الجمهوريّة الجزائريّة الدّيمقراطيّة الشّعبيّة
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزارة التّعليم العالي و البحث العلمي
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جامعة فرحات عبّاس – سطيف –
UNIVERSITÉ FERHAT ABBES -SÉTIF-

Vice rectorat chargé de la post graduation نيابة الجامعة المكلّفة بدراسات ما بعد التّدرّج
مدرسة الدكتوراه في الإعلام الآلي تخصّص هندسة الأنظمة المعلوماتيّة
École doctorale d'informatique
Option
Ingénierie des Systèmes Informatiques

Mémoire

Présenté pour l'obtention du

Diplôme de Magister

par **mourad KEZAI**

Thème:

Optimisation automatique des performances dans les entrepôts de données : étude comparative

Soutenu le 19/06/2011 : devant la commission d'examen :

Dr : KHABABA Abdallah	MC à l'université de Sétif	Président
Dr : MOUSSAOUI Abdelouahab	MC à l'université de Sétif	Rapporteur
Dr : TOUAHRIA Mohamed	MC à l'université de Sétif	Examinateur

Remerciement

*En tout premier lieu, nous remercions Allah le tout Puissant, à la sagesse et au savoir infinis, " Gloire à Toi ! Nous n'avons de savoir que ce que Tu nous as appris. Certes c'est Toi l'Omniscient, le Sage. " (Sourate al-Baqarah, verset 32).**

Je remercie sincèrement M. MOUSSAOUI Abdelouahab, pour son encadrement, son très grande disponibilité. Ainsi que ses qualité humaine et scientifiques qui m'ont permis de mener à bien et avec grand plaisir et liberté ce travail et surtout pour son soutient sans faille. Je le remercie d'avoir eu confiance en ce mémoire.

Je tiens à remercier très sincèrement l'ensemble des membres du jury qui me font le grand honneur d'avoir accepté de juger mon travail.

Je voudrais remercier tous mes amis et collègues pour leur aide, leur soutien et leur gentillesse.

Et surtout

Merci à mes parents (Ma Mère, Mon Père). Je leur serai éternellement reconnaissant de d'avoir assurer de leurs confiances, de leurs encouragements et de leurs soutiens sans limite. Merci aussi à mon frère et ma sœur. Merci à mes, cousins et cousines.

Enfin, je souhaite remercier tous les personnes que j'ai oubliées

Dédicaces

A la mémoire de mon grand-père mohamed et tout ceux qui nous on quitté

Á mes parents

et à toute ma famille.

Je dédie ce modeste travail.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L’histoire de l’évolution de l’informatique a vécu plusieurs époques ,principalement deux l’informatique classique ou transactionnel et l’informatique décisionnelle, l’aspect transactionnel qui l’exprime surtout au rapport requête / résultat par l’utilisation des transaction entre des support comme les bases de données et le SGBD, mais ce genre de système a preuve sa limitation devant des nouveau situation comme des opération très complexé davantage historisées.

Cette difficulté a pavé à l’apparition des systèmes décisionnelle qui répondit à tous les nouveaux besoins des utilisateurs ou décideurs, ce qu’implique la naissance des nouveaux concepts ainsi que nouvelle structure comme le data mining et l’entrepôt de données.

Ces entrepôts de données a nous facilites plusieurs taches surtout au niveau logistique quand la décision doit être claire et précis, donc l’informatique a vécu son succès dual avec l’économie, En effet, l'entrepôt de données est le meilleur moyen que les professionnels ont trouvé pour modéliser de l'information pour des fins d'analyse.

Ce concept d'entreposage de données a basés sur un processus qui commence par la collecte de l'information de différent source hétérogène de donnée, puis elle passe par quelque étape comme le nettoyage et l'élimination des donnée inutile et puis le de les homogénéiser et de leur donner un sens unique compréhensible par tous les utilisateurs ,et enfin le chargement de ces données et le structuré dans une masse physique c'est l'entrepôt de données final prêt a l'exploité par les prioritaires comme les décideurs , mais en parallèle à ce nouveau concept et comme l'indique son caractéristique apparente c'est la volumétrie, a cause de la croissance de l'information dans tout les domaines sources de construire de ce entrepôt ainsi que ça augmentation avec le temps donc le traitement effectué au niveau de l'entrepôt risque de ne pas atteindre ces buts dans des condition parfaite , ce que nous guidons de commencé a chercher les technique qui nous permet de l'optimisé les opération sur ce entrepôt de donnée sois d'une manière manuel ou automatique .

En fait, il est devenu très important de réduire les tâches d'administration des systèmes de gestion de base de données. Les systèmes auto-administratifs ont pour objectif de s'administrer et de s'adapter eux-mêmes, automatiquement, sans perte ou même avec un gain de performance.

L'objectif principal de ce travail consiste à étudier l'auto-administration des structures d'optimisation dans les entrepôts de données relationnelles, en effet d'étudier l'ensemble des méthodes d'optimisation et de gestion d'un entrepôt de données relationnel existe dans la littérature qui modélisé souvent par un schéma en étoile. Ces méthodes peuvent les classer en deux catégories principales : techniques redondantes et techniques non redondantes. Les techniques redondantes optimisent les requêtes, mais exigent un coût de stockage et de maintenance. Parmi ces techniques il existe les vues matérialisées, les index, la fragmentation verticale .et les techniques non redondantes qui ne nécessitent ni coût de stockage ni coût de maintenance. Ce genre de technique englobe la fragmentation horizontale, le traitement parallèle.

Le lancement d'un nouveau étude sur un sujet prédéterminé passe toujours par une étude détaillé sur l'état de l'art ,et l'avancement a jour dans ce domaine donc le choix de la technique d'optimisation la plus appropriées a nos exigence passe inévitablement par l'étude de chacun de ces techniques et de faire une comparaison théorique entre eux par rapport au critères soigneusement sélectionnés.

La fragmentation horizontale a connu une évolution importante durant ces dernières années. Utilisée au début des années 80 pour la conception logique des bases de données réparties, elle est devenue une technique incontournable pour la conception physique des bases et entrepôts de données. La plupart des éditeurs des SGBD l'ont adoptée.

Objectives :

Notre travail est beaucoup plus orienté vers la technique de fragmentation horizontale d'un entrepôt de données qui considère comme une structure d'optimisation non redondante, il illustre par la suite l'utilisation d'une approche de fragmentation basée sur deux techniques de fragmentation : la fragmentation horizontale primaire pour fragmenter les tables de dimension et la fragmentation horizontale dérivée pour fragmenter la table de faits.

Afin de valider nos travaux, une étude comparative a été faite avec les travaux déjà existe, basées sur les trois célèbre approches , le premier à base de l'algorithme Hill Climbing le deuxième à base du recuit simulé et enfin à base d'algorithme génétique, nos tests réalisés sur un banc d'essais (BENCHMARK) APB-1.Après la validation théorique, en utilisant un modèle de coût, nous avons prouvé l'efficacité et les performances de notre approche, par rapport aux différentes approches proposées dans le domaine, sur un SGBD réel ORACLE 10g.

Organisation du mémoire :

Le manuscrit se décompose en cinq (05) chapitres - l'état de l'art (chapitre de 1 à 4) et un chapitre décrivant notre contribution ainsi que quelques résultats expérimentaux.

Chapitre 1 : nous présenté l'historique des systèmes informatique ainsi que quelque notion ce que concerne le système transactionnel et son passage vers le décisionnel il illustre aussi le besoin des utilisateurs d'un nouveau concept représentant au entrepôt de données et ces fonctionnalités.

C**hapitre 2** : nous nous attacherons à présenter les principales notions relatives aux entrepôts de données. Ainsi que leur Caractéristique et son architecture et même ces différents modes de modélisation de cette nouvelle structure en résumé on étale les plus

importants points que l'on aide à bien comprendre notre problématique et les diverses approches existe aux les sections suivantes.

Chapitre 3 : illustre les principales techniques d'optimisation déjà existent dans la littérature qui est plus répondu aux principaux systèmes de gestion de bases de données (SGBD) comme les index, les vues matérialisées, la fragmentation. Il se base essentiellement sur les travaux réalisés dans le but de la sélection des techniques d'optimisation, Pour chaque technique d'optimisation, nous détaillons le processus de sa sélection et quelques algorithmes principaux, nous étalons par la suite les avantages et les limites des techniques de sélection proposées. Cette étude nous offrir la possibilité de proposer une classification des techniques d'optimisation existantes.

Chapitre 4 : ce chapitre est consacré aux techniques de fragmentation horizontale primaire et dérivée, on exposant ces avantages. On expose aussi le problème de la sélection d'un schéma de fragmentation horizontale, il aborde par la suite notre approche de fragmentation horizontale dans les entrepôts de données relationnels.

Nous avons montré les différentes approches utilisées pour sélectionner le meilleur schéma de fragmentation horizontale .où nous avons focalisés sur trois principaux algorithmes de méta-heuristiques (l'algorithme génétique, le recuit simulé et le hill climbing), ce qui nous a permis d'avoir une idée claire sur les travaux existants, et d'extraire leurs avantages et inconvénients.

Chapitre 5 : nous avons fait une comparaison théorique entre les algorithmes proposés pour la fragmentation horizontal et puis on montré la faisabilité de la démarche que nous avons proposé sur un banc d'essais APB-1 avec une charge requêtes, et puis un éclaircissement sur la méthode d'installation d'un entrepôt de données réel sur le SGBD ORACLE. On appliquant par la suite Les 3 algorithmes de sélections d'un schéma de fragmentation horizontal sur notre entrepôt expérimentale et le compare ces résultats.

Enfin le manuscrit se termine par une conclusion générale rappelant tous les apports de l'approche traité avec une proposition de quelques perspectives très intéressante et très objectives.

1

Système d’information décisionnel

Introduction

Face à la mondialisation, les entreprises doivent être en mesure de livrer une grande concurrence sur les marchés mondiaux et presque dans tous les domaines, ce qui est de plus en plus complexe et changeant.

Les entreprises sont confrontées à une concurrence de plus en plus féroce, les clients de plus en plus exigeants que ces entreprises doivent s'appuyer sur des informations pertinentes, pour être efficace et faire face aux nouveaux défis économiques

Mais souvent, les données dans les entreprises sont dispersées à travers de multiples systèmes hétérogènes ne sont pas organisées dans une vision décisionnelle. Il est donc nécessaire d'uniformiser et de les rassembler pour faciliter la prise de décision, cela que ne permet de définir la naissance de nouveau architecture qui serve de fondations aux applications décisionnelles, où le nouveau concept de «d’informatique décisionnelle».

1.1 Historique des systèmes décisionnels

Les origines des systèmes informatiques aident à la décision (Decision Support System) de retour aux premiers Apparence de systèmes informatiques et d'information. Ces systèmes ont connu une importante et complexe évolution liée à la technologie particulière. Cette tendance se poursuit jusqu'à aujourd'hui.

La figure (**FIG. 1.1**) montre les grandes étapes que les systèmes décisionnels sont passés depuis son Apparence.

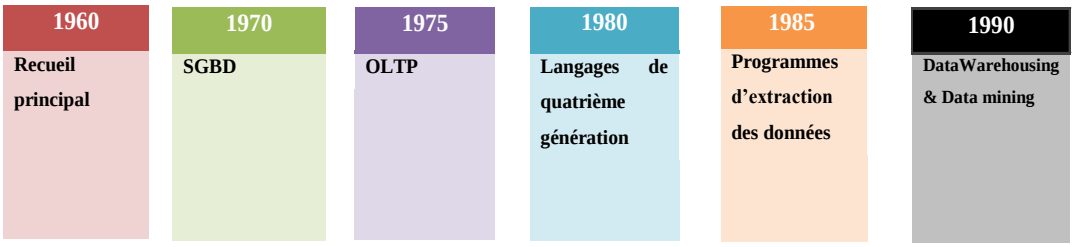

FIG.1.1 L'évolution des systèmes décisionnels.

1.1.1 Le recueil principal

Le côté logiciel de l'informatique n'émergent qu'au début des années 1960, et ces premiers tentatives consistait à la création d'applications individuelles qui ont été effectuées en utilisant le recueil principal (Fichier principal portant tout les définitions et les paramètres essentielles à son fonctionnement).

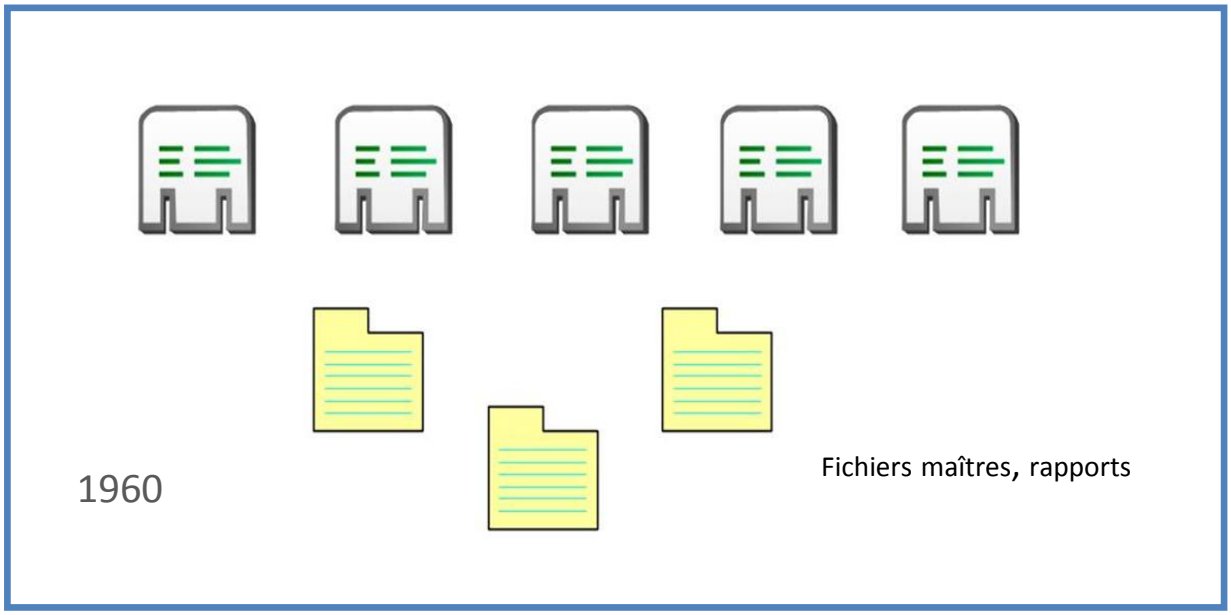

FIG. 1.2 le recueil principal.

A cette époque COBOL est considéré comme le langage de programmation le plus employé à la construction des programmes et des états imprimés.

L'inconvénient des outils de cette période est la nécessité de parcourir les données de façon séquentielle avec le déroulement d'une bande magnétique, ce qui peut prendre beaucoup de temps atteindre jusqu'à 20 à 30 minutes.

A partir de milieu des années 60 les problèmes commence a apparu et cet outil preuve son échec.

Parce que la taille des fichiers « recueils principal » explose et produit d'énormes quantités de données redondantes et parmi ces vrai obstacles et problèmes on peut le cite :

- le besoin de synchroniser les données.
- la complexité de gestion.
- la complexité d'ajout de nouveaux programmes et le besoin de matériel adéquat pour gérer cette situation.

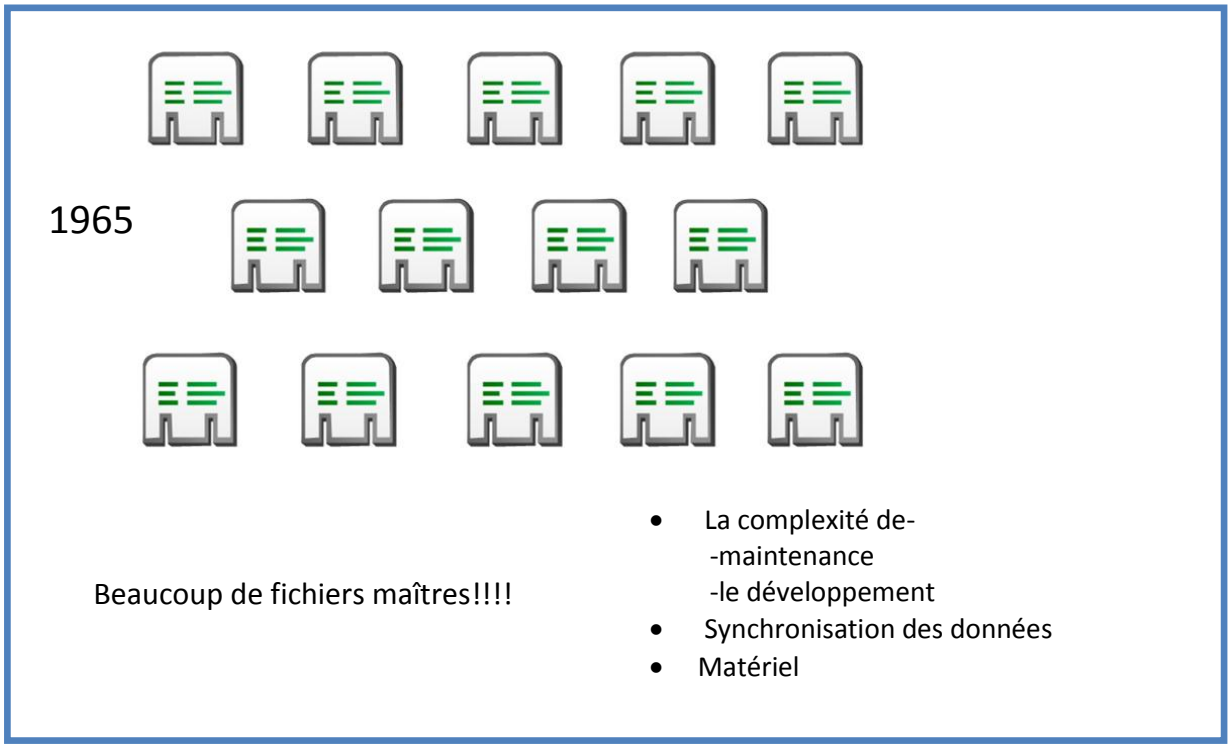

FIG. 1.3 lots de fichiers.

1.1.2 Les systèmes de gestion de base de données

Au début des années 1970, l'émergence des disques à accès direct a fait une véritable révolution technique qui donne une permission d'accès direct aux données, Et c'est ainsi que la réduction du temps d'accès à l'ordre de millisecondes.

De l'autre côté, la partie logicielle, l'apparition d'un nouveau type de logiciel en parallèle avec ces disques. Qu'il consiste aux systèmes de gestion de base de données (SGBD), son rôle est de faciliter le stockage et l'accès aux données.

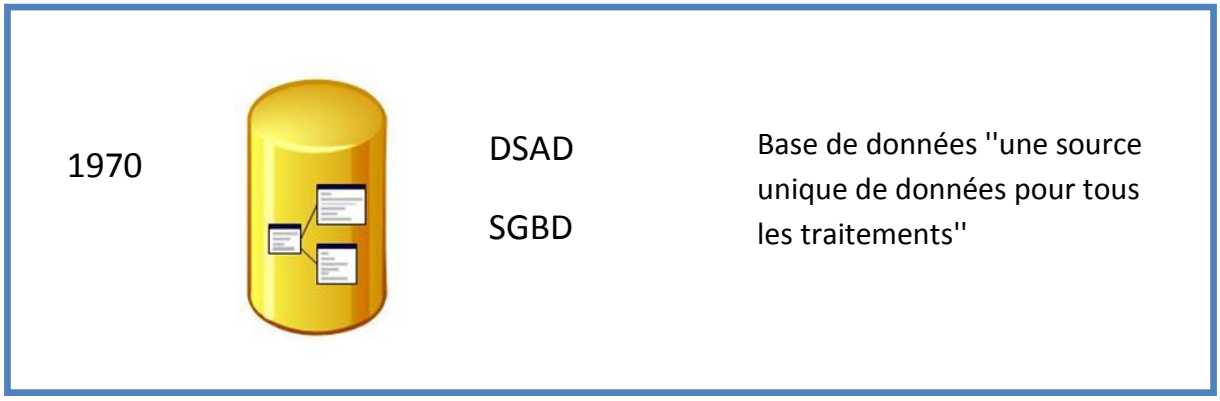

FIG. 1.4 Système de gestion de base de données.

1.1.3 Les systèmes OLTP

A partir de milieu des années 70 un nouveau système a apparu, c'est l OLTP qui rend encore l'accès aux données plus souple et plus rapide, qui influe directement a plusieurs domaines et les permet de bénéficié de ce système tel que : système bancaire, système de réservation …etc.

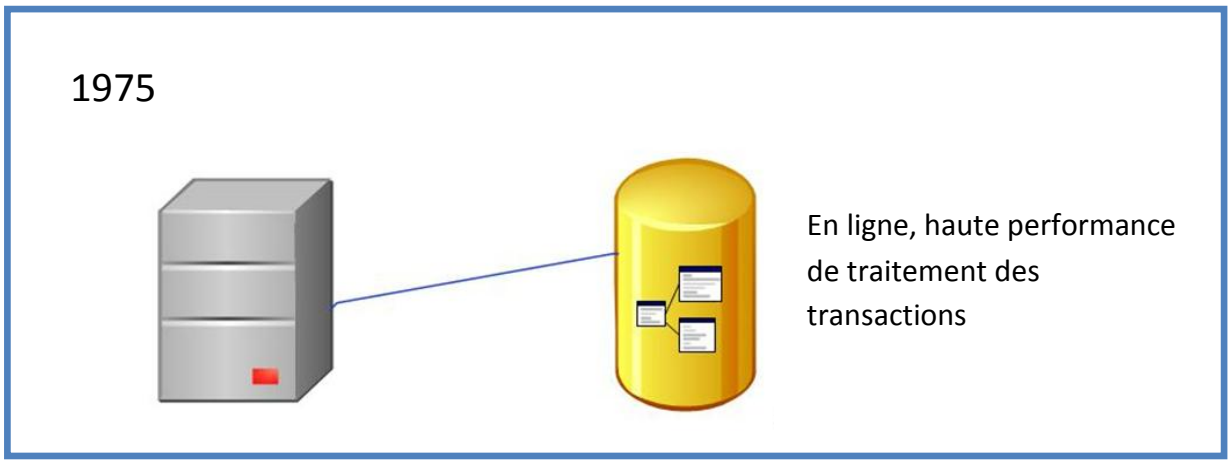

FIG. 1.5 Système OLTP.

1.1.4 Les langages de quatrième génération

Pendant les années 1980, de nouvelles technologies sont apparu, telles que les PC et les langages de quatrième génération et qu'elle apporte des nouveau concepts tel que le changement de rôle de l'utilisateur final par son permission à introduire dans le traitement des transactions.

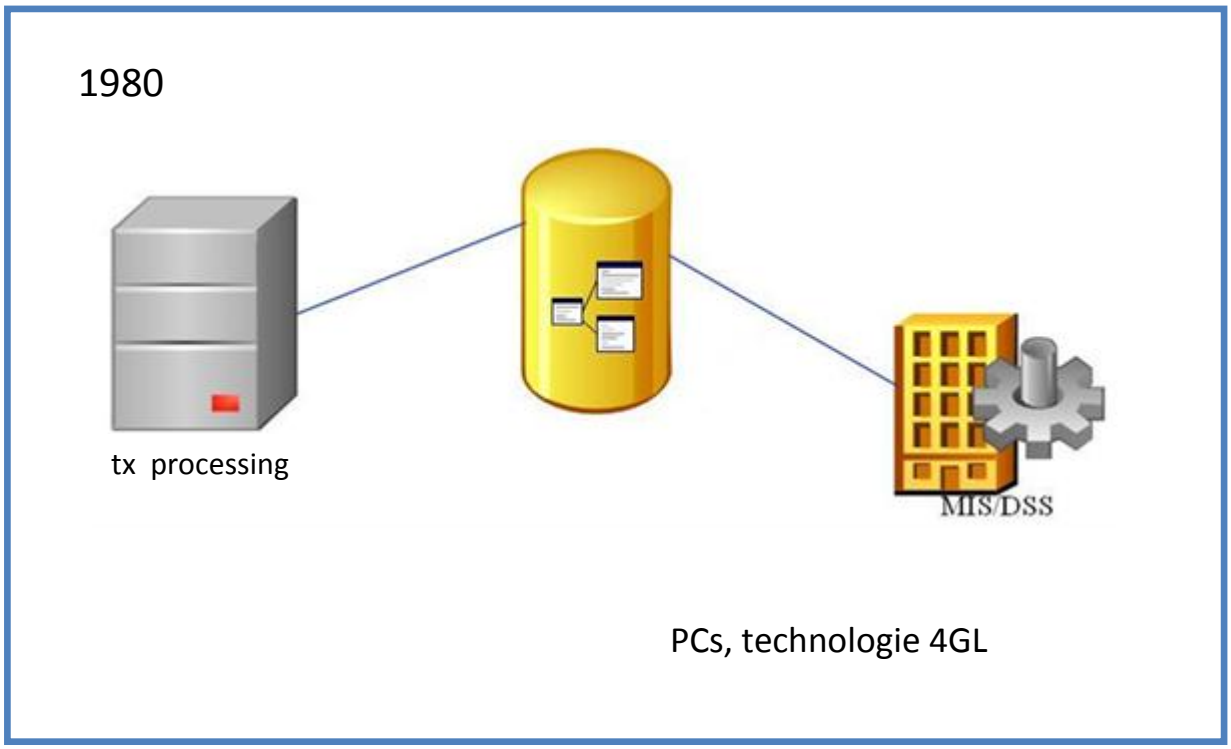

FIG. 1.6 Les langages de quatrième génération.

1.1.5 Les programmes d'extraction des données

L'émergence des systèmes OLTP massif c'était en 1985, simultanément les programmes d'extraction des données ont commencé à apparaître. Ces programmes permettent l'extraction et le chargement des données d'une base de données à un autre, sous certaines conditions.

Le chargement est fait en dehors des heures de travail pour éviter d'affecter négativement les performances des sources de base de données.

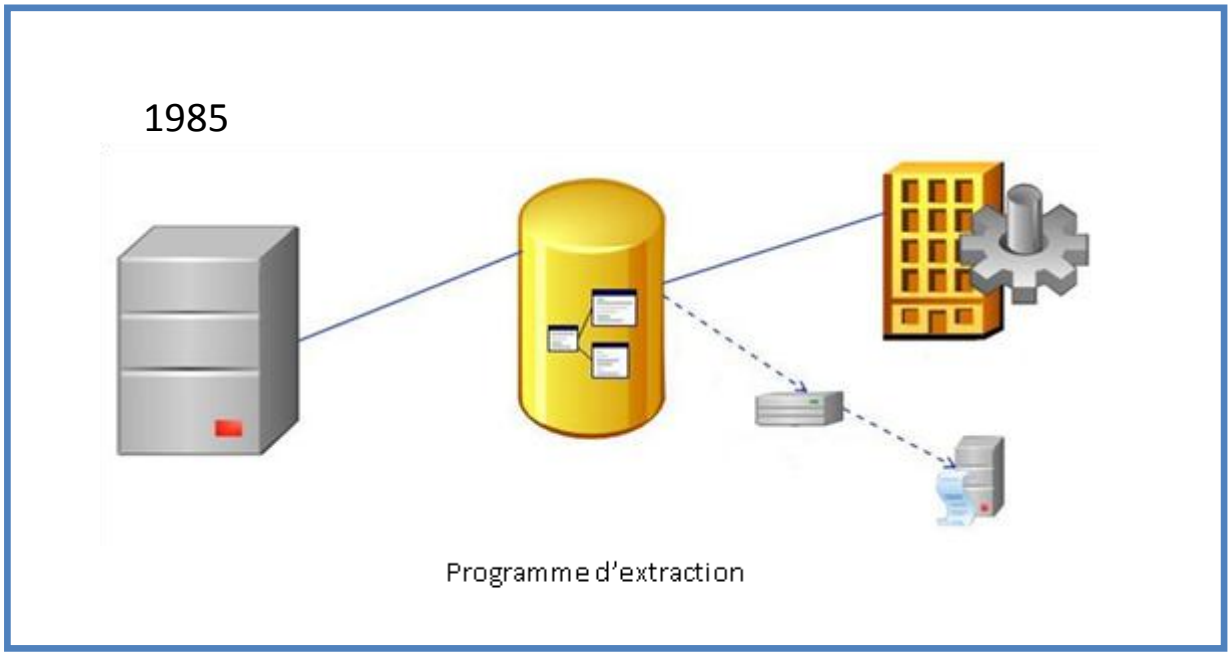

FIG. 1.7 Les programmes d'extraction des données.

La progression des besoins des décideurs en analogue les systèmes d'information ont résulte la création de plusieurs niveaux d'extraction varié entre les simples et les complexes. L'intendance de cette complexité Peut être considérée comme un problème pour les informaticiens de point de vue logiciel et même matériel.

Ces problèmes sont subdivisés en trois sortes :

1- Des problèmes liés à la qualité des données :

- La masse des sources de données et son hétérogénéité.
- les programmes d'extraction.
- les niveaux d'extraction et le manque de synchronisation de diverses bases de données fournis fréquemment des données de pauvre qualité.

2- Des problèmes liées au rendement pour que l'utilisateur réussir dans la construction des situations décisionnels il est obligé de passer par trois étapes :

- repérer les niveaux et les bases de données sur lesquelles son analyse est basée.
- Exiger à l'informaticien d'engendrer les programmes nécessaires pour la création de l'état.
- Disposer de ressources indispensables pour l'exécution de ces programmes.

3- Des problèmes liés à la difficulté de convertir les données en information et les informations en connaissances

Ce n'est pas le même principe entre les deux systèmes opérationnels et décisionnels parce que le décideur a encore besoin de plusieurs opérations ou un processus complet consiste à convertir les données en informations et puis transformer ces informations en connaissances et enfin le préparé pour prendre la décision.

Le facteur le plus important qui inclut a cette transformation c'est le temps donc cette dernier est basé sur des données historiques, Les systèmes opérationnels entreposer les données courantes, et rassemble les données historiques dans des appuis externes. Le rapport des données courantes et des données historiques devient alors une opération terrible. Donc il ne suffit pas de faire rassembler les données dans une même base, mais il faut restructurer la base de données en mettant en avant l'aspect « temps » qui est le pivot de l'analyse.

1.2 Sid vs transactional

L'informatique décisionnelle est apparue dernièrement et devient un domaine en plein développement. Le choix de systèmes d'aide à la décision (SAD) est de plus en plus fort, au vu de l'accroissement exponentiel de données traitées par les entreprises.

1.2.1 Système d'information (SI)

Le système d'information transactionnel est formé de l'ensemble des modules et applications métiers de l'organisation qui gèrent les données au quotidien. Il garantit la gérance de toutes les transactions qui ont lieu au cœur de l'organisation. Ce deuxième groupe intègre le dimension métier de l'organisation. Ces systèmes ne sont ni capable ni adaptés pour faire des analyses complexes de données [1] et ne préparent pas les données pour la prise de décision. Les anciennes approches s'avèrent donc rapidement insuffisantes et les décideurs on besoin aux systèmes qui facilitent leur processus de prise de décision tous cela entre dans le cadre de garantir une plus grande réactivité et une plus grande compétitivité du système.

Définition 1.1 (*Système d'information*) Le système d'information est l'assortiment des procédés et moyens collectant, contrôlant et distribuant les informations nécessaires à l'exercice de l'activité en tout point de l'organisation. Sa tache principale consiste a la production et la mémorisation des informations, et de représenter de l'activité du système

opérant (système opérationnel), puis de les mettre à la porté du système de décision (système de pilotage)[2].

1.2.2 Système d'information décisionnel (SID)

On peut définir Le système d'information décisionnel comme un ensemble bien structuré et organisé d'informations, aisément accessible, qui est approprié pour la prise de décision. Il est extrait à partir des opérations de traitements sur les données de systèmes d'information transactionnels internes ou externes à l'organisation.

Définition 1.2 (*Système d'information décisionnel*) Un système d'information décisionnel, SID, est un système qui réalise la collecte, la transformation des données brutes issues de sources de données et le stockage dans d'autres espaces ainsi que la caractérisation des données résumées en vue de faciliter le processus de prise de décision.

Le grand intérêt de ce genre de système, est que il Ya une possibilité de poser une large gamme de questions vers le système, certaines caractères sont planifiées et prévisibles et d'autres imprévisibles, et de permettre à l'utilisateur de réaliser les requêtes qu'il envisager traiter, par lui-même, sans l'aide ou l'intervention de programmeurs. Ils doivent garantir une cohésion globale des données. Pour ce faire, leur alimentation doit être un Processus bien étudié et planifiée dans le temps. Les chargements de données dans le système opérationnel à partir du système décisionnel seront réguliers avec une périodicité bien choisie et relative à l'activité de l'entreprise. A l'inverse des systèmes transactionnels, aucune information n'y est jamais modifiée.

En résumé, contrairement au système d'information classique où on trouve que les données opérationnelles sont souvent disséminées et éparpilles dans l'entreprise dans des systèmes varié entre l'homogène et l'hétérogènes, le SID se construit à partir de ces données. Il récupère ces données provenant de ces sources et les prépare pour le système de décision. Pour ce faire, le Système d'Information Décisionnel va prendre en charge trois fonctions : extraction des données, leurs stockages et la restitution des données sous une forme exploitable. Le stockage sert à structurer les données au sein d'un entrepôt de données. Il s'agit de mettre en place un schéma relationnel orienté décisionnel.

Le tableau suivant montre la différence d'usage entre les systèmes transactionnels et ceux décisionnels

Systèmes transactionnels	SID
Assure l'activité au quotidien	**Permet analyse et prise de décision**
Pour les opérationnels	**Pour les décideurs**
Mises à jour et requêtes simples	**Lecture uniquement et requêtes complexes transparentes**
Temps de réponse immédiats	**Temps de réponse moins critiques**
Faible volume à chaque transaction	**Large volume manipulé**
Conçue pour la mise à jour	**Conçu pour l'extraction**
Usage maîtrisé	**Usage aléatoire**

TAB 1.1 la différence de l'usage entre SID vs transactionnels.

Ce dernier constitue la partie la plus important et la plus complexe du processus de développement des SID. C'est la raison pour laquelle, nous nous intéressons à cette partie du système d'information décisionnel.

De ce fait, il est indispensable d'élaborer une structure pour les héberger, les organiser et les restituer à des fins d'analyse. Cette structure est l'entrepôt de données ou «Datawarehouse».

Conclusion

On peut considérer que les années 90 est une grand tournant entre deux époques a cause de l'apparition de deux concepts fondamentaux. Le Premier est la possibilité de concevoir des environnements spécialisés pour l'aide à la décision et le Deuxième c'est que plusieurs algorithmes permettant d'extraire des informations à partir de données brutes sont arrivés à maturité. Certains sont provenant de l'intelligence Artificielle, d'autres des statistiques et d'autres sont inspirés de phénomènes biologiques. Tous ces algorithmes sont collectés et organiser dans des logiciels de fouille de données (data mining) qui nous offre une possibilité de l'exploite ces données d'une manière formidable par la recherche d'informations cachées ou nouvelles à partir de données. Les données sont alors organisées dans de grandes bases de données historisées, appelées entrepôts de données.

Les entrepôts de données et la fouille de données sont les éléments d'un domaine de recherche et de développement très actifs actuellement : l'extraction de connaissances à partir de données en raison de l'énorme potentiel et les moyens accordés pour obtenir l’information indispensable avec un temps raisonnable et dans d'excellentes conditions.

2

Les Entrepôts de Données

Introduction

Actuellement, le concept d'entreposage de données et l'analyse en ligne constituent des techniques essentielles intervenant dans les processus de prise de décisions. Cependant, le développement constant des technologies de l'information et de la communication ainsi que la part grandissante de l'informatique dans la plupart des champs disciplinaires produit toujours plus de données, offrant un nombre croissant de sources d'informations disparates, fortement évolutives, distribuées, changeantes et autonomes. Cette évolution remet en cause les techniques actuelles et nécessite la définition de nouvelles approches pour les architectures, l'intégration, la modélisation, l'interrogation et l'optimisation.

Dans ce chapitre, nous nous attacherons à présenter les principales notions relatives aux entrepôts de données. Dans les premières sections nous présentons la définition, les caractéristiques, l'architecture, les spécificités et les objectifs d'un entrepôt de données. Nous décrivons par la suite la modélisation multidimensionnelle utilisée pour la conception des entrepôts de données ainsi que les différents modèles de données associés aux entrepôts puis au cycle de vie d'un entrepôt et on finir de voire quelque grands projets de recherche .

2.1 Le data warehouse

La notion d'entrepôt de données a apparue au début des années 90, il est devenu l'espèce qui joue le rôle essentiel au sein de ce que l'on appelle l'informatique décisionnelle. Et puis ce concept a vécu un énorme succès grâce a l'évolution de l'informatique dans les organisations.

Le processus consiste à fournir aux gestionnaires des organisations sois entreprise ou autre, tout les données qui ont été accumulées dans les fichiers ou d'une maniéré plus globale les bases de données des systèmes transactionnels depuis plusieurs années et qui ne peuvent être traités de manière efficace.

L'acquisition de ces données a été coûteux pour les entreprises et leurs gestionnaires sont de plus en plus conscients de ce qu'ils perdent en n'ayant pas accès pour prendre leurs décisions dans les bons conditions .en effet, ces données contient des informations cruciales aussi bien sur le plan stratégique que sur le plan opérationnel. C'est ce qu'on appelle la transformation des données en information.

2.1.1 Définition

L'entrepôt de données était la solution pour répondre aux exigences de l'entreprise.
Il a été formalisé dans les débuts des années 90 par Inmon Inmon et Richard Hackathorn.
Qu'ils propose ça définition comme suit : «*Un entrepont de données est une collection de données orientées sujet, intégrées, non volatiles, historisées et organisées pour supporter un processus d'aide a la décision*» [29].

Cela a été d'établir une base de données orientées sujet, intégrée, contenant des informations historiées, non volatiles et se préoccupe exclusivement à soutenir les processus décisionnels. En dehors de l'utilité de stocker de grandes quantités de données, l'entrepôt de données offre un moyen de visualiser, d'explorer et de résumer les données qui y sont stockées par l'analyse en ligne OLAP.

L'intérêt d'utilisation des entrepôts de données consiste à fournir un accès aux données même si la source de ces données n'est pas accessible, ce qui limiter l'accès à distance aux systèmes de gestion des données sources.

La fonction essentielle de l'Administrateur est de réduire et limiter l'accès aux sources de données qui se caractérisent aussi d'être hétérogènes et volumineuses par l'utilisation de techniques d'optimisation des requêtes.

Parmi les techniques existant, nous voyons dans cette mémoire les indexes et les vues matérialisées et la fragmentation. Il existe d'autres techniques telles que le clustering et le traitement parallèle des requêtes.

2.2 Caractéristiques des EDS

En retour de la notion précédente d'un entrepôt qui rassembler tout les caractéristiques suivante à la collection de données : orientées sujet, intégrées, non volatiles et basé aussi sur le critère de temps, organisées pour le support d'un processus d'aide à la décision, pour bien illustré ces caractéristiques on retourne au chercheur bill inmon qui explique les points de cette définition. Les données d'un entrepôt sont donc :

- ***Orientées sujet*** Les données sont orientées domaine et donc classées par thèmes.

L'incorporation et unification dans une structure unique est nécessaire pour contourner aux données relatif à plusieurs sujets d'être dupliquées. Cependant, dans la réalité, il existe également des entrepôts plus petits que l'on s'appelé les magasins de données (data marts) :

L'entrepôt est subdivisé en plusieurs bases qui peuvent supporter l'orientation sujet. Le major intérêt de cette structure est de pouvoir organiser de la totalité des informations utiles sur un thème ou sujet le plus fréquemment transversal aux deux structures principales de l'entreprise sois fonctionnelles et organisationnelles.

- ***Intégrées*** Les données niché dans le système proviennent de nombreuses sources disparates et hétérogènes.

Avant d'être intégrées dans l'entrepôt, les données doivent êtres mises en forme et unifiées afin d'avoir un état cohérent. Cela nécessite un gros travail de normalisation, de gestion des référentiels et de cohérence. Une donnée doit avoir une description et un codage unique. Cette phase d'intégration ou de nettoyage des données est très complexe et représente 60 à 90% de la charge totale d'un projet.

- ***Non volatiles*** L'un des principaux caractères de ce nouveau concept nommé l'entrepôt de données c'est sa stabilités de données parce que, ce dernier sont non

modifiables et très rare a changer. D'une autre maniéré on peu dire que un entrepôt de données doit garantir qu'une requête lancée à différentes dates sur les mêmes conditions guide toujours vers les mêmes résultats. De plus, les données d'un entrepôt sont mises à jour périodiquement, donc ces données ne sont pas exploités en temps que informations en temps réel.

- ***Historisées*** Les données sont historisées et datées donc le facteur de temps joue un rôle primordiales: l'historisation est nécessaire pour suivre simultanément l'évolution des différentes valeurs des indicateurs à analyser avec le temps. Ainsi, un référentiel temps doit être associé aux données afin de permettre l'identification de valeurs précises dans la durée.
- ***Aide à la décision*** le rôle principale d'un entrepôt de données ce n'est pas le stockage de données mais ce nouveau concept est destiné essentiellement pour un but prédéterminé c'est l'aide à la décision, donc les opérations et les traitements qui s'y appliquent sont de nature totalement différente par rapport aux systèmes transactionnels.

Les traitements transactionnels ne touche dans la majorité des cas que quelques données, amènent des changements dans la base de données et nécessite une réponse presque instantanée en revanche de traitements mis en œuvre pour l'aide à la décision qui exiger la lecture de nombreuses données mais n'engendre pas de changement dans la base de données et ne requièrent pas une réponse instantanée.

2.3 Entrepôt de données vs base de données

Les applications qui interagit avec les entrepôts de données sont très différentes aux celle exploités avec les bases de données opérationnels selon plusieurs point de vue. Par conséquent, les SGBD relationnels orientés vers l'environnement opérationnel, ne peuvent pas être directement transplantés dans un système d'entrepôt de données.

L'objectif principal de ce nouveau concept est de stocker les données pertinentes et le rendre prêt pour répondre aux besoins de prise de décision. Par contre aux SGBD qui sont conçues pour supporter des opérations journalières, un entrepôt est conçu pour supporter des opérations d'analyse utile à la prise de décision.

Le tableau suivant résume les différences entre les systèmes de gestion de bases de données et les entrepôts de données.

Caractéristiques	Systèmes opérationnels	Entrepôts de données
Objectif	Supporter les opérations courantes de l'entreprise	Supporter le processus de décision des gestionnaires
Type de données	Représentation courante de l'état de l'entreprise	Historiques ou moment précis dans le temps
Principaux utilisateurs	Commis, vendeurs, administrateurs, employés	Gestionnaires, analystes, clients, Décideurs
Envergure de l'utilisation	Étroit, simples mises à jour et requêtes	Large, requêtes complexes et analyse
But de la conception	Performance	Facilité d'accès et d'utilisation
Taille	Méga octets	Tera octets
Nombre d'utilisateurs	Mille	Cent
Mode d'accès	Lecture/Ecriture	Lecture

TAB 2.1 Différences entre une base de données et un entrepôt de données.

2.4 Architecture d'un entrepôt de donnes

L'architecture d'un ED est souvent basée sur un SGBD distinct au système de production de l'entreprise qui contient les données de l'entrepôt. Le processus d'extraction est utilisé pour fournir des données périodiquement à ce SGBD. Toutefois avant d'exécuter ce processus, une phase de transformation est appliquée aux données opérationnelles.

Cette phase consiste concrètement à les préparer (mise en correspondance des formats de données), puis il passe par quelque étapes parmi eux le nettoyage, le filtrage,..., pour arrive finalement à les stocker dans l'entrepôt.

La figure (**FIG.2.1**) ci-dessous illustre en quatre axes le schéma d'un entrepôt de données.

Ce dernier qui assure la tache de stockage des données brutes ou les modifiées à partir de sources d'information (hétérogènes, distribuées).

