République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ibn Khaldoun -Tiaret-

Faculté des sciences et des sciences de l'ingénieur
Département d'Informatique

École Doctorale
Sciences et **T**echnologies de l'**I**nformation et de la **C**ommunication

Optimisation des bases de données par la fragmentation verticale

Mémoire
Pour l'obtention du diplôme de magistère
Option : Systèmes d'**I**nformation et de **C**onnaissances (SIC)

Présenté par :
Hichame Chaalel

Jury

Mlle. Latifa MAHDAOUI	Maître de Conférences -A- Université USTHB Alger	Présidente
Mr. Youcef AKLOUF	Maître de Conférences -A- Université USTHB Alger	Examinateur
Mr. Samir KECHID	Maître de Conférences -A- Université USTHB Alger	Examinateur
Mr. Youcef DAHMANI	Maître de Conférences -A- Université Ibn Khaldoun Tiaret	Examinateur
Mme. Zaia ALIMAZIGHI	Professeur Université USTHB Alger	Directrice de mémoire
Mr. Kamel BOUKHALFA	Maitre de Conférence -B- USTHB Alger	Codirecteur de mémoire

ANNEE UNIVERSITAIRE 2010/2011

TABLE DES MATIÈRES.

TABLE DES MATIÈRES.

Table des Matières.

Liste des tableaux :

Liste des figures :

I. Introduction

Le développement actuel de l'économie de l'information et des technologies de communications et de l'informatique a rendu l'utilisation des systèmes d'information indispensable pour les différents acteurs de l'économie, organisations, et structures publiques.

Aujourd'hui les systèmes d'information sont devenus monnaie courante pour tout secteur confondue, une existence virtuelle de l'organisme, un support permettant de disposer de l'information sous ces différentes formes indépendamment de l'espace temporel et géographique, une aubaine pour les usagers, voir même une source de richesse, et surtout une condition suffisante pour la subsistance et le développement des entreprises.

Ainsi, se doter d'un système d'information ou d'une base de données est devenu une exigence pour la gestion et le pilotage des organisations et des firmes. Conséquence de tout cela, le marché de l'information connait un énorme essor, et la demande sur les systèmes d'information ne cesse de croitre.

Mais cette affluence et cette demande croissante entraine derrière elle beaucoup de défis pour les concepteurs et éditeurs des systèmes d'information, et remet en question les solutions proposées.

En effet les éditeurs SI (systèmes d'information) et des bases de données sont amenés à résoudre à plusieurs contraintes et proposer des solutions pour satisfaire la demande et les perspectives souhaitées par les usagers et consommateurs.

Sur le marché, on constate que le volume d'information exploité ne cesse de croitre, et que le coût de stockage des données ne cesse de baisser, ce qui rajoute plus de contraintes et de problèmes pour les concepteurs et éditeurs, comme le stockage des données, latences du système, réduction des performances.

Une des solutions qui se présente pour résoudre la latence du système et de recourir à un bon paramétrage de la conception physique de la base de données, en effet une bonne conception physique permet d'augmenter les performances du système.

La conception physique est un maillon dans le cycle de vie de la base de données, elle inclut la sélection d'index, de fragments, le clustering, etc. La conception physique permet la gestion des méthodes de stockage et d'accès physique des données sur disque, ce qui permet à la base de données de fonctionner avec une grande efficacité, et permet d'optimiser les performances de la base de

données et des applications fonctionnant dessus. En effet, si la conception physique n'est pas bien ajustée, les performances de la base de données chutent considérablement et vice versa.

La conception physique n'es guère une tâche facile, c'est un art que la majorité des gestionnaires et administrateurs de bases de données maitrisent peu, car la conception physique entraine derrière elle souvent de dizaines, voir de centaines de paramètres, qui sont difficiles à manipuler et à réguler.

Rien que pour la sélection d'une bonne fragmentation horizontale dérivée ou verticale l'administrateur du système se trouve indécis devant de centaines de milliers de combinaison qui s'offre à lui, idem pour la sélection des index et autres paramètres.

L'ajustement de ces paramètres se répercute souvent sur les différentes architectures proposées et sur les performances de la base de données. Aussi, Il est à noter que l'impact de la conception physique se fait ressentir plus sur les bases de données volumineuses que les bases de données non volumineuses.

Les calculs individuels et manuels des performances basés sur un mécanisme d'indexation donné ou d'un algorithme de partitionnement peuvent prendre plusieurs heures, et l'analyse des performances est souvent basée sur la comparaison de nombreuses configurations différentes et en prenant en considération la charge de travail, ce qui nécessite beaucoup de calculs et le résultat obtenu n'est pas forcément satisfaisant.

Cela a donné naissance à des outils automatisés proposant l'aide à la conception physique des bases de données ; tels qu'IBM DB2 Design Advisor, Oracle SQL Access Advisor, Oracle SQL Tuning Advisor et Microsoft Database Tuning Advisor (DTA), etc.

Ces outils d'aide à la conception physique et d'analyse des performances, permettent à l'analyste de se concentrer sur d'autres tâches de sauvegarde et maintenance, et facilite la tâche des administrateurs en leurs proposant plusieurs choix de conception.

Noté que les trois fournisseurs (Oracle, Microsoft, IBM) offrent des outils sophistiqués qui recommandent des index, des vues matérialisées, ainsi que la modélisation de l'impact (coût) sur les transactions INSERT, UPDATE, DELETE en simulant les nouvelles structures.

Dans cette étude, nous nous somme fixé pour objectif d'étudier l'impact de la fragmentation verticale sur les performances des bases de données. Pour faire nous avons étudié et testé différents algorithmes et approches, afin d'aboutir à un schéma de fragmentation verticale optimisé, et d'augmenter les performances de la base de données.

Pour valider et testé les différentes approches et algorithmes, nous avons développé un outil d'assistance pour la fragmentation verticale et intégrants les différentes approches pour la FV.

L'outil en question est FragMan, il permet d'offrir des recommandations sur le meilleur schéma de fragmentation verticale du système, ses recommandations se basent sur l'étude de la structure de la base de données et l'analyse de la charge de travail.

L'optimisation du schéma FV est obtenue par l'application des algorithmes d'optimisation, nous avons implémentés dans cette étude les algorithmes Génétiques et l'algorithme Apriori et l'algorithme Affiner. Affiner est une nouvelle approche que nous avons développé, une méthode qui est simple à implémenter et ne nécessite aucun paramétrage, et permet de générer un schéma de fragmentation verticale optimisé.

En résumé, notre contribution s'est formalisée sur deux volets, le premier concerne une nouvelle approche dite Affiner permettant l'optimisation de la structure physique d'une base de données par la fragmentation verticale. Le deuxième volet concernera l'outil FragMan, un outil qui permet de tester l'approche Affiner et deux autres algorithmes : Algorithme génétique et Algorithme Apriori, et procure des recommandations pour l'assistance dans la conception physique des bases de données.

Le présent mémoire est organisé de la manière suivante :

Chapitre II État de l'art

Dans le chapitre II, nous présentons un état de l'art du domaine de notre étude. Nous commençons par présenter les bases de données et les entrepôts de données où nous mettons l'accent sur la conception physique et les différentes techniques d'optimisation utilisées.

Nous nous concentrons ensuite sur une technique d'optimisation importante qui est la fragmentation verticale et les différents travaux effectués pour sélectionner un schéma de fragmentation verticale.

Nous terminons ce chapitre par présenter quelques outils qui ont été proposés pour aider l'administrateur dans sa tâche de conception physique.

Chapitre III contribution

Le chapitre III représente la première partie de notre contribution. Nous commençons par présenter en détail les deux algorithmes de sélection d'un schéma de fragmentation verticale que nous avons implémenté : un algorithme génétique et un algorithme de fouille de données « data mining » : Apriori.

Nous présentons par la suite l'algorithme Affiner que nous avons proposé. L'explication de cet algorithme sera illustrée grâce à un exemple détaillé.

Chapitre IV FragMan

Le chapitre IV traitera la deuxième partie de notre contribution, à savoir l'outil d'assistance FragMan.

Nous présentons l'architecture générale de l'outil, les principales fonctionnalités ainsi que les résultats de tests effectués sur un benchmark.

Conclusion et perspectives

Enfin pour conclure, la partie conclusion qui récapitule ce qui a été réalisé, et nous proposons quelques perspectives et proposition pour continuer sur la même voie.

II. État de l'art

II.1. INTRODUCTION

Dans ce monde, tout existant ou phénomène soumit à l'espace temporel génère un historique. Cet historique peut être traduit et décomposé en informations, qui peuvent être matérialisées et transformé en un ensemble de données.

Les informations sont d'une grande utilité, car être détenteur d'informations sur un sujet ou une entité signifie détenir des connaissances sur ce dernier. La connaissance d'un sujet, ou d'un phénomène quelconque permet de comprendre sont comportement passé, sont évolution et surtout la prédiction à un certain degré de sont état futur, ou celui d'un cas similaire.

Les chercheurs de diverses disciplines scientifiques se basent dans leurs études et recherches sur le regroupement du maximum d'information sur un phénomène ou une entité. L'étude, l'analyse et la compréhension de ces informations permet d'avoir une idée globale sur l'existence du phénomène, et sa compréhension. Cela permettra de développer le modèle mathématique ou des équations d'états de ce dernier, et nous procure une matière pour l'aide à la décision.

C'est à force de vouloir comprendre l'environnement qui nous entour, que le volume d'informations et des données traitées et récolté ne cessent de croitre sur une échèle exponentielle.

Autre facteur important qui a joué un très grand rôle dans le développement et l'expansion des systèmes d'informations et les bases de données, est celui de la baisse du coût du matériel informatique, précisément le coût de stockage a connus une décroissance ces dernières décennies.

L'étude faite par Lyman et Varian [35] à l'université de Berkley, nous dévoile la baisse du coût de stockage, est l'augmentation du volume de stockage (voir la Figure 2 :), il est à noter que cette étude date d'il y a dix ans.

Dans la Figure 1 nous voyons bien que durant ces dernières années le coût de stockage a baissé d'une manière impressionnante, et le volume d'informations utilisé ne cesse de croitre exponentiellement.

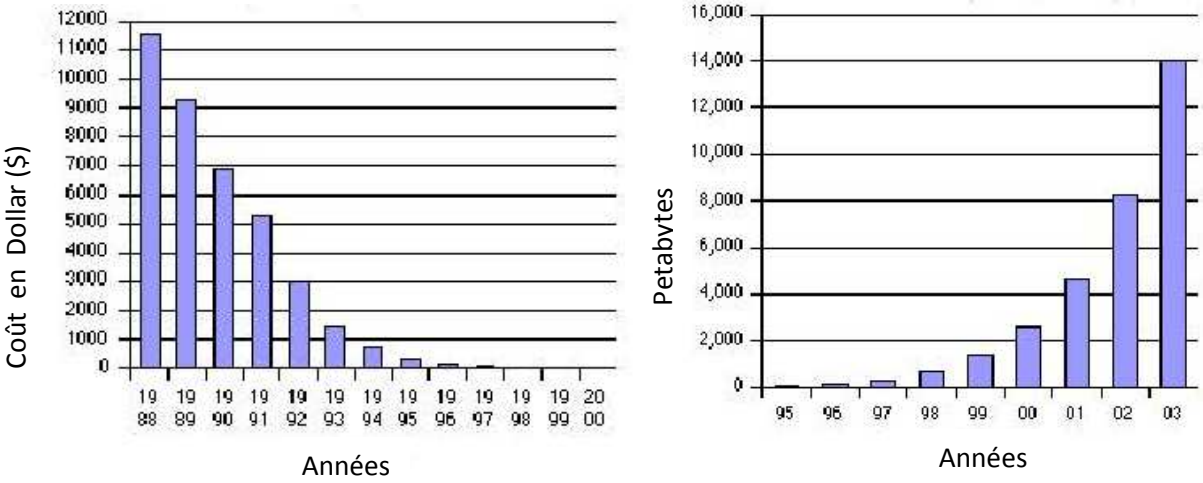


Figure 1: Réduction du coût de stockage, Croissance du taux de stockage 35]

Dans le même contexte, le Tableau 1 exprime bien la croissance du volume d'informations en chiffre pour l'année 1999 [35], il faut s'attendre à ce que ce volume augmentera d'une façon exponentiel ces dernières années.

