de données, les deux types d'index sont utilisés : index sur liste de valeur, index de projection (mono-index) et index de jointure en étoile (star join index) pour les index multi-index [ZIE10].

***Exemple 1* :** Index de projection

Soit *Col* la colonne d'une table à indexer. L'index de projection (figure 3.3) sur *Col* est constitué par une séquence des valeurs de *Col*. Ces dernières apparaissent dans le même ordre que le numéro des tuples dans la table d'origine.

Table client *Col*

Nom	**Age**	**…**	**Sexe**
Sofiane	32		M
Nesrine	17		F
Zineddine	43		M
Feriel	37		F
Kamel	28		M
Siham	41		F

Index de projection

Age
32
17
43
37
28
41

Figure 3.3- Index de projection sur l'attribut Age.

Parmi les techniques d'indexation proposées dans le cadre des bases de données classiques, nous pouvons citer l'index B-tree, l'index de hachage, l'index de projection, l'index de jointure, etc. La majorité de ces types index est utilisée dans les entrepôts de données. Certaines techniques d'indexation sont apparues dans le contexte d'entrepôts de données comme les index binaires, les index de jointure binaires, les index de jointure en étoile [BOK09].

En effet, les index de jointure sont parfaitement adaptés pour les requêtes des systèmes OLTP parce qu'elles possèdent fréquemment des jointures entre deux tables. Par contre, pour les entrepôts de données modélisés par un schéma en étoile, ces index sont limités. En effet, les requêtes décisionnelles définies sur un schéma en étoile possèdent plusieurs jointures (entre la table des faits et les tables de dimensions). Il faut dans ce cas subdiviser la requête en fonction des jointures. Or le nombre de jointures possibles est de l'ordre de $N!$ (N étant le nombre de tables à joindre).

Afin de résoudre ce problème, un nouvel index appelé index de jointure en étoile (star join index) a été introduit [SYS97], et adapté aux requêtes définies sur un schéma en étoile. Un index de jointure en étoile peut contenir toute combinaison de clés étrangères de la table des faits. Il peut être sous la forme de n'importe quelle combinaison contenant la clé de la table des faits et une ou plusieurs clés primaires des tables de dimensions. Ce type d'index est dit complet s'il est construit en joignant toutes les tables de dimensions avec la table des faits. Un index de jointure partiel est construit en joignant certaines tables de dimensions avec la table des faits.

De même que pour les PSV, au niveau de la conception physique, l'administration de l'entrepôt de données doit sélectionner des index afin d'accélérer l'exécution des requêtes. Ce problème est connu sous le nom de **problème de sélection d'index (PSI),** plusieurs solutions ont été proposées [CHS98, CHS99, CHS04, AOK05, BEL00, BOK09].

3.2.3 Le traitement parallèle

Les volumes des données gérés et exploités dans les entrepôts de données sont très importants. De plus, les requêtes décisionnelles issues des outils d'analyse peuvent être complexes. Les architectures distribuées et le traitement parallèle des requêtes sont considérés, à cet effet, comme des solutions d'optimisation les plus adaptées aux entrepôts de données.

Le parallélisme au sens le plus général pourrait être défini comme une technique qui permet d'utiliser simultanément plusieurs machines pour mener à bien l'exécution d'un programme. Du point de vue SGBD, la gestion de l'entrepôt par une machine parallèle

nécessite le support du parallélisme inter-requête et surtout du parallélisme intra-requête.

- **Parallélisme inter-requête (Inter-Query Parallelism)**

Définition 1 : *le parallélisme inter-requête est une technique de parallélisme permettant de faire prendre en charge les requêtes par des processeurs différents.*

Le parallélisme inter-requête nécessite un SGBD à serveurs multiples capable d'assigner un serveur à chaque processeur et de dispatcher les requêtes entre les serveurs. Les synchronisations entre serveurs accédant aux mêmes données doivent aussi être gérées. Les gains en performance restent limités.

- **Parallélisme intra-requête (Intra-Query Parallelism)**

Définition 2 : *le parallélisme intra-requête est une technique de parallélisme consistant à découper une requête en unités fonctionnelles, chaque unité pouvant être prise en charge par un ou plusieurs processeurs différents travaillant en parallèle.*

Pour le parallélisme intra-requête, plusieurs processeurs coopèrent pour exécuter en parallèle les opérations d'une même requête. Dans ce dernier niveau, deux formes de parallélisme peuvent être extraites :

1. Le parallélisme *intra-opération* permettant l'exécution parallèle d'une même opération sur des données obtenues par une méthode de répartition,
2. Le parallélisme *inter-opération* permettant l'exécution parallèle de plusieurs opérations d'une même requête.

Dans le parallélisme inter-opération, on peut dégager deux *types de parallélisme* : le parallélisme *indépendant* et le parallélisme *dépendant* (appelé aussi parallélisme *pipeline ou flux*). Le parallélisme dépendant correspond à une exécution parallèle des opérations qui communiquent.

L'implémentation des entrepôts de données dans un environnement parallèle engendre de nouveaux paramètres à considérer par rapport à une architecture centralisée, notamment :

- Le choix de l'architecture parallèle. En effet, le type d'architecture parallèle influe sur la détermination du meilleur algorithme de jointure puisque la fonction temps de réponse d'un algorithme dépend de l'organisation des données en mémoire (mémoire commune, mémoire distribuée, ...) ;
- Le choix d'une topologie de réseau d'interconnexion. La topologie du réseau a un impact direct sur les temps de communication de données et de contrôle ;

- Les communications de données et de contrôle engendrées par l'exécution parallèle d'une requête. Le temps de communication peut devenir prohibitif, ce qui dégrade le temps de réponse et diminue l'apport du parallélisme,
- Le nombre de processeurs optimal alloués à une opération relationnelle. Il s'agit de déterminer le point de compromis traitement-communication. L'analyse des algorithmes de jointure montre que la courbe du temps de réponse en fonction du nombre de processeurs est constituée de deux parties. Dans la première partie de la courbe l'ajout de processeurs diminue rapidement le temps de réponse. Dans la seconde partie, au delà d'un certain nombre de processeurs, l'ajout d'un processeur augmente le temps de réponse. Ceci est dû au fait que le temps de communication est supérieur au temps de traitement gagné,
- L'équilibrage de charge sur tous les processeurs. La difficulté est de trouver une fonction de répartition garantissant l'équi-répartition des données pour assurer une rentabilité maximale des capacités du système parallèle.

La suite de ce chapitre sera consacrée à la présentation des techniques de fragmentation. Il est à noter, que dans la littérature la plus part des travaux ne font pas de distinction entre fragmentation et partitionnement. Dans cette thèse nous ne ferons pas également de différence entre les deux appellations.

Nous présentons, dans ce qui suit l'intérêt de la fragmentation pour les entrepôts de donnée, puis nous détaillons les approches de fragmentation ainsi que les différentes contributions qui traitent de l'adaptation de la fragmentation dans les entrepôts de données. Nous concluons ce chapitre par une description détaillée de la problématique relative à l'application des techniques de fragmentation au contexte des entrepôts de données.

3.3 Motivations pour la fragmentation dans les entrepôts de données

Une implémentation réussite d'un entrepôt de données suppose une planification de la pise en charge de la croissance des volumes de données et la simplicité de leurs administrations. Il peut donc être implémenté à l'aide d'une approche par partition (distribuée) ou une approche centralisée. Le choix entre ces deux approches dépend, en grande partie, de certains facteurs à savoir :

- ✓ le volume des données ;
- ✓ le chargement des données;
- ✓ l'indexation ;
- ✓ stratégie d'expiration des données;

- ✓ stratégie d'archivage;
- ✓ performance des requêtes.

- **Le volume des données**

Lorsque la table des faits est de petite taille une approche de partitionnement s'avère inadaptée et rend la gestion des données plus complexes, sans apporter un plus par rapport à une implémentation centralisée. La taille de la table des faits dépend de la nature du contexte de l'application. A titre d'information, certains éditeurs recommandent que la taille de la table des faits devra être d'au moins 100 Go pour prévoir le partitionnement de données.

- **Chargement des données**

Les entrepôts sont périodiquement alimentés par de nouvelles données. La disponibilité de ces dernières dépend de l'efficacité du processus de chargement. L'approche centralisée entraîne une indisponibilité des données comparée à l'approche par partition, en raison du chargement des données. En effet, avec une table unique non partitionnée, aucun utilisateur ne pourra plus accéder à la table au cours du chargement des données. La solution optimale consiste à planifier et mettre en place un processus de maintenance basée sur un chargement incrémental des données.
En revanche, le partitionnement permet d'effectuer le chargement des données dans des tables intermédiaires, chacune représentant une plage de partitions déterminée. La table intermédiaire est ensuite ajoutée à la vue partitionnée ou basculée dans la table partitionnée en tant que nouvelle partition. Etant donné que chaque partition est logiquement représentée par une table intermédiaire, le chargement n'affectera pas la disponibilité et les performances des requêtes sur les données en utilisation.

- **Indexation**

L'amélioration du traitement des requêtes nécessite la création d'index. Les performances des requêtes sur les tables des faits sont généralement médiocres en l'absence d'index. Pour une table des faits centralisée, une solution optimale peut consister à supprimer tous les index, à charger les données et à recréer les index. Cette approche entraîne la réduction de la disponibilité et une maintenance plus difficile au fur et à mesure que la taille de la table des faits augmente. Le partitionnement apporte des solutions à ce type de problème, par la création d'index sur les tables sous-jacentes, ce qui permet de prendre en charge la recréation et la réorganisation des index sur les partitions, ce qui revient donc à gérer des index eux mêmes partitionnés.

- **Expiration des données**

Les données anciennes font l'objet d'accès moins fréquent que les données récentes. Devant le besoin de conserver les anciennes données et de préserver la haute disponibilité des données en cours d'utilisation, l'expiration des données peut être gérée efficacement à travers une approche de partitionnement (généralement un partitionnement basée sur le type date).

- **Archivage des données**

Si les données sont partitionnées, une sauvegarde par étape peut être implémentée. Les opérations de sauvegarde et de restauration par étape peuvent se faire sur les partitions individuellement sans affecter l'entrepôt de données en entier.

- **Performance de requêtes**

La prise en charge des requêtes, généralement de type ad-hoc, font partie des caractéristiques des entrepôts de données. Le partitionnement améliore d'une manière significative les performances lors de l'exécution des requêtes en favorisant l'exploitation du traitement parallèle des différentes opérations.

On conclut, de ce qui précède, que la conception et l'implémentation d'un entrepôt de données ne peuvent être envisagées sans les techniques de partitionnement. La complexité des requêtes OLAP et les différents processus relatifs au chargement, à l'administration et à l'archivage d'un volume important de données, nécessitent le recours à ce type d'optimisation.

Dans les sections qui suivent nous présentons plus en détaille les techniques de fragmentation, les algorithmes de conception des schémas de fragmentation ainsi que les différents travaux qui ont traité leur adaptation aux entrepôts de données.

3.4 Les techniques de fragmentation des données

Définition 3 : *La fragmentation ou partitionnement consiste à diviser un ensemble de données en plusieurs partitions, appelées fragments, de manière à ce que la combinaison des fragments recouvre l'intégralité des données sources sans ajout ni perte d'information* [MAH06].

Définition 4 : *Un schéma de fragmentation est le résultat d'un processus de fragmentation* [BEL00].

Définition 5 : *Une fragmentation est sans perte d'information si la base de données logique peut être entièrement recomposée à partir de ses fragments. Cette recomposition est exprimée en utilisant les instructions du langage de manipulation de données*

associé au modèle de données utilisé (par exemple l'algèbre ou SQL pour une BD relationnelle).

La technique de fragmentation des données a été proposée par Eswaran [ESK74], ce qui a donné par la suite naissance à divers approches de fragmentation, en l'occurrence l'approche verticale [NAS84], horizontale [CES82] et l'approche mixte [SAG86, [ZHY94].

3.4.1 La fragmentation horizontale

La fragmentation horizontale (FH), largement étudiée et implémentée dans les SGBD commerciaux, consiste à regroupe les tuples d'une relation, utilisés collectivement par les requêtes importantes. Un fragment horizontal s'obtient en spécifiant un prédicat qui applique une restriction sur les tuples de la relation. Etant donné une relation *R*, un fragment horizontal est défini par : $\sigma_p(R)$ où *p* est un prédicat établi sur un ou plusieurs attributs de la relation.

La FH constitue un aspect important dans la conception physique des bases de données [SAA04, PAP04]. Elle est considérée comme une technique d'optimisation non redondante du fait qu'elle ne réplique pas de données. Elle a un impact significatif sur la performance des requêtes définies sur des volumes de données importants. Elle a aussi un impact sur la facilité de gestion et la maintenance des données [BOK08].

La FH se décline en deux versions [BEL00, CES82] : la fragmentation horizontale primaire (FHP) et la fragmentation horizontale dérivée (FHD).

- La FHP, qui favorise le traitement des requêtes de restriction sur les attributs utilisés dans le processus de fragmentation, est effectuée grâce à des prédicats de sélection définis sur la relation [OZM99].
- La FHD, utile pour le traitement des requêtes de jointure, consiste à fragmenter une relation selon le schéma de fragmentation d'une autre relation [BEL00].

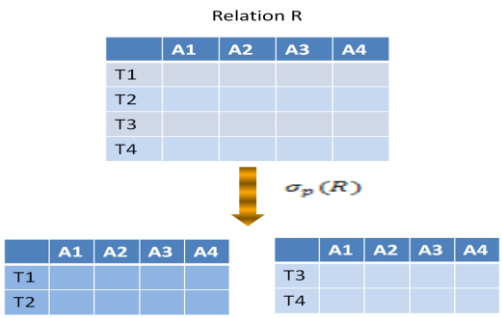


Figure 3.4- Fragmentation horizontale d'une relation

3.4.2 La fragmentation verticale

La fragmentation verticale (FV) consiste à regrouper des attributs d'une relation utilisés conjointement par les requêtes importantes. Un fragment vertical s'obtient à l'aide d'une opération de projection de l'algèbre relationnelle. Etant donné une relation *R* (*k*, a_1, ..., a_n), un fragment vertical est défini par : $\pi_{a1,\ldots,an}(R)$ où *K* est la clé et a_1, ..., a_n sont des attributs de la relation *R* [GEG005].

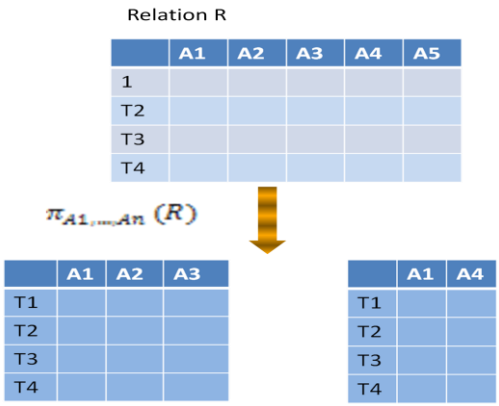


Figure 3.5- Fragmentation verticale d'une relation

La FV favorise le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de fragmentation, en limitant le nombre de fragment à accéder. Son principal inconvénient est qu'elle nécessite des jointures supplémentaires lorsqu' une requête à plusieurs fragments.

3.4.3 La fragmentation mixte

La fragmentation horizontale ou verticale d'un schéma de base de données ne suffit quelquefois pas pour distribuer adéquatement les données dans certaines applications, de sorte qu'il faut alors faire appel à la fragmentation mixte (FM) ou hybride. Un fragment mixte est constitué d'un fragment horizontal fragmenté ensuite verticalement ou d'un fragment vertical fragmenté ensuite horizontalement. Un fragment mixte est défini à l'aide d'opérations de sélection et de projection de l'algèbre relationnelle. Étant donné une relation *R*, un fragment mixte est défini par : $\sigma_p(\pi_{a1,\ldots,an}(R))$ ou $\pi_{a1,\ldots,an}(\sigma_p(R))$ où *p* est un prédicat établi sur un ou plusieurs attributs de *R* et a_1, ..., *an* sont des attributs de *R*.

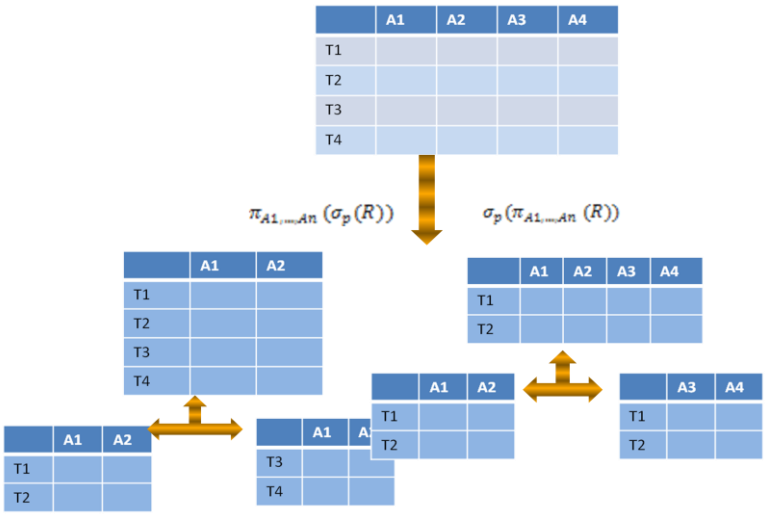


Figure 3.6- Fragmentation mixte d'une relation

3.4.4 Validité des approches de fragmentation des données

Pour s'assurer que les données n'ont pas été modifiées sémantiquement durant la fragmentation, ces approches doivent, pour être valide, respecter certaines règles à savoir :

- ***La complétion*** : si une instance de la relation *R* est décomposée en partition R_1, R_2, ……R_n, chaque donnée qui se trouve dans la relation *R* doit apparaître au moins dans l'une des partitions. Cette règle permet d'interdire toute perte de données pendant la répartition.
- ***La reconstruction*** : possibilité de définir une opération relationnelle permettant de reconstruire la relation *R* à partir de ses partitions. Cette règle garantit la préservation des dépendances fonctionnelles.
- ***La disjointure*** : si une donnée doit apparaître dans la partition R_i alors elle ne doit apparaître dans aucune autre partition. Cette règle garantit la redondance minimale des données.

3.5 Les algorithmes de conception d'un schéma de fragmentation optimal

La conception d'un schéma de fragmentation optimisant les performances dépend sur une analyse statistique des requêtes les plus fréquentes en se basant sur des informations tant qualitatives que quantitatives de l'utilisation des données.

Les informations quantitatives servent plus particulièrement dans la phase de l'allocation ou le placement des données, tandis que les informations qualitatives sont

utilisées pour la conception d'un schéma de fragmentation. Ces informations portent, notamment sur :

- les relations, attributs et tuples qui reçoivent un accès ;
- le type d'accès (en lecture ou en écriture) ;
- le nombre de requête d'écriture sur ces attributs ;
- le nombre de requête de lectures ;
- la fréquence d'exécution d'une requête ;
- la localisation de ses attributs sur les nœuds ou sites dans un environnement distribué;
- le site à partir du quel la requête est exécutée ;

3.5.1 Les algorithmes de conception d'un schéma de fragmentation vertical

La conception d'un schéma de fragmentation vertical est un problème difficile en raison du grand nombre d'alternatives possibles [BEL00]. Il s'agit de déterminer le nombre optimal de fragments maximisant les performances du système. Le nombre de solutions possibles est égale au nombre de Bell qui satisfait l'équation suivante :

$$B_{n+1} = \sum_{k=0}^{n} B_k \binom{n}{k}, \text{ Par exemple } B30 = 1023$$

Les travaux qui ont traité cette problématique se basent sur les travaux de Navathe et al., [NAB89, NAS84]. Ces travaux utilisent le principe de l'affinité entre attributs pour concevoir des fragments verticaux. Leur approche consiste à construire trois matrices, *1) la matrice d'usage des attributs* ; *2) la matrice des affinités d'attributs* ; et 3) *la matrice d'affinité ordonnée*.

1. Construction de la matrice d'usage des attributs

Soit un ensemble de requête $Q = \{q_1, q_2, \ldots, q_q\}$ qui s'exécutent sur la relation $R[A_1, A_2, \ldots, A_n]$.

$$use(q_i, A_j) = \begin{cases} 1 & si\ l'attribut\ A_j\ est\ accédé\ par\ q_i \\ 0 & \sin on \end{cases}$$

Cela signifie que la valeur de la colonne est 1 quant l'attribut *j* est accédé par la requête *i,* sinon cette valeur est nulle.

***Exemple 2* :** *définition d'une matrice d'usage*

q_1: `SELECT` BUDGET FROM PROJ `WHERE` PNO=Value

q_2: `SELECT` PNAME, BUDGET `FROM` PROJ

q_3: `SELECT` PNAME FROM PROJ `WHERE` LOC=Value

q_4: SELECT SUM(BUDGET) FROM PROJ WHERE LOC=Value

Let A_1= PNO, A_2= PNAME, A_3= BUDGET, A_4= LOC

La matrice d'usage correspondant à cet ensemble de requête est :

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

2. Construction de la matrice d'affinité d'attributs

La matrice d'affinité est constituée, en ligne et en colonne, des attributs de la relation à fragmenter. Les éléments de cette matrice reflètent l'affinité entre attributs. Cette affinité correspond à la somme des fréquences d'accès des requêtes accédant simultanément aux deux attributs.

L'affinité $aff(A_i, A_j)$ entre deux attributs A_i et A_j d'une relation $R[A_1, A_2,\ldots, A_n]$ sur laquelle s'exécute un ensemble de requête $Q = \{q_1, q_2,\ldots, q_q\}$est définie par :

$$aff(A_i, A_j) = \sum_{k \mid use(q_k, A_i)=1 \wedge use(q_k, A_j)=1} \sum ref_l(q_k) acc_l(q_k)$$

Avec : $ref_l(q_k)$ est le nombre des accès aux attributs (A_i, A_j) pour chaque exécution de la requête q_k. $acc_l(q_k)$ est le nombre d'exécution de la requête.

Le résultat de ce calcul est une matrice AA, appelé matrice d'affinité d'attributs, de $n \times n$ ou n est le nombre d'attributs.

***Exemple 3* :** *Calcul d'une matrice d'affinité d'attributs :*

Prenant la matrice d'usage d'attributs, définie dans l'exemple ci-dessus :

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

L'affinité d'attributs de cette matrice d'usage est :

$$aff(A_1, A_3) = \sum_{k=1}^{3} \sum_{l=1}^{3} acc_l(q_1) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

La matrice d'affinité d'attributs sera donc :

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{bmatrix} \end{array}$$

3. **Construction de la matrice d'affinité d'attributs ordonnée**

La dernière étape de cette approche consiste à regrouper et ordonner les attributs selon leurs affinités par l'application d'un algorithme de groupement tel que l'algorithme *BEA* (*Bond Energy Algorithm*) sur la matrice d'affinité d'attributs. Le résultat de de cette opération sera une matrice de groupement de mesures d'affinité *CA* (*clustered affinity matrix*) de *n* x *n* ou *n* est le nombre d'attributs. L'intersection entre la ligne *i* et la colonne *j* représente la fréquence d'accès entre *i* et *j*. L'algorithme *BEA*, par exemple, retrouve et réorganise les attributs de telle sorte que l'équation de mesure d'affinité globale, ci-après, est minimisé :

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) + aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j)]$$

Avec : $aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0$

A la fin la matrice *CA*, obtenue, aura une forme de blocks en diagonal où chaque block d'attributs est considéré comme un fragment.

Ci-dessous, une illustration de la conception d'un schéma de fragmentation par l'utilisation de l'algorithme *BEA*

Bond Energy Algorithm (BEA)

- *En entrée : la matrice d'affinité d'attributs AA*
- *En sortie : La matrice de groupement d'affinité CA*
- *Les étapes :*
    1) *Initialisation : placer et fixer une colonne de AA dans CA*
    2) *Itération : placer les n-i colonnes restantes dans les i+1 positions de la matrice CA. Pour chaque colonne, choisir l'emplacement qui fait plus de contribution à la mesure d'affinité globale.*
    3) *Ordonner-ligne : Ordonner les lignes conformément à la colonne d'ordonnancement.*

Algorithme 1. Algorithme BEA

Begin
{initialisation : AA est une matrice n x n }
CA(., 1) ← AA(., 1)
CA(., 2) ← AA(., 2)
index ← 3
While *index ≤ n do {choisir le "meilleur" placement pour l'attribut* AA_{index}*}*
begin
for *i* ***from*** *1* ***to*** *index-1* ***by*** *1* ***do***
calculer cont(A_{i-1}*,* A_{index}*,* A_i*)*
end-for
calculer cont($A_{index-1}$*,* A_{index}*,* $A_{index+1}$*) {condition limite}*
loc ← placement donnée par la valeur maximum de cont
for *j* ***from*** *index* ***to*** *loc* ***by*** *-1* ***do*** *{remanier les deux matrices}*
CA(., j) ← CA(., j-1)
end-for
CA(., loc) ← AA(., index)
index ← index + 1
end-while
ordonner les lignes conformément aux colonnes d'ordonnancement y relatives
End*.*

3.5.2 Les algorithmes de conception d'un schéma de fragmentation horizontal

La fragmentation horizontale a fait l'objet de plusieurs travaux de recherche. Elle est également utilisée par plusieurs outils et SGBD relationnel et objet dans des environnements centralisés et distribués. Cet intérêt est justifié par son apport en matière d'optimisation de performance que se soit en termes de temps de traitement des requêtes, de gestion et de maintenance des données.

Pour une entité donnée, plusieurs schémas de fragmentation peuvent être conçus selon les prédicats utilisés pour générer des fragments horizontaux. La plus part des travaux qui ont traité cette problématique, se basent sur la fragmentation horizontale primaire.

Pour une relation $R\ [A_1, A_2, \ldots, A_n]$ et une charge de requête $Q = \{q_1, q_2, \ldots, q_q\}$ ou chaque q_k possède une fréquence d'accès $acc(q_k)$. La conception d'un schéma de fragmentation horizontal consiste à partitionner R en f fragments horizontaux $R_1, R_2, \ldots, R_f$ tel que le coût d'exécution de Q sur la table fragmentée soit minimal.

La conception d'un schéma de fragmentation horizontal optimal nécessite des informations qualitatives et des informations quantitatives. Ces informations portent sur :

- le schéma conceptuel global particulièrement sur la manière dont les tables sont jointes;
- la cardinalité de la relation *Card(R) ;*
- les prédicats : les prédicats sont soit simples, c'est-à-dire qu'ils impliquent des attributs simples, soit pertinent ou complexes, faisant appel à plusieurs attributs. Les prédicats sur les attributs peuvent être à valeur unique ou multivalués, auquel cas, les valeurs peuvent être discrètes ou s'inscrivent dans des intervalles de valeurs ;
- les minterms et leur sélectivité des minterms : $sel(m_i)$;
- la fréquence d'accès de la requête : $acc(q_i)$

Définition 6 : *Soit une relation R* $[A_1, A_2, \ldots, A_n]$*, un prédicat simple p est de la forme : P :* a_i θ *Valeur ; avec* $\theta \in \{=, <, >, \leq, \geq, \neq\}$ *Valeur* $\in$ D_i *, avec* D_i est le *Domaine de l'attribut* A_i.

Définition 7 : *Soient P un ensemble de prédicat simple. Un prédicat p est pertinent par rapport à l'ensemble P s'il existe un prédicat* $p' \in P$ *tel que les fragments horizontaux définis par* $(p \wedge p')$ *et* $(p \wedge \neg p')$ *sont accédés individuellement par au moins une requête.*

Définition 8 : *un ensemble de prédicat est dit minimal s'il ne contient que des prédicats qui lui sont pertinents.*

Définition 9 : *Les minterms : soit une relation R et un ensemble de prédicat* $P_r=\{p_1, p_2, \ldots, p_m\}$*, l'ensemble des minterms* $M=\{m_1, m_2, \ldots, m_r\}$ *est généré par :*

$$M = \{ m_{ij} \mid m_{ij} = \bigwedge_{p_{ik} \in \Pr_i} p_{ik}^* \}, 1 \leq k \leq m, 1 \leq j \leq z$$

Avec : $p_{ik}^* = p_{ik}$ or $p_{ik}^* = \neg p_{ik}$.

Les algorithmes de sélection d'un schéma de fragmentation horizontal, développés dans le contexte relationnelle et objet, se déclinent en trois catégories : *1) les algorithmes basés sur la minimalité et la complétudes des prédicats* [CES82, OZM91] ; 2) *les algorithmes dirigé par l'affinité des prédicats* [NAB89] ; *et 3) les algorithmes à base de modèle de coût* [BEL00],[BOK08]. Ces algorithmes partagent les mêmes entrées c'est-à-dire, qu'ils partent d'un ensemble de requêtes et leur fréquence d'accès. Ces requêtes respectent la règle de 80/20, qui signifie que 20% des requêtes produisent 80% des accès aux données.

3.5.2.1 Algorithmes dirigés par la minimalité et la complétude des prédicats

L'approche basée sur la construction des prédicats consiste à partitionner une relation grâce à un ensemble complet et minimal de prédicats construit à partir de l'utilisation de l'algorithme COM-MIN [NOA99]. La complétude signifie que deux instances d'une relation dans un même fragment ont la même probabilité d'être accédés séparément. La minimalité garantit qu'il n'existe pas de redondance dans les prédicats. Cette approche s'effectue selon les étapes ci-après :

1- Génération d'un ensemble complet et minimal de prédicats,

L'algorithme de construction de prédicats complet et minimal (COM-MIN) à partir d'un ensemble de prédicats simple se déroule en deux étapes.