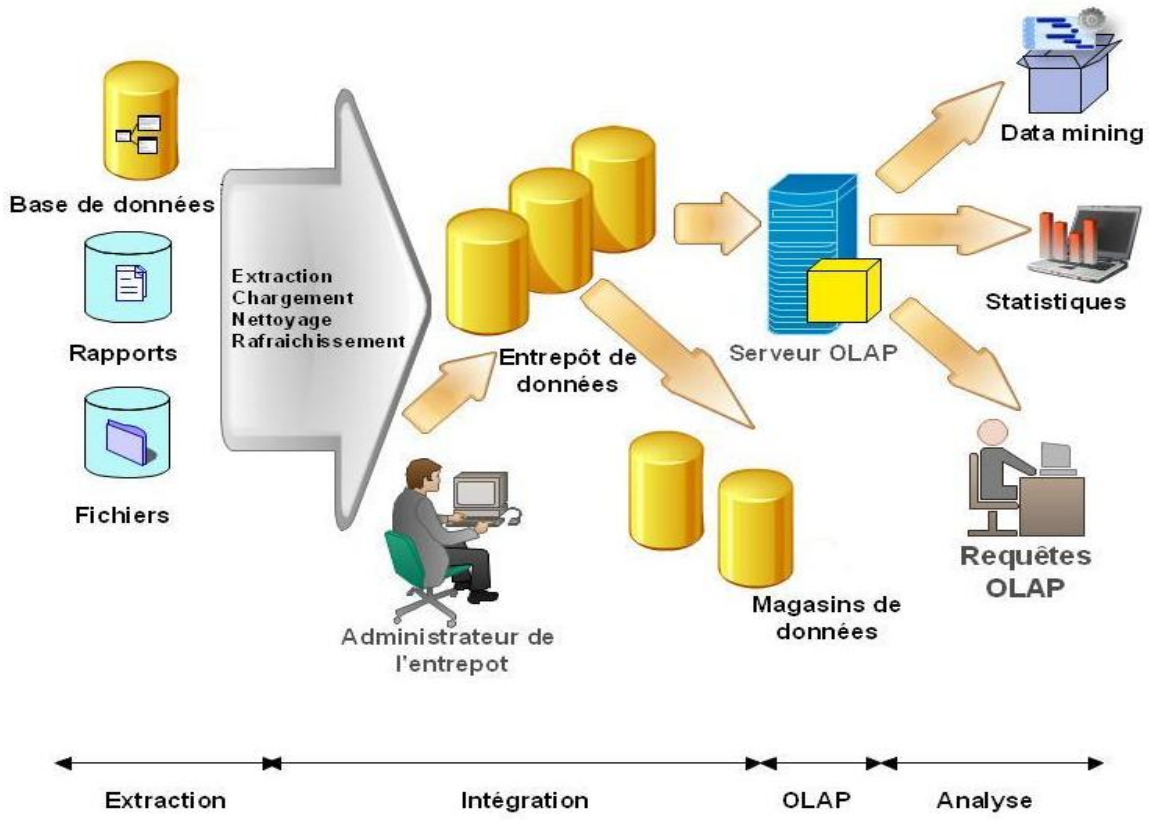

FIG. 2.1 Schéma conceptuel d'un entrepôt de données

- ***Sources de données*** L'entrepôt de données stocke des données provenant de différentes sources d'informations hétérogènes et distribuées. Ces sources peuvent être des bases de données, des fichiers de données, des sources externes à l'entreprise.
- ***Stockage*** Avant de pouvoir être stockées, les données des sources doivent d'abord être nettoyées. Le processus de *nettoyage* consiste à sélectionner et à épurer les données pour éliminer toute erreur et réconcilier les différences sémantiques entre ces données. Une fois nettoyées, ces données, seront intégrées dans l'entrepôt. Le processus de rafraîchissement consiste à propager vers l'entrepôt, les changements effectués sur les données des sources.

Les données concernant la création, la gestion, et l'usage de l'entrepôt sont stockées dans un répertoire indépendant de l'entrepôt. Ces données sont appelées « méta-données ». Les méta-données peuvent contenir des informations sur les sources et leurs contenus, le schéma des règles de rafraîchissement, les profils et groupes d'usagers.

Un ED peut partager en sous partie il s'appelé magasins de données (data marts). Ces magasins sont des extraits de l'entrepôt global destiné spécialement à un type d'utilisateurs et répondant à un besoin spécifique. Ils sont dédiés aux analyses décisionnelles de type OLAP.

- ***Serveur OLAP*** L'interprétation des requêtes des clients d'une requêtes d'accès à l'ED et fournit des vues multidimensionnelles des données à des outils d'aide à la décision. A couver par un serveur OLAP.
- **Outils de front end** La tache finale consiste à l'unification des données par des outils de type front end, conformément aux besoins des utilisateurs, sous différentes formes : tableaux, courbes, rapports, statistiques.

2.5 La modélisation d'un entrepôt de donnes

Les modèles de conception des systèmes transactionnels OLTP ne sont pas adaptés aux systèmes OLAP dont les requêtes sont souvent très complexes, utilisent beaucoup de jointure, demandent beaucoup de temps de calcul et sont de nature ad hoc. Pour ce type d'environnement OLAP, une nouvelle approche de modélisation a été proposée : la modélisation multidimensionnelle. Popularisée par Ralph Kimball dans les années 90, cette modélisation est aujourd'hui reconnue comme la modélisation la plus appropriée aux besoins d'analyse et de prise de décision [6].

Cette nouvelle modélisation est considérer le sujet étudies comme un point dans un espace à quelque dimensions. Donc Les données Constituent de l'entrepôt sont rangés et organisées de manière à mettre en évidence le sujet traités sous différentes axes de l'analyse .Cette modélisation est très intéressante d'une part parce qu'elle représente une modélisation similaire de la façon de penser des analystes, et d'autre part, elle rendre facile aux utilisateurs la compréhension des données. Le modèle multidimensionnel renferme les trois concepts fondamentaux de ***fait*** de ***dimension*** et d'***hiérarchie.***

- ***Concept de fait*** Le sujet analysé est représenté par le concept de fait.

Définition 2.1 : L'utilité de fait consiste à modélise le sujet de l'analyse. Un fait est formé de mesures concernant les informations de l'activité. Les mesures d'un fait sont numériques et généralement valorisées de manière continue. Par exemple, dans le fait Ventes, nous pouvons avoir la mesure "Quantité de produits vendus par magasin".

- ***Concept de dimension*** Le sujet analysé, i.e. le fait, est analysé suivant différentes aspects. Ces aspects correspondent à une sorte utilisée pour indiquer les mesures d'activité étudiés ou analysées, on parle de dimensions.

Définition 2.2 : Une dimension modélise un aspect ou une perspective de l'analyse. Ce qui signifie q' une dimension est composée de certains paramètres correspondant aux informations faisant varier les mesures de l'activité.

- ***Concept hiérarchie*** ce concept est employé par les analyses pour limiter ou accroître les niveaux de détail de l'analyse.

Définition 2.3 : Une hiérarchie est exploité pour l'organisation des paramètres d'une dimension selon sa célèbre relation "est-plus-fin" conformément à leur niveau de détail.

Les deux phases principales pour la création d'un entrepôt de données c'est la conception et la construction, ces deux derniers représentent un enjeu important pour les chercheurs au monde de bases de donné. Cette conception doit concerner les trois niveaux de la modélisation classique, conceptuelle et logique puis physique qui consiste à proposer des techniques d'optimisation adaptées aux données l'envisagé traiter.

La principale application des modèles multidimensionnels est le cube de données.la cellule c'est l'unité élémentaire constituant un cube, chacun de ces cellules représentent une mesure (les attributs du fait). Le cube ci-dessous permet d'analyser les mesures selon les différentes dimensions : produit, marchés et temps. Les hiérarchies définies sur une dimension peuvent être simples ou multiples.

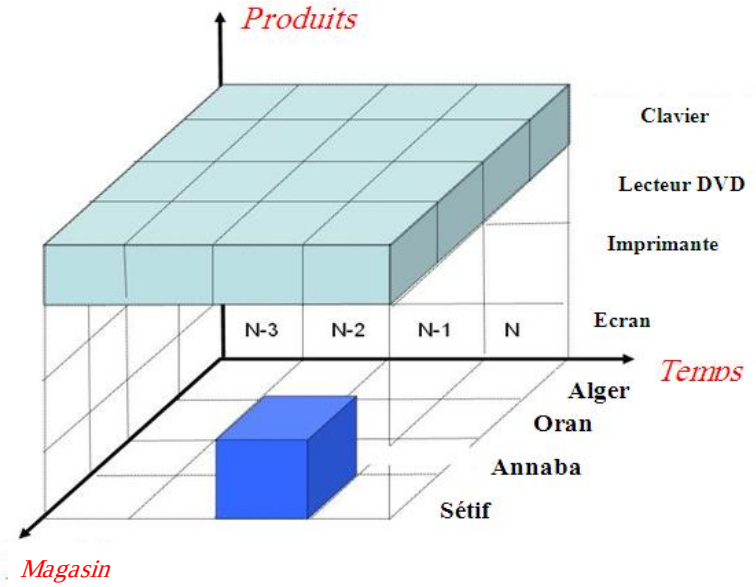

FIG. 2.2 Schéma conceptuel d'un cube multidimensionnel de vente

2.6 Operations sur le cube multidimensionnel

On peut le subdivisé de deux sorte d'opération sur le cube multidimensionnel, les Opérations liées à la structure et les autres liées à la granularité.

2.6.1 Opérations liées à la structure

Les opérations agissant sur cette nouvelle structure de l'information sont motivées par le coté Interactif de l'analyse en ligne de données, et le souci d'offrir des possibilités d'animation de la représentation.

De plus, ce genre d'opérations illustre l'intérêt et l'ampleur des liens entre la manipulation des données et la représentation du cube à l'écran.

Ces opérations sont réunir et rassembler sous le nom de restructuration. Tout cube obtenu à partir d'une opération de restructuration d'un cube primaire nommé cube initial qui contient tout ce qu'il faut pour régénérer et reconstituer le cube initial par restructuration mutuelle. Ces opérations sont : **pivot**, **swich**, **split**, **nest**, **push**, et **pull**. [58]

Pivot Cette opération consiste à pivoter un cube autour d'un des trois axes passant par le centre de deux faces adverses, de sort à présenter l'ensemble de faces différent sous une seul vue.

Switch Selon son terme anglais cette opération assurer d'interchanger directement de position des membres d'une dimension.

Split Elle consiste à présenter chaque tranche du cube, et garantie aussi le passage d'une représentation tridimensionnelle d'un cube vers sa représentation sous la forme d'un ensemble de tables. Donc cette opération permet de réduire le nombre de dimensions d'une représentation d'une façon souple. On notera que le nombre de tables résultant d'une opération split n'est pas connu à l'avance mais dépend des informations contenues dans le cube de départ.

Nest la fonction apparente de cette opération c'est faire d'imbriquer des membres. Mais l'intérêt de cette opération, c'est qu'elle permet de grouper sur une même représentation bi-dimensionnelle toutes les informations soit mesures ou membres d'un cube, quel que soit le nombre de ses dimensions. L'opération réciproque, "unnest", reconstitue une dimension séparée à partir des membres imbriqués.

Push la tache principale de cette opération c'est faire combiner les membres d'une dimension avec des mesures du cube, D'une manière générale, cette opération permet de passer des membres comme contenus de cellules. L'opération accompagner de ce dernier appelé pull, permet de changer le statut de certaines mesures d'un cube en membres, et de constituer une nouvelle dimension pour la représentation du cube, à partir de ces nouveaux membres.

2.6.2 Opérations associées à la granularité

Le deuxième classe d'opération qui base sur l'aspect de la vision de l'analyste est de hiérarchiser l'information selon différents niveaux de détail appelés niveaux de granularité. Les opérations permettant la hiérarchisation sont : roll-up et drill-down. Ces deux opérations autorisent l'analyse de données différents niveaux d'agrégation en utilisant des hiérarchies associées à chaque dimension.

Roll-up Cette application effectuer pour l'agrégation entre les mesures en allant d'un niveau inférieur de la hiérarchie vers un niveau plus général.

Drill-down cette opération consiste à exhiber les données d'un cube à un niveau inférieur, et donc elle considéré comme l'opération réciproque du roll-up.qui représente le cube sous une forme plus détaillée.

2.7 Les implémentations d'un cube multidimensionnel

Il existe trois approches dans la littérature concernant l'implémentation et l'application d'un entrepôt de données : la première approche c'est Multidimensionnelle OLAP (MOLAP) puis l'approche Relationnelle OLAP (ROLAP) et enfin Hybride OLAP (HOLAP).

2.7.1 L'approche MOLAP

Les systèmes MOLAP (*Multidimensional* On-Line Analytical Processing) exploite le cube sous forme d'un tableau multidimensionnel (SGBD multidimensionnel) ou Chaque dimension du tableau représente comme une dimension du cube.et le major intérêt c'est que les données de chaque cellule sont stockées. Par exemple, le cube multidimensionnel de la figure (**FIG.2.2**) peut être représenté par le tableau multidimensionnel ci dessous:

produit	Temps	Trimestre 1			Trimestre 2			Trimestre 3			Trimestre 4			Total		
	site	N	S	Tot	N	S	Tot	N	S	Tot	N	S	Tot	N	S	Tot
Imprimante		12	34	46	22	36	58	24	37	61	33	55	88	91	162	253
Clavier		29	66	95	44	50	94	56	55	111	44	39	83	173	210	383
Ecran		55	34	89	69	27	96	31	26	57	68	70	138	223	157	380
Total		96	134	230	135	113	248	111	118	229	145	114	309	487	529	1016

TAB 2.2 Représentation du cube multidimensionnel selon MOLAP.

L'avantage décisif d'un système MOLAP est sa performance en temps d'accès (l'accès aux données est direct). Outre le fait que ce système ne présente aucun standard, ses principaux inconvénients sont [7]:

- Le besoin de redéfinir des opérations pour manipuler les structures multidimensionnelles.
- Difficulté de la mise à jour et de la gestion du modèle.

- Consommation de l'espace lorsque les données sont éparses, ce qui nécessite l'utilisation des techniques de compression.

2.7.2 L'approche ROLAP

Les systèmes ROLAP (Relational On-Line Analytical Processing) stockent les données du cube en utilisant un SGBD relationnel. Chaque dimension du cube est représentée sous forme d'une table appelée *table de dimension*. Chaque fait est représenté par une *table de fait*. Les mesures sont stockées dans les tables de faits qui contiennent les valeurs des mesures et les clés vers les tables de dimensions. Par exemple, le cube multidimensionnel de la figure (**FIG.2.2**) peut être représenté par les tables suivantes :

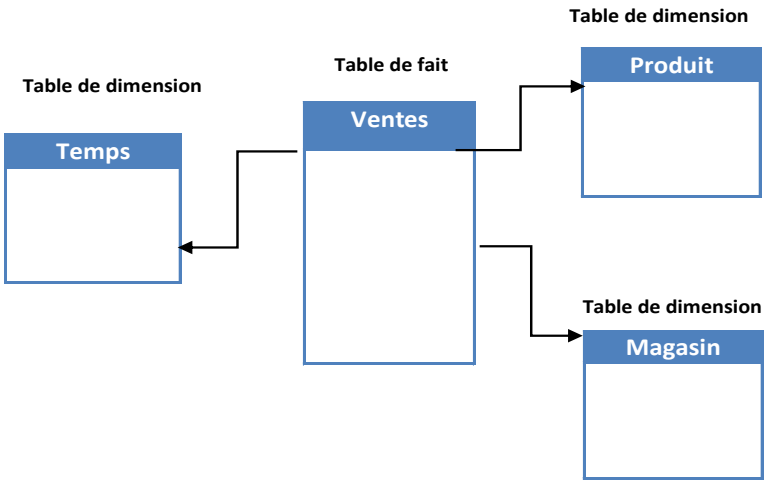

FIG. 2.3 Représentation du cube multidimensionnel selon ROLAP

Le principal inconvénient de ces systèmes est qu'ils peuvent présenter un temps de réponse aux requêtes élevé. Leurs principaux avantages sont :

- Exploitation des capacités d'un standard bien établi et maîtrisé „le relationnel, ce qui facilite leur intégration dans les SGBD relationnels existants.
- Stockage de grandes quantités de données.

Trois schémas sont utilisés pour modéliser les systèmes ROLAP :

- Le schéma en étoile.
- Le schéma en flocon de neige.
- Le schéma en constellation

2.7.2.1 Le schéma en étoile

Chaque dimension du cube est représentée par une table de dimension et les mesures par une table de faits qui référence les tables de dimension en utilisant une clé étrangère pour chacune d'elles. La table de faits est normalisée, les tables de dimension sont généralement dénormalisées. Cette représentation facilite l'analyse selon différentes perspectives. Les requêtes généralement appliquées sur ce schéma sont appelées « requêtes de jointure en étoile », et ont les propriétés suivantes [7] :

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

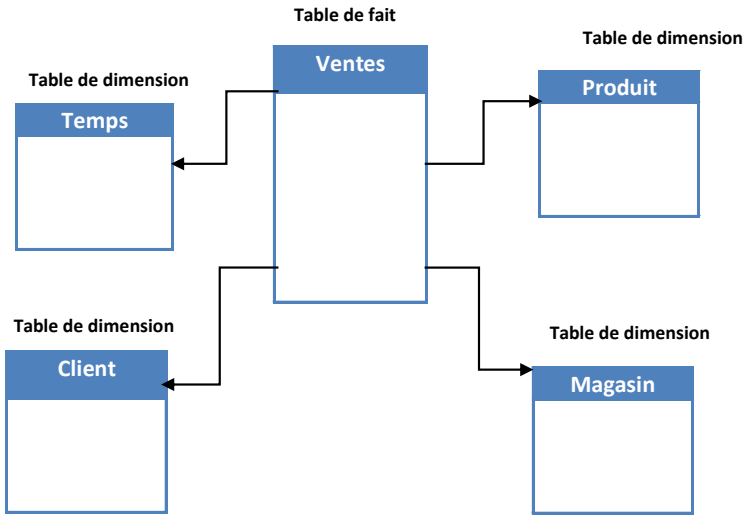

FIG. 2.4 présente un exemple d'un schéma en étoile

2.7.2.2 Le schéma en flocon de neige

On peut considérer que le schéma en flocon de neige est l'extension primaire du schéma en étoile. Puisque dans un schéma en étoile, les informations associées à une hiérarchie de dimension sont représentées dans une seule table, même si les différents niveaux de la hiérarchie ont des propriétés différentes. Mais Le schéma en flocon est le résultat de la décomposition d'une ou plusieurs dimensions en plusieurs niveaux formant une hiérarchie.

Les tables de dimensions dans ce nouveau modèle sont éclatées en plusieurs tables, ce qui peut être vu come une normalisation des tables de dimensions. La table de faits reste la même. Ce type de schéma permet d'une meilleure visualisation et compréhension des données, mais peut altérer les performances de l'entrepôt lors de son utilisation. En effet, une requête nécessitera plusieurs jointures ce qui augmente son temps de réponse.

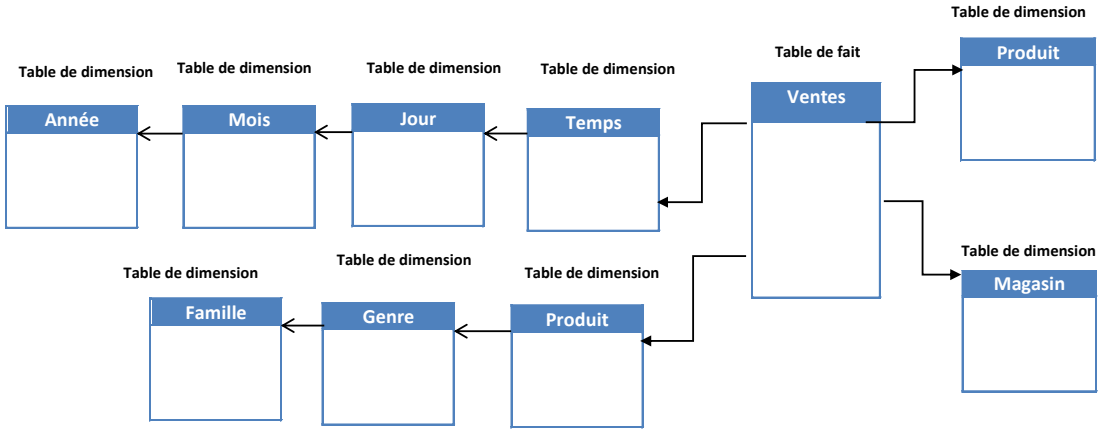

FIG. 2.5 Exemple d'un schéma en flocon de neige

2.7.2.3 Le schéma en constellation

Le dernier genre dans l'approche ROLAP c'est Les schémas en constellations qui se caractérisent par la multiplicité du modèle dimensionnel qui le compose et qui se partagent les mêmes dimensions, c'est-à-dire les tables de faits ont des tables de dimensions en commun. Les tables de dimensions partagées par plusieurs tables de fait doivent être exactement les mêmes.

2.7.3 L'approche HOLAP

Les systèmes HOLAP (Hybrid On-Line Analytical Processing) selon sont nom ''hybrid'' qui rassemble les deux sorts, donc ce type sont des systèmes qui base sur les fréquences d'accès ça veut dire que les données fréquemment utilisées (généralement les données agrégées) sont adhérées par un SGBD multidimensionnel, et les données moins fréquemment utilisées dans un SGBD relationnel. Ceci afin de profiter des avantages des deux systèmes cités précédemment. La séparation des données doit être transparente à l'utilisateur final

Mendelzon propose une comparaison entre les deux approches ROLAP et MOLAP **:**

Stockage	Avantages	Inconvénients
ROLAP	Technologie familière	Lenteur
	Scalable (capacité d'expansion élevée)	
	Ouvert	
MOLAP	Modèle multidimensionnel	Technologie non prouvée
	Traitement de requête spécialisé	Non scalable
	Techniques d'indexation spécialisées	

TAB 2.3 ROLAP Vs MOLAP.

2.8 Cycle de vie d'un entrepôt de données

Le cycle de vie d'un entrepôt de données peut être ordonné en trois phases [8] :

- **Planification** : cette phase vise à préparer l'espace pour le développement de l'entrepôt. Elle inclut les tâches suivantes :

- Déterminer l'étendue du projet ainsi que les buts et objectifs de l'entrepôt à développer.
- Evaluer la faisabilité technique et économique de l'entrepôt.
- Identifier les futurs utilisateurs de l'entrepôt.

- **Conception et implémentation** : cette deuxième phase consiste à développer le schéma primaire de l'entrepôt, et à mettre en place toutes les ressources nécessaires à son implémentation et à son déploiement. Cette conception comporte cinq principales étapes.

- **Maintenance et évolution** : la maintenance de l'entrepôt rassemble à l'optimisation de ses performances périodiquement conformément a nos exigences. L'évolution de l'entrepôt concerne la mise à jour de son schéma selon les différents changements survenant au niveau des sources de son alimentation ou des besoins des utilisateurs.

2.8.1 Conception d'un entrepôt de données

Même s'il n'existe actuellement aucun consensus sur une méthode de conception précise, la plupart des travaux s'accordent à distinguer les étapes suivantes pour la conception d'un ED : analyse des besoins, modélisation conceptuelle, modélisation logique, processus ETL et modélisation physique.

2.8.1.1 Analyse des besoins

L'analyse des besoins joue un rôle clé dans tout projet logiciel en permettant de réduire significativement le risque d'échec du projet [9]. L'ingénierie des besoins a été définie comme le processus de développement des besoins selon un processus itératif et coopératif d'analyse du problème, de documentation des observations résultantes dans divers formats de représentation et de vérification des résultats obtenus [10]. La spécification des besoins d'un projet d'entrepôt de données permet de déterminer quelles seront les différentes fonctions de l'entrepôt ainsi que l'ensemble des informations requises que ce dernier doit couvrir (quelles sont les données qui doivent être accessibles, comment ces données sont transformées, organisées, calculées et agrégées) [11][12]. Deux méthodes sont utilisées pour collecter les besoins : une *collecte orientée source* et une *collecte orientée utilisateur*. Une approche orientée source se limite à identifier les besoins de l'ED à partir de l'ensemble des données disponibles au niveau des sources. La collecte orientée utilisateurs identifie les besoins des utilisateurs et décideurs de l'entreprise et présente l'avantage principal de se concentrer à fournir un modèle répondant à ce qui est exigé plutôt que ce qui est disponible. Un ED, par essence, doit être construit à partir des sources de données. Une collecte orientée sources est par conséquent indispensable. Les besoins utilisateurs n'ont cependant pas toujours été considérés lors du développement des EDs. Ce sont généralement les travaux récents qui prennent en compte les besoins utilisateurs. Même si les méthodologies de développement d'un ED diffèrent généralement, elles reconnaissent de plus en plus la nécessité d'identifier et de modéliser les besoins des utilisateurs [13].

La modélisation des besoins (collectés à partir des sources et/ou des utilisateurs) passe par les activités suivantes [14] : La collecte des besoins, l'analyse des besoins, la validation des besoins et la modélisation des besoins.

- ***La collecte des besoins*** consiste à collecter les besoins souvent selon des approches itératives. La collecte à partir des sources passe par une analyse détaillée des sources

de données. la collecte orientée utilisateur passe par des techniques de réunions, d'étude de la documentation existante, d'interviews et de brainstorming. Ces besoins sont ensuite documentés (généralement de manière informelle). La collecte a pour but de comprendre le domaine qui devra être modélisé.

- ***L'analyse des besoins*** consiste à étudier les besoins collectés, où des modèles initiaux peuvent être construits.

- ***La validation des besoins*** consiste à valider les modèles initiaux construits lors de l'analyse des besoins, avec les utilisateurs ainsi qu'avec les sources de données existantes. L'entrepôt est ainsi construit de manière itérative, ce qui garantit selon [14] un développement rapide. L'expérience a montré que même après implémentation de l'entrepôt de données, un retour vers la phase d'identification des besoins est possible. L'utilisation et la maintenance du modèle de données est effectué continuellement durant tout le cycle de vie de l'entrepôt.

Les travaux sur l'analyse des besoins en ED restent assez limités. Certains travaux suggèrent l'utilisation des techniques utilisés pour l'analyse des besoins en BD, d'autres estiment qu'il faut des méthodes adaptées aux spécificités des EDs. On retrouve quelques travaux portant sur des *méthodologies* présentant des directives générales permettant de mener à bien l'analyse des besoins dans un projet d'ED. Nous présentons deux travaux (*Rilston et al., Winter et al.*) souvent référencés : *Rilston* [15] proposent une méthodologie comprenant quatre phases :

- Phase d'initialisation qui consiste à identifier les utilisateurs cibles ainsi que le type d'application d'analyse le plus dominant ce qui permettra de choisir les modèles de données à utiliser, les sources de données pertinentes et les processus de décisions devant être considérés.

- Une deuxième phase d'analyse des sources de données existantes et les rapports utilisés régulièrement par les utilisateurs, et à développer des schémas de données agrégés.

- Une troisième phase qui consiste à établir des priorités entre les besoins en informations.

- Une quatrième phase consistant à concevoir un modèle des besoins initiaux servant de base au modèle conceptuel.

Winter [16] proposent également une méthodologie comprenant quatre phases :

- ***Planning de gestion des besoins*** consiste à définir les objectifs du projet par les utilisateurs et les concepteurs, définir les règles d'intégration des sources de données et à établir un planning de gestion des besoins du projet (contraintes de temps, délimiter les frontières du projet…).

- ***Spécification des besoins*** s'effectue par un processus itératif d'acquisition, représentation et validation des besoins. La spécification s'effectue à travers plusieurs (sous) processus en plusieurs itérations en utilisant un modèle en spirale, où chaque itération produit un ensemble de spécifications plus raffiné.

- ***Validation des besoins*** par l'équipe de développement, les utilisateurs, et les experts du domaine. L'équipe de validation attache une liste d'actions à chaque problème identifié et des retours arrière vers les phases précédentes peuvent être envisagés.

- ***Suivi et gestion de l'évolution des besoins*** la gestion doit être effectuée à deux niveaux : une gestion de l'évolution des besoins utilisateurs, et une gestion de l'évolution de l'architecture des bases de données sources afin d'étudier l'impact d'éventuels changements sur les modèles suivants (conceptuels, logiques et physiques).

2.8.1.2 Modélisation conceptuelle

Cette étape consiste à établir une méthode permettant de définir le schéma conceptuel d'un entrepôt, annoté par les concepts multidimensionnels (faits, mesures et dimensions). Le but de cette modélisation est de fournir une représentation abstraite de la situation en cours d'étude indépendamment de tout logiciel technique. Un modèle conceptuel est caractérisé par le domaine concerné, le formalisme utilisé pour modéliser le schéma conceptuel, et le point de vue correspondant aux besoins des utilisateurs [17]. Le modèle conceptuel obtenu doit se conformer à la modélisation multidimensionnelle adaptée aux modèles d'EDs permettant d'organiser les données entreposées de manière à faciliter leur analyse décisionnelle.

2.8.1.3 Modélisation logique

Cette étape consiste à dériver le schéma logique de l'entrepôt à partir de son schéma conceptuel, et à l'adapter aux particularités du serveur de l'entrepôt choisi (ROLAP, MOLAP ou HOLAP). Ce passage implique la gestion de certains aspects techniques comme la fragmentation des dimensions en hiérarchies, la gestion de l'agrégation des données, la gestion des dimensions génériques1 et des mesures non additives2.

2.8.1.4 Processus ETL (Extract-Transform-Load)

Cette étape consiste à effectuer les transformations nécessaires au chargement des données des sources au niveau du schéma logique de l'entrepôt. Ceci se fait en trois étapes [18] :

- **Extraction :** consiste à récupérer les données à partir des systèmes sources. Cette étape nécessite de gérer la synchronisation des processus d'extraction afin d'assurer l'intégrité des données chargées.

- **Transformation :** consiste en une série de règles permettant le formatage des données extraites selon le schéma cible de l'entrepôt, comme par exemple assigner de la sémantique aux données sources et associer les champs sources aux champs cibles.

- **Chargement :** permet l'alimentation de l'entrepôt par les données en respectant les contraintes du SGBD cible. Cette étape doit être validée afin de détecter et corriger toute erreur ayant survenue durant le chargement.

2.8.1.5 Modélisation physique

Cette étape consiste à implémenter physiquement le schéma de l'entrepôt, et à spécifier les techniques et schémas d'optimisation de l'entrepôt. Les techniques d'optimisation peuvent être [7] :

- **Les vues matérialisées :** une vue matérialisée est une table contenant les résultats d'une requête.

- **Les techniques d'indexation :** représentées par des structures permettant d'associer à une clé d'un n-uplet l'adresse relative de ce n-uplet. Plusieurs types d'index ont été proposés, on distingue les index définis sur une seule table ou vue (index en B-arbre,

index de hachage et index bitmap) et les index de jointures définis sur plusieurs tables dans un schéma en étoile.

- **La fragmentation :** technique consistant à partitionner le schéma d'une base en plusieurs sous schémas pour réduire le temps d'exécution des requêtes. On distingue la fragmentation horizontale (en fonction des tuples) de la fragmentation verticale (en fonction des attributs).

2.9 Grands projets de recherche

Dans cette partie, nous décrivons les projets de recherche sur les entrepôts de données.

2.9.1 Dwq

DWQ (Foundations of Data Warehouse Quality) [Jarke, et al 1998] [Jeusfeld et al, 1998] c'est l'un des premier projet industriel développer en Europe basé sur des fondements sémantiques qui permettront d'offrir au concepteurs d'entrepôts de données de choisir des modèles, et de combiner des structures de données varié entre simples et avancées et des techniques d'implantation efficaces en s'appuyant sur des facteurs de qualité de service.

2.9.2 Sirius

Le projet SIRIUS (Supporting the Incremental Refreshment of Information warehoUseS) [Vavouras et al, 1999] c'est un projet académique développé en suisse au sein de l'université de Zurich, et peut comporte comme un système d'entrepôt de données qui a pour objectif l'augmentation de performance par l'étude des techniques permettant le rafraîchissement incrémental de l'entrepôt en réduisant les temps de mise à jour.

2.9.3 Squirrel

Squirrel [Zhou et al, 1995, 1996] [Hull et Zhou, 1996] est un projet académique américain developé au niveau de l'université du Colorado, son domaine de recherche consiste à proposer un cadre pour l'intégration de données basé sur la notion de médiateur d'intégration. Ce dernier est un module actif qui supporte des vues intégrées au travers de multiples bases de données.

2.9.4 Tsimmis

TSIMMIS c'est l'abréviation de (The Stanford IBM Manager of Multiple Information Sources)] [Chawathe et al, 1994] [Garcia-Molina et al, 1995] [Papakonstantinou et al, 1995] est un projet académique lui aussi ,développé à l'université du Stanford, son contexte consiste à fournir des outils qui permette d'accès intégré à des entrepôts d'information. Chaque source d'information est équipée d'un raducteur qui encapsule la source, convertissant les objets sous-jacents dans un modèle de données commun.

2.9.5 whips

WHIPS c'est l'abréviation de (WareHouse Information Prototype at Stanford) [Widom 1995] [Hammer et al, 1995] [Wiener et al, 1996] est un système de gestion d'entrepôts de données utilisé comme banc d'essai. L'objectif de ce projet est de développer des algorithmes, le rôle de ces derniers est de collecter, intégrer et maintenir des informations émanant de sources hétérogènes, distribuées et autonomes. L'architecture du prototype WHIPS consiste en un ensemble de modules indépendants implantés comme des objets CORBA.

Conclusion

L'exploite des entrepôts de données et leurs outils deviennent indispensables dans le milieu des entreprises. C'est pour cela on essai dans ce deuxième chapitre présente un état de l'art des notions et concepts fondamentaux concernant les entrepôts de données à travers sa définition, son architecture, ses modèles de données employant le modèle multidimensionnel (MOLAP, ROLAP et HOLAP), et son cycle de développement comprenant trois principales phases (planification, conception et implémentation, évolution et maintenance).

Cette étude nous permet de cerner les principaux concepts clé des EDs dont on aura besoin pour tenter de répondre à notre problématique l'optimisation d'un ED. Le chapitre 3 présente un état de l'art sur les différentes approches et méthodes d'optimisation d'EDs proposées dans la littérature, que nous avons classées et étudier.

3

Classification des principales techniques d'optimisation

Introduction

On a vue au deux chapitre précédent l'avantage majeur que nous a apporté l'informatique décisionnel présentant au entrepôt de donnes notre domaine de recherche et comme tout les nouvelles technologies presque chaque aspect a des qualités et des inconvénients ou des problèmes et le problème au domaine d'entrepôt de données c'est l'agrandissement de volume de données et le délai de réponse de requête de ces entrepôts donc nous somme dans l'essai de trouver des méthode ou technique d'optimisation de ces entrepôt pour cela on essai d'étudier les technique existant de les classer et les comparer .

3.1 Classification des techniques d'optimisation

La conception physique d'une base ou entrepôt de données consiste à établir une configuration physique sur le support de stockage. Cela comprend la spécification détaillée des éléments de données, les types de données et la sélection des techniques d'optimisation.

Cette dernière est au cœur de la conception physique. Dans la première génération de moteurs d'exécution de requêtes dédiés aux bases de données traditionnelles, la conception physique n'avait pas autant d'importance. Aujourd'hui face à la complexité des requêtes à traiter et le volume important des données, la conception physique a reçu une importance phénoménale [19].

Plusieurs techniques d'optimisation ont été proposées dans la littérature et supportées par les systèmes de gestion de bases de données commerciaux. Nous pouvons les classer en deux catégories principales [20,21]: *techniques redondantes et techniques non redondantes*. Les techniques redondantes optimisent les requêtes, mais exigent un coût de stockage et de maintenance. Cette catégorie regroupe les vues matérialisées [22], les index [23], la fragmentation verticale [24,25], etc.

Les techniques non redondantes ne nécessitent ni coût de stockage ni coût de maintenance, cette catégorie regroupe la fragmentation horizontale [25,26], le traitement parallèle [27], etc.

La figure (**FIG. 3.1**) montre une classification des principales techniques d'optimisation.

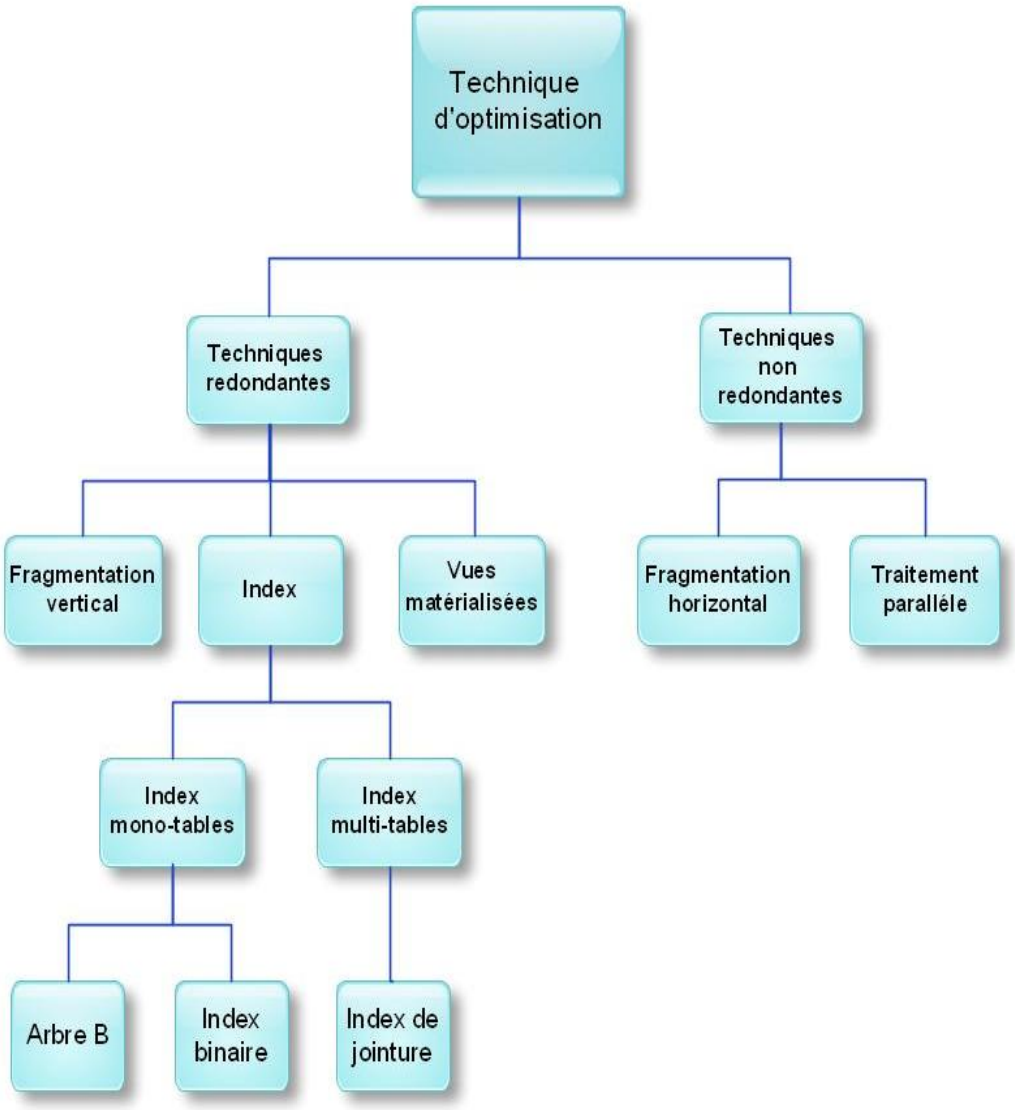

FIG. 3.1 Classification des techniques d'optimisation

Nous avons étalé dans les sections suivantes deux techniques d'optimisation redondantes existant dans la littérature: les index et les vues matérialisées et puis on étudier les caractéristiques de chacun des deux techniques en voir en détails ces propriétés et les deux techniques d'optimisation non redondantes : la fragmentation horizontale en ces deux formes primaire et dérivée. Pour chaque technique, nous illustrons les principaux travaux effectués pour résoudre le problème de sa sélection.

3.1.1 Techniques d'optimisation redondantes

On montre dans cette première partie du chapitre 3 deux techniques d'optimisation redondantes étudiées dans cette mémoire : l'indexation et les vues matérialisées. Et on

focalise également sur leurs problèmes de sélection et présentons des algorithmes proposés pour les résoudre.

3.1.1.1 Les index

Consulter des tables de bases ou d'entrepôt de données qui porte généralement le caractère de volumétrie via un ensemble de requêtes pour accéder à un certain nombre de n-uplets est une tâche fréquente dans le monde d'entrepôt de données. Répondre efficacement à ces requêtes est souvent difficile compte tenu de la nature complexe des requêtes OLAP et des volumes de données. La façon la plus facile d'avancer consiste à appliquer un balayage entier aux tables et vérifier pour chaque n-uplet s'il satisfait le prédicat de la requête. Ce balayage peut être très coûteux lorsque les tables scannées sont volumineuses. Pour surmonter ce problème, plusieurs techniques d'indexation ont été proposées. Un index est une structure redondante ajoutée à la base de données pour permettre les accès rapides aux données. Il permet à partir d'une clé d'index prédéterminé de découvrir l'emplacement physique des n-uplets recherchés.

Parmi les techniques d'indexation déjà proposées dans les bases de données classiques, On peut citer l'index en mode B-tree, l'index de jointure, l'index de hachage, l'index de projection...etc.la plupart de ces sortes d'index est aussi utilisée dans le cadre des entrepôts relationnels. 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…etc.

Les chercheurs dans le monde de bases de données ont classé les index en deux groupes, les index mono-table et les index multi-tables. Les index mono-table sont connais comme des index définis sur un ou plusieurs attributs de la même table, parmi eux les index B-tree, de hachage, binaires, etc.et le deuxième groupe nommé Les index multi-tables sont des index définis sur plusieurs tables comme les index de jointure standards, en étoile et binaires.

3.1.1.1.1 Techniques d'indexation

Nous présentons dans cette section les techniques d'indexation les plus célèbres comme les index B-arbre, les index binaires, les index de jointure, etc.

3.1.1.1.1.1 Index B-arbre

L'index B-arbre est considéré comme l'index par défaut pour la majorité des SGBD commerciaux [28]. Cet index est structuré comme un arbre à plusieurs niveaux.et que chaque nœud d'un niveau pointe directement vers le niveau inférieur. Les nœuds feuilles (niveau le plus bas) contiennent les entrées d'index accompagné d'un pointeur qui pointer à l'emplacement physique de l'enregistrement correspondant (généralement un identifiant physique, ROWID). La figure (**FIG. 3.2**) représente un exemple d'index B-arbre construit sur l'attribut PID de la table Produit.

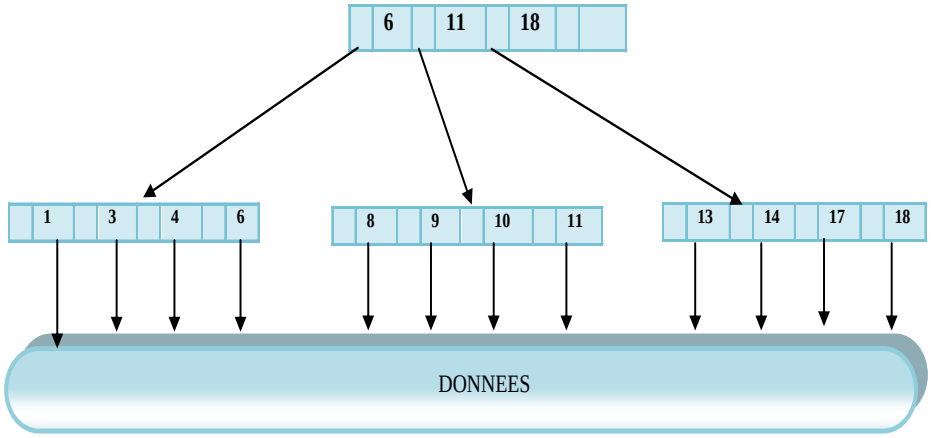

FIG. 3.2 Index B-arbre

3.1.1.1.1.2 Index de projection

Un index de projection [27] est défini sur un ou plusieurs attributs d'une table. Il repose sur le stockage de toutes les valeurs de ces attributs dans l'ordre de leur apparition dans la table. D'une manière générale, les requêtes accèdent à un sous-ensemble d'attributs d'une table. Si ces attributs sont contenus dans un index de projection, l'optimiseur ne charge que cet index pour répondre à la requête. La figure (**FIG. 3.3**) montre un index de projection défini sur l'attribut Ville de la table Client.