Support de Stockage	Type du contenu	Estimation sup: Téraoctet / année	Estimation min: Téraoctet / année	Taux de croissance
Papier	Livres	8	1	2
	Journaux	25	2	-2
	Périodiques	12	1	2
	Documents officielles	195	19	2
	Sous-Total	240	23	2
Film	Photos	410,000	41,000	5
	Cinéma	16	16	3
	X-Rays	17,200	17,200	2
	Sous-Total	427,216	58,216	4
Optique	CD-Musique	58	6	3
	CD-Données	3	3	2
	DVD	22	22	100
	Sous-Total	83	31	70
Magnétique	Cassette Caméscope	300,000	300,000	100
	Disquette pilotes PC	766,000	7,660	100
	Serveurs départementaux	460,000	161,000	100
	Serveurs entreprise	167,000	108,550	55
	Sous-Total	2.120,539	635,480	50
Total		2.120,539	635,480	50

Tableau 1: Volume d'informations dans le monde, 1999. [35]

Cette croissance se fait ressentir plus dans les pays industrialisés là où les prises de décisions et la gestion puisent ses ressources et se base sur des systèmes d'informations.

Selon la source web[1] rien qu'EBay à lui seul traite un volume de données, plus de 50 pétaoctets[2] d'informations quotidiennes tout en ajoutant 40 téraoctets de données chaque jour, tandis que des millions de personnes vendent et achètent des éléments dans plus de 50 000 catégories.

[1] http://www.decideo.fr/eBay-et-Teradata-s-allient-pour-developper-des-solutions-approfondies-en-termes-d-analyse-des-donnees_a3723.html.

[2] 1 Pétaoctet = 10^3 Téraoctet = 10^6 Gigaoctet.

S'ensuit les sites de réseaux sociaux comme face book, et plein d'autres systèmes.

Aussi essentiels pour le secteur de recherche, que pour le secteur économique, les systèmes d'informations et les bases de données sont devenus quasi présent dans tous les domaines, là où il existe une organisation, un suivi, une administration, même pour des utilisations individuelle.

Il est devenu indispensable de se doter d'un système d'information ou d'une base de données qui gère l'ensemble des données, et c'est l'une des raisons qui a joué un très grand rôle dans la prolifération des systèmes d'informations, ce qui a ouvert un vaste marché fruiteux pour les concepteurs et éditeurs des systèmes d'information.

Les bases de données représentent la matérialisation des systèmes d'informations sur des supports électroniques.

Nous présentons dans la section suivante les bases de données et les systèmes logiciels qui les gèrent.

II.2. LES BASES DE DONNÉES

Par définition, une base de données est une collection ordonné et structuré représentant la mémorisation des données. Elle permet le stockage, la manipulation et la gestion des données.

C'est une sorte de projection structurée et ordonnée d'une réalité organisationnelle qui évolue dans le temps.

Les bases de données peuvent avoir une architecture centralisée ou distribuée, selon les nécessités et les besoins de l'organisation. Nous citons comme exemple la base de données du personnel.

« Historiquement, c'est en 1970, dans une thèse mathématique, qu'Edgar F.Codd (1923-2003), chercheur chez IBM, propose d'utiliser les informations sous forme d'enregistrements pour assurer les liens entre les informations et de regrouper les enregistrements dans des tables »[3].

Ce regroupement est motivé par le fait que le résultat de chaque recherche dans une base de données est une table. Le concept est repris par Lawrence Ellison qui créa une startup devenue Oracle Corporation.

Cette proposition est la base des modèles de données relationnelles, modèles utilisés par la quasi-totalité des systèmes de gestion de base de données actuels.

Dans ces modèles, tout comme dans les modèles réseaux, les entités sont reliées par des associations selon une organisation arbitraire.

[3] http://fr.wikipedia.org/wiki/Base_de_données

Cependant contrairement à ce dernier, les informations d'associations ne sont pas des pointeurs mais des informations contenues dans les enregistrements que le système de gestion de base de données utilise pour effectuer des produits cartésiens.

Le modèle entité-association a été inventé par Peter Chen en 1975 ; il est destiné a clarifié l'organisation des données dans les bases de données relationnelles.

Dans le modèle relationnel, la relation désigne l'ensemble des informations d'une table, tandis que l'association, du modèle entité-association, désigne le lien logique qu'il existe entre deux tables contenant des informations connexes.

Les premières bases de données étaient calquées sur la présentation des cartes perforées : répartis en lignes et colonnes de largeur fixe.

Une telle répartition permet difficilement de stocker des objets de programmation ; en particulier, elles ne permettent pas l'héritage entre les entités, caractéristique de la programmation orientée objet.

Apparues dans les années 1990, les bases de données objet-relationnel utilisent un modèle de données relationnel tout en permettant le stockage des objets.

L'un des modèles de base de données le plus utilisé est les bases de données de type OLTP.

OLTP (On Line Transaction Processing)[4] :

OLTP est une variété de base de données dont le mode de fonctionnement est transactionnel. L'objectif d'OLTP est de pouvoir insérer, modifier et interroger rapidement et en sécurité la base. Ces actions doivent pourvoir être effectuées très rapidement par de nombreux utilisateurs simultanément.

Chaque transaction opère sur de faibles quantités d'informations, et toujours sur les versions les plus récentes des données. Les modifications et les suppressions des données sont fréquentes dans OLTP.

Dans ces bases de données les associations d'héritage des objets s'ajoutent aux associations entre les entités du modèle relationnel.

II.3. SYSTÈME DE GESTION DE BASES DE DONNÉES

Le système de gestion de base de données (Data Base Management System) est une solution logicielle qui permet toutes les opérations sur les bases de données : nous citons par exemple la création de la base de données, le stockage, le déploiement, la sécurité, Tuning [5] et bien d'autres opérations.

[4] http://datawarehouse4u.info/OLTP-vs-OLAP.html

[5] Tuning: réglage des ressources et des paramètres physique de la base de données afin d'optimiser les performances de la base.

Un SGBD rend les informations disponibles et partagées pour différents usagers, ces derniers accèdent aux sources d'informations sur le SGBD via des applications qui effectuent l'interrogation de la base par les requêtes (SQL ou autres), le SGBD devient en quelque sorte un serveur pour répondre aux requêtes des utilisateurs.

Les nouvelles stratégies et orientations des entreprises exigent un accès garanti et universel aux données de l'organisation.

La disponibilité de l'information représente un grand intérêt pour les utilisateurs, et permet un rapprochement entre la décision à l'action.

Un SGBD doit garantir la bonne mémorisation des données ainsi que leur cohérence, aussi il doit permettre aux usagers de retrouver aisément les informations qu'ils cherchent en un temps acceptable.

Gérer des pétaoctets de données entre des milliers d'usagers en simultanéité pour différents lieus et garantir la cohérence et l'efficacité n'es pas une chose évidente.

Chaque opération engagée par l'utilisateur peut représenter une transaction financière ou autre informations d'une importance capitale, et malheur arrive si des erreurs ou des pertes se produisent.

Ce qui pourrait couter cher à l'organisme gérant, par conséquent, un SGBD ne doit tolérer aucune erreur, et garantir l'efficacité et la disponibilité des données.

De plus en plus les entreprises recourent pour le pilotage et la prise de décision aux systèmes d'aide à la décision basés sur l'analyse des données historiques de l'activité de l'entreprise, on appelle ces systèmes les entrepôts de données.

II.3.1. ENTREPÔT DE DONNÉES (DATAWAREHOUSE)

Les entrepôts de données sont des bases de collections des données, qui stockent les données par rapport à la dimension historique. L'analyse de données permet de récolter des informations utiles pour la prise de décision de l'entreprise.

Le concept des entrepôts de données peut être représenté par une centrale de collections de données, dont le traitement et l'analyse permet d'en extraire des informations et des connaissances pour aide à la décision, au moyen d'outils d'analyse et de synthèse (*data mining*).

Le concept des entrepôts de données a vu le jour en 1980 quand les chercheurs d'IBM Barry Devlin et Paul Murphy ont développé le " business data warehouse ". Le concept d'entrepôt de données avait pour but de fournir un modèle pour enregistrer et traiter le flux de données provenant de systèmes opérationnels pour approvisionner l'environnement d'aide à la décision.

Les entrepôts de données sont orientés sujets, leurs objectifs est de servir les besoins de l'entreprise, en fournissant un support pour l'aide à la décision.

En général les entrepôts de données gèrent des volumes de données assez important essentiellement dans des modèles logiques relationnels comme les schémas en étoile ou flocon de neige.

Ils sont composés de plusieurs tables de dimensions et une table centrale nommée table de faits, ces derniers sont accédés par des requêtes décisionnelles complexes caractérisées par de multiples opérations de sélection, de jointure et d'agrégation.

Il existe quatre modèles de schéma d'entrepôts de données le schéma étoile, schéma flocon de neige, schéma en galaxie, et schéma en constellation.

Souvent cité parmi les processus décisionnels les plus utilisés, les systèmes OLAP se basent et exploitent les entrepôts de données.

II.3.2. *OLAP* (ON LINE ANALYTICAL PROCESSING) [29]:

OLAP est un service qui se trouve généralement au dessus d'un entrepôt de données. L'entrepôt de données fournit l'infrastructure qui permet à OLAP de procurer des données analysées.

Il est possible que le processus OLAP puisse puiser ces sources de données à partir des bases de données de type OLTP.

Les entrepôts de données contiennent souvent des centaines de millions de lignes de données historiques. Répondre aux requêtes posées directement à l'encontre des données détaillées peut consommer des ressources informatiques importantes.

OLAP est caractérisé par de grands volumes d'informations traités, peu de mise à jour, et des insertions fréquentes.

Le but d'OLAP est de répondre aux requêtes rapidement de la grande quantité de données sous-jacentes. En général, les requêtes SQL posées sur les systèmes OLAP sont "group by" de certains attributs, et d'appliquer des fonctions d'agrégation contre d'autres attributs.

Par exemple, un gérant peut-être intéressé par la recherche du coût total, regroupé par année et par région.

La plupart des systèmes OLAP offre une représentation graphique des résultats (voir Figure 2 :).

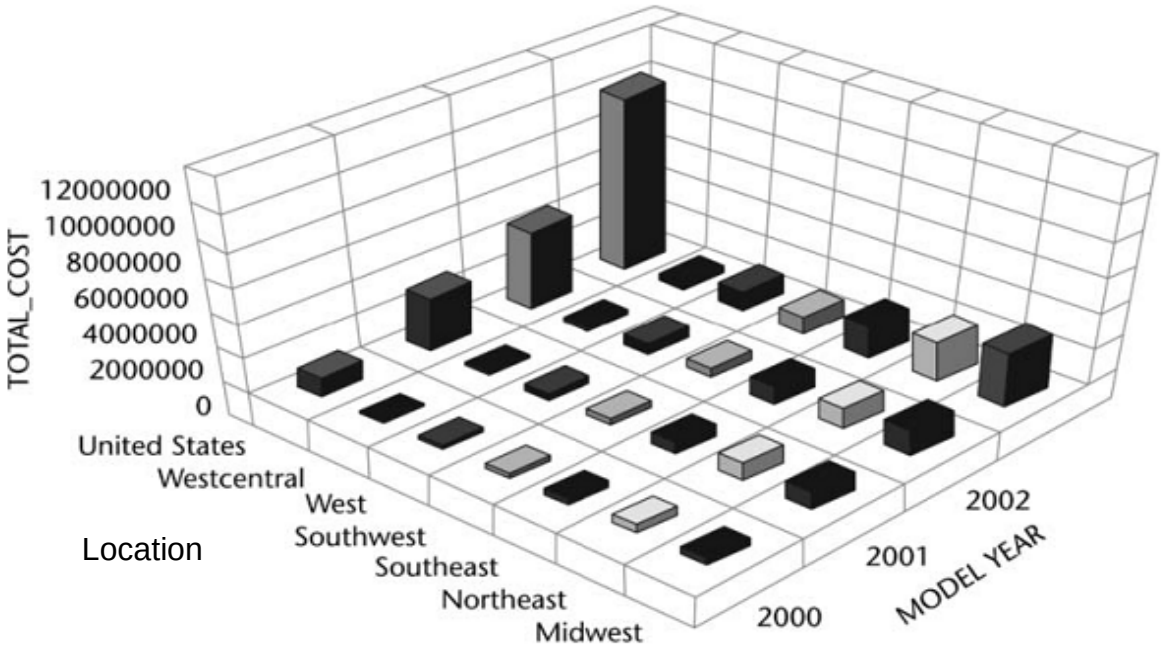


Figure 2 : rapport obtenu à partir d'un processus OLAP [4]

Le graphe à trois dimensions dans la Figure 2, représente un rapport obtenu d'un traitement à partir d'un système OLAP.