- Dans la première étape, il sélectionne un prédicat complet d'un ensemble de prédicats *P* et l'affecte à un autre ensemble *P'* ;
- Dans la seconde étape, l'algorithme procède de manière itérative. A chaque itération, il affecte à *P'* un prédicat complet de *P* pertinent par rapport aux prédicats présents dans l'ensemble *P'*. *P'* est l'ensemble complet et minimal de prédicats. L'algorithme s'arrête quand *P* est vide.

❖ **Algorithme COM-MIN :**

- *Entrée : une relation R et un ensemble de prédicats simples P ;*
- *Sortie : un ensemble de prédicats complet et minimal Pr' pour Pr ;*
- *Règle 1 : Une relation ou fragment est fragmenté en moins en deux partitions qui sont accédées différemment par en moins une requête ;*
- *Etapes :*
  1) *Initialisation;*
  2) *Chercher $p_i \in Pr$ tel que p_i fragmente la relation R selon la règle 1 ;*
     Mettre $Pr' = p_i$, $Pr \leftarrow Pr - p_i$; $F \leftarrow f_i$
     Si $\exists p_k \in Pr'$ qui n'est pas pertinent alors $Pr' \leftarrow Pr' - p_k$; $F \leftarrow F - f_k$

Algorithme 2. Description de l'algorithme COM-MIN

```
Declare
    F: set of minterm fragments
Begin
    find a p_i ∈ Pr such that p_i partitions R according to Rule 1
    Pr' = p_i ;
    Pr ← Pr – p_i;
    F ← f_i  {f_i is the minterm fragment according to p_i}
    do
    begin
        find a p_j ∈ Pr such that p_j partitions some f_k of Pr' according to Rule 1
        Pr' = Pr' ∪ p_j ;
```

```
                Pr ← Pr – pj;
                F ← F ∪ fj
                if ∃pk ∈ Pr' which is nonrelevant then
                begin
                        Pr' ← Pr' – pk
                        F ← F – fk
                end-if
        end-begin
        until Pr' is complete
End
```

2- Génération des minterms : selon la formule présentée dans la définition 7,ci-dessus .

3- Suppression des minterms insignifiant

L'ensemble des minterms *M* est réduit en éliminant les minterms contradictoires par rapport à un ensemble d'implications entre les prédicats.

***4*- Génération des fragments horizontaux**: chaque minterm m_i restant dans l'ensemble *M* permet de générer un fragment R_i de la table *R* comme suit : $R_i = (\sigma_{mi}(R))$. Selon l'algorithme 3, décrit ci-dessous.

Algorithme 3. Algorithme de génération des fragments horizontaux

```
Begin
        Pr' ← COM_MIN (R, Pr)
        determine the set M of minterm predicates
        determine the set I of implications among pi ∈ Pr'
        eliminate the contradictory minterms from M
        for each mi ∈ M do
                if mi is contradictory according to I then
                        M ← M – mi
                end-if
        end-for
End
```

3.5.2.2 Algorithmes dirigés par l'affinité des prédicats

Pour réduire la complexité de l'approche de conception d'un schéma de fragmentation horizontal basé sur les prédicats, les travaux développés dans le cadre de la fragmentation verticale, ceux de [NAB89] relatifs à l'utilisation de l'affinité entre attributs, ont été adaptés au contexte de la fragmentation horizontale.

Le principe de l'approche dirigée par l'affinité des prédicats considère les mêmes matrices que celle construites dans le cadre d'une fragmentation verticale (voir la section 3.5.1, ci-dessus), à savoir : la matrice d'usage des prédicats, la matrice d'affinité des prédicats et la matrice d'affinité ordonnée.

Cependant, pour la génération des fragments horizontaux, Zhang et al., [ZHY94] proposent de construire les fragments à partir des semi-blocs diagonaux générés après l'application de la l'algorithme *BEA*. La construction d'un fragment à partir d'un semi bloc consiste à lier tous les prédicats simples comprenant le même attribut par l'opérateur logique *OU* et les prédicats avec des attributs distincts avec l'opérateur logique *ET*. Ils définissent, ensuite, le fragment résiduel en construisant la négation de la disjonction des différents prédicats de fragmentation.

La complexité de cet algorithme est : $O(n \times N + N^2)$ [BEL00] où n et N représentent, respectivement, le nombre de requêtes et le nombre de prédicats simples contenus dans ces requêtes. Cette approche est donc moins complexe que celle basée sur les prédicats dont la complexité est $O(2^n)$. L'inconvénient de cette approche c'est qu'elle ne permet pas de contrôler le nombre de fragments finaux générés.

3.5.2.3 Algorithmes à base de modèle de coût

Les algorithmes, présentés dans les sections ci-dessus, ne permettent pas d'évaluer le schéma de fragmentation. Dans ce contexte, Bellatreche et Boukhalfa [BEL00, BOK08] proposent une stratégie de sélection qui emploie un modèle de coût afin d'évaluer chaque schéma de fragmentation généré. Elle est réalisée en trois étapes :

- *Phase de génération des schémas de fragmentation* : à partir de la charge des requêtes, les prédicats de sélection sont extraits. Ils permettent d'identifier les différents fragments à travers les minterms. Un générateur est conçu pour produire tous les schémas de fragmentation possibles.
- *Phase d'évaluation* : se base sur un modèle de coût, elle permet d'évaluer le coût d'exécution des requêtes sur chaque schéma de fragmentation généré.
- *Phase de sélection du schéma optimal* : permet de déterminer le schéma de fragmentation le plus bénéfique pour l'exécution de la charge des requêtes.

Pour ce faire, le premier algorithme proposé est exhaustif. Il consiste à construire tous les schémas de fragmentation possibles par la fragmentation horizontale. Il énumère ensuite ces schémas et calcule pour chacun d'entre eux le coût d'exécution des requêtes de la charge. Il sélectionne finalement le schéma qui correspond au coût minimum. Le deuxième algorithme est approximatif. Il construit un schéma initial par l'algorithme de fragmentation dirigé par les affinités, puis l'améliore par des opérations de fusion ou de décomposition des fragments. Et enfin, le troisième algorithme exploite un algorithme génétique pour sélectionner le schéma de fragmentation optimal.

3.5.3 Les algorithmes de conception d'un schéma de fragmentation mixte

La fragmentation mixte consiste à combiner la fragmentation horizontale et verticale. L'approche la plus utilisée pour concevoir un schéma de fragmentation mixte est l'approche par création de grille [KAK96]. Elle consiste à fragmenter horizontalement et verticalement une relation puis à construire une grille sur la base des résultats de ces deux fragmentations. Chaque cellule de cette grille correspond un fragment mixte définie par un prédicat se sélection de la fragmentation horizontale et un prédicat de projection de la fragmentation verticale.

3.6 Techniques de fragmentation implémentées dans les SGBDs Commercialisés

3.6.1 Microsoft SQL Server

Le concept de partitionnement sur SQL Server n'est pas nouveau. Chaque version du SGBD (version 6, 7, 2000 et 2005) autorise des formes de partitionnement. Microsoft SQL Server prend en charge le partitionnement des données par l'intermédiaire de vues partitionnées. Dans SQL Server 2000, la fonctionnalité a été améliorée afin de prendre en charge les vues partitionnées pouvant être mises à jour. Une vue partitionnée est particulièrement adaptée lorsqu'une table peut être naturellement partitionnée en tables distinctes par plages de données. Les tables sous-jacentes de la vue partitionnée font l'objet d'une *union* afin de présenter un ensemble de données unifié. Les vues partitionnées réduisent considérablement la complexité des applications, car l'implémentation physique est déduite des méthodes d'accès aux données par l'application. De plus, les vues partitionnées peuvent être étendues afin d'inclure des vues partitionnées distribuées, ce qui permet la fédération des bases de données sur plusieurs serveurs/instances.

Dans SQL Server 2005, les fonctions de partitionnement des tables et index offrent flexibilité et performances et simplifient la création et la maintenance de telles tables. SQL Server 2005 permet le partitionnement horizontal par intervalle avec la ligne de données comme plus petite unité de partitionnement. Les partitions par intervalle sont des partitions de table définies par des plages personnalisables de données. L'utilisateur définit la fonction de partition avec des valeurs limites, un schéma de partition avec mise en correspondance des groupes de fichiers, ainsi que des tables mises en correspondance avec le schéma de partition. Une fonction de partition détermine à quelle partition appartient une ligne particulière d'une table ou d'un index. Chaque partition définie par une fonction de partition est mise en correspondance avec un emplacement de stockage (groupe de fichiers) via un schéma de partition.

3.6.2 Oracle

Toutes les versions d'Oracle (depuis la version 8i, ou il intègre l'outil de développement des entrepôts de données « *Warehouse Builder* ») intègrent des méthodes de partitionnement, chacune est adaptée à un contexte d'utilisation particulier. Le SGDB Oracle utilise les méthodes de partitionnement suivantes :

- *Le partitionnement par intervalle* (range partitioning) qui consiste à partitionner les tables (les vues ou les index) par des intervalles de valeurs pour la clé (la clé de partitionnement est généralement un champ de type date). Cette méthode est particulièrement adaptée pour les applications conservant des données historiques, lors de la phase de chargement de nouvelles données ou lors de l'archivage des anciennes données. Elle facilite également la gestion des partitions (ajout et suppression de partition, simplifie la mise à jour d'une table volumineuse)
- *Le partitionnement par hachage* utilise une fonction de hachage pour partitionner les données. Il distribue équitablement les données sur les différentes partitions.
- *Partitionnement composé ou hybride*, dans lequel les données sont partitionnées par intervalle puis sont subdivisées par l'utilisation d'une fonction de hachage. Ce type est généralement adapté pour découper une très grosse table de faits sur un environnement parallèle.
- *Le partitionnement par liste (by List)* permet de regrouper les données en fonction d'une liste de valeurs spécifiques sur les enregistrements d'une dimension.
- Un autre type de partitionnement intégré dans la version 10g, est le partitionnement orienté jointure, qui consiste à diviser une opération de jointure, en séquences plus petites exécutées en parallèle.

Dans la version 10 g d'Oracle, une nouvelle méthode a été introduite en plus de méthodes décrites ci-dessus, il s'agit de la méthode des tables organisées en index (*Index Organized Tables, IOTs*). Cette méthode combine les caractéristiques d'une table traditionnelle avec un accès rapide à un index dans un seul segment.

3.7 Application de la fragmentation aux entrepôts de données

L'application de la fragmentation aux entrepôts de données est d'un apport considérable en terme d'optimisation des performances plus particulièrement lors de l'exécution des requêtes décisionnelles en évitant le balayage des grandes tables. A ce titre les techniques de fragmentation plus précisément horizontale et verticale ont été

adaptées pour les entrepôts de données. Deux cas d'application de la fragmentation dans les entrepôts de données ont été considérés :

- Fragmentation de la table des faits selon une approche verticale. la reconstruction de la table s'effectue à travers une opération de jointure ;
- Fragmentation de la table des faits selon une approche horizontale dérivée. C'est-à-dire de la table des faits est fragmentée en fonction des fragments générés par la fragmentation d'une ou de plusieurs tables de dimension.

La fragmentions horizontale est considéré comme étant la plus adéquate compte tenu des gains de performances obtenus lors de l'exécution des opérations coûteuses en temps et en ressources, en l'occurrence les opérations de jointures en étoile entre la table des faits et les tables de dimension [BEL00]. L'application de la fragmentation horizontale (primaire et dérivée) pour les entrepôts de données peut être réalisée selon les scénarii, ci-après :

1. Fragmenter uniquement les tables de dimension selon une approche horizontale primaire. Ce scénario n'apporte pas des améliorations significatives étant donné qu'il n'intègre pas la table de faits, considéré généralement comme étant très volumineuse comparé aux tables de dimension. De plus L'avantage de fragmenter l'entrepôt selon ce scénario réside dans la possibilité d'optimiser les opérations de sélection définies sur les tables de dimension. L'inconvénient de ce scénario réside dans l'incapacité d'optimiser les opérations de jointures entre la table des faits et les tables de dimension. ;

2. Fragmenter uniquement la table des faits selon une approche horizontale primaire en utilisant un de ces attributs comme attribut de fragmentation. Ce scénario n'optimise ni les opérations de sélection ni les opérations de jointure car tous les fragments de la table des faits seront joints avec les tables de dimension pour pouvoir exécuter ces requêtes ;

3. Fragmenter et la table des faits et les tables de dimension selon une approche horizontale primaire. Dans ce scénario les opérations de jointure ne seront pas optimisées étant donné que la fragmentation primaire de la table des faits ne prend pas en considération ses liens avec les tables de dimensions ;

4. Fragmenter la table de faits par une horizontale dérivée après une fragmentation horizontale primaire d'une ou de plusieurs tables de dimension. Ce dernier scénario est le plus approprié pour les entrepôts de données, il per-

met d'améliorer les opérations de sélection sur les dimensions et les opérations de jointure entre faits et dimension.

Le tableau, ci-dessus, présente les avantages et les inconvénients de ces scénarii [BOK09].

Scénario	Avantages	Inconvénients
Scénario 1	Sélection optimisée	Jointure non optimisée
Scénario 2		Sélection non optimisée Jointure non optimisée
Scénario 3	Sélection optimisée	Jointure non optimisée
Scénario 4	Sélection optimisée Jointure optimisée	Nombre de fragments élevé ; Difficulté de réaliser une fragmentation dérivée avec plusieurs tables

Tableau 3.1- Avantages et inconvénients des scénarii de fragmentation

Le scénario 4 relative à la fragmentation de la table des faits, selon une approche dérivée après fragmentation des tables de dimension, est considéré comme étant le plus adapté aux requêtes décisionnelles. Ce scénario présente de meilleures performances lors de l'exécution des opérations de jointure et de sélection.

Néanmoins, le principal problème qui entrave l'application de ce scénario réside dans le nombre important de fragments qui seront générés. La réussite de la mise en place d'une approche de fragmentation horizontale dérivée dans les entrepôts de données consiste à déterminer un schéma de fragmentation satisfaisant le compromis entre le temps d'exécution des requêtes et le nombre de fragments à générer. Beaucoup de travaux ont traité cette problématique en proposant l'utilisation des métaheuristiques. Cet aspect sera détaillé dans le chapitre 4, dans lequel nous proposons notre approche pour la sélection d'un schéma de fragmentation optimal.

3.8 Conclusion

Dans ce chapitre nous avons présenté les approches de fragmentation des données et les algorithmes utilisés pour concevoir des schémas de fragmentation optimaux.

Il en ressort, que ces techniques d'optimisation sont d'un apport considérable pour l'exploitation des entrepôts de données, qui sont caractérisés, pour rappel, par un volume très important et des requêtes décisionnelles très complexes. Le recours, à cet ef-

fet, à la fragmentation des données pour exploiter, gérer et maintenir les entrepôts est devenu une nécessité que les administrateurs ne peuvent s'en passé.

Dans le contexte des entrepôts de données, la fragmentation horizontale primaire favorise le traitement des requêtes de restriction portant sur les attributs utilisés par le processus de fragmentation. Quant à la fragmentation horizontale dérivée est utile pour le traitement des requêtes de jointure. La fragmentation horizontale est considérée comme étant la plus adaptée en raison des avantages qu'elle offre en matière d'amélioration des performances, en permettant surtout l'exécution parallèle des requêtes OLAP. Elle facilite également la gestion de l'entrepôt en permettant de gérer un fragment à la fois et d'utiliser toutes les opérations usuelles (définies sur les tables globales) au niveau de la partition. En outre, la plupart des SGBD commerciaux implémentent la fragmentation horizontale et offrent des DDL pour fragmenter les objets de l'entrepôt de données et manipuler les partitions obtenues.

S'agissant de l'approche verticale, qui est considérée dans certains cas comme inappropriée aux entrepôts de données en raison des caractéristiques des requêtes analytiques, dans lesquelles l'opération de jointure est très utilisée. Ceci est justifié par le fait que le partitionnement vertical, qui consiste à diviser l'entrepôt en partitions de schémas différents par projection en dupliquant la clé, nécessite pour la recomposition de la table initiale à utiliser des opérations de jointure sur la clé, ce qui influe sur le traitement des requêtes OLAP (par la double utilisation de l'opérateur de jointure). De plus, la principale caractéristique de l'entrepôt est que les données sont dénormalisées, or ce type de partitionnement prolonge la normalisation en permettant d'éloigner d'une relation tous les attributs peu fréquemment utilisés. En résumé, la fragmentation verticale favorise naturellement le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de fragmentation, en limitant le nombre de fragments à accéder. Son inconvénient et qu'elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments.

En ce qui concerne les stratégies de conception d'un schéma de fragmentation optimal, beaucoup de travaux ont été réalisés pour parvenir à la mise en place d'approches permettant la sélection des schémas de fragmentation les plus adaptés aux entrepôts de données. Tel qu'il a été présenté dans le présent chapitre, les techniques développées, dans ce contexte, pour la sélection d'un schéma de fragmentation optimal, se basent sur des algorithmes basés soit sur la minimalité et la complétude des prédicats ou sur l'affinité des prédicats ou les attributs. Leurs principes consistent à générer des fragments en procédant au regroupement des prédicats ou attributs selon les fréquences d'accès des requêtes les plus fréquentes.

Cependant, dans le contexte des entrepôts de données, ces algorithmes deviennent parfois inadaptés voir impraticable [BEL00]. Ils sont statiques et ils se basent dans leurs entrées sur des informations quantitatives (la sélection des prédicats et la fréquence d'accès des requêtes) et qualitatives (le schéma globale de la base de données) préalablement définies. Si un changement intervient dans les entrées de ces algorithmes, ces derniers doivent être réexécutés afin de déterminer un nouveau schéma de fragmentation. Ainsi, nous pouvons constater qu'en cas d'évolution des modèles et/ou des changements de la charge de travail ces algorithmes deviennent très complexes et peuvent générer des schémas de fragmentation qui ne seront pas toujours les plus optimaux.

Le problème, qui peut être également relevé pour ces algorithmes, c'est l'utilisation des fréquences d'accès des requêtes comme seul critère de regroupement des prédicats ou des attributs pour générer, respectivement, des fragments horizontaux ou verticaux. Or, dans les entrepôts de données, il existe d'autres paramètres qui sont également très importants et méritent d'être considérés, à savoir : la taille des tables à fragmenter, la taille des vues matérialisées, la tailles des index ainsi que des paramètres physiques, taille de la mémoire centrale par exemple.

De plus, une conception d'un schéma de fragmentation selon les fréquences d'accès des requêtes se trouve parfois inadaptée aux entrepôts de données en raison des caractéristiques spécifiques des requêtes OLAP. Ces requêtes sont longues, complexes et nécessitent parfois un grand nombre d'opérations de sélection, d'agrégation. Elles peuvent manipuler des centaines voire des milliers de tuples. Les requêtes analytiques sont extrêmement variables, elles sont généralement composées d'une manière interactive et peuvent être exécutées une ou plusieurs fois. Il serait donc intéressant de développer ou d'aménager des algorithmes pour tenir compte de l'évolution des requêtes tant dans leurs structures que dans leurs fréquences.

Dans le cadre des bases de données relationnelles ou objets et quelque soit l'environnement (centralisé, parallèle, réparti) une grande partie de la littérature a traité cette problématique. Les travaux se sont focalisés sur les techniques de redistribution des données ou de réallocation des fragments en cas de détérioration des performances. Dans ce contexte, on a considéré que la solution réside au niveau physique en appliquant des stratégies d'équilibrage de charge des traitements et des données entres les sites. Le volet logique, à savoir la conception de l'approche de fragmentation, demeure adapté étant donné que la charge de travail est pratiquement stable.

Dans le même contexte, les SGBDs actuels (Oracle, SqlServer, DB2) fournissent uniquement une fragmentation statique des données. Si une table monte en échelle, l'administrateur de la base de données doit redéfinir manuellement le schéma de fragmentation et exécute des utilitaires de redistribution des données. Ceci est devenu une

préoccupation croissante, particulièrement, pour les utilisateurs des bases de données parallèles [GAG05].

Par ailleurs, en matière d'estimation de la qualité du schéma de fragmentation, les travaux qui ont abordé cette problématique mesurent la pertinence d'un algorithme par rapport à un autre algorithme, de sélection d'un schéma de fragmentation, selon une fonction de coût à minimiser. Cette fonction porte généralement sur le coût d'exécution des requêtes. Il n'existe pas à priori une approche qui évalue et mesure la pertinence du schéma de fragmentation déjà implémenté et selon de nouvelles informations d'exploitation de la base de données.

En résumé trois principaux problèmes, relatifs à l'application de la fragmentation aux entrepôts de données, ont été énumérés dans ce chapitre, à savoir :

1. Le problème de la NP-complétude de la sélection d'un schéma de fragmentation optimal ;
2. L'évaluation de la pertinence des schémas de fragmentation ;
3. L'impact de l'évolution des données et la « dynamicité » des requêtes OLAP sur le schéma de fragmentation.

En conclusion, dans ce chapitre nous avons voulu montré l'impact et l'importance que revêt la fragmentation dans le contexte des entrepôts de données, en présentant les techniques utilisées, les travaux qui ont traité ce domaine et surtout nous avons voulu mettre en exergue les problèmes qui entravent l'application de la fragmentation aux entrepôts de données et par conséquent la nécessité de rechercher de nouvelles stratégies qui prennent en considération, notamment, les caractéristiques des requêtes OLAP pour concevoir un schéma de fragmentation optimal, adapté à l'évolution des entrepôts de données.

Dans la seconde partie, nous allons présenter nos contributions qui traitent les problèmes, supra-cités, relatifs à l'adaptation de la fragmentation aux entrepôts de données. Ces contributions portent : 1) sur la proposition d'une approche basée sur l'optimisation par essaim particulaires pour la sélection d'un schéma de fragmentation optimal ; 2) une approche formelle pour l'évaluation d'un schéma de fragmentation ; et 3) l'utilisation des histogrammes pour la réfragmentation des données.

PARTIE II : CONTRIBUTIONS

Cette partie sera consacrée à la présentation de nos contributions. Elle comporte trois chapitres. L'approche portant utilisation de l'Optimisation par Essaim Particulaires, pour la conception d'un schéma de fragmentation optimale, sera détaillée dans le chapitre 4. l'approche permettant l'évaluation d'un schéma de fragmentation sera présentée dans le chapitre 5. Enfin, la solution au problème du changement de la charge de travail et son impact sur la fragmentation des données, sera décrite dans le chapitre 5, dans lequel, nous décrivons notre proposition relative à la fragmentation dynamique des données à travers l'implémentation des histogrammes pour la sauvegarde de la sélectivité aux données.

Chapitre 4

Optimisation par Essaim de Particules pour la conception d'un schéma de fragmentation optimal

4.1 Introduction

Les approches de conception d'un schéma de fragmentation optimal se concentrent sur le regroupement des prédicats ou des attributs selon leurs pertinences, indépendamment des fragments générés. Elles font partie des problèmes NP-complet [BOK09]. Si on considère *n* simple prédicats pour concevoir une approche de fragmentation horizontale primaire, 2^n est le nombre de fragments horizontaux utilisant des prédicats minterms. Pour un système distribué constitué de *k* nœuds, la complexité pour le placement de ces fragments horizontaux est $O(k2^n)$.

La fragmentation horizontale dérivée, considérée comme la plus adaptée aux entrepôts de données, peut augmenter le nombre de fragments des tables de faits de façon spectaculaire et rend leur maintenance très coûteuse. Soit , à titre d'exemple, un schéma en étoile avec *d* tables de dimension reliées à une table des faits. Soit g $(g \leq d)$ le nombre de tables de dimension fragmentées. Le nombre de fragments horizontaux de la table des faits (noté *N*) est donnée par : $N = \prod_{i=1}^{g} m_i$ avec m_i le nombre de fragments.

Dans le cas d'une fragmentation verticale, une relation de *m* attributs peut être fragmentée en $B(m)$ manières [OZM91]. $B(m)$ étant le nombre de Bell. Pour un grand nombre d'attributs $B(m)$ est approximativement m^m.

La sélection d'un schéma de fragmentation optimal consiste, donc, à trouver un compromis entre le coût de maintenance et le coût d'exécution des requêtes. Comme indiqué, ci-dessus, ceci relève des problèmes combinatoires NP-complet. Cela signifie, qu'il n'existe pas vraisemblablement un algorithme capable de les résoudre d'une manière exacte en un temps polynomial par rapport à la taille des données : le temps des résolutions risque d'être exponentiel par rapport à la taille des instances.

Dans les sections suivantes nous présentons, succinctement, les notions relatives aux problèmes d'optimisation et les différents algorithmes proposés pour les résoudre ainsi que certains travaux qui ont traité le problème de la NP-complétude de la conception d'un schéma de fragmentation optimal.

4.2 L'optimisation combinatoire

Définition 9 : *Un problème d'optimisation combinatoire est généralement caractérisé par un ensemble fini de solutions admissibles* Ω *et une fonction objective* $f : \Omega \rightarrow R$ *associant une valeur à chaque solution admissible. La résolution du problème consiste à déterminer la (ou les) solution(s) de* Ω *minimisant ou maximisant f.*

Il s'agit donc de trouver les paramètres à donner à cette fonction telle que la valeur retournée par celle-ci soit optimale (minimale ou maximale). Des contraintes peuvent être imposées pour la recherche de la meilleure solution (espace de recherche, par exemple). On peut diviser l'optimisation en deux domaines, selon que le problème soit discret (optimisation combinatoire) ou continu. On pourra aussi trouver des problèmes mixtes, où certaines variables seront discrètes, d'autres continues.

Pour résoudre ce type de problèmes, il est parfois nécessaire de renoncer à l'obtention des réponses exactes. Ceci, a conduit à développer des algorithmes approchés exploitant le non déterminisme et faisant le compromis entre une qualité et une vitesse de calcul acceptable. Il y a trois manières d'atteindre cet objectif [CAA01] :

- Construire un algorithme d'approximation qui retourne un résultat dont on sait qu'il est contenu dans un certain intervalle autour de l'optimum ;
- Construire un algorithme probabiliste, qui garantit que la probabilité d'obtenir un mauvais résultat est petite si la taille des instances est assez grande ;
- Admettre un algorithme qui n'offre aucune garantie théorique dans le cas général mais qui est réputé, sur base expérimentale, donner une moyenne de bons résultats. On parle dans ce cas d'heuristiques et de métaheuristiques.

4.2.1 Heuristiques et métaheuritiques

Définition 10 : *Une heuristique est une solution qui exploite les propriétés structurelles d'une solution admissible, de façon à ce quelle devienne rapidement une « bonne » solution admissible (en terme de coût). Elle correspond à un mécanisme de l'exploration de l'espace de recherche.* On distingue :

1. *Les heuristiques constructives*, consistent à trier les composantes du problème par ordre de coût et à les combiner de façon itérative et conditionnelle pour obtenir la solution finale ;
2. *Les heuristiques à recherche local*, consistent à définir un voisinage, c'est-à-dire d'associer à chaque solution un ensemble de solutions « voisines », puis d'améliorer au fur et à mesure une solution donnée initialement en rempla-

çant progressivement la solution dont on dispose par une meilleure solution située dans son voisinage.

Définition 11 : *Une métaheuristique est une stratégie d'exploration de l'espace des solutions. Elle définit la manière dont est utilisé une heuristique. C'est la méta-heuristique qui est chargé de guider l'heuristique vers des zones prometteuses de l'espace de recherche. L'objectif est de trouver des solutions dont la qualité est au-delà de ce qu'il aurait été possible de réaliser avec une simple heuristique.*

On distingue également les métaheuristiques constructives et les métaheuristiques à recherche local. La plupart des métaheuristiques actuelles appartiennent à la seconde classe.

Pour résumer, nous pouvons dire que la métaheuristique est à l'heuristique ce que la stratégie est à la tactique [ANE98]. La frontière entre ces deux termes est toutefois floue.

De plus, de part leur variété et leur capacité à résoudre des problèmes très divers, les métaheuristiques sont assez facilement sujettes à extensions. Parmi celles-ci, on peut citer :

- Les métaheuristiques pour l'optimisation multiobjectif. Il ne s'agit pas ici de trouver un optimum global mais de trouver un ensemble d'optima qui forment une surface de compromis pour les différents objectifs du problème ;
- Les métaheuristiques pour l'optimisation multimodale [GOD87], où l'on ne cherche plus l'optimum global, mais l'ensemble des meilleurs optima locaux ;
- Les métaheuristiques pour l'optimisation dynamique [BRJ01, OUC02, DIG98], où il faut approcher l'optimum à chaque pas de temps, car la fonction objectif change de topologie au cours du temps ;
- Les métaheuristiques hybrides [TAE02], qui combinent différentes métaheuristiques, afin d'en tirer les avantages respectifs ;
- Les métaheuristiques parallèles [ALE05], pour lesquelles on cherche à accélérer le calcul, en distribuant la charge de calcul sur plusieurs calculateurs.

Les métaheuristiques ont également comme caractéristiques communes leur caractère plus ou moins stochastique, ainsi que leur inspiration par une analogie avec d'autres sciences (physique, biologie, etc.) [EIA03].

Le domaine des métaheuristiques est un domaine en constante évolution. De nombreuses méthodes sont proposées chaque année pour améliorer la résolution des problèmes les plus complexes. Du fait de cette activité permanente, un grand nombre de classes de métaheuristiques existe actuellement. Parmi lesquelles, on peut citer :

- Les algorithmes génétiques [HOJ73] ;
- La recherche avec tabou [GLF97] ;
- Le recuit simulé [KIS83] ;
- Les systèmes de fourmis [COA92].

Ces métaheuristiques ont fait l'objet de plusieurs travaux de recherches et d'application pour la résolution des problèmes d'optimisation combinatoire [ALE05, DRJ03, BLC05, CAE06].