TABLE CLIENT				
CID	Nom	Age	Sexe	Ville
616	Rafik	15	M	Sétif
515	Rachida	25	F	Alger
414	Kamel	33	M	Oran
313	Samir	50	M	Oran
212	Sarah	21	F	Sétif
111	Farid	20	M	Sétif

Index de projection sur l'attribut ville
ville
Sétif
Alger
Oran
Oran
Sétif
Sétif

FIG. 3.3 Index de projection

*Sybase IQ*exploite l'index de projection, il porte le nom *Fast Projection Index* [30].
Le principe de ce genre d'index c'est que un index de projection est créé par défaut pour chaque colonne de la table, de cette propriétés en remarque que Ce type d'index est très approprié pour les SGBD qui travaille avec le stockage des données par colonnes, comme Sybase IQ qui propose deux manières différentes d'implémenter cet index, une implémentation en bitmap pour les attributs de faible cardinalité et un B-tree pour les attributs de forte cardinalité.

3.1.1.1.1.3 Index de hachage

Ce genre d'index basé sur l'application d'une fonction de hachage. Cette fonction permet d'utilisé le clé c et à partir d'une valeur de ce dernier, de donner l'adresse équivalente f(c)d'un espace de stockage où l'élément doit être placé. La figure (**FIG.3.4**) présente un index de hachage crée sur l'attribut PID d'une table Produit. Dans ce type d'index,un facteur est très important c'est le choix de la fonction de hachage, pour assurer une meilleur performance de l'index. Par exemple, si la fonction donne la même valeur à plusieurs éléments, alors l'accès devient proche d'un balayage séquentiel.

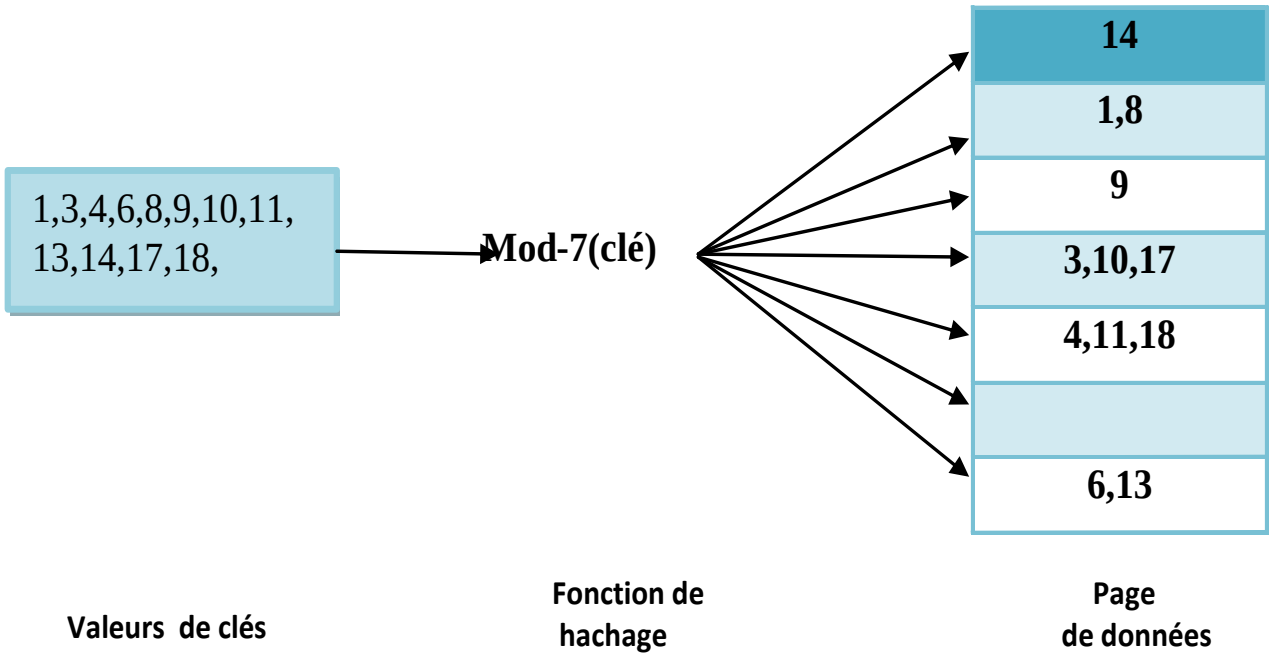

FIG. 3.4 index de hachage

3.1.1.1.1.4 Index binaire (bitmap index)

La technique d'index binaire basé sur l'emploi d'un ensemble de vecteurs binaires (contenant juste des valeurs 0 ou 1) pour indiquer l'ensemble des n-uplets d'une table. Pour chaque valeur de l'attribut indexé, un vecteur de bits dit bitmap est stocké. Ce vecteur contient autant de bits qu'il y a de tuples dans la table indexée. L'index binaire a été considéré comme le résultat le plus important obtenu dans le cadre de l'optimisation de la couche physique des entrepôts de données [31].

TABLE CLIENT				
CID	Nom	Age	Sexe	Ville
616	Rafik	15	M	Setif
515	Rachida	25	F	Alger
414	Kamel	33	M	Oran
313	Samir	50	M	Oran
212	Sarah	21	F	Setif
111	Farid	20	M	Setif

Index binaire sur L'attribut ville		
Setif	Alger	Oran
1	0	0
0	1	0
0	0	1
0	0	1
1	0	0
1	0	0

FIG. 3.5 Index binaire

Supposons un attribut A ayant n valeurs distinctes $v_1, v_2, \ldots\ldots, v_n$($n$ est appelé cardinalité de A) en relation avec une table T constitué de m instances. La création de l'index binaire IB défini sur A se fait selon les étapes ci dessous :

1. Créer n vecteurs composés chacun de m entrées ;
2. Pour chaque tuple i, (1 ≤ i ≤ m) dans T, si $i.A = v_k$,alors mettre 1 dans le $i^{éme}$bit du vecteur correspondant à v_j , mettre 0 dans les i ^{ème} bits des autres vecteurs.

Si une requête recherche les tuples vérifiant un prédicat d'égalité (par exemple$A = v_k$), alors il suffit de lire le vecteur associé à v_k , chercher les bits ayant la valeur 1, ensuite accéder aux tuples correspondant à ces bits. La figure (**FIG.3.5**) montre un exemple d'index binaire construit sur l'attribut Ville de la table Client. Par exemple, la ville du premier tuple est Sétif, par conséquent la première case du vecteur correspondant à Sétif sera mise à 1, les cases des autres vecteurs sont mises à 0. Cet index pourra être construit à l'aide de la commande SQL suivante :

CREATE BITMAP INDEX Client_Ville_BI_idx
ON Client.ville;

Ce type d'index est très efficace pour les requêtes de type count(*) où seule la lecture de l'index suffit pour répondre à ces requêtes. Par exemple, soit la requête Q suivante :

SELECT count(*)

FROM Client
WHERE Ville='Sétif'.

Pour exécuter cette requête, l'optimiseur lit le vecteur correspondant à la ville de Setif et retourne le nombre de 1 trouvés (il retourne 3).

La nature binaire de ce type d'index permet d'améliorer les performances des requêtes en permettant d'appliquer des opérations logiques AND, OR, NOT, etc. Ces opérations permettent de rechercher des tuples vérifiant des conjonctions ou des disjonctions de prédicats.

3.1.1.1.1.5 Index de jointure

L'opération de jointure est l'une des opérations souvent répétitives dans les systèmes OLAP. donc c'est une action très coûteuse et pénible, puisqu'elle traite de grands volumes de données. Plusieurs applications de la jointure ont été implémentées lors des bases de données classiques :

Chacun des boucles imbriquées et des fonctions de hachage et de tri-fusion ce sont des implémentations limitées et moins performants a cause d'expansion de la taille des tables concernées par cette opération de jointure. Le chercheur dans le domaine de bases de données Valduriez a proposé un index de jointure qui fait l'opération de précalcule la jointure entre deux tables [32].puis L'index de jointure matérialise les liens existant entre deux tables par l'utilisation d'une table à deux colonnes ou chacune correspond à l'identifiant d'une table. Soit X et Y deux tables qui peuvent être jointes par les attributs $X.a$ et$Y.b$. L'index de jointure est l'ensemble des n-uplets$< X.ID_i, Y.ID_j >$ tel que les tuples de X et Yayant respectivement pour identifiantID_iet ID_jvérifient la condition de jointure. L'exploitation de l'index de jointure IJ entre X et Y se fait de la manière suivante :

- Charger l'index de jointure IJ.
- Effectuer la semi-jointure X⋉ IJ.
- Effectuer la semi-jointure Y⋉ IJ.
- Effectuer la jointure des deux résultats.

On suppose que la taille de l'index de jointure dépend de la sélectivité de la jointure. Si la jointure est de forte sélectivité alors la taille de l'index est très petite.

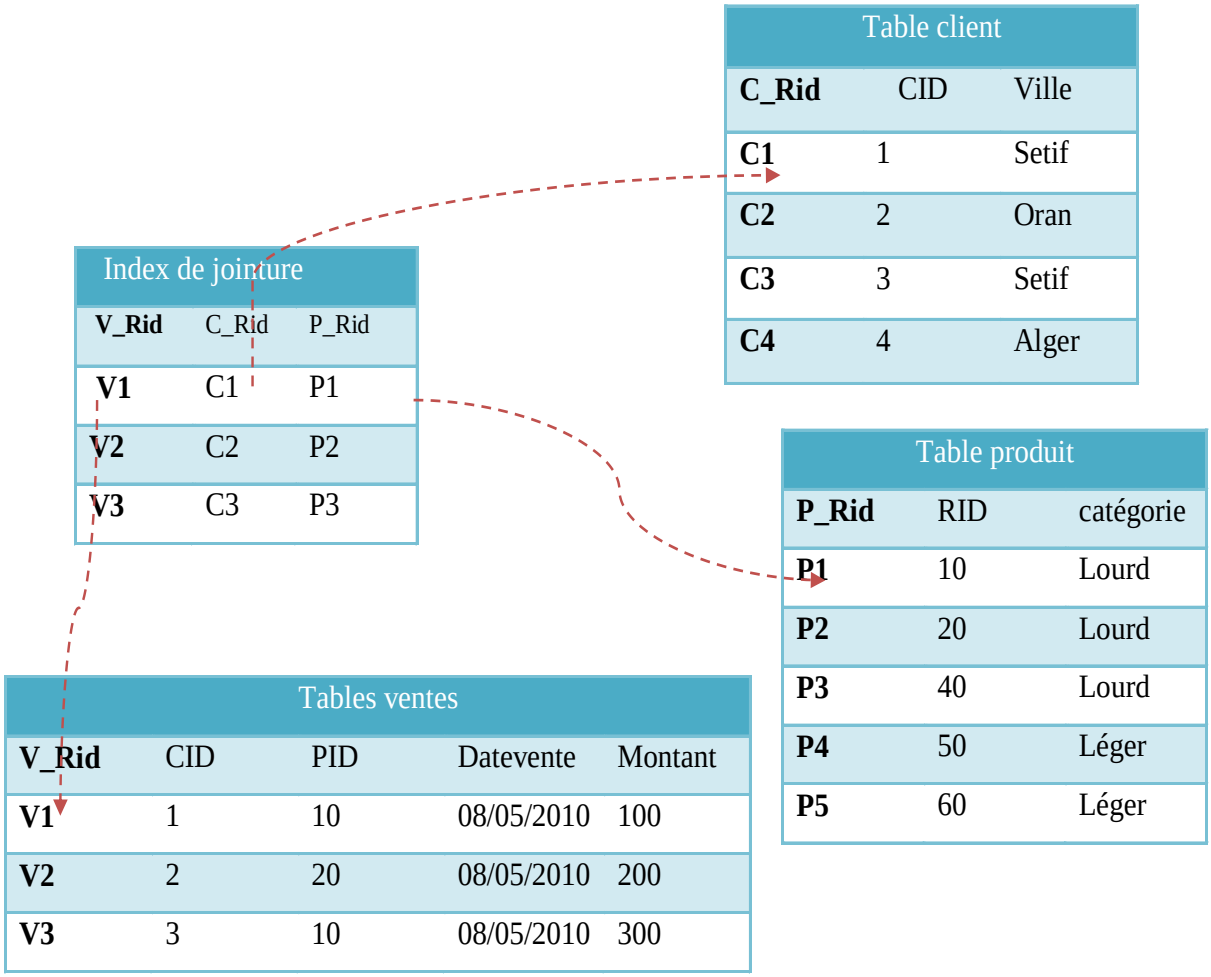

Table client

C_Rid	CID	Ville
C1	1	Setif
C2	2	Oran
C3	3	Setif
C4	4	Alger

Index de jointure

V_Rid	C_Rid	P_Rid
V1	C1	P1
V2	C2	P2
V3	C3	P3

Table produit

P_Rid	RID	catégorie
P1	10	Lourd
P2	20	Lourd
P3	40	Lourd
P4	50	Léger
P5	60	Léger

Tables ventes

V_Rid	CID	PID	Datevente	Montant
V1	1	10	08/05/2010	100
V2	2	20	08/05/2010	200
V3	3	10	08/05/2010	300

FIG. 3.6 Index de jointure

3.1.1.1.1.6 Index de jointure en étoile

Valduriez a travaillée sur les index de jointure dans le contexte des bases de données de type OLTP pour joindre les tables deux à deux. Mais dans le monde d'entrepôts de données, les requêtes OLAP peut comportent une ou plusieurs jointures entre la table des faits et les tables de dimension. Par exemple pour utiliser ces index dans un entrepôt modélisé par un schéma en étoile, il faut décomposer la requête en fonction des jointures. Effectuer cette décomposition selon le choix d'un ordre de jointure, or pour N jointures, N! ordres sont possibles (célèbre problème d'ordre de jointure). Pour pallier ce problème, RedBrick [33] ont proposé un index de jointure spécial, appelé Index de Jointure en Etoile (*star joinindex*)qui est plus approprié aux entrepôts modélisés par un schéma en étoile. Un (IJE) possède toutes

les transactions possibles qui faite d'une manière directe entre l'identifiant de la table des faits d'un coté et toutes les clés étrangères des tables de dimension de l'autre coté.

En cas ou toutes les tables de dimension sont jointer avec la table des faits c'est ce que nous appelons un IJE complet. et s'il est construit par la jointure de certaines tables de dimension avec la table des faits il s'appelle IJE partiel. Donc l'intérêt de l'index complet c'est qu'il est bénéfique pour n'importe quelle requête, mais il nécessite aussi un coût de maintenance et de stockage important. On à aussi que cet genre d'index n'est pas destiné aux entrepôts modélisés par un schéma en flocon de neige, a cause de ces jointures supplémentaires qui sont appliquées entre les hiérarchies de dimension et qui ne sont pas précalculées par cet index.

3.1.1.1.1.7 Index de jointure binaire (bitmap join index)

L'index de jointure binaire (IJB) est une sorte des index de jointure. Il construit d'après un accouplement entre l'index de jointure et l'index binaire. Il a été développé pour précalculer les jointures possibles entre une ou plusieurs tables de dimension et la table des faits dans le cadre des entrepôts de données modélisés par un schéma en étoile [34,27].par contre aux index binaires classiques où les attributs indexés dépend d'une seule table, l'IJB est défini sur un ou plusieurs attributs appartenant à plusieurs tables.

Plus précisément, il est défini sur la table des faits en utilisant des attributs appartenant à une ou plusieurs tables de dimension. Supposons un attribut A ayant *n* valeurs distinctes$v_1, v_2, \ldots\ldots, v_n$.appartenant à une table de dimension *D*. Si on Suppose que la table des faits *F* est subdivisée de m instances. La création de ce genre d'index ou *IJB* qui lui défini sur l'attribut A se fait par les étapes ci dessous :

- Créer *n* vecteurs ou chaque vecteur composé de m entrées.
- Le i^{ime}bit du vecteur correspondant à une valeur v_k est mis à 1 si le tuple de rang i de la table des faits est joint avec un tuple de la table de dimension D tel que la valeur de A de Ce tuple est égale à v_k. Il est mis à 0 dans le cas contraire. Plus formellement :

$IJB_j^k = 1$ si et seulement si $\exists\, td \in D$ tel que $tf_j.D_{id} = td\,.D_{id} \wedge td.A = v_k$où $IJB_j^k, tf_i, td, D_{id}$ représentent respectivement le $j^{éme}$ bit du vecteur correspondant à la valeurv_k, le n-uplet de F de rang *j*,unn-uplet de la table D, la clé étrangère de D.

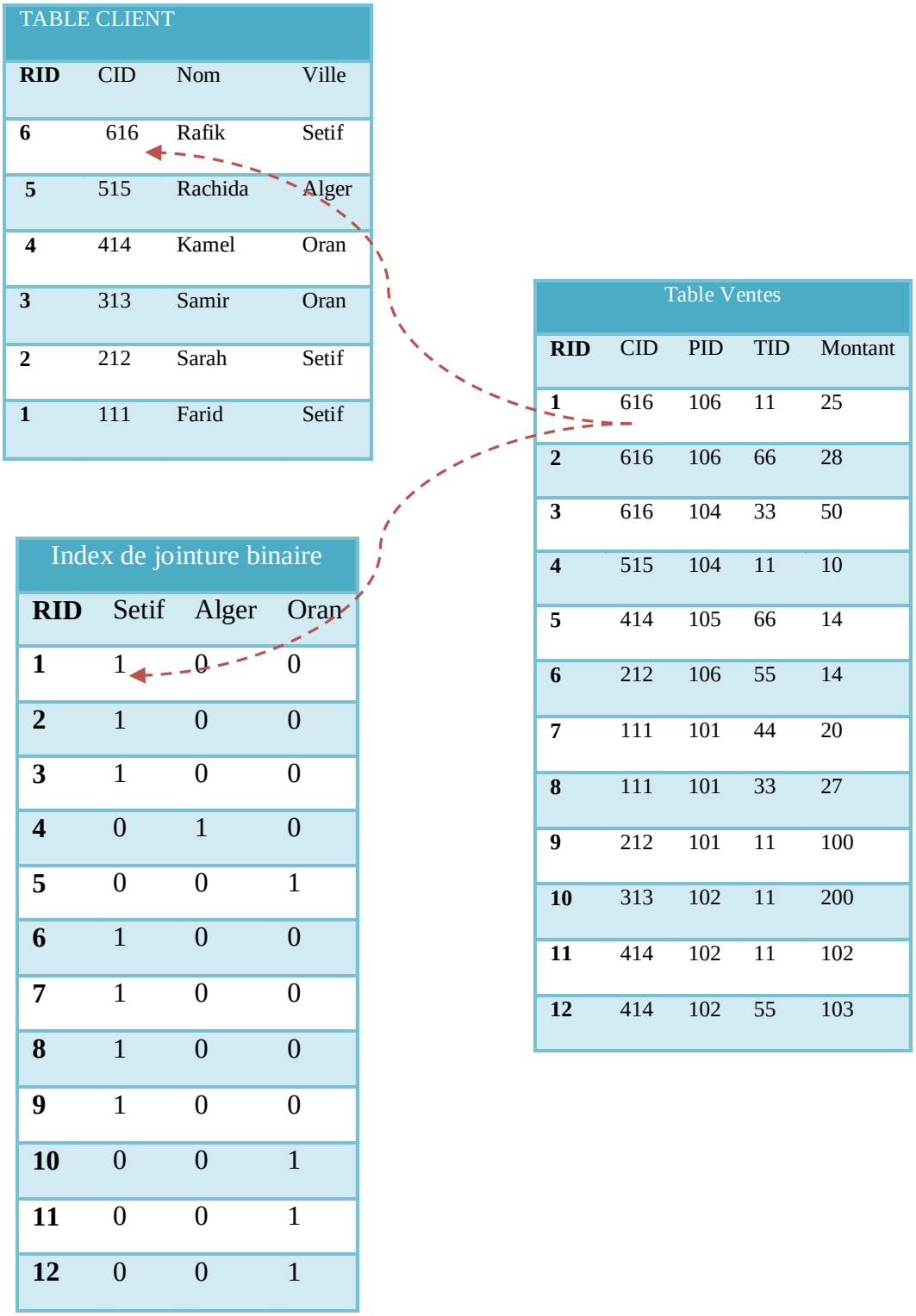

TABLE CLIENT

RID	CID	Nom	Ville
6	616	Rafik	Setif
5	515	Rachida	Alger
4	414	Kamel	Oran
3	313	Samir	Oran
2	212	Sarah	Setif
1	111	Farid	Setif

Table Ventes

RID	CID	PID	TID	Montant
1	616	106	11	25
2	616	106	66	28
3	616	104	33	50
4	515	104	11	10
5	414	105	66	14
6	212	106	55	14
7	111	101	44	20
8	111	101	33	27
9	212	101	11	100
10	313	102	11	200
11	414	102	11	102
12	414	102	55	103

Index de jointure binaire

RID	Setif	Alger	Oran
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	0	1
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	0	0	1
11	0	0	1
12	0	0	1

FIG. 3.7 Index de jointure binaire

3.1.1.1.1.8 Index de jointure de dimension

Les IJB ne sont pas exploitées dans le cas ou les entrepôts de données sont modélisés par un schéma en flocon de neige, parce que les transactions existantes entre les structures descendantes de table de la dimension ne sont pas précalculées. le chercheur Bizarro [36] a conçu un index qui peu considéré comme une extension de ce genre d'index appelé index de jointure de dimension son rôle c'est l'empois parfait des requêtes définies sur un schéma en flocon de neige. Donc ce nouveau genre d'index est un index binaire permettant de rapprocher les tables de dimension de la table des faits en précalculant des jointures intermédiaires. Cet index basé sur le même principe que l'IJB. Mais le calcule et le remplis des vecteurs de cet index selon toutes les combinaisons nécessaires pour joindre les tables de dimension indexées quel que soit leur niveau de hiérarchie avec la table globale nommé table des faits.

3.1.1.1.2 Problème de sélection des index

L'objectif principale pour le bonne fonctionnement dans les entrepôts des données c'est la réduction de temps d'exécution des requêtes, donc l'administrateur doit sélectionner une charge d'index appelé configuration sous une contrainte prédéterminé qui peut représenter soit l'espace de stockage ou le temps de maintenance ou n'importe quelle autre facteurs.

Le problème de sélection de ces index est devenu un enjeu très important dans l'étape de conception physique des bases et entrepôts de données. Dans le monde d'entrepôts, la sélection d'index est prouve son importance a cause d'augmentation et d'expansion de nombre d'attributs et de tables à indexer. En plus, les index peuvent être définis sur plusieurs attributs issus de plusieurs tables.

Le problème de sélection d'une configuration d'une charge d'index peut être formalisé comme suit :

Étant donné :

- un entrepôt de données disposé d'un ensemble de tables de dimension $D = \{D_1, D_2, \ldots\ldots, D_d\}$et une table des faits F.
- Il comporte aussi d'une charge de requêtes$Q = \{Q_1, Q_2, \ldots\ldots, Qm\}$, où chaque requête $Q_j$possède une fréquence d'accès f_j.
- Un ensemble de contraintes comme les coûts maximums de stockage et de maintenance autorisés.

Le problème de sélection des index consiste à trouver une configuration *CI* tel que :

- Le coût d'exécution de *Q* en présence de *CI* est minimal.
- La configuration CI respecte les contraintes du problème.

Le problème de sélection d'index est avéré comme un problème NP-Complet [5]. Et alors, il n'existe pas d'algorithme Spécifiques suggérant une solution optimale en un temps raisonnable ou fini.

Plusieurs travaux de recherches proposent des solutions proches de la solution optimale en utilisant des heuristiques réduisant la complexité du problème.

Les algorithmes proposés dans le cadre de sélection d'index possèdent généralement deux étapes comme montré à la figure (**FIG.3.8**) :

- La détermination des attributs candidats à l'indexation.
- La construction d'une configuration d'index.

L'idée de sélection des attributs indexables peut effectuer de deux façons différentes: manuelle ou automatique. L'approche manuelle est réalisé par l'administrateur lui même qui fait le choix d'un certain nombre d'attributs candidats pour l'indexation [37, 38, 39].Cette approche dépend entièrement de l'expérience de l'administrateur. En revanche la sélection automatique basé sur le principe d'analyse syntaxique des requêtes prises en compte pour la sélection des index [40, 41, 42]. L'objectif de cette étape est de réduire l'espace de recherche des index. Notons que la taille de l'espace de recherche est exponentielle par rapport au nombre d'attributs candidats. Plusieurs approches ont été proposées pour la réduction et l'élagage de l'espace de recherche des index [40, 41, 43, 44, 4].

La phase de construction d'index consiste à déterminer un ensemble d'index à partir des attributs indexables candidats ainsi qu'un ensemble des requêtes définies sur la base de données.

On a deux genre de démarches de construction d'index sont placé d'une manière contradictoire: démarches ascendantes et démarches descendantes. Les démarches ascendantes débute par une configuration à base vide et enrichissent cette configuration a chaque application pendant plusieurs itérations dans L'objectif de réduire le coût d'exécution des requêtes chaque itération, par conséquent, le processus s'arrête lorsqu'aucune réduction

n'est possible en terminant tous les ajouts possibles [37, 39, 40]. Les démarches descendantes débutent par une configuration globale ou complète. Cette configuration possède tous les index candidats.et puis on commence à éliminés par un ou plusieurs index chaque itération jusqu'à réduction total des coûts de stockage et d'exécution. Le processus s'arrête lorsqu'aucune amélioration du coût n'est possible par n'importe quelle élimination d'index.
La majorité des travaux sur les approches de sélection d'index concerne des index sur une seule table. Peu de travaux ont destiné vers le problème de sélection de configuration d'index multi-tables [21, 4]. Les chercheurs dans le monde du SGBD proposent quelque outils de sélection une configuration d'index comme l'outil AutoAdmin [46] qui a été développé dans le cadre de l'auto-administration des bases de données.

Nous étalons dans ce qui suit quelques travaux sur la sélection d'index dans les bases et entrepôts de données.

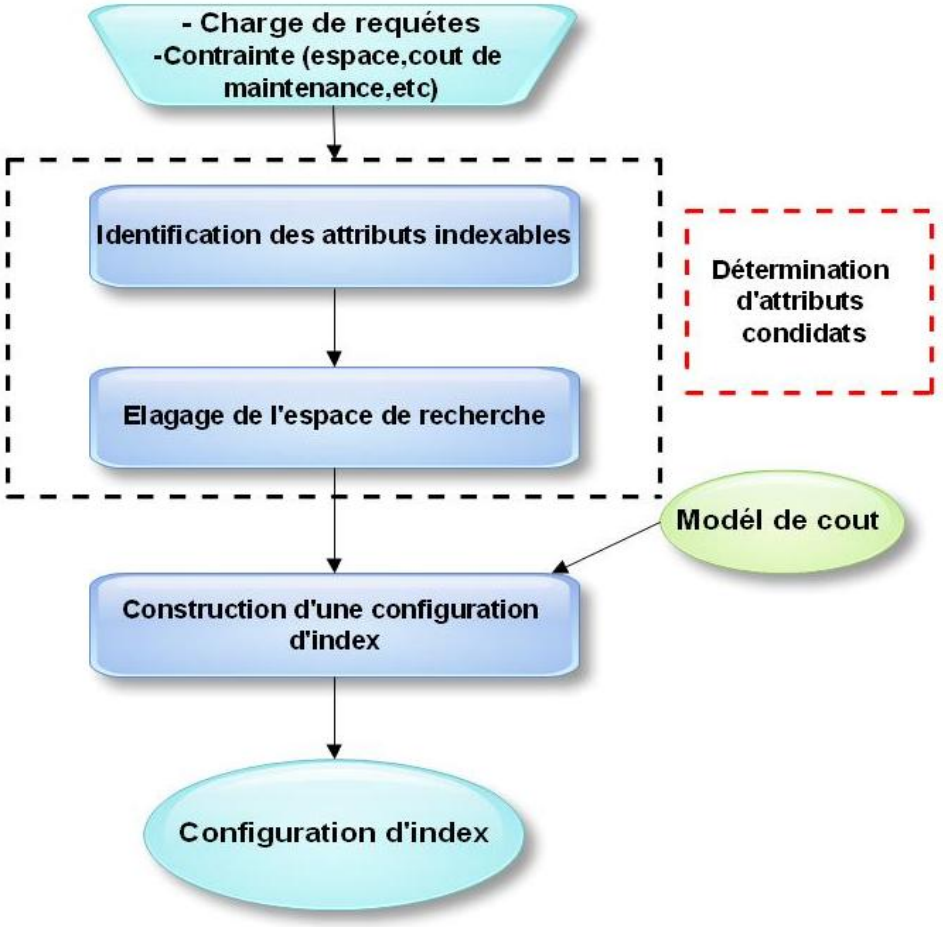

FIG. 3.8 Approche générique de sélection d'index

3.1.1.1.2.1 Travaux de Chaudhuri

Le fameux chercheur Chaudhuri et son groupe de base de données ont développé l'un des premier outil Commercial de sélection d'index IST (Index SelectionTool) sous Microsoft SQL Server 7.0. Ces travaux sont entrés dans le projet AutoAdmin. Ce dernier qui a été lancé par Microsoft pour trouver de nouvelles techniques capable d'auto-administrer une base de données, tout en garantissant des performances similaire à celles d'une base de données gérée seulement par un administrateur humain.

Le but d'outil IST fournie dans la fin de ce projet est de sélectionner une configuration d'index mono et multi-attributs à partir d'une charge de requêtes via Certain contrainte, dans cette cas l'espace de stockage. Les principaux phases de sélection sont illustrés dans la figure 3.9.Pour évaluer la qualité des configurations d'index trouvés, le paradigme utilise deux modules : le rôle d'optimiseur est d'estimer le coût d'exécution pour les index candidats existants et le module What-if pour estimer le coût des index candidats non existants, en simulant leur présence [46]. La sélection de configuration d'index passe par trois principales phases :

- La sélection des index candidats.
- La sélection des configurations.
- La génération des index multi-attributs.

1. **Sélection des index candidats** : Cette phase consiste à énumérer l'ensemble des attributs indexables. Puis ils sont choisis parmi ceux présents dans les clauses WHERE.,GROUP BY et ORDER BY. L'intérêt de cette étape est d'atteindre à une meilleure configuration pour chaque requête d'une manière indépendante. Les configurations candidates sont évaluées en utilisant le précédent *what-if*. La configuration générant le minimum de coût, sera choisie comme la meilleure configuration pour la requête.

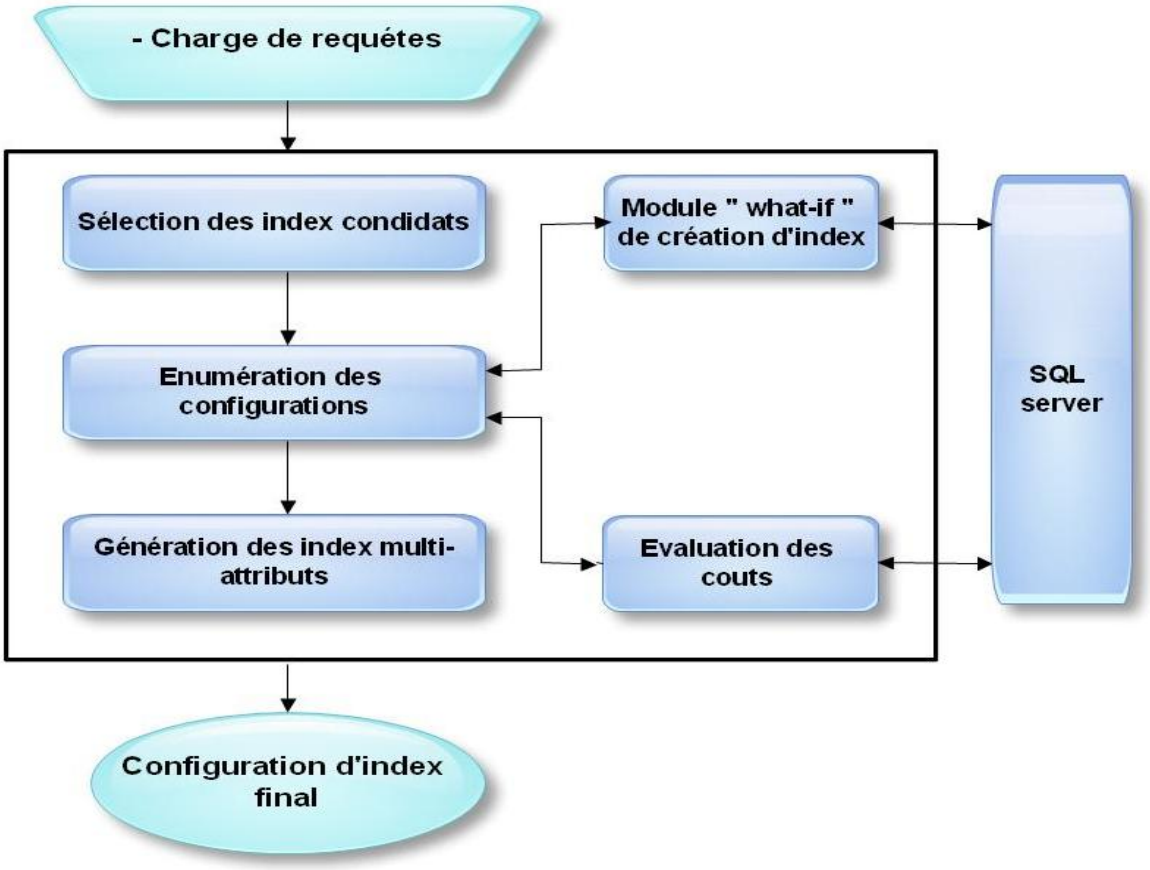

FIG. 3.9 Architecture de l'outil de sélection d'index IST

2. **Génération des configurations** : cette deuxième phase consiste à éliminer les index non utiles qui sont apporter moins de gain en coût. Pour réaliser cette tâche, un algorithme glouton est appliqué. Il permet d'une certaines élection des k meilleurs index parmi les n index candidats. Le composant correspondant à cette phase interroge l'optimiseur pour évaluer la qualité des index. Cette sélection des configurations d'index se fait en trois étapes comme suit:

- Trouver premièrement la meilleure configuration de taille m selon quelque critère prédéterminés (m très petit par rapport à k). Avec l'aide d'un algorithme exhaustif qui guide vers cette configuration.
- Commence à ajouter l'index qui minimise le coût d'exécution à la configuration actuelle.
- Répéter l'étape 2 tant qu'il y a des index à ajouter réduisant le coût de requêtes.

3. **Génération des index multi-attributs** : La construction des index multi-attributs repose sur l'ensemble des index mono-attributs sélectionnés dans l'étape précédente.

Cette génération basée sur l'utilisation de deux fonctions : MC-LEAD et MC-ALL. Les configurations candidates sont évaluées en utilisant le précédent *what-if*. La configuration avec le coût minimal, sera considérée comme la meilleure configuration pour la requête.

Dans le cadre de minimiser les charge d'emplois des entrepôts de données comme le coût de stockage et de maintenance, Chaudhuri et son groupe. Ont développé une technique sous le nom fusion d'index (index merging). Elle consiste à prendre un ensemble d'index ayant une capacité d'espace S et testé s'il existe un ensemble d'index ayant une capacité d'espace S inférieure à celle de départ (S' < S). La tache de fusion est guidée par un modèle de coût, la fusion est appliquée s'il y a une réduction dans le coût d'exécution des requêtes.

3.1.1.1.2.2 Travaux de Frank

Le chercheur Frank [37] travaille dans le cadre de développer un outil d'aide pour le choix d'index dans une base de données. La sélection des index basé sur des échanges directs entre l'outil développé et l'optimiseur de requêtes. Cette techniques repose sur le calcule de gain de la performance qu'apporte l'utilisation d'un ou plusieurs index par rapport au performance des requêtes. Le gain est calculé par la différence entre le coût fourni avant et après la construction des index. Pour déterminer ce coût, la technique basé sur le modèle de coût utilisé par l'optimiseur de requêtes.

La sélection d'index s'appliqué au niveau des requêtes et elle se réalisé une par une avec plusieurs itérations. Dans chaque itération, l'optimiseur présente à l'outil développé le meilleur index parmi un ensemble d'index candidats avec le coût de la charge de requêtes en matérialisant cet index. Cette opération d'échange entre l'outil et l'optimiseur s'arrête lorsqu'il n'y a plus d'index à proposer.

Pour un ensemble d'index à créer $I = \{I_1, I_2, \ldots, I_n\}$l'outil permet de sélectionner un sous ensemble $I' \subseteq I$ Pour cela, l'outil construit un schéma d'index représente tout les combinaisons d'index possibles. L'approche de sélection d'index est composée des étapes ci dessous :

- L'outil envoie la première requête à l'optimiseur ainsi qu'un ensemble d'index initial.
- Les index sélectionnés par l'optimiseur pour la requête courante sont sauvegardés ainsi que le gain qu'ils apportent à la requête.
- L'étape 2 est réitérée pour toutes les requêtes.
- Les gains apportés par chaque index sur l'ensemble des requêtes sont cumulés.

- L'outil recommande les index présentant un gain positif pour l'ensemble des requêtes.

3.1.1.1.2.3 Travaux de Whang

Whang [5,49] a proposé deux approche repose sur deux algorithmes de sélection d'index, le premier a base descendante DROP et le deuxième a base ascendante ADD.

L'approche descendante commence par un état initial contenant tous les index possibles.et Durant chaque itération, l'index engendrant la plus grande décroissance du coût d'exécution de la charge de requête est détecté puis éliminé. Quand l'élimination d'un seul index ne permet pas d'atteindre a un niveau satisfaisant de réduction du coût d'exécution des requêtes, l'algorithme DROP continuer à éliminé les index mais cette fois deux index à la fois, ensuite trois index, et ainsi de suite.

L'approche ascendante c'est le cas opposé de la premier approche puisque elle repose sur un état initial vide où aucun index n'est encore sélectionné.et puis Durant chaque itération de l'algorithme ADD, on teste chaque index et en détecte chacun réduisant le coût d'exécution des requêtes est ajouté à l'état courant. Cette opération s'arrête lorsque tous les index sont créés ou aucune réduction de coût n'est possible.

3.1.1.1.2.4 Travaux de Gundem

Le chercheur Gundem [50] à travaillé sur le problème de choix du type d'index construit sur un attribut donné. Ce problème a apparue puisque on a que chaque attribut peut être indexé en exploitant plusieurs techniques d'indexation.et chaque technique apporte un gain en temps d'exécution et occupe un espace de stockage différent.

L'approche de Gundem propose la subdivision de l'ensemble des index possibles d'une certaine table en plusieurs sous-ensembles appelé classe d'équivalence représenter les index définis sur le même attribut.

Les travaux de Gundem supposent que l'administrateur doit pouvoir fournir les informations suivantes :

- L'ensemble des index candidats concernant par le choix.
- L'espace de stockage disponible pour stocker les adresses physiques des index.
- Le choix des index multiples sur chaque attribut.
- Les fréquences d'accès des requêtes de sélection et de mise à jour.

- Un seuil du taux d'erreur toléré.

On utilisant ces informations pour que l'approche de Gundem commence à choisir et trier un sous-ensemble d'index qui minimise, via le taux d'erreur toléré, le coût d'exécution des requêtes avec le contrôle et sans dépasser la capacité de l'espace de stockage. Deux principales phases caractérisent cette approche, l'optimisation locale et l'optimisation globale.

L'optimisation locale consiste à sélectionner un ensemble d'index selon la classe d'équivalence. Donc l'ensemble d'index disjoints est donc sélectionné et on l'appelé disjoints parce que au plus un index est créé sur chaque attribut. Pour chaque attribut, tous les index candidats sont évalués par une fonction de coût. Cette fonction estime le gain apporté par la matérialisation des index. Ce dernier qui calcule par la différence entre le gain de coût en matérialisant l'index et le coût de création des index. Apres la vérification de cout des index, donc on commence à éliminé automatiquement les index qui sont engendrant un coût de création supérieur au gain apporté. Le résultat de cette étape est un ensemble d'index I où chaque élément de cet ensemble est un index défini sur une seule classe d'équivalence.

La deuxième approche rassemble a l'optimisation globale qui permet de minimiser le coût d'exécution total en pris en compte que tous les index dans l'ensemble I construit dans l'étape précédente. La fonction objectif utilisée représente le gain en coût d'exécution avant et après l'opération d'engendre un ensemble d'index. Le but à atteint est de trouver le sous-ensemble d'index dans I respectant les contraintes suivantes :

- Le gain apporté par ce sous-ensemble d'index est maximum.
- La taille de ce sous-ensemble d'index ne dépasse pas l'espace de stockage réservé aux index.

3.1.1.1.2.5 Travaux de Golfarelli

L'approche de Golfarelli [42] a permet de sélectionner un ensemble d'index optimal parmi un schéma logique d'un entrepôt de données comprenant des vues matérialisées, une charge, des statistiques et une contrainte sur l'espace disque dédié aux index. L'objective est de déterminer le schéma physique optimal ainsi qu'un jeu d'index son rôle c'est la minimisation de temps d'exécution des requêtes tout en vérifiant la condition de disponibilité d'espace de stockage. Golfarelli à défini un modèle d'optimiseur qui engendre un schéma d'exécution pour chaque requête et un modèle de coût pour comparer les différentes solutions. Le modèle d'optimiseur développé par Golfarelli et son groupe de recherche est basé sur des

règles RuleBasedOptimizer (RBO). Le RBO a été sélectionné d'une maniéré à obtenir toujours le même résultat pour deux requêtes dont la seule différence porte sur des valeurs de prédicats.

Les index concernés par les travaux de ce groupe de recherche sont de deux types, liste de valeurs et bitmap. Les requêtes montre les agrégations sur des jointures en étoile entre une table des faits et quelque tables de dimensions ; les sélections pouvant être formulées sur les attributs des tables de dimensions. On suppose aussi qu'un ensemble d'index candidats utiles est prédéterminé, compte tenu de la charge. Puis un algorithme glouton choisit peu à peu,on utilisant les index candidats, les index les plus utile tant que la contrainte d'espace est satisfaite.

La figure (**FIG.3.10**) illustre l'architecture fonctionnelle de l'approche proposée par les auteurs.