Il est constitué de trois axes qui sont : l'axe [MODEL YEAR] indique les résultats calculés et regroupés par année, l'axe [LOCATION] indique les résultats calculé et regroupés par ville, et le troisième axe [TOTAL_COST] indiquant le coût total par ville et par année.

II.3.3. MODE DE FONCTIONNEMENT OLAP

Les systèmes OLAP sont organisés d'une manière à faciliter la manipulation des données. L'ajout, la suppression ou le remplacement des dimensions est souple.

Les systèmes OLAP permettent d'explorer de grandes quantités de données facilement et rapidement. La plupart des systèmes OLAP se basent sur l'enregistrement des résultats sommaires sur des vues matérialisées.

A partir des vues matérialisées on extrait des données sommaires, elles sont généralement beaucoup plus petites que les tableaux contenant les informations détaillées. Quand une requête est posée, les vues matérialisées sont utilisées pour obtenir une réponse rapide, en évitant le calcul des résultats sur d'énorme quantité de données sous-jacentes.

[6] Source: Cognos PowerPlay.

Il existe trois catégories générales de mécanismes de stockage dans les systèmes OLAP: OLAP relationnel (ROLAP), OLAP multidimensionnel (MOLAP) et OLAP hybride (HOLAP).

ROLAP utilise des tables relationnelles pour stocker des données, typiquement en utilisant les dimensions comme une clé primaire composite. Chaque ligne représente une cellule du cube de données. Un cube est la table qui contient les données sommaires (les vues matérialisées).

Les cellules vides ne sont pas explicitement représentées dans ROLAP. MOLAP stocke les données sur le disque d'une façon organisée par rapport à l'emplacement dans la table. Chaque cellule est représentée par un emplacement fixe calculable.

Les cellules vides consomment de l'espace dans les représentations MOLAP. Il est possible d'utiliser des techniques de compression pour réduire l'espace perdu dans MOLAP.

Il est généralement reconnu que ROLAP est plus adapté sur les SGBD dont les données sont espacées et éparses, et MOLAP est meilleur lorsque les données sont denses.

Une stratégie utile pourrait consister à stocker les données éparses utilisant ROLAP, et les données denses en utilisant MOLAP. Cette approche hybride est appelé HOLAP.

L'objectif des bases de données OLTP est de fournir des réponses rapides à des requêtes simples, exemple : les ventes des produits cosmétiques.

Les bases OLAP permettent des requêtes plus complexes à qui s'ajoute la notion de périodes: les ventes des produits Cosmétiques par vendeur, région et par mois.

La conception physique des bases de données représente l'un des maillons dans le cycle de vie d'une base de données, elle inclut la sélection d'index, de partition, de cluster, etc.

Le cycle de vie d'une base de données intègre quatre étapes : définition des besoins ou cahier de charge, étape de la conception logique, étape la conception physique, et enfin la mise en œuvre et la réalisation de la base de données.

La conception physique de base de données débute après l'étape de la définition et la normalisation des tables, et permet l'optimisation des performances de la base de données.

Une explication plus détaillée sur la conception physique est présentée dans la section qui suivra.

II.4. LA CONCEPTION PHYSIQUE

La conception physique des bases de données a pour rôle la gestion des méthodes de stockage et d'accès physique des tables sur disque, ce qui permet à la base de fonctionner avec une grande efficacité.

L'objectif de la conception physique consiste à optimiser les performances de la base de données et des applications fonctionnant sur cette base.

La conception physique sur les bases de données est une rude tâche, que les gestionnaires et les administrateurs de bases de données maitrisent peu. Elle implique des dizaines et souvent des centaines de variables et contraintes, qui sont difficiles à gérer et à régler, sachant bien que ces modifications sont très souvent liées aux différentes architectures proposées.

L'intervention sur les différentes architectures et configurations de la conception physique d'une base de données se répercute sur les performances de cette dernière.

Les calculs individuels de performance basée sur un mécanisme d'indexation donnée ou d'un algorithme de fragmentation peuvent prendre plusieurs heures à la main, et l'analyse de la performance est souvent basée sur la comparaison de nombreuses configurations différentes et des conditions de charge, ce qui nécessite beaucoup de calculs et le résultat n'est pas toujours garanti.

L'étude de Lyman et Varian [35] à révéler que le taux de stockage ne cesse de croitre, ce qui influe négativement sur les performances des bases de données, d'où la nécessité d'une intervention permanente du gestionnaire de la base de données pour optimiser la conception physique et les performances de la base.

Dans cette étude nous nous somme intéressé à l'optimisation des performances de la base de données par la conception physique.

Plusieurs techniques de conceptions physiques existent dont nous citons l'indexation, les vues matérialisées, fragmentation, etc.

II.4.1. TECHNIQUES D'OPTIMISATIONS POUR LA CONCEPTION PHYSIQUE [36]

Index : un index est une organisation de données mis en place pour accélérer la recherche (requête) des données de tables. Dans les systèmes de gestion de base de données, les index peuvent être spécifiés par les programmeurs d'applications de base de données en utilisant les commandes SQL[7].

Il existe plusieurs sortes d'index dont nous citons les index uniques, index binaire, index de jointure, index de jointure binaire, etc.

[7] Structured Query Language

Vues Matérialisées : lorsque une ou plusieurs tables sont interrogés, le résultat peut être stocké dans ce qu'on appelle une vue matérialisée. Les vues matérialisées sont stockées sous forme de tables dans la base de données comme n'importe quelle autre table.

Fragmentation : dans notre travail nous nous somme intéressé à la technique de fragmentation, c'est pour cette raison que nous avons consacré la section « *Techniques de fragmentations* » pour l'étudiée en détail.

Dans les entrepôts de données, des vues matérialisées sont maintenus comme des agrégats de données dans les tables de base.

Clustering : le Clustering multidimensionnel (MDC) est une technique par laquelle les données peuvent être regroupés selon les dimensions, tels que l'emplacement, le calendrier, ou type de produit.

En particulier, MDC permet aux données d'être regroupés selon plusieurs dimensions en même temps, comme les patins à glace vendue dans le Wisconsin durant le mois de Décembre. Les pôles visent à tirer parti de la charge de travail connus et prévus sur ces données.

Autres techniques :

Il existe d'autres techniques d'optimisation de la conception physique pour rendre l'accès aux données plus efficaces sur une base de données.

A titre d'exemple, la compression de données est une technique qui permet de stocker plus de données dans espace disque réduit, les données comprimées sont donc accessibles plus rapidement.

Une autre façon d'améliorer la fiabilité de base de données comprend des techniques comme la réplication des données (*mirroring*), dans lequel les données sont dupliquées sur plusieurs disques ou plusieurs sites.

L'inconvénient de la redondance est d'avoir à mettre à jour plusieurs copies des données chaque fois qu'un changement est nécessaire dans la base de données, ainsi que l'espace de stockage supplémentaire nécessaire. L'espace de stockage est devenu bon marché, mais le temps ne l'est pas.

Dans notre travail nous nous somme intéressés aux techniques de fragmentation des données, plus particulièrement à la fragmentation verticale.

Dans ce qui suit nous présenterons les techniques de fragmentation sur les bases de données.

II.4.2. TECHNIQUES DE FRAGMENTATIONS

La fragmentation est une technique qui consiste à décomposer les objets de la base (index, table, vue matérialisée, …) en de sous ensembles pour permettre une exploitation meilleure des données et optimiser le temps d'exécution de la charge de travail[8].

Pour qu'une fragmentation soit correcte il faut qu'elle respecte les règles suivantes :

- Complétude : la fragmentation doit être complète, donc chaque élément du schéma doit au moins appartenir à un fragment du sous schéma.
- Reconstructible : après avoir fragmenté physiquement un schéma donné, on doit pouvoir recomposer le schéma initial à partir des fragments générés.

II.4.3. INTÉRÊTS DE LA FRAGMENTATION

La fragmentation sur les bases de données nous permet de gagné sur plusieurs plans, elle avantage une amélioration de la performance du système, la fragmentation rend les données disponibles pour les bases de données distribuées, en favorisant les accès locaux.

Aussi, en appliquant la fragmentation, la requête n'accède qu'aux données requises pour son exécution, cela permet de réduire énormément le nombre d'accès physique aux fichiers de données et de réduire les traitements et les calculs au niveau du processeur, ce qui permet à la fois d'augmenter le débit du système et réduit l'exécution des requêtes.

Aussi, cela permet à quelques transactions concurrentes de s'exécuter parallèlement, puisque chaque transaction accède à un fragment séparé physiquement de l'autre.

Grace à la fragmentation la capacité et le coût de stockage deviennent plus équilibrés et moins condensé, ce qui permet une plus grande disponibilité des données. Cela est particulièrement adaptée aux architectures de disques comme le RAID (réseaux redondants de disques indépendants) où les données peuvent être accessibles en parallèle sur plusieurs disques d'une manière organisée.

Un système fragmenté est beaucoup plus tolérant aux pannes, il est robuste et moins vulnérable, car si un fragment d'un objet de la base est endommagé, le reste des fragments n'en serra pas affecté.

La fragmentation offre un autre avantage pour le cas des bases de données distribuées, elle permet la réduction des transactions de communication entre les

[8] Ensemble des requêtes à laquelle la base de données elle est soumise

sites, qui est d'une manière générale assez considérable, ce qui signifie que les performances de la base et le temps de réponse seront meilleurs.

II.4.4. INCONVÉNIENTS DE LA FRAGMENTATION :

La fragmentation sur les bases de données nous apporte des avantages mais aussi des inconvénients, il est à considérer que le schéma de fragmentation se base sur l'analyse de la base de données mais aussi sur la charge de travail, et dans un environnement où les requêtes changes fréquemment il est déconseillé de modifié la conception physique de la base de données à chaque variation de la charge de travail.

La fragmentation d'une base de données est une arme à double tranchant, en effet, elle peut jouer un rôle positif sur la performance de la base si elle bien implémenter, mais elle peut jouer un rôle négatif si la fragmentation n'es pas adéquate au système. C'est pour cette raison qu'il faut se doter d'outils performants pour l'assistance à la fragmentation.

Il faut prendre en considération qu'une fois la base de données fragmenter le schéma de la base sera changé, et le chemin pour l'accès aux données changera, c'est pour cette raison qu'il faut penser à reproduire les clés primaire des relations, préserver l'intégrité référentielle entre les relations en dupliquant sur chaque fragment les clés étrangère, et enfin réécrire les requêtes selon la nouvelle conception.

Il est à noter que pour certaine bases de données supportant la fragmentation ces indications seront prises en charge par le SGBD, qui se chargera de ces indications.

Enfin, pour le cas des bases de données distribuées, il peut s'ajouter le problème d'allocation, le bon fragment au bon endroit (site), et là aussi il est conseillé de se doter d'algorithmes et des outils nécessaires pour mener à bien cette opération.

II.4.5. TYPES DE FRAGMENTATION

Il existe trois variantes de fragmentation sur les bases de données :

- La fragmentation horizontale : fragmenter la relation (table de la base) par rapport à ses lignes (enregistrement),
- La fragmentation verticale : fragmenter une relation par rapport à ses colonnes (attributs),
- Fragmentation mixte : une combinaison entre la fragmentation horizontale et la fragmentation verticale.