4.2.2 Utilisation des métaheuristiques pour la conception d'un schéma de fragmentation optimal

Dans le contexte des entrepôts de données, beaucoup de travaux ont proposé l'utilisation des métaheuristiques pour la conception d'un schéma de fragmentation optimale. Nous citons ci-dessous, quelques travaux :

Travaux de Golfareli et al., (1999) les auteurs considèrent que la fragmentation verticale permet de réduire le temps de réponse d'une requête globale en optimisant les requêtes nécessitant un sous ensemble de mesures de la table des faits. Ils formalisent le problème de la fragmentation verticale comme un problème d'optimisation linéaire en entier et proposent une approche permettant de minimiser une fonction coût préalablement définie afin de déterminer un schéma de fragmentation optimal des vues matérialisées [GOM99]. Dans le même contexte, Maniezzo et al., utilisent un algorithme d'optimisation par colonies de fourmis artificielles (ACO) pour optimiser le problème de la fragmentation verticale [MAV01].

Travaux de Song et Gorla (2000) utilisent un algorithme génétique pour obtenir simultanément une fragmentation verticale et un chemin d'accès à ces partitions. Ils ont utilisé également le nombre d'accès disque comme critère d'évaluation des partitions obtenues [SOS00].

Travaux de Bellatreche et al., (2000, 2005) s'intéressent à la conception d'un schéma de fragmentation optimal en utilisant la fragmentation horizontale dérivée. Ils proposent une technique pour partitionner la table des faits en fonction des schémas de fragmentation des tables de dimension. Les auteurs considèrent que l'application du partitionnement horizontal aux entrepôts de données rend l'espace de recherche très important pour la sélection du schéma de fragmentation et peut générer un nombre important de partitions qui sera difficile à gérer. Pour remédier à ces problèmes, les auteurs formalisent le problème de sélection d'un schéma de fragmentation comme un problème d'optimisation et proposent pour sa résolution une approche qui combine un algorithme génétique et un recuit simulé [BEL00, BEL05].

Travaux de Boukhelfa (2009), sont à considérer, à notre sens, comme les travaux les plus approfondis ayant traiter l'utilisation des métaheuristiques pour la conception d'un schéma de fragmentation optimal dans le contexte des entrepôts de données. L'auteur a proposé trois algorithmes pour sélectionner un schéma de fragmentation horizontal : un algorithme Hill Climbing (HC), un algorithme génétique (AG) et un algorithme de recuit simulé (RS). Il a détaillé pour chaque algorithme les principales fonctions qu'il utilise ainsi que sa fonction objectif. Les différents tests expérimentaux effectués, soit par l'utilisation d'un modèle de coût mathématique, soit sur un entrepôt réel implémenté sous le SGDB Oracle 10g, ont montré que son approche donne une bonne réduction du temps d'exécution des requêtes, à condition de bien choisir les algorithmes de sélection, les tables de dimension à fragmenter ainsi que les attributs de fragmentation [BOK09].

Travaux de Ziyati et al., (2010) portent sur l'adaptation de la fragmentation mixte aux entrepôts de données par l'utilisation des algorithmes génétiques pour la sélection d'un schéma de fragmentation mixte optimal [ZIE10].

Travaux d'Aleksandar Dimovski et al. (2011), les auteurs proposent une nouvelle méthode, pour la fragmentation horizontale des relations, basée sur l'abstraction des prédicats en utilisant un ensemble fini de prédicats arbitraires définis sur l'ensemble des domaines de relations. La méthode est formelle, des fragments arbitraires de relations peuvent être partitionnés par un nombre quelconque de prédicats, sans tenir compte de leurs affinités. Ils appliquent cette approche pour déterminer un schéma de fragmentation approprié pour un entrepôt de données. Les auteurs utilisent un algorithme génétique pour produire une solution optimale à ce type de problème d'optimisation [DIA11].

Travaux de Barr (2012), les auteurs ont modélisé le problème de sélection d'un schéma de fragmentation horizontal, qui minimise le coût global de la charge de requêtes, par une approche basée sur les colonies de fourmis artificielles en définissant les variables d'entrées qui sont : l'entrepôt de données non fragmenté, la charge des requêtes fréquentes et le nombre maximal de fragments exigé par l'administrateur de l'entrepôt de données [BAM12].

4.2.3 Discussion

Les métaheuristiques ne sont pas en général des algorithmes exacts et n'offrent pas de garantie de trouver une solution en un temps "raisonnable". Ils cherchent plutôt à trouver une solution sous-optimale de "bonne qualité" en un temps raisonnable. On sait que pour une classe de problèmes démontrée NP-complète, telle que la fragmentation des données, il existe dans la plupart des cas des instances de ces problèmes solubles en

temps polynomial. Malheureusement, les propriétés des problèmes qui les rendent plus ou moins faciles à résoudre sont souvent mal connues. Cette information est d'autant plus importante à découvrir qu'elle est en générale la clé permettant la conception d'un algorithme de résolution efficace.

La principale difficulté à laquelle est confrontée la communauté scientifique, en présence d'un problème d'optimisation, est celui du choix d'une méthode « efficace », capable de produire une solution « optimale » ou de qualité acceptable – au prix d'un temps de calcul « raisonnable ». Face à ce souci, la théorie n'est pas d'une grande utilité, car les théorèmes de convergence sont souvent inexistants ou applicables seulement sous des hypothèses très restrictives. En outre, le réglage optimal des divers paramètres d'une métaheuristique, qui peut être préconisé par la théorie, est souvent inapplicable en pratique, car il induit un coût de calcul prohibitif. En revanche, le choix d'une « bonne » méthode, et le réglage des paramètres de celle-ci, font généralement appel au savoir faire et à l'expérience de l'utilisateur, plutôt qu'à l'application stricte de règles bien établies.

De ce fait, le réglage des paramètres des métaheuristiques est un processus qui s'avère parfois long et complexe et sur lequel dépend fortement le comportement de l'algorithme [SHY98, VAF02]. Une mauvaise définition de ces paramètres peut rendre l'algorithme inapplicable, particulièrement dans un contexte industriel. Il est important de trouver un jeu de paramètres bien adapté au problème posé. Chaque problème nécessiterait en théorie une étude qui permettrait de dégager le jeu de paramètres optimal pour le traiter. Or, un tel procédé est souvent long à réaliser et demande une bonne connaissance au préalable du comportement de l'algorithme.

L'utilisateur est, certes, demandeur de méthodes rapides et efficaces, mais il est aussi demandeur de méthodes simples d'utilisation. Un enjeu majeur des métaheuristiques est de faciliter le choix des méthodes et de simplifier leurs réglages, afin de les adapter au mieux aux problèmes posés.

Dans ce contexte, il convient de souligner, qu'il existe une autre façon qui consiste tout simplement à ne pas choisir les paramètres. Au lieu de définir des paramètres avant l'exécution de la méthode, on préfère modifier les valeurs des paramètres pendant le traitement, en fonction des résultats trouvés au cours des différentes itérations. De nombreuses métaheuristiques ont été conçues dans ce domaine: des algorithmes génétiques [SCV96, SAH99, MUY02], des algorithmes de colonies de fourmis [CHL04, [DIG98, FOM07], des algorithmes de recherche tabou [BAR96] ou encore des algorithmes de recuit simulé [INL96].

Cependant, avec cette démarche les résultats recherchés ne sont pas forcément les résultats optimaux, mais du moins une approximation « correcte » de l'optimum. Ce qui

est perdu en efficacité est gagné en facilité de mise en œuvre de l'algorithme. Il est normal qu'un algorithme qui doit rechercher par lui-même les valeurs de ses propres paramètres présente des résultats inférieurs à un algorithme dont le comportement est « guidé » par des paramètres correctement définis par l'utilisateur.

C'est dans cette logique d'une meilleure définition des paramètres adéquats et adaptés à l'utilisation des métaheureustiques, que nous proposons l'utilisation des algorithmes d'Optimisation par Essaim Particulaires (OEP) pour la conception d'un schéma de fragmentation optimal. L'OEP partage plusieurs similarités avec d'autres métaheuristiques comme les algorithmes génétiques (AGs). Le système est initialisé par une population de solutions aléatoires et cherche des optima à travers les générations. Cependant, contrairement aux AGs, l'OEP n'utilise aucun opérateur évolutionnaire comme le croisement ou la mutation. Aussi, les algorithmes sont faciles à programmer et des versions adaptives avec un minimum de paramétrage sont disponibles. A cet titre, afin de limiter l'intervention de l'utilisateur, des études sur le choix des paramètres ont été effectuées [VAF02, TRI03] en menant particulièrement des tests sur les performances de l'OEP en fonction des valeurs de la vitesse et du coefficient d'inertie (voir section 4.3.3). Des versions adaptatives des algorithmes ont été testées et mises en place [CLM03, CLM06, [YAK04]. Ce type d'OEP limite considérablement l'intervention des utilisateurs, qui devront se concentrer seulement sur la spécification du problème à résoudre.

De plus, notre choix de l'utilisation de l'OEP par rapport à d'autres métaheuristiques est justifié principalement par le fait que cette heuristique [DEH08, DEH12] :

- met l'accent sur la coopération plutôt que sur la compétition et il n'y a pas de sélection (au moins dans les versions de base), l'idée étant qu'une particule même actuellement médiocre doit d'être conservée ;
- permet d'effectuer un regroupement distribué, donc sans contrôle ;
- c'est une approche dynamique, applicable dans le cas ou la fonction objectif se modifie dans le temps ;
- possibilité d'utiliser des fonction multi-objectifs ;
- existence des versions adaptatives de l'OEP, qui nécessitent moins de paramétrage ;
- facile à programmer.

Dans les sections qui suivent nous présentons d'abord le principe de l'OEP et nous détaillons, par la suite, notre approche de conception d'un schéma de fragmentation optimal.

4.3 Optimisation par Essaim de Particules

4.3.1 Origine

L'Optimisation par Essaim de Particules ou Particulaires (OEP) est une méthode née en 1995 aux Etats-Unis sous le nom de Particle Swarm Optimization (PSO). Initialement, ses deux concepteurs, Russel Eberhart et James Kennedy [KEJ95], cherchaient à modéliser des interactions sociales entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun. Chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information. La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Un modèle simple fut alors élaboré.

Dès les premières simulations, le comportement collectif de ces agents évoquait celui d'un essaim d'êtres vivants, convergeant parfois en plusieurs sous-essaims vers des sites intéressants. Ce comportement se retrouve dans bien d'autres modèles, explicitement inspirés des systèmes naturels. La métaphore, la plus représentative, est celle de l'essaim d'abeilles, du fait qu'une abeille ayant trouvé un site prometteur sait en informer certaines de ses consœurs et que celles-ci vont tenir compte de cette information pour leur prochain déplacement. Finalement, le modèle s'est révélé être trop simple pour vraiment simuler un comportement social, mais par contre très efficace en tant qu'outil d'optimisation [KEJ99].

Kennedy et Eberhart se sont, donc, inspirés de ces comportements socio-psychologiques pour créer l'OEP. Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation, « survole » l'espace de recherche, en quête de l'optimum global. Le déplacement d'une particule est influencé par les trois composantes suivantes :

- *Une composante physique* : la particule tend à suivre sa direction courante de déplacement ;
- *Une composante cognitive* : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée ;
- *Une composante sociale* : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

4.3.2 Principe

On considère, dans l'espace de recherche, un essaim de particules. Chaque particule est en mouvement selon une vitesse. A partir des informations dont elle dispose, une

particule doit décider de son prochain mouvement, c'est-à-dire décidé de sa nouvelle vitesse. Pour ce faire, elle combine linéairement trois informations :

- sa vitesse actuelle ;
- sa meilleure performance ;
- la meilleure performance de ses voisines (ses informatrices).

A l'aide de trois paramètres parfois appelés *coefficients de confiance ou d'accélération*, qui pondèrent trois tendances :

- tendance à suivre sa propre voie ;
- tendance conservatrice (revenir sur ses pas) ;
- tendance « panurgienne » (suivre le meilleur voisin) ;

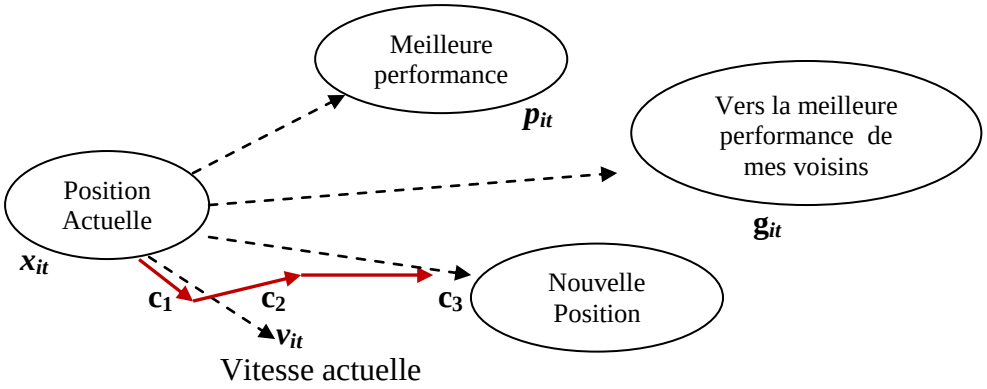


Figure 4.1- Schéma de principe du déplacement d'une particule. Pour réaliser son prochain mouvement, chaque particule combine trois tendances : suivre sa vitesse propre, revenir vers sa meilleure performance, aller vers la meilleure performance de ses informatrices.

4.3.3 Algorithme de base

Chaque particule représente une solution potentielle dans l'espace de recherche.

La nouvelle position d'une particule est déterminée en fonction de sa propre valeur et celle de ses voisines. Soit $x_i(t)$ la position de la particule i au temps t, sa position est modifiée en ajoutant une vitesse $v_i\ (t)$ à sa positon courante :

$$x_i\ (t) = x_i\ (t-1) + v_i\ (t) \qquad (4.1)$$

La vitesse de chaque particule est mise à jour suivant l'équation suivante:

$$v_i(t+1) = v_i(t) + c_1 r_1\ [pi(t) - xi(t)] + c_2 r_2\ [gi(t) - xi(t)] \qquad (4.2)$$

Avec :

$v_i(t)$: est la vitesse de particule i à l'instant t ;

$x_i(t)$: est la position de particule i à l'instant t ;

c_1 et c_2 : sont deux constantes appelées, *coefficients d'accélération*, fixées généralement par l'utilisateur ;

r_1 et r_2 : sont des nombres aléatoires tirés uniformément dans [0,1] à chaque itération,

$gi(t)$: est la meilleure solution trouvée jusqu'à l'instant t

$p_i(t)$: est la meilleure solution trouvée par le particule i.

$c_1r_1\ [pi(t) - xi(t)]$: correspond à la composante cognitive du déplacement. c_1 contrôle le comportement cognitif de la particule.

$c_2r_2\ [gi(t) - xi(t)]$: correspond à la composante sociale du déplacement. c_2 contrôle l'aptitude sociale de la particule.

C'est la vitesse qui dirige le processus de recherche et reflète la "sociabilité" des particules. Si l'on considère N particules et que chaque particule compare sa nouvelle position à sa meilleure position obtenue, cela donne l'algorithme 4, $\mathcal{F}$ étant la fonction objectif.

[Les variables et paramètres de l'algorithme]
N nombre de particules
x_i position de la particule P_i
v_i vitesse de la particule P_i
$pbest_i$ meilleure fitness obtenue pour la particule P_i
x_{pbesti} position de la particule P_i pour la meilleure fitness
ρ valeur aléatoire positive
[———————————————————————————————-]
[Initialisations]
Initialiser aléatoirement la population
. . .
[Traitement]
Répéter
  Pour *i* **de** *1* **à** *N* **faire**
    Si $(\mathcal{F}(x_i) > pbesti)$ **Alors**
      $pbest_i \leftarrow \mathcal{F}(xi)$
      $x\ pbesti \leftarrow x$
    Fin Si
    $v_i \leftarrow v_i + \rho\ (x_{pbesti} - x_i)$
    $x_i \leftarrow x_i + v_i$
  Fin Pour
Jusqu'à ce que (le processus converge)

Algorithme 4- Algorithme OEP de base

Il est à noter que le terme vitesse est ici inadéquat. Il serait plus approprié de parler de « direction de déplacement ». Cependant, pour respecter l'analogie avec le monde animalier, les auteurs ont préféré utiliser le terme « vitesse ».

L'OEP est un algorithme à population. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. A chaque itération de l'algorithme, chaque particule

est déplacée suivant les équations (4.1) et (4.2). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. p_i et g_i sont alors mis à jour. L'algorithme 4, ci-dessus, illustre cette procédure.

Ce premier algorithme ne prend pas en compte le voisinage, puisqu'on utilise uniquement l'amélioration obtenue sur la particule elle-même. En considérant un voisinage en étoile, l'algorithme 4 devient :

[Les variables et paramètres de l'algorithme]
N nombre de particules
x_i position de la particule P_i
v_i vitesse de la particule P_i
$pbest_i$ meilleure fitness obtenue pour la particule P_i
x_{pbesti} position de la particule P_i pour la meilleure fitness
x_{gbesti} position de la particule ayant la meilleure fitness de toutes
ρ_1, ρ_2 valeurs aléatoires positives
[———————————————————————————————-]
[Initialisations]
Initialiser aléatoirement la population
. . .
[Traitement]
Répéter
  Pour *i* **de** *1* **à** *N* **faire**
    Si $(\mathcal{F}(x_i) > pbest_i)$ **Alors**
      $pbest_i \leftarrow \mathcal{F}(x_i)$
      $x\ pbest_i \leftarrow x_i$
    Fin Si
    Si $(\mathcal{F}(x_i(t)) > gbest)$ **Alors**
      $gbest \leftarrow \mathcal{F}(x_i)$
      $x\ gbest \leftarrow x_i$
    Fin Si
  Fin Pour
  Pour *i* **de** *1* **à** *N* **faire**
    $v_i \leftarrow v_i + \rho_1 (x_{pbesti} - x_i) + \rho_2 (x_{gbest} - x_i)$
    $x_i \leftarrow x_i + v_i$
  Fin Pour
Jusqu'à ce que (le processus converge)

Algorithme 5- Algorithme OEP avec un voisinage en étoile

Plus une particule est éloignée de la meilleure solution globale et de sa meilleure solution, plus sera importante la variation de sa vitesse afin de faire bouger la particule vers les meilleures solutions. Les variables aléatoires ρ_1 et ρ_2 sont définies par :

$$\begin{cases} \rho_1 = r_1 c_1 \\ \rho_2 = r_2 c_2 \end{cases}$$

D'une manière résumée la forme générale de l'algorithme OEP est la suivante :

```
Début
    Initialisation des paramètres
    Initialisation de l'essaim (position et vitesse)
    Évaluation
    Répéter
        {Déterminer la meilleure position P
         Déterminer la meilleure position de tout ou partie de l'essaim E
         Mettre à jour les vitesses
         Mettre à jour les positions
         Evaluation
         Recherche locale (optionnel)
        }Jusqu'à (Critère d'arrêt satisfait)
Fin
```

Algorithme 6- Forme générale d'un algorithme OEP

L'algorithme s'exécute tant le critère d'arrêt n'a pas été atteint. Cela peut être :

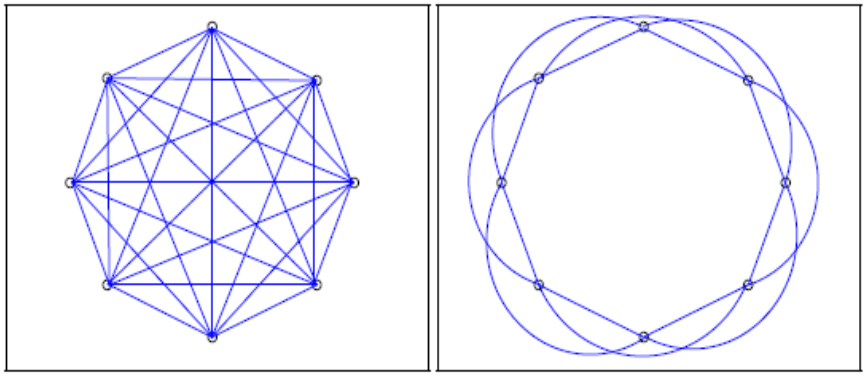
- un nombre fixe d'itérations ;
- en fonction de la fitness ;
- lorsque la variation de vitesse est proche de 0.

Cependant, au regard du problème posé et des exigences de l'utilisateur, d'autres critères d'arrêt peuvent être utilisés.

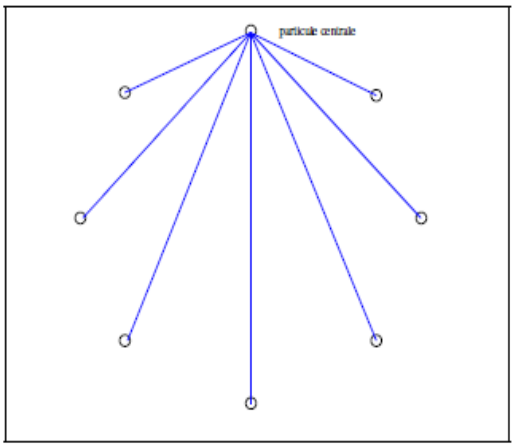
❖ **Le voisinage**

Le voisinage constitue la structure du réseau social. Les particules à l'intérieur d'un voisinage communiquent entre-elles. Différents voisinages ont été étudiés [KEJ99] et sont considérés en fonction des identificateurs des particules et non des informations topologiques comme les distances euclidiennes dans l'espace de recherche :

- topologie en étoile (figure 1(a)) : le réseau social est complet, chaque particule est attirée vers la meilleure particule notée *gbest* et communique avec les autres ;
- topologie en anneau (figure 1(b)) : chaque particule communique avec *n* (*n = 3* 1(b)) voisines immédiates. Chaque particule tend à se déplacer vers la meilleure dans son voisinage local notée *lbest* ;
- topologie en rayon (figure 1(c)) : une particule "centrale" est connectée à toutes les autres. Seule cette particule centrale ajuste sa position vers la meilleure, si cela provoque une amélioration, l'information est propagée aux autres.



(a) Etoile (b) Anneau



(c) Rayon

Figure 4.2- Voisinage

❖ **Voisinage basé sur la distance euclidienne**

Kennedy et Eberhart utilisent un voisinage basé sur les indices des particules, Suganthan [SUP99] utilise un voisinage spatial entre les particules. Une particule P_b est voisine d'une particule P_a si :

$$\frac{\|x_a - x_b\|}{d_{max}} < \varepsilon$$

Avec d_{max} la plus grande distance entre deux particules et

$$\varepsilon = \frac{3t + 0.6\, t_{max}}{t_{max}}$$

Avec t l'itération courante et t_{max} le nombre maximale d'itérations. On remarque que la taille du voisinage croît avec le temps.

4.4 Application de l'OEP à la fragmentation des données

Nos premiers travaux relatifs à l'application de l'OEP pour la sélection d'un schéma de fragmentation optimal, ont été basés sur des versions classiques de l'OEP [KEJ95]. Ce type d'OEP nécessite, à l'instar des autres métaheuristiques, d'effectuer plusieurs tests, notamment, pour déterminer les meilleurs valeurs des coefficients d'accélération permettant de converger vers les solutions les optimales [BEB11]. Cette version d'OEP a été appliquée dans nos travaux relatifs à la sélection d'un schéma de fragmentation vertical de la table des faits [DEH08] puis à la sélection d'un schéma de fragmentation horizontal primaire [DEH12].

Pour palier au problème de paramétrage qui entrave l'application des algorithmes évolutionnaires, nous avons opté sur l'utilisation de versions adaptatives de l'OEP. Tel que présenté dans la section 4.2.3, ce type d'OEP limite considérablement l'intervention des utilisateurs, qui devront se concentrer seulement sur la spécification du problème à résoudre.

A la différence des autres travaux, l'utilisation des algorithmes OEP, par rapport aux autres algorithmes évolutionnaires, notamment les algorithmes génétiques, offre plusieurs avantages liés, notamment, à leur simplicité en termes de programmation et de paramétrage, comme indiqué dans les sections ci-dessus.

Dans nos approches, nous avons opté pour l'utilisation d'une version adaptative de l'algorithme OEP, nommé TRIBE [CLM03, CLM06, COY07, COY08a, COY08b]. TRIBES peut être considéré comme un algorithme sans paramètres de contrôle. Il est défini comme une « boîte noire », pour laquelle l'utilisateur n'a qu'à spécifier le problème à résoudre.

En revanche l'utilisation d'une OEP adaptative impose de répondre à deux types de questions : « Comment la structure de l'essaim évolue au cours du temps ? » et « Quel comportement doit adopter une particule ? ». La première question est relative à la définition du paramètre *N*. Et la deuxième question est relative à la définition des paramètres c_1, c_2 et de la vitesse *v*, présentés dans la section 4.3.3.

Nous détaillons dans les sections qui suivent nos approches d'application de l'OEP aux trois techniques de sélection d'un schéma de fragmentation optimal, à savoir la fragmentation verticale, horizontale primaire et dérivée.

4.5 OEP pour la conception d'un schéma de fragmentation vertical

La fragmentation verticale (FV), telle qu'elle a été présenté dans le chapitre 3, a pour but de placer dans un même fragment les attributs qui sont généralement interrogés ensemble. Une mesure indiquant la proximité des attributs est leur affinité, c'est-à-dire la somme des fréquences d'accès des requêtes accédant simultanément à deux attributs.

Dans le contexte des entrepôts de données, l'application de la FV, n'a pas suscité l'intérêt escompté. Ceci est dû plus particulièrement au problème du nombre important des schémas que peut générer ce type de fragmentation.

Dans cette section, à l'instar d'autres travaux qui ont proposé l'utilisation des heuristiques pour résoudre le problème inhérent à la fragmentation verticale, nous formalisons le problème de la FV comme un problème d'optimisation et nous proposons l'utilisation d'un algorithme basé sur l'OEP pour la sélection d'un schéma de fragmentation vertical (*SFV*) [DEH08]. Nous présentons notre démarche ainsi qu'un algorithme basé sur TRIBE pour la sélection d'un *SFV* optimal d'un entrepôt de données.

4.5.1 Démarche

Dans cette section nous présentons notre approche, dénommée OEP-FV, de fragmentation verticale d'un entrepôt de données, modélisé en schéma en étoile, ayant *D* tables de dimension $\{D_1, D_2, \ldots, D_d\}$ reliées et une table des faits *F* de *n* attributs. Soit $Q = \{q_1, q_2, \ldots, q_q\}$ l'ensemble des requêtes, les plus fréquentes, ou chaque requête q_i possède une fréquence d'accès $freq_i$. Aussi, pour permettre de contrôler le nombre de fragments qui sera généré, on fixe un seuil *W* qui représente le nombre maximum de fragments verticaux. Le problème consiste donc de déterminer un *SFV* composé de *m* fragments $(Fr_1, Fr_2, \ldots, Fr_m)$, dans lequel le coût d'exécution des requêtes sur le schéma en étoile sera réduit et que le nombre des fragments ne dépasse pas le seuil *W*.

Notre démarche se compose de trois étapes : 1) *construction de la matrice d'usage*; 2*)*; *construction de la matrice d'affinité ; 3) groupement des attributs* ; et *4) construction des fragments verticaux.*

Il convient de préciser que, dans notre approche, nous considérons les hypothèses ci-après :

- Les tables de dimension sont de taille petite et sont stockées en mémoire pendant l'exécution des requêtes.
- le partitionnement sera effectué uniquement sur la table des faits selon une approche verticale.

1. Construction de la matrice d'usage des attributs

Tel qu'il a été présenté dans la section 3.5 du chapitre 3, la matrice d'usage des attributs, *MUA*, consiste à présenter l'utilisation des attributs par les requêtes les plus fréquentes. Les lignes de cette matrice représentent les requêtes q_q et les colonnes les attributs. L'élément de la ligne *i* et de la colonne *j* a pour valeur 1 si l'attribut *j* est accédé par la requête *i*, sinon cette valeur est nulle.

Attributs / Requêtes	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
q1	0	1	0	0	1	0	1	0	0	0
q2	1	1	1	0	0	0	0	1	1	0
q3	0	0	0	1	0	1	0	0	0	1
q4	1	0	0	0	0	0	1	1	0	0
q5	1	1	1	0	1	0	1	1	1	0
q6	0	1	0	0	1	0	0	0	0	0
q7	0	0	1	0	0	0	0	0	1	0
q8	0	0	1	1	0	1	0	0	1	1

Tableau 4.1- Exemple d'une matrice d'usage des attributs

2. Construction de la matrice d'affinité des attributs

La matrice d'affinité des attributs, *MAA* , est constituée, en ligne et en colonne, des attributs de la relation à fragmenter. Les éléments de cette matrice reflètent l'affinité entre attributs. Cette affinité correspond à la somme des fréquences d'accès des requêtes accédant simultanément aux deux attributs.

Attributs / Attributs	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	25	0	0	25	0	25	0
A2	50	5	50	0	0	0	0	50
A3	0	0	0	25	0	25	0	0
A4	35	0	0	0	0	0	35	35
A5	25	25	25	0	25	0	25	25
A6	0	25	0	0	25	0	0	0
A7	0	0	25	0	0	0	0	0
A8	0	0	15	15	0	15	0	0

Tableau 4.2- Exemple d'une matrice d'affinité des attributs

3. **Regroupement des attributs et création des fragments**

A partir de la *MAA*, nous proposons l'utilisation d'un algorithme basé sur l'OEP pour le regroupement des attributs selon leurs fréquences d'accès. Ce regroupement permettra de former des partitions qui composeront le *SFV*. Ce schéma sera, après chaque itération de l'algorithme OEP, évalué jusqu'à atteindre la solution optimale en satisfaisant, bien entendu, le critère d'arrêt préalablement fixé.