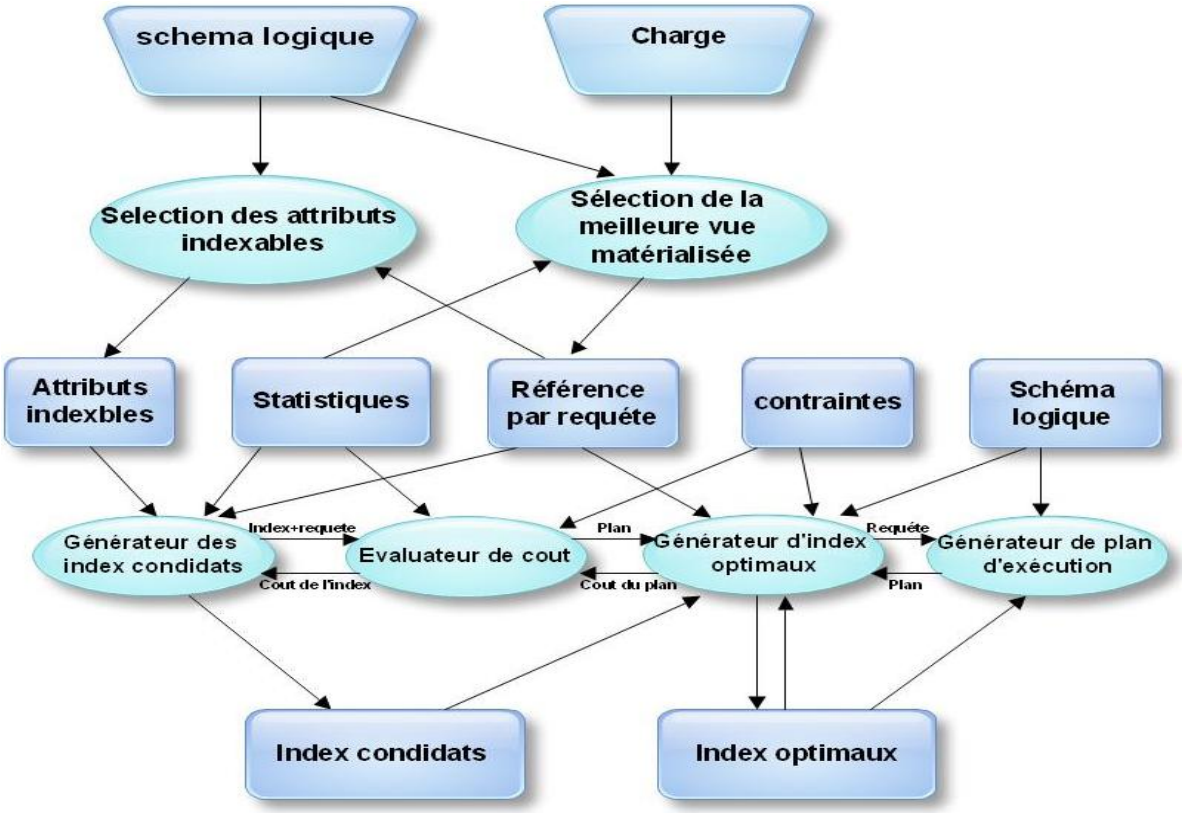

FIG. 3.10 Architecture de fonctionnement de l'approche de Golfareli

L'approche exige d'utiliser les informations suivantes comme entrées :

- **Le schéma logique :** il représente la structure des tables de l'entrepôt et leurs éventuelles relations.
- **La charge :** elle décrit l'ensemble des requêtes à exécuter sur notre entrepôt. Chaque requête est caractérisée par sa fréquence d'accès.
- **Les statistiques :** elles décrivent toutes natures d'informations quantitatives requis pour l'optimiseur pour estimer le coût d'exécution des requêtes, comme la cardinalité des tables, le nombre de valeurs distinctes de chaque attribut, etc.
- **Les contraintes :** la sélection des index nécessite le connaitre des limites imposées par le matériel, comme l'espace disque réservé pour les index.

Plusieurs modules sont implémentés par l'approche. Chaque module est responsable d'une fonction particulière.

- **Le module de sélection des attributs indexables :** ce moyen se base sur la structure des requêtes pour déterminer les attributs des tables de dimensions pouvant être utilement indexés.
- **Le générateur des index candidats :** ce module permet d'évaluer le type d'index le plus adapté. Les index candidats sélectionnés par ce module sont définis par couple (attribut, type d'index).
- **Le générateur de l'ensemble d'index optimal :** ce module exécute l'algorithme qui sélectionne les index devant être créés.
- **L'évaluateur de coût :** la nécessité de ce composant pour garantir le générateur des index candidats et en simultanément pour le générateur de l'ensemble d'index optimal afin d'estimer le coût de chaque index.
- **Le générateur de plan d'exécution :** une requête exécutable sur l'entrepôt de données peut implémenter avec plusieurs plans d'exécution selon l'ordre des jointures utilisées ainsi que les structures physiques implémentées. La tache principale de ce composant consiste à sélectionner le meilleur plan d'exécution d'une requête parmi les plans d'exécution disponibles.

Chaque module ou composant dans cette approche fournit des résultats qui peuvent être considéré comme finaux (telle que les index finaux sélectionnés) ou intermédiaires utilisés comme entrées pour d'autres composants (comme les index candidats). Nous pouvons citer parmi ces résultats les éléments ci dessous:

- **Les attributs indexables :** l'ensemble des index pouvant être utilement indexés pour faciliter l'exécution et rendre plus souple le traitement de quelques requêtes.
- **Les index candidats :** représente l'ensemble d'index générant sur les attributs indexables.
- **Les index optimaux :** représente les index générant a la fin de cette approche sur l'entrepôt de données.

3.1.1.1.2.6 Travaux de Aouiche

Les travaux d'Aouiche .[4] sont destiné vers le problème de sélection des IJB dans le contexte d'entrepôts de données modélisés par un schéma en étoile et on remarque aussi que peu de chercheurs sont travaillé dans ce problème. La démarche développée se repose sur une technique de datamining (recherche des motifs fréquents) pour réduire l'espace de recherche des index de jointure.

Un algorithme glouton est exploité pour l'utiliser dans la sélection d'une configuration d'index. La figure (**FIG.3.11**) montre les principales étapes de l'approche proposée.

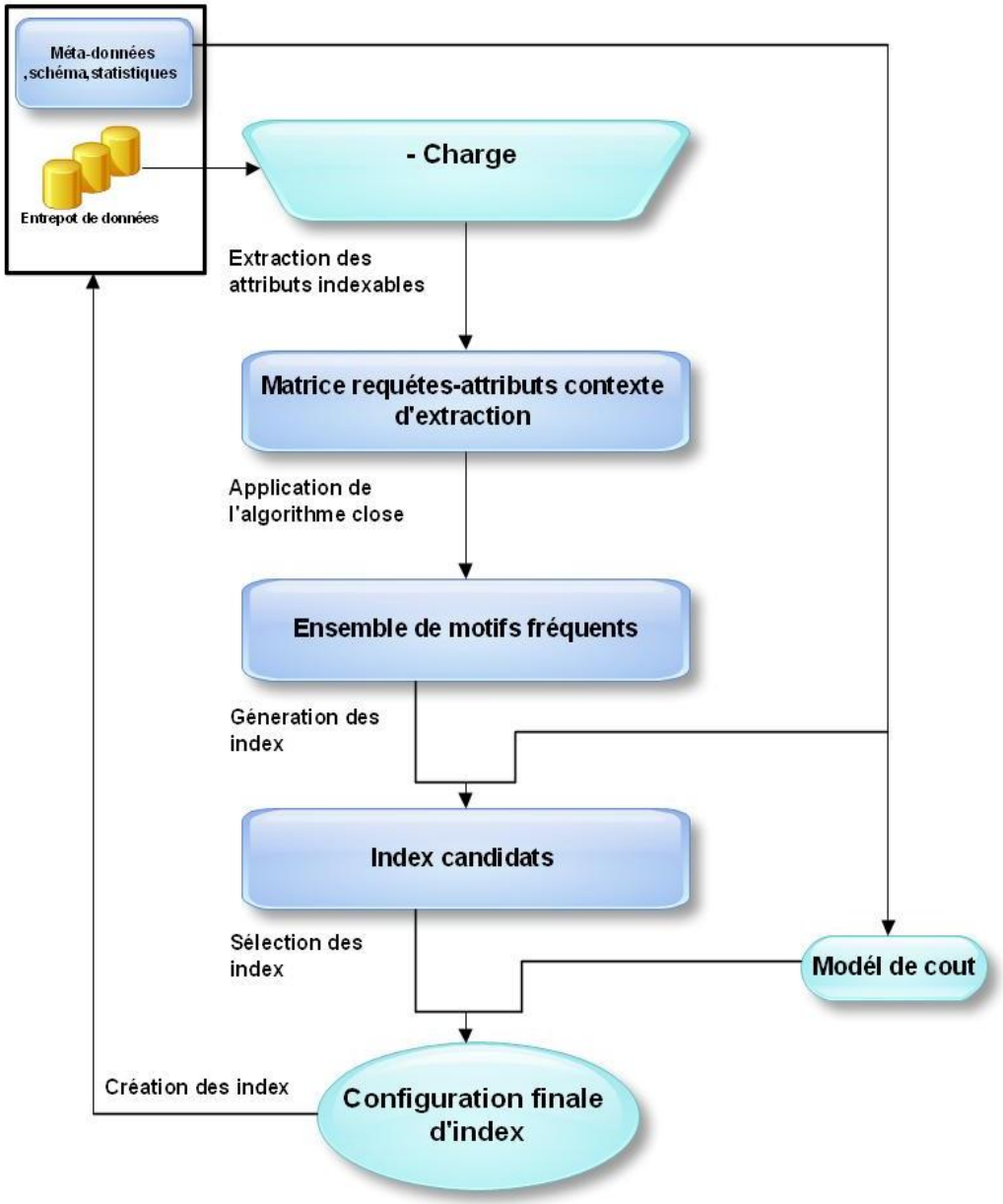

FIG. 3.11 Architecture de fonctionnement de l'approche d'Aouiche

3.1.1.1.2.7 Travaux de Bellatreche

Les travaux de Bellatreche.[21,51] se rentre dans le cadre de progrès les travaux de Aouiche[4]. Ces derniers considèrent en premier lieu les fréquences d'accès des attributs comme critère de génération des motifs fréquents fermés.

Le chercheur Bellatreche est preuve à travers son travaille que la fréquence d'accès n 'est pas le seul facteur rentrant dans la sélection d'un ensemble d'index efficace. En réalité, les IJB sont créés pour le but d'optimiser des jointures entre la table des faits et les tables de

dimension. A l'aide des travaux d'Aouiche., l'algorithme peut éliminer des index sur des attributs moins fréquemment utilisés mais qui dépendent principalement à des tables d'énorme dimension, ce qui ne permet pas d'optimiser une opération de jointure. Pour éviter ce problème, Bellatreche proposent d'intègre d'autres nouveaux paramètres dans la détection des motifs fréquents tell que la taille des tables de dimension, la taille de la page système, etc. Le chercheur developpe deux algorithmes *DynaClose* et *DynaCharm* qui constituent une adaptation aux travaux de Aouiche. Au lieu d'utiliser le support seulement comme critère de détermination des motifs fréquents, les algorithmes proposés basé sur une fonction fitness permettant de pénaliser chaque motif fréquent selon les paramètres suivants :
Pour un motif fréquent mi, cette fonction est définie comme suit :

$$Fitness(m_i) = \frac{1}{n} \times (\sum_{j=1}^{n} \alpha_J \times sup_j)$$

Où n c'est le nombre d'attributs non clés A_j dansm_i. Sup_jIndique le support de $A_j$et $\alpha_j$est un paramètre de pénalisation défini par l'équation suivante : $\alpha_j = \frac{|D_j|}{|F|}$où $|\left|D_J\right|, |F|$ représentent respectivement le nombre de pages nécessaires pour stocker la table de dimension D_j et la table des faits F.

Nous avons un support minimum minsup, une valeur minimum de la fonction fitness *minfitness* est calculée comme suit : $minfit = \frac{minsup}{|F|} \times \left\lceil \left(\sum_{j=1}^{d} \frac{|D_j|}{d}\right) \right\rceil$où d représente le nombre de tables de dimension.

Les deux algorithmes proposés dans les travaux de belatreche*DynaClose* et *DynaCharm* sont appliqués dans l'étape de réduire et d'élagage de l'espace de recherche des index. Quand les motifs fréquents commence à générés, une deuxième étape appelé phase de purification travaillé à éliminer les motifs qui ne générer aucun index de jointure. tell que, un motif fréquent ne contenant aucun attribut non clé des tables de dimension sera supprimé inévitablement. L'opération de purification permet de générer un ensemble d'attributs indexables candidats. Cet ensemble est défini par l'union des attributs non clés appartenant aux motifs fréquents générés. On utilisant l'ensemble d'attributs candidats, un algorithme glouton proposé par ce chercheur consiste à sélectionné une configuration d'index finale selon une certain contrainte rassemble a l'espace de stockage. L'algorithme glouton commence par l'index défini sur l'attribut ayant la cardinalité minimum, ajouter ensuite d'autres index

itérativement jusqu'à ce que l'espace de stockage soit consommé ou tous les index sélectionnés.

3.1.1.1.3 Bilan et discussion

Nous avons montré dans cette précédente partie les principaux travaux sur le problème de sélection des index. La majorité des approches développés et proposées commencent premièrement par la reconnaissance et la fixation des attributs indexables, selon ces deux formes manuelle et automatique, puis elles utilisent des algorithmes de sélection (algorithme glouton ou dirigé par des techniques de data mining) afin de construire ou générer la configuration d'index finale. La qualité de cette configuration est mesurée soit par un modèle de coût mathématique, soit par le modèle de l'optimiseur du SGBD. Le tableau (**TAB 3.2**) montre une comparaison entre les principaux travaux en se repose sur quelque critères que nous venons de citer.

Travaux	Sélection des attributs candidats	Algorithme de sélection	Modèle de coût
Chaudhuri. [46]	Automatique	Glouton	Optimiseur + module what-if
Frank. [45]	Manuelle	Glouton	Optimiseur
Wang. [49]	Manuelle	Glouton	Mathématique
Gundem. [50]	Manuelle	Glouton	Mathématique
Golfarelli .[42]	Automatique	Glouton	Optimiseur
Aouiche . [4]	Automatique	Data Mining + Glouton	Mathématique
Bellatreche . [21]	Manuelle	Data Mining + Glouton	Mathématique

TAB 3.1 Comparaison des travaux effectués sur la sélection d'index.

3.1.1.2 Vues matérialisées

Si on essaie de mettre une illustration de ce concept les vues matérialisées, en effet une vue est une sorte de requête. Elle appelé matérialisée parce que son résultat est stockée d'une manière physique. Ces vues aidé a l'amélioration l'exécution des requêtes, et elle le rendre

exécuté dans un délai optimale, en précalculant les opérations les plus coûteuses comme la jointure et l'agrégation et en lance une opération de stockage de ces résultats dans la base. Et effet, quelques requêtes nécessitent seulement l'accès aux vues matérialisées et sont ainsi exécutées plus rapidement. Les vues dans le cadre d'OLTP ont été largement utilisées pour répondre à plusieurs rôles : la sécurité, la confidentialité, l'intégrité référentielle, etc.

Les vues matérialisées peuvent être utilisées pour Atteint plusieurs objectifs, comme l'augmentation et l'optimisation de la performance des requêtes ou la fourniture des données dupliquées. Cette idée est très utilisée dans l'informatique distribuée. En réalité elles sont utilisées pour dupliquer des données au niveau des sites distribués. Les réplicas permettent de résoudre des requêtes uniquement par des accès locaux.

Et comme toutes sortes de nouvelles technologies, plusieurs problèmes sont apparus au lancement des vues matérialisées principalement deux :

- Le problème de sélection des vues matérialisées.
- Le problème de maintenance des vues matérialisées.

Nous présentons brièvement ces deux problèmes dans les sections suivantes

3.1.1.2.1 La sélection des vues matérialisées

La sélection des vues matérialisées permet de choisir un sous-ensemble de vues candidates capable de réduire le coût d'exécution d'une charge de requêtes. La sélection des vues peut être réalisée sous certaines contraintes, généralement un quota d'espace et/ou un seuil de temps de maintenance à ne pas dépasser. Le problème de sélection des vues matérialisées (PSV) peut donc être formalisé comme suit [52, 53] :
On a S une contrainte de ressource (capacité de stockage, par exemple), le PSV permet de sélectionner un ensemble de vues minimisant une fonction appelé fonction objectif (coût total d'évaluation des requêtes et/ou coût de maintenance des vues sélectionnées) et satisfaisant la contrainte.

Le concept PSV dans les entrepôts de données a été exploité avec l'approche MOLAP plus qu'avec l'approche ROLAP. Dans la premier approche, on considéré le cube de données comme une structure primordiale pour déterminer les vues matérialisées. Ou chaque cellule du cube est considérée comme une vue potentielle. Avec la deuxième approche, chaque

requête est montrée par un arbre algébrique [55]. Chaque nœud (non feuille) est connu comme une vue potentielle.

Les chercheurs dans les vues matérialisés on les subdivise on deux genres : le PSV statique et le PSV dynamique. Le PSV statique permet de sélectionner un ensemble de vues à matérialiser afin de minimiser le coût total d'évaluation de ces requêtes, le coût de maintenance ou les deux, et ce, selon la contrainte de la ressource. Le problème considère donc que l'ensemble des requêtes ne sont pas évolutive .Si des évolutions sont faite dans des requêtes, donc il est inévitable de reconsidérer de nouveau le problème (en reconstruisant les vues à matérialiser). Le PSV dynamique suppose que l'ensemble des requêtes sont évolutive par rapport au temps. Dans ce cas, l'algorithme dynamique peut être amené à modifier l'ensemble des vues matérialisées en fonction de nouvelles requêtes. En réalité, il peut intégré de nouvelles vues et en supprimer d'autres dans le cas où l'espace autorisé atteint son maximum. Pour combler les lacunes du PSV statique, le chercheur Kotidis et son groupe. [56] ont développé un système sous le nom DynaMat, qui travaille à matérialiser les vues d'une façon dynamique. DynaMat combine en fait les problèmes de la sélection et de la maintenance des vues.

Le PSV est formulé comme un problème mathématique, en effet le PSV est un problème NP-Complet [57, 3]. Le nombre de vues candidates pour la tache de matérialisation peut atteint un nombre très important. Si d représente le nombre de dimensions dans le schéma d'un entrepôt de données et qui ne contient aucune hiérarchie, alors ce nombre est égal à 2^d. La complexité du PSV est de $O(2^n)$où n indique le nombre de vues candidates dans le schéma [58,3].

Plusieurs travaux ont étudié le problème de sélection des vues matérialisées. Ces travaux peuvent être classés en deux grandes classes :

- Travaux orientés par le temps d'interrogation.
- Travaux orientés par le temps de maintenance.

Le premier genre vise à optimisé les performances de l'interrogation de l'entrepôt et fonctionne sous contrainte d'espace de stockage des vues matérialisées. La seconde privilégie la réduction du temps de maintenance des vues matérialisées [52]. Elle est exploitée dans le cas où les mises à jour de l'entrepôt seraient conséquentes ou leur fréquence élevée. Quelques approches ont été proposées dans le souci de satisfaire simultanément les deux besoins [59].

Les travaux proposés pour la sélection des vues matérialisées utilisent souvent des heuristiques pour trouver une solution quasi optimale.

Les algorithmes proposés procèdent en deux principales phases : (1) génération des vues candidates et (2) sélection un sous-ensemble de ces vues. Dans la première phase, l'ensemble de vues matérialisées candidates V est construit à partir d'une charge de requêtes Q les plus fréquentes.

Cet ensemble est structuré de manière à prendre en considération les relations susceptibles d'exister entre les vues candidates. Plusieurs structures ont été proposées aux travaux pour représenter les relations entre les vues. On peut citer les structures suivantes : les treillis [61, 62, 84, 60, 90], les graphes [69, 74, 80, 65], les plans d'exécution des requêtes [63, 72, 78], etc.

Dans la deuxième phase, les algorithmes sélectionnent un sous-ensemble des vues candidates en fonction de la fonction objectif utilisée ainsi que des contraintes du problème de sélection.

Plusieurs types d'algorithmes ont été proposés pour traiter cette sélection. On peut citer Les algorithmes gloutons [61, 62, 84, 87, 60, 64, 72, 45], les méthodes issues de la recherche opérationnelle (sac à dos [47]) ou d'un algorithme génétique [63].

3.1.1.2.2 La maintenance des vues matérialisées

Les tables de base changent et évoluent à chaque rafraichissement. Alors, si ces évolutions et modifications ne sont pas reportées dans les vues matérialisées, parce que leurs contenus deviendront dépassés et leurs objets ne simuleront plus la réalité. La tache de maintenance des vues matérialisées permet d'amener les changements survenus sur les tables de base au niveau des vues. Cela peut se faire selon trois démarches: périodique, immédiate et différée. Dans la première démarche, les vues sont mises à jour continûment et périodiquement à des moments précises ; dans cette démarche, ces vues peuvent être considérées comme des photographies (snapshots).

Dans la deuxième démarche la tache de rafraichissement des vues est immédiatement terminée, exactement à la fin de chaque transaction. Dans la dernière démarche, les

modifications sont propagées d'une manière différée. Dans ce cas, une vue est mise à jour juste au moment où elle est utilisée par une requête d'un utilisateur.

La tache de maintenance des vues peut être réalisée en recalculant ces vues sur les tables de base, en effet, cette démarche est totalement inefficace (très coûteuse). Parce que, une bonne maintenance des vues est effectuée lorsque les modifications (insertions, suppressions, modifications) réalisées dans les tables sources peuvent être propagées aux vues sans être dans l'obligation de recalculer entièrement leur contenu. Pour traiter ce problème, trois types de maintenance ont été proposées : incrémentale, autonome et en batch.

Le premier type permet d'identifier le nouvel ensemble de n-uplets à ajouter à la vue dans la phase d'une insertion, ou le sous-ensemble de n-uplets à retirer de la vue dans le cas d'une suppression, sans réévaluer intégralement la vue. Le deuxième type garantie que la maintenance d'une vue V peut être calculée seulement à partir de V et des modifications survenus sur les tables de bases sur lesquelles elle est définie. Le dernier type est effectué en utilisant des transactions de mise à jour.

Une transaction de maintenance est relativement longue et peut donc interrompre l'usage de l'entrepôt. Par conséquent, elle est exécutée fréquemment durant les périodes d'activité creuse (la nuit par exemple). Cette maintenance n'est pas souhaitable, car avec l'apparition d'internet, un entrepôt doit être opérationnel continuellement (24/24 h).

3.1.1.2.3 Similarité entre les index et les vues

En comparant les vues matérialisées et les index, nous remarquons qu'ils possèdent de fortes similarités :

- Chacun des deux techniques sont redondantes et partagent la même ressource qui est l'espace de stockage. L'administrateur doit partager l'espace de stockage entre elles.
- Elles nécessitent des mises à jour régulières.
- Une vue est matérialisée sous forme d'une table relationnelle (dans le contexte ROLAP), donc son indexation peut Contribuer a l'amélioration des performances pour les requêtes référençant cette vue.

La présence d'un index peut rendre une vue plus avantageuse.

Plusieurs travaux ont identifié cette similarité et proposé une sélection simultanée des vues et des index.

3.1.1.3 Fragmentation verticale

La fragmentation verticale consiste à partitionner une relation ou une vue en un ensemble de sous-relations appelées fragments verticaux qui sont des projections appliquées à la relation.

Chaque fragment de la relation est composé d'un ensemble d'attributs de la relation d'origine. Le résultat du processus de fragmentation verticale est appelé schéma de fragmentation. Ce schéma est défini par l'ensemble des fragments ainsi que les attributs composant chaque fragment Soit une relation$R(K, A_1, A_2, \dots, A_n)$ composée de n attributs non clés, soit K la clé de *R*. La fragmentation verticale de la relation R en p fragments $R_1, R_2, \dots\dots, R_p$est effectuée en regroupant certains attributs dans un seul fragment. Chaque fragment vertical R_i de R est défini par :

$$R_i = (K, A_i^1, A_i^2, \dots\dots, A_I^j)$$

Où chaque attribut A_i^jpeut être un attribut quelconque de $R\left(A_i^j \in \{A_1, A_2, \dots, A_n\}\right)$
Notons que la clé K est dupliquée dans tous les fragments. Cela est nécessaire pour permettre la reconstruction des tuples (ou la totalité de la relation) pour les requêtes accédant à des attributs appartenant à plusieurs fragments en même temps. Cette reconstruction se fait à l'aide de l'opération de jointure entre les fragments.

La fragmentation verticale peut être définie avec ou sans réplication. La réplication dans la fragmentation verticale consiste à dupliquer certains attributs sur plusieurs fragments qui deviennent donc non disjoints. Cette réplication permet d'éviter certaines jointures coûteuses entre fragments.

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 nécessaires à charger pour répondre à la requête. Son inconvénient est qu'elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments.

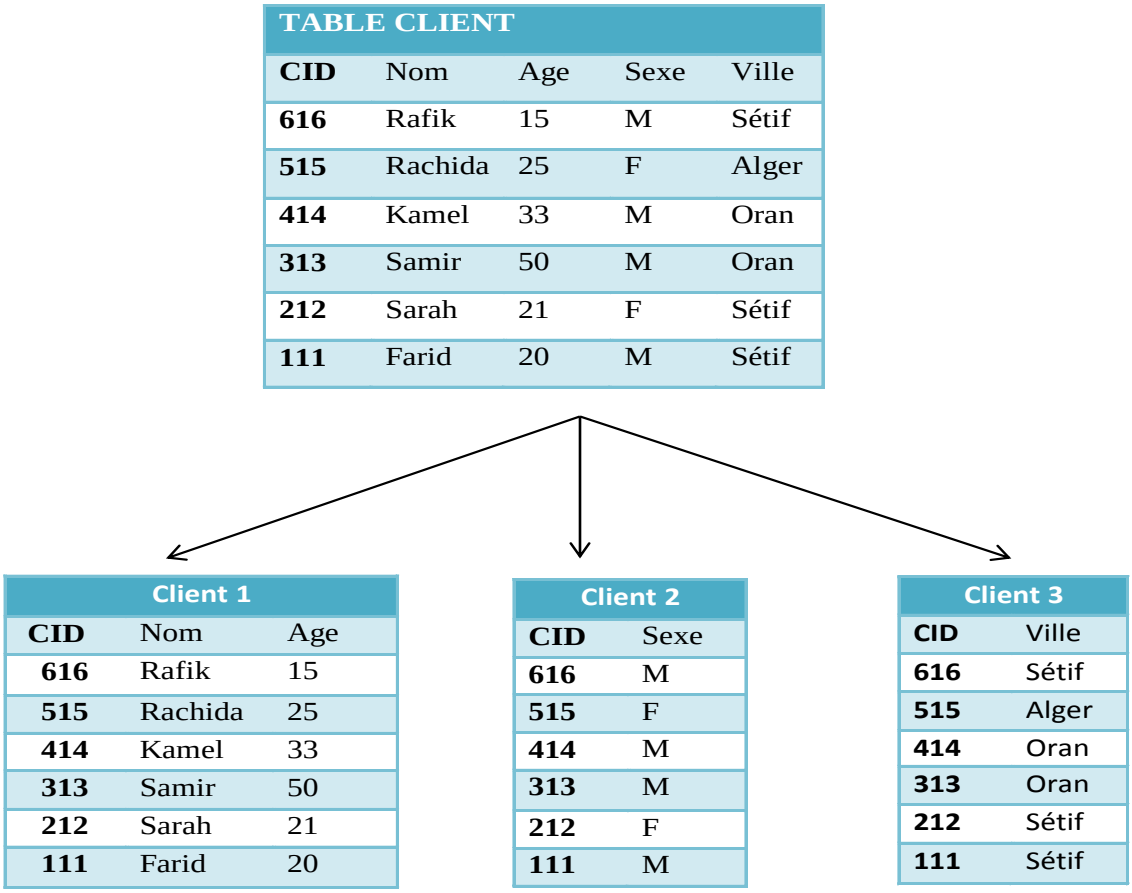

TABLE CLIENT

CID	Nom	Age	Sexe	Ville
616	Rafik	15	M	Sétif
515	Rachida	25	F	Alger
414	Kamel	33	M	Oran
313	Samir	50	M	Oran
212	Sarah	21	F	Sétif
111	Farid	20	M	Sétif

Client 1

CID	Nom	Age
616	Rafik	15
515	Rachida	25
414	Kamel	33
313	Samir	50
212	Sarah	21
111	Farid	20

Client 2

CID	Sexe
616	M
515	F
414	M
313	M
212	F
111	M

Client 3

CID	Ville
616	Sétif
515	Alger
414	Oran
313	Oran
212	Sétif
111	Sétif

FIG. 3.12 La fragmentation verticale de la table client

3.1.2 Techniques d'optimisation non redondantes

Dans cette section, nous nous attachons à présenter deux techniques d'optimisation non redondantes, à savoir la fragmentation horizontale primaire définie et la fragmentation horizontale dérivée

3.1.2.1 La fragmentation horizontale

La fragmentation horizontale permet de partitionner les objets de la base de données soit des tables, vues ou index en plusieurs ensembles de segments nommés *fragments horizontaux*, ou chaque segment est une instance de l'objet fragmenté. Les instances appartenant au même fragment horizontal vérifient souvent un prédicat de sélection. Chaque fragment horizontal T_id'une table T est défini par une clause de sélection sur la table T comme suit :

$$T_i = \sigma_{cl_i}(T)$$

La fragmentation horizontale est un aspect très important rentrant dans la conception physique des bases de données [67, 76]. Elle repose sur une technique d'optimisation non redondante

parce qu'elle ne réplique pas de données. Elle a un impact similaire sur la performance des requêtes définies sur des volumes de données importants. Elle a aussi un effet significatif sur la manageabilité et la maintenance des données.

Deux sortes de fragmentation horizontale sont disponibles : primaires et dérivée. Le premier type de fragmentation horizontale d'une table dite primaire se repose sur les prédicats de sélection définis sur cette table. Le deuxième type de fragmentation horizontale dite dérivée permet d'exploiter le lien existant entre deux tables pour fragmenter l'une d'entre elles en fonction des fragments de l'autre. Donc, la fragmentation horizontale dérivée d'une table se base sur les prédicats de sélection définis sur une autre table. En réalité, la fragmentation dérivée d'une certaine table S n'est possible seulement si elle est liée avec une table T via sa clé étrangère. Quand la table T fragmentée par la fragmentation primaire, les fragments de S sont commence à générer par une opération de semi-jointure entre S et chaque fragment de la table T. Les deux tables seront équi-partitionnées grâce au lien père-fils.

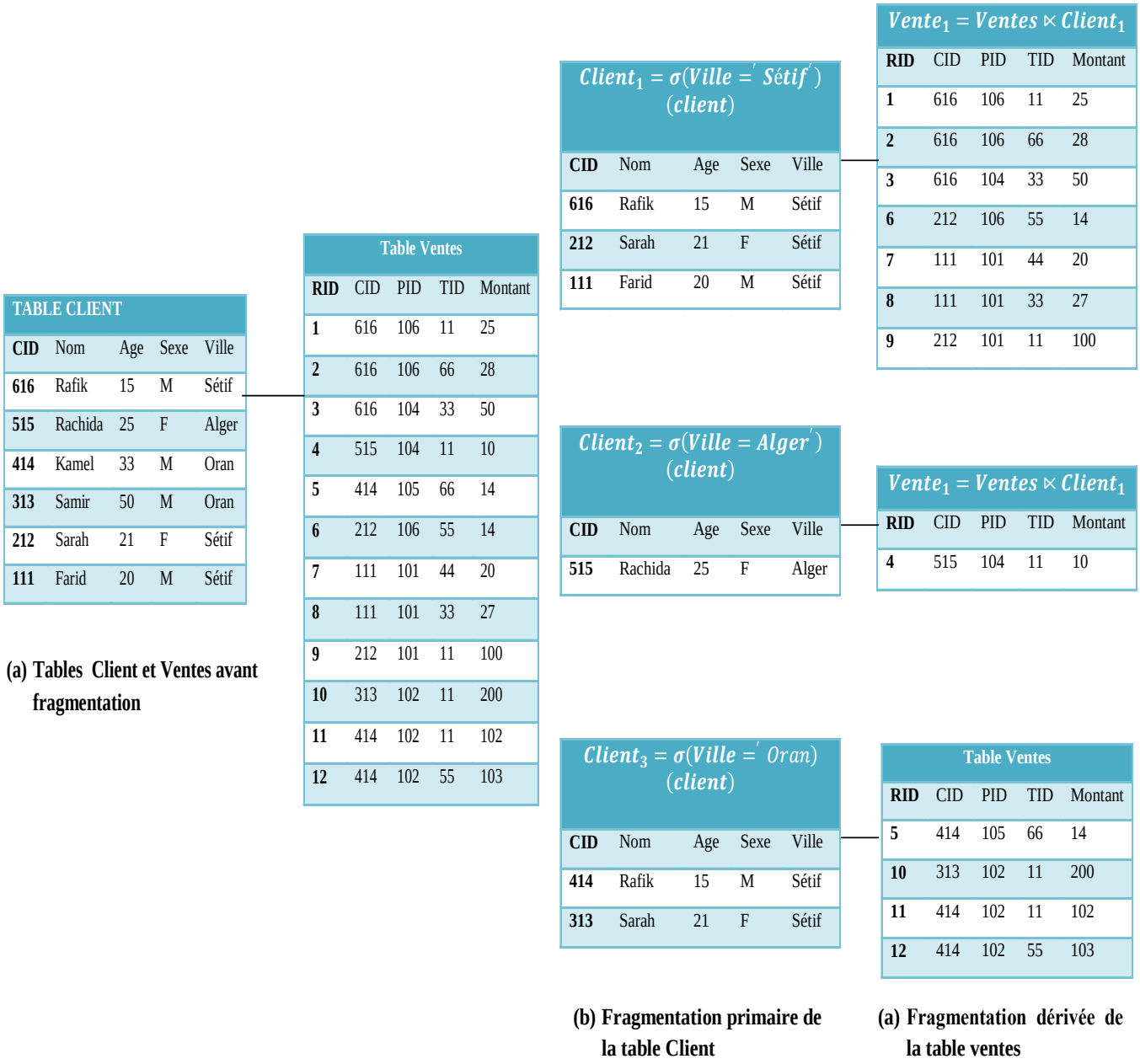

FIG. 3.13 Exemple d'une fragmentation primaire et dérivée

3.1.2.2 Évolution de la fragmentation horizontale

La fragmentation horizontal est considéré comme une crucial technique d'optimisation c'est pour ça, plusieurs chercheurs ainsi que leurs travaux sont destinés vers ce genre de fragmentation. Ils proviennent de deux aspects différents : académique et industrielle. Les travaux académiques s'attachent surtout au traité de la fragmentation horizontale du point de vue théorisation et proposition d'approches de fragmentation.

L'aspect industriel s'attache à mettre en œuvre des commandes DDL (langage de définition de données) consiste à fragmenter physiquement les objets de la base de données et d'offrir des fonctions de manipulation des partitions obtenues. Nous montrons la progression et l'évolution de la fragmentation horizontale à travers ces deux aspects dans la section suivante.

Contexte académique : on remarque que plusieurs travaux de recherche se sont captivés à la fragmentation horizontale [70, 66, 82, 85, 68]. Ces travaux intéressent principalement aux différents contextes, comme les bases de données centralisées, les bases de données distribuées et parallèles, les entrepôts de données centralisées, les entrepôts de données distribués et parallèles.

La fragmentation horizontale a été premièrement apparue vers la fin des années 70 pour la conception logique des bases de données dans l'objective d'augmenter et d'améliorer les performances des requêtes. La fragmentation horizontale a vécu son grand succès durant les années 80. Elle a été largement utilisée pour la conception des bases de données distribuées [70, 66] et les bases de données parallèles [82, 85, 68].

Dans le cadre des bases de données distribuées, elle consiste à partitionner une table dite globale en un ensemble de fragments horizontaux dans lequel chaque fragment doit être affecté à un nœud du système. Les utilisateurs finaux sur chaque nœud peuvent réaliser des transactions locales sur les fragments alloués à ce nœud, ce qui diminué énormément le coût d'exécution des requêtes.

Dans le cadre de bases de données parallèles, la fragmentation horizontale a été exploitée pour permettant d'accélération d''exécution des requêtes où les données sont horizontalement partitionnées et allouées sur un ensemble de nœuds indépendants. La fragmentation horizontale acquiesce de concrétiser une exécution parallèle intra-requêtes et inter-requêtes [82].

Dans le cadre d'entrepôt de données, notre domaine de recherche dans cette mémoire, la fragmentation horizontale a été introduite comme un choix important de conception physique [25].en effet, Elle consiste à partitionner les tables de l'entrepôt (tables de dimension et/ou table des faits) [58, 71], les vues matérialisées et les index en un ensemble de fragments de plus petite taille. Elle a été utilisée principalement pour le but d'améliorer la performance des

requêtes en leur permettant de ne charger que les fragments pertinents, ce qui réduit le volume de données remplis ou chargées. Elle a été aussi exploitée pour atteint l'objective d'améliorer la manageabilité de l'entrepôt en permettant de gérer un fragment à la fois et d'utiliser toutes les opérations employées (définies sur les tables globales) au niveau de la partition.

Dans le cadre d'entrepôts de données parallèles, la fragmentation horizontale a été considérée comme la technique la plus importante. Parce que elle suppose une augmentation des performances du système en permettant, tell que dans le cadre de bases de données parallèles, une exécution parallèle intra-requête et inter-requêtes des requêtes OLAP définies sur l'entrepôt.

Le majeur intérêt de La fragmentation horizontale, c'est sa capacité de combinaison avec d'autres techniques d'optimisation, toujours pour notre but l'amélioration des performances des requêtes. [71] ont travaillé sur la combinaison de la fragmentation horizontale avec les index et les vues matérialisées, [68] ont travaillé aussi sur une combinaison avec le traitement parallèle. Sanjay et al. [67] ont traité ce problème avec une combinaison de la fragmentation horizontale avec la fragmentation verticale dans la conception physique des bases de données volumineuses. Stohr. [29] proposent à combiné de la fragmentation horizontale avec les index binaires et le traitement parallèle dans la conception des entrepôts de données parallèles. [35] ont traité la combinaison de la fragmentation horizontale avec les vues matérialisées, les index et le clustering.

2. Contexte industriel : comme on a vue ou contexte académique, Les éditeurs de SGBD commerciaux se captivent surtout à la fragmentation horizontale. Plusieurs sortes de fragmentation ont été proposées. Actuellement, la majorité des SGBD commerciaux soutiennent la fragmentation horizontale et proposent des commandes DDL pour fragmenter les objets de la base de données et manipuler les partitions obtenues. On peut classer les sortes de fragmentation en deux catégories, simple (à un seul niveau) et composé (à deux niveaux). Une fragmentation simple permet de partitionner une table selon les valeurs d'un seul attribut en un ensemble de partitions. Et la fragmentation composée consiste à fragmenter une table en un ensemble de partitions en utilisant plusieurs attributs.

3.1.2.3 Bilan et discussion

Parmi ces technique d'optimisation d'entrepôt de données présenté, ce que nous attirons effectivement La fragmentation horizontale, elle considéré aussi comme une technique d'optimisation très importante de la conception physique d'un entrepôt de données. Grace a s aptitude d'optimiser les requêtes et faciliter la gestion des données selon le principe de diviser pour mieux gérer.

Ce chapitre montre que les algorithmes de fragmentation présentés sont généralement guidés par les prédicats de sélection utilisés par les requêtes. Sélectionner un schéma de fragmentation c'est une tâche pénible, cependant, plusieurs approches de sélection ont été proposées. Ces dernières présentent plusieurs limites.

Les approches de sélection repose sur les prédicats de sélection sont définies par une grande complexité (pour n prédicats, 2minterms sont générés). Les approches repose sur les affinités sont moins complexes, mais n'existe aucune métrique pour estimer la qualité du schéma de fragmentation obtenu. Ce que on essai de le faire au chapitre suivant.

Le tableau (**TAB.2.3**) résume les principaux travaux effectués sur la fragmentation horizontale.

Travaux	**Type de fragmentation horizontale**	**Approche de sélection**	**Algorithme de sélection**	**Modèle de coût**
Ceri[70]	Primaire	Basée sur les prédicats		Non
Zhang [30]	Primaire	Basée sur les affinités	BEA + Groupement graphique	Non
Ozsü [66]	Primaire	Basée sur les prédicats		Non
Bellatreche[58]	Primaire et dérivée	Basée sur un modèle de coût	Hill Climbing	Mathématique

TAB 3.2 Comparaison des travaux effectués sur la fragmentation horizontale.

Ce tableau montre pour chaque travail le type de fragmentation effectué (horizontale primaire, horizontale dérivée), l'approche de fragmentation suivie (basée prédicats, affinité ou modèle de coût), l'algorithme utilisé et le modèle de coût utilisé.

Conclusion

Nous avons présenté dans ce chapitre un état de l'art sur les principales techniques d'optimisation de requêtes définies dans le contexte des entrepôts de données relationnels : les index avancés, les vues matérialisées, la fragmentation horizontale. Pour chaque technique, des algorithmes de sélection sont présentés. Chaque implémentation est analysée en mettant en évidence ses avantages et ses inconvénients.

4

La fragmentation horizontale

Introduction

La fragmentation est une technique de conception logique, elle représente une structure d'optimisation très importante inspiré de la conception des bases de données et exploité largement dans le domaine des entrepôts de données, cette technique caractérisé par un impacte très important sur la performance des SGBDs [48], puisque elle est équipée d'une méthode afin de fragmenter soit des tables, index ou même des vues en plusieurs fragments disjoints qui seront stockés et accédés séparément.

4. Les fragmentations

Dans la littérature, il existe deux types de méthode effectué la segmentation d'une relation: le partitionnement et la fragmentation. La différence entre ces deux structures consiste à la taille des partitions dans chacun technique parce que pour le partitionnement, la division de la relation se faite en plusieurs partitions disjointes. En revanche a la fragmentation ou l'opération de division se termine en plusieurs fragments qui peuvent être non disjoints. Et il existe deux classes de fragmentation : la fragmentation horizontale et la fragmentation verticale.

4.1 Fragmentation verticale

Comme on a déjà vue dans le chapitre précèdent, la fragmentation verticale d'une relation *R* répartie les attributs de la relation afin de générer plusieurs tables$\{R_1, R_2, \ldots\ldots, R_k\}$, ou chacun de ces tables contenant un sous-ensemble des attributs de *R* et bien sur de la clé primaire de R.

Cette technique nous admettons d'accélérer l'exécution des requêtes décisionnelles appliques sur les entrepôts de données, puisqu'elle élimine la duplication et la redondance des données par le stockage d'index, cette opération de stockage concernant en réalité les index lui même et non pas les colonnes indexées, ce que résulte d'une réduction d'espace nécessaire pour ces index.

Plusieurs travaux ont été offerts dans la littérature exploitent les fragments verticaux comme principe d'optimisation, cette technique est basé sur deux majeurs concepts premièrement les affinités des attributs [24], et deuxièment a base d'un modèle de simulation [54].

4.1.1 Problème de la fragmentation verticale

La fragmentation verticale est considérée comme un enjeu intéressant dans les entrepôts, Cette importance est née à la manière de division et a le choix des tables candidats à fragmenter soit tables de dimensions ou des faits. Les choix possible sont illustrés comme suit :

- Fragmenter verticalement les tables de dimensions. Ce choix est très captivant lorsque la taille des tables de dimensions est importante, telle que, le cas de modélisation par un schéma en étoile. Donc cette sorte de fragmentation permet de réduire le coût des opérations de jointures en étoile.
- Le deuxième choix consiste à partitionner seulement la table de faits. Cette dernière qu'elle a constitué des clés étrangères des tables de dimensions et des mesures. En effet, la fragmentation verticale de la table de faits permet de fragmenter l'ensemble des mesures en différents fragments. Et en remarquant que les clés étrangères seront dupliquées dans chaque fragment vertical.

D'une manière générale, le problème de fragmentation verticale peut être formalisé comme suit : Soit un ensemble de tables de dimension $D = \{D_1, D_2, \ldots\ldots, D_d\}$et une table de faits *F*. et une charge de requêtes *OLAP* fréquentes $Q = \{Q_1, Q_2, \ldots\ldots, Q_m\}$, où chaque requête $Q = (1 \leq i \leq m)$ possède une fréquence d'accès f_{Qi}et un seuil □*M*□qui considère comme un nombre prédéterminé de fragments verticaux que l'administrateur souhaiterait avoir.

Le problème dans ce type de fragmentation est subdivisé en deux grand sorte : premièrement la difficulté de déterminer un ensemble de tables de dimension à fragmenter, et deuxièment la tache de fragmenter une ou des tables de dimension de manière ou le coût total d'exécution de charge de requêtes sur le schéma en étoile fragmenté soit minimale et que le nombre de fragments verticaux respectant le seuil *M*.

4.1.2 Les algorithmes de la fragmentation verticale

La technique de fragmentation verticale et ces algorithmes sont mois exploitable par rapport au autre techniques et parmi les approche existant on cite la célèbre approche de Navathe [73], qui suppose l'existence d'une relation ayant *m* attributs prêt à fragmenter et un ensemble de *n* requêtes les plus fréquentes. Elle s'exécute en deux étapes principales : dans la première étape, on commence par la construction de trois matrices : la matrice d'usage des attributs, la matrice des affinités des attributs et la matrice d'affinité ordonnée.