II.4.5.1. FRAGMENTATION HORIZONTALE

La fragmentation horizontale décompose un objet de la base de données (relation, vue matérialisée, index) en des sous ensembles d'une manière horizontale dans le sens des enregistrements, chaque fragments représente une sélection selon un prédicat.

Les fragments étant le regroupement d'un sous ensemble d'enregistrements de l'objet initial.

L'opération union permet reconstruire le schéma à partir de fragments horizontaux.

Il existe deux variantes de fragmentation horizontale :

- Fragmentation horizontale primaire : on effectue la fragmentation horizontale sur la base de prédicats de sélection définis sur la relation ou l'objet lui même.
- Fragmentation horizontale dérivée : dans cette variante, la décomposition est effectuée sur une relation R en utilisant des prédicats définis sur une autre Relation R'.

L'exemple suivant nous permettra de bien comprendre la fragmentation horizontale d'une table.

(1) ILLUSTRATION :

Soit la table Client (Code, Nom, Prénom, Genre, Coordonnés), sur la base du genre féminin ou masculin du client.

Soit les requêtes suivantes :

Q1: **Select** * **from** Clients **where** Genre = F;

Q2: **Select** * **from** Clients **where** Genre = M;

Pour exécuter ces deux requêtes le system doit parcourir toute la table client, et vérifier sur chaque table le genre F ou M, ce qui nécessite beaucoup d'opérations et d'accès physique aux fichiers de la base.

Une des solutions qui se présente à l'administrateur de la base est de fragmenter horizontalement la relation Client selon les prédicats de sélection :

(Genre = F) & (Genre = M).

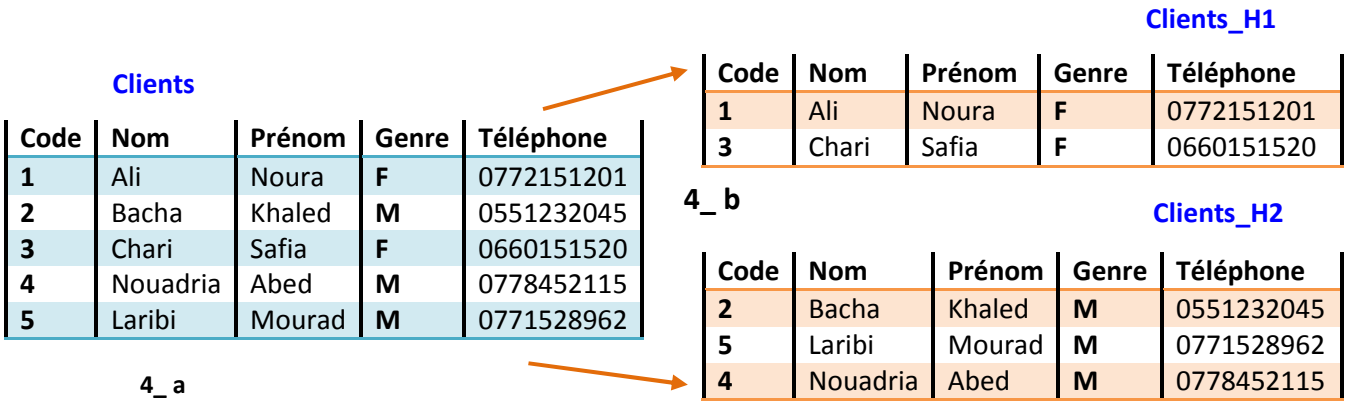


Figure 3 : illustration de la fragmentation Horizontale.

La figure 4_a, représente une instance de la relation clients initiales.

Dans la figure 4_b, les deux tables Clients_H1 et Clients_H2 représentent les deux fragments générés sur la base de la table Client.

Une fois la table partitionnée, le SGBD réécrit les requêtes pour qu'elles puissent s'exécutées correctement sur le nouveau schéma et les fragments qui lui sont valides[9], les deux requêtes seront réécrites comme suit :

Q1: **select** * **from** Clients_H1;

De cette manière nous éviterons le parcours entier de la table client, pour l'exécution des requêtes Q1, ce qui signifie un gain en temps de parcours de la table, et ce qui réduit énormément le temps d'exécution des requêtes.

II.4.5.2. FRAGMENTATION VERTICALE

La fragmentation verticale d'une table T permet de décomposer la relation T en un ensemble de fragments (T1,…, Tn) regroupant chacun un ensemble d'attributs. Pour chaque fragment la clé primaire de la table T doit être reproduite (afin de repérer les tuples[10] dans les sous fragments). Les fragments verticaux sont obtenus en utilisant l'opération de projection de l'algèbre relationnelle.

L'opération de jointure permet de reconstruire le schéma à partir de fragments verticaux, mais il est nécessaire que pour chaque fragment vertical généré a partir d'une relation R il va falloir dupliquer la clé primaire de la relation R.

Pour mieux comprendre la fragmentation verticale sur les bases de données, nous vous proposons de consulter l'illustration suivante.

[9] Un fragment est valide pour une requête Q si et seulement si Q accède au moins à un n-uplet du fragment.

[10] Instance d'une relation.

(2) ILLUSTRATION :

Dans cet exemple nous traiterons le concept de la fragmentation verticale sur la table Client, le but par l'application de la fragmentation verticale est de réduire les coûts (temps de réponse) des requêtes Q1' et Q2', avec :

Clients (Code, Nom, Prénom, Genre, Coordonnés)

Q1 ': **select** Code, Nom, Prénom **from** Clients ;

Q2' : **select** Code, Genre, Coordonnés **from** Clients ;

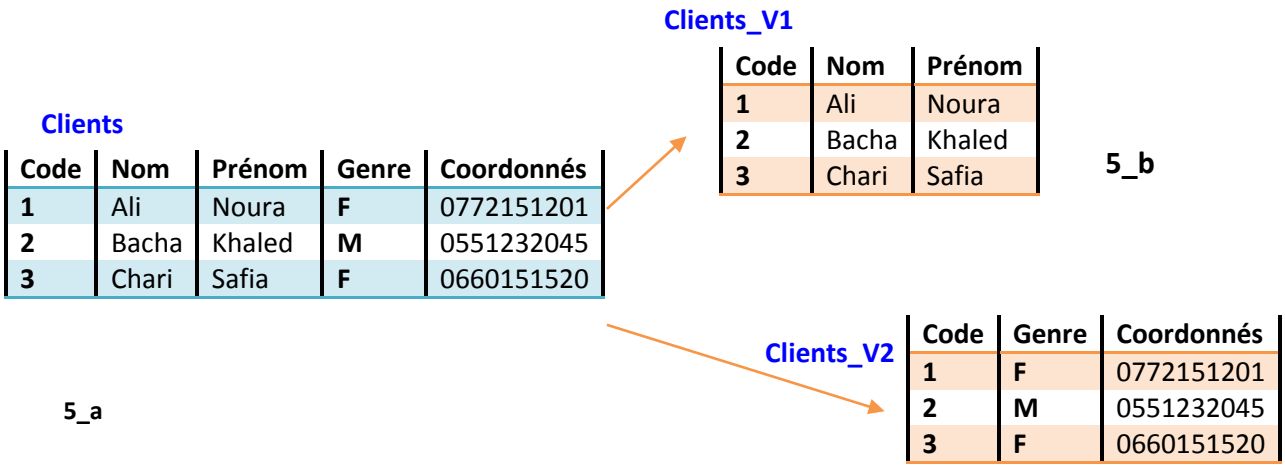


Figure 4 : Illustration de la fragmentation verticale

La Figure 4_a, représente une instance de la table Clients. Le SGBD lors de l'exécution des requêtes Q1' et Q2' parcours toute la table, et des colonnes non utile aux requêtes seront chargées.

En fragmentant la table Clients verticalement (Figure 4_b), le SGBD ne chargera que les données utiles à l'exécution des requêtes, ce qui réduit le temps d'exécution des requêtes et l'accès physique aux tables.

Nous devons réécrire les requêtes pour qu'elles puissent s'exécutées correctement sur le nouveau schéma.

Q1': **select** * **from** Clients V1;

Q2': **select** * **from** Clients V2;

La fragmentation verticale sur les tables relationnelles est complexe. Ceci est essentiellement dû au grand nombre d'alternatives possibles :

- En partitionnant horizontalement: Özsu et al [34] on montré que pour N prédicats simple, le nombre de fragments possibles est 2^N; dans le cas où les prédicats contradictoires existes se nombre diminue par l'élimination ces dernières.

– Par contre si une table R à m attributs non clés primaires, l'ensemble des possibilités de fragmentation est calculé par $B(m) \approx m^m$, pour de grande valeur de m (Hammer & Niamir [25]).

B(M) étant le nombre de Bell, les valeurs suivantes nous permettent d'avoir une idée sur la grandeur du nombre de Bell où :

$m = 5 ; \quad B(m) \approx 52.$

$m = 10 ; \quad B(m) \approx 115{,}000.$

$m = 15 ; \quad B(m) \approx 1{,}38*10^{9}.$

$m = 22 ; \quad B(m) \approx 4{,}50*10^{15}.$

$m = 30 ; \quad B(m) \approx 8{,}46*10^{23}.$

Ainsi, pour un schéma de base de données composée de n relations, le nombre de fragments possibles devient très important, et il dit prendre en considération les fragments de chaque relation le nombre de possibilité est donnée par : B(A1)*B(A2)… B(An), donc pour un schéma de 10 relations, chacune de 15 attributs, le nombre possible de fragments est de $(10^{10})^9$.

Le nombre de fragments verticaux possibles augmente avec des grandeurs exponentielles, ce qui rend presque impossible l'évaluation du meilleur schéma de fragmentation pour les tables composées d'un nombre important de colonnes.

Par conséquent, obtenir le meilleur schéma de fragmentation verticale par le calcul et l'évaluation de toutes les combinaisons n'es pas évident, c'est pour cette raison que nous serions obligé de recourir à des heuristiques pour obtenir des solutions performantes avec des temps raisonnables.

Nous présentons dans la section suivante, une étude détaillée sur des travaux proposés pour sélectionner un schéma de fragmentation verticale, ainsi qu'une étude sur les outils disponibles sur le marché pour assister les administrateurs de bases de données dans la conception physique.

II.5. TRAVAUX SUR LA FV

La fragmentation verticale sur les bases de données a fait l'objet de plusieurs travaux pour des différents contextes.

Tous les algorithmes proposés dans ces études convergent vers l'optimisation des performances de la base de données, mais les approches et les orientations des chercheurs diffèrent, et peuvent être classées en trois types.

(1) Algorithmes basée sur les affinités entre les attributs : Cet algorithme calcul de l'affinité entre les attributs, et regroupe les attributs les plus proches dans la même partition, plusieurs auteurs ont étudié cette approche Hoffer & al [26], Navathe et al [32], Navathe et Ra [33].

(2) Algorithmes basés sur le modèle de coût : Effectuer l'évaluation de chaque solution (schéma de fragmentation) selon un modèle de coût qui estime le temps de réponse des applications et requête s'exécutant sur la base de données, parmi les travaux qui ont suivi cette voie : Hammer et Niamir, [25], Cornell et Yu [18], Chu [17].

(3) Algorithmes basés sur la fouille de données : Les travaux de Cheng et al. [16], Travaux de Gorla [23], Song et Gorla [24] se sont basés sur cette méthode, il procède par la génération des partitions en se basant sur les méthodes de fouille de données.

II.5.1. APPROCHE BASÉE SUR MATRICE D'AFFINITÉ

Nous allons développer dans la section suivante le principe de la matrice d'affinité et les quelques recherches basées dessus :

II.5.1.1. PRINCIPE DE LA MATRICE D'AFFINITÉ

Plusieurs recherches dans le domaine de fragmentation verticale des bases de données proposent d'utiliser la matrice d'affinité entre les attributs, en commençant par la matrice d'utilisation : pour le cas de n attributs, la matrice d'affinité (MA) est une matrice carré n x n, chaque valeur dans la matrice MAij représente le facteur d'affinité entre les attributs Ai et Aj, et qui est calculé par le nombre d'accès des transactions référençant les deux attributs Ai et Aj.