4.5.2 Algorithme OEP-FV

Le problème de la sélection d'un schéma de fragmentation vertical est formulé de la manière suivante : soit *N* le nombre d'attributs dans un espace de recherche à *M* dimen-

sions. Le problème consiste à concevoir *Fr* fragments de telle sorte que les attributs au sein d'un même fragment présentent une similarité entre eux comparativement aux attributs des autres fragments et ce pour atteindre un compromis entre le temps d'exécution des requêtes et le coût de maintenance.

4.5.2.1 Structure de l'essaim

Tel qu'il a été présenté, ci-dessus, nous utilisons la version adaptative de l'OEP TRIBE pour la conception du *SFV*. Dans TRIBES, l'essaim de particules est divisé en plusieurs sous-essaims appelés tribus. Les tribus sont de tailles différentes, qui évoluent au cours du temps. Le but est d'explorer simultanément plusieurs régions de l'espace de recherche, généralement des optima locaux, avant de prendre une décision globale. Les tribus échangent, alors, leurs résultats tout au long du traitement. Un tel type de structure est comparable aux structures multi-essaims utilisées dans d'autres algorithmes [PAK04, NIB05].

Dans notre approche chaque tribu correspond à un fragment Fr_i. Les relations entre les attributs (les particules) à l'intérieur d'un fragment sont définies par une topologie connectée. Chaque attribut connait la meilleure et la plus mauvaise position jamais atteinte par le fragment, c'est-à-dire que chaque attribut connait les positions pour lesquelles le fragment, dans lequel il appartient, a trouvé la plus petite et la plus grande valeur de la fonction objectif. Cette forme de communication dans TRIBE est appelée *communication intra-tribu*.

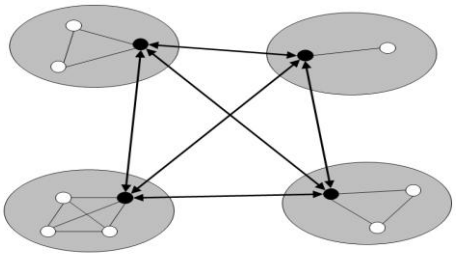


Figure 4.3- Essaim de particule composé de tribus

Au fur et à mesure du traitement, chaque fragment Fr_i va converger vers un optimum local. Il est donc nécessaire qu'ils communiquent entre eux pour définir lequel de ces optima est à retenir par l'utilisateur. La communication entre les fragments est établie par l'intermédiaire des meilleurs attributs de chaque fragment. Ce type de communication est appelé *communication inter- tribu*. Il est à noter que le meilleur attribut, de chaque fragment, est celui qui détient la fréquence d'accès la plus élevé.

En résumé, chaque attribut est informé par lui-même (mémoire cognitive pA_i), par tous les éléments de son fragment (appelés informateurs internes), et, si cet attribut est le meilleur attribut de son fragment, alors il est aussi informé par les autres meilleurs attributs des autres fragments (appelés informateurs externes). Toutes ces positions sont appelées les informateurs de l'attribut. La mémoire sociale de la position gA_i de chaque attribut est l'informateur pour lequel la valeur de la fonction objectif est la plus petite.

4.5.2.2 Evolution des fragments

Pour mesurer la pertinence des attributs et des fragments, deux indicateurs de qualité sont définis. Un indicateur pour les attributs et un autre pour les fragments. Un attribut est « bon » s'il a amélioré sa meilleure performance au cours de la dernière itération. Si non il est dit neutre. On constate bien que cette définition est qualitative, on ne mesure pas l'amélioration, on examine simplement si elle est strictement positive (amélioration réelle) ou bien nulle (pas d'amélioration).

Dans cette phase de conception des fragments, ce qui nous intéresse le plus est la performance globale d'un fragment. Nous allons donc définir deux qualités, *bonne* et *mauvaise*, et utiliser deux règles très simples :

Règle 1 : *Plus le nombre de bons attributs du fragment est grand, plus il est lui même bon et inversement*.

Règle 2 : *Un fragment Fr_i est dit « bon » ou mauvais suivant le nombre de bons attributs présents en son sein. Un fragment est mauvais si aucun de ses attributs n'a amélioré sa meilleure performance au cours de la dernière itération.*

En pratique, dans TRIBE, la qualité d'une tribu est évaluée de la manière suivante. Soit T sa taille (son nombre de particules) et son nombre de bonnes particules B, au plus égal à T. Un nombre p entre 0 et T est généré aléatoirement, selon une distribution uniforme. Si $B \leq p$, alors la tribu est dite mauvaise, sinon elle est dite bonne. A ces qualités seront associées des règles d'évolution, tendant à favoriser la création de nouvelles tribus et, partant, l'exploration de l'espace de recherche.

Par ailleurs, il convient de souligner que, par rapport à l'OEP classique, dans TRIBE la mémoire de la particule est légèrement augmentée, de façon à ce qu'elle se souvienne de ses deux dernières variations de performance, décrivant ainsi un historique de ses déplacements. A partir de là, un troisième statut peut être défini : une particule sera dite *excellente* si ces deux variations sont des améliorations. Ceci sera surtout utile pour définir une stratégie de déplacement, adaptée, des particules.

Deux fonctions *Merge* et *Split* sont utilisées pour la gestion des fragments et l'amélioration de la qualité de la solution. La fonction *Merge* permet de fusionner deux attributs ou deux fragments et la fonction *Split* permet de séparer deux attributs ou

d'éclater un fragment en deux. Ces fonctions sont largement utilisées dans les SGBD tel que Oracle pour le gestion des partitions.

4.5.2.3 Paramètres de l'algorithme

La plupart des paramètres dans les algorithmes adaptatifs OEP sont prédéfinis empiriquement, tels que les coefficients c_1 et c_2. Toutefois, d'autres paramètres sont à déterminer selon le contexte de l'utilisation de l'algorithme, c'est le cas, par exemple de la taille de l'essaim.
Dans l'OEP, les paramètres à considérés par l'utilisateur sont les suivants :

- **Les particules**

Tel qu'il a été présenté dans la section 4.3.3, l'OEP est un algorithme à population. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. Dans notre approche les particules de l'essaim sont les attributs de la table des fais. Chaque attribut est caractérisé par sa position et sa vitesse et sa fréquence d'accès.

$$A_i(t) = \begin{cases} xA_i(t) \\ vA_i(t) \\ frA_i \end{cases}$$

Pour les attributs de la table des faits les équations (4.1) et (4.2) seront donc :

$$xA_i(t) = Ax_i(t-1) + vA_i(t) \quad (4.3)$$

$$vA_i(t+1) = vA_i(t) + c_1r_1[pAi(t) - xAi(t)] + c_2r_2[gAi(t) - xAi(t)] \quad (4.4)$$

Rappelons qu'à chaque itération de l'algorithme, chaque attribut A_i est déplacé suivant les équations (4.3) et (4.4). Une fois le déplacement des particules effectué, des partitions sont crées et les nouvelles positions pA_i et gA_i sont alors mis à jour. Rappelons que p_i et g_i sont respectivement, la meilleure position trouvée par le particule i et la meilleure solution trouvée jusqu'à l'instant t.

- **La vitesse maximale**

Il est possible que le déplacement d'une particule la conduise à sortir de l'espace de recherche, ce qui peut conduire à une divergence du système. De plus, et pour éviter que les particules se déplacent trop rapidement d'une région à une autre dans l'espace de recherche, on introduit un nouveau paramètre une vitesse maximale V_{max} [EBR96]. Une étude plus détaillée sur le comportement de l'OEP selon les valeurs du paramètre V_{max} est disponible dans [FAH01]. Ainsi si $vA_i(t)$ est la vitesse de la particule A_i au temps t, alors

$$vA_i(t) = V_{max} \text{ si } vA_i(t) > V_{max} \text{ et } vA_i(t) = -V_{max} \text{ si } vA_i(t) < -V_{max}$$

Une stratégie visant à éviter la dispersion des particules peut être mise en place. Une telle stratégie permet de ramener une particule sortie de l'espace de recherche à l'intérieure de celui-ci. Il existe plusieurs approches qui peuvent être employées :

- la particule est laissée à l'extérieur de l'espace de recherche, mais on n'évalue pas sa fonction objectif. Ainsi, elle ne pourra pas attirer les autres particules en dehors de l'espace de recherche ;
- la particule est stoppée à la frontière et les composantes correspondantes de la vitesse sont annulées ;
- la particule « rebondit » sur la frontière. La particule est stoppée sur la frontière, mais les composantes correspondantes de la vitesse sont multipliées par un coefficient a tiré aléatoirement dans l'intervalle [-1,0].

Il convient de noter, que V_{max} n'est pas obligatoire si on utilise un coefficient de constriction (resserrement) *k* [CLM02]. En effet, l'utilisation de ce facteur de constriction permet de prévenir l'explosion de l'essaim, d'assurer la convergence, mais aussi de s'éviter la définition arbitraire d'un paramètre V_{max}.

$$v_i(t+1) = k(\ v_i\ (t) + \rho_1\ (p_i\ -\ x_i\ (t)) + \rho_2\ (g_i\ -\ x_i\ (t))) \qquad (4.5)$$

avec $k = 1 - \frac{1}{\rho} + \frac{\sqrt{|\rho^2 - 4\rho|}}{2}$ et $\rho = \rho_1 + \rho_2 > 4$

❖ **Facteur d'inertie**

Pour contrôler l'influence de la vitesse obtenue au pas précédent nous utilisons une constante *w*, appelée, *coefficient d'inertie* qui décroit en fonction du temps.

$$vA_i(t+1) = wvAi\ (t) + \rho_1\ (x_{pbesti}\ -\ x_i\ (t)) + \rho_2\ (x_{gbest}\ -\ x_i\ (t))) \qquad (4.6)$$

wvAi(t) : correspond à la composante physique du déplacement. Le paramètre *w* contrôle l'influence de la direction de déplacement sur le déplacement futur. Il est à noter que dans certaines applications, le paramètre *w* peut être variable [SIP06].

Un grand facteur d'inertie provoque une grande exploration de l'espace de recherche alors qu'un petit facteur d'inertie concentre la recherche sur un petit espace.

4.5.3.3 La fonction objectif

La définition de notre fonction objectif est inspirée des travaux de Boukhelfa [BOK09] et Xinjian et Lowenthal [XIL04]. Initialement, elle est calculée en utilisant la sélectivité des prédicats simples et tenant compte d'autres paramètres tels que la taille de tables, la taille de la mémoire, les entrées /sorties disques.

La fonction objectif consiste à calculer la somme des accès (I/O) nécessaires pour exécuter chaque requête sur un entrepôt de données fragmenté.

Pour chaque attribut A_j, nous définissons un facteur de sélectivite sur la table des faits noté $Sel_{Fr}^{A_j}$. Chaque requête q_q est exécute sur un schéma de fragmentation *SFV*. Afin d'identifier le fragment qui a été traité par cette requête, nous introduisons une variable booléenne notée $V(q_q , Fr_i)$ et définie par $V(q_q, Fr_i)=1$ si le fragment Fr_i est utilisé par la requête q_q , *0* si non. Cependant, le coût de la requête q_q en terme d'Entrées/Sorties (I/O) est calculé par :

$$IOC\ (q_q, Fr_i) = \sum_{j=1}^{N_i} V(q_q, Fr_i)\ \prod_{i=1}^{M_j}\ Sel_{Fr}^{A_j} \times \left\lceil \frac{\|F\| \times L}{PS} \right\rceil$$

Avec M_j , *F, L, PS*, respectivement, le nombre d'attributs utilisé pour définir les fragments FR_i, la cardinalité de la table des faits (nombre de tuples), la longueur en octets de chaque tuple et enfin, la taille des pages disque (en octets). Le coût total, donc, pour toutes les requêtes est calculé par :

$$TIOC\ (q, Fr_i) = \sum_{k=1}^{m} IOC\ (q_q, Fr_i)$$

Notre approche consiste à déterminer un schéma de fragmentation tel que le coût total de l'exécution des requêtes est minime.

Dans l'algorithme OEP-FV, présenté dans l'algorithme 7 ci-dessous, la solution initiale est choisie d'une manière aléatoire, les déplacements des particules vont permettre la création des partitions et par conséquent améliorer itérativement la qualité de la solution. Les mouvements de ces particules se baseront sur leurs vitesses et la stratégie de déplacement adoptée.

Pour l'implémentation de l'algorithme OEP-FV, nous utilisons une grille G de forme carrée. La taille de la grille est déterminée automatiquement en fonction du nombre d'attribut à traiter. Pour *n* attributs, G comporte L cases par côté avec : $L = \lceil \sqrt{2n} \rceil$. Cette formule permet de s'assurer que le nombre de cases est au moins égal au nombre d'individus. A l'inverse des autres approches qui utilisent les grilles [LUE94], nous permettons à plusieurs individus d'être placés dans une même case, ce qui forme donc une partition. Le remplissage de la grille se fait à partir de la matrice d'affinité des attributs *MAA*.

Entrée :
MAA: Matrice d'affinité des attributs
W : nombre de fragments maximum
EP : ensemble de paramètres d'entrées relatif au modèle de coût (taille de la table, données sy système, ect...)
$pA_i = gA_i = xA_i$
$c_1 = c_2$
Sortie :
SFV_{opt} : *schéma de fragmentation*

Notation :
SF_{init} *: schéma de fragmentation initial*
ES_0 *: Essaim initiale de particules*
$\mathcal{F}$ *-FV: fonction objectif*
SF_{opt} *: schéma de fragmentation optimal*

Début
Initialisation aléatoire des attributs
$ES_0 = Essaim(SF_{init})$
$\mathcal{F}$*- FV = OEP- Fonction (W, EP)*
Tant que *le critère d'arrêt n'est pas atteint* ***faire***
Déterminer le statut de chaque particule
Déplacement des particules selon la stratégie adoptée
Adaptations structurelles par l'exécution des fonctions Merge et split
Mise à jour des $pA_i = gA_i$
Evaluer la fonction objectif $\mathcal{F}$*- FV*
SFV_{opt} *= OEP- Meilleure Solution*
Fin Tant que
FV (SF, F) / fragmentation de la table des faits */*
Fin

Algorithme 7- Algorithme OEP-FV

4.6 OEP pour la conception d'un schéma de fragmentation horizontal dérivée

Dans cette section nous présentons notre approche de fragmentation horizontale dérivée, dénommée OEP-FHD, d'un entrepôt de données, modélisé en schéma en étoile ayant *d* tables de dimension $\{D_1, D_2, \ldots, D_d\}$ reliées et une table des faits *F*.

Tel qu'il a été présenté, dans le chapitre 3, selon l'approche de fragmentation horizontale dérivée, certaines ou toutes les tables de dimension seront fragmentées selon une approche de fragmentation primaire, à l'issue, la table des faits sera fragmentée selon les fragments générés par la fragmentation des tables de dimension.

Rappelons que ce type de fragmentation nécessite deux types d'informations essentiels, à savoir des informations sur les prédicats de sélection simples et leurs sélectivités et des informations sur les fréquences d'accès des requêtes.

La démarche permettant de concevoir un schéma de fragmentation optimal selon l'approche horizontale dérivée se décline en six étapes : 1) *extraction des prédicats de sélections* ; 2*) attribution des prédicats aux tables* ; 3) *identifications des tables de dimension à fragmenter* ; 4) *vérification de la complétude et la minimalité des prédicats* ; 5) *fragmentation des tables de dimension* ; 6) *fragmentation de la table des faits*.

Comme indiqué, dans l'introduction de ce chapitre, l'approche de fragmentation horizontale dérivée a été démontrée comme étant un problème NP-complet. L'application de cette approche aux entrepôts de données génère un nombre important de sous-schémas en étoile qui sera par la suite difficile à maintenir. Ce nombre est calculé par l'équation suivante :

$$N = \prod_{i=1}^{g} m_i$$

Avec g le nombre de tables de dimension fragmentées en m_i fragments.

Exemple 4 : *Supposons que les tables dimensions d_1, d_2 et d_3, sont fragmentées, respectivement en 5, 10 et 20 fragments. La table des faits sera donc fragmentée en : 5* x *10* x *20 = 1000 fragments.*

D'après cet exemple, on constate que le nombre de fragments de la table des faits est proportionnel au nombre de fragments des tables de dimension. En conséquence, maitriser l'explosion du nombre de sous-schéma en étoile revient à maitriser le nombre de fragments des tables de dimension. Ceci pourra être réalisé par le développement de stratégies permettant de sélectionner les tables de dimension les plus appropriées à fragmenter et en fixant, au départ, le nombre maximum de fragments à générer.

Notre approche consiste à utiliser l'OEP pour la sélection d'un schéma de fragmentation optimal pour les tables de dimension. Ce schéma devra trouver un compromis entre le temps d'exécution des requêtes et le coût de maintenance des fragments [DEH14b].

4.6.1 Démarche

Notre approche OEP-FHD s'inspire des travaux déjà réalisés dans ce domaine, plus particulièrement ceux de Belletrache et Boukhalfa [BEL00, BOK09]. Elle consiste, a l'issue d'une phase d'extraction des prédicats de sélection à : *1) découper les domaines des attributs ; 2) L'utilisation de l'OEP pour la sélection du schéma de fragmentation optimal ; 3) fragmentation des tables de dimension ; 4) fragmentation de la table des faits*.

4.6.1.1 Découpage des domaines des attributs

Cette étape consiste, à partir des domaines des attributs, à concevoir un schéma de fragmentation primaire des tables de dimension. Il s'agit, de découper les domaines de chaque attribut candidat à la fragmentation en un ensemble de sous-domaines. Il convient de souligner, que ce découpage peut être effectué à priori ou à postériori [BOK09] :

- *à priori* : le découpage est réalisé selon des critères relatifs à l'exploitation de l'entrepôt de données ;
- *à postériori* : l'administrateur de l'entrepôt de données découpe le domaine de chaque attribut selon les prédicats définis sur cet attribut dans les requêtes accédant à l'entrepôt.

Cela signifie, que la notion du découpage du domaine des attributs de fragmentation dans les travaux académiques est implicite à travers les prédicats utilisés dans les clauses de fragmentation. Par contre, elle est explicite dans les travaux industriels où l'administrateur décompose le domaine pour la fragmentation selon les modes, par intervalle, par hachage, ou par liste. Dans notre approche, nous avons opté pour un découpage qui ne prend pas en considération les prédicats de sélection pour la décomposition des attributs de fragmentation.

On commence, donc, par sélectionner des attributs candidats pour la fragmentation. Ces derniers seront découpés en sous-domaines, et ce afin de sélectionner un schéma de fragmentation horizontal primaire pour les tables de dimensions. Soit $AF = \{AF_1, AF_2, ..., AF_t\}$ l'ensemble des attributs candidats pour la fragmentation. Chaque attribut AF_i, de la table de dimensions D_d, possède un domaine de valeur $Dom(AF_i)$. Le découpage du domaine $Dom(AF_i)$ en n_i sous domaines, $SDom(AFi)_{ni}$, permet de spécifier n_i fragments de la table de dimension D_d.

A travers le découpage des attributs candidats pour la fragmentation des tables de dimension, on constate que le nombre de fragments d'une table de dimension D_d, à générer, sera Πn_i fragments. Cependant, comme il a été précisé ci-dessus, le nombre de fragments de la table des faits, et par conséquent le nombre des sous-schémas en étoile de l'entrepôt de données, est proportionnel au nombre de fragments des tables de dimension. Minimiser le nombre des sous-schémas revient donc à chercher un schéma de fragmentation optimal des tables de dimension qui satisfait le compromis entre le coût d'exécution des requêtes et le nombre de fragments des tables dimension. Soit W le nombre de fragments maximum à générer. Ce seuil est généralement fixé par l'administrateur de l'entrepôt de données.

En résumé, chaque schéma de fragmentation obtenu doit être d'abord évalué et le meilleur schéma de fragmentation, en termes de temps d'exécution des requêtes et de respect de la contrainte du seuil fixé, sera retenu. Ceci a été démontré comme étant un problème NP-complet qui nécessite de faire appel à des heuristiques pour le résoudre.

Rappelons que dans ce contexte, les travaux qui ont proposé l'utilisation des algorithmes génétiques pour la réduction des fragments des tables de dimensions [BOK09], [ZIE10] proposent un codage pour le schéma de fragmentation. Dans ce codage, chaque attribut de fragmentation AF_i sera représenté par un tableau d'entiers de n_i cellules, où n_i correspond au nombre de sous-domaines, les valeurs dans les cellules de ce tableau varient entre 1 et n_i. Si deux cellules ont la même valeur, leurs sous-domaines seront regroupés en un seul sous-domaine. Si tous les sous-domaines d'un attribut possèdent la même valeur, l'attribut ne participe pas à la fragmentation. Chaque individu (un schéma de fragmentation) est donc représenté par un tableau d'entiers (tableau 4.3).

Mois	**1**	**1**		
Région	**2**	**1**	**3**	**3**
Description	**2**	**1**	**2**	

Tableau 4.3- Exemple d'individu

Ce codage montre que les dimensions concernées par la fragmentation sont Magasin et Produit. L'attribut Région en trois fragments et l'attribut Description en deux fragments. L'attribut Mois ne participe pas à la fragmentation car ses sous- domaines ont la même valeur.

Une représentation en tableau multidimensionnel, dénoté *H*, peut être utilisée pour calculer le nombre de tous les schémas de fragmentation possibles [ZIE10].

$$H = 2^{\left(\sum_{i=1}^{k} n_i\right)}$$

Où k et n_i représentent, respectivement, le nombre d'attributs de fragmentation et le nombre de sous-domaines de l'attribut AF_i.

Exemple 5 : *Si on considère les trois attributs de fragmentation : Région et Description le nombre de schémas possibles est* $2^{(2+3)} = 2^5 = 32$.

Dans notre approche OEP-FHD, nous adoptons le même codage pour représenter le schéma de fragmentation des tables de dimension.

4.6.1.2 Construction de la matrice d'usage des sous domaines des attributs

Pour chaque attribut AF_i, une matrice d'usage de ces sous-domaines, *MUSDom,* est construite. Cette matrice d'usage est une matrice *(m* x *n)* ou *m* représente le nombre des requêtes et *n* le nombre des sous-domaines de l'attribut AF_i. Les éléments cette matrice sont les fréquences des accès des requêtes aux sous-domaines.

	$SDom_1$	**$SDom_2$**	**$SDom_3$**
q_1	25	0	50
q_2	0	45	70
q_3	55	35	25
q_4	0	0	55

Tableau 4.4- Exemple de matrice d'usage des sous-domaines

4.6.1.3 Algorithme OEP-FHD pour la conception d'un schéma de fragmentation horizontal dérivée

L'algorithme OEP-FHD, pour la sélection d'un schéma de fragmentation optimal des tables de dimension, consiste à regrouper certains sous-domaines en un seul sous-domaine selon leurs usages par les requêtes les plus fréquentes. Ceci permettra de diminuer le nombre de fragments des tables de dimension. A partir des sous-domaines, conçus pour chaque attribut de fragmentation, l'algorithme donne en sortie un ensemble de partitions dont chacune représente le domaine de l'attribut traité. La génération du schéma de fragmentation s'effectue selon le codage présenté dans la section précédente.

L'algorithme OEP-HFD, considère en entrée la matrice d'usage des sous- domaines de l'attribut de fragmentation. L'essaim représente l'attribut et les particules sont les sous-domaines à regrouper en partition selon leurs fréquences d'usage par les requêtes les plus fréquentes.

Chaque particule est caractérisée par sa position et sa vitesse de déplacement selon les équations (4.1) et (4.2). La solution initiale est choisie d'une manière aléatoire, les déplacements des particules vont permettre d'améliorer itérativement la qualité de la solution. Les mouvements de ces particules se baseront sur leurs vitesses et la stratégie de déplacement adoptée. De la même manière de ce qui a été préconisé par notre approche OEP-FV, la stratégie de déplacement des particules prendre en considération la fréquence d'accès des requêtes. Ceci permettra d'avoir des vitesses de déplacement qui varieront d'une particule à une autre. Il devra, cependant, être réalisé en tenant compte du facteur d'inertie qui devra réguler la vitesse pour chaque itération.

Nous avons, également, opté pour l'utilisation de la version OEP adaptative TRIBE, décrite dans la section 4.4. Rappelons que dans cette version la plupart des paramètres sont prédéfinis. Toutefois certains paramètres, en l'occurrence la taille de l'essaim et les coefficients c_1 et c_2, doivent être préalablement définis comme entrée pour l'algorithme OEP-FHD.

Aussi, la fonction objectif pour l'évaluation des solutions est la même que celle utilisée dans notre approche OEP-FV. Pour rappel, cette fonction est calculée en utilisant la sélectivité des prédicats simples et tenant compte d'autres paramètres tels que la taille de tables, la taille de la mémoire, les entrées /sorties disques. Cette fonction objectif consiste à calculer la somme des accès (I/O) nécessaires pour exécuter chaque requête sur un entrepôt de données fragmenté.

4.6.1.4 Fragmentation des tables de dimension

Le meilleur schéma de fragmentation obtenu, à l'issue de l'étape précédente, propose une nouvelle manière de fragmenter les tables de dimension en se basant sur des nouveaux sous domaines obtenus par le découpage des domaines des attributs de fragmentation. Ainsi, selon le codage présenté précédemment, il ne sera considéré dans le processus de fragmentation des tables de dimension que les attributs qui ont le nombre de sous domaines ≥ 2. Si le nombre de sous-domaines égale à 1, cela signifie que l'ensemble des sous-domaines ont été combinés pour former un seul domaine et par conséquent l'attribut y relatif n'est pas considéré comme étant un attribut de fragmentation.

Ainsi, une fragmentation horizontale primaire est effectuée selon les sous domaines des attributs de fragmentations. Chaque attribut AF_i est un sous-domaine *SDomj* constitue un prédicat de fragmentation (*AFi = SDomj*). La génération des fragments des tables de dimension est effectuée à travers une opération de produit cartésien entre les prédicats de chaque attribut avec les prédicats des autres attributs de la même table. Cela permet de vérifier la disjonction des fragments et la reconstruction du schéma initial.

4.6.1.5 Fragmentation de la table des faits

A la fin du processus décrit dans les sections précédentes, la table des faits pourra être fragmentée en utilisant les schémas de fragmentation des tables de dimension. Il est à noter que plusieurs combinaisons peuvent être considérées pour générer les fragments de la table des faits [BOK09]. Chaque fragment de la table des faits est obtenu par une semi-jointure entre la table des faits et une combinaison des fragments de dimension :

$$F_i = F \ltimes D_{1j} \ltimes D_{2l} \ltimes \ldots \ltimes D_{gk} \ (1 \leq j \leq m_1, 1 \leq l \leq m_{2,\ldots}, 1 \leq k \leq m_g).$$

Exemple 6 :

Pour montrer comment les fragments de la table des faits seront générés, nous prenons un schéma de fragmentation qui sélectionne pour la fragmentation primaire les tables de dimension Magasin et Temps en deux fragments chacune. La table des faits Ventes sera alors fragmentée en quatre fragments notés : Ventes1, Ventes2, Ventes3 et Ventes4. Ces fragments sont définis comme suit :

- *Ventes1 = Ventes ⋉ Magasin1 ⋉ Temps1*
- *Ventes2 = Ventes ⋉ Magasin1 ⋉ Temps2*
- *Ventes3 = Ventes ⋉ Magasin2 ⋉ Temps2*
- *Ventes4 = Ventes ⋉ Magasin2 ⋉ Temps2*

Ainsi, le schéma en étoile, présenté dans la figure 2.7, sera fragmenté en quatre sous-schémas, notés SC1, SC2, SC3, SC4, définis comme suit :

- *SC1 comporte les tables Ventes1, Magasin1,Temps1 et Produit*
- *SC1 comporte les tables Ventes2, Magasin1,Temps2 et Produit*
- *SC1 comporte les tables Ventes3, Magasin2,Temps1 et Produit*
- *SC1 comporte les tables Ventes4, Magasin2,Temps2 et Produit*

Ci-dessous, l'algorithme OEP-FHD utilisé pour la sélection d'un schéma de fragmentation horizontal dérivé.

Entrée :
Tab [A] : Tableau d'attributs de fragmentation sélection
Tab[SDom] : Tableau à deux dimensions des sous domaine de l'attribut A
W : nombre de fragments maximum
D : ensemble des tables de dimension
F : table de fait
EP : ensemble de paramètres d'entrées relatif au modèle de coût (taille de la table, données sy système, ect...)
$pSDomFA_i = gSDomFA_i = xSDomFA_i$
$c_1 = c_2$
Sortie :
SFH : *schéma de fragmentation*

Notation :
SF_{init} *: schéma de fragmentation initial*
ES_0 *: Essaim initiale de particules*
$\mathcal{F}$ *-FHD: fonction objectif*
SFH : schéma de fragmentation horizontal

Début
TAB [SDom] = Complet (A, SDom$_i$) / compléter les domaines de chaque attribut*/*
SFH = Générer_SFH (A, SDom$_i$) / codage des domaines d'attributs*/*
ES$_0$ = Essaim(SF$_{init}$)
$\mathcal{F}$- FHD = OEP- Fonction (A, W, ED)
Tant que *le critère d'arrêt n'est pas atteint* ***faire***
Déterminer le statut de chaque particule
Déplacement des particules selon la stratégie adoptée
Mise à jour des pSDomFA$_i$ et gSdomFA$_i$
Evaluer la fonction objectif $\mathcal{F}$- FHD
SFH = OEP- Meilleure Solution (A, SD, ES$_0$, $\mathcal{F}$- FHD)
Fin Tant que
FHPrimaire (SFH, D) / fragmentation primaire des tables de dimension retenues*/*
FHD (D,F) / fragmentation dérivée de la table des faits*/*
Fin

Algorithme 8- Algorithme OEP-FHD

4.7 OEP pour la conception d'un schéma de fragmentation horizontal primaire

Dans cette section nous présentons notre approche, dénommée OEP-FHP, de fragmentation horizontale basée sur l'affinité des prédicats, pour fragmenter un entrepôt de données, modélisé en schéma en étoile, ayant *d* tables de dimensions *{D_1, D_2,..., D_d}* reliées et une table des faits *F*.