Pour la première matrice ou la matrice d'usage des attributs, on remplit l'élément de la ligne i et dela colonne j par la valeur 1 si l'attribut j est accédé par la requête i, sinon on considère cette valeur comme nulle. Les champs de cette matrice sont complétés par des valeurs de fréquences d'accès pour chacune des requêtes représentées dans une colonne supplémentaire.

La deuxième matrice ou la matrice des affinités d'attributs possédant m lignes et n colonnes représente les attributs de la relation à fragmenter. La valeur de l'élément de la ligne $i$et de la colonne $j$transcrit les valeurs d'affinités définies entre ces attributs. Cette affinité ressemble à la somme des fréquences d'accès des requêtes accédant simultanément aux deux attributs.

La troisième matrice ou la matrice d'affinité ordonnée est généré par l'application de l'algorithme de B.E.A (Bound Energy Algorithm) [75] sur la matrice des affinités d'attributs.

On permutant les lignes et les colonnes, l'algorithme précèdent travaille a regroupé les attributs qui sont traités simultanément en fournissant une matrice sous la forme d'un semi-bloc diagonal.

Les informations exploitées dans la 1ère partie sont de type quantitatif. En réalité, elle se pose sur le comportement des applications et a pour atteint l'objective de s'installer dans une même partition les attributs qui sont fréquemment accédés ensemble. Une mesure présentant la proximité des attributs est l'affinité des attributs.

Etant donnée $Q = \{Q_1, Q_2, \dots., Q_m\}$l'ensemble de requêtes candidat prêt à s'exécuter sur une relation$R(A_1, A_2, \dots\dots, A_n)$. A chaque requête $q_i$et chaque attributA_j, on accompagné d'une valeur d'usage de l'attribut noté $use(q_i, A_j)$définie comme suit :

$$use(q_i, A_j) = \begin{cases} 1 \; si \; l'attribut \; est \; référencé \; par \; une \; requete \\ 0 \; sinon \end{cases}$$

L'ensemble de requêtes Q et l'ensemble d'attributs $\{A_1, A_2, \dots., A_k\}$sont formé une matrice nommée matrice d'usage des attributs

$(A_1 A_2 \dots\dots\dots\dots\dots.. A_k)$

$$\begin{pmatrix} q_1 \\ q_2 \\ . \\ . \\ q_n \end{pmatrix} \begin{pmatrix} 1 & 0 & \dots\dots\dots & . & . \\ 0 & . & & & \\ 1 & & . & & . \\ . & & & 1 & 0 \\ . & & & 0 & 1 \end{pmatrix}$$

Mais l'utilisation de cette matrice n'est pas suffisante pour admet une fragmentation robuste d'une relation puisque les valeurs qu'elle contient ne représentent pas les fréquences des différentes applications. Navathe[73] a ajouté la quantité $aff(A_1, A_j)$, qui fait le mesure d'affinité entre deux attributs $A_i$et A_jd'une relation R en fonction de l'ensemble d'applications Q .

$$aff(A_i, A_j) = \sum_{k/use(q_k, A_i)=1 \wedge use(q_k, A_j)=1} \sum_{\forall S_l} ref_l(q_k) acc_l(q_k)$$

Où $ref_l(q_k)$représente le nombre d'accès aux attributs (A_i, A_j) pour chaque exécution de l'application $q_k$au site$S_l$et $acc_l\ (q_k)$représente la fréquence d'accès de l'application.

On peut note aussi que la matrice d'affinité est applique d'une façon accompagné au processus de fragmentation, qui permet dans un premier lieu de grouper l'ensemble des attributs caractérisé par une forte affinité, et dans un deuxième temps effectué la fragmentation de la relation selon ces groupements.

Le deuxième algorithme de la fragmentation verticale est proposé par Hammer et all [54] : qui ont proposé une solution repose sur un modèle de simulation pour l'évaluation des performances présentées par le schéma à la suite d'une fragmentation verticale (**FIG. 4.1**). Cette technique est comporte deux composantes fondamentales :

- Un générateur de partitions.
- Un évaluateur de performances qui offre une valeur de mérite à chaque partition.

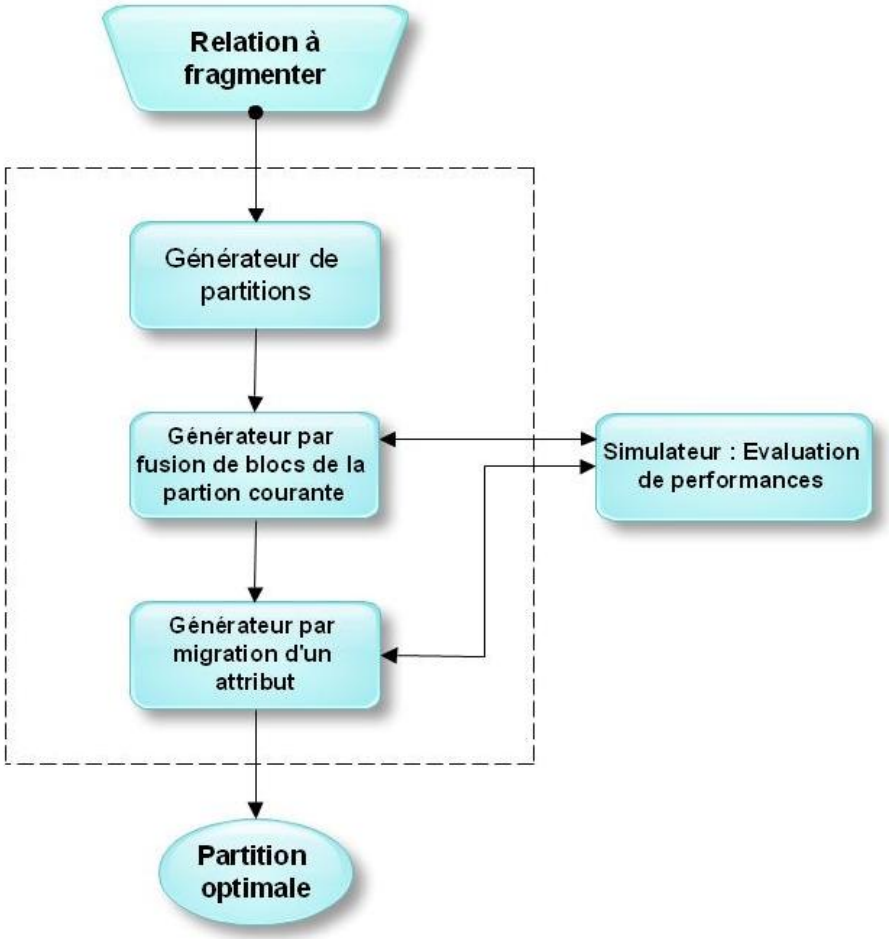

FIG. 4.1 Principe général de l'algorithme de Hammer

4.2 Fragmentation horizontale

La fragmentation horizontale permet de partitionner des tables ou vues en fragments horizontaux selon les valeurs des attributs. Ou chaque fragment généré contient uniquement un sous-ensemble de n-uplets ayant des propriétés similaire [79].

L'intérêt d'utilisation de cette technique représente à la possibilité et à la flexibilité de contrôler des petites unités physiques de données que des grandes unités qu'elle admet pour les administrateurs et les concepteurs [81].

La fragmentation horizontale est divisé en deux catégories ; la fragmentation primaire et dérivée [83].

La fragmentation primaire d'une relation est réalisée par l'utilisation d'un ensemble de prédicats définie sur cette relation. Ce type de fragmentation permet d'optimiser l'ensemble

des opérations sur une relation par la réduction des accès aux données impertinentes et permet ainsi aux requêtes d'être exécutées concurremment pour l'objective de supporter le parallélisme. Par exemple dans le cas ou il existe une requête contient un attribut de sélection dans la clause WHERE, l'optimiseur du SGBD guide cette requête directement vers la partition valide.

D'une manière Formelle, étant donnée D une base de données possèdent un ensemble de relation $R_1, \dots, R_n$ où chaque relation $R_i(A_1, \dots . A_m)$ contient un ensemble d'attributs.

Chaque attribut A_j , $1 < j < M$ contient un domaine de valeurs$Dom(A_j) = \left(d_{j1}, d_{jN_j}\right)$. Les fragments R_{i1} , … … R_{ik}sont le résultat d'une fragmentationhorizontale primaire FHP de la relation $R_i$doivent satisfaireun ensemble de prédicats de sélection. Des opérations d'union sont nécessairespour reconstruire une relation$R = \bigcup_{i=1}^{n} R_i$.

Par contre à la fragmentation primaire, la fragmentation dérivée est réalisée avec des prédicats définis sur une autre relation [86]. La fragmentation dérivée est considérée comme une tache très complexe par rapport à celui de la fragmentation primaire. Ce dernier est basé sur des schémas de fragmentation et aussi sur des algorithmes de sélection et de validation [88].

D'une manière Formelle, supposant qu'on a deux relations *R* et *S, R* possédant une clé secondaire de S. La relation R est partitionnée horizontalement en un ensemble de fragments $R_1, \dots . R_k (1 \leq k \leq M)$, avec $R_j = \sigma_{cl_j}$où cl_jreprésente l'ensemble desprédicats de sélection. La relation *S* est partitionnée avec la fragmentation horizontale dérivée en ensemble de fragments $\{S_1, \dots . , S_k\}$, $1 \leq k \leq M$. Chaquefragment S_k est calculé par la formule: $Sk = S \ltimes R_k (1 \leq k \leq M)$et $S_j$est une opération de semi-jointure, chaque fragment $S_k$contient tout les N uplets de la relation *S* jointe avec la relationR_k. Le principal objectif de lafragmentation horizontale et de minimiser le coût de jointure entre les deuxrelations *R* et *S*.

4.2.1 Problème de la fragmentation horizontale

Le domaine des entrepôts de données relationnels a exploité largement la fragmentation horizontale, cette technique permet de partitionner soit des tables, des index ou même des vues matérialisées en plusieurs ensembles de lignes distincts, stocké d'une façon physique et prête a consultés d'une maniéré séparé [89]. Contrairement aux autres techniques

d'optimisation telle que les vues matérialisées et les index, la fragmentation ne duplique pas les données, en conséquence elle réduit le coût de mises à jour [92]. Le problème dans la fragmentation horizontal consiste à la possibilité de recherche d'un ensemble des tables de dimension à fragmenter et d'exploiter ses schémas de fragmentation pour fragmenter la table de faits *F* en un nombre de fragments horizontaux sous le nom fragments de faits, tels que le coût d'exécution de ces requêtes sur le schéma en étoile fragmenté soit réduit.

4.2.2 Principe de la fragmentation

Donc, on peut dire que le réel problème dans ce genre de fragmentation c'est un problème de sélection d'un schéma de fragmentation, cependant, ce dernier problème est considéré comme un enjeu crucial dans le domaine des entrepôts de donnée et pour mieux comprendre ce problème on essai de le formulé (**FIG. 4.2**) comme suit :

Soit un ensemble de tables de dimension $D = \{D_1, D_2, \dots D_d\}$ et une table de fait F , un ensemble de requêtes OLAP fréquentes $Q = \{Q_1, Q_2 \dots, Q_m\}$, ou chaque requête Q_i $(1 \leq i \leq m)$possède une fréquence d'accès, et un seuil W qui représente le nombre maximum de fragments qu'il peut maintenir(cette valeur est fournie par l'administrateur de la base de données). Et que la satisfaction de cette contrainte évite le problème d'expansion de nombre des fragments de la table de fait (noté *N)*.

$N = \prod_{i=1}^{g} m_i$ *o*u m_i représentent le nombre de fragments de la table de dimension $D_i$et g le nombre de tables de dimensions adhérent à l'opération de fragmentation.

Donc, le but à atteindre est de subdiviser le schéma de l'entrepôt de données N en n sous schémas $\{m_1, m_2, \dots\dots, m_n\}$ sachant que l'exécution de la charge de requêtes Q soit terminé avec un coût minimale. En conséquent le choix d'un schéma de fragmentation horizontale optimale est un problème NP-Complet [88].

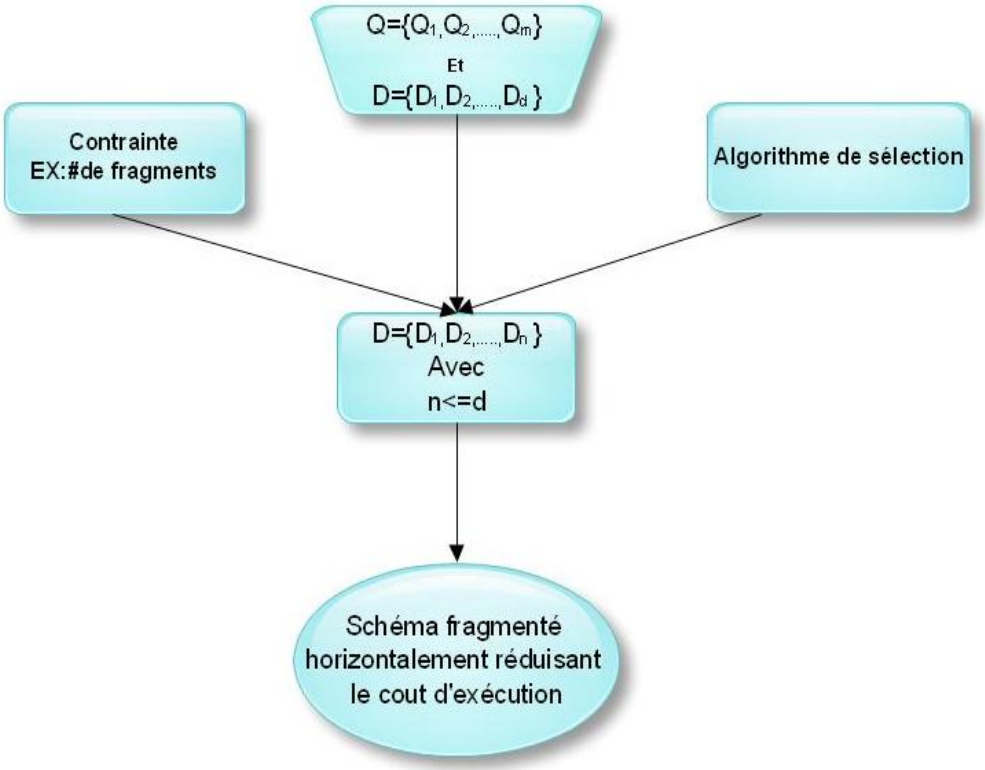

FIG. 4.2 Processus de sélection des dimensions.

4.2.3 Propriétés de la fragmentation

Pour le bon fonctionnement du processus de fragmentation dans les entrepôts de données il faut qu'assurer trois propriétés [93] : la complétude, la reconstitution, et la disjonction.

La complétude permet d'assurer que tous les n-uplets d'une relation sont associés à au moins un fragment.

La reconstitution consiste a garantir que la relation peut être reconstruite à partir de ses fragments. L'application de cette propriété sur la fragmentation horizontale nous permettant de reconstitué la table des faits et des tables de dimensions par l'opération d'union, c'est-à-dire

$$F = \bigcup_{i=1}^{N} F_i \; et \; D_i = \bigcup_{j=1}^{m_i} D_{ij}$$

La dernière propriété consiste à la disjonction, cette propriété permet d'assurer que tous les fragments d'une relation sont distincts deux à deux, pas d'intersection.

Enfin, on note que les algorithmes proposé pour traités le problème de sélection d'un schéma de fragmentation horizontale doit respecter les propriétés derniers, sinon on rencontre des difficultés au niveau de traitement de donnée comme des pertes de données ou même des redondances au niveau des données, et cela induit l'incohérence des données initiales.

4.3 Les approches de sélection d'un schéma de fragmentation

4.3.1 Approche à base de génération des prédicats

Presque tout les travaux qu'ont étudié la fragmentation horizontale, toucher inévitablement le problème de sélection d'un schéma de fragmentation [94, 88, 42], soit dans le cadre des bases de données classiques, bases de données distribuées ou entrepôts de données, ces travaux sont divisés en Quatre principales approches [88]. Dans ce que suit, on montre les principales étapes de cette approche :

- Générer l'ensemble M de minterme des prédicats : M ={$m|m = \bigwedge(1 \leq k \leq N)p_k^*$} où p_k^* soit p ou $\bar{p}$.
- La réduction de l'ensemble M par la suppression des prédicats inutiles.
- Appliquer la tache de génération des fragments, où chaque minterme m génère des fragments par une sélection $\sigma_m(R)$ où σest le symbole représente la sélection.

Les critiques sur cette première approche, que elle considéré comme une approche très simple mais en même temps, elle est coûteuse. Si on prend n prédicats, l'algorithme peut génère jusqu'a 2^n mintermes, donc l'application de cette algorithme devient une tache laborieuse avec ce grand nombre de prédicats envisagé.

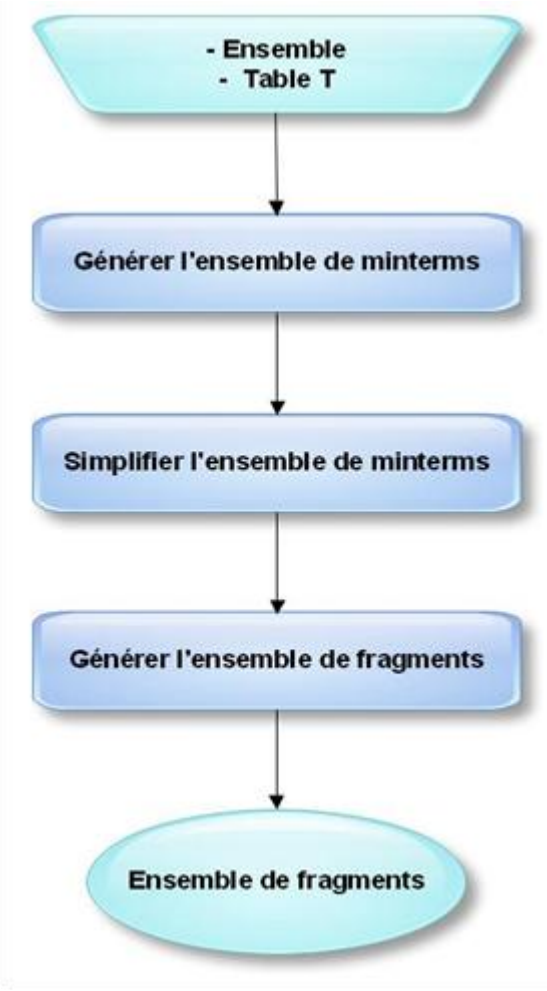

FIG. 4.3 Approche à base de génération des prédicats.

4.3.2 Approche à base d'affinité

La deuxième approche pour traiter le problème de sélection d'un schéma de fragmentation c'est l'approche à base d'affinité, elle caractérise par une complexité réduite par rapport à la premier approche, l'ide globale de cette approche est parvient d'une solution concernant à la résolution du problème de fragmentation verticale [24].

Elle repose sur le regroupement des prédicats qui ont une très grande affinité. L'opération de calcul d'affinité entre deux prédicats P1 et P2 se fait en additionnent les fréquences des requêtes qui accède simultanément aux prédicats P1 et P2, enfin on découvre que chaque groupe indique un seul fragment horizontal.et on peut résumer les principales étapes de cette comme suit :

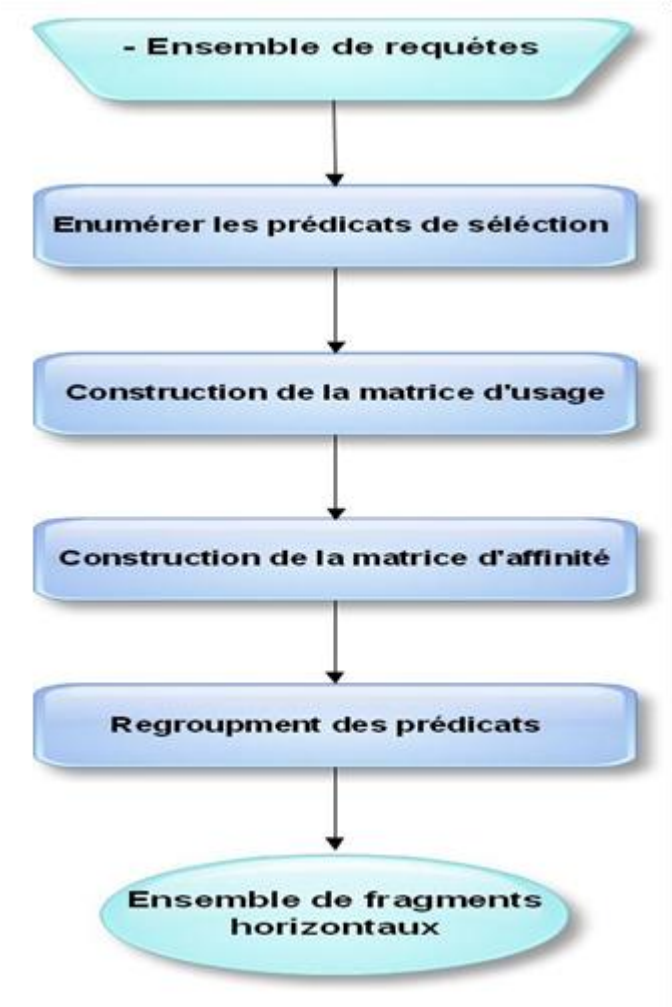

FIG. 4.4 Approche à base d'affinité.

- Premièrement, parmi l'ensemble de requêtes qu'on a, identifier puis énumérer tous les prédicats de sélection définis sur les attributs de fragmentation.
- La procédé de construction de la matrice d'usage, une matrice de dimension m×n oùm est le nombre de prédicats et n le nombre de requêtes, cette matrice est remplis de la manière suivante : Les valeurs des cases de cette matrice peut être 1 si la requête q_j a utilisé le prédicat Pi et 0sinon.
- Le procédé de construction de la matrice d'affinité, une matrice de dimension m×m oùm le nombre de prédicats, le remplissage de cette matrice est en fonction du calcul d'affinité.
- L'étape avant dernière consiste à regrouper les prédicats dans des sous-groupes.
- Générer ou détecter les fragments horizontaux.

L'approche d'affinité est considérée comme une approche traditionnelle parce qu'elle prend en considération uniquement les fréquences des requêtes sans qu'elle entre d'autres facteurs comme la taille des tables ou les facteurs de sélectivité des prédicats.

4.3.3 Approche à base d'un modèle de coût

La première progression de l'approche d'affinité vient avec les travaux de Bellatreche et al. [94] qui ont travaillés a développé une approche à base d'un modèle de coût mathématique permet de mesurer les entrées/sorties nécessaires pour exécuter une requête séparément de l'optimiseur du SGBD.et on peut résumer les principales étapes de cette comme suit :

- Premièrement : les prédicats de sélection sont énumérés à partir d'une charge de requêtes.
- Puis : un ensemble de solutions potentielles de schémas de fragmentation est proposé.
- Finalement : ces schémas sont évalués avec un modèle de coût pour sélectionner un schéma optimal qui a un coût réduit.

L'approche base d'un modèle de coût exploite à la fragmentation horizontale dérivée, par contre aux deux approches précédentes qui n'exploite qu'à la fragmentation horizontale primaire. Donc l'emploie des approches à base d'affinité et génération de prédicats sont inefficaces dans le cas des entrepôts de données, car on est toujours besoin de fragmenter les tables de dimension et la table de faits à la fois.

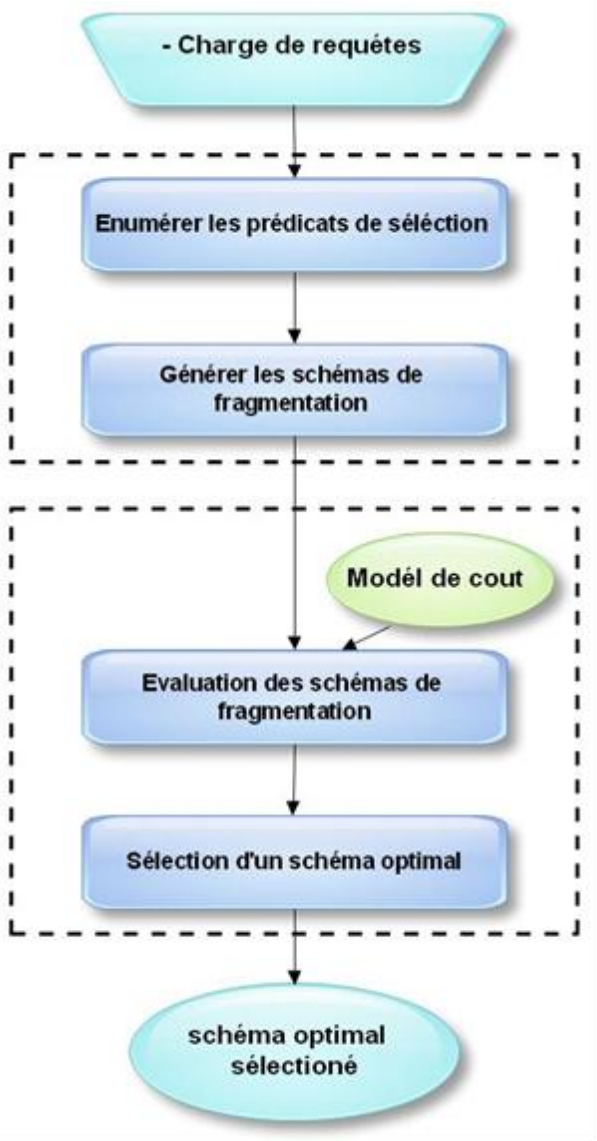

FIG. 4.5 Approche à base de model de cout .

4.3.4 Approche à base d'un modèle de coût et contrôle des schémas

L'approche à base d'un modèle de coût et contrôle des schémas repose sur l'architecture précédente avec une addition de Contrainte de maintenance W qui permet à l'administrateur de l'entrepôt de données AED de tester le nombre de sous-schémas de fragmentation générer par la deuxième étape du processus de fragmentation. Donc on ne prend en compte que les schémas qui ont vérifié la contrainte de maintenance Par la suite les schémas valides seront évalués sur un modèle de coût mathématique [88]. La figure (**FIG. 4.6**) représente l'architecture complète de cette approche.

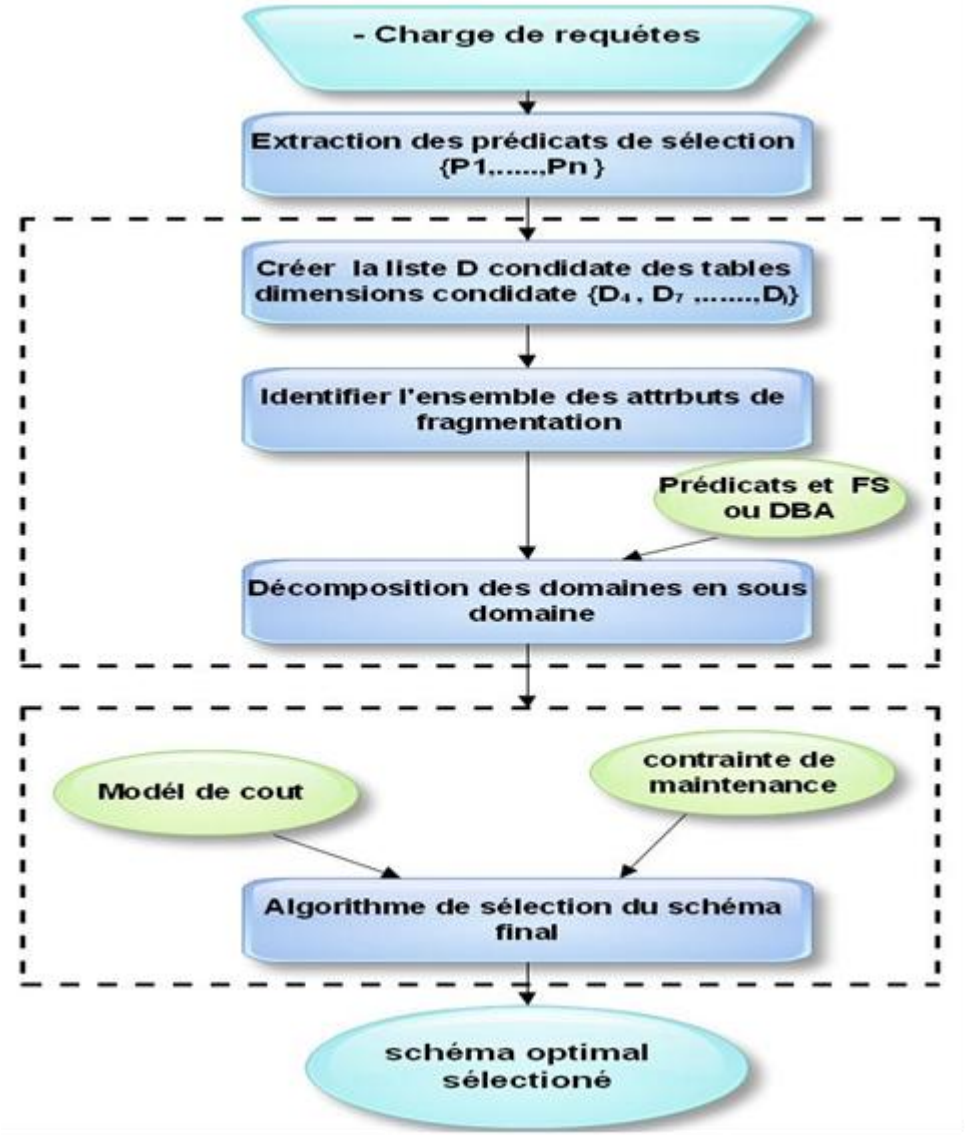

FIG. 4.6 Approche à base d'un model de cout et contrôle de schémas.

Le majeur intérêt de cette approche comparativement à l'approche précédente est qu'elle permet d'identifier l'ensemble des tables de dimension participantes dans la fragmentation selon les prédicats de sélection extraites d'une charge de requêtes Q.

On remarque que la contrainte de maintenance W a été intégrée dans l'application de cette approche afin de permettre à l'administrateur de l'entrepôt de données AED de vérifier les fragments générés.

4.4 Les algorithmes métaheuristiques pour la fragmentation horizontale

Le concept du métaheuristiques rassemble a une méthode de résolution heuristique, de caractère stochastiques et convergeant vers une solution optimal, par une évaluation d'une fonction objectif dont le but est la résolution d'un problème difficile a résoudre avec les algorithmes classique finie, Généralement, un problème d'optimisation est défini par une ou plusieurs fonctions ,cet dernière nous permettons de distinguer la bonne solution face à une autre moins performant.

Dans ce qui suit nous étalons le principe de deux algorithmes métaheuristiques (les recuits simulés et les algorithmes génétiques), et nous rappelons l'apport et la contribution de chaque algorithme dans la résolution du problème de sélection d'un schéma de fragmentation.

4.4.1 Hill climbing

Cet algorithme que nous présentons est dit algorithme Hill Climbing. L'algorithme de Hill Climbing est une méthode de voisinage appartient à la famille des techniques de recherche locale ,la puissance cette technique se résume dans sa simplicité à mettre en œuvre. Le HillClimbing peut être utilisé pour l'objective de résoudre des problèmes, d'optimisation combinatoire, qui peuvent avoir plusieurs solutions, parmi eux certaines sont meilleures et plus exacts que d'autres.

Le principe de cet algorithme est simple et composée de deux étapes essentielles :

- On commence par une solution potentiellement malgré que peut être faible, dans la majorité des cas elle choisit aléatoirement.
- On applique itérativement des petites modifications à la solution initiale, en le progressant à chaque fois. L'algorithme se termine quand il n'existe aucun progrès.

Si la solution finale obtenue est près de l'optimalité, on peut la considérer comme un meilleur cas, mais en même temps, n'a aucun rapport garantis que la solution obtenue par le Hill Climbing est optimale. Le principe de l'algorithme Hill Climbing se montré dans la Figure (**FIG. 4.7**)

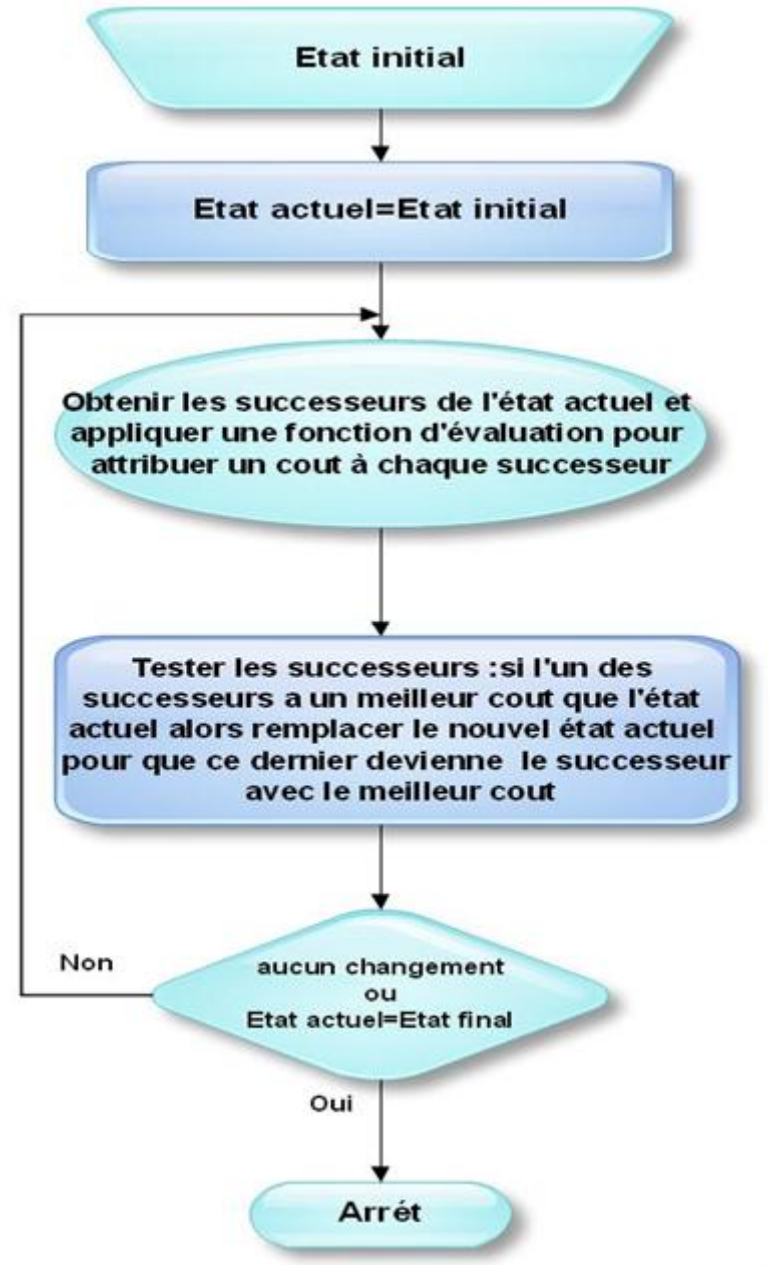

FIG. 4.7 L'algorithme hillclimbing.

4.4.1.1 Hill climbing pour la sélection d'un schéma de fragmentation

Boukhalfa et al.[95] ont proposé d'exploité le principe d'optimisation par le hillclimbing pour sélectionner un meilleur schéma de fragmentation. Ils ont criée une fonction objective pour évaluer le coût de chaque schéma. Le principe du recuit simulé exploité dans ce cas est présenté comme suit :

– **Etat initial :** La solution initiale peut être sélectionné de diffèrent manières. Comme le choix d'une solution aléatoire ou par l'enlever d'un autre algorithme. La solution aléatoire ne garantit aucune performance de l'algorithme.

C'est pour cela que, les concepteurs de cette méthode ont proposé d'exploiter les résultats de l'algorithme à base d'affinité et l'utilise comme un état initial de l'algorithme hillclimbing. Ce départ influence directement le résultat obtenu par le hillclimbing.

– **Mouvements effectués :** Pour découvrir des successeurs de l'état initial, ce groupe de recherche ont basé sur l'utilisation des deux fonctions Split et Merge, pour réaliser des améliorations sur l'état actuel et atteindre un successeur. Ce successeur doit être évalué par une fonction de coût telle que l'algorithme génétique ou le recuit simulé.

Le nouveau problème à rencontrer, c'est l'indécise de l'ordre d'application des fonctions Split et Merge sous la contrainte de maintenance W. Dans cette situation, si le nombre de schémas existe est supérieur à W, on admet une nouvelle fonction sous le nom Best Merge son emplois est d'obtenir un successeur avec un nombre de schémas inférieur ou égal à W. L'objectif d'utilisation de cette fonction Best Merge est de compter toutes le fusions possibles et de choisir la meilleure qui à le moindre coût.

4.4.1.2 Avantages et limites du hillclimbing

Nous avons montré dans cette section la technique Hill Climbing permettant d'améliorer une solution initiale existe déjà, le hillclimbing a des avantages et des limites. Parmi ses avantages, on à que cet algorithme est moins coûteux en temps d'exécution car il se base sur deux opérations simples.

Les limites de cette technique consistent à son blocage dans des optimums locaux, on remarque aussi que, ni l'emploi de plusieurs points de départ, ni l'accroissement du nombre de successeurs, ni l'élargir d'espace de recherche pendant chaque itération. n'ont prouvé en réalité une solution satisfaisante.

4.4.2 Recuit simulé

Le recuit simulé est une métaheuristique d'optimisation qui s'inspire principalement du monde de la métallurgie. L'origine de cette approche se récapitule dans le principe de refroidissement d'un morceau de métal, l'état global de ce morceau est similaire à une

fonction objectif d'un problème d'optimisation. On peut résoudre un problème difficile à l'aide de ce principe d'équilibre thermodynamique.et on peut le résume la méta-heuristique recuit simulé par les étapes suivante :

1. Solution initiale: le choix d'une solution initial ne dépend d'aucune règle, il peut être que notre système soit choisi aléatoirement ou généré par une autre heuristique. Cependant, plus que la solution initial est bien choisi, la convergence vers un équilibre énergétique sera plus rapide.

2. Température initiale: le concepteur de la solution choisie et considère une température initiale d'une maniéré arbitraire, cette phase consiste à tester l'algorithme avec plusieurs valeurs de température pour obtenir les bonnes températures.

3. Modification élémentaire: cette phase permet au système d'évoluer en dégradant légèrement sa température. Une fois cette transformation est faite, la différance $\Delta E = (E_{nouvelle} - E_{ancienne})$ entre la nouvelle valeur et l'ancienne est calculée.

4. Critère d'évaluation : comme dans le domaine de la thermodynamique, le principe de recuit simulé consiste à montrer l'évolution du système. Pour le but d'évaluer la température du système, donc, si $(\Delta E < 0)$ alors on admet la variation, sinon $(\Delta E > 0)$on admet la variation avec une probabilité $e^{\left(\frac{\Delta E}{T}\right)}$. Le deuxième cas offre la possibilité à l'algorithme de ne pas rester bloqué dans un minimum local et poursuivre le chemin de parcourir l'espace de recherche. Le critère d'évaluation nous permet d'accepter quelquefois des solutions qui semblent mauvaises dés le début, mais qui peuvent par la suite converge vers un résultat acceptable.

Comme chaque technique, le recuit simulé a possédé des limites se résume essentiellement dans le choix des meilleurs paramètres dans la phase initiale, comme la température initiale ou la variation élémentaire.

4.4.2.1 Recuit simulé pour la sélection d'un schéma de fragmentation

Boukhalfa et al.[95]ont proposé d'exploité le principe du recuit simulé pour notre problématique, la sélection d'un meilleur schéma de fragmentation ,c'est pour cela, en définissant une fonction objectif pour évaluer le coût de chaque schéma. Le principe du recuit simulé développés par les chercheurs est étalé comme suit :

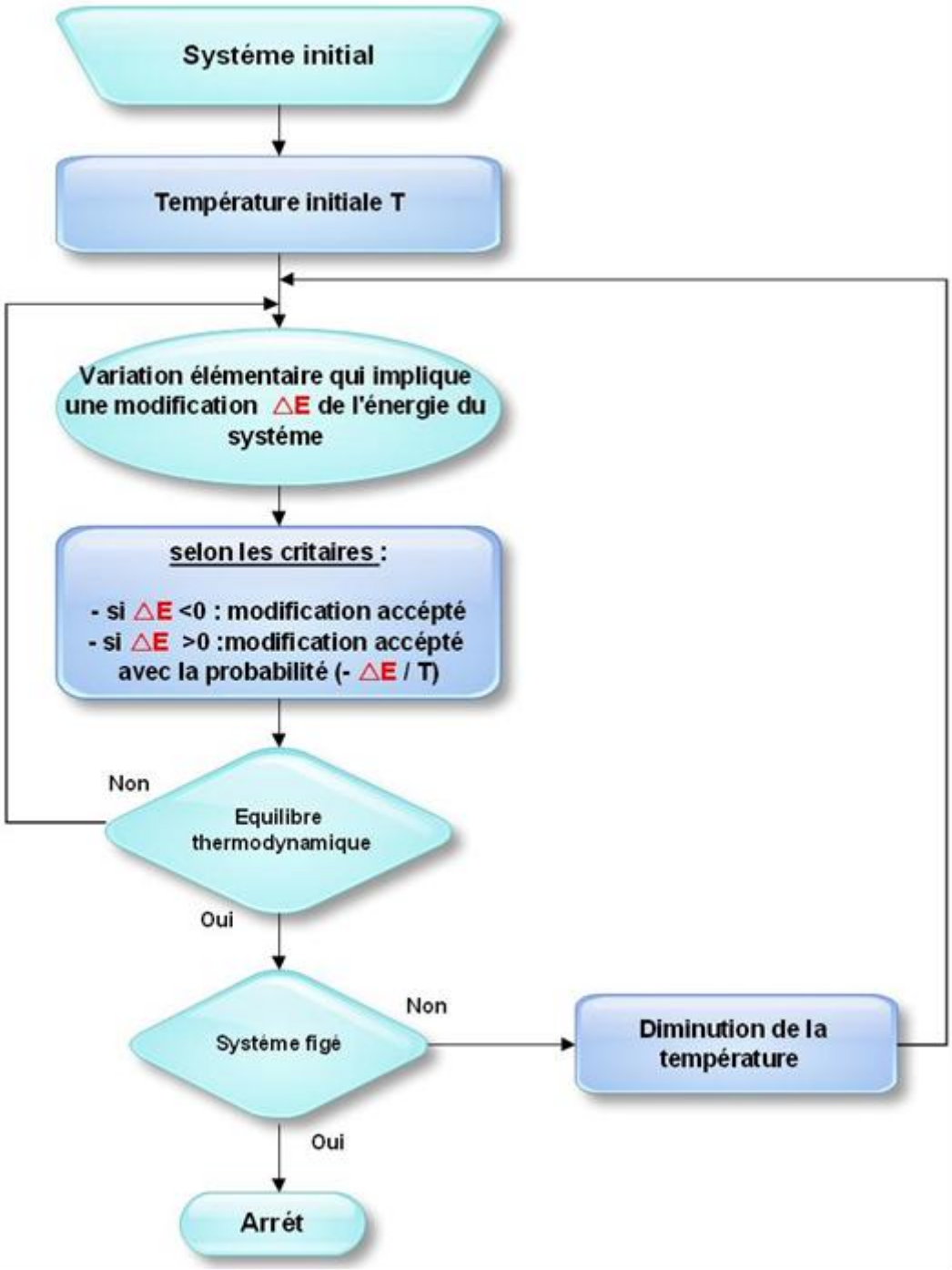

FIG. 4.8 Le processus d'un recuit simulé.

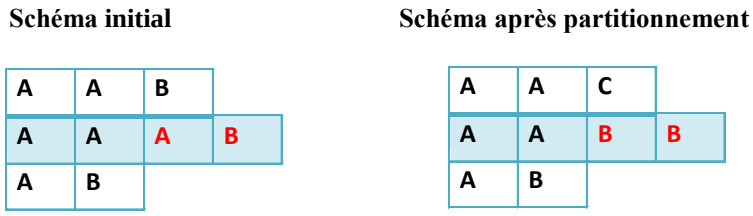

FIG. 4.9 La fonction split.