Les fréquences d'accès des requêtes sont prises en considération afin de regrouper les attributs dans les mêmes fragments accédés fréquemment et simultanément. Soit P_{ij} La probabilité que les attributs a_i et a_j soient nécessaires pour la même requête. Une fonction de coût fondée sur cette hypothèse est déduite, cette fonction reflète la quantité prévue de données qui doivent être transmises afin de répondre à la requête. L'objectif ici est de choisir une partition qui minimise cette fonction de coût.

Les algorithmes basée sur la MA commencent par une matrice d'usage des attributs (AUM). AUM est une matrice, qui a des attributs pour colonnes, et requêtes pour lignes et fréquence d'accès des requêtes comme valeurs dans la matrice. La plupart des algorithmes de fragmentation emploient une matrice d'affinité des attributs (AAM) dérivée de l'AUM fourni comme entrée.

Les résultats des différents algorithmes (Hoffer & al [26], Navathe et al [32]) sont parfois différents même pour la même matrice d'affinité des attributs puisque les fonctions objectives employées par ces algorithmes sont différentes.

La majeure partie des algorithmes de fragmentation verticale basés sur l'affinité n'a pas un mécanisme pour évaluer la « qualité » des partitions générées [1].

L'affinité entre les attributs mesure le lien entre deux attributs d'une relation en prenons en compte comment ils sont accédés par l'application.

Exemple : Dans ce qui suit nous allons donnée un exemple sur la l'application de l'algorithme basé sur l'affinité sur une table (relation) d'une base de donnée.

Soit la table R composée de 10 attributs R :{A1, A2, A3, A4, A5 ,A6, A7, A8 ,A9 ,A10} , les transactions (requêtes) {T1,T2, T3, T4 ,T5 ,T5 ,T6 ,T7 ,T8} s'exécutent sur cette la base de données englobant cette relation avec des fréquences indiquées sur la colonne fréquence sur le Tableau 2.

Transactions	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Fréquence
T1	1	0	0	0	1	0	1	0	0	0	25
T2	0	1	1	0	0	0	0	1	1	0	50
T3	0	0	0	1	0	1	0	0	0	1	25
T4	0	1	0	0	0	0	1	1	0	0	35
T5	1	1	1	0	1	0	1	1	1	0	25
T6	1	0	0	0	1	0	0	0	0	0	25
T7	0	0	1	0	0	0	0	0	1	0	25
T8	0	0	1	1	0	1	0	0	1	1	15

Tableau 2 : Matrice d'usage

La matrice d'usage est représentée sur le Tableau 2, la valeur 1 sur la matrice indique que la transaction sur la ligne requiert l'attribut de la colonne, la valeur 0 indique que la transaction ne requiert pas l'attribut sur la colonne.

L'affinité entre deux attributs Ai et Aj est définie par :

$$Aff_{ij} = \sum_{t=1}^{T} q_{t,ij} \quad \textbf{Équation 1}$$

Où $q_{t,ij}$ représente le nombre d'accès des transactions t référençant à la fois i et j.

En appliquant la formule de calcul des affinités entre les attributs sur la matrice d'usage du Tableau 2, nous obtenons en résultat la matrice d'affinité représentée dans le Tableau 3.

Attributs	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A1	75	25	25	0	75	0	50	25	25	0
A2	25	110	75	0	25	0	60	110	75	0
A3	25	75	115	15	25	15	25	75	115	15
A4	0	0	15	40	0	40	0	0	15	40
A5	75	25	25	0	75	0	50	25	25	0
A6	0	0	15	40	0	40	0	0	15	40
A7	50	60	25	0	50	0	85	60	25	0
A8	25	110	75	0	25	0	60	110	75	0
A9	25	75	115	15	25	15	25	75	115	15
A10	0	0	15	40	0	40	0	0	15	40

Tableau 3 : Matrice d'affinité.

A titre d'exemple l'affinité entre les attributs A9 et A2 est calculée en appliquant la formule (1), application :

$$Aff(9,2) = Aff(2,9) = T2 + T5 = 50 + 25 = 75.$$

Une fois la matrice d'affinité obtenue, la prochaine étape consiste à dériver une nouvelle matrice ordonnée, sur la base de la matrice d'usage, et d'en dériver les ensembles conjoints et présentant une affinité proche. Ces derniers représenteront les partitions.

Il existe plusieurs algorithmes qui permettent de générer la matrice ordonnées, nous en citons l'algorithme Bond Energie.

II.5.1.2. L'ALGORITHME DE BOND ÉNERGIE (BEA) [32]:

Cet algorithme est employé pour grouper les attributs d'une relation sur la base des valeurs d'affinités entre les attributs dans l'AAM. On le considère approprié pour plusieurs raisons.

- Il est conçu particulièrement pour déterminer des groupes d'items semblables (C.A.D Il groupe les attributs qui ont une grande d'affinité ensemble, et ceux avec de petites valeurs ensemble).
- Les groupements finaux sont peu sensibles à l'ordre dans lequel les items sont exposés à l'algorithme.
- AAM est symétrique, ce qui permet la permutation par paires des lignes et des colonnes, qui réduit la complexité.
- La complexité de l'algorithme est raisonnable, de l'ordre $O(n^2)$, où n est le nombre d'attributs.

Cet algorithme prend comme entrée la matrice d'affinité d'attribut, permute ses lignes et colonnes et produit une matrice d'affinité groupée (CAM). La permutation est faite de telle manière à maximiser la mesure globale d'affinité (AM).

La génération de la matrice d'affinité ordonnée se fait en trois étapes :

Initialisation: Placer et fixer une des colonnes AAM arbitrairement dans CAM.

Itération: Sélectionner chacune des colonnes n - i restantes (où i est le nombre de colonnes déjà placées dans CAM) dans les positions i + 1 restantes dans la matrice CAM. Choisir le positionnement qui apporte la plus grande contribution à la mesure globale d'affinité. Continuer ceci jusqu'à ce qu'il ne reste plus de colonnes à placer.

Tri des lignes : Une fois que le rangement des colonnes est déterminé, le placement des lignes devrait également changé de sorte que leurs positions relatives suivent les positions relatives aux colonnes.

La formule qui permet d'évaluer la bonne position est la suivante :

$$\boldsymbol{cont(A_i, A_i, A_k) = 2bond(A_i, A_j) + 2bond(A_j, A_k) - 2bond(A_i, A_k)} \quad \textbf{Équation 2}$$

Avec : $$\boldsymbol{bond(A_x, A_y) = \sum_{z=1}^{n} aff(A_z, A_x) * aff(A_z, A_y)} \quad \textbf{Équation 3}$$

Après avoir appliqué l'algorithme Bond Energy sur la matrice du Tableau 3 : Matrice d'affinité., on obtient la matrice ordonnée suivante :

Attributs	A5	A1	A7	A2	A8	A3	A9	A10	A4	A6
A5	75	75	50	25	25	25	25	0	0	0
A1	75	75	50	25	25	25	25	0	0	0
A7	50	50	85	60	60	15	25	0	0	0
A2	25	25	60	110	110	75	75	0	0	0
A8	25	25	60	110	110	75	75	0	0	0
A3	25	25	25	75	75	115	115	15	15	15
A9	25	25	25	75	75	115	115	15	15	15
A10	0	0	0	0	0	15	15	40	40	40
A4	0	0	0	0	0	15	15	40	40	40
A6	0	0	0	0	0	15	15	40	40	40

Tableau 4 : Matrice d'affinité ordonnée.

Une fois que la matrice ordonnée est obtenue, plusieurs ensembles d'attributs sont formés, et plusieurs partitions sont possibles, pour dérivé les meilleurs partitions possible on peut utiliser l'algorithme de fragmentation verticale binaire [33].

II.5.1.3. L'ALGORITHME DE FRAGMENTATION VERTICALE BINAIRE [1]:

Le but de cet algorithme est de regrouper les attributs ayant une haute affinité. Le principe du regroupement consiste à trouver un point dans la matrice de façon à diviser la matrice en deux ensembles, par la permutation des colonnes et des lignes de la matrice, jusqu'à l'obtention d'une matrice ordonnée.

Si *At* est l'ensemble des attributs utilisés par la transaction *t*, il est alors possible de calculer les ensembles suivants:

$$\boldsymbol{T = (t \,/\, t\ est\ une\ transaction)}$$

$$\boldsymbol{LT = (t/At \in L)} \qquad \textbf{Équation 4}$$

$$\boldsymbol{UT = (t/At \in U)} \qquad \textbf{Équation 5}$$

$$\boldsymbol{IT = T - (LT \cup UT)} \qquad \textbf{Équation 6}$$

T représente l'ensemble de toutes les transactions. *LT* et *UT* représentent l'ensemble des transactions qui sont concernées par la fragmentation, comme ils peuvent être entièrement traités en utilisant les attributs de la partie inférieur *L* ou supérieur *U*.

$$\boldsymbol{CT = \sum t \in T\ qt} \qquad \textbf{Équation 7}$$

$$\boldsymbol{CL = \sum t \in LT\ qt} \qquad \textbf{Équation 8}$$

$$\boldsymbol{CU = \sum t \in UT\ qt} \qquad \textbf{Équation 9}$$

$$\boldsymbol{CI = \sum t \in IT\ qt} \qquad \textbf{Équation 10}$$

CT compte le nombre total d'accès des transactions à l'objet considéré. *CL* et *CU* compte le nombre total d'accès de transactions qui ont besoin d'un seul fragment; *CI* compte le nombre total d'accès de transactions qui ont besoin de deux fragments. D'une manière générale n-1 emplacements possibles du point (X) le long de la sont considérés, où n est la taille de la matrice (c.-à-le nombre d'attributs). Une partition non recouvrant est obtenue en sélectionnant le point (X) le long de la diagonale de telle sorte que la fonction objectif suivante z soit maximisée:

$$\boldsymbol{\max Z = CL * CU - CI^2} \qquad \textbf{Équation 11}$$

La partition qui correspond à la valeur maximale de la fonction (*Z*) est acceptée si la fonction (*Z)* est positive et a rejeté dans le cas contraire. La fonction ci-dessus provient d'un jugement objectif empirique de ce qui devrait être considéré comme une "bonne" partitionnement. La fonction est croissante dans *CL* et *CU* et décroissante dans

CI. Pour une valeur donnée de *CI*, il sélectionne *CL* et *CU* de telle façon que le produit *CL* * *CU* soit maximisée.

Cela se traduit par la sélection de valeurs pour *CL* et de *CU* qui sont aussi proches qu'égales que possible. Ainsi, la fonction ci-dessus (*Z)* produira des fragments qui sont «équilibrées» à l'égard des transactions.

Cet algorithme pose l'inconvénient de ne pas être en mesure de partitionner un bloc en présence d'un autre bloc inclut.

Attributs	A5	A1	A7	A2	A8	A3	A9	A10	A4	A6
A5	75	75	50	25	25	25	25	0	0	0
A1	75	75	50	25	25	25	25	0	0	0
A7	50	50	85	60	60	15	25	0	0	0
A2	25	25	60	110	110	75	75	0	0	0
A8	25	25	60	110	110	75	75	0	0	0
A3	25	25	25	75	75	115	115	15	15	15
A9	25	25	25	75	75	115	115	15	15	15
A10	0	0	0	0	0	15	15	40	40	40
A4	0	0	0	0	0	15	15	40	40	40
A6	0	0	0	0	0	15	15	40	40	40

X

Tableau 5: Matrice d'affinité ordonnée divisée.

Le Tableau 5 représente la matrice d'affinité ordonnée divisée par le point X.

La matrice du Tableau 5 est divisée en deux parties par rapport au point X, deux fragments sont générés par cette opération : **{5, 1, 7, 2, 8, 3, 9}, {10, 4, 6}.**

L'objectif du partitionnement est de maximiser les nombre d'accès à un seul fragment, et minimiser les accès à plusieurs fragments pour la même transaction (requête).

Dans le cas où le nombre des attributs est important, il sera nécessaire de réitérer l'opération de regroupement sur les fragments obtenus, ou trouver plusieurs points de division sur la matrice.