Rappelons que la fragmentation horizontale primaire, est effectuée grâce à des prédicats de sélection définis sur la table à fragmenter. Elle favorise le traitement des requêtes de restriction sur les attributs utilisés dans le processus de fragmentation, appelé *clé de fragmentation*. Deux méthodes sont généralement utilisées pour la fragmentation primaire :

1. *Approche basée sur la construction de prédicats* : consiste à partitionne une relation grâce à un ensemble complet et minimal de prédicats. La complétude signifie que deux instances d'une relation dans un même fragment ont la même possibilité d'être accédées séparément. La minimalité garantie l'absence de redondance dans les prédicats ;
2. *Approche basée sur l'affinité entre les prédicats*: utilise le principe d'affinité qui désigne la somme des fréquences d'accès des requêtes introduisant simultanément ces prédicats.

Notre approche de fragmentation horizontale primaire utilise l'affinité des prédicats pour la génération des fragments horizontaux. Rappelons que le principe de l'approche dirigée par affinité des prédicats a été inspiré des travaux relatifs à l'utilisation des affi-

nités entre attributs pour la conception d'un schéma de fragmentation vertical [NAB89]. Cette approche consiste à construire trois matrices pour identifier les affinités entre prédicats, à savoir : la matrice d'usage des prédicats, la matrice d'affinité des prédicats et la matrice d'affinité ordonnée et de générer, enfin, les fragments horizontaux à travers l'utilisation, généralement, des algorithmes de regroupement (voir section 3.5.2 du chapitre 3).

Notre approche reprend le même principe pour la sélection d'un schéma de fragmentation optimal en procédant au regroupement des prédicats selon leurs affinités. Ceci nécessite deux types d'informations essentiels, à savoir des informations sur les prédicats de sélection simples et des informations sur les fréquences d'accès des requêtes.
L'objectif de l'utilisation de l'OEP est de concevoir un schéma de fragmentation permettant de trouver un compromis entre le temps d'exécution des requêtes et le coût de maintenance des fragments.

Notre démarche se décline en cinq étapes : 1) *extraction des prédicats de sélections* ; 2*) construction de la matrice d'usage des prédicats* ; 3) *groupement des prédicats selon leurs affinités* ; 4) *construction des fragments horizontaux*.

Les premières phases de notre démarche, en l'occurrence les phases 1, 2 sont semblables aux travaux qui ont utilisé l'affinité des prédicats pour la conception d'un schéma de fragmentations horizontal. La particularité de notre approche consiste à utiliser l'OEP pour procéder au regroupement des prédicats selon leurs affinités et de générer, à l'issue, un schéma de fragmentation permettant de minimiser le temps d'exécution des requêtes respectant un seuil, préalablement fixé, relatif au nombre maximum de fragments à générer. Les sections qui suivent décrivent d'une manière détaillée notre approche.

4.7.1 Démarche

Notre approche de fragmentation horizontale se déroule selon les étapes ci-après :

1- Extraction, à partir de la charge de travail, de l'ensemble des prédicats simples ;
2- Construction de la matrice d'usage des prédicats;
3- Fixation du seuil *W* relatif au nombre de fragments à générer ;
4- Tenant compte des paramètres relatifs à la fonction coût à minimiser, du seuil W et des prédicats simples, l'application de l'OEP va permettre de sélectionner les « bon » prédicats et les regrouper en sous ensemble de partitions du même attribut.
5- Génération des minterms et conception des fragments horizontaux.

Il convient de préciser que, dans notre approche, nous considérons les hypothèses ci-après :

- les tables de dimensions sont de taille petite et sont stockées en mémoire pendant l'exécution des requêtes.
- le partitionnement sera effectué uniquement sur la table des faits selon une approche horizontale.

L'approche OEP-FHP utilise également la version TRIBE d'OEP. Dans les sections qui suivent nous décrivons notre approche de sélection d'un SF optimal pour la fragmentation d'un entrepôt de données.

1- Enumération des prédicats de sélection

Dans l'approche de fragmentation horizontale, on considère uniquement des prédicats de sélection qui sont définis dans les requêtes. A partir des *q* requêtes les plus fréquentes, nous énumérons, un ensemble des prédicats, notés, $PS = \{ps_1, ps_2, \ldots, ps_n\}$.

2- Construction de la matrice d'usage

A l'issue de l'énumération des prédicats de sélection, nous construisons une matrice d'usage des prédicats, qui représente l'utilisation des prédicats par les différentes requêtes. Les lignes de cette matrice, que l'on nomme *MUPS*, représentent les requêtes q_q et les colonnes les prédicats de sélection ps_n. Les termes de cette matrice sont les fréquences, fr_i , d'utilisation du prédicat ps_j de la requête q_i.

Prédicats / Requets	ps_1	ps_2	ps_3	ps_4	ps_5	ps_6	ps_7	ps_8	ps_9	ps_{10}
q_1	0	25	0	0	25	0	25	0	0	0
q_2	50	5	50	0	0	0	0	50	50	0
q_3	0	0	0	25	0	25	0	0	0	25
q_4	35	0	0	0	0	0	35	35	0	0
q_5	25	25	25	0	25	0	25	25	25	0
q_6	0	25	0	0	25	0	0	0	0	0
q_7	0	0	25	0	0	0	0	0	25	0
q_8	0	0	15	15	0	15	0	0	15	15

Tableau 4.5- Exemple d'une matrice d'usage des prédicats

Pour mesurer la qualité du schéma de fragmentation, l'OEP-FHP utilise la même une fonction objectif utilisée dans nos précédentes approches.

4.7.2 Description de l'approche OEP-FHP

4.7.2.1 Structure de l'essaim

Le même principe, d'utilisation de la version TRIBE d'OEP, appliqué dans les deux précédentes approches sera également reconduit pour la conception d'un schéma de fragmentation horizontal primaire, *SFHP*, dans laquelle chaque tribu correspond à un sous-ensemble de prédicats, noté *SEP*. Sauf, pour l'OEP-FHP un autre aspect est utilisé, il s'agit du la suppression d'une particule.

4.7.2.2 Suppression des prédicats

Dans n'importe quel algorithme d'optimisation, l'évaluation de la fonction objectif constitue l'élément le plus coûteux en temps d'exécution. Dans un souci de gain de temps, il est important d'évaluer la fonction objectif le moins de fois possible. Il serait opportun, à cet effet, d'éliminer les mauvais prédicats à chaque fois que cela est possible, étant donné que ces derniers n'apportent plus d'informations pertinentes au sous-ensemble, et donc à l'essaim, au regard des performances de ses congénères.

Enfin, les mêmes paramètres de vitesse, du facteur d'inertie, de voisinage et de stratégie de déplacement utilisé dans OEP-FV et OEP-FHD sont adaptés à la l'OEP-FHP.

L'approche OEP-FHP consiste à sélectionner, pour chaque attribut, les prédicats les plus intéressants en fonction de leurs fréquences d'accès et les regrouper en sous-ensembles.

- chaque particule (prédicat) va se positionner aléatoirement dans l'espace de recherche ;
- selon leurs vitesses et sa stratégie de déplacement, les particules vont se déplacer et tenant compte des informations de leurs voisinages, c'est-à-dire tendance à suivre le meilleur voisin ayant la fréquence la plus élevée ;
- des fonctions Merge et split sont également utilisées pour la gestion des prédicats et des sous ensembles ;
- des partitions vont se constituer est le processus s'arrête quat le nombre *W* est atteint et le schéma est évalué et sauvegardé ;
- ce processus est répété pour chaque attribut et la fin, l'attribut dont le *SF* est minimal sera retenu comme étant la clé de fragmentation de la table des faits.

4.8 Conclusion

Dans cette contribution nous avons proposé trois approches de conception d'un schéma de fragmentation optimal. Tel qu'il a été présenté en introduction à ce chapitre, la sélection d'un schéma de fragmentation vertical ou horizontal est démontrée comme un problème d'optimisation NP-complet. Beaucoup de travaux ont proposé l'utilisation des heuristiques, particulièrement les algorithmes génétiques et des colonies de fourmis artificielles, pour rechercher des solutions optimales satisfaisant le compromis entre le temps d'exécution des requêtes et le nombre de fragments à générer. Or, les solutions trouvées demeurent dépendantes d'une définition précise des paramètres des algorithmes utilisés.

C'est dans ce contexte que nous avons proposé l'utilisation de l'optimisation par essaim particulaires et plus particulièrement la version adaptative TRIBE, pour la sélection d'un schéma de fragmentation optimal. L'utilisation de l'OEP adaptative présente plusieurs avantages, particulièrement la simplicité de paramétrage et de programmation.

Nous avons adapté l'OEP aux trois techniques de fragmentation des données, en l'occurrence la fragmentation verticale, horizontale dérivée et horizontal primaire.

Notre première approche OEP-FV, permet de sélectionner un schéma de fragmentation vertical de la table des faits en se basant sur la fréquence d'accès des attributs. L'approche OEP-FHD porte sur la sélection d'un schéma de fragmentation optimal d'un entrepôt de données selon une approche de fragmentation horizontale dérivée. Cette approche consiste à chercher d'abord la ou les tables de dimension candidates pour une fragmentation primaire et procéder par la suite à la fragmentation de la table des faits selon les fragments générés par la fragmentation des tables de dimension. S'agissant de l'approche OEP-FHP elle consiste à déterminé, sur la base d'un groupement des prédicats, le schéma de fragmentation horizontal primaire de la table des faits.

Les tests expérimentaux de nos approches sont présentés dans le chapitre 7.

Chapitre 5
Vers une approche formelle pour l'évaluation d'un schéma de fragmentation

5.1 Introduction

Les algorithmes de conception d'un schéma de fragmentation n'offrent aucune garantie sur l'optimisation de leurs solutions. Les algorithmes dirigés par la complétude et la minimalité des prédicats et ceux dirigés par l'affinité ne possèdent pas de mesures qui permettent d'évaluer la pertinence des schémas générés.
De plus, les travaux qui ont traité cette problématique proposent des approches pour évaluer les bénéfices apportés par l'utilisation d'un algorithme par rapport à un autre, dans la phase logique et selon une fonction de coût, qui porte généralement sur le temps d'exécution des requêtes. On cite plus particulièrement les travaux de Bellatreche et Boukhalfa [BOK08] qui consistent à évaluer, lors de la phase de conception et selon un modèle de coût, la pertinence des schémas de fragmentation qui seront générés (voir section 3.5.2.3 du chapitre 3). Il n'existe pas, à notre connaissance, une approche qui permet d'évaluer l'optimalité du schéma de fragmentation déjà implémenté et de mesurer la pertinence des fragments en tenant compte, notamment, des changements de la charge de travail.
Dans cette contribution nous proposons une approche formelle d'évaluation d'un schéma de fragmentation, c'est-à-dire la pertinence des fragments générés, en se basant sur les accès des requêtes les plus fréquentes.

5.2 Principe

Etant donné que toute approche de fragmentation se conçoit à partir d'informations portants sur les fréquences d'accès des requêtes aux données. L'idée consiste à utiliser ce facteur commun entre toutes les approches pour définir une fonction de coût permettant d'évaluer un schéma de fragmentation selon les accès des requêtes aux différents fragments. Pour ce faire, nous adaptons une technique d'estimation utilisée dans le domaine de la statistique, à savoir le calcul de l'erreur quadratique (*Square-Error*). Le calcul de cette fonction permet de comparer les fragments selon la pertinence de leurs attributs, en terme de fréquence d'accès des requêtes [DEH14a].

La formulation utilisée, est inspirée des travaux de Jain et Dubes [JAA98] qui l'ont utilisé dans le cadre d'une méthode d'apprentissage non supervisée pour le regroupement d'attributs.

Pour des raisons de simplification et d'expérimentation, on considère dans notre cas un entrepôt de données évoluant dans un contexte local et dont lequel la table des faits est fragmentée selon une approche verticale. Nous considérons que les tables de dimension sont de petite taille et, par conséquent, elles ne seront pas fragmentées. De plus, notre approche ne considère pas une matrice d'affinité d'attributs, mais une matrice d'usage d'attributs composée, en colonnes des attributs et en lignes des requêtes fréquentes. Les termes de la matrice sont les fréquences d'accès des requêtes aux attributs.

5.3 La fonction coût pour l'évaluation d'un schéma de fragmentation

Soit la formulation suivante :

n : nombre total des attributs de la table de faits ;

Q : nombre total des requêtes fréquentes ;

fq : fréquence d'accès de la requête q pour $q = 1,2,.., Q$;

M : nombre total des fragments ;

n_i : nombre d'attributs dans le fragment i ;

f_{qj}^{i}: la fréquence d'accès de la requête q à l'attribut j dans le fragment i, avec $f_{qj}^{i} \neq 0$;

A_{ij} : le vecteur attribut de l'attribut j dans le fragment i, où f_{qj}^{i} est une composante de ce vecteur ;

S_{iq} : l'ensemble d'attributs du fragment i accédé par la requête q ; égale à 0 si la requête q n'accède pas au fragment i ;

$| S_{iq} |$: nombre d'attributs du fragment i accédé par la requête q ;

Soit une table de faits F de n attributs fragmentée verticalement en M fragments (F_1, F_2, ..., F_M) contenant chacun n_i attributs. Le vecteur moyen Vi pour le fragment i est défini par :

$$V_i = \frac{1}{n_i} \sum_{j=1}^{n_i} A_{ij} \qquad 0 < i < M \tag{5.1}$$

Le vecteur moyen V représente la moyenne des accès des requêtes à tous les attributs du fragment i. Pour un vecteur d'attributs A_{ij}, (A_{ij} - V_i) est dénommé « le vecteur différence » de l'attribut j dans le fragment i. L'erreur quadratique pour le fragment F_i est la somme des carrées de la longueur des vecteurs différences de tous les attributs dans le fragment i. Il est calculé par la formule suivante :

$$e_i^2 = \sum_{j=1}^{n_i} (A_{ij} - Vi)^Q \, (A_{ij} - Vi) \qquad 0 < i < M \tag{5.2}$$

Si $A_{ij} = V_i$ alors $e_i^2 = 0$. Ce cas signifie : soit qu'il y a un seul attribut dans chaque fragment soit que tous les attributs, dans chaque fragments, sont nécessaires pour l'exécution de la requête. Dans notre approche, on s'intéresse au cas où $A_{ij} \neq V_i$ afin de pouvoir comparer les fragments selon la pertinence des attributs.

L'erreur quadratique du schéma de fragmentation globale est calculée par la formule suivante :

$$E_M^2 = \sum_{i=1}^{M} e_i^2 \tag{5.3}$$

D'où : $E_M^2 = \sum_{i=1}^{M} \sum_{j=1}^{n_i} (Aij - Vi)^Q \, (Aij - Vi)$ (5.4)

Aussi une autre écriture de l'équation (5.4) permettra de mieux percevoir la contribution qu'apporte chaque requête sur le calcul de l'erreur quadratique pour chaque fragment. Ainsi, le vecteur moyen V_i pour le fragment *i* peut être défini comme suit :

$$Vi = \begin{bmatrix} \frac{|s_{i1}| * f_1}{n_i} \\ \frac{|s_{i2}| * f_2}{n_i} \\ \dots\dots \\ \dots\dots \\ \frac{|s_{iq}| * f_q}{n_i} \end{bmatrix}$$

Le vecteur attribut A_{ij}, est le suivant :

$$A_{ij} = \begin{bmatrix} f_{1j}^i \\ f_{2j}^i \\ \dots\dots \\ \dots\dots \\ f_{qj}^i \end{bmatrix}$$

D'où : $E_M^2 = \sum_{i=1}^{M} \sum_{j=1}^{n_i} \left[f_{1j}^i - \frac{|s_{i1}| * q_1}{n_i}, \dots, f_{qj}^i - \frac{|s_{it}| * q_t}{n_i} \right] \begin{bmatrix} f_{1j}^i - \frac{|s_{i1}| * f_1}{n_i} \\ f_{2j}^i - \frac{|s_{i2}| * f_2}{n_i} \\ \dots\dots \\ \dots\dots \\ f_{qj}^i - \frac{|s_{it}| * f_q}{n_i} \end{bmatrix}$ (5.5)

Afin d'identifier les différents composants des accès aux attributs qui ne sont pas pertinents pour une requête, l'équation peut être écrite comme suit :

$$E_M^2 = \sum_{i=1}^{M} \sum_{j=1}^{ni} \sum_{q=1}^{Q} \left[\delta_{jt} * f_q^2 \left(1 - \frac{|s_{iq}|}{n_i}\right)^2 + \left(1 - \delta_{jq}\right) \left(f_q * \frac{|s_{iq}|}{n_i}\right)^2 \right] \tag{5.6}$$

Où : $\delta_{jq} = 1$ si l'attribut *j* est accédé par la requête *q* ; = 0 si l'attribut *j* n'est pas accédé par la requête *q* ; Le premier terme $f_q^2 \left(1 - \frac{|s_{iq}|}{n_i}\right)^2$ représente les accès à l'attribut pertinent et le second terme représente les accès à l'attribut impertinent.

Alors :

$$E_M^2 = \sum_{i=1}^{M} \sum_{q=1}^{Q} \left[|s_{iq}| * f_q^2 \left(1 - \frac{|s_{iq}|}{n_i}\right)^2 + \left(n_i - |s_{iq}|\right) \left(f_q * \frac{|s_{iq}|}{n_i}\right)^2 \right] \quad (5.7)$$

Où :

$$\sum_{j=1}^{ni} \delta_{jq} = |s_{iq}| \quad \text{et} \quad \sum_{j=1}^{ni} (1 - \delta_{jq}) = \left(n_i - |s_{iq}|\right)$$

On aura donc :

$$E_M^2 = \sum_{i=1}^{M} \sum_{q=1}^{Q} \left[f_q^2 * |s_{iq}| \left(1 - \frac{|s_{iq}|}{n_i}\right)^2 + f_q^2 * \left(n_i - |s_{iq}|\right) \left(\frac{|s_{iq}|}{n_i}\right)^2 \right] \quad (5.8)$$

Si $n_i = |s_{iq}|$, alors $E_M^2 = 0$, ce qui implique qu'à chaque exécution, la requête q accède à tous les attributs du fragment i. On peut toujours réduire l'équation ci-dessus comme suit :

$$E_M^2 = \sum_{i=1}^{M} \sum_{q=1}^{Q} \left[f_q^2 * |s_{iq}| \left(1 + \frac{|s_{iq}|^2}{{n_i}^2} - 2 * \frac{|s_{iq}|}{n_i}\right) + f_q^2 * |s_{iq}| \left(n_i - |s_{iq}|\right) \left(\frac{|s_{iq}|}{{n_i}^2}\right) \right] \quad (5.9)$$

Par simplification de l'équation ci-dessus, on aura :

$$E_M^2 = \sum_{i=1}^{M} \sum_{q=1}^{Q} \left[f_q^2 * |s_{iq}| \left(1 - \frac{|s_{iq}|}{n_i}\right) \right] \quad (5.10)$$

Cette équation est la même que l'équation (5.4), sous une autre forme. On peut donc percevoir d'après cette équation l'apport des attributs impertinents pour le calcul de E_M^2 ; $\left(1 - \frac{|s_{iq}|}{n_i}\right)$ porte sur les attributs impertinents concernés par la requête q.

Ceci, signifie que la valeur de E_M^2 est proportionnelle au coût dû à l'accès aux fragments contenants des attributs non pertinents. Plus la valeur de cette erreur se rapproche de 0 plus le schéma de fragmentation est optimal.

5.4 Conclusion

Dans cette contribution, nous avons présenté une approche pour l'évaluation des fragments d'un entrepôt de données partitionné selon une approche verticale. Nous avons utilisé le calcul de l'erreur quadratique pour évaluer la pertinence d'un fragment par rapport à un autre. Dans notre approche, nous avons considéré l'impertinence données, c'est-à-dire les données qui ne sont pas fréquemment utilisées par les requêtes, come critère d'évaluation des fragments implémentés.

La validation expérimentale de notre proposition est présentée dans la section 7.4 du chapitre 7.

Chapitre 6

Les histogrammes pour une fragmentation dynamique des données

6.1 Introduction

Les algorithmes de conception d'un schéma de fragmentation optimal nécessitent des calculs combinatoires des probabilités des accès des requêtes, les plus fréquentes, pour déterminer une stratégie de fragmentation des données. Ce sont des algorithmes complexes et statiques en cas de changement dans leurs entrées ces derniers doivent être réexécutés. Dans le contexte des entrepôts de données, ces algorithmes génèrent des schémas de fragmentation qui ne seront pas les plus optimaux. Et ce, à cause particulièrement, de l'évolution de la charge de travail et des caractéristiques des requêtes OLAP.

Pour une exploitation efficiente des techniques de fragmentation, il ne s'agit pas seulement d'analyser les fréquences d'accès aux données pour choisir un schéma de fragmentation optimal, mais de rendre ce choix dynamique et adapté aux changements de la charge de travail.

Dans la littérature beaucoup de travaux ont abordé cette problématique, particulièrement dans le contexte des bases de données objets et relationnelles. Ces travaux se sont concentrés sur l'aspect physique par le développement d'outils automatisant le processus de distribution des données en cas de détérioration des performances.

Dans ce qui suit, nous allons présenter certains des travaux qui ont traité ce problème.

Travaux de Stöhr (2001), l'auteur a proposé, pour l'automatisation de la fragmentation et le placement des données d'un entrepôt dans un environnement parallèle, un outil dénommé *WARLOCK*. Le principe de cet outil est de déterminer une allocation des disques qui optimise les entrées sorties lors des accès à la table des faits et ce par l'utilisation du traitement parallèle des requêtes [STE01].

Travaux de Karahoca et al (2002), ont proposé un algorithme non linéaire basé sur les réseaux de neurones afin de détecter automatiquement l'approche de fragmentation la plus adaptée à une base de données distribuée [KAA02].

Travaux de Papadomanolakis et Ailamaki (2004), ont proposé l'automatisation de la phase de conception logique de la fragmentation à partir des informations récentes sur l'exploitation des données. Leur outil dénommé *AutoPart*, fragmente les tables selon une charge de travail actualisée. *AutoPart* reçoit en entrée les informations de la charge

de travail et conçoit un nouveau schéma de fragmentation selon une approche verticale [PAP04].

Travaux d'Alexandre et al. (2004), les auteurs ont proposé, pour le traitement efficace des requêtes OLAP, une approche appelée *partitionnement adaptative virtuelle* qui permet d'ajuster dynamiquement les tailles des partitions, sans faire appel aux informations relatives à la base de données [ALA04].

Enfin, plusieurs travaux ont proposé l'automatisation de la conception logique de la fragmentation des données pour le traitement du problème de l'évolution de la charge [AGS04, AGS06, SAA04, RAJ02].

Notre proposition au problème de l'évolution de la charge de travail et son impact sur la conception et l'implémentation d'un schéma de fragmentation optimal dans le contexte des entrepôts de données, consiste en une approche de fragmentation dynamique des données basée sur des informations statistiques récentes. Nous développons, à ce titre, un algorithme itératif basé sur la sélectivité des données et nous introduisons pour sa mise œuvre l'utilisation des histogrammes. Le choix de cette structure est justifié par le fait que les histogrammes :

- peuvent contenir assez d'informations issues des modèles des accès ;
- permettent un traitement efficace des mises à jour, sachant que dans les entrepôts de données ces dernières sont très fréquentes ;
- ne nécessitent pas beaucoup d'espace mémoire ;
- permettent de manipuler avec souplesse les informations conservées, c'est-à-dire de supprimer facilement l'ancienne historique et de garder uniquement le nouveau historique ;
- sont faciles à implémenter.

Dans les sections qui suivent, nous présentons le principe de notre approche et nous détaillons par la suite notre approche de fragmentation dynamique des données.

6.2 Principe

Le principe de notre approche se base sur l'observation et l'enregistrement des informations statistiques relatives aux accès aux données. La sauvegarde de ces informations s'effectue d'une manière continue, en écartant périodiquement les anciennes informations pour que la réfragmentation ne porte que sur des informations récentes. Les composants essentiels de notre approche sont: le critère d'évaluation et la structure de données permettant de conserver et d'observer les accès aux données [DEH09], [DEH13].

Le critère d'évaluation permet d'estimer l'opportunité ou non de l'exécution de la réfragmentation. Pour une raison d'adaptabilité, ce critère pourra être la minimisation d'une fonction objectif qui portera sur le coût d'accès aux données, le coût de transfert

des données entre fragments ou, dans un contexte distribué, sur le coût de communication. Il peut être également une valeur d'un seuil alloué au temps d'exécution d'une requête décisionnelle.

En ce qui concerne la conservation et l'exploitation des informations statistiques, elle est réalisée par l'utilisation des histogrammes. Ces derniers sont un moyen flexible pour la construction de structures sommaires pour les grandes bases de données. Leur utilité a été vérifiée dans de nombreux domaines, tel que l'optimatisation des requêtes [POV96, BRN02], les réponses approchées aux requêtes (*Approximative Query Answering*) [ACS99]. Dans le cadre des entrepôts de données, les histogrammes ont été principalement utilisés pour l'optimisation des requêtes OLAP [POV99] ou pour donner des réponses rapprochées aux requêtes complexes [EAT08].

Dans [IOY03], l'auteur dresse une taxinomie détaillée de l'historique et les domaines d'utilisation des histogrammes.

D'une manière générale, le principe de base de l'utilisation des histogrammes dans le cadre de l'optimisation des requêtes repose sur la connaissance des statistiques portant sur les objets manipulés. Les histogrammes permettent d'offrir une vision réelle de la distribution des données d'une colonne par une meilleure estimation de leurs sélectivités Ces estimations sont utilisées par l'optimiseur pour évaluer les plans de requêtes.

Exemple 6 : *Utilisation des histogrammes pour l'estimation des cardinalités*.

*Considérant la requête suivante : Select * from R where R.a < 20 et supposant qu'on a un histogramme en R.a. Pour estimer la cardinalité de chaque requête, on considère, alternativement, tous les buckets (appelé également groupe, plage ou fraction) de l'histogramme qui sont complètement ou partiellement couvert par le prédicat et on agrège par la suite tout les résultats intermédiaires. Cette procédure est illustrée ci-dessous :*

Considérons quatre buckets de l'histogramme sur le l'attribut R.a. le bucket b1, par exemple, couvre $0 \leq x < 10$ et sa fréquence est 100 (qui représente 100 tuples dans l'ensemble de données). De la même manière, les buckets b2, b3 et b4, représentent respectivement 50, 80 et 100 tuples. Supposons qu'on souhaite estimer la cardinalité du prédicat P=R.a < 20. Etant donné que P est totalement inclus dans le bucket b1, on peut garantir que les 100 tuples de b1 vérifie le prédicat P. De plus, P est disjoint du groupe b3 et b4 , de ce fait, il n'existe aucun tuple dans b3 et b4 vérifiant le prédicat P. Et enfin, le prédicat P est partiellement couvert par le bucket b2 (P est vérifié par 50 % des valeurs distinctes uniformément propagées dans b2. on utilisant l'hypothèse, citée si-dessus, on estime que 50% des tuples de b2 vérifies P. En conclusion, le nombre de tuples vérifiant le prédicat P = R.a < 20 est estimé à : 100+50/2 = 125.

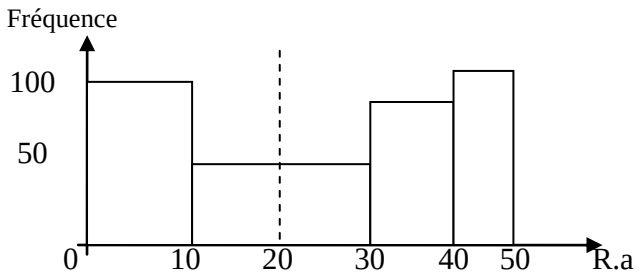


Figure 6.1- Estimation de la sélectivité d'une requête utilisant les histogrammes.

Les histogrammes sont utilisés par la plupart des SGBDs commerciaux. Dans la version 10g du SGBD Oracle, c'est le package DBM_STAS, composé de plusieurs procédures telle que : GATHER_TABLES_STAT, GATHER_SHEMA_STAT, INDEX_STAT, qui permet d'avoir des statistiques sur les objets de la base de données. Les statistiques qui portent sur une colonne d'une table sont stockées sous forme d'histogrammes. Oracle utilise deux types d'histogrammes : les histogrammes de hauteur équilibrée (*height-balanced histograms*) et les histogrammes de fréquence (*frequency histograms*).

Notre approche de réfragmentation des données s'effectue en quatre phases :

1- Fragmenter horizontalement l'entrepôt de données ;
2-Implémenter les histogrammes pour l'enregistrement des accès aux différents fragments ;
3- Lancer le processus d'évaluation ;
4- Réfragmenter l'entrepôt de données, si condition satisfaite, sur la base des statistiques recueillies.