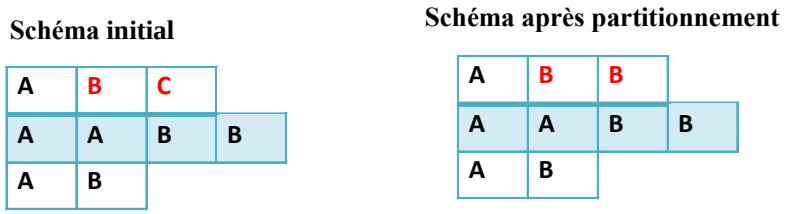

FIG. 4.10 La fonction merge.

– **La modification élémentaire : p**our atteint l'objective d'évoluer notre système, les concepteurs de cette métaheuristique offre la possibilité de travailler avec deux fonctions pour transformer l'état du système, la première est une fonction de partitionnement SPLIT , sa tache consiste à partitionner un groupe de sous domaines en deux sous groupes, cette fonction est montrée dans la Figure (**FIG.4.9**) et la seconde est une fonction de fusionnement MERGE, sa tache consiste à fusionner un sous-domaine avec un groupe, la fonction est illustrée dans la Figure (**FIG. 4.10**).

– **La fonction objective** : la fonction objective à maximiser utilisée dans le cas du recuit simulé est définie par :

$$F(SF) = \begin{cases} cout(SF) \times Pen(SF) \;\; Si \;\; Pen(SF) > 1 \; et \; N > W \\ count(SF) \qquad\quad sinon \end{cases}$$

$$Pen(SF) = 1 + \alpha(N - W)$$

Où N est le nombre de sous-schémas en étoile engendrés par SF est α le facteur de pénalité.

4.4.2.2 Avantages et limites du recuit simulé

Nous avons étalé dans cette partie un algorithme de recuit simulé permettant de progresser et d'améliorer une solution déjà obtenue, en exécutant un certain nombre de transformations sur cette solution.

Le recuit simulé apporte des qualités et des limites. Parmi ses qualités, on évoque sa capacité d'adaptation facile et son aptitude à trouver de bonnes solutions pour un très grand nombre de problèmes combinatoires.

En revanche des qualités de cette algorithme, les limites rencontré se récapitulent dans l'accroissement du temps d'exécution, qui peut considéré comme très élevé par rapport au autre technique, ainsi qu'au difficulté de choisir les paramètres de départ qui peuvent influencer la qualité de la solution obtenue. Le recuit simulé peut se coincé dans des optimums locaux et aussi loin de la solution optimale globale recherché.

4.4.3 Algorithme génétique

L'algorithme génétique proposé dans cette section rentre dans la famille des algorithmes évolutionnaires. ce type d'algorithmes utilisent un langage semblable à celui de la génétique naturelle, et son principe repose sur une population d'individus où chaque individu est représenté par un chromosome qui est composé d'un ensemble de gènes représentant le caractère congénital de l'individu [96].

L'objectif principal d'un algorithme évolutionnaire est de simuler les processus d'évolutions normales où le principal concept est la survie du plus fort : la nécessité de destruction pour les faibles. Dans l'évolution naturelle, la survie est effectuée parla reproduction, la nouvelle génération est générée à partir de deux parents (parfois plus de deux), elle englobe le matériel génétique des deux parents, si cette opération se terminé dans les bonne conditions, en effet, elle extraire seulement les meilleures caractéristiques de chaque parent [97]. Les autres individus qui obtiennent de mauvaises caractéristiques sont considérés comme faibles et sont inévitablement détruit.

Le principe dans les algorithmes évolutionnaires, c'est que ils sont utiliser une population d'individus, dans lequel chaque individu est montré sous le nom d'un

chromosome. Donc un chromosome apporte les caractéristiques de chaque individu, chaque caractéristique est appelée gène, la valeur d'un gène est nommée allèle.

Nous avons aussi, que pour chaque génération, les individus qui ont la concurrence comme caractère, sont choisis automatiquement pour reproduire une nouvelle génération à partir de la population initiale. Les individus qui ont les plus hautes possibilités de survie ont plus de chances de se reproduire [96]. La nouvelle génération est produite par l'opération de combinaison des parties de chacun du chromosome des deux parents par l'utilisation du processus de croisement.

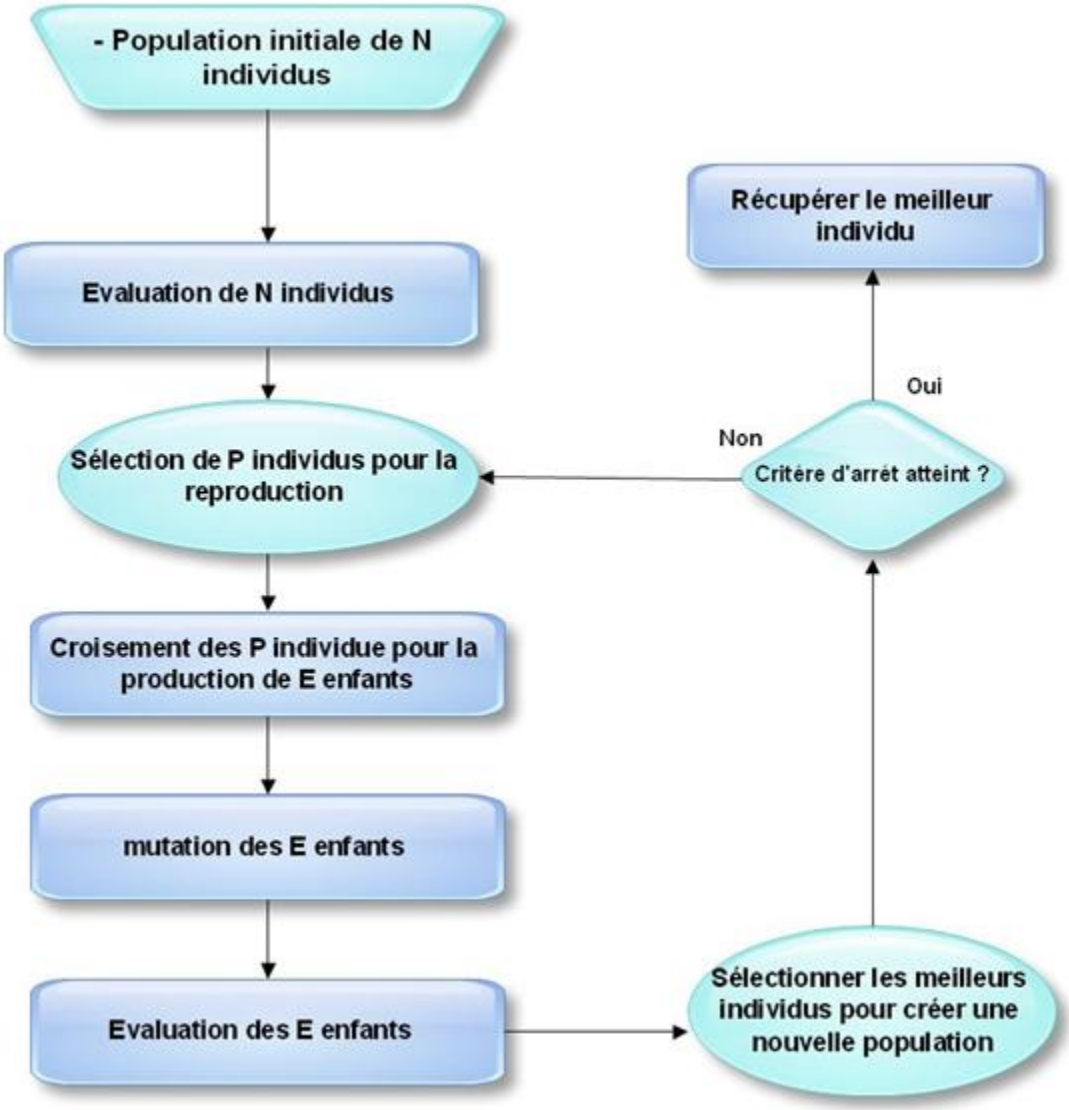

FIG. 4.11 Le processus d'un algorithme génétique.

Chaque chromosome dans la population peut également supporter indépendamment le processus de mutation, en effet le changement d'une partie du chromosome ayant déjà subit le processus de mutation.

La puissance de survie d'un individu dans la population est estimée par une fonction d'évaluation qui renvoie les objectifs et les contraintes du problème à résoudre.
Cette opération est continuer d'une maniéré itératif, et après chaque nouvelle génération produite, les individus subissent un processus de sélection, où certaines individus peuvent survivre et rester à la prochaine reproduction, les autres individus seront détruit.
Les chercheurs travaillé à trouver des solutions pour des problèmes selon un algorithme génétique, commence premièrement par la recherche d'une représentation chromosomique (génotype) appropriée pour chaque individu de la population. Il existe plusieurs codages dans la littérature pour un chromosome, parmi eux [98] :

- La représentation binaire où le chromosome représente comme une suite de bit (0 et 1).
- La représentation par des caractères : par contre au premier mode de représentation, dans ce deuxième cas le chromosome est composé de plusieurs caractères différents (chaîne de caractères).
- La représentation flottante : le chromosome est représenté par un nombre réel.
- La représentation sous forme d'un arbre : chaque individu représente comme un nœud d'un arbre, cette représentation augmente la visualité de l'effet des opérateurs génétiques (mutation et sélection). Cette représentation est essentiellement utilisée dans la résolution des problèmes dont les solutions n'ont pas une taille finie.

On peut résument les étapes d'un algorithme génétique comme suit :

1. Initialisation: presque dans la majorité des cas la population initiale est générée au hasard.

2. Sélection : grâce à une fonction d'évaluation qui détermine la qualité d'une solution proposée l'algorithme génétique déterminé les performances de chaque chromosome et il a mettre comme candidat pour le choisis a entré dans l'étape suivante.

3. Croisement : résulte à la reproduction avec l'exploite des meilleurs individus sélectionnée selon leurs qualités pour termine un croisement, où deux individus sont croisés pour produire deux nouveaux individus.

4. Mutation : Le principe de mutation a été mené pour éviter que l'AG ne soit bloquée ou coincée dans un optimum local. Après quelques générations, tous les chromosomes de la population soient identiques sans que nous atteint aune solution optimale. L'action d'introduire une chance de mutation permet d'assurer une diversité génétique au contexte de la population.

5. Évaluation : Les individus (enfants) générés au niveau des étapes de reproduction et mutation sont évalués afin de pouvoir comparer leur qualité.

6. Remplacement : dans cette étape, les meilleurs individus (enfants) sont sélectionnés pour accomplir un remplacement générationnel sur les anciens individus pour créer une nouvelle population.

7. Critère d'arrêt : cette séquence d'étape est cadencée par un critère d'arrêt, si ce dernier est atteint on arrête l'algorithme, sinon on recommence à partir de l'étape de sélection pour une nouvelle itération (Sélection → Croisement → Mutation → Évaluation). Le critère d'arrêtpeut prendre plusieurs images différentes comme un temps idéal ou un nombre de population définie etc.

4.4.3.1 AG pour la sélection d'un schéma de la fragmentation

Boukhalfa et al. [95] ont proposé d'exploiter les principes des algorithmes génétiques pour résoudre notre problème de la sélection d'un schéma de fragmentation optimale, où chaque chromosome représente un schéma de fragmentation et où la fonction objectif de ce problème est une fonction d'évaluation repose sur un modèle de coût utilisé pour évaluer le coût de chaque chromosome qui est considéré comme un schéma de fragmentation. Les chromosomes sélectionnés supportent un processus d'évolution, où les opérateurs génétiques sont utilisés pour faire évoluer les schémas initiaux et générer de nouveaux schémas. Le principe des opérateurs génétiques utilisés par les auteurs sont étalés comme suit :

– **Codage** : cette étape est peut considérer comme la plus délicate dans un algorithme génétique, pour dénouer un problème. Comme dans notre cas de la fragmentation horizontale, le problème consiste à la recherche d'un meilleur schéma de fragmentation. Pour trouver une solution à ce problème les chercheurs ont proposés de représenter un schéma de fragmentation par un chromosome de nombres entier.

On a comme modèle d'illustration le schéma de fragmentation représenté par le tableau multidimensionnel **4.1.(1)** est identifié par le chromosome **4.1.(2).**

La valeur de chaque gène doit être limitée par un intervalle de valeurs tel qu'elle est illustrée dans le tableau **4.1.(3)**

A	B	C	
A	A	C	D
A	B		

(1)

A	B	C	A	B	C	D	A	B

(2)

A...B	A...B	A...B	A...D	A...D	A...D	A...D	A...B	A...B

(3)

TAB 4.1 Tableau multidimensionnel

– Croisement : cette étape est primordiale dans un algorithme génétique, pour générer ou produire de nouveaux individus dans la population à partir d'un ensemble d'individus sélectionnés. Pour l'application à notre problème de sélection d'un schéma de fragmentation.

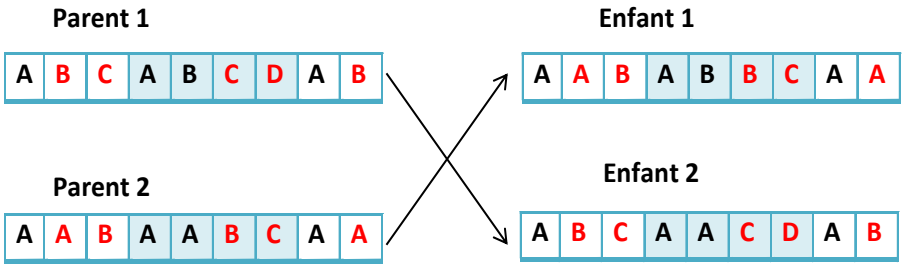

FIG. 4.12 L'opérateur de croisement.

Les chercheurs ont proposé d'utiliser un assistant de croisement multipoints comme montré dans la Figure (**FIG. 4.12**). La nouvelle difficulté limite les travaux des chercheurs après l'application du croisement est celui de la multi-instanciation des solutions, alors ils ont utilisé une fonction appelée renuméroter, son rôle consiste à éviter la multi-instanciation par la renumérotation des chromosomes, cette opération est montrée dans la Figure (**FIG. 4.13**).

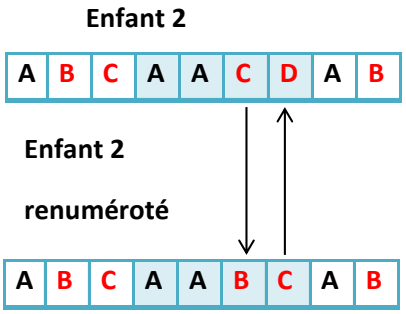

FIG. 4.13 La fonction de la renumérotation après croisement.

– **Mutation** : cette opération consiste à produire une diversité dans la population et éviter le problème des optimums locaux, l'assistant de mutation est permet de créer la diversité selon une probabilité de mutation. La Figure (**FIG. 4.14**). montre le principe de l'assistant mutation utilisé.

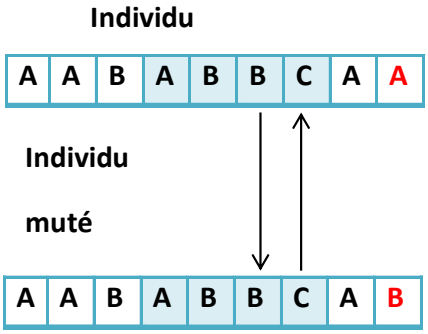

FIG. 4.14 L'opérateur de mutation.

– **Fonction d'évaluation** : le but a atteint par la recherche d'une solution quasi optimale pour le problème de fragmentation horizontale dérivée, est de minimiser le coût engendré par un schéma de fragmentation en étoile. La fonction d'évaluation F(SF) proposée a été sophistiquée sur un modèle de coût, le rôle de ce dernier consiste à évaluer un schéma de fragmentation, ainsi qu'une fonction de pénalité Pen(SF) est ajoutée, qui permet de pénaliser les schémas de fragmentation dépassant la contrainte de maintenance W.

$$F(SF) = \begin{cases} -cout(SF) \times Pen(SF) & Si\ Pen(SF) > 1\ et\ N > M \\ -cout(SF) & Sinon \end{cases}$$

$$Pen(SF) = 1 + \alpha(N - M)$$

où N est le nombre de sous-schémas en étoile générés par SF et α est le facteur de pénalité.

4.4.3.2 Avantages et limites

Le principal intérêt d'un algorithme génétique consiste a exploité le principe de parallélisme. Ce principe peut guider vers plusieurs solutions à la fois, grâce a sa progéniture multiple. Donc on peut considère que l'algorithme génétique est typique pour répondre aux problèmes de grande taille où l'évaluation de toutes les solutions possibles prend beaucoup de temps. Le deuxième intérêt d'un algorithme génétique, s'apparue dans son adaptabilité aux problèmes complexes où la fonction d'évaluation (Fitness) est pénible et très difficile à estimer et calculant, dans la majorité des cas, à des optimums locaux.

En revanche à tous ces avantages, les algorithmes génétiques dévoilent certaines limites, telle que l'impact d'une mauvaise représentation des gènes sur la qualité finale de la solution obtenue. Donc les paramètres parfaitement choisis au départ de processus, comme la taille de la population, les taux de mutation et de croisement déterminent la qualité de la solution.

Un autre problème peut aussi rencontrer, c'est la convergence prématurée, ce problème indique que si un individu, de la première génération, est observé comme l'élément le plus adapté, il peut aux étapes suivantes dominer le processus de la reproduction. Dans cette situation l'algorithme génétique se destine vers à un optimum local au lieu de finir à une solution global.

Conclusion

Nous avons exposé dans ce chapitre les notions fondamentales et la méthodologie exploitée pour les deux modes de fragmentation et on se focalise sur la tache de fragmenter horizontalement un entrepôt de données relationnel, ainsi que l'intérêt de cette sorte de fragmentation.

Nous distinguons que l'opération de choisir un schéma de fragmentation optimale est un choix décisif et reste une tache pénible pour l'administrateur de l'entrepôt de données AED. Ce problème est apparu à cause de la complexité du choix d'un meilleur schéma de fragmentation.

Enfin, nous avons étudié les différentes approches existantes, où nous avons cité pour chaque approche ses avantages et ses limites.

5

Réalisation et validation

Introduction

Dans ce chapitre, nous avons fait une comparaison théorique entre les algorithmes proposés pour la fragmentation horizontal et puis on montré la faisabilité de la démarche que nous avons proposé sur un banc d'essais APB-1 avec une charge requêtes, et puis une éclaircissement sur la méthode d'installation d'un entrepôt de données réel sur le SGBD ORACLE.

Nous allons exposer notre adaptation de l'algorithme génétique, concernant le codage, mutation et croisement, ainsi que les opérations primitives à utiliser, pour la fragmentation mixte. Le choix d'utilisation est la garantie d'aboutir à la fin du processus à une solution presque optimale et qui répond à toutes les exigences et contraintes établies par l'administrateur. De plus et pour confirmer notre approche une étude détaillée avec des expérimentations est envisagée dans la fin du chapitre. Nous considérons cette approche d'optimisation comme une alternative aux systèmes qui doivent gérer, manipuler et optimiser des données historiques.

5.1 Concept d'un méta heuristique

Un grand nombre des problèmes d'optimisation combinatoire sont caractérisé par sa facilité à définir et à formuler, en revanche sont très difficiles à résoudre. C'est pour ça que on considéré que la plupart de ces problèmes sont des problèmes NP-Complets, de ce fait, ne disposent pas actuellement de solution algorithmique efficace valable pour ces données entières. Face à ce genre primordial de problème, les auteurs ont essai de développer plusieurs méthodes de résolution. Ces derniers peuvent être subdivisés en deux grandes classes: la première classe c'est les méthodes exactes qui reposent sur la garantie de la complétude de la résolution et la deuxième classe consiste aux méthodes approchées qui sacrifie sa complétude pour gagner de l'efficacité.

La méthode exacte basée d'une manière générale à énumérer, d'une façon implicite, les solutions de l'espace de recherche. L'objective des méthodes exactes consiste aussi de trouver des meilleur solutions pour des problèmes de taille raisonnable. Le temps de calcul envisagé pour atteint une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent habituellement des difficultés devant les applications de grand taille.

La deuxième classe de méthodes qu'il s'appelé les méthodes approchées forment un choix très captivant pour résoudre les problèmes d'optimisation de taille importante si l'optimalité n'est pas décisive. Parmi les méthodes approchées on peut mentionner une catégorie de méthodes robustes et générales, sous le nom du métaheuristiques

Une heuristique est une méthode qui essai de trouver une bonne solution en tenant d'un temps de réponse acceptable sans assurer d'aucun optimalité. En effet, une heuristique est une méthode, créée pour répondre au certain problème d'optimisation, qui fournit une solution non forcément optimale quand on lui procure une instance de ce problème.

Une métaheuristique est définie de façon semblable, mais à un niveau d'abstraction plus haut. Ainsi les méta-heuristiques sont adéquates et applicables à une vaste classe de problèmes.

5.2 Comparaison de nos algorithmes

On peut subdiviser les méta-heuristiques selon le nombre de solutions traitées en même temps, en deux classes (**FIG. 5.1**) :

- Méthodes à base population.
- Méthodes basées solution.

Les méthodes repose sur une population traite un ensemble de solutions de l'intervalle de recherche simultanément et profitent l'évolution de cet ensemble pour atteint de solutions optimale. Parmi eux, les algorithmes génétiques et les algorithmes de colonies de fourmis. Les méthodes basées à une seule solution traitent, quant à elles, une seule solution à la fois. Elle porte le nom méthode à trajectoire, parce qu'elles décrivent une trajectoire de solutions au sein de l'espace de recherche, parmi ce genre de méthodes, le hillclimbing, le recuit simulé et la recherche tabou.

Ces méta-heuristiques nous permet actuellement d'offrir des solutions approchées pour répondre aux problèmes d'optimisation classiques de plus grande taille et pour de beaucoup d'applications qu'il était impossible de traiter auparavant.

Devant la complexité du problème de sélection d'un schéma de fragmentation que nous avons étalé, l'objective de recherche d'un meilleur schéma de fragmentation en utilisant une recherche exhaustive est quasi impossible.

L'ensemble d'algorithmes heuristiques déjà proposé pour sélectionner un schéma de fragmentation quasi optimal permettant de minimiser le temps d'exécution d'un ensemble de requêtes et respectant la contrainte de maintenance. On a déjà vue trois algorithmes, à savoir un algorithme génétique (AG), un algorithme de recuit simulé (RS) et un algorithme de Hill Climbing (HC). Les auteurs ont choisi ces algorithmes pour plusieurs considérations.

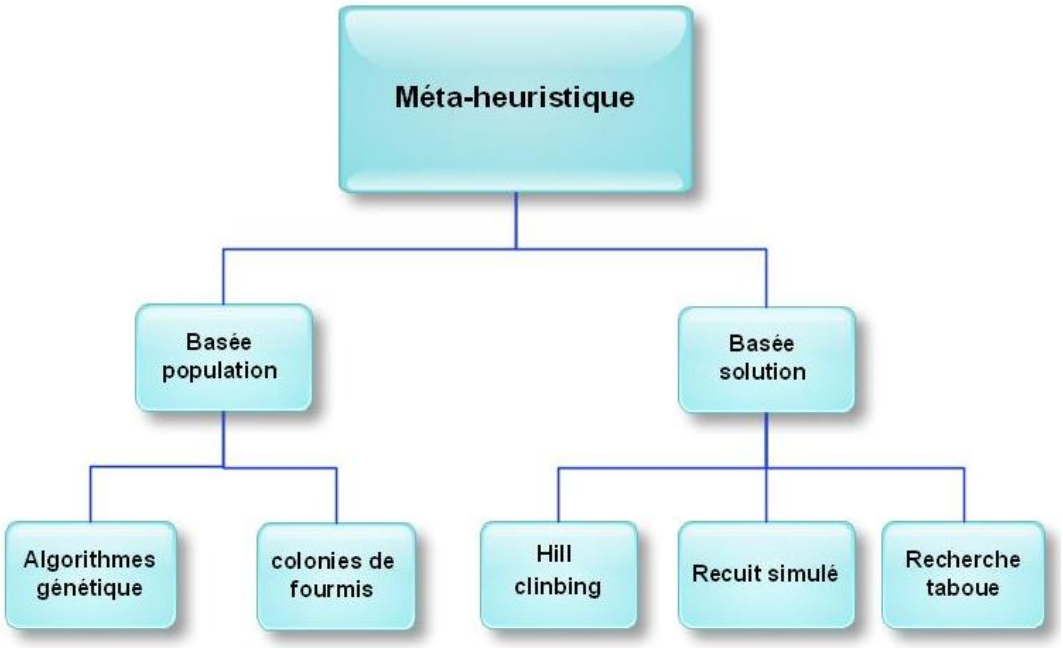

FIG. 5.1 Classification des principales méta-heuristiques.

La puissance du HC c'est sa rapidité comme algorithme qui permet aussi de donner une solution en utilisant un minimum de ressources (temps d'exécution, complexité des fonctions, etc.). Le HC a été utilisé pour résoudre le problème de FH dans les BDOO.

Les auteurs ont utilisé RS du fait qu'il a été utilisé pour résoudre le problème d'optimisation des requêtes de jointure. Il est souvent utilisé pour éviter les problèmes d'optimums locaux inhérents à HC. Certains études ont étalé que sous certaines conditions concernant la manière dont la prochaine solution est créée et la façon dont la température est décrémentée jusqu'à ce qu'elle soit proche de zéro, donc l'algorithme du RS converge inévitablement vers la solution optimum.

Les AGs ont été beaucoup utilisés dans la conception physique des bases de données. Telle que, le problème de sélection des vues matérialisées, le problème d'optimisation des requêtes de jointure et l'automatisation de la conception physique des bases de données parallèles. Les auteurs ont adopté les AGs pour deux raisons capitales :

- leurs apports à l'optimisation de l'opération de jointure.
- leur emploi par les optimiseurs de SGBD comme PostgreSQL16.

Pour évaluer la qualité d'un schéma de fragmentation, on commence d'abord par cette étude comparative théorique déjà expliqué à la section précédente et résumé dans le tableau suivant :

	Nombre de solution	**Base résolution**	**nature**	**paramètre**	**Temps D'exécution**	**Réduction De cout**
Hill clinbing	Basée solution	Le prb FH Dans BDOO	Alg stochastique	Pas de Paramètre	+	+++
Recuit simulé	Basée population	Le prb d'op des requêtes de jointure	Alg stochastique	Existence de paramètre	+++	+++++
Algorithmes génétique	Basée population	Le prb d'op -Requête de jointure - Sélection d vue matérialisé - L'auto conception physique	Alg stochastique	Existence de paramètre	+++++	++++

TAB 5.1 Résume les principales différences entre les 3méta heuristique.

5.3 Un nouveau modèle d'optimisation avec les AGs

L'idée de notre contribution consiste à combiner l'algorithme génétique déjà vue dans chapitre précèdent, avec au autre algorithme concerné par la sélection d'un schéma de fragmentation verticale, car cette dernière n'a pas célébré et n'a pas la même puissance que celui de la fragmentation horizontale, en effet, ce nouveau algorithme lui-même est considéré comme un algorithme génétique.

5.3.1 Processus génétique de fragmentation mixte

La fragmentation mixte est composée d'une fragmentation horizontale primaire concernant les dimensions, puis une fragmentation horizontale dérivée de la table des faits [26] et [88], enfin par une fragmentation verticale suivant les dimensions.

L'algorithme 1 nous illustre les étapes essentielles de cette approche pour traiter le problème d'optimisation dans l'entrepôt de données relationnel. La première étape permet d'extraire les données essentielles selon les fréquences d'utilisations et les prédicats de sélection à partir des requêtes. Les deux dernières étapes sont indépendantes.

Algorithme 1 : Algorithme mixte proposé.

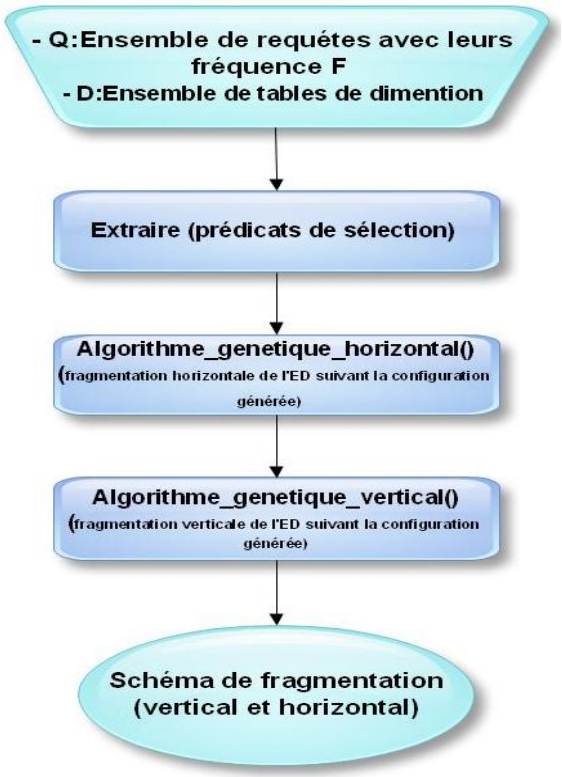

FIG. 5.2 Algorithme mixte proposé.

5.3.1.1 Codage d'individu

L'origine de fragmentation verticale repose sur le principe d'affinité, par conséquent, la plupart des algorithmes de fragmentation verticale demande de connaitre l'ensemble de requêtes les plus fréquentes. A partir de ces requêtes, et avec l'exploite d'une table de dimension candidate pour la tache de fragmentation, nous avons criée une matrice d'usage des attributs. Soitn_i le nombre d'attributs non clés de la table de dimensionD_i. On considéré que chaque individu comme un schéma de fragmentation est donc montré par un tableau d'entiers de taille $N = \sum_{k=1}^{D} n_i$ (D représente le nombre de dimensions) (**FIG. 5.3**). S'il existe des cellules avec la même valeur, ce qu'il indique que ces attributs représentent le même fragment vertical. Si on trouve dans un tableau de codage que toutes les cellules ont remplis par la même valeur, ce qu'il indique que cette table de dimension ne sera pas partitionnée.

Exemple 5.1 : si on suppose, qu'il existe un individu qui expose une dimension ayant une représentation comme suit:[4,6,2,6,2,1,1,4]. Cette représentation produira trois fragments verticaux de cette dimension : les attributs $A_1 et\ A_3$forment le premier fragment, A_2 *et* A_4forment le deuxième fragment, tandis que$A_5 et\ A_6$forment le troisième fragment et

A_0 *et* A_7 forment le dernier fragment (il ne faut pas oublier d'ajouter la clé primaire de la relation dans chaque fragment).

Exemple 5.2 : la figure suivante montre que le chromosome présentant la dimension Custlevel désigne la construction de deux fragments verticaux : FV_1 (Key, X_A, X_B)et FV_2 (Key, X_C, X_D).

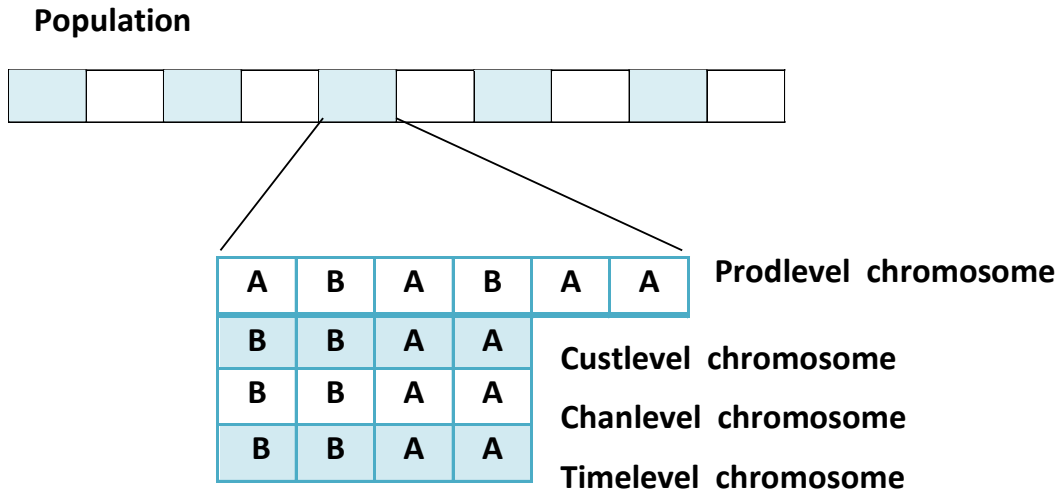

FIG. 5.3 Exemple de codage des individus pour un chromosome vertical.

L'exemple précèdent montré à la figure (**FIG. 5.3**) représente un schéma de fragmentation verticale concernant un exemple de codage des individus pour un chromosome vertical.

5.3.1.2 Sélection des individus

Nous avons utilisé la méthode de la roulette pour sélectionner nos individus parents comme pour la fragmentation horizontale.

5.3.1.3 Croisement

Cette opération consiste a appliqué l'action de croisement à une paire d'individus (les parents) d'un certain rapport de la population - avec une probabilité *c P*, généralement autour de 0.6 - pour généré de nouveaux (chromosomes), et on décide si on continuer à croiser ou non suivant une variable d'échantillonnage aléatoire et examinée avec la probabilitéP_c. Le caractère de croisement appliqué est celui d'un seul point pour la génération de la population suivante (Algorithme 2).

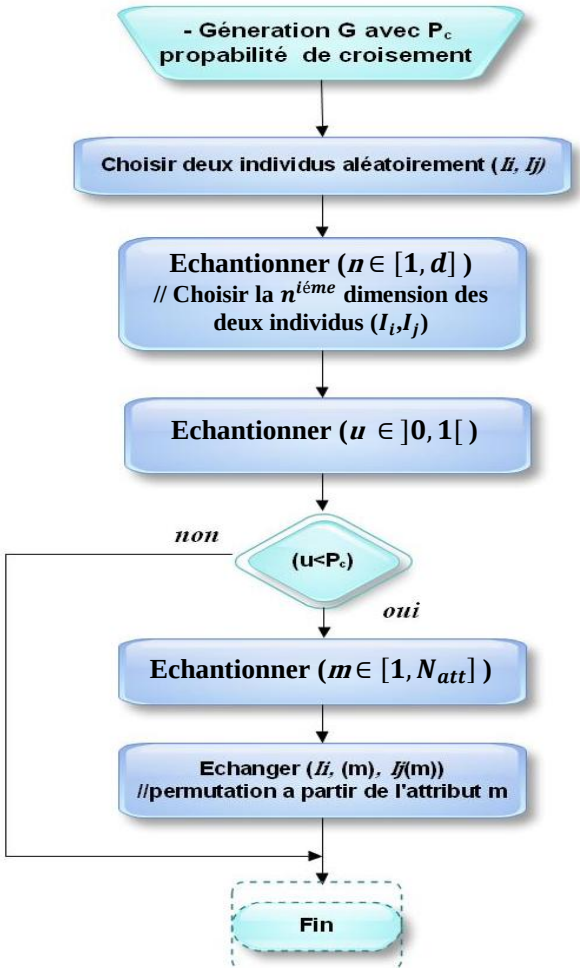

FIG. 5.4 Algorithme de croisement pour la fragmentation verticale.

La figure (**FIG.5.5**) montre un exemple de croisement pour un entrepôt de données de trois dimensions (produit,client,stockage)

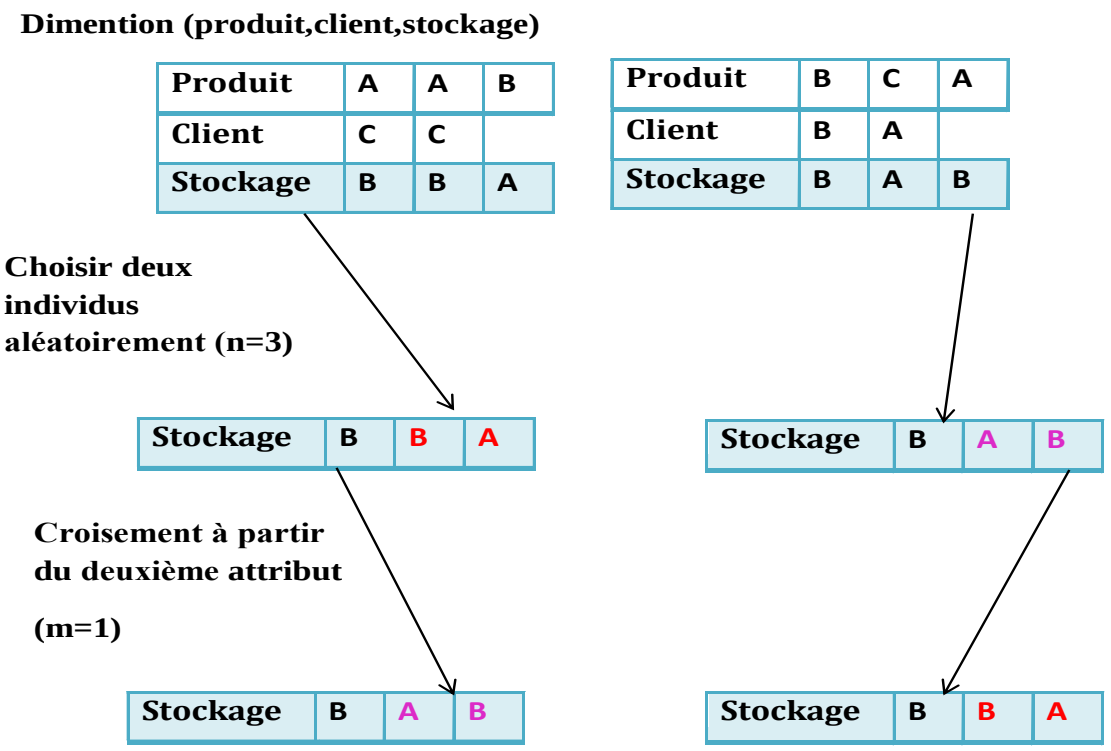

FIG. 5.5 Croisement suivant la dimension stockage.

Dans cet exemple (**FIG. 5.5**), la dimension stockage est choisie suivant une valeur prise hasardeusement, puis on applique la même sorte de croisement employé pour la fragmentation horizontale.

5.3.1.4 Mutation

L'algorithme 3 nous illustre cette étape qui est presque la même que celle de la fragmentation horizontale avec une légère adaptation

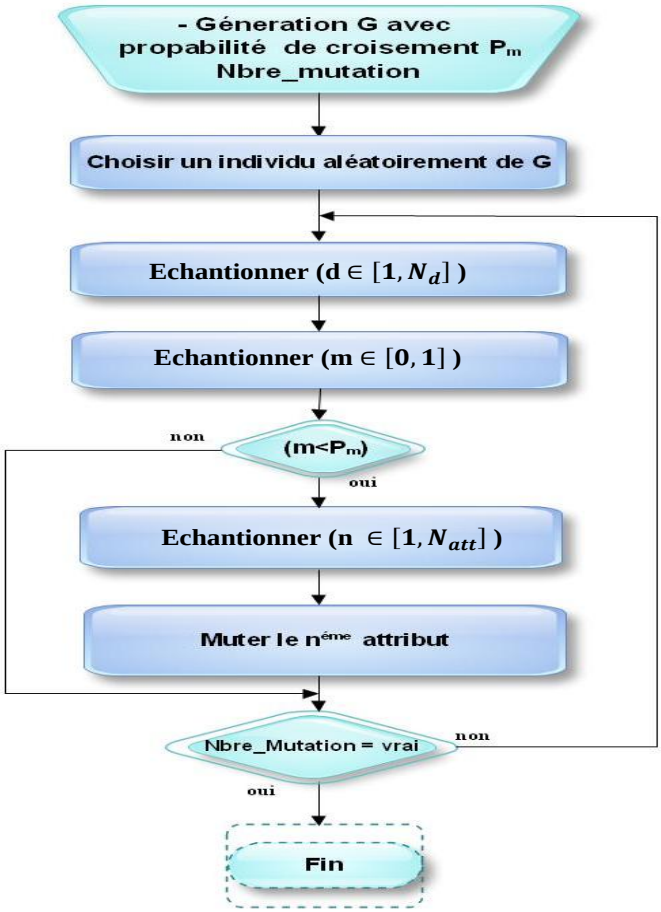

FIG. 5.6 Exemple de mutation pour une dimension (Produit) choisie suivant une valeur donnée aléatoirement.

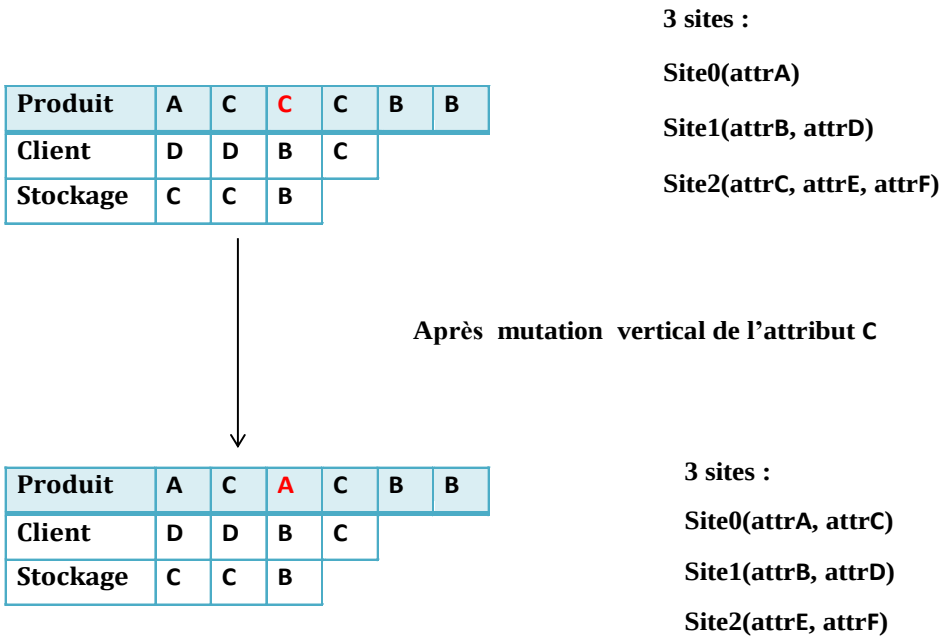

FIG. 5.7 Exemple de mutation verticale.

5.3.1.5 Processus d'amélioration

Pour l'objective d'effectuer cette amélioration, on suit les deux sortes de l'algorithme mixte, donc on entamer le processus par une population variée et aléatoire qui n'appartient pas des informations de l'entrepôt de données et ne possède aucune signe qui lui permet d'avoir une idée sur la solution initiale, ceci donne plus de chances de mieux limiter la recherche et de se rapprocher plus vers la solution optimale.

L'idée de cette approche est de voir chaque population d'une manière itérative, dans l'objective de diminuer le plus possible le coût consommé dans l'évaluation des requêtes. On remarque que notre AG peut trouver des solutions ne respecte pas la contrainte d'espace. Et afin de progresser et d'améliorer et essai de corrigé ces solutions, une valeur de pénalité est introduite à la fonction objective (à minimiser ou maximiser) comme une partie de la fonction de fitness [99].

On Suppose que nous considérons le problème suivant : Maximiser $F(x)$sous la contrainte $C_x \leq C$où Cx est un coût généré par la solution x et C montre le seuil de coût à ne pas dépasser. La fonction de pénalité (notée $Pen\square x\square$) a charge de pénaliser les solutions non respectable de la contrainte, en réduisant la valeur de la fonction à maximiser suivant le degré de violation de la contrainte.

Il existe trois sortes de la fonction de pénalité, logarithmique, linéaire et exponentielle :

- Pénalité logarithmique : $Pen(x) = \log_2\left(1 + \alpha\left(C_x - C\right)\right)$ ou α représente le facteur de pénalité.
- Pénalité linéaire :$Pen(x) = 1 + \alpha\left(C_x - C\right)$.
- Pénalité exponentielle : $Pen(x) = (1 + \alpha(C_x - C)^2)$.

L'exploite de la fonction de pénalité dans la fonction $F(x)$permet dechanger le problème à un problème d'optimisation sans contraintes d'une fonction $F'(x)$définie à partir des fonctions $F(x)$et $Pen\,(x)$Cela peut être accompli en utilisant trois façon, soustraction, division et division-soustraction qui sont définis comme suit.