Nous déduisant bien que pour une relation avec un nombre important d'attributs la génération des fragments verticaux par l'approche d'affinité, nécessite beaucoup de calculs.

II.5.1.1. GRAPHE D'AFFINITÉ [1]:

L'approche basée sur le graphe d'affinité débute par la construction de la matrice d'affinité, sur la base de cette matrice, un graphe d'affinité est construit.

Chaque arête du graphe est marquée par son poids, qui représente l'affinité entre ses sommets. Les sommets représentent les attributs de la relation, l'affinité entre les sommets représente le nombre de requêtes dans lequel les attributs sont requis simultanément.

La deuxième étape consiste à formé un arbre couvrant linéaire. Les cycles représentent les partitions candidates.

Reprenons l'exemple du Tableau 4 : Matrice d'affinité ordonnée. La figure 1 montre le résultat de l'application de l'algorithme à du graphe d'affinité. Dans la figure 1, les nœuds représentent les attributs de la relation.

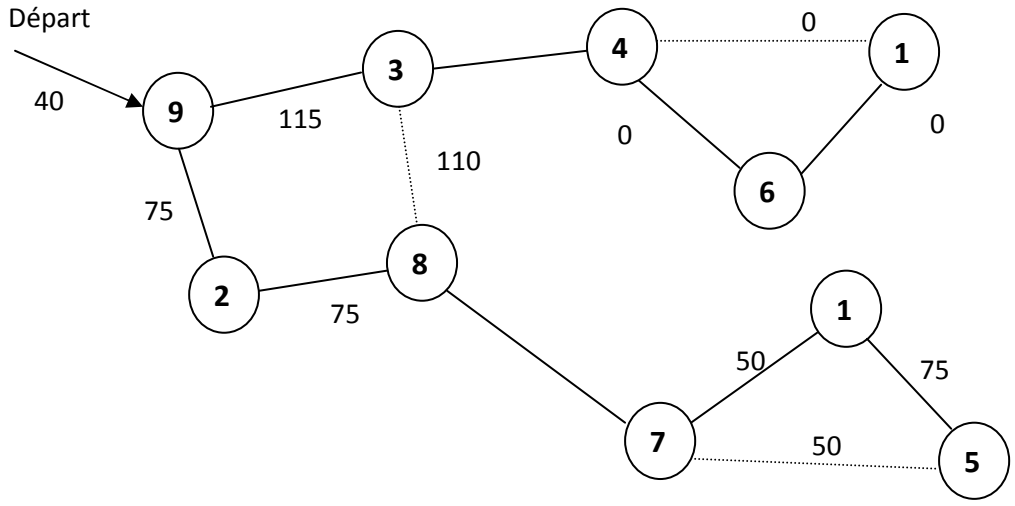


Figure 5 : graphe d'affinité.

On en déduit le schéma de fragmentation suivant :

1. (A1, A5, A7) 2. (A2, A3, A8, A9) 3. (A4, A6, A10)

Dans la section suivante nous allons présenter les travaux et études sur la fragmentation verticale basée sur l'approche de la matrice d'affinité.

II.5.1.2. LITTÉRATURE SUR LA MATRICE D'AFFINITÉ

Hoffer & al [26] ont été les premiers à utiliser la matrice d'affinité dans la fragmentation verticale des bases de données.

Les auteurs dans [26] ont employés l'affinité entre les attributs pour la fragmentation verticale, qui est basée sur le regroupement des attributs ayant un nombre d'accès disque rapproché.

Cette méthode se base sur le calcule de l'affinité entre les attributs, ensuite il effectue le regroupement des attributs dont l'affinité est la plus forte.

Cette méthode permet de réduire au minimum les coûts de récupération et de mise à jour, l'algorithme est assez simple à implémenter, il est constitué de trois étapes :

- Génération de la matrice d'usage,
- Génération la matrice d'affinité des attributs,
- Génération des fragments par l'application de l'algorithme de bond énergie.

Navathe et al [32] ont étendue l'approche BEA en proposant un algorithme qui produit des fragments disjoints et non disjoints.

Cette approche améliore et réduit le nombre de fragments accédés par transaction en prenant en compte le coût de stockage.

Cette méthode consiste en deux étapes :

1. Dans la première étape les paramètres d'entrées sont rangés dans une matrice d'usage par laquelle on va construire la matrice d'affinité des attributs, qui nous permet de générer les propositions de fragments verticales.
2. Dans la seconde étape on prend en considération les coûts de stockage afin d'affiner les résultats.

L'étude [32] discute la fragmentation verticale dans trois contextes : bases de données centralisées stockées dans un seul niveau de mémoire ; base de données centralisées stockés dans différents niveau de mémoire ; et base de données distribuées.

La première étape est identique pour les trois types de bases de données, pour la deuxième étape les modèles de coût seront calculés différemment pour les bases de données distribuées et les bases de données centralisées.

Le coût de transfert est pris en considération. En effet, le coût de transfert d'un bloc de données dans une base de données distribuées est plus important que celui d'une base de données centralisée.

Cependant, le coût de transfert entre deux nœuds est fixé pour toutes les transactions entre chaque paire de nœuds du réseau.

Navathe et Ra [33] étendent les travaux de Navathe [26] sur la fragmentation verticale en proposant une méthode de fragmentation verticale utilisant des techniques graphiques.

La majeure contribution de cette méthode est que tout les fragment sont générés en une seule itération avec une complexité de $O(n^2)$, ce qui est meilleur que $O(n^2 \log(n))$ qui représente la complexité de l'algorithme [26].

Autre avantage de cette méthode, on n'a pas besoin d'une fonction objective empirique pour générer les partitions.

Il est à noter que le résultat des fragments générés peut être plus important que le nombre de nœuds dans un système distribué, si le cas se présente une opération de fusionnement des fragments sera nécessaire.

Lin et Zhang [30] a souligné que la restriction d'un cycle d'affinité résultant dans une formulation est un problème NP-difficile, et donc les propriétés revendiquées dans [33] ne peuvent être garantis. De ce fait, un nouvel graphe est proposé en utilisant 2-connectivité au lieu du cycle d'affinité pour construire des fragments disjoint, ce qui est plus tard attribués à des nœuds du réseau distribué.

Lin et Zhang. [30] ont poursuivi le travail [31] en présentant un algorithme pour la construction des fragments non disjoint.

Cheng et al. [16] ont formulé la problématique de fragmentation de base de données dans le contexte distribué comme un problème du voyageur de commerce (TSP), pour lequel une approche basée sur la recherche génétique pour la fragmentation est proposé de réaliser des performances système élevées.

Encore une fois, cette approche est basée sur matrice d'affinité, qui calcule l'affinité entre les attributs d'une partition en fonction de la distance entre les attributs.

L'objectif de cette approche est de regrouper les attributs tels que la différence entre la distance moyenne au sein des groupes et la distance moyenne entre les groupes soit réduite au minimum.

Cependant, il n'existe aucune preuve que cette approche peut en effet minimiser le coût total des requêtes.

Après avoir survolé quelques travaux sur la fragmentation verticale par la matrice d'affinité, nous étudierons dans la section suivante les travaux et approches basés sur le modèle de coût.

II.5.2. APPROCHE ORIENTÉE MODÈLE DE COÛT

Le principe de cette approche se base sur l'estimation du coût de chaque transaction. Le calcul du coût est calculé par l'estimation du nombre d'accès physique aux disques et l'estimation du temps de calcul du CPU, il existe une multitude de modèle de coût.

Ces algorithmes emploient des facteurs physiques spécifiques tels que le nombre d'attributs, leur longueur et sélectivité, cardinalité de la relation etc.…

L'un des travaux important dans le domaine est celui de Yao [40] là ou il présente une estimation rapprochée basée sur des fondements mathématique.

Cardenas10] a donné l'expression estimant le nombre de bloques accédés pour une requête donnée.

Le principe de la fragmentation verticale basé sur le modèle de coût et une analyse de l'approche sont détaillés dans la section suivante.

II.5.2.1. PRINCIPE DES APPROCHES BASÉES SUR LE MODÈLE DE COÛT

Lorsqu'une transaction ou une requête s'exécute sur un SGBD, plusieurs tâches seront déclenchées en arrière plan par le serveur, en l'occurrence analyse syntaxique et lexicale de la requête, optimisation de la requête, accès physiques aux données :

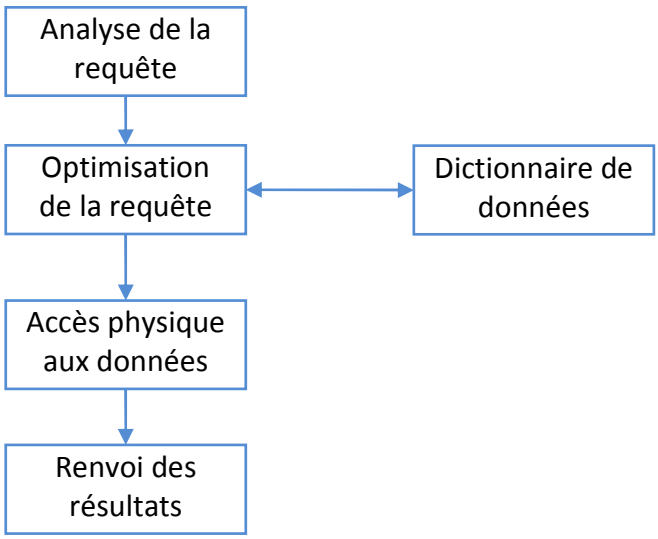


Figure 6 : exécution d'une requête sur un SGBD.

Toutes ces tâches consomment des ressources et du temps, et engendrent des calculs qui s'effectuent au niveau du processeur en arrière plan, mais l'accès physique aux fichiers de données sur disques déclenche le processus du pivotement des têtes pour effectuer la lecture sur disque afin d'atteindre l'adresse des blocs de données.

Le modèle de coût est une formule qui permet pour une requête donnée de rapprocher une estimation sur le nombre d'accès aux fichiers de données et le temps de calcul du CPU.

Comparé aux processus du pivotement des têtes de lecture, le temps que consomme le processeur (CPU) pour le calcul est négligeable. Dans la majorité des études on néglige le temps de calcul du CPU.

Une fonction de coût fondée sur cette hypothèse est déduite, cette fonction reflète le temps que consomment les ressources de la machine pour exécuter la requête.

Les facteurs sur lesquels se basent le modèle de coût sont : la quantité prévue de données qui doivent être transmises afin de répondre à la requête, et le nombre d'entrée/sortie (pivotement de la tête de lecture) pour lire les données sur disque.

Il est à noter que le modèle de coût ne représente qu'une estimation du coût d'exécution des requêtes, et ce modèle varie selon le contexte de la base (base de données distribuée, base de données centralisée).

Dans notre étude, l'objectif est de choisir un schéma de fragmentation verticale qui minimise cette fonction de coût.

La compréhension de l'accès physique fichiers de la base sur disque nous permettra de mieux comprendre le fondement du modèle de coût, nous vous proposons de vous reporté à la partie Annexe A consacrée à l'organisation physique des fichiers.

Nous présentons le modèle de coût de Yao 40], le modèle sur lequel se base la majorité des études orientées modèle de coût.

II.5.2.2. MODÈLE DE COÛT DE YAO [40]:

Dans la littérature sur la fragmentation des bases de données en général, on trouve plusieurs travaux sur l'estimation du coût des requêtes, cette estimation se base sur le calcul la quantité de données transférées et le nombre d'accès disque, d'ailleurs elles ont été les méthodes les plus communément utilisées pour évaluer les schémas de fragmentation.

Le modèle le plus utilisé et le plus stable est celui de YAO [40]. Dans ses travaux, YAO [40] à ignorés le coût d'accès par index et le coût de stockage.

Le contexte dans lequel est formulé le modèle de YAO est celui d'une base de données centralisée, avec deux niveaux de mémoire, la formule décrite par YAO permet de calculer ou d'estimer le nombre de bloques lus durant la sélection aléatoire d'enregistrements afin d'exécuter une requête.

Théorème de YAO [40] : soit **N** enregistrements groupé dans **M** bloques avec : **(1 < M ≤ N)**, chaque bloque contenant **N / M** enregistrements.