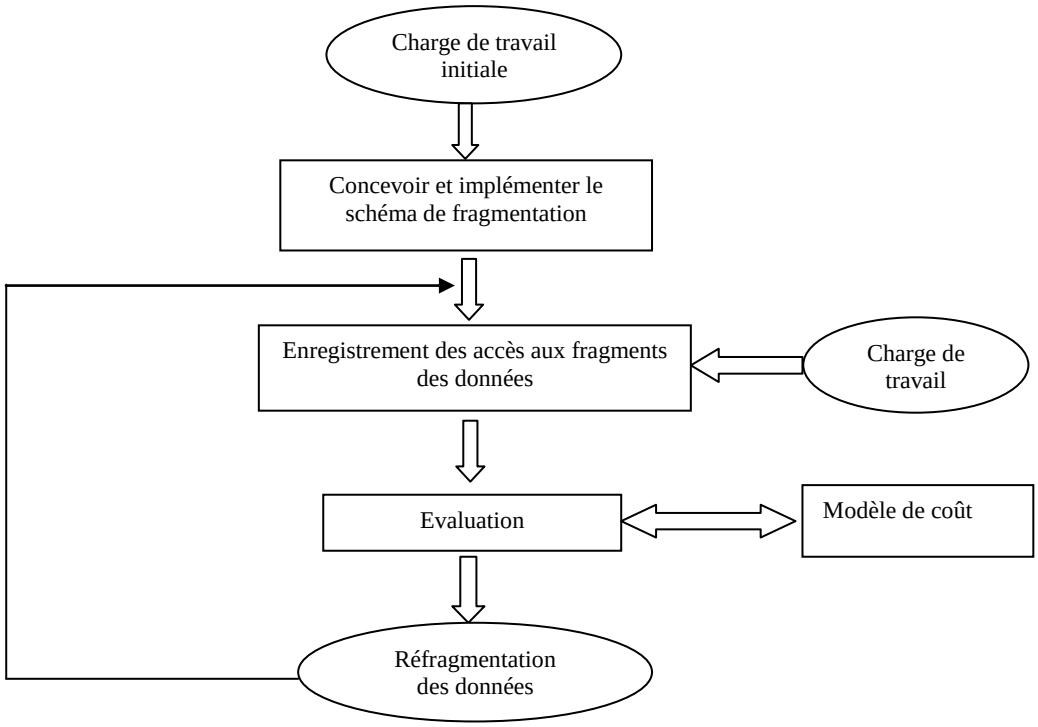


Figure 6.2- Approche de fragmentation dynamique des données

6.3 Approche de fragmentation dynamique des données

6.3.1 Fragmentation de l'entrepôt de données

Considérons un entrepôt de données constitué d'un schéma en étoile avec une table des faits *F* reliée à *N* tables de dimensions $\{d_1, \ldots, d_N\}$. Pour des raisons de simplification, on fragmente uniquement la table des faits *F* selon une approche horizontale par intervalle (*Range*). Le fragment *i* de la table des faits est noté *fi*. Soit *D* le domaine de valeur de l'attribut de fragmentation. Chaque fragment couvre un intervalle du domaine de l'attribut, qu'on va le dénommer Domaine de la Valeur du Fragment *DVF*. Le *DVF* d'un fragment *fi* est : $DVF(fi) = f_i[min_i, max_i]$. Aussi, deux fragments f_i, f_j sont adjacents si leur *DVF* sont contigus, c'est-à-dire :

$$Adj\ (fi\ ,fj) \Longrightarrow max_i = min_j \vee max_j = min_i$$

D'autre part, deux ou plusieurs fragments adjacents peuvent être regroupés en un nouveau fragment f_{nouv} si le *DVF* du nouveau fragment égale à la somme des *DVF* des fragments précédents :

$$f_{nouv} = \cup_{i=1}^{n} f_i$$

$$\forall f_i \in \{f_1, \ldots, f_n\} \exists \left(f_j \in \{f_1, \ldots, f_n\} \right) Adj\left(f_i, f_j \right)$$

6.3.2 Implémentation des histogrammes pour la sauvegarde des accès aux données

6.3.2.1 Création des histogrammes

Pour chaque clé étrangère FR_k de la table des faits un histogramme est construit. Un histogramme sur l'attribut de fragmentation se compose d'un ensemble de buckets. Chaque bucket b_k correspond à un fragment, c'est-à-dire chaque bucket couvre un domaine de valeur de fragment $DVF(f_i)$. Dans notre approche nous avons opté pour l'utilisation de l'histogramme de largeur égale (*equi-width histogram*). Ces derniers, sont plus simples à accéder et à mettre en œuvre. La distribution des données pour ce type d'histogramme est uniforme. De plus, les buckets ont la même taille, ce qui cadre parfaitement avec l'approche de fragmentation que nous avons utilisée, à savoir l'approche horizontale par intervalle.

Pour permettre de prendre en compte, lors de la réfragmentation des données, uniquement l'historique récent, on utilise deux jeux d'histogrammes : l'ancien et le nouveau jeu. Au départ, toutes les informations sont enregistrées dans l'histogramme récent. A chaque exécution de l'algorithme de réfragmentation, l'ancien jeu est effacé et les jeux d'histogrammes sont échangés. Cela signifie que le jeu récent détient les opérations enregistrées depuis la dernière fois que l'algorithme a été exécuté. Alors que l'ancien jeu conserve les opérations enregistrées entre les deux dernières exécutions.

Pour manipuler les histogrammes, nous utilisons les notations ci-après : l'histogramme pour chaque fragment est noté H_{FRk} contenant b_k buckets. Le nombre de buckets est $B_i[b_k]$. T_b est la taille des buckets. Elle correspond au nombre des lignes couvert par le bucket. La limite de l'intervalle de valeurs commence et se termine par un multiple de T_b. Cela signifie que la valeur de l'intervalle couverte par le bucket est $B_i[b_k] = [b_k * T_b, (b_{k+1}) * T_b]$.

Pour limiter l'utilisation de la mémoire, il existe un nombre maximum de buckets MAX_B à sauvegarder (dans Oracle 10g le nombre maximum de buckets est limité à 225).

Comme indiqué, ci-dessus, pour chaque clé étrangère deux jeux d'histogrammes sont construits : l'ancien histogramme $H_{anc}^{FR_k}$ et le nouveau histogramme $H_{rec}^{FR_k}$. Dans une première étape, toutes les informations sont enregistrées dans l'histogramme récent. La figure 6.3 schématise la création des histogrammes.

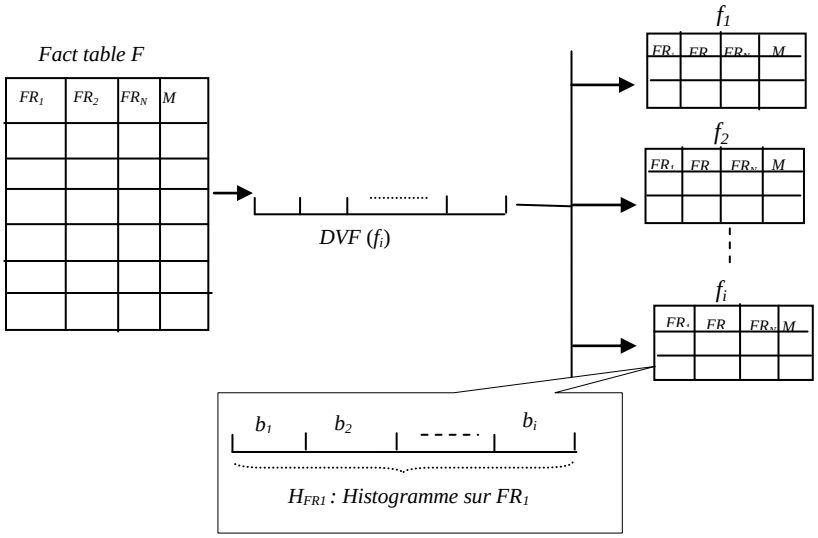


Figure 6.3- Création des histogrammes

6.3.2.2 Critère d'évaluation

Le composant principal de notre approche est le critère d'évaluation qui permet de décider sur l'opportunité ou non d'exécuter le processus de réfragmentations. Il peut être une fonction coût à définir ou un seuil de temps d'exécution de requête fixé, c'est-à-dire si le temps d'exécution d'une requête dépasse ce seuil on lance le processus de réfragmentation des données. Dans notre approche nous avons choisi comme critère d'évaluation, un seuil du temps de traitement des requêtes qui sera fixé d'une manière empirique.

Notre approche de réfragmentation de l'entrepôt de données consiste à déterminer un schéma de fragmentation tel que le coût total de l'exécution des requêtes sera inférieur au temps seuil préalablement fixé. Il convient de noter, que cette réfragmentation des données est testée dans un environnement centralisé. Le coût de communication ne sera pas considéré (voir section 7.3 du chapitre 7).

6.4 Algorithme de réfragmentation des données

Le principe de l'algorithme de réfragmentation, décrit ci-dessous, consiste à examiner les accès à chaque fragment et évalue une possible réfragmetation en se basant sur l'historique récent des accès. Il s'agit donc d'identifier les parties des fragments qui doivent être extraites pour former un nouveau fragment :

$$DVF(f_{inouv}) = f_{nouv}\ [min_{nouv}, max_{nouv}].$$

D'une manière formelle, soit un schéma de fragmentation *Sf* avec un ensemble de fragments f_i avec $DVF(f_i) = f_i\ [min_i, max_i]$. Le coût d'accès à une donnée à l'instant *t* dépend du schéma de fragmentation Sf_t. Il s'agit de déterminer, à l'issue d'une dégradation des performances, un nouveau schéma de fragmentation optimal Sf_{nouv} composé d'un ensemble de fragments $f_m,\ldots,f_n$, avec $\cup f_m, \ldots, f_n = F$.

Initialement les deux jeux d'histogrammes, récent et ancien, sont construits pour chaque clé étrangère FR_k de la table des faits. A chaque accès aux données d'un fragment le bucket correspondant est mis à jour par l'exécution de la fonction *HistogramUpdate()*. A un instant donné, le critère d'évaluation est estimé. Si ce critère n'est pas satisfait, on garde le schéma de fragmentation actuel. Sinon, la réfragmentation est exécutée et on aura un nouveau schéma de fragmentation Sf_{nouv} composé d'un ensemble de fragments $f_m,\ldots,f_n$, avec $\cup f_m, \ldots, f_n = F$ dont le nombre ne devra pas dépasser le seuil MAX_B.

A l'issue d'une réfragmentation des données, la fonction *HistogramOrganize()*,décrite dans la section 6.4.1.2, est exécutée pour réorganiser et redimensionner les buckets des deux histogrammes selon le nouveau schéma de fragmentation.

Entrées *: - Fragmenter la table des faits F selon le schéma de fragmentation Sf.*
- Création des histogrammes pour toutes les clés étrangères : $H_{anc}^{FR_k}$, $H_{rec}^{FR_k}$
Sortie *: Nouveau schéma de fragmentation optimal schéma* Sf_{nouv}
Début
Pour tous $B_i[b_k] \in H_{rec}^{FR_k}$ **faire**
HistogramUpdate () // calcul de la sélectivité //
Si *Critère d'évaluation est satisfait* **Alors**
Refragmentation de F
HistogramRorganize() // Réorganisation et redimensionnement des bcukets //
Sinon *Garder le schéma de fragmentation existant*
Fin Si
Fin Pour
Fin

Algorithme 9- Algorithme de réfragmentation des données

6.4.1 Opérations sur les histogrammes

6.4.1.1 Mise à jour des histogrammes

A chaque accès aux fragments des données f_i, la sélectivité du bucket concerné est mise à jour. Soit un fragment f_i avec $DVF(f_i) = f_i\ [min_i, max_i]$ et un tuple t_j et v_j la valeur qui correspond au numéro de la ligne ombre de lignes du tuple dans la table des faits. Etant donné que l'intervalle de valeur du bucket b_k est : $[b_k * T_b\ , (b_{k+1}) * T_b\ ]$

Le traitement consiste à déterminer le bucket b_k qui contient le tuple qui a été accédé selon la valeur de v_j et à incrémenter le nombre de ses accès noté $SEL[b_k]$. La formule est alors : $b_k = v_j / T_b$.

Exemple 7*: On suppose $T_b = 5$ et $v_j = 30$, le bucket affecté par cette sélectivité est : 30 / 5 = 6 (cette valeur est toujours arrondie).*

La sélectivité pour n'importe quel attribut de la table des faits (dans notre cas les clés étrangères) est la somme des fréquences des accès ou la sélectivité des différents buckets de l'histogramme correspondant. Soit $SEL\ [FR_N]$ la sélectivité (fréquence des accès) à un attribut de la table des faits. La sélectivité est calculée par la formule :

$$SEL\ [FR_N] = SEL\ [H_{FRN}] = \sum_1^k SEL[b_k]$$

6.4.1.2 Réorganisation et redimensionnement des histogrammes

A l'issue d'une réfragmentation des données et dans le cas ou on veut garder le nombre et la taille des anciens fragments inchangé, c'est-à-dire $DVF\ (f_i) = f_i\ [min_i, max_i] = DVF\ (f_{nouv}) = f_{nouv}\ [min_{nouv}\ , max_{nouv}]$, alors la fonction de réorganisation des histogrammes, *HistogramReorganize(),* consistera uniquement à altérer les deux jeux d'histogrammes, c'est-à-dire, l'ancien devient récent et le récent jeu est mis à zéro (vidé). Avec :

$$H_{anc}[F] \leftarrow H_{rec}\ [F] \text{ et } H_{rec}[F] \leftarrow 0.$$

Dans le cas $DVF(f_{nouv}) \neq DVF\ (f_i)$ alors le nombre de buckets et leur taille pourra diminuer ou augmenter selon le nombre de buckets maximum MAX_B. Si une mise à jour rend le nombre de buckets sauvegardés au dessus de la valeur MAX_B, la largeur du bucket est diminuée proportionnellement par l'utilisation d'un facteur Z_T. De la même manière, si le nombre de buckets est petit, c'est-à-dire en dessous de MAX_B la largeur du buckets est augmentée par le même facteur. Ce qui permettra de s'assurer que le nombre de buckets correspond toujours au nombre de fragments et ce pour une meilleure capture de l'historique des accès aux données des fragments.

6.4.2 Etude de complexité de l'algorithme

Pour évaluer notre algorithme de réfragmentation, nous avons procédé par l'étude de complexité des différentes phases qui le compose, il en ressort que :

- le traitement de la mise à jour des histogrammes est O(1) donc une complexité constante. Les histogrammes sont gardés en mémoire principale, cette mise à jour n'engendre aucun accès disque.
- l'évaluation de la taille des fragments, pour le redimensionnement des histogrammes, est de complexité polynomiale O(n^2) ou n représente le nombre de buckets.
- la détermination d'un nouveau schéma de fragmentation est de complexité polynomiale.

6.5 Conclusion

Dans ce chapitre nous avons proposé une approche de fragmentation dynamique des données. Les caractéristiques des entrepôts de données, particulièrement l'évolution des données et de la charge de travail, influent sur les performances et par conséquent sur les techniques d'optimisation implémentées, particulièrement la fragmentation des données. Dans cette contribution, nous avons proposé d'intégrer l'aspect « dynamicité » dans la fragmentation des données. Pour ce faire, nous avons utilisé les histogrammes pour la sauvegarde de l'historique des accès afin d'estimer la sélectivité des données des différents fragments. Pour réfragmenter les données nous avons développé un algorithme itératif qui permet de déterminer un nouveau schéma de fragmentation optimal selon des informations statistiques récentes de l'exploitation des données.
Nous avons également évalué et validé cette proposition. Les expériences que nous avons menées sont détaillées dans la section 7.3 du chapitre 7.

PARTIE III : EXPERIMENTATION ET CONCLUSION GENERALE

*Cette partie sera consacrée à la présentation des résulats expérimentaux et à la conclusion. Nous avons mené plusieurs tests qui nous ont permis d'évaluer et de comparer nos approches par rapport à d'autres approches qui ont traité les mêmes problématiques. Ces expérimentations, décritent dans le chapitre 7, nous ont également permis de déceler les asepcts qui méritraient d'être améliorer dans nos travaux futurs. Nous présentons dans ce même chapitre l'environnement de simulation mis en place, les conditions expérimentales et les résullats obetnus à l'issus des tests effectués.*Enfin,*le dans chapitre 8 nous dressons le bilan de nos contributions et nos présentons nos prespectives de recherche.*

Chapitre 7

Validation expérimentales

7.1 Conditions expérimentales

7.1.1 L'entrepôt de données

L'évaluation, de nos approches relatives à l'utilisation de l'OEP pour la conception d'un schéma de fragmentation optimal et à la fragmentation dynamique des données, a été effectuée sur le benchmark APB-1 release II Council (1998) implémenté sous Oracle 10g. Ce benchmark utilise un schéma en étoile composé de quatre tables de dimensions (Prodlevel, Custlevel, Timelevel et Chanlevel) et une table de faits (Actvars). La figure 6.1 présente le schéma en étoile de ce benchmark.

La figure 7.1 et le tableau 7.1, présentent respectivement le schéma en étoile du benchmark APB-1 et les caractéristiques des tables de l'entrepôt de données.

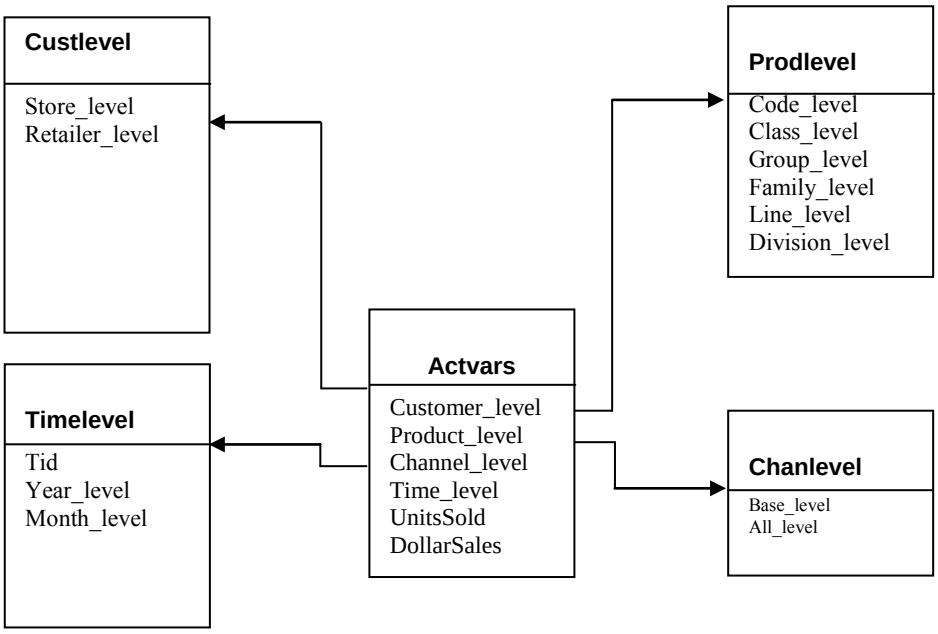


Figure 7.1 Schéma en étoile du benchmark APB-1 release II.

Table	Nombre d'enregistrements	Taille d'un enregistrement
Actvars	24 786 000	74
Prodlevel	9000	72
Custlevel	900	24
Timelevel	24	36
Chanlevel	9	24

Tableau 7.1 Caractéristiques des tables de l'entrepôt de données.

Le banc d'essais APB-1 est livré avec un fichier exécutable APB.exe. Ce fichier est utilisé pour générer les fichiers de données pour charger l'entrepôt. Il est livré aussi avec un fichier contenant les scripts de création des tables constituant l'entrepôt. Le Chargement de l'entrepôt a été réalisé à l'aide de l'utilitaire SqlLoader, fourni avec Oracle10g.

7.1.2 Charge de requêtes

Pour mener nos tests, nous avons utilisé 55 requêtes décisionnelles englobant différents opérateurs : opérations de jointure, de sélection et des fonctions de calcul et d'agrégations (SUM, COUNT, AVG, MIN, MAX).

7.1.3 Architecture matérielle et logicielle

Nos avons testé nos approches sur deux machines interconnectées (pour se rapprocher à une simulation centralisée, réduisant ainsi au maximum le temps de traitement des requêtes et de distribution des données). Nos algorithmes ont été implémentés avec Visual C++ sur une machine, i7 avec 4GB de RAM hébergeant également l'entrepôt de données.

Un premier programme a été implémenté au niveau de la machine cliente. Il permet l'envoi des requêtes vers le serveur. Un autre programme permet d'estimer la charge au niveau du serveur d'une façon périodique par l'exécution des appels systèmes.

L'utilisation de la bibliothèque de gestion des sockets a permis de bénéficier de l'utilité des sockets qui représentent un tunnel de communication inter-processus en permettant à divers processus de communiquer aussi bien sur une même machine, qu'à travers un réseau TCP/IP. La machine cliente peut lancer en parallèle une soixantaine de clients logiques (sous forme de processus indépendants). Chaque processus joue lui-même le rôle d'une machine cliente.

7.1.4 Identification des paramètres optimaux de simulation

Dans une première série de tests, nous avons procédé à la comparaison du partitionnement par intervalle, utilisée dans notre troisième contribution portant réfragmentation des données décrite dans le chapitre 6 de la 2ème partie, avec d'autres approches de fragmentation horizontale, en l'occurrence par hachage et par liste. Ceci, pour démontrer que l'approche de fragmentation horizontale par intervalle est la plus adaptée aux entrepôts de données et pour montrer, également, l'importance de la charge de travail et le choix de l'attribut de partitionnement lors de la conception d'un schéma de fragmentation optimal.

Nous avons commencé par fragmenter la table des faits selon l'approche horizontale par intervalle. La clé de partition peut être l'une des clés étrangère de la table des faits. Dans les instructions SQL, ci-après, nous avons choisi la clé étrangère Time-level comme clé de partitionnement.

```
CREATE TABLE Actavrs (
Customer_level   char(12) not null,
Product_level    char(12) not null,
Channel_level    char(12) not null,
Time_level       varchar(12) not null,
UnitsSold        float not null,
DollarSales      float not null,
DollarCost       float not null,
Foreign key (Customer_level) references custlevel
(store_level),
Foreign key (Product_level) references prodlevel
(code_level),
Foreign key (Channel_level) references chanlevel
(base_level),
Foreign key (Time_level) references timelevel (tid))

Partition by range (Time_level)
  (Partition Sales_q1_1995 values less than 199504,
   Partition Sales_q2_1995 values less than 199507,
   Partition Sales_q3_1995 values less than 199510,
   Partition Sales_q4_1996 values less than 199600,
   Partition Sales_q5_1996 values less than 199606);
```

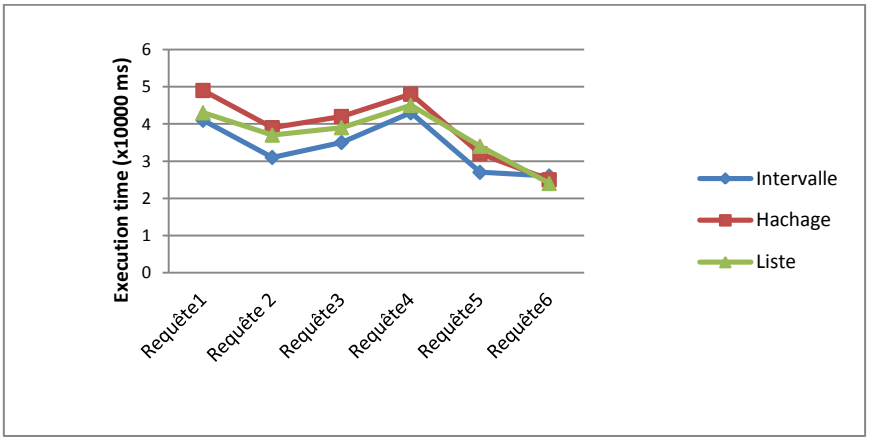


Figure 7.2 Evaluation des différentes techniques de fragmentation horizontales.

Tel que démontré dans la figure 7.2, ci-dessus, l'approche par intervalle est plus optimale en terme de temps de réponse des requêtes, par rapport aux autres approches de fragmentation horizontale. Cette approche est bénéfique particulièrement pour les entrepôts de données, étant donné que la plupart des requêtes OLAP sont exécutées pour analyser des mesures selon la dimension temps.

Pour montrer l'influence de la charge de travail et le choix de la clé de partitionnement, nous avons effectué plusieurs tests en procédant, dans une première étape, à la fragmentation de la table des faits selon différentes clés de partition en gardant la même charge de travail. La figure 7.3 illustre la clé de partitionnement.

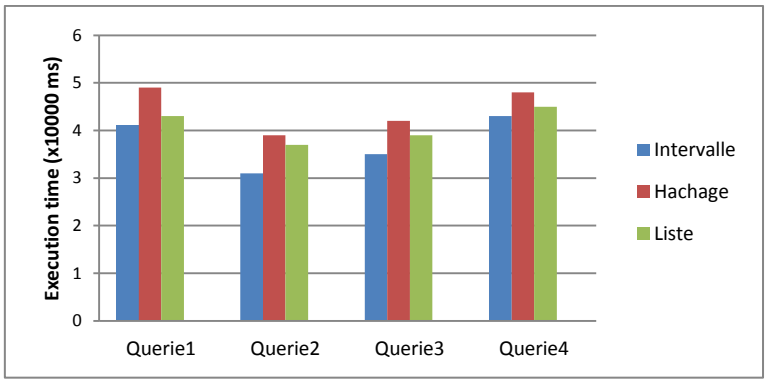


Figure 7.3 Influence de la clé de partition sur la fragmentation des données.

Dans une seconde étape, nous avons fragmenté la table des faits en prenant l'attribut Time-level de la table Timelevel comme clé de partition. Nous avons exécuté un ensemble de requêtes sur ce schéma de fragmentation, tout en procédant au changement des prédicats des requêtes afin que ces derniers portent sur l'analyse des données d'autres tables telles que : Custlevel et Prodlevel et non sur la table contenant la clé de

partition. La figure 7.4 montre l'influence de la charge de travail sur la stratégie de fragmentation.

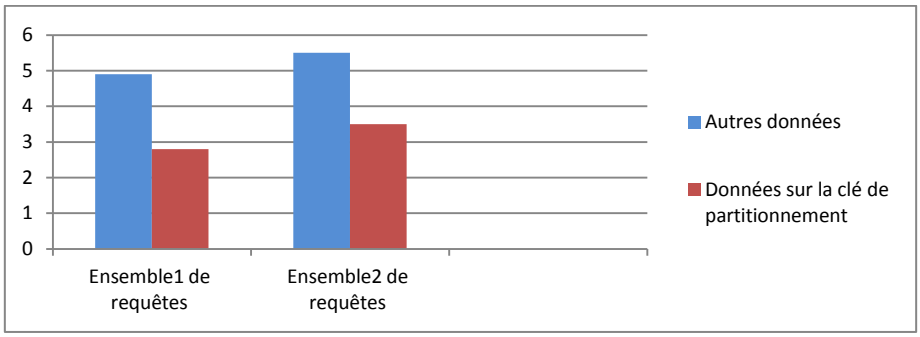


Figure 7.4 Influence de la charge de travail.

Dans ce qui suit, et pour des raisons de présentation, nos décrivons aborderons d'abords les tests expérimentaux réalisés sur les approches de sélection d'un schéma fragmentation optimal et sur l'approche de fragmentation dynamique des données, étant donné de ces dernières utilisent la même plateforme de test. La validation expérimentale de l'approche d'évaluation de la pertinence d'un schéma de fragmentation sera présentée en dernier.

7.2 Expérimentation des approches OEP-FHP, OEP-FHD et OEP-FV

Les paramètres qui ont été définis pour nos tests expérimentaux sont présentés dans le tableau ci-après :

Paramètres	**Valeurs**
Nombre d'attributs	9
Nombre de prédicats	35
Nombre d'itération	1500
c_1 et c_2	1.6319 et 0.6239
r_1 et r_2	Dans [0,1]
W (Nombre maximum de fragments)	Nombre de fragments entre 50 et 100
w (facteur d'inertie)	0.6571

Tableau 7.2- Paramètres des algorithmes

Il est à rappeler que les coefficients et les facteurs d'inertie sont prédéfinis par des expérimentations effectuées sur l'OEP TRIBE.

7.2.1 Expérimentation de l'approche OEP-FHP

Pour effectuer nos tests, nous avons commencé par fragmenter la table des faits selon une approche horizontale par intervalle. La clé de partition pourra être l'une des clés étrangères de la table des faits. Dans les instructions SQL, ci-dessous, nous avons choisi comme clé de partition l'attribut Time-level.

```
CREATE TABLE Actavrs (
Customer_level   char(12) not null,
Product_level    char(12) not null,
Channel_level    char(12) not null,
Time_level       varchar(12) not null,
UnitsSold        float not null,
DollarSales      float not null,
DollarCost       float not null,
Foreign key (Customer_level) references custlevel
(store_level),
Foreign key (Product_level) references prodlevel
(code_level),
Foreign key (Channel_level) references chanlevel
(base_level),
Foreign key (Time_level) references timelevel (tid))
Partition by range (Time_level)
  (Partition Sales_q1_1995 values less than 199504,
   Partition Sales_q2_1995 values less than 199507,
   Partition Sales_q3_1995 values less than 199510,
   Partition Sales_q4_1996 values less than 199600,
   Partition Sales_q5_1996 values less than 199606);
```

Tel que dans les sections précédentes, la fragmentation horizontale par intervalle est l'approche la plus adaptée aux entrepôts de données. Dans les premières séries de tests, nous avons procédé à la comparaison de notre approche, OEP-FHP, de fragmentation horizontale primaire, avec l'approche horizontale par intervalle, notée (RP). Nous avons, d'abord, exécuté une série de requêtes sur le schéma de fragmentation, décrit ci-dessus. Nous avons, par la suite, déterminé un nouveau schéma de fragmentation par l'exécution de notre approche de fragmentation et nous avons exécuté la même série de requêtes sur les fragments générés par l'approche OEP-FHP. La figure 7.5, ci-dessous, montre qu'il n'existe pas une grande différence, en terme de temps d'exécution de requêtes OLAP (ms), entre les deux approches, OEP-FHP et RP.