1. Mode Soustraction :

$$F'(x) = \begin{cases} F(x) - Pen(x)\ si\ F(x) - Pen(x)\ \geq 0 \\ \\ 0 \quad sinon. \end{cases}$$

2. Mode division :

$$F'(x) = \begin{cases} \dfrac{F(x)}{Pen(x)}\ si\ Pen\ > 1 \\ \quad F(x) \quad sinon \end{cases}$$

3. Mode soustraction-division :

$$F(SF) = \begin{cases} \quad F(x) - Pen(x) \quad si\ F(x) > Pen(x) \\ \dfrac{F(x)}{Pen(x)}\ si\ F(x) \leq Pen(x)\ et\ Pen(x) > 1 \\ F(x)\ \ si\ F(x) \leq Pen(x)\ et\ Pen(x) \leq 1 \end{cases}$$

On exploite ces opérations pour atteint les meilleur solutions, on procède ainsi à:

Écarter et négliger les solutions non valables et ne préserver que les solutions admissibles. Cette approche peut être moins attirante dans le cas où le passage entredeux solutions admissibles passerait inévitablement par une solution non admissible. On rappel aussi qu'il est nécessaire dans ce cas d'utiliser des opérateurs génétiques qui prend en charge le but de protègent l'admissibilité des solutions puisque on sait que l'emploi d'un opérateur génétique sur une solution admissible génère toujours une solution admissible, et pour pénaliser les solutions non admissibles en ajoutant une fonction de pénalité comme partie supplémentaire de la fonction d'évaluation.

5.3.1.6 Caractéristiques de notre algorithme

Nous avons que pour chaque technique parmi les techniques d'optimisation existe, plusieurs d'algorithmes est disponible.et pour sélectionner un schéma de FH optimale, plusieurs algorithmes de sélection ont été offerts [88].

Suivant l'algorithme mixte proposé (algorithme 1), et pour effectuer cette fragmentation (verticale), l'administrateur doit identifier les paramètres de configuration liés premièrement à l'algorithme génétique, et séparés au processus de sélection, donc les attributs de dimension engagé à ce processus, nommés les attributs de fragmentation **(TAB. 5.2).**

En conséquence, tout algorithme est caractérisé par quelque paramètre qu'il faut la configurer. Les paramètres les plus utilisés par ces algorithmes sont résumé dans Le tableau ci dessus.

On sait aussi qu'il n'existe pas de paramètres qui sont similaire et ajustés à la résolution de tous les problèmes qui nous rencontrons dans un A.G. Mais, certaines valeurs sont fréquemment utilisées peuvent être de meilleur points de bon commencement d'une recherche de solution selon un A.G.

- Taille de la population: entre 30 et 50 individus.
- Taux de crossover : entre 70% et 95%.
- Taux de mutation : entre 0,5% et 1%.

Algorithme	Paramètres
Algorithme génétique	- Nombre d'individus - Nombre de générations - Taux de croisement - Taux de mutation
Algorithme de fragmentation verticale	- Les attributs de fragmentation (à partir de la clause Select). - Les fréquences d'utilisations des requêtes OLAP.

TAB 5.2 Principaux paramètres des différents algorithmes.

5.4 Evaluation globale de l'approche sur un benchmark

Pour valider nos contributions, nous avons obtenue des expériences qui nous ont aidés de comparer et d'évaluer les techniques de fragmentation étudié. Ces expériences permet d'exploiter un entrepôt de données produit par un banc d'essai (Data warehouse Benchmark APB-1 release II [91].

5.4.1 Banc d'essai Benchmark

L'objective d'utilisation de bancs d'essais, c'est l'évaluation des performances des SGBDs,et plus particulièrement pour testé l'efficacité des techniques d'optimisation au niveau de ces performances, Dans le cadre des entrepôts de données, nous avons exploité le célèbre banc d'essais Benchmark APB-1 release II [91].

Le schéma en étoile généré à partir de ce banc d'essais est composé (**FIG. 5.8**) de quatre tables de dimensions *Prodleve*l, *Custlevel*, *Timelevel*et *Chanlevel* et une table de faits Actvars, telles qu'elles sont représenté dans la figure, les caractéristiques de chaque table sont illustrées par le tableau (**TAB 5.1**) Le détail de ces tables est représenté dans l'annexe A.

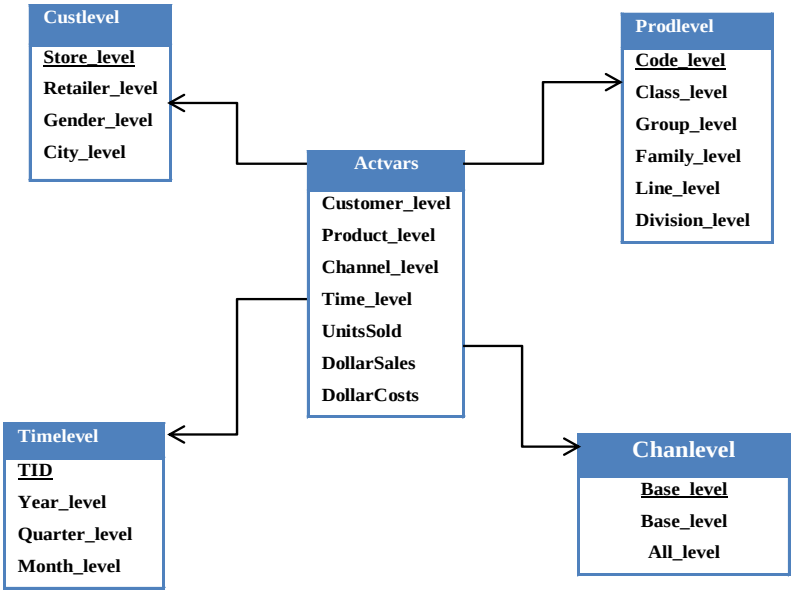

FIG. 5.8 Schéma de l'entrepôt utilisé APB benchmark.

Table	Nombre de Tuples	Taille du Tuple
Actvars	24 786 000	74
Chanlevel	9	24
Custlevel	900	24
Prodlevel	9 000	72
Timelevel	24	36

TAB 5.3 Caractéristiques des tables.

5.4.2 Implémentation

Pour réalisé la démarche d'évaluation de nos techniques de fragmentation génétique, nous avons exploité un moteur génétique relatif à la fragmentation verticale développé par les auteurs ….et son principe illustré comme suit :(**FIG. 5.9**) [88].

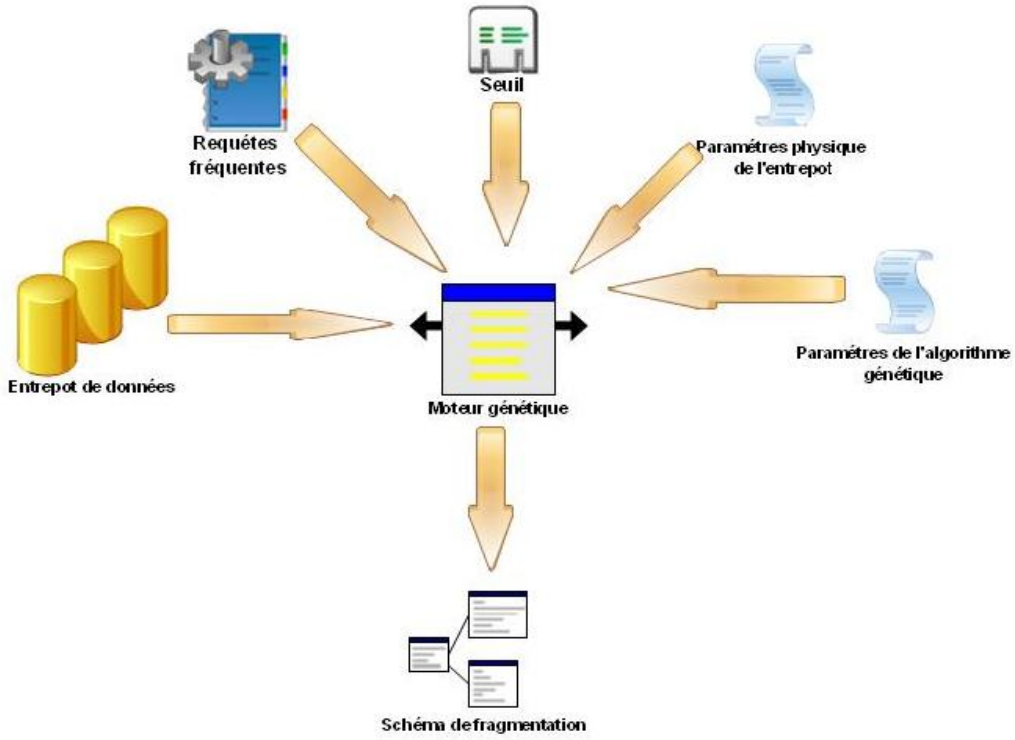

FIG. 5.9 Architecture du moteur générique

5.4.2.1 Pseudo code

Pour proposer un nouveau modèle de coût générique il faut que prenant en considération tous les types de requêtes existe et son adaptation à tous les SGBDs aussi, c'est pour ça que cette tâche est très difficile.

Les travaux concernés à la proposition de modèles de coût sont largement offert, pour évaluer le coût d'exécution d'une requête. On subdivise les modèles existe dans la littérature en deux classes :

- modèles repose sur une fonction mathématique ad-hoc [95].
- modèles repose sur l'optimiseur de requêtes. Le coût dans ce genre d'optimiseur est évalué par l'utilisation des formules avec un certain nombre de paramètres et de statistiques réunies sur la base de données.

Le premier modèle de coût utilisé au niveau de la fragmentation horizontale permet d'évaluer la qualité des différents schémas de fragmentation engendré ou trouvés par l'algorithme de fragmentation. Sa tache principale consiste à estimer le nombre d'entrées/sorties indispensable pour terminé l'exécution d'une requête sur un schéma fragmenté ou non (**FIG. 5.10**).

Le deuxième modèle de coût utilisé lors de la fragmentation verticale est considéré comme une adaptation du premier model, il permet d'évaluer les différentes localisation ou configurations des attributs engendrés par l'algorithme de sélection. On ne prend pas en considération le temps CPU, puisque on le négligé par rapport au temps nécessaire pour accomplir les E/S.

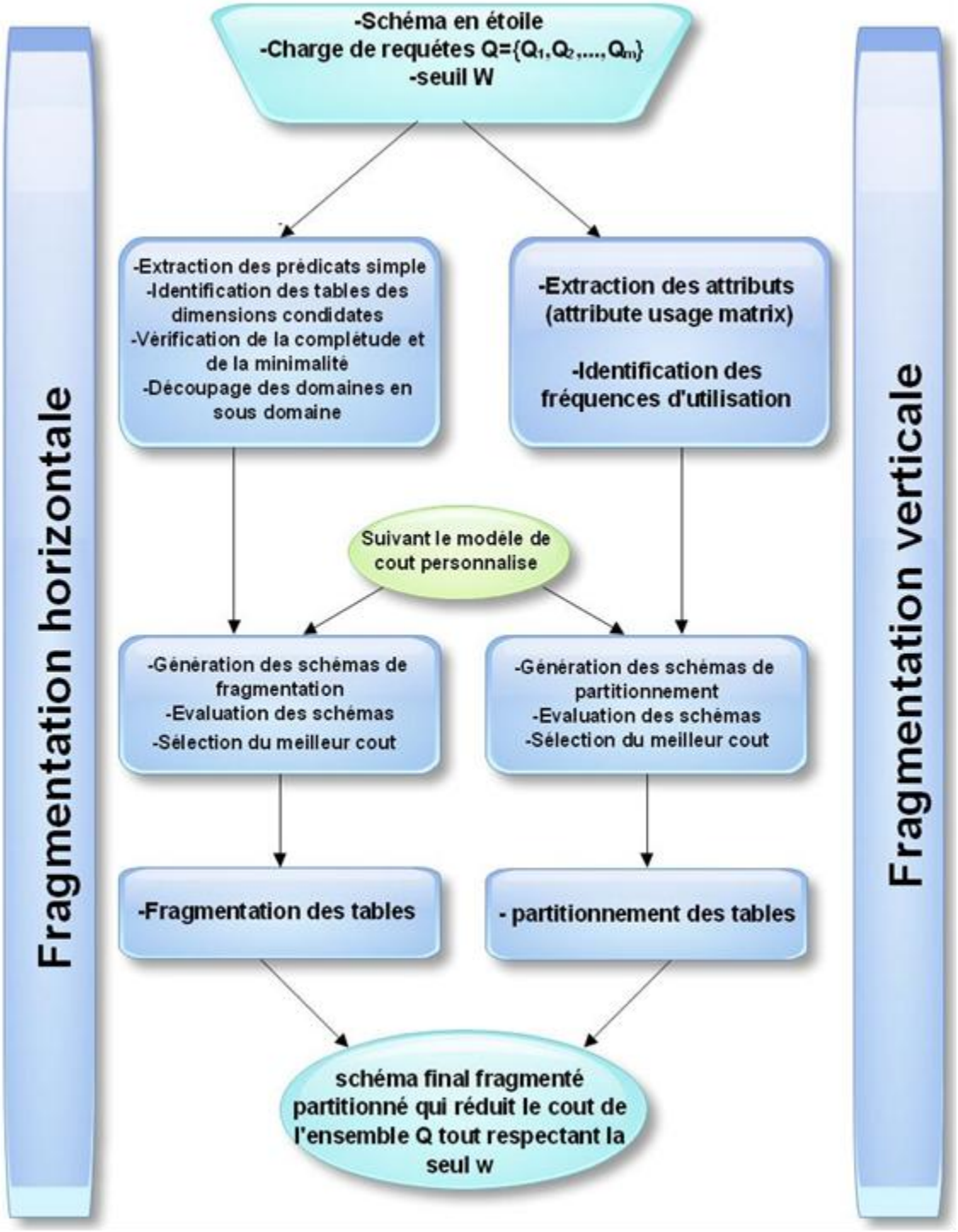

FIG. 5.10 Démarche de fragmentation proposée.

Dans le contexte de nos contributions, nous avons offert un modèle de coût dépendant à la première classe pour être libre du SGBD utilisé et atteint vite le coût de chaque schéma de fragmentation sans passer par l'optimiseur.

5.4.2.2 Chargement de l'entrepôt

Un fichier générateur APB.exe [91] est fournit avec le banc d'essais ABP-1, ce fichier exécutable permet de générer les fichiers de données servant à remplir l'entrepôt final. Un autre fichier est fournit aussi avec ce benchmark ABP-1 contenant les scripts de création des tables constituant l'entrepôt. Nous avons modifié ces scripts pour créer les tables de dimension *ProdLevel*, *TimeLevel*, *CustLevel* et *ChanLevel*et la table des faits *Actvars*. Le remplissage et le peuplement de l'entrepôt se fait à l'aide de l'outil *SQLLoader* fourni avec Oracle10g en utilisant des fichiers de contrôles que nous avons créés.

Les fichiers de contrôles spécifient les formats des fichiers de données et des tables dans lesquelles ces fichiers seront chargés. La figure (**FIG. 5.11**) montre l'opération de chargement de l'entrepôt en utilisant l'outil de génération APB.EXE fourni avec le banc d'essais APB-1 [91] et l'outil SQL Loader fourni avec Oracle

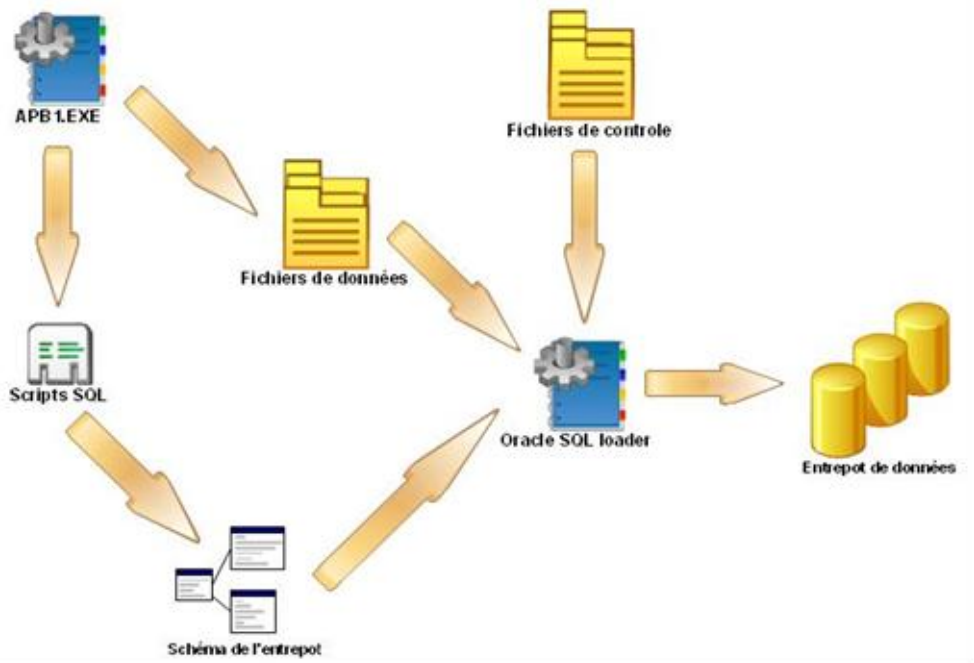

FIG. 5.11 La construction de l'entrepôt de données APB-1 sous ORACLE.

5.4.2.3 Types de requêtes prises en compte

Les requêtes prises en compte dans ces expériences et lors de l'élaboration de notre modèle de coût sont des requêtes de jointure en étoile ayant la structure suivante :

```
SELECT [SGA], FC1(AA), FC2(AA),..., FCn(AA)
     FROM F, D1, D2,...Dk
     WHERE PJOIN AND PSEL
[GROUP BY GA]
```

Où :

SGA : représente l'ensemble des attributs revenus dans la réponse à la requête (il peut l'initiale vide pour les requêtes retournant un calcul). Ces attributs peuvent être des attributs de dimension et/ou de faits.

FC1, ***FC2***,…, ***FCn***: représente les fonctions de calculs tels que le MIN, MAX, COUNT, SUM, AVG.

AA : représente l'ensemble des attributs d'agrégation, ces attributs peut être sont des mesures définies dans la table des faits.

k: le nombre de tables de dimension utilisées exploité comme source par la requête.

PJOIN : considère comme un ensemble de prédicats de jointure entre les tables de dimension et la table des faits sous forme de conjonction. Un prédicat de jointure (équi-jointure) a la forme suivante :$F.fk_i = D_i.k_i$ où $fk_i,.k_i$représentent respectivement la clé étrangère et la cléprimaire de la table de dimensionD_i.

PSEL : l'ensemble de prédicats de l'opération de sélection relier sous forme de conjonction ou chaque conjonction est constitué de plusieurs prédicats définis sur la même table.

GA : l'ensemble des attributs d'agrégation ou groupement. Ces attributs sont des attributs de faits et/ou de dimension.

Les requêtes que nous avons utilisées pour notre modèle de coût sont représentées dans l'annexe. La difficulté de ce type des requêtes, c'est localise dans le schéma en étoile parce qu'elles sont plusieurs jointures (entre les tables de dimensions et la table des faits). Dans cette situation, Il faut diviser la requête suivant les jointures. D'une façon que le nombre de jointures possibles est de l'ordre de *N*!et*N* étant le nombre de tables de dimensions à joindre. On peut même formaliser ce problème d'ordonnancement, on suppose qu'on a une requête de jointure en étoile *Q* engageant m jointures entre la table des faits *F* et *m* tables de dimension *D1, D2,…, Dm.*

On peut définir le problème de sélection d'un ordre de jointure, comme, la possibilité de trouver une permutation de l'ensemble $\{1,2,\ldots\ldots,m\}$dont le coût d'exécution de la requête suit l'ordre de jointure engendré par cette action de permutation soit minimal.

On peut considère aussi que chaque élément *i* de cet ensemble indique l'ordre de représentation ou d'apparition de la table de dimension *Di* dans l'enchaînement de jointures. Pour traiter ce problème, une démarche de recherche exhaustive peut être appliquée. Elle permet de détecter tous les ordres et chemins possibles, puis elle suit une procédure commence par l'évaluation, ensuite le chois du meilleur ordre. Enfin et comme résultat en trouve que pour des valeurs importantes de *m,* le problème de sélection d'un
ordre de jointure devient plus en plus compliqué, donc on récapitule qu'aucun algorithme ne pourra donner la solution optimale en un temps fini.

5.4.3 Evaluation

5.4.3.1 Sans pénalisation

Nous avons subdivisé notre réalisation concernant ces expérimentations en deux étapes :

La première étape permet de fragmenter horizontalement l'entrepôt de données par l'utilisation de l'algorithme génétique. Cependant, la deuxième étape consiste à partitionner les fragments horizontaux verticalement en appliquant l'algorithme 1.
Tout les solutions proposé sont implémentée sous Oracle 10g. Pour le calcul et l'évaluation des temps d'exécution de requêtes, nous avons le mesuré par l'outil Aqua data studio (ADS).pour abouti a une bonne évaluation, il faut qu'il bien choisit nos paramètres, qu'il existe parmi eux le nombre de requête, nous avons fixé un ensemble de 12 requêtes (voir annexe), ayant la forme précédemment citée. Où chaque requête est exécutée sur l'entrepôt non fragmenté et son temps d'exécution est calculé avec ADS (**FIG. 5.12**).

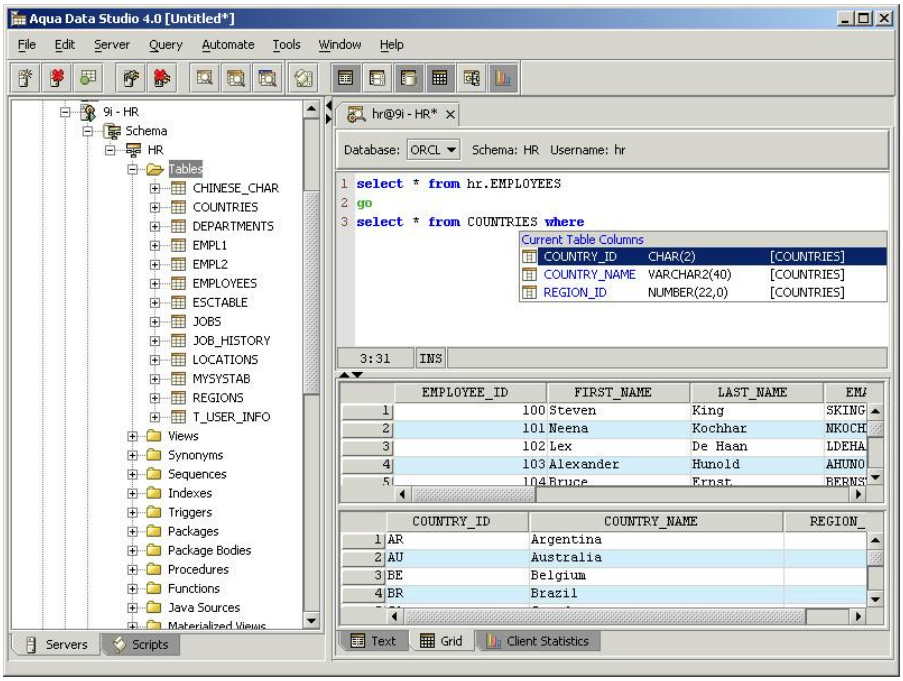

FIG. 5.12 Fenêtre d'exécution d'une requête sous Aqua data studio.

Intérêt de la fragmentation horizontale

Nous avons fait plusieurs expériences. nous avons voulu voir l'intérêt de la fragmentation horizontale, en effet, nous commençant par l'étude de temps d'exécution par rapport à un ordre de requête, L'axe des abscisses étal les différentes requêtes et l'axe des ordonnées représente le temps d'exécution de chaque requête mesurent en milliseconde. La figure (**FIG. 5.13**) illustre que la fragmentation horizontale est très intéressante pour cette charge de requêtes, beaucoup plus au niveau des quatre premières. Les résultats obtenus confirment les travaux exposés dans [26] et avec une baisse de 23 % du coût total.

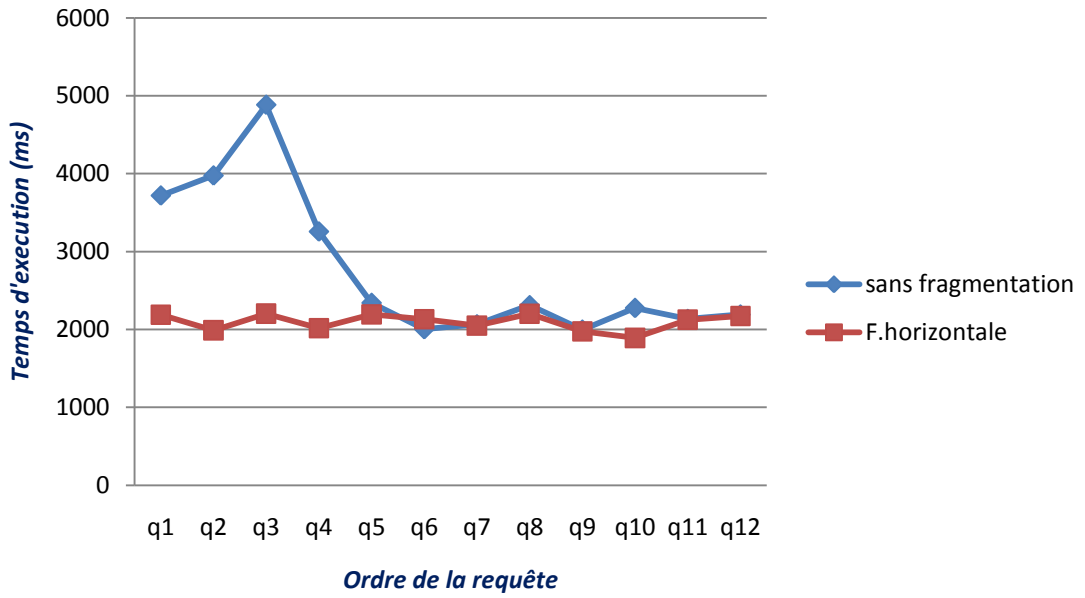

FIG. 5.13 Temps d'exécution des requêtes selon deux mode de schémas (fragmenté et non fragmenté).

Ces résultats sont obtenus suivant le schéma de fragmentation horizontale généré :

Dimension	Nombre de partition
Timelevel	7
Prodlevel	48
Custlevel	2
Chanlevel	3

TAB 5.4 Nombre de partition.

Impact de la fragmentation mixte

Dans la deuxième expérience nous voulons savoir le rapport et l'expertise qu'apporté la fragmentation mixte (horizontale puis verticale) devant la fragmentation horizontale. Pour ce détail, nous avons exécuté des requêtes semblables sur un schéma fragmenté horizontalement

Timelevel	0	1	2		
Prodlevel	1	0	2	2	4
Chanlevel	1	1			
Custlevel	1	1			

TAB 5.5 Configuration de schéma fragmenté.

puis sur un schéma mixte fragmenté horizontalement et verticalement selon la configuration :

On a comme modèle, suivant la dimension de cette configuration :

Timelevel (Tid, Year_level,Month_level) s'interprète par la génération de 3 partitions verticales PV1 (Tid, Year_level), PV2(Tid, Month_level) et PV3 (Tid, Quarter_level)). Si on observe le coût total de requêtes, on remarque que la réduction généré par la fragmentation mixte est significative. La figure (**FIG. 5.14**) illustre l'intérêt de la fragmentation mixte apportés aux les entrepôts de données, avec un gain est de 3,8 %.

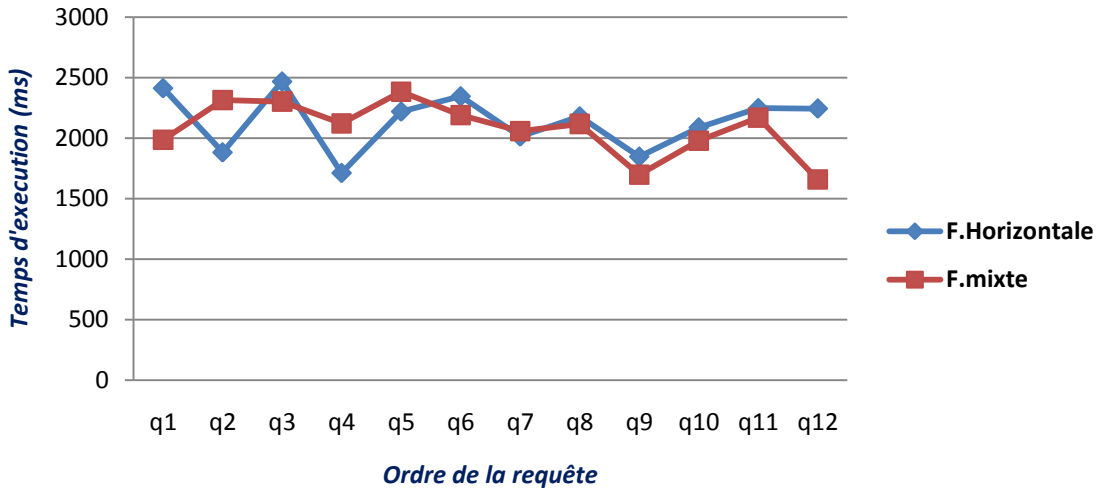

FIG. 5.14 La différence entre la fragmentation mixte proposée et horizontale.

Impact du nombre de fragments verticaux

Notre troisième expérience consiste à la variation du nombre de fragments verticaux sur l'entrepôt. On a examiné trois configurations différentes d'entrepôts: avec 7, 8,9 et 10 fragments verticaux.

On commence par l'exécution de notre algorithme de fragmentation mixte (fragmentation horizontale suivie par une verticale), suivant la contrainte de seuil (dans ce cas c'est le nombre de fragments verticaux). Toutes nos requêtes tests sont exécutées à chaque configuration. Les résultats sont montrés dans les figures (**FIG. 5.15 et FIG. 5.16**).

Les premiers résultats que nous pouvons extraire, c'est que la performance dans les entrepôts de données dépend fortement du nombre de fragments ; par exemple dans notre expérience on bénéfice d'un gain similaire a une réduction de 18,21%. De ce fait, on peut conclure que l'augmentation du nombre de fragments verticaux, provoque une augmentation des opérations de jointures. Ce que résulte l'augmentation du temps d'exécution.

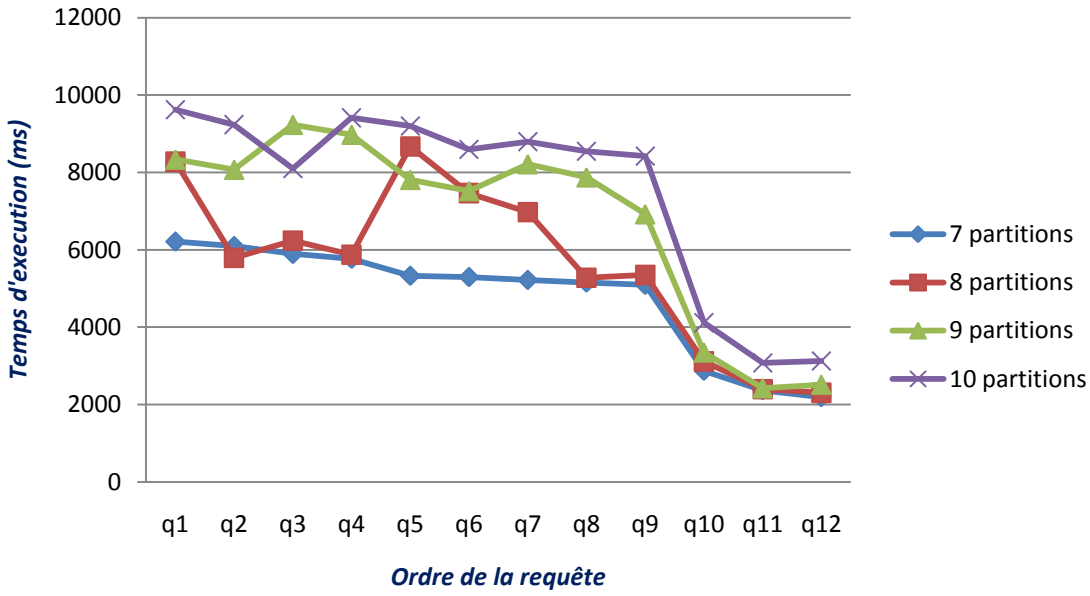

FIG. 5.15 L'impact du nombre de fragments verticaux sur la performance

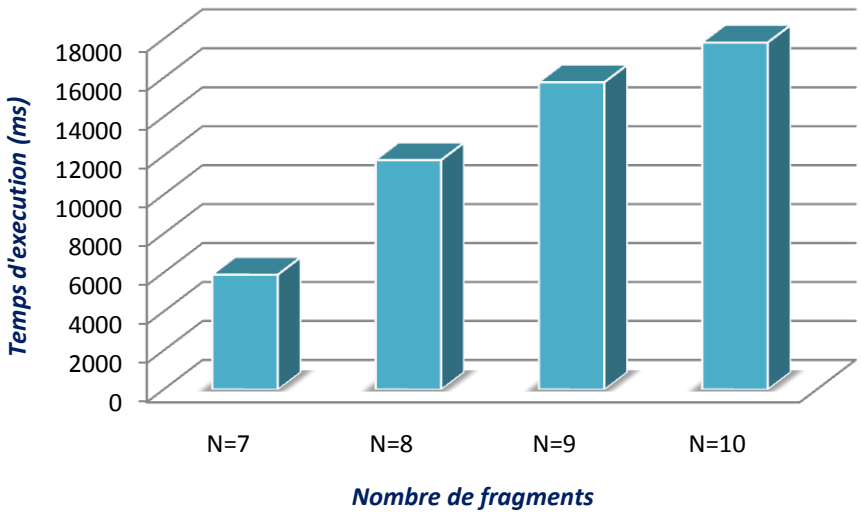

FIG. 5.16 Cout total d'exécutions de l'ensemble suivant le nombre de partitions.

5.4.3.2 Avec pénalisation

L'expérience suivante consiste à l'évaluation de chaque solution, par l'exploitation de la fonction de coût qu'estime le coût d'exécution des nos requêtes tests sur l'entrepôt fragmenté en partition.

On peut accepter certain solutions au cours de la recherche même s'ils ont de mauvaises qualités et ces solutions peuvent être inadmissibles. Pour admettre les solutions inadmissibles, nous avons utilisé une fonction de pénalité pour la solution violant une contrainte Lee et Hammer [100].

Ainsi, et pour les réévaluer, il faut l'inclure une fonction de pénalité. Certaine forme de cette fonction de pénalité est définies. Dans [100] les auteurs Lee et Hammer ont proposé trois formes de la fonction de pénalité, et trois modes pour l'inclure dans la fonction objective. Ils ont traité ce problème d'une vision de maximisation du gain produit par l'action de matérialisation d'un ensemble de vues sous une contrainte du coût de maintenance.

L'exploitation de la fonction de pénalité exprime que le résultat obtenu est typique (**FIG. 5.17**), car le temps d'exécution de la charge des requêtes tests diminue d'une façon importante et très remarquable, ce qu'il expliqué que les résultats trouvée est quasi- optimal.

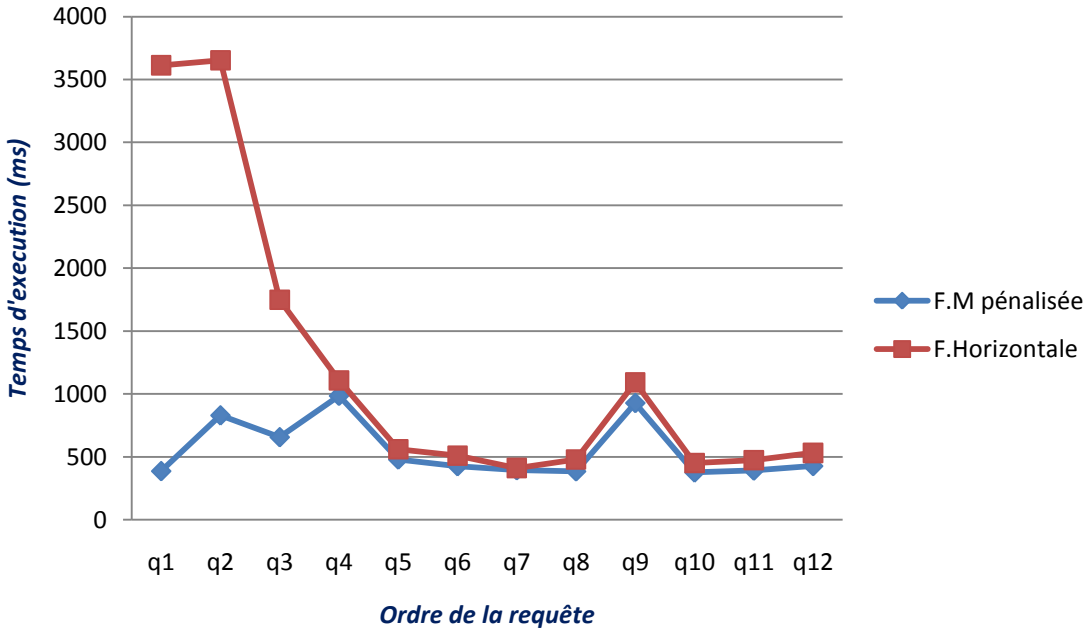

FIG. 5.17 Temps de réponse avec et sans pénalité.

Impact des modes de pénalité

Comme on a déjà cité qu'il existe trois forme de pénalité, La tache de choisir un des formes de pénalité rentre comme un enjeu crucial et joue aussi un rôle additif pour atteint une solution affine, on essai d'évaluer le rapport apporté par les trois modes durant la recherche, la figure (**FIG. 5.18**) étale les résultats obtenue par l'exécution de l'AG exploitant les opérateurs adéquats à chaque mode

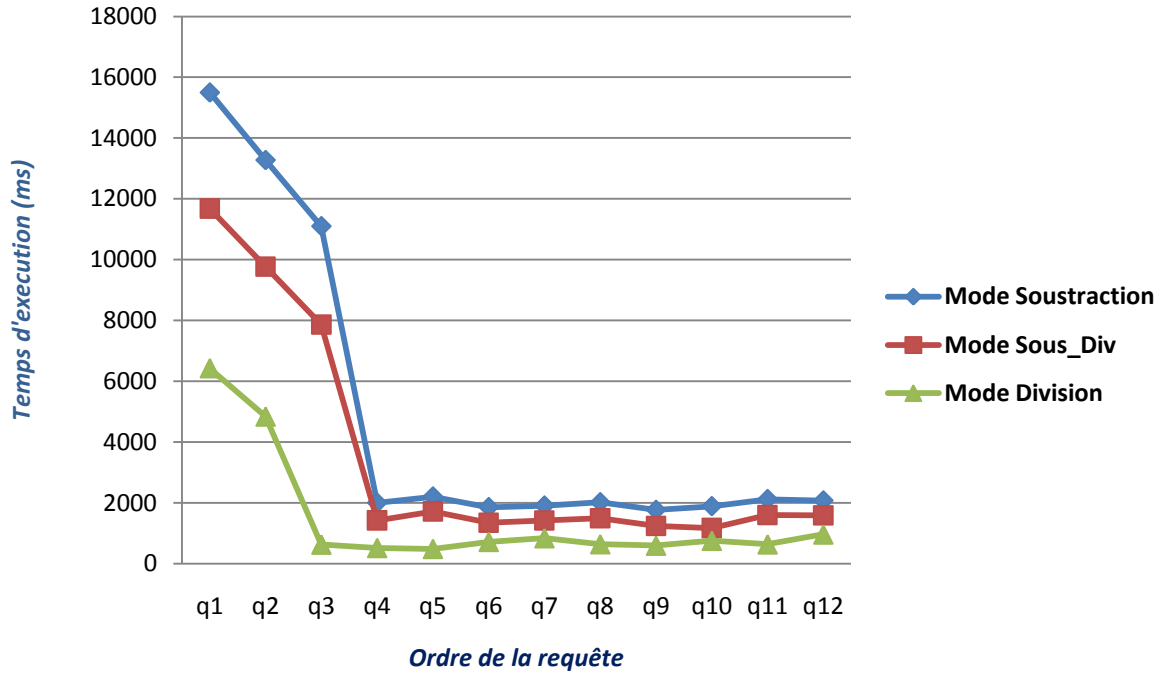

FIG. 5.18 L'impact du choix entre les trois modes de pénalité.

Si on examine la figure précédente, on trouve que, l'intérêt de l'utilisation et la variation entre mode est nettement apparaît, on arrivant a une réduction importante du coût.

On distingue aussi que l'utilisation de mode division nous apporté la meilleure réduction par rapport au deux autre mode, qui peut atteint un gain jusqu'au demi de cout du deuxième mode rassemblé au mode soustraction division, qui positionne en deuxième place parmi ces mode, et enfin le mode soustraction en troisième position, on remarque aussi que ces trois mode suit un trajet similaire durant l'opération d'exécutions de nos requêtes.

Selon les trois types de pénalité déjà définis, on à la possibilité de choisir entre logarithmique, exponentielle ou bien linéaire, la figure (**FIG. 5.19**) montre la différence entre ces trois types.

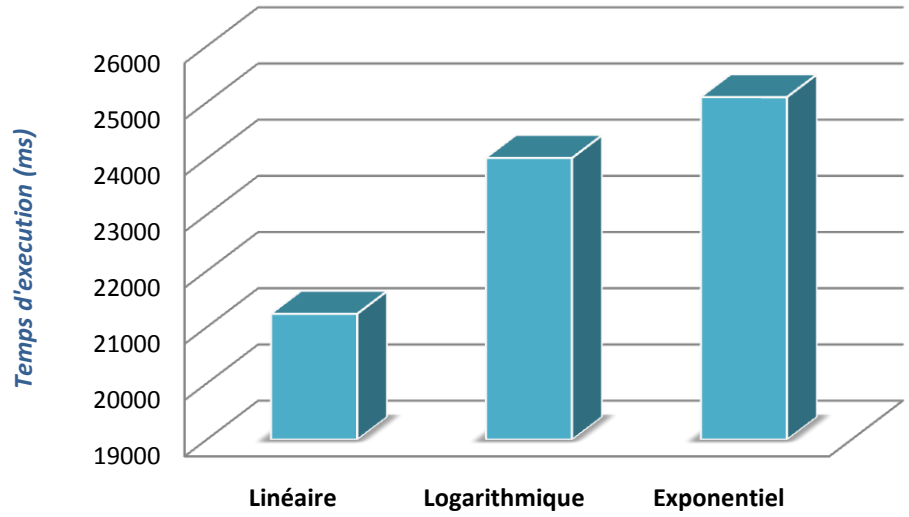

FIG. 5.19 La relation entre le coût total et le type de pénalité.

Cette figure nous montre que le choix de pénalité est capital, est de type linéaire puis que lors qu'on change l'écart d'une sorte logarithmique ou exponentielle, cela réduit l'acte de pénalité. En conséquence, nous travaillons à prouvé l'efficacité de la valeur prédéterminé $\alpha = 0{,}6$ et le trouve si elle est justifiable ou non, nous avons travaillé a varié cette valeur d'une manière ascendante dans l'intervalle de$[0{,}1 \ldots . 0{,}9]$ et pour chaque valeur on a appliqué l'ensemble des requêtes sur l'entrepôt de données fragmenté, la figure (**FIG. 5.21**) illustre les résultats obtenue. exponentiel logarithmique linéaire .