Si **K** enregistrements **(K ≤ N – (N/M))** sont sélectionnés aléatoirement parmi les N enregistrements, le nombre prévu de bloque atteint (bloques contenant au moins un enregistrement sélectionné) est donné par :

$$\boldsymbol{M\left[1-\prod_{i=1}^{k}\frac{(N*D)-i+1}{N-i+1}\right]} \quad \textbf{Équation 12}$$

où: $\boldsymbol{D=1-(1/M)}$ **Équation 13**

En se basant sur le modèle de Yao [40], Gorla [24] a donné l'expression de l'exécution d'une requête *q* sur une table partitionnée sur *j* segments.

- ***freq q*** = fréquence de la requête q
- ***Segment qj*** = le nombre de partitions j requis par q.
- ***Mi*** = *[T * Li / BS].*
- ***Mi*** = le nombre de bloques à être accédé dans la partition i.
- ***Kq*** = le nombre de tuples satisfaisant la q.
- ***T*** = nombre totale de tuples
- ***Li*** = taille de la partition en octet
- ***BS*** = taille d'un bloque

$$Cost = freq_q * \left(\sum_{i=1}^{Segment_{qi}} \left[M_i \left(1 - \left(1 - \frac{1}{M_i} \right) \right)^{Kq} \right] \right) * \left(1 + \left(0.1 * \left(Segment_{qi} - 1 \right) \right) \right)$$

Équation 14

La première partie de la formule calcule le nombre total de bloques à accéder pour chaque segment *qi*, la seconde partie calcule le coût de concaténation des segments *qi* dans la mémoire.

Dans ce qui suit nous présentons les différents travaux se basant sur le modèle de coût.

II.5.2.3. LITTÉRATURE SUR LE MODÈLE DE COÛT

Hammer et Niamir, [25] L'objectif de ce travail est de concevoir des mécanismes qui permettent de sélectionner automatiquement une partition quasi-optimale des attributs, basé sur le modèle d'utilisation du fichier et sur les caractéristiques des données dans le fichier.

L'approche adoptée pour ce problème est basée sur l'utilisation d'un évaluateur de partition et d'une heuristique qui guide une recherche à travers le vaste espace des partitions possibles.

L'heuristique est de proposer un ensemble restreint de partitions prometteuses de soumettre à une analyse détaillée.

Pour faire les auteurs [25] ont développé deux heuristiques, de groupement et de regroupement, et les ont utilisées pour effectuer la fragmentation. Le groupement heuristique commence en assignant chaque attribut à un fragment différent.

Dans chaque itération, tous les groupements possibles de ces partitions sont considérés et celle avec l'amélioration maximum est choisie en tant que candidate pour le groupement de la prochaine itération.

Pendant le regroupement, des attributs sont déplacés entre les fragments pour réaliser toutes les améliorations additionnelles possibles.

Cornell et Yu [18] ont discutés la fragmentation verticale pour les bases de données relationnelles et ont estimés que le temps de réponse des transactions est influencé par le nombre d'accès disque par transaction.

Considérant que l'utilité de la fragmentation verticale est de réduire au minimum le nombre d'accès disque, Cornell et Yu [18], ont proposés une méthode de deux étapes qui consiste en une étape d'analyse de requête pour estimer les paramètres, et une étape de fragmentation binaire qui peut être appliqué de manière récursive.

Avec le modèle de coût proposé, le résultat de [32] à été testé pour montrer que la solution à base d'affinité ne conduit pas toujours à une réduction du nombre d'accès disque.

Chu [17] Cette étude présente les concepts de moyens et l'appui à la conception de deux méthodes pour résoudre le problème de fragmentation verticale: MAX et FORWARD SELECTION. Ces concepts expriment l'idée simple que, pour minimiser les coûts, un segment doit contenir tous les attributs requis par une transaction donnée et excluent les autres attributs.

Contrairement à toutes les études antérieures qui, ces deux méthodes traitent une transaction comme une variable de décision et leur exécution n'est pas affectée par le nombre d'attributs. Les deux méthodes produisent des résultats excellents, sans l'aide de calculs mathématiques compliqués.

Ils prennent également en compte la complexité résultant de l'interaction entre l'attribut de partitionnement et de sélection du chemin d'accès.

Par ailleurs, les deux méthodes donnent des informations détaillées normalement non disponible à partir d'autres méthodes. Ces informations donnent un nouvel aperçu du processus de partitionnement attribut.

MAX est adapté pour les cas impliquant un petit nombre de transactions, FORWARD SELECTION pour les cas impliquant un grand nombre de transactions.

La caractéristique importante de ces deux procédures est qu'elles traitent les transactions (au lieu des attributs) en tant que variables de décision. Ainsi, le temps d'exécution de ces procédures ne dépend pas du nombre d'attributs et peut être efficacement exécuté lorsque le nombre d'attributs est très important.

La fonction objective ne compte que le nombre d'accès disque. Les deux approches [17] et [18] ne sont pas adaptées à des bases de données distribuées.

Golfarelli et al. [22] les auteurs utilisent la fragmentation verticale pour partitionner des vues matérialisées définies sur un entrepôt [22]. Cette fragmentation est basée sur une charge de requêtes et un modèle de coût.

Selon les auteurs, la fragmentation verticale désigne deux opérations : d'une part la fragmentation d'une vue en plusieurs fragments et, d'autre part, l'unification en une seule vue de deux ou plusieurs vues ayant une clé commune.

L'unification respecte la règle de reconstruction d'une table fragmentée à partir de ses fragments verticaux [34] et vise à réduire la redondance des vues.

Les auteurs supposent que leur approche peut être bénéfique pour la distribution de l'entrepôt sur une architecture parallèle et proposent de combiner leur algorithme de fragmentation avec un algorithme d'allocation des fragments sur les nœuds distants.

Chakravarthy et al. [12] ont posé qu'il devrait y avoir un moyen de mesurer la qualité d'un schéma de fragmentation verticale.

À cet effet, ils ont créé une fonction objective, Partition Evaluator (PE), pour évaluer les différents algorithmes de fragmentation verticale en utilisant les mêmes critères.

Le PE est composé de deux éléments, coût d'accès aux attributs locaux et le coût d'accès aux attributs distant. Dans ce cas, le but est de minimiser les accès à distance et de maximiser les accès locaux sans tenir compte de la réplication.

En d'autres termes, l'objectif est que chaque site soit capable de traiter les transactions au niveau local avec un nombre minimum de d'accès à des sites distants. L'idée est de parvenir à un coût minimum de traitement pour toute transaction, peu importe où il est situé.

Il est clair que le PE ne peut être utilisé dans les bases de données distribuées, car ni la taille ni les facteurs de coûts de transaction du réseau n'ont été pris en compte.

Son et Kim [38] on supposés que le problème de fragmentation verticale doit prendre en compte le nombre de fragments finalement généré.

Ils ont discuté du problème de génération de n_fragments verticaux qui est plus souple que le problème de génération d'un partitionnement optimal.

Leur nouvelle contribution est une fonction objective qui vise à minimiser non seulement la fréquence des accès de requête pour différents fragments, mais aussi la fréquence des accès interféré entre les requêtes.

Dans la fonction objective, la localisation des données n'est pas considérée car les requêtes ne se distinguent pas entre les sites.

Gorla [23] L'auteur étend les recherches précédentes qui sont basées la fragmentation d'une seule relation, et effectue la fragmentation de plusieurs relations, en prenant en compte les contraintes d'intégrité référentielles, clefs étrangère et primaires.

Le temps d'accès est basé sur l'estimation du temps d'exécution, le calcul correspond à un modèle de coût complet qui prend en charge la plupart des types de transaction comprenant des mises à jour et les jointures.

L'auteur a employé un procédé heuristique pour résoudre le problème utilisant un index d'affinité à 2 attributs et un algorithme de groupement (clustering) à 2 étapes. L'application de cette méthodologie sur de petits problèmes a rapporté les solutions optimales obtenues par énumération approfondie.

Nous présentons dans la section suivante les approches basées sur les algorithmes génétiques pour la sélection d'un schéma de FV.

II.5.3. APPROCHE BASÉE SUR LES ALGORITHMES GÉNÉTIQUES

Les algorithmes génétiques sont des algorithmes métaheuristiques, qui permettent de résoudre les problèmes combinatoires. De nombreux travaux se sont basés sur les AG pour optimiser la conception physique par la technique de FV.

Puisque dans notre étude nous avons implémenté une approche basée sur les AG pour la fragmentation verticale, nous avons consacré toute une section pour détailler les études se référant de cette dernière.

Il est à considérer que la fonction objective ou fitness décrite dans toutes ces études, représente l'estimation du coût de la charge de travail appliquer la base de données.

Cheng et al. [16] ont proposé une recherche basée sur l'algorithme génétique pour le partitionnement de données pour atteindre de hautes performances de récupération dans les bases de données réparties, en considérant que le problème de la fragmentation verticale est formulé comme le problème du voyageur de commerce.

Song et Gorla [24] ont employé des algorithmes génétiques pour obtenir une solution simultanée pour générer les fragments verticaux et repérer le chemin d'accès aux fragments. Ils ont également employé le nombre d'accès de disque comme critère d'évaluation de la fragmentation.

Concernant la conception de base de données orientée objet, Gorla a employé l'algorithme génétique pour déterminer les variables d'instance qui devraient être stockées dans chaque sous-classe/classe dans une hiérarchie de sous-classe, de sorte que tous les coûts des opérations de base de données soient réduits au minimum.

Angel et Zevala [7] ; Dans le contexte des bases de données distribuées un nouveau problème est souvent traité en complément du problème de fragmentation. Il s'agit du problème d'allocation des fragments sur les nœuds du réseau, ça revient à allouer les fragments dans les différents sites du réseau de telle sorte à ce que l'accès à l'information soit le plus rapide que possible.

Dans ce contexte, les auteurs [7] proposent l'adaptation de l'algorithme génétique pour résoudre à la fois le problème de la fragmentation et de l'allocation des fragments sur les nœuds du réseau.

Un vecteur de nombre entier de langueur égale aux nombre d'attribut de la relation sera employé pour représenter les solutions possibles au problème.

Ce vecteur prendra les nombres entiers entre 1 et N, où N dénote le nombre d'emplacement dans le réseau.

Exemple :

Soit une base de données distribuée sur un site composé de 5 sites, et soit une relation R appartenant à la base composée de 10 attributs :

R (A, B, C, D, E, F, G, H I, J)

Selon l'approche d'Angel [7], le vecteur suivant est un codage d'une solution potentielle pour la fragmentation et l'allocation de la relation R:

(5, 2, 4, 3, 1, 3, 3, 5, 2, 4)

Ce vecteur sera interprété comme suit :

- La position dans le vecteur indique le numéro d'attribut,
- Les composants du vecteur indiquent les emplacements sur site.

Par conséquent, le vecteur donné en solution sera interprété comme suit :

Sites	Allocation de l'attribut		
1	E		
2	B	I	
3	D	F	G
4	C	J	
5	A	H	

Cette approche permet de donner une solution qui résout à la fois la fragmentation de la relation et l'allocation des fragments sur les sites du réseau.

Nous allons aborder dans la section suivante le sujet de la fragmentation verticale basée sur la fouille de données.

II.5.4. APPROCHE BASÉE SUR LA FOUILLE DE DONNÉES (DATAMINING)

La fouille de données est technique sui permet d'extraire des connaissances et des règles à partir d'un important volume de données et d'informations. La fouille de données est techniques qui est utilisées dans plusieurs domaines dont nous citons les statistiques, le commerce et marketing, système experts, industriels, profilage des clients et consommateurs.

Le travail de Gorla et Pang [24] s'est distingué par son originalité, les détailles fournis, et les bons résultats obtenus.

Ce travail fera l'objet d'une analyse et étude détaillée dans le prochain chapitre, et une implémentation logicielle sur l'outil FragMan.

Gorla & Pang Wing [24] proposent une nouvelle approche pour fragmenter verticalement les relations d'une base de données en utilisant des règles d'association, une technique de datamining qui consiste à adapter l'algorithme Apriori sur le problème de la fragmentation.