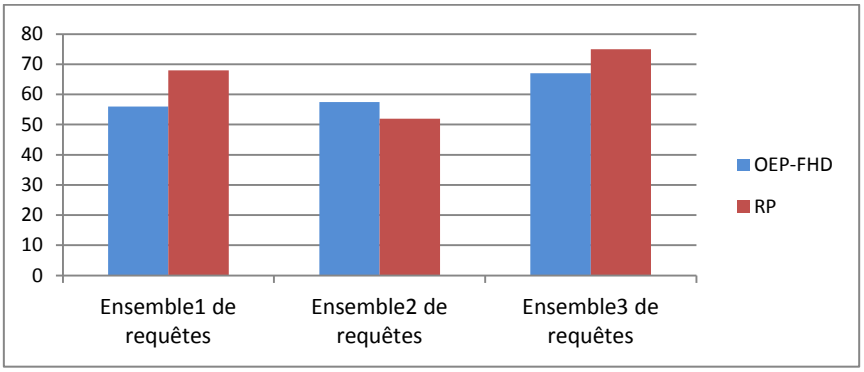


Figure 7.5- Comparaison de l'approche OEP-FHP avec l'approche RP

Dans une seconde étape d'expérimentation, nous avons procédé à la comparaison de l'approche OEP-FHP à d'autres approches classiques de conception d'un schéma de fragmentation, en l'occurrence l'approche par construction de prédicats, notée (PC) et l'approche basée sur l'affinité des attributs, notée (AB). En même temps, nous avons enregistré les résultats dans le cas ou l'entrepôt de données n'est pas fragmenté (NF). Les résultats sont obtenus, en nombre d'E/S disques, selon le modèle de couts présenté dans la section 4.4.3.3 du chapitre 4,

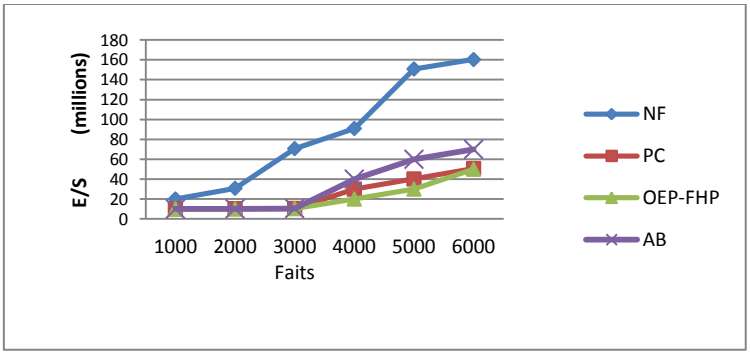


Figure 7.6- Comparaison des approches de fragmentation

On constate clairement, selon la figure 7.6, que la fragmentation des données améliore considérablement les performances par rapport à un entrepôt données non fragmenté. Quant à l'entrepôt de données monte en échelle, l'approche OEP-FHP présente des performances meilleures que les approches PC et AB. Aussi, l'OEP-FHP génère un nombre de fragments réduit par rapport aux autres approches classiques. Ceci permettra une meilleure maintenance des fragments et une amélioration significative des performances étant donné que la distribution des requêtes et le coût de reconstruction des résultats sont réduits.

Pour montrer le choix de l'attribut de fragmentation sur l'approche horizontale primaire, nous avons exécuté notre approche de fragmentation en variant à chaque

l'attribut de fragmentation. La figure 7.7, montre que la fragmentation de la table des faits en utilisant l'attribut *Time-level* présente les meilleures performances étant donné que cet attribut est le plus utilisé par les requêtes, et par conséquent c'est l'attribut qui renferme le plus grand nombre de prédicats.

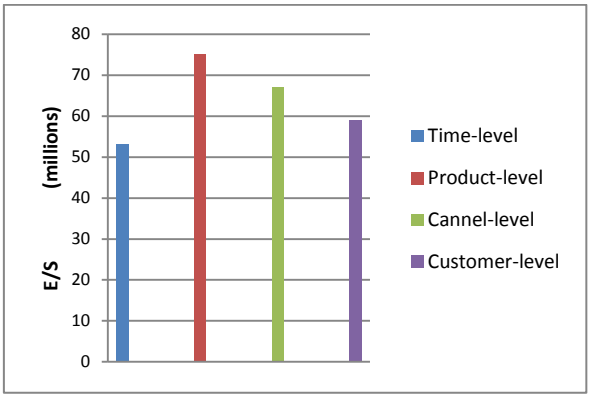


Figure 7.7- Effet du choix de l'attribut sur la fragmentation des données

7.2.2 Expérimentation de l'approche OEP-FHD

Nous avons commencé par tester l'algorithme OEP-FHD par rapport au seuil *W*, relatif au nombre maximum de fragments à générer, en le variant entre 50 et 100 fragments et fixant le nombre de prédicats de sélection à 35. La figure 7.8, montre, qu'à l'instar des autres approches, le coût d'exécution des requêtes sur un entrepôt fragmenté, selon l'approche OEP-FHD, diminue avec l'augmentation du nombre de fragments.

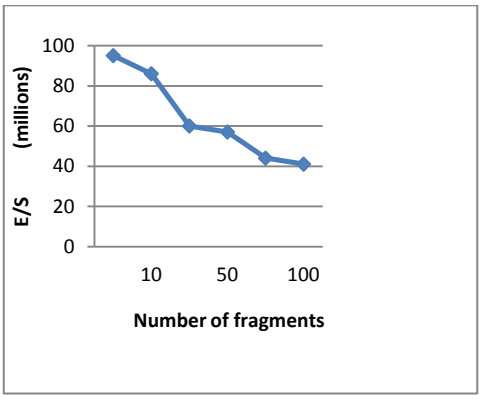


Figure 7.8- évaluation de l'OEP-FHD

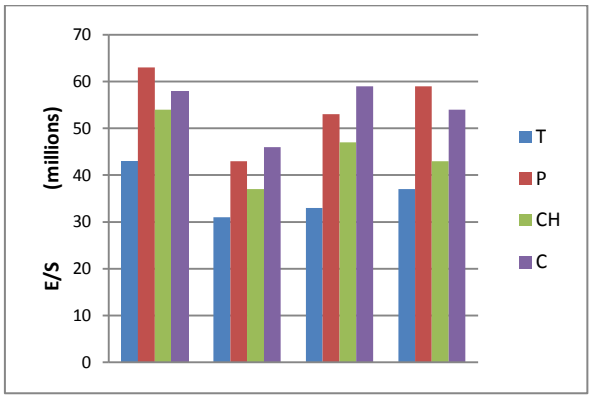


Figure 7.9- Choix de la table de dimension

Pour montrer l'impact du choix de la table de dimension sur la fragmentation horizontale dérivée, nous avons mené des tests en modifiant à chaque fois la table de dimension : prodlevel (P), Timelevel (T), Custlevel (C) et enfin Chanlevel (H). La figure 7.9 montre les résultats obtenus. Il en ressort, plus particulièrement, que la table le plus fréquemment utilisée par les requêtes sera la table la plus adaptée pour la fragmentation.

Selon les tables utilisées dans nos expérimentations, la table de dimension Timelevel est la table qui présente les meilleures performances, et par conséquent elle sera la table candidate pour la fragmentation sur laquelle s'effectuera la fragmentation de la table des faits.

Dans une seconde étape d'expérimentation, nous avons procédé à la comparaison de l'approche de fragmentation horizontale dérivée, OEP-FHD avec l'approche OEP-FHP, en choisissant pour la fragmentation primaire l'attribut *Time-level* comme attribut de fragmentation. La figure 7.10, illustre les résultats obtenus. L'approche de fragmentation primaire présentera les meilleures performances lorsque les requêtes, (l'ensemble des requêtes 2 et 3) utilisent beaucoup les prédicats de l'attribut de fragmentation de l'OEP-FHP, en l'occurrence *Time-level*. Si non l'approche dérivée est la plus adaptée.

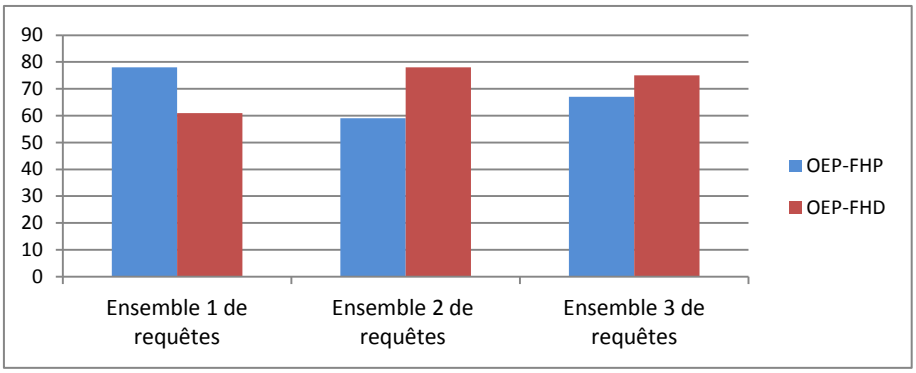


Figure 7.10- OEP-FHP Vs OEP-FHD

7.2.3 Expérimentation OEP-FV

L'implémentation de l'approche verticale a été réalisée à travers des vues matérialisées. Pour tester l'approche de fragmentation OEP-FV, nous avons commencé nos tests par l'examen des performances de l'algorithme para rapport au seuil W. nous avons exécuté l'algorithme OEP-FV en utilisant 9 attributs et nous avons, à chaque fois, procédé à la variation du seuil entre 10 et 100 fragments. Comme indiqué dans la figure 7.11, ci-dessous, à l'instar des autres approches le nombre de fragments à un impact sur les performances.

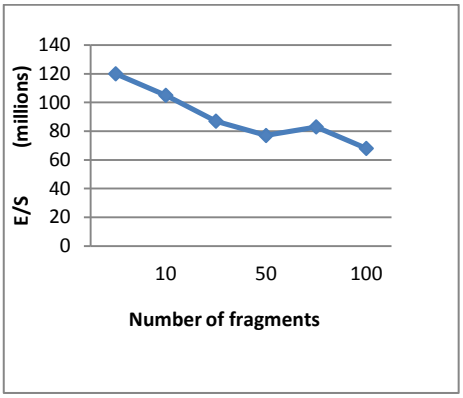


Figure 7.11- Evaluation de l'OEP-FV

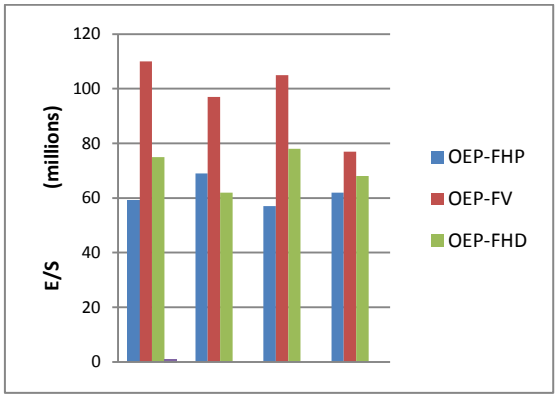


Figure 7.12- Comparaison des approches

Aussi on constate également que l'approche OEP-FV présente des performances médiocres par rapport aux approches horizontales, en l'occurrence OEP-FHP et OEP-FHD. Ceci est en adéquation avec ce qui a été déjà démontrer, soit pour les entrepôts de données ou les bases de données, que la fragmentation verticale est moins performante à cause du nombre élevé des opérations de jointures effectuées pour l'exécution des requêtes. La figure 7.12 illustre cette affirmation.

7.2.4 Conclusion

Dans cette section nous avons présenté des tests expérimentaux effectués sur nos trois approches de fragmentation, OEP-FHP, OEP-FHD et OEP-FV, utilisant l'Optimisation par Essaim Particulaire comme métaheuristique pour résoudre le problème de la sélection d'un schéma de fragmentation optimal. Pour des raisons de simplification, lors de la définition des paramètres des algorithmes, nous avons choisi l'OEP TRIBE pour l'implémentation de nos approches. A l'instar des autres approches qui ont traité la même problématique, il en ressort que le choix de l'attribut de fragmentation et les tables de dimension à fragmenter constituent les aspects les plus critiques sur lesquels dépendent fortement toutes les approches de fragmentation. Aussi, nos expérimentations ont également permis de percevoir que la fragmentation horizontale primaire, effectuée uniquement sur la table des faits, demeure la plus facile et la plus pratiques pour les entrepôts de données, malgré que dans certaines conditions d'exploitation des entrepôts de données, la fragmentation horizontale dérivée présente les meilleures performances. Ceci reste toujours conditionner par la limitation du nombre de fragments à générer et plus particulièrement du choix de la ou des tables de dimensions à fragmenter.

Le plus important de nos travaux futurs, consistera à poursuivre nos expérimentation en procédant à la comparaison des résultats obtenus, en termes de temps d'exécution des requêtes et en nombre de fragments générés, avec les travaux qui ont utilisé d'autres

types de métaheuristique, tel que les algorithmes génétiques. Il est également intéressent de continuer les essais expérimentaux en n'intégrant d'autres paramètres dans la fonction objectif, notamment ceux relatifs au contexte des entrepôts de données distribués.

7.3 Expérimentation de l'approche de fragmentation dynamique des données

Pour identifier les conditions optimales pour nos tests et pour déterminer le temps seuil d'exécution des requêtes, qui sera notre critère d'évaluation, pour la fragmentation dynamique, nous avons fixé une série de requêtes OLAP et nous avons varié le nombre de fragments pour observer son influence sur le temps de réponse de la charge de travail. Comme présenté dans la figure 7.13, ci-dessous, le nombre optimal de fragments se situe entre 50 et 100 et le temps maximum des requêtes OLAP varie entre 1,5 et 4 (s).

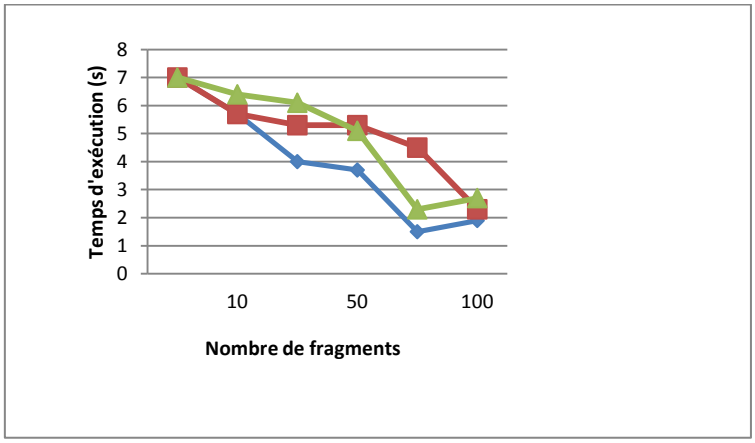


Figure 7.13. Définition du nombre optimal de fragments et le temps seuil d'exécution des requêtes OLAP.

7.3.1 Création et implémentation des histogrammes

Après avoir fragmenté la table des faits selon une approche horizontale par intervalle, nous avons procédé à la création des jeux d'histogrammes, sous le SGBD Oracle 10g, pour chaque attribut de la table des faits en utilisant la procédure DBMS_STATS.GATHE décrite ci-après :

```
BEGIN
     DBMS_STATS.GATHER_TABLE_STATS
     (OWNAME=>'OE',TNAME='ACTAVRS',
      METHOD_OPT=>'FOR COLUMNS SIZE 10 TIME_LEVEL');
END
```

Cette procédure permet la création d'histogrammes qui ont 10 buckets. Si nous nous ne précisons pas la taille, par défaut 75 buckets seront utilisés. Par exemple, avec les instructions SQL, ci-après, les histogrammes sont crées avec 75 buckets pour toutes les colonnes indexées de la table Actvars :

```
  ANALYZE TABLE ACTVARS  COMPUTE STATISTICS FOR ALL
INDEXED COLUMNS;
  DBMS_STATS
```

Il est à noter, que la commande ANALYZE TABLE est une autre manière de créer les histogrammes. Aussi, dans le SGBD oracle 10g le nombre maximum de buckets est de 254.

Pour vérifier que les histogrammes ont été crées pour la table des faits fragmentée et pour visualiser les statistiques des fréquences des histogrammes, nous avons utilisé l'instruction suivante :

```
  Select column_name, partition_name, num_bucket, histogram
  From Dba_part_col_statistics
  Where  table_name='actavrs',  AND  column_name='time-levelAND
         Partition_name ='Sales_q4_1996';
```

Pour supprimer un histogramme d'une colonne nous avons utilisé la procédure :

COL_STAT_TYPE of DBMS_STATS.DELETE_COLUMN_STATS

Par exemple, pour supprimer l'histogramme crée sur l'attribut Time-level de la table actvars, nous avons utilisé la syntaxe suivante :

```
  Execdbms_stats.delete_column_stats(ownname=>user,
   tabname=>'Actavrs',
   colname=>'time-level',
  col_stat_type=>'HISTOGRAM')
```

Il est important de souligner que nous pouvons avoir des informations statistiques sur n'importe quelle structure de données : sur les tables (par DBA_TAB_STATISTICS), sur les partitions (par DBA_TAB_PARTITIONS), sur les sous-partitions (par DBA_TAB_SUBPARTITIONS), même sur une colonne partitionnée (par DBA_PART_COL_STATISTICS).

7.3.2 Réfragmentation des données

La figure 7.14, ci-dessous, montre la fréquence des accès des requêtes aux attributs. L'attribut le plus accédé est l'attribut Time-level (histogramme sur FR4) et les données du bucket 4, en l'occurrence les données du fragment 4, sont les plus utilisées. De ce fait, utiliser les données de ce fragment pour exécuter une fragmentation va réduire l'accès des requêtes à d'autres données impertinentes et par conséquent une amélioration des performances.

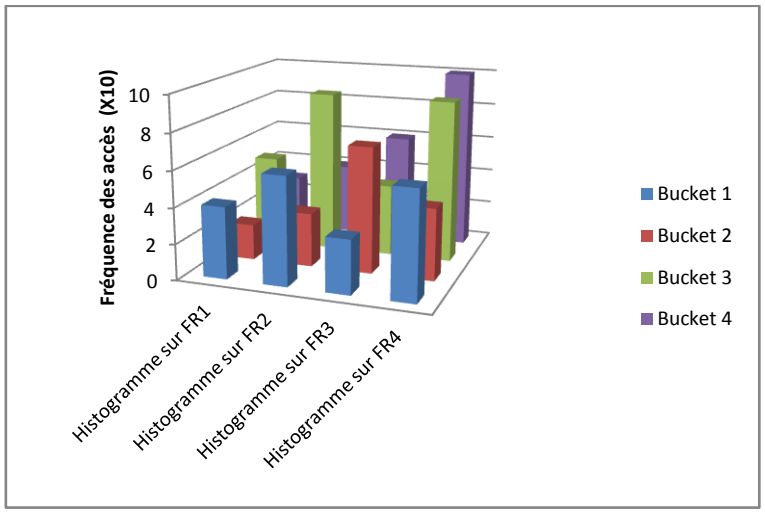


Figure 7.14- Fréquences des accès des requêtes.

Dans la figure 7.14 :

- Histogramme FR1 est crée sur la clé étrangère Channel_level;
- Histogramme FR2 est crée sur la clé étrangère Customer_level;
- Histogramme FR3 est crée sur la clé étrangère Product_level;
- Histogramme FR4 est crée sur la clé étrangère Time_level.

Pour évaluer notre approche de réfragmentation, nous avons fixé le temps seuil d'exécution des requêtes OLAP à 3,5 (s) et le nombre de fragments à 70. Nous avons, dans une première étape, procédé à l'exécution de plusieurs séries de requêtes, et ce à l'effet de déterminer les requêtes dont le temps d'exécution dépasse le temps seuil. Une fois ces requêtes identifiées, nous avons lancé le processus de réfragmentation et réexécuter lesdites requêtes sur le nouveau schéma de fragmentation.

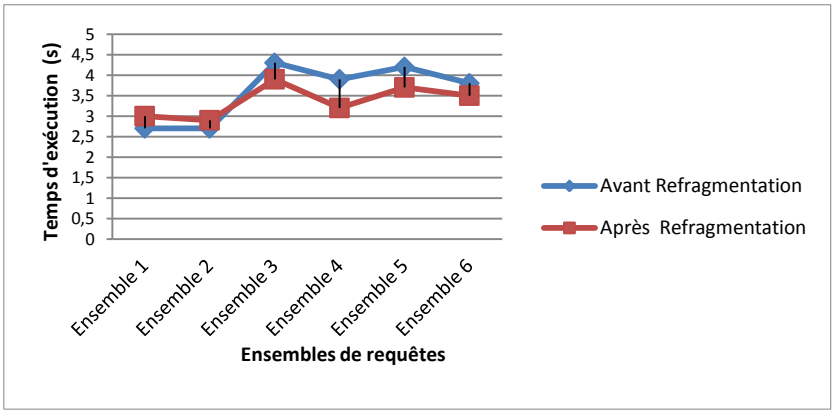


Figure 7.15- Evaluation de l'approche de réfragmentation des données.

La figure 7.15, ci-dessus, montre que le seuil est atteint avec l'exécution de la $3^{ème}$ série de requêtes. Nous constatons, qu'après l'exécution de la réfragmentation, le temps d'exécution de cette série de requête diminue. Ce qui permet de valider notre approche selon les séries de requêtes considérés dans nos tests expérimentaux.

Néanmoins, la réorganisation et le redimensionnement des buckets des histogrammes, après une réfragmentation, nécessitent encore des ajustements notamment lors de la création de nouveau buckets ou lors de la modification de leurs tailles. En effet, si à tout moment un accès à une donnée se produit à l'extérieur de l'intervalle couvert par le bucket courant, un nouveau bucket est alors construit. Si le seuil MAX_B des buckets a été atteint, la taille T_b des buckets est alors augmentée et l'histogramme est réorganisé. Ceci est réalisé en multipliant la taille T_b par un facteur de grandeur Z_T. Cependant, augmenter la largeur des buckets réduit l'exactitude de l'histogramme et peut engendrer une inadaptation des histogrammes avec les attributs sur lesquels ils ont été crées. La même chose se produira en cas de diminution de la taille des buckets pour limiter leur nombre au nombre MAX_B autorisé. De ce fait, il est nécessaire d'approfondir d'avantage l'étude, notamment par le développement d'autres algorithmes consacrés spécifiquement pour cet aspect de manipulation des histogrammes.

7.3.3 Conclusion

Dans cette contribution, nous avons proposé une approche de fragmentation dynamique des données. Pour pouvoir intégrer l'aspect « dynamicité », nous avons utilisé les histogrammes pour la sauvegarde de l'historique des accès afin d'estimer la sélectivité des données des différents fragments. Pour réfragmenter les données, nous avons développé un algorithme itératif qui permet de déterminer un nouveau schéma de fragmentation optimal selon des informations statistiques récentes de l'exploitation des données.

7.4 Expérimentation de l'approche d'évaluation d'un schéma de fragmentation

Pour valider notre approche d'évaluation de la pertinence d'un schéma de fragmentation, nous avons considéré une matrice d'usage des attributs, utilisé dans les travaux de navathe et al [NAB89], pour concevoir un schéma de fragmentation vertical optimal (figure 7.14). Pour rappel, les lignes de cette matrice sont les requêtes fréquentes et les colonnes sont les attributs de la table des faits. Les termes de cette matrice sont les fréquences d'accès des requêtes. Cette matrice est constituée de 10 attributs et de 08 requêtes fréquentes, et dont les termes représentent la fréquence d'accès d'une requête q_i à un attribut A_i.

Attributs / Requêtes	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
q1	0	25	0	0	25	0	25	0	0	0
q2	50	5	50	0	0	0	0	50	50	0
q3	0	0	0	25	0	25	0	0	0	25
q4	35	0	0	0	0	0	35	35	0	0
q5	25	25	25	0	25	0	25	25	25	0
q6	0	25	0	0	25	0	0	0	0	0
q7	0	0	25	0	0	0	0	0	25	0
q8	0	0	15	15	0	15	0	0	15	15

Figure 7.16- Matrice d'usage des attributs.

Pour mener nos tests, nous avons considéré quatre schémas de fragmentation. Pour chaque schéma, nous avons calculé son erreur quadratique et ce, après avoir calculé la valeur moyenne des accès et l'erreur quadratique pour chaque fragment de chaque schéma.

Soit les schémas de fragmentation suivants :

- *Schéma de fragmentation 1* : Composé d'un seul fragment composé des attributs:(A1,A2,A3, A4,A5,A6,A7,A8,A9,A10)
- Schéma de fragmentation 2 : Composé de deux fragments :
  F1: (A1,A4,A5,A6,A7,A10) ; F2: (A2,A3,A8,A9).
- Schéma de fragmentation 3: composé de trois fragments : F1(A2,A5,A7); F2(A1, A3, A8, A9) ; F3:(A4,A6,A10).
- Schéma de fragmentation 4: composé de quatre fragments : F1:(A1,A5) ; F2:(A2,A3,A8,A9); F3:(A4,A6,A10); F4:(A7).

Nous avons d'abord commencé par le calcul de la valeur moyenne des accès pour chaque schéma de fragmentation. Pour rappel la valeur moyenne est calculé selon l'équation suivante : $V_i = \frac{1}{n_i} \sum_{j=1}^{n_i} A_{ij}$

Prenons le schéma de fragmentation 3 composé des fragments :
Fragment 1 : (A2,A5,A7) ; Fragment 2 : (A1, A3, A8, A9) ; Fragment 3 : (A4, A6, A10). Le calcul de la valeur moyenne des accès de chaque fragment est illustré dans le tableau 7.3, ci-dessous.

	Fragment I	*Fragment II*	*Fragment III*
Valeur moyenne des accès par fragment	25	0	0
	0	50	0
	0	0	25
	12	18	0
	25	25	0
	17	0	0
	0	13	0
	0	18	15

Tableau 7.3- Calcul des moyennes des fragments.

A partir des résultats du calcul de la valeur moyenne des accès, nous calculons l'erreur quadratique de chaque fragment. L'équation de l'erreur quadratique des fragments est :

$$e_i^2 = \sum_{j=1}^{n_i} (A_{ij} - Vi)^Q (A_{ij} - Vi)$$

Pour les valeurs moyennes des accès des fragments du schéma de fragmentation 3, ci-dessus, les erreurs quadratique pour chaque fragment est :

Fragment 1 : *1234 ;* Fragment 2 : 2078 ; Fragment 3 : 0

L'erreur quadratique du schéma de fragmentation est la somme des erreurs quadratique de chaque fragment.

Pour le schéma de fragmentation 3, le calcul de l'erreur quadratique donne le résultat suivant :

$$\mathbf{E_M^2} = 1234 + 2078 + 0 = \mathbf{3312}$$

Nous avons effectué ce calcul pour tous les schémas de fragmentation. Les erreurs quadratiques obtenues sont :

- Schéma de fragmentation 1 : $\boldsymbol{E_M^2}$ **= 12577**
- Schéma de fragmentation 2 : $\boldsymbol{E_M^2}$ **= 5949**
- Schéma de fragmentation 3 : $\boldsymbol{E_M^2}$ = **3312**
- Schéma de fragmentation 4 : $\boldsymbol{E_M^2}$ **= 3519**

On conclut donc que le schéma de fragmentation 3 est le schéma de fragmentation le plus optimal, étant donné que son erreur quadratique a la plus petite. Cela signifie que les fragments de ce schéma contiennent le moins d'attributs impertinent pour les requêtes utilisées.

En outre, les résultats obtenus montrent que le schéma de fragmentation, dont la valeur de son erreur quadratique est la plus petite, correspond au schéma de fragmentation

optimal déterminer par l'algorithme d'affinité de navathe et al., [NAB89]. Ce qui démontre de la validité du principe utilisée de notre approche, à savoir mesurer la pertinence d'un schéma de fragmentation selon l'affinité des attributs qui composent les fragments. Reste donc à élargir le champ d'étude par le calcul de l'erreur quadratique pour évaluer un schéma de fragmentation horizontal en mesurant l'affinité des prédicats de sélection.

Par ailleurs, différents tests ont été réalisés sur le calcul de l'erreur quadratique selon différents schéma de fragmentation et nous avons constaté que le nombre de fragments est inversement proportionnel à la valeur de l'erreur quadratique. Plus le nombre de fragments est grand plus le coût d'accès aux attributs impertinents devient minime (figure 7.16). Ce qui montre également l'apport de la fragmentation verticale si les attributs de fragmentation ont été bien définis.

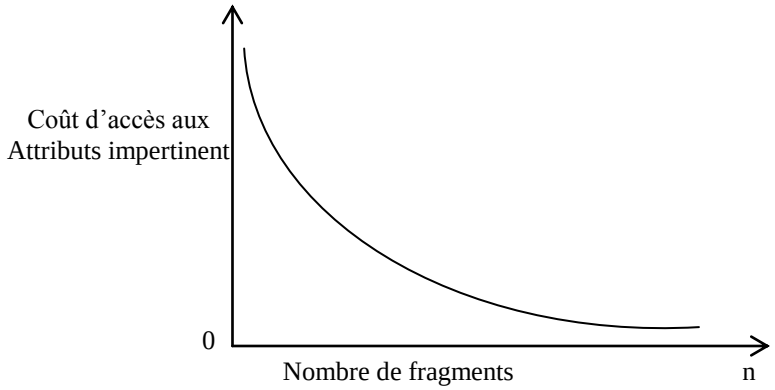


Figure 7.17- Influence du nombre de fragments sur le coût d'accès aux attributs

7.4.1 Conclusion

Les tests effectués, nous ont permis de valider le principe présenté dans notre approche relative à l'évaluation d'un schéma de fragmentation. Comme démontré, l'exploitation des informations se rapportant aux fréquences d'accès des requêtes, les plus fréquentes aux données, peut être considérée, non seulement comme paramètre de base pour la conception d'un schéma de fragmentation, mais également pour définir une fonction coût permettant d'évaluer un schéma de fragmentation en cours d'exploitation et voire même définir un autre schéma de fragmentation plus optimal, en terme de pertinence des attributs des différents fragments.