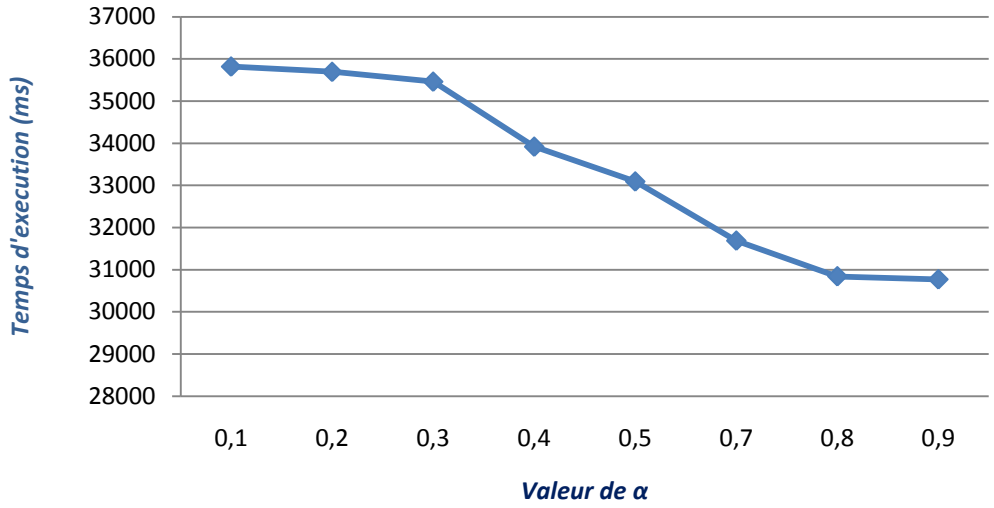

FIG. 5.20 Relation entre la valeur αet le cout total d'exécution.

Impact du nombre de dimensions

On passe maintenant a une nouvelle expérience consiste a l'étude de l'impact du nombre des dimensions participant a ce processus génétique, ce dernier pour le but de soutenir une optimisation de requête sur les résultats de la recherche. Pour cela, on travaille a fragmenté l'entrepôt de données d'une manière ascendante: initialement par une dimension puis par deux dimensions ensuite par trois dimensions…etc. les résultats réussis nous dévoilent qu'il existe un rapport entre la fragmentation et le nombre de dimensions consiste au seuil de fragmentation (**FIG. 5.21**).

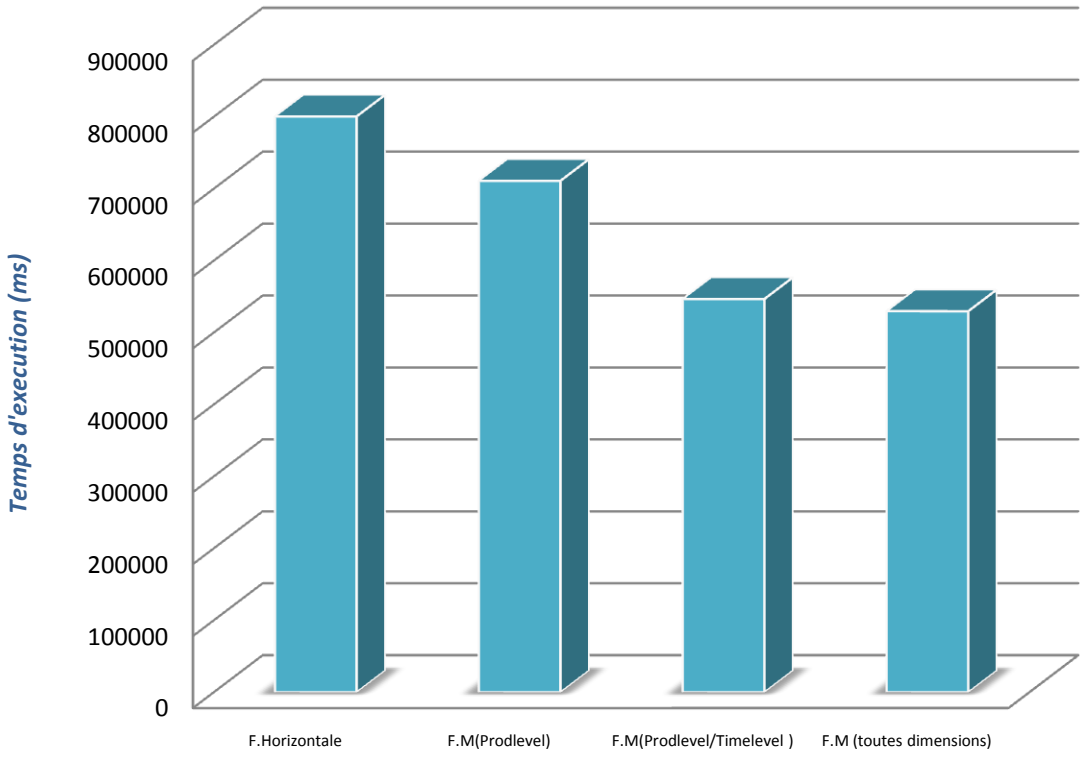

FIG. 5.21 Effet du nombre de dimensions sur le coût total d'exécution.

Notre évaluation repose sur le facteur ***Treshold*** qui consiste à l'estimation de dimensions pertinentes ainsi que le nombre de dimensions.

Conclusion

Dans ce dernier chapitre, nous avons commencé par une étude comparative théorique des trois approches existe de la fragmentation horizontale puis on étalé notre approche d'optimisation des requêtes OLAP dans l'entrepôt de données. Nous avons montré

l'application de processus génétique de fragmentation mixte sur ce genre de système. Enfin nous évaluons notre modèle par l'utilisation d'APB Benchmark release II [91].

En réalité, nos travaux sur ce chapitre consistent à l'évaluation des techniques de fragmentation dans un entrepôt de données précisément dans le schéma en étoile, cette schéma est généré par l'utilisation du SGBD oracle 10g,ce genre de schéma est composé de quatre tables de dimensions (chanlevel, prodlevel, custlevel et timelevel)et d'une table de faits (Actvars).

Pour abouti l'objective d'optimisation de temps d'exécution des requêtes OLAP, on suppose qu'il existe un ensemble de requêtes $Q = \{Q_1, Q_2, \dots\dots, Q_m\}$joignant la table de fait avec les tables de dimensions. En premier lieu nous avons utilisé l'algorithme basé sur la fragmentation horizontale, puis, nous avons utilisé notre algorithme suivant ces deux modes avec pénalité ou sans elle, enfin nous avons affirmé que nos contributions donnent de meilleurs résultats devant ces expérimentations.

Conclusion et perspectives

Les travaux effectués dans ce mémoire se rentrent dans le contexte de la conception physique de l'entrepôt de données relationnel. Particulièrement nous avons développé notre propre approche d'optimisation de requêtes reposée sur les concepts de la génétique.

Même s'il existe plusieurs modèles et techniques d'optimisation dans la littérature, et que chacun de ces techniques a des avantage et des limites donc il n'existe pas d'optimalité totale localise dans l'une de ces techniques. A cet effet, Nous avons présenté les différentes techniques exploités dans le domaine d'entreposage de données, Telle que les vues matérialisées, l'indexation, et la fragmentation en ces deux modes. Ces techniques se basent sur les différentes modélisations des entrepôts de données : le modèle ROLAP et le modèle MOLAP. Nous avons vue aussi les avantages et les inconvénients liés à chaque technique.

Dans le cadre de ce travail, nous avons captivés à la fragmentation horizontale et à l'exploitation d'un algorithme génétique d'optimisation de requêtes OLAP. Les algorithmes génétiques sont des approches d'optimisation caractérisant par d'excellentes propriétés, telle que, la recherche coopérative, le traitement parallèle et efficace d'espaces de recherche définie a sa complexité, la construction progressive de solutions partielles.

On commence d'abord par la formalisation du problème de la fragmentation horizontale dans les entrepôts de données. Puis par la présentation des travaux réalisés dans la fragmentation horizontale, en inspirant parmi les techniques existe dans les bases de données relationnelles. Ensuite, nous avons éclairci quelque problème concernant la fragmentation de la table des faits suivant les tables de dimensions, et nous avons proposé l'utilisation algorithmes génétique pour la fragmentation mixte.

Notre contribution comporte une fragmentation horizontale d'un schéma relationnel d'un entrepôt de données, puis fragmenté verticalement dans le but de baisser le coût d'exécution de requêtes. Par la suite, on travaille a formalisé le problème de sélection de schéma de fragmentation verticale et le rendre comme un problème d'optimisation avec contrainte. Cette problématique indique le nombre de fragments verticaux nécessaire que l'administrateur peut fixer. Pour le répondre, nous avons offert un algorithme génétique qui rassemble au même codage que l'algorithme déjà proposé [26].

Un processus de codage des schémas de fragmentation est exposé. Un modèle de coût qui étale la fonction sélective est exploité pour essai d'évaluer la qualité de la solution choisie.

L'étude et l'évaluation de nos contributions, particulièrement l'approche proposé afin d'assurer une optimisation de l'entrepôt de données, et d'illustré l'impact positif de cette approche sur le temps d'exécution des requêtes de type OLAP.

Les résultats engendrés par l'utilisation de l'algorithme génétique sont encourageants et très captivants et dévoilent que la fragmentation mixte peut être appliquée sur les entrepôts de données possédant des tables de dimensions importantes. Avec une fragmentation appropriée des tables de dimensions, on peut augmenter les performances de déroulement la démarche d'exécution des requêtes sans violant le seuil des fragments que l'administrateur doit proposer ou fixer.

L'idée de proposer une nouvelle approche pour sélectionner un schéma de fragmentation est très original et prometteuse. Les résultats présentés dans cette mémoire montrent l'importance de nos travaux tout en contribuant à un axe de recherche très prometteur et d'actualité.

Enfin, en récapitule que les résultats abouti, derrière l'utilisation de cette approche, explique le choix des algorithmes génétiques pour traiter le problème de sélection d'un schéma de fragmentation dans un entrepôt de données.

Finalement, nous proposons d'éventuelles perspectives pour l'extension de notre travail, comme l'application de notre algorithme sur des tables très volumineuses. On essai aussi d'établir notre algorithme et le rendre compatible, même on cas ou il existe une évolution des requêtes soit dans leurs structures ou dans leurs fréquences.

De ce fait, nous proposons d'éventuelles perspectives pour l'extension de notre travail, on cite :

- Essai d'étudier ces techniques d'optimisation en mode combiné et le traité soit de les rassemblés deux a deux ou plus.
- Utiliser le modèle de coût complet déjà existe qui donne une estimation très exacte avec la prise en charge de la taille du tampon(Buffer), ainsi que la prise en compte du coût des agrégations et des groupements.
- Inspirer des nouveaux techniques d'optimisation qui prévenant essentiellement de l'ancien techniques appliqué aux bases de données.
- Réaliser un outil d'aide à l'administration sous nous vision.

Ou au moins, nous travaillons d'augmenter mieux les propriétés de notre approche proposés par :

- La proposition d'une population initiale mieux dépendante de la solution recherchée.
- Le rend de la modalité d'arrêt plus flexible.

En revanche, il est important de ne pas se localiser sur la recherche d'une réponse exacte pour les requêtes OLAP, parce que même les approches aide a l'approximation des résultats de ces requête est considérer comme très intéressante. Ces approches amènent une possibilité d'exploitation et d'extraction de données nécessaires qui assure la réduction du temps de réponse et de la taille des données traités.

Glossaire

AE : Algorithme Evolutionnaire

AED : Administrateur de l'Entrepôt de Données.

AF : Attribut de Fragmentation.

AG : Algorithme Génétique.

DM : Data Mining (fouille de données).

DSAD : Dispositif stockage à accès direct.

DSS : decision support systems (Informatique décisionnelle)

FH : Fragmentation Horizontale.

FHD : Fragmentation Horizontale Dérivée.

FHP : Fragmentation Horizontale Primaire.

FM : Fragmentation mixe

FV : Fragmentation verticale

HASH : Mode de fragmentation horizontale permettant de partitionner une table selon une fonction de hachage.

HC : Hill Climbing.

IJB : Index de Jointure Binaire.

MIS : management information system (Management du système d'information)

MOLAP: Multidimensional On-Line Analytical Processing.

OLAP: On-Line Analytical Processing.

OLTP: On-Line Transaction Processing

OOLAP: Object OLAP

PSV : Problème de sélection des vues matérialisées

RANGE : Mode de fragmentation horizontale permettant de partitionner une table selon des intervalles de valeurs.

RJE : Requêtes de Jointure en Etoile.

ROLAP: Rolational On-Line Analytical Processing.

RowID : Identificateur de ligne dans un SGBD relationnel.

RS : Recuit Simulé.

SE : Sous-schéma en Etoile.

Sel : Facteur de Sélectivité dans la table des faits pour le prédicat.

SF : Schéma de Fragmentation.

SI : Système d'information

SGBD: Système de Gestion de Base de données.

SPJ : Sélection, Projection et Jointure

W : Le nombre de fragments que l'administrateur souhaite avoir (paramètre utilisé par nos algorithmes de fragmentation).

Bibliographie

[1] E.F. Codd, S.B. Codd, C.T. Salley, Providing OLAP (On Line AnalyticalProcessing) to user analyst : an IT mandate, rapport technique, E.F. Coddandassociates, (white paper de Hyperion Solutions Corporation), 1993.

[2] la théorie du système générale**,**Théorie de la modélisation , 1977, PUF. Rééditions en 1986, (ISBN 2-13038-483-8).

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

[4] K. Aouiche, J. Darmont, O. Boussaid, and F. Bentayeb. Automatic Selection of BitmapJoin Indexes in Data Warehouses. 7th International Conference on Data Warehousingand Knowledge Discovery (DAWAK 05), August 2005.

[5] C. D. The dificulty of optimum index selection.ACM Transactions on Database Systems(TODS), 3(4) :440–445, 1978.

[6] Adamson C. "Mastering data warehouse aggregates solutions for star schema performance".Wiley Pub, 2006.

[7] LadjelBellatreche. « Utilisation des index et de la fragmentation dans la conception logique et physique d‟un entrepôt de données » .Thèse pour obtenir le grade de Docteur en Informatique, université de Clermond Ferrand II, 2000.

[8] Golfarelli M. "From user requirements to conceptual design in data warehouse design – a survey", 2009.

[9] Rizzi, S., Abello, A., Lechtenborger, J., Trujillo J., « Research in data warehouse modeling and design: dead or alive? », In: DOLAP, 2006.

[10] Bubenko J., Rolland C., Loucopoulos P., De Antonellis V., « Facilitating „fuzzy to formal‟ requirements modelling ». IEEE 1st Conference on Requirements Engineering, ICRE‟94 pp. 154-158, 1994.

[11] Bruckner R., List B., Schiefer J. « developing requirements for data warehouse systems with use case ». Seventh Americas Conference on Information Systems, 2001.

[12] Schiefer J., List B., Bruckner R.M., « A Holistic Approach For Managing Requirements of Data Warehouse Systems », Eighth Americas Conference on Information Systems, 2002.

[13] Sen A. and Sinha A P. « A Comparison of Data Warehousing Methodologies ». Communications of the ACM.March 2005/Vol. 48, No. 3.

[14] Ballard C., Herreman D., Schau D., Bell R., Kim E., Valencic A. « Data Modeling Techniques for Data Warehousing » International Technical Support Organization, IBM, February 1998, http://www.redbooks.ibm.com.

[15]Rilston F., Paim S., Jaelson F. B. Castro. « DWARF: An Approach for Requirements Definition and Management of Data Warehouse Systems ». Proceeding of the 11th IEEE International Requirements Engineering Conference, September 08 - 12 (2003), pages1090-1099.

[16] Winter R., Strauch B. « Information Requirements Engineering for Data Warehouse Systems ». Proceedings of the 2004 ACM symposium on Applied computing, 2004.

[17] Fankam C., Jean S., Pierra G., Belltreche L., Ait-Ameur Y. « Towards Connecting Database Applications to Ontologies », First International Conference on Advances in Databases, Knowledge, and Data Applications, 2009.

[18] Marhoumi F. E., Entrepôts de données XML : Développement d'un outil Extraction Transformation Load (ETL), Memoire de fin d'etudes en vue de l'obtention du grade d'Ingenieur Civil Informaticien en Sciences Appliquées, Université de Bruxelles, 2006.

[19] S. Chaudhuri and V. R. Narasayya.Selftuningdatabase systems : A decade of progress.In VLDB, pages 3–14, 2007.

[20] L. Bellatreche, K. Boukhalfa, and M. K. Mohania.Pruning search space of physical databasedesign.In DEXA, pages 479–488, 2007.

[21] L. Bellatreche, R. Missaoui, H. Necir, andH. Drias. A data mining approach for selectingbitmap join indices. Journal of ComputingScience and Engineering, 2(1) :206–223,2008.

[22] H. Gupta. Selection of views to materialize ina data warehouse. Proceedings of the 6th InternationalConference on Database Theory(ICDT ’97), pages 98–112, 1997.

[23] H. Gupta, V. Harinarayan, A. Rajaraman,and J. Ullman. Index selection for olap.Proceedings of the International Conferenceon Data Engineering (ICDE), pages 208–219,April 1997.

[24] S. Navathe and M. Ra. Vertical partitioningfor database design : a graphical algorithm.ACM SIGMOD, pages 440–450, 1989.

[25] A. Sanjay, V. R. Narasayya, and B. Yang.Integrating vertical and horizontal partitioninginto automated physical database design.Proceedings of the ACM SIGMOD InternationalConference on Management ofData, pages 359–370, June 2004.

[26] L. Bellatreche, K. Boukhalfa, and H. I. Abdalla.Saga : A combination of genetic andsimulated annealing algorithms for physicaldata warehouse design. in 23rd British NationalConference on Databases, (212-219), July2006.

[27] T. Stöhr, H. Märtens, and E. Rahm.Multidimensional database allocation for paralleldata warehouses.Proceedings of the International Conference on Very Large Databases,pages 273–284, 2000.

[28] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. The Data Warehouse LifecycleToolkit : Expert Methods for Designing, Developing, and Deploying Data Warehouses.John Wiley and Sons, 1998.

[29] Inmon et Hackarton, 1994]. Using the Data Warehouse. John Wiley & Sons, 304.,ISBN : 0471059668..

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

[31] M. Golfarelli, D. Maio, and S. Rizzi. Conceptual design of data warehouses from e/r schemes. in the 31th Hawaii Conference on System Sciences, January 1998.

[32] P. Valduriez. Join indices. ACM Transactions on Database Systems, 12(2) :218–246, June1987.

[33] R. B. Systems. Star schema processing for complex queries. White Paper, July 1997.

[34] P. O'neil. Multi-table joins through bitmapped join indioces. SIGMOD, 24(03), 1995.

[35]D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, andS. Fadden. Db2 design advisor : Integrated automatic physical database design. Proceedingsof the International Conference on Very Large Databases, pages 1087–1097, August2004.

[36] B. P. and M. H. The dimension-join : A new index for data warehouses. in 16th SimpósioBrasileiro de Banco de Dados (SBDD 2001), Rio de Janeiro, Brazil, pages 259–273, 2001.

[37] M. Frank, E. Omiecinski, and S. Navathe.Adaptive and automated index selection inrdbms. In 3rd International Conference onExtending Database Technology (EDBT 92), Vienna, Austria, pages 277–292, 1992.

[38] S. Choenni, H. Blanken, and T. Chang. Indexselection in relational databases. In 5th InternationalConference on Computing and Information(ICCI 93),Ontario, Canada, pages491–496, 1993.

[39] S. Choenni, H. Blanken, and T. Chang. Onthe selection of secondary indices in relationaldatabases. Data Knowledge Engineering,11(3) :207–238, 1993.

[40] S. Chaudhuri and V. Narasayya.An efficientcost-driven index selection tool for Microsoft.

[41] G. Valentin, M. Zuliani, D. Zilio, G. Lohman,and A. Skelley. Db2 advisor : An optimizersmart enough to recommend its ownindexes. In 16th International Conferenceon Data Engineering (ICDE 00), San Diego,USA, pages 101–110, 2000.

[42] M. Golfarelli, , and E. Rizzi, S. Saltarelli.Index selection for data warehousing. Proceedings4th International Workshop on Designand Management of Data Warehouses(DMDW'2002), Toronto, Canada, pages 33–42, 2002.

[43] S. Chaudhuri. Index selection for databases :A hardness study and a principled heuristicsolution. IEEE Transactions on Knowledgeand Data Engineering, 16(11) :1313–1323, November 2004.

[44] W. Labio, D. Quass, and B. Adelberg.Physicaldatabase design for data warehouses.Proceedingsof the International Conference onData Engineering (ICDE), 1997.

[45] H. Gupta and I. Mumick.Selection of views to materialize in a data warehouse. IEEETransactions on Knowledge and Data Engineering, 17(1) :24–43, 2005.

[46] S. Chaudhuri and V. Narasayya.Autoadmin'what-if' index analysis utility.Proceedingsof the ACM SIGMOD International Conferenceon Management of Data, pages 367–378, June 1998.

[47] P. D. A. Shukla and J. Naughton. Materialized view selection for multi-cubedata models.7 th International Conference on Extending DataBase Technology (EDBT 00), Konstanz,Germany, pages 269–284, 2000.

[48] S.Agrawal, V.Narasayya, B Yang. Integrating Vertical and Horizontal Partitioninginto Automated Physical Database Design. Proceedings of the 2004ACM SIGMOD Paris, France PP. 359 - 370.

[49] K.Whang. Index selection in relational databases. In International Conference on Foundations of Data Organization (FODO 85), Kyoto, Japan, pages 487–500, 1985.

[50] G. T.I. Near optimal multiple choice index selection for relational databases. Computers &Mathematics with Applications, 37(2) :111–120, 1999.

[51] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias. Selection and pruning algorithmsfor bitmap index selection problem using data mining. 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'07), LNCS, pages 221–230, September 2007.

[52] H. Gupta. Selection and maintenance of views in a data warehouse.Ph.d. thesis, StanfordUniversity, September 1999.

[53] J. Ullman. Efficient implementation of data cubes via materialized views. in the Proceedingsof the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96), pages 386–388, 1996.

[54] Hammer M., Niamir B. « A heuristic appoach to attribute paritioning »,Proceedings of the ACM SIGMOD International Conference on Management
of Data, pages 93-101, 1979.

[55] P. A. Bernstein and D.-M. W. Chiu. Using semi-joins to solve relational queries. Journalof the ACM, 28(1) :25–40, January 1981.

[56] N. A. R. Metropolis, M. Rosenbluth, A. Teller, and E. Teller.Equation of state calculationsby fast computing.

[57] H. Gupta. Selection and maintenance of views in a data warehouse.Ph.d. thesis, StanfordUniversity, September 1999.

[58]L. Bellatreche. Techniques d'optimisation des requêtes dans les data warehouses.,LISI/ENSMA,Poitiers, 2003.

[59] M. de Souza and M. Sampaio.Efficient materialization and use of views in data warehouses.SIGMOD Record, 28(1) :78–83, 1999.

[60] P. D. A. Shukla and J. Naughton. Materialized view selection for multi-cubedata models.7 th International Conference on Extending DataBase Technology (EDBT 00), Konstanz,Germany, pages 269–284, 2000.

[61] V. Harinarayan, A. Rajaraman, and J. Ullman. Implementing data cubes efficiently.Proceedings of the ACM SIGMOD International Conference on Management of Data,pages 205–216, June 1996.

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

[63] X. Y. C. Zhang and J. Yang. An evolutionaryapproach to materialized view selection in adata warehouse environment. IEEE Transactionson Systems, Man, and Cybernetics,31(3) :282–294, 2001.

[64] T. Nadeau and T. Teorey. Achieving scalabilityin olap materialized view selection. 5thACM International Workshop on Data Warehousingand OLAP (DOLAP 02), McLean,USA, pages 28–34, 2004.

[65] H. Gupta and I. Mumick.Selection of viewsto materialize in a data warehouse. IEEETransactions on Knowledge and Data Engineering,17(1) :24–43, 2005.

[66] M. T. Özsu and P. Valduriez. Principles ofDistributed Database Systems : Second Edition.Prentice Hall, 1999.

[67] A. Sanjay, V. R. Narasayya, and B. Yang.Integrating vertical and horizontal partitioninginto automated physical database design.Proceedings of the ACM SIGMOD InternationalConference on Management ofData, pages 359–370, June 2004.

[68] J. Rao, C. Zhang, G. Lohman, and N. Megiddo.Automating physical database designin a parallel database. Proceedings of theACM SIGMOD International Conference on Management of Data, pages 558–569, June2002

[69] M. de Souza and M. Sampaio.Efficient materializationand use of views in data warehouses.SIGMOD Record, 28(1) :78–83, 1999

[70] S. Ceri, M. Negri, and G. Pelagatti. Horizontaldata partitioning in database design. Proceedings of the ACM SIGMOD InternationalConference on Management of Data. SIGPLAN Notices, pages 128–136, 1982.

[71] L. Bellatreche, M. Schneider, H. Lorinquer,and M. Mohania. Bringing together partitioning,materialized views and indexes tooptimize performance of relational data warehouses.Proceeding of the InternationalConference on Data Warehousing and KnowledgeDiscovery (DAWAK'2004), pages 15–25, September 2004.

[72] S. R. Valluri, S. Vadapalli, and K. Karlapalem.View relevance driven materializedview selection in data warehousing environment.13th Australasian Database TechnologiesConference (ADC 2002), Melbourne,Australia, pages 187–196, 2002.

[73] S. Navathe, S. Ceri, G.Wiederhold, and D. J. Vertical partitioning algorithms for database design. ACM Transaction on Database Systems, 9(4) :681–710,December 1984.

[74] M. Golfarelli and S. Rizzi. View materializationfor nested gpsj queries. 2nd InternationalWorkshop on Design and Management ofData Warehouses (DMDW 00), Stockholm,Sweden, pages 10.1–10.9, 2000.

[75] Shamkant B. Navathe, Stefano Ceri, GioWiederhold, Jinglie Dou: VerticalPartitioning Algorithms for Database Design. ACM Trans. Database Syst. 9(4):680-710 (1984).

[76] S. Papadomanolakis and A. Ailamaki.Autopart: Automating schema design for largescientific databases using data partitioning.Proceedings of the 16th International Conferenceon Scientific and Statistical DatabaseManagement (SSDBM 2004), pages 383–392,June 2004.

[78] X. Baril and Z. Bellahsène. Selection of materializedviews : a cost-based approach. 15thInternational Conference on Advanced InformationSystems Engineering (CAiSE 03),Klagenfurt, Austria, pages 665–680, 2003.

[79] C.I. Ezeife. Selecting and materializing horizontally partitioned warehouseviews. Data and Knowledge Engineering jornal, vol 36, elsevier 2001.

[80]D. Theodoratos and W. Xu. Constructingsearch spaces for materialized view selection.7th ACM International Workshop on DataWarehousing and OLAP (DOLAP 04), WashingtonDC, USA, pages 112–121, 2004.

[81] W. H. Inmon. Building the Data Warehouse. Third Edition, Wiley ComputerPublishing 2002.

[82] D. DeWitt and J. Gray. Parallel databasesystems : the future of high performance databasesystems. Commun. ACM, 35(6) :85–98, 1992.

[83] L. Bellatreche, K.Karlapalem, M. Mohania. OLAP Query Processing for PartitionedData Warehouses.International Symposium on Database Applications in Non-Traditional Environments Kyoto, Japan, PP. 35, 1999.

[84] K. R. H. Uchiyama and T. Teorey. A progressive view materialization algorithm.2nd ACM International Workshop on Data warehousing and OLAP (DOLAP 99), KansasCity, USA, pages 36–41, 1999.

[85] R. B. Systems. Star schema processing forcomplex queries. White Paper, July 1997.

[86] F.Baiao, Mattoso.F, Zaverucha.M. Horizontal fragmentation in objectDBMS : new issues and performance evaluation. Performance, Computing,and Communications Conference , Phoenix, USA, PP. 108 - 114, 2000.

[87] A. Sanjay, C. Surajit, and V. R. Narasayya.Automated selection of materialized viewsand indexes in microsoftsql server. Proceedingsof the International Conference onVery Large Databases, pages 496–505, September2000.

[88] K. Boukhalfa, L. Bellatreche, P. Richard. Fragmentation Primaire et Dérivée: Étude de Complexité, Algorithmes de Sélection et Validation sousORACLE10g. Rapport de recherche N 01 - 2008. Ecole nationale supérieurede mécanique et d'aérotechnique Poitier.

[89] Sanjay A., Narasayya V. R., Yang B., « Integrating Vertical and HorizontalPartitioning Into Automated Physical Database Design », Proceedings of theACM SIGMOD International Conference on Management of Data, vol., p. 359-370, June, 2004.

[90] T. Nadeau and T. Teorey. Achieving scalability in olap materialized view selection. 5th ACM International Workshop on Data Warehousing and OLAP (DOLAP 02), McLean, USA, pages 28–34, 2004.

[91] O. Council. Apb-1 olap benchmark, release ii. http ://www.olapcouncil.org/ research/ resrchly.htm,1998.

[92] Papadomanolakis, S. et A. Ailamaki (2004). Autopart : Automating schemadesign for large scientific databases using data partitioning. Proceedings of the16th International Conference on Scientific and Statistical Database Management(SSDBM 2004), 383–392.

[93] S. Ceri and G. Pelagatti. Distributed Databases: Principles and Systems.McGraw Hill International Editions, 1984.

[94] L. Bellatreche, K. Karlapalem, M. Mohania, M. Schneider. What can partitioningdo for your data warehouses and data marts ? Database Engineeringand Applications Symposium International Yokohama, Japan, PP.437 - 445 ,2000.

[95] K. Boukhalfa, L. Bellatreche, H. I. Abdalla. Algorithms for Physical DataWarehouse Design to Speed up Decision-making Processes.The Saudi National
Computer Conference NCC'06, SaudiArabia, Mars 2006.

[96] N. Barnier. Optimisation par algorithme génétique sous contraintes. Techniqueet science informatiques, Volume 18, PP. 1 - 29, 1999.

[97] E. Alba. EvolutionaryAlgorithms. Dans A. Y. Stephan Olariu, Handbookof Bioinspired Algorithms and Applications. PP. 3 - 19 .Taylor and FrancisGroup, LLC 2006.

[98] T. Blickle. Theory of Evolutionary Algorithms and Application to SystemSynthesis.Ph.Dthesis University of Saarbrucken Germany, January 1967.

[99] Yu J. X., Choi C.-H., Gou G.(November, 2004), “Materialized View Selection as Constrained Evolution Optimization”, IEEE Transactions On Systems, Man, andCybernetics, Part 3, vol. 33, n° 4, p. 458-467.

[100] M. Lee and J. Hammer.Speeding up materialized view selection in data warehouses usinga randomized algorithm. Int. J. Cooperative Inf. Syst., 10(3) :327–353, 2001.

Annexe

		nombre de valeurs distinctes	nombre d'enregistrements	Taille d'enregistrement
custlevel	store_level	900		
	retailer_leve	99		
			900	24
timelevel	tid	24		
	year_level	2		
	quarter_level	4		
	month_level	12		
	week_level	1		
	day_level	2		
			24	36
chanlevel	base_level	9		
	all_level	0		
			9	24
prodlevel	code_level	900		
	class_level	605		
	group_level	300		
	family_level	75		
	line_level	15		
	division_level	4		
			9000	72
actvars	customer_level	37		
	product_level	6500		
	channel_level	9		
	time_level	17		
	unitsSold			
	dollarSales			
	dollarCost			
			1500000	74

Intérêt de la fragmentation horizontale

L'Expérienceétale lerapport de Temps d'exécution des requêtes selon deux mode de schémas (*fragmentéetnon fragmenté*) suivant une charge de 12 requêtes.

	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12
sans fragmentation	3718	3975	4882	3256	2341	2003	2065	2311	1997	2276	2134	2191
F.horizontale	2189	1989	2203	2016	2193	2132	2049	2200	1976	1891	2125	2171

Impact de la fragmentation mixte

Les résultats de deuxième expérience, qui montre la différence entre la fragmentation mixte proposée et horizontale.

	q1	q2	q3	q4	q1	q2	q3	q4	q1	q2	q3	q4
F.Horizontale	2412	1882	2467	1712	2219	2345	2013	2179	1847	2088	2247	2244
F.Mixte	1986	2314	2302	2121	2383	2190	2057	2117	1698	1979	2169	1658

Impact du nombre de fragments verticaux

Les résultats de troisième expérience, qui expose L'impact du nombre de fragments verticaux sur la performance.

	q1	q2	q3	q4	q1	q2	q3	q4	q1	q2	q3	q4
7 partitions	6213	6097	5894	5769	5329	5295	5221	5154	5093	2871	2367	2189
8 partitions	8277	5789	6240	5871	8669	7465	6971	5275	5349	3115	2398	2304
9 partitions	8335	8073	9235	8979	7812	7519	8215	7876	6922	3349	2425	2517
10 partitions	9622	9236	8097	9411	9198	8597	8793	8548	8421	4116	3076	3126

Les résultats de l'expérience concernant au cout total d'exécutions de l'ensemble suivant le nombre de partitions.

	N=7	N=8	N=9	N=10
Temps d'execution (ms)	5897	11781	15776	17816

Le rapport de pénalisation

Les résultats remplis a ce tableau expose le temps de réponse avec et sans pénalité.

	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12
F.M pénalisée	387	829	656	984	479	426	394	385	927	376	391	427
F.Horizontale	3613	3652	1749	1108	561	509	411	479	1092	451	473	531

Impact des modes de pénalité

Le tableau suivant étale les résultats obtenus par l'exécution de l'AG exploitant les opérateurs adéquats aux trois modes de pénalité.

	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12
Mode Soustraction	15494	13272	11097	1997	2204	1854	1905	2015	1764	1886	2118	2073
Mode Sous_Div	11681	9772	7862	1424	1719	1349	1417	1491	1243	1172	1602	1585
Mode Division	6427	4839	629	518	487	718	839	642	595	754	636	962

Cette expérience montré au tableau suivant représente la relation entre le coût total et le type de pénalité.

	Linéaire	Logarithmique	Exponentiel
Temps d'execution (ms)	21237	24011	25094

Le résultat exposé au tableau présent montre la relation entre la valeur αet le cout total d'exécution.

	0,1	0,2	0,3	0,4	0,5	0,7	0,8	0,9
Series1	35821	35695	35462	33918	33092	31687	30841	30768

La dernière expérience étale les résultats d'étude de l'impact du nombre des dimensions sur le coût total d'exécution.

	F.Horizontale	F.M(Prodlevel)	F.M(Prodlevel/Timelevel)	F.M (toutes dimensions)
Temps d'execution (ms)	800376	710912	546543	529601

Q1 :

```
Select     Customer_Level,
           Product_level,
           Time_level,
           Sum(Dollarcost)
from       ACTVARS A, CUSTLEVEL
C,PRODLEVEL P,TIMELEVEL T
where
           A.customer_level=C.store_level
           and A.prodcut_level=P.code_level
           and A.time_level=T.TID
           and T.year_level='1996'
           and C.retailer_level ='NXEYFSIQE3JM'
           and P.line_level='MJ1F1U1EG009'
group by
           Customer_level,
           Product_level,
           Time_level
```

Q2 :

```
Select     prodlevel.code_level,
           prodlevel.family_level,
           chanlevel.all_level,
           sum(Actvars.dollarsales),
           Count(actvars.dollarsales),
           count(*)
from       dw.actvars, dw.chanlevel, dw.prodlevel
where
    chanlevel.base_level = actvars.channel_level
    and prodlevel.code_level = actvars.product_level
    and prodlevel.family_level = 'IKKIOVKDGTWO'
    and chanlevel.all_level = 'UFEJCH4EE4C5'
    group by
           prodlevel.code_level,
           prodlevel.family_level,
           chanlevel.all_level
```

Q3 :

```
SELECT
        code_level,
        sum(dollarsales),
        sum(UnitsSold)
FROM  dw.actvars, dw.timelevel, dw.prodlevel
WHERE
        time_level = tid
        and product_level = code_level
        and month_level between '01' and '03'
        and family_level = 'M8VWHZM5BS2N'
group by    code_level
```

Q4 :

```
SELECT
        base_level,
        sum(dollarsales) ,
        sum(UnitsSold)
FROM  dw.actvars, dw.timelevel, dw.prodlevel,
dw.chanlevel
WHERE
        time_level = tid
        and product_level = code_level
        and channel_level = base_level
        and month_level between '10' and '12'
        and family_level = 'M8VWHZM5BS2N'
         and all_level = 'MX0TV6Y3ZYIX'
group by base_level
```

Q5

```
SELECT
       code_level,
       sum(dollarsales)
FROM   dw.actvars, dw.prodlevel, dw.chanlevel
WHERE
```

Q6

```
SELECT
       code_level,
       sum(dollarsales)
FROM   dw.actvars, dw.timelevel, dw.prodlevel
WHERE
```

```
        product_level = code_level
        and channel_level = base_level
        and family_level = 'M8VWHZM5BS2N'
        and all_level <> 'FQYP80SGU24G'
group by      code_level
```

```
        time_level = tid
        and product_level = code_level
        and month_level <> '06'
        and group_level = 'SV5L8W0J8UMZ'
group by      code_level
```

Q7:

```
SELECT
      product_level,
      sum(dollarsales),
      sum(UnitsSold)
FROM   dw.actvars, dw.prodlevel, dw.timelevel
WHERE  time_level = tid
       and product_level = code_level
       and month_level = '03'
       and class_level = 'KPB8DCM3VDMK'
       and division_level = 'DVWQ0OHI8M00'
Group by      product_level
```

Q8:

```
SELECT
      product_level,
      sum(dollarsales),
      sum(UnitsSold)
FROM   dw.actvars, dw.prodlevel, dw.timelevel,
dw.chanlevel
WHERE  time_level = tid
       and product_level = code_level
       and channel_level = base_level
       and year_level = '1995'
       and month_level='03'
       and class_level = 'G7DPNCB3JSFW'
       and division_level = 'D0NS4TAC8RAW'
droup by     product_level
```

Q9:

```
SELECT
       time_level,
       sum(dollarsales),
       sum(UnitsSold)
FROM  dw.actvars, dw.prodlevel, dw.timelevel,
dw.chanlevel
WHERE
      time_level = tid
      and product_level = code_level
      and channel_level = base_level
      and year_level = '1995'
      and month_level='01'
      and class_level = 'G7DPNCB3JSFW'
      and division_level = 'D0NS4TAC8RAW'
group by time_level
```

Q10 :

```
select
       prodlevel.group_level,
       custlevel.retailer_level,
       chanlevel.base_level,
       timelevel.month_level,
       sum(a.UNITSSOLD),
       sum(a.DOLLARSALES)
from dw.actvars a, dw.prodlevel P, dw.timelevel T,
dw.custlevel C, dw.chanlevel CH
where
       a.product_level = P.code_level
       and a.customer_level = C.store_level
       and a.channel_level = CH.base_level
       and a.time_level = T.tid
       and P.line_level = 'WEINIRDJIU32'
       and C.retailer_level = 'R5MPR2102564'
       and CH.all_level = 'UFYXSH4EE4C6'
       and T.month_level between '7' and '9'
group by
```

	prodlevel.line_level, custlevel.retailer_level, chanlevel.all_level, timelevel.month_level
Q11 : select prodlevel.division_level, custlevel.retailer_level, timelevel.month_level, sum (a.unitssold) AS Units, sum (a.dollarsales) as DollarSales, sum (a.DollarCost) as DollarCost, sum (a.dollarsalles) - sum (a.dollarCost) as Margin,(sum(a.dollarsales)- sum(a.DollarCost))/sum(a.DOLLARSALES) as MarginPct from dw.actvars a, dw.prodlevel P, dw.timelevel T, dw.custlevel C where a.product_level = prodlevel.code_level and a.time_level = timelevel.tid and a.customer_level = custlevel.store_level and T.year_level = '1995' and C.retailer_level = 'R5MPR2102564' and P.division_level in ('NF7WO9GBZ20C', 'D0NS4TAC8RAW', 'PE6Z2EN6Y7LQ') group by prodlevel.division_level, custlevel.retailer_level, timelevel.month_level	**Q12 :** select prodlevel.family_level, custlevel.retailer_level, chanlevel.all_level, timelevel.month_level from dw.actvars a, dw.prodlevel P, dw.timelevel, dw.custlevel C, dw.chanlevel CH where a.product_level = prodlevel.code_level and a.customer_level = custlevel.store_level and a.channel_level = chanlevel.base_level and a.time_level = timelevel.Tid and P.family_level = 'M8VWHZM5BS2N' and C.retailer_level = 'R5MPR2102564' and CH.all_level = 'MX0TV6Y3ZYIX' and T.month_level = '01' group by prodlevel.family_level, custlevel.retailer_level, chanlevel.all_level, timelevel.month_level

ملخص :

تعتبر مشكلة تضخم حجم المعلومات في مخازن المعطيات أحد أهم المعوقات التي تحد من الاستخدام الأمثل لهاته التقنية.

حاولنا في هذه المذكرة دراسة تقنيات التحسين الآلي لمخازن المعطيات عن طريق دراسة كل على حدا ثم بدراسة مقارنة للوصول إلى مزايا ومساوئ كل تقنية وبالتالي تكوين نظرة شاملة على هذا الإشكال.

تطرقنا أيضا بصفة أدق لمشكل اختيار مخطط التجزئة الأمثل للتجزئة الأفقية والخوارزميات المرافقة له ودراسة تجريبية مقارنة لهم.

لأجل هذا إستغللنا مساهمة الخوارزميات الجينية لمعالجة هذه المشكلة، وقمنابإقتراح حل يعتمد على هذه الأخيرة وتطبيقها على تجزئة المختلطة ، والمتمثلة في تجزئة أفقية متبوعة بتجزئة عمودية لتقليل تكلفة تنفيذ الاستعلام.

أخيرا للمصادقة على النتائج النظرية ،قمنا بإجراء تجارب تطبيقية على النموذج النظري باستعمال محاكي لمخزن المعطيات (APB-1) على نظام حقيقي (oracle 10g)و حاولنا تقييم النتائج المحصل عليها آنفا والوصول إلى أهداف مرضية.

مفاتيح : مخازن المعطيات, تقنيات التحسين الآلي, تحسين التعليمات,التجزئة الأفقية, خوارزمية الإختيار, الخوارزميات الجينية, تجزئة المختلطة.

Résume :

Le problème d’expansion du volume de l'information dans les entrepôts de données, est considère comme l’un des obstacles les plus importants qui limitent l'utilisation optimale des techniques suivantes.

Nous avons essayé de noter à cet étude des techniques d'optimisation s’auto-administration pour ces entrepôts de données en examinant chaque technique individuellement, puis par une étude comparative pour atteindre au avantages et inconvénients de chaque technique et en formant ainsi un aperçu complet de ce problème.

On concentre aussi plus précisément au problème de la sélection de schéma de fragmentation optimal et ces algorithmes de fragmentation horizontale avec une étude comparative expérimentale.

Pour cela on exploite les apports de l’algorithmique génétique pour répondre à ce problème, et on propose une solution a base de cette dernière appliqué a une fragmentation mixte, qu’elle consiste à fragmenter un schéma d’un entrepôt horizontalement, ensuite verticalement afin de réduire le coût d’exécution de requêtes.

Enfin, pour valider les résultats théoriques, nous avons mené des expériences sur un modèle théorique appliqué à l'aide d’un simulateur d’un entrepôt de données ou benchmark APB-1 sur un système réel (oracle 10 g), et nous avons essayé d'évaluer les résultats obtenus ci-dessus, et atteindre les objectifs de satisfaction.

Mots clé : *Entrepôts de données, auto-administration, optimisation des requêtes, fragmentation horizontale.algorithme de selection, l’algorithmique génétique, fragmentation mixte*

Abstract:

The problem of expansion in the volume of information in data warehouses volume of information is considered as a most significant barrier that limits the optimal use of the following technical.

We tried to note in this study optimization techniques for automated stores data by examining each study individually and then compared to gain access to the advantages and disadvantages of each technique and thereby forming a comprehensive look at this problem.

Also discussed in more accurate to the problem of selection of optimal partitioning scheme for the horizontal segmentation algorithms and the accompanying experimental study and compared them.

To do this we exploit the contribution of genetic algorithms to answer to our problem, and proposes a solution based on the latter applied a mixed fragmentation, it is to break up a diagram of a warehouse horizontally, then vertically to reduce the cost of query execution.

Finally to validate the theoretical results, we have conducted experiments on a theoretical model applied by using the simulator of data werhous (APB-1)on a real system(oracle 10 g), and we tried to evaluate the results obtained above, and reaching the goals of satisfactory

Keywords: *Data warehouses, self-administration, query optimization,horizontale partitioning.algorithme of Selection, genetic algorithms, mixed fragmentation*