Chapitre 8
Conclusion générale et perspectives

8.1 Conclusion générale

Dans cette thèse, nous avons présenté et apporté des solutions aux problèmes d'adaptation des techniques de fragmentation aux entrepôts de données.

La première problématique, traitée dans le cadre de cette thèse, porte sur le problème de la NP-complétude de la conception d'un schéma de fragmentation et la nécessité de proposer l'utilisation de métaheuristiques simples avec un minimum de paramétrage. En effet, déterminer un schéma de fragmentation optimal par le regroupement des attributs ou des prédicats selon leurs fréquences d'accès, tout en minimisant une fonction objectif et en contrôlant le nombre de fragments générés, est un problème d'optimisation combinatoire qui nécessite le recours à des algorithmes approchés pour le résoudre. Dans ce contexte, nous avons proposé l'utilisation d'une métaheuristique, en l'occurrence l'Optimisation par Essaim Particulaire (OEP), pour déterminer une stratégie de fragmentation optimale. Nous avons proposé l'utilisation d'une version adaptative des algorithmes OEP, en l'occurrence TRIBE. Cette version est considérée comme étant un algorithme sans paramètres de contrôle. Il est défini comme une « boîte noire », pour laquelle l'utilisateur n'a qu'à spécifier le problème à résoudre. Nous avons, alors adapté l'OEP aux trois approches de fragmentation, verticale, horizontale primaire et horizontale dérivée, pour la sélection d'un schéma de fragmentation optimal. Les algorithmes proposés permettent de manipuler une population à travers l'application itérative d'opération de comportement, de déplacement et de changement de direction pour générer de nouvelles solutions satisfaisant des contraintes préalablement définies.

La seconde problématique, concerne l'évaluation de la pertinence d'un schéma de fragmentation implémenté. Nous avons remarqué que les travaux, qui ont traité cette problématique, l'abordent uniquement dans la phase logique de la conception de la fragmentation en proposant des métriques basées sur des modèles de coûts pour évaluer l'apport d'un algorithme par rapport à un autre. Or, une fois le schéma de fragmentation est implémenté physiquement on ne dispose pas d'approches permettant d'estimer son optimalité et de le considérer comme un schéma qui offre toujours les meilleurs performances. Pour ce faire, nous avons proposé une approche d'évaluation qui mesure la pertinence du schéma de fragmentation sur la base des fréquences des accès aux données des différents fragments. Nous avons, à cet effet, utilisé le calcul de l'erreur qua-

dratique, une fonction d'estimation statistique, pour évaluer les fragments selon la pertinence de leurs attributs.

Enfin, la dernière problématique traitée porte sur l'impact de l'évolution et le passage à l'échelle de l'entrepôt de données sur le schéma de fragmentation. Sélectionner un schéma de fragmentation sur la base de l'exécution des requêtes OLAP, de type interactive, sans tenir compte de l'évolution de la charge de travail, ne peut offrir des performances optimales. Le principe de base de notre troisième contribution, consiste à proposer une stratégie qui permet de capturer les changements intervenus sur la charge de requêtes à l'effet de réfragmenter les données. Il s'agit, concrètement, d'observer et de sauvegarder l'historique relatif aux accès aux données et de procéder, à l'issue d'une dégradation des performances, à une réfragmentation des données en tenant compte des informations statistiques récentes. Nous avons développé, à cet effet, un algorithme itératif d'une complexité polynomiale et nous avons proposé l'utilisation des histogrammes pour la sauvegarde et la manipulation des informatisons statistiques relatives aux accès aux données.

8.2 Perspectives

A l'issue des différentes propositions qui ont été présentées, dans le cadre de cette thèse, plusieurs axes de recherche peuvent être explorés.

S'agissant de la conception d'un schéma de fragmentation optimale par l'utilisation de l'OEP, nous envisageons d'abord de poursuivre les expérimentations en effectuant des tests comparatifs avec d'autres travaux qui ont utilisé des heuristiques pour la sélection d'un schéma de fragmentation optimal, notamment ceux utilisant les algorithmes génétiques. Aussi, dans notre proposition, nous avons utilisé une seule contrainte à respecter, à savoir limiter le nombre de fragments à générer. Il serait intéressant d'étudier la possibilité de la prise en charge d'autres contraintes aussi importantes, tels que : le coût de maintenance des vues et des index.

En ce qui concerne notre deuxième contribution, relative à l'évaluation des schémas de fragmentations, nous envisagerons de valider notre approche par son adaptation à la fragmentation horizontale, étant donné qu'elle est très utilisée dans les entrepôts de données. Il s'agira de reformuler le calcul de l'erreur quadratique par la prise en compte de la sélectivité et de l'affinité des prédicats. Aussi, il est intéressant d'étendre l'approche d'évaluation de la fragmentation par l'intégration dans la fonction coût d'autres métriques, tel que le temps d'exécution des requêtes décisionnelles.

Pour nos travaux futurs, relatifs à la fragmentation dynamique des données, il serait utile d'appliquer notre approche à la fragmentation horizontale dérivée. Cette approche a fait l'objet de plusieurs travaux qui ont démontré son apport en terme d'optimisation

des performances dans les entrepôts de données. Aussi, développer des outils pour automatiser notre approche permettra réellement de surmonter le problème de l'évolution de la charge de travail. Cependant, il convient de souligner que l'automatisation de la fragmentation nécessite de définir un mécanisme de réécriture de requêtes décisionnelles d'un schéma de fragmentation à un autre. Aussi, notre périmètre d'étude de la fragmentation dynamique des données s'est limité uniquement à la fragmentation des données. Il serait utile d'intégrer la notion de placement ou l'allocation des fragments, dans un contexte distribué, pour en mesurer réellement les avantages.

Bibliographie

[ACS99] Acharya.S, P.Gibbons, V.Pooaala and S. Ramaswamy. The Aqua Approximate Query Answering System. *In Proceedings of ACM SIGMOD Philladephia PA*, pages 574-578, 1999.

[AGR97] Agrawal.R., A.Gupta, and S.Saragwi. Modeling multidimensional databases. *IEEE Conference on Data Engineering, Birmingham*, pages 232-243. 1997.

[AGS04] Agrafiotis.S, V.R.Narasayya, and B.Yang. Intrating vertical and horizontal partitioning into automated physical database design .*In proceeding of SIGMOD*, 2004.

[AGS06] Agrafiotis,S, E. Chun and V.R.Narasayya. Automatic physical design tuning: workload as a sequence. *In proceeding of SIGMOD*, 2006.

[ALE05] Alba.E. Parallel Metaheuristics : A New Class of Algorithms. *John Wiley & sons*, 2005.

[ALA04] Alexandre. A. B.Lima. M.Mattoso and P.Valduriez. Adaptive Virtual partitioning for OLAP Query Processing in a Database Cluster. *In 19 th proceeding on SBBD, Brazil*, 2004.

[ANS97] Anahory .S and D. Murray D. Data Warehousing in the Real World : A Practical Guide for Building Decision Support Systems. *Harlow, England : Addison Wesley Longman*, 1997.

[ANE98] Angel.E. La rugosité des paysages : Une théorie par la difficulté des problèmes d'optimisation combinatoire relativement aux méta-heuristiques. *Thèse de l'Université Paris XI Orsay*. (1998*).*

[AOK05] Aouiche.K. Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données. *Ph.d. thesis, Université Lumière Lyon 2, December*, 2005.

[AOK06] Aouiche K., P.E. Jouve et J.Darmont. Clustering-Based Materialized View Selection. *InTenth East-European Conference on Advances in Databases and Information Systems (ADBIS 06*), Greece, 2006.

[BAS97] Baralis.E, S. Paraboschi, and E. Teniente. Materialized view selection in a multidimensional database. *In Proceedings of the International Conference on Very Large Databases, pages 156–165,* August, 1997.

[BAX03] Baril.X and Z. Bellahsène. Selection of materialized views: a cost-based approach. *In 15th International Conference on Advanced Information Systems Engineering (CAiSE 03), Klagenfurt, Austria, pages 665–680*, 2003.

[BAM12] Barr, M. Self – monitoring for the horizontal fragmentation evolution based on ants in the Relational datawarehouses. *In 8th International Conference Computing Technology and Information Management (ICCM), page(s): 538 – 541, Volume: 1*, 2012.

[BAR96] Battiti.R. Reactive search: toward self tuning heuristics. *Modern Heuristic Search Methods, pp. 61-83, John Wiley & Sons,* 1996

[BEB11] Benmessahel.B, M.Touahria. An Improved Combinatorial Particle Swarm Optimization Algorithm to Database Vertical Partition. *Journal of Emerging Trends in Computing and Information Science. Volume 2 No.3*. 2011.

[BEL00] Bellatreche. L. Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données. *Thèse de doctorat, Université de Clermont-Ferrand II*, 2000.

[BEL05] Bellatreche. L and K.Boukhalfa. An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse Environment. *In Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'05)*, 2005.

[BEJ02] Bernardino.J, Furtado.P, and P.H.Madeira. Approximate Query Answering Using Data Warehouse Striping. *Journal of Intelligent Information Systems-Integrating Artificial Intelligence and Database Technologies, Volume 19, Issue 2, Elsevier Science Publication*, 2002.

[BLC05] Blum.C. Ant colony optimization : Introduction and recent trends. *Physics of Life Review, Vol. 2, pp. 353-373*, 2005.

[BOK08] Boukhalfa.K, L. Bellatreche and R. P. Fragmentation primaire et dérivée : Étude de complexité, algorithmes de sélection et validation sous oracle10g. *Revue des Nouvelles Technologies de l'Information RNTI*, 2008.

[BOK09] Boukhalfa, K. De la conception physique aux outils d'administration et de tuning des entrepôts de données. *Thèse de doctorat, Université USTHB, Alger*, 2009.

[BRJ01] Branke.J. Evolutionary Optimization in Dynamic Environments. *Genetic Algorirhms and Evolutionary Computation, Vol. 3, 2001, Kluwer Academic Publishers*, 2001.

[BRN02] Bruno.N and S. Chaudhuri. Exploiting statistics on query expressions for optimization. *ACM SIGMOD, june 4-6, Madison, Wisconsin, USA*, 2002.

[CAE06] Campana.E.F, G. Fasano, D. Peri, and A. Pinto. Particle Swarm Optimization : Efficient Globally Convergent Modifications. *In Proceedings of the 3rd European Conference on Computational Mechanics, Solids and Coupled Problems in Engineering, 2006, Springer*, 2006.

[CAE01] Carbonaro.A and MAniezzo.V. Ant colony optimization : an overview. *In Essay and Survey in Metaheuristics Conference, Angra dos Reis, Brazil*, 2001.

[CES82] Ceri.S, M.Negri, and G.Pelagatti. Horizontal data partitioning in data base design. *In Proceeding of the ACM SIGMOD, International Conference on Management of Data. SIGPLAN Notices, page 128-136*, 1982.

[CHS97] Chaudhuri.S., U.Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record, 26(1): pages 65-74*, 1997.

[CHS98] Chaudhuri.S., and V. Narasayya. Autoadmin 'what-if' index analysis utility. *In Proceedings of the ACM SIGMOD International conference on Management of Data, pages 367-378, June*, 1998.

[CHS99] Chaudhuri.S., and V. Narasayya. Index merging. *In Proceedings of the International conference on Data Engineering (ICDE), pages 296-303, March*, 1999.

[CHS04] Chaudhuri, S. Index selection for databases: A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering 16(11), 1313–1323*, 2004.

[CHL04] Chen.L, X.H. Xu, and Y.X. Chen. An Adaptive Ant Colony Clustering Algorithm. *In Proceedings of the 3rd Conference on Machine Learning and Cybernetics, 2004, pp. 1387-1392, IEEE Press*, 2004.

[CHC02] Cheng. C., W.Lee, and K.Wong. A Genetic Algorithm-Based Clustering Approach for Database Partitioning. *IEEE Transactions on Systems, Man, and Cybernetics, 32(3), 215-230*, 2002.

[CLM02] Clerc.M., J. Kennedy. The particle swarm: explosion, stability, and convergence in multi-dimensional complex space. *IEEE Transactions on Evolutionary Computation, Vol. 6, pp. 58-73,* 2002.

[CLM03] Clerc.M. TRIBES – Un exemple d'optimisation par essaim particulaire sans paramère de contrôle. *Conférence OEP'03, 2 Octobre, Paris, France,* 2003.

[CLM06] Clerc.M. Particle Swarm Optimization. *International Scientific and Technical Encyclopedia*, John Wiley & sons, 2006.

[COE93] CODD. E.F. Providing OLAP (on-line analytical processing) to user-analysts : an IT mandate. *Technical report, E.F Codd and Associates*, 1993.

[COA92] Colorni.A, M.Dorigo, and V.Maniezzo. Distributed Optimization by Ant Colonies. *In Varela, F. and Bourgine, P., editors, Proceedings of ECAL'91 - First European Conference on Artificial Life, pages 134_142, Paris, France. Elsevier Publishing*. 1992.

[COY07] Cooren.Y, M. Clerc, and P. Siarry. Initialization and Displacements of the Particles in TRIBES, a Parameter-free Particle Swarm Optimization Algorithm. *Adaptive and Multilevel Metaheuristics, pp. 199-219, Springer*, 2007.

[COY08a] Cooren.Y, M. Clerc, and P. Siarry. Performance Evaluation of TRIBES, an Adaptive Particle Swarm Optimization Algorithm. *Swarm Intelligence, Springer. En revision,* 2008.

[COY08b] Cooren.Y, M. Clerc, and P. Siarry. MO-TRIBES, an adaptive multiobjective particle swarm optimization algorithm. *Computational Optimization and Applications, Springer. Soumis*, 2008.

[DEH08] Derrar, H., M.Ahmed-Nacer, and O.Boussaid. Une approche de répartition des données d'un entrepôt basée sur l'Optimisation par Essaim Particulaire. *Revue des nouvelles technologies de l'information*, RNTI, pp.141-150, 2008.

[DEH09] Derrar.H, O. Boussaïd, M. Ahmed-Nacer. Les histogrammes pour une fragmentation dynamique dans les entrepôts de données. *4ème Atelier sur les Systèmes Décisionnels (ASD'09), Jijel, Algérie*, 2009.

[DEH12] Derrar, H., M.Ahmed-Nacer, and O.Boussaid. Particle swarm optimisation for data warehouse logical design. *Int. J. Bio-Inspired Computation, Vol. 4, No. 4, pp.249–257*, 2012.

[DEH13] Derrar.H, M.Ahmed-Nacer, and O. Boussaid. Exploiting data access for dynamic fragmentation in data warehouss. *Int. J. Intelligent Information and Database Systems, Vol. 7, No. 1, pp.34–52*, 2013.

[DEH14a] Derrar.H, M.Ahmed-Nacer, and O. Boussaid. An objective function for evaluation of fragmentation schema in data warehouse. *To appear in Encyclopedia of Information Science and Technology, Third Edition, IGI Global,* 2014.

[DEH14b] Derrar.H, O. Boussaïd, M. Ahmed-Nacer. Une approche basée sur l'OEP adaptatif pour la sélection d'un schéma de fragmentation horizontal. *In proceeding of COSI'2014, colloque sur l'optimisation et les systèmes d'snformation*. Bejaia, Algérie. 2014

[DIA11] Dimovski.A, G.Velinov, D.Sahpaski. Horizontal Partitioning by Predicate Abstraction and Its Application to Data Warehouse Design. *Advances in Databases and Information Systems Lecture Notes in Computer Science Volume 6295, 2010, pp 164-175*, 2011.

[DIG98] DiCaro.G, and M. Dorigo. Ant colonies for adaptive routing in packetswitched communications networks. *In Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, LNCS 1498,*

pp. 673-682, Springer, 1998)

[DOA01] Doucet.A., S.Gancarski. Chapitre 12, Entrepôts de données et bases de données multidimensionnelles. *In JOMIER G. et DOUCET A., Editions Hermès*, 2001.

[DRJ03] Dréo.J, and P. Siarry. Un algorithme de colonie de fourmis en variables continues hybridé avec un algorithme de recherche locale. *In 5ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2003), Avignon, France*, 2003.

[EBR96] Eberhart.R.C., P. Simpson, R. Dobbins. Computational PC Tools. *Chapter 6, pp. 212-226, AP Professional*, 1996.

[EAT08] Eavis.T, G.Dimitrov, I.Dimitrov, D.Cueva, A.Lopez, and A.Taleb. *Parallel OLAP with the Sidera server, Science direct*, 2008.

[EIA03] Eiben.A.E, J.E. Smith. Introduction to Evolutionary Computing. *Natural Computing Series, 2003, Springer*, 2003.

[ESK74] Eswaran, K.P. Placement of Records in a File and File Allocation in a Computer Network. *Proceedings of IFIP Congress on Information Processing, Stockholm*, Sweden, pp: 304-307, 1974.

[FAH01] Fan.H.Y., Y. Shi. Study on Vmax of particle swarm optimization. *Proceedings of the 2001 Workshop on Particle Swarm Optimization, Indiana University-Purdue University Indianapolis Press*, 2001.

[FOM07] Förster.M, B.Bickel, B.Hardung, G. Kókai. Self-adaptive ant colony optimization applied to function allocation in vehicle networks. *In Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation, pp. 1991-1998, ACM Press*, (2007).

[FUP04] Furtado. P. Experimental Evidence on Partitioning in Parallel Data Warehouses. *DOLAP'04*, Washington, DC, USA, 2004.

[GAC06] Garcia.C. Real Time Self-Maintenable Data Warehouse. *In Proceedings of the 44th annual Southeast regional conference Melbourne, Florida, Session: Database systems II, Pages: 518 - 524*. 2006.

[GAG05] Gardarin.G. (2005). *Base de données*. Edition Eyrolles.

[GHF05] Ghozzi.F., F.Ravat, O.Teste , H.Zurflu. Méthode de conception d'une base multidimensionnelle contrainte. *1ère journée francophone sur les Entrepôts et Analyse en ligne, lyon, France*, 2005.

[GLF97] Glover.F., M. Laguna. Tabu Search. *Kluwer Academic Publishers*, 1997.

[GOD87] Goldberg D.E. Richardson J. Genetic algorithms with sharing for multimodal function optimization, GA and their applications p.41-49 Hillsdale New Jersey, Lawrence Erlbaum ass., 1987

[GOM99] Golfarelli,M., D.Maio, S.Rizzi. Vertical Fragmentation of Views in Relational Data Warehouses. *Université de Bologne, Italy*, 1999.

[GOM02] Golfarelli M., S.Rizzi, and E.Saltarelli. Index selection for data warehousing. I*n 4th International Workshop on Design and Management of Data Warehouses (DMDW 2002), Canada*. 2002.

[GUH99] Gupta.H. Selection and maintenance of views in a data warehouse. *Ph.d. thesis, Stanford University, September*, 1999.

[GUH05] Gupta. H, and I.S.Mumick. Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions On Knowledge And Data Engineering, Vol. 17, No. 1*, 2005.

[HAJ00] Han.J., V.Kamber. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann edition.

[HOLJ73] Holland.J.H. Genetic Algorithms and the optimal allocation of trials. *SIAM Journal of Computing, Vol. 2, pp. 88-105*, 1973.

[INL96] Ingber. L. Adaptive Simulated Annealing (ASA): lessons learned. *Control and Cybernetics, Vol. 25, N° 1, pp. 33-54*, 1996.

[INB05] Inmon.B. Building the data warehouse. *John Wiley and Sons. NY NY*, 2005.

[IOY03] Ioannadis.Y. The History of Histograms (abridged). *In Proceeding of 29th VLDB Conference, Berlin, Germany*, 2003.

[JAH99] Jagadish. H, L.V.S.Lakshmanan, and D.Srivastava. Snakes and sandwiches: Optimal clustering strategies for a data warehouse', *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 37–48, 1999.

[JAA98] Jain,A., R. Dubes. Algorithms for clustering Data. *Prentice Hall Advanced Reference Series, Englewood Clis, NJ*, 1998.

[KAA02] Karahoca, A., N.Osman, and D.Erkan. Random Neural Network Approach in Distributed Database Management Systems. *Technical report*, 2002.

[KAK96] Karlapalem.K., Q. Li, and S. Vieweg. Method induced partitioning schemes in object oriented databases. *In 16th International Conference on Distributed Computing System (ICDCS'96), Hong Kong, pages 377–384, May*, 1996.

[KEJ95] Kennedy.J, and R.Eberhart. Particle Swarm Optimization. *In proceedingsof IEEE International Conference on Neural Network, volume IV, pages 1942–1948*, 1995.

[KEJ99] Kennedy, J. Small Worlds and Mega-Minds : Effects of Neighborhood Topology on Particle Swarm Performance. *In IEEE Congress on Evolutionary Computation, volume III, pages 1932–1938*, 1999.

[KIR96] Kimball.R. The Data Warehouse Toolkit : Practical Techniques for Building Dimensional Data Warehouses. *Edition John Wiley & Sons*, 1996.

[KIR98] Kimball R. et Merz R. The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses. *Wiley Computer Publishing*, 1998.

[KIMR13] Kimball.R., M.Ross. The Data Warehouse toolkit. Third edition. *Lifecycle Toolkit. John Wiley, NY*, 2013.

[KIN07] Kim. N. and S. Moon. Concurrent view maintenance scheme for Soft real-time data warehouse systems. *Journal of information science and engineering 23, 723-739*, 2007.

[KIS83] Kirkpatrick.S., C. D. Gelatt, and M. P. Vecchi.Optimization by simulated annealing. *Science, 220(4598) :671–680*, 1983.

[LUE94] Lumer.E.D and B. Faieta. Diversity and adaptation in populations of clustering ants. *In D. Cliff, P. Husbands, J.A. Meyer, et Stewart W., editors, Proceedings of the Third International Conference on Simulation of Adaptive Behavior, pages 501–508. MIT Press, Cambridge, Massachusetts*, 1994.

[MAH06] Ma.H., K.D.Schewe, and Q.Wang. *A heuristic approach to cost-efficient fragmentation and allocation of complex value databases. 17th Australasian Database Conference on Database Technologies (ADC 06), Hobart, Australia (pp. 183{192)*, 2006.

[MAV01] Maniezzo,V., A.Carbonaro, M.Golfarelli, S. Rizzi. ANTS for Data Warehouse Logi-cal Design. *Proceding 4th Metaheuristics International Conference, Porto*, 2001.

[MUY02] Murata.Y. Agent Oriented Self Adaptive Genetic Algorithm. *In Proceedings of the IASTED Communications and Computer Networks, pp. 348-353, Acta*

Press, 2002.

[NAS84] Navathe,S., S. Ceri, G.Wierhold, et J.Dou. Vertical Partitioning Algorithms for Database Design. *ACM Transactions on Database Systems, Vol. 9, No. 4, Decembre 1984, pages 680-710*, 1984.

[NAB89] Navathe,B., M. Ra. Vertical Partitioning for Database Design: A Graphical Algorithm. *ACM SIGMOD International Conference on Management of Data pp. 44-450*, (1989).

[NIB05] Niu.B., Y. Zhu, X. He, W. Henry. *MCPSO :* A multi-swarm cooperative particle swarm optimizer. *Applied Mathematics and Computation, Vol. 185, N° 2, pp. 1050-1062*, 2005.

[NOA99] Noaman. A. Y., K. Barker. A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse. *In the 8th International Conference on Information and Knowledge Management (CIKM'99), 154–161*, 1999.

[OUC02] Ourique.C, E.J. Biscaia, J. Pinto. The use of particle swarm optimization for dynamical analysis in chemical processes. *Computers & Chemical Engineering, Vol. 26, N°12, pp. 1783-1793*, 2002.

[OZM91] Özsu.M.T and P. Valduriez. Distributed database systems :Where are we now? *IEEE COMPUTER, 24(8) :68–78, August*, 1991.

[OZM99] Özsu.M. and P. Valduriez. Principles of Distributed Database Systems : Second Edition. *Prentice Hall*, 1999.

[PAP04] Papadomanolakis and A. Ailamaki. Autopart : Automating schema design for large scientific databases using data partitioning. *In Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), pages 383–392, June*, 2004.

[PAK04] Parsopoulos.K.E., D. Tasoulis, M.N. Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Vol. 2, pp. 823-828, Acta Press*, 2004.

[PEN00] Pensde.N., R.Creeth. The OLAP report. *Business Intelligence Inc., London*, 2000.

[POU00] Pourabbase, Rafanelli.M. Hierarchies and relative operators in the OLAP environment. *ACM SIGMOD record, 29(1): pages 32-37*, 2000.

[POV96] Poosala.V, Y.Ionnidis, P.Hass and E.Shekita. Improved Histograms For selectivity Estimation of Range Predicates. *In Proceedings of ACM SIGMOD Montreal Canada, page 294-305, June*, 1996.

[POV99] Poosala.V V.Ganti. (1999). Fast approximate answers to aggregate queries on a data cube.*In Proceedings of the 11th International Conference on Statistical and Scientific Database Management, Cleveland, OH, USA pp.24–33*, 1999.

[RAM03] Rafanelli.M. Operators for Multidimensional Aggregate Data. *Chapter V,Multidimensional Databases : Problems and Solutions*, 2003.

[RAJ02] Rao J., C.Zhang , G.Lohman , N.Megiddo. Automating Physical Database Design in a Parallel Database System. *SIGMOD*, 2002.

[SAH99] Sawai.H, S. Adachi. *Genetic Algorithm Inspired by Gene Duplication*. *Proceedings of the Congress on Evolutionary Computation, pp. 480-487, IEEE Computer Society*, 1999.

[SAG86] Sacco. G. Fragmentation: A Technique for Efficient Query Processing. *ACM Transactionon Database Systems, 11: 113-133*, 1986.

[SAA04] Sanjay.A, 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.

[SCV96] Schnecke.V., O. Vornberger. An Adaptive Parallel Genetic Algorithm for VLSI- Layout Optimization. *In Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, pp. 859-868, LNCS, Springer*,1996.

[SHY98] Shi.Y., R. Eberhart. Parameter Selection in Particle Swarm Optimization. *Proceedings of the 7th Annual Conference on Evolutionary Programming, pp. 591-600, LNCS 1447, Springer*, 1998.

[SIP06] Siarry.P, J. Dréo, A. Pétrowski, E. Taillard. *Métaheuristiques pour l'optimisation défficile*. *Edition Eyrolles*, 2006.

[SIH97] Singh H.S. Data Warehousing : Concepts,Technologies, Implementation and Management. *Upper Saddle River, NJ : Prentice-Hall*, 1997.

[SOS00] Song.S.K and Gorla, N. *A genetic Algorithm for Vertical Fragmentation and Access Path Selection*. *The Computer Journal, vol. 45, n° 1,pp 81-93,* 2000.

[SOA05] Soussi.A., J.Feki , F.Gargouri. Approche semi-automatisée de conception de schémas multidimensionnels valides. *1ère journée francophone sur les Entrepôts et Analyse en ligne, lyon, France*, 2005.

[STE01] Stöhr. E.R. WARLOCK: A Data Allocation Tool for Parallel Warehouses. *Proceedings of the 27th VLDB Conference, Roma, Italy*, 2001.

[SUP99] Suganthan. P. Particle Swarm Optimizer with Neighborhood Operator. *In IEEE Congress on Evolutionary Computation, volume III, pages 1958–1961*, 1999.

[SYS97] Red Breck Systems. Star schema processing for complex queries. *White paper, july 1997*.

[TAE02] Talbi.E.G. A taxonomy of Hybrid Metaheuristics. *Journal of Heuristics, Vol. 8, N°5, pp. 541-564*, 2002.

[TAA08] Talebi.A.Z, , R. Chirkova, Y. Fathi, and M. Stallmann. Exact and inexact methods for selecting views and indexes for olap performance improvement. *11th International Conference on Extending Database Technology (EDBT'08), Mars*, 2008.

[THD00] Theodoratos.D., T.SELLIS. Answering Multidimensional Queries on Cubes Using Other Cubes. *12th International Conference on Scientific and Statistical Database Management*, 2000.

[TRI03] Trelea.I.C. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters, Vol. 85, pp. 317-325*, 2003.

[VAF02] Van den Bergh. F. An Analysis of Particle Swarm Optimizers. *PhD thesis, Department of Computer Science, University of Pretoria*, 2002.

[VAP98] Vassiliadis.P. Modeling Databases, Cubes and Cube Operations. *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM), Capri, Italy, IEEE Computer Society, July* 1998.

[XIL04] Xinjian.L and F. Lowenthal. Arranging fact table records in a data warehouse to improve query performance. *Computers & Operations Research 31 2165–2182*, 2004.

[YAK04] Yasuda.K, N. Iwasaki. Adaptive particle swarm optimization using velocity information of swarm. *Proceedings of the IEEE Conference on System, Man and Cybernetics, 2004, pp. 3475-3481, IEEE Press*, 2004.

[ZHC01] Zhang.C and.Y, J. Yang. An evolutionary approach to materialized view selection in a data warehouse environment. *IEEE Transactions on Systems, Man, and Cybernetics, 31(3) :282–294*, 2001.

[ZHY94] Zhang.Y and M.E. Orlowska. On Fragmentation Approaches for Distributed Database Design. *Information Sci., 1: 117-132*, 1994.

[ZIE10] Ziyati.E. Optimisation de requêtes OLAP en entrepôt de données, approche basée sur la fragmentation génétique. *Thèse de doctorat. Université Mohamed V-Agdal. Maroc*, 2010.