La méthode se résume en trois étapes : estimation de la charge de travail, calcul des formules de coût pour l'exécution de transaction dans un environnement fragmenté, et un algorithme pour générer les fragments.

L'approche [24] a été expérimentée sur deux bases de données réelles, et les résultats obtenus sont assez concluants.

II.6. BILAN ET DISCUSSION :

Aboutir à un schéma de fragmentation verticale optimale sur une base de données est une opération complexe qui nécessite beaucoup de calculs, ceci se fait ressentir plus sur les relations composées d'un nombre important d'attributs.

Les travaux que nous avons cités proposent des méthodes différentes dans des contextes différents pour aboutir à un schéma acceptable ou optimal de fragmentation verticale, nous avons synthétisé les différentes approches et études dans le Tableau 7.

Travaux	Matrice d'affinité	Modèle de coût	Algorithme Génétique	Algorithmes et contribution	Contexte	
					BD centralisée	BD distribuée
Hoffer & al [26]	X			Bond énergie	X	
Navathe et al [32]	X			Bond énergie, allocation, réplication	X	X
Navathe et Ra [33]	X			Graphe d'affinité	X	X
Lin et Zhang [30]	X			Graphe d'affinité amélioré	X	X
Cheng et al. [16]	X		X	Voyageur de commerce	X	X
Hammer et Niamir [25]		X		Groupement regroupement récursive	X	
Cornell et Yu [18]		X		Fragmentation binaire récursive	X	
Chu [17]		X		MAX et FORWARD SELECTION	X	
Golfarelli et al. [22]		X		Fragmentation, unification	X	X
Chakravarthy et al[12]		X		Partition Evaluator, maximiser les accès locaux	X	X
Son et Kim [38]		X		Nombre de fragments peut être pré-renseigné.	X	
Gorla [23]	X	X		Prise en compte intégrité référentielle, plusieurs relations.	X	
Song et Gorla [24]		X	X	Fragmentation et chemin d'accès aux fragments	X	
Angel et Zevala [7]		X	X	Fragmentation et allocation	X	X
Gorla & Pang Wing [24]		X		Datamining : Apriori	X	

Tableau 6 : sommaire des travaux sur la fragmentation verticale

L'application des approches et algorithmes d'optimisation afin de déduire le meilleur schéma de fragmentation verticale devient indispensable. Ceci a donné naissance aux outils d'assistance des gestionnaires et administrateurs de bases de données pour la conception physique.

Ces outils offrent la possibilité d'optimiser la conception physique d'une base de données sur la base de plusieurs techniques, avec la possibilité de simuler l'impact de l'application d'une conception sur les performances de la base de données.

Nous présentons dans la section suivante, les principaux outils d'assistance proposés par les éditeurs de SGBD commerciaux les plus connus.

II.7. OUTILS D'AIDE POUR LA CONCEPTION PHYSIQUE :

Avec le développement des SGBD, et leurs expansions dans tous les domaines confondus, et le volume de données en perpétuelle croissance, la conception physique des bases de données devient de plus en plus difficile.

La conception physique implique une dizaine et souvent une centaine de contraintes et de variables et paramètres, qui sont difficiles à régler et à stabiliser, surtout lorsque que leurs impacts se répercute souvent sur la conception de la base, et qui pourrait jouer un rôle néfaste sur les performances de la base.

Donner des solutions pour la conception physique intuitivement n'est pas une chose évidente, car proposer des index ou des partitions sur une base de données peut avoir des répercutions néfastes sur les performances du serveur, et ne présente aucune garantie de réussite pour atteindre ces objectifs.

Cela a donné lieu à des outils automatisés tels qu'IBM DB2 Design advisor, Oracle SQL Access Advisor, Oracle SQL Tuning Advisor et Microsoft Database Tuning Advisor (DTA), et bien d'autres.

Ces outils de réglage de base de données et d'analyse des performances, permettent à l'analyste de se concentrer sur les solutions et leurs impacts sur le système, lui procurant des conseils et recommandations pour l'assister à bien choisir les bons ajustements dans l'optimisation des ressources de la base de données.

Une conception physique et un tuning assisté qui représente une aubaine pour les administrateurs et usagers des bases de données, car elle représente un gain de temps énorme pour l'administrateur pour augmenter efficacement les performances de la base de données.

Dans la section suivante, nous présentons une étude et analyse sur les outils d'aide à la conception physique existant sur le marché des SGBD commerciaux les plus répandue, en l'occurrence Oracle, DB2 d'IBM, SQL Server de Microsoft.

Cette étude nous fournira une synthèse sur les techniques adoptées par les éditeurs et développeurs de bases de données commerciales disponible sur le marché.

II.7.1. L'OUTIL INDEX TUNING WIZARD

En 1998, Microsoft SQL Server 7.0 a développé un outil pour l'optimisation et la recommandation de la conception physique des bases de données, appelé l'index Tuning

Wizard (ITW) sur la base des techniques présentées dans [13], [14]. Dans la version de Microsoft SQL Server 2000, ITW a été améliorée pour fournir des recommandations sur les index, vues matérialisées (connu sous le nom indexed views in Microsoft SQL Server) et les index sur les vues matérialisées.

Dans la version la plus récente de Microsoft SQL Server 2005, la fonctionnalité de ITW a été remplacé par une véritable application appelée Database Engine Tuning Advisor (DTA) [5].

DTA peut fournir des recommandations sur les index, les vues matérialisées, les index sur les vues matérialisées et le partitionnement horizontal. DTA permet aux administrateurs de bases de données d'exprimer les contraintes sur le partitionnement, de stockage, les structures physiques qui doivent être conservées, etc. En outre, le DTA expose un riche ensemble d'options, par exemple, quelle table optimiser, etc. Un usage important du DTA est l'option de réglage d'un problème lié à une requête unique.

D'autre part, le DTA peut également être invoqué directement à partir de SQL Server Management Studio, l'outil intégrer à l'environnement administrateur.

Les recommandations DTA sont accompagnées par un ensemble de rapports d'analyse détaillés qui calculent l'impact de la validation des recommandations du DTA sur la base de données.

L'outil expose également la fonctionnalité «What if» afin de faciliter le réglage manuel par le DBA. Pour plus de détails sur le DTA, son usage et les meilleures pratiques veuillez vous référer [6].

II.7.2. L'OUTIL DB2 DESIGN ADVISOR

IBM DB2 Universal Database (UDB) version 6 ont développé en 1999 DB2 advisor [39] qui peut recommander des index pour une charge de travail. Ensuite, le DB2 Design Advisor outil de DB2 version 8.2 [39] fournit des recommandations sur les index, vues matérialisées, partitionnement et clustering multidimensionnelle.

Contrairement au DTA où la recherche sur toutes les structures se fait en même temps, DB2 Design Advisor est structuré de façon à proposer des recommandations indépendamment pour chaque technique de conception physique. L'étape de recherche qui génère les recommandations finales invoque itérativement chacun des assistants à part.

II.7.1. L'OUTIL SQL ACCESS ADVISOR

Oracle 10g a propose SQL Access Advisor [19], qui a comme entrée la charge de travail (généré par l'Oracle Automatic Tuning Optimizer sur la base de l'analyse des requêtes), et fournit en sortie des recommandations sur la base de la charge de travail globale. L'outil recommande des index et des vues matérialisées.

Le projet Microsoft Research AutoAdmin [1,15] a mis au point des assistants pour sélectionner automatiquement des index et des vues matérialisées. Leurs outils exploitent le modèle de coûts utilisé par l'optimiseur pour estimer l'intérêt des index et propose des vues matérialisées.

Plus récemment, leurs outils ont été étendus à la fois à pour la fragmentation verticale et la fragmentation horizontale sur un prototype de recherche [4].

Enfin, nous notons que les récents manuels sur la conception des bases de données, par exemple [29], consacrent d'importantes publications pour avancer et progresser dans ce domaine et suggèrent le recours aux outils automatisés pour la conception physique sur les systèmes de bases de données commerciaux.

Le tableau suivant récapitule et résume les différente fonctionnalités et avancés des outils d'aide à la conception physique des principaux éditeurs SGBD (Oracle, IBM, SQL Server) sur le marché.

	MS SQL Database Tuning Advisor
Procédure d'appel	Database Engine Tuning Advisor (outil GUI) + utilitaire DTA (commande).
Paramètres d'entrée	- Les états individuels à partir de ***SQL Server Management Studio*** - Fichier texte avec l'état de la charge du travail, - Table ou fichier de trace à partir du SQL Affiner (dérivation automatique des priorités), - Fichiers XML de la charge de travail (allocation directe des priorités possibles), Entre autre d'autres paramètres peuvent être stipulés en option : - Nombre possible des conceptions permis pour la structure, - une partie de l'objectif de la configuration, - la quantité de tables sur la base l'espace maximum disponible, - les délais pour l'optimisation.
Approche d'optimisation utilisée	approche d'optimisation simultanée de plusieurs BD par : - Utiliser de la méthode « What If » index, - Utiliser du modèle de coût pour l'estimation de la valeur d'une configuration.
Recommandation :	Propositions de créer ou de supprimer - Index, - vues matérialisées, - le partitionnement horizontal,

	DB2 Design Advisor
Procédure d'appel	- manuellement db2advisor (commande) ou Control Center.
Paramètres d'entrée	- Paramètres de ligne de commande (simple déclaration), - Fichiers (estimations), - Tableau (ADVISE_WORKLOAD), - Dynamic SQL cache, - Affiner (Patroller),
Approche d'optimisation utilisée	Etude de la moyenne du temps de réponse pour une charge de travail sur différentes configurations.
Recommadation :	Propositions de créer ou de supprimer : - Index, - vues matérialisées (MQT), - tables cluster multidimensionnelles (MDC tables), - partitions horizontales.
Observations	- la fonction Multidimensionnel Clustering gère des cubes de type OLAP et offre la fonction d'agréger les données par une interface Wysiwyg. - Intégré, l'outil DataJoiner procure en une vue unique sur les données issues de bases comme celles d'Oracle, de Microsoft, d'Informix, de Sybase, etc.
	Oracle SQL Access Advisor
Procédure d'appel	- manuel (Enterprise Manager ou par commande) ou à partir des recommandations de l'ADDM, - complément de SQL Tuning Advisor
Paramètres d'entrée	- Le contenu actuel du cache SQL, - Charge de travail de l'AWR (Automatic Workload Repository), - Table SQL remplie par le DBA, - Génération d'une charge de travail virtuelle pour les systèmes (phase de conception).
Approche d'optimisation utilisée	- Considération du compromis entre les performances d'une requête et l'espace de stockage pour les objets proposés, en estimant le coût de la configuration.
Recommandation	Propositions de créer ou de supprimer - Index, - vues matérialisées, - suivi des vues matérialisées (Materialized View Logs), - recommandations de différents modes de fragmentation horizontale (en option),
Observation	Les recommandations doivent être confirmées par le DBA

Tableau 7 : caractéristiques des outils d'aide à la conception physique.

II.8. CONCLUSION :

Les techniques de conception physique nous permettent d'optimiser les performances des bases de données, et de remédier au problème de la latence du système.

Aujourd'hui, Tous les éditeurs et concepteurs de SGBD ont pris conscience du fait que la gestion de la conception physique et l'autorégulation des ressources de la base doivent être incluse et prise en charge par le système.

Nous pouvons bien comprendre l'ampleur de l'assistance des administrateurs de base de données dans la conception physique. Ceci n'est plus devenu qu'une simple option pour les administrateurs et gestionnaires de bases de données, mais une exigence, surtout pour les SGBD gérant de grands volumes de données, pour le compte d'utilisateurs et de clients de plus en plus impatient et exigeant.

L'Etude de l'impact de la fragmentation verticale sur les performances des bases de données, et l'analyse de la littérature sur les travaux et recherches réalisés, nous permettra d'implémenter et de réaliser des algorithmes permettant de sélectionner un schéma de fragmentation verticale optimisé.

Sur le marché des SGBD, la technique de la fragmentation verticale ne connait pas une grande popularité, surtout pour les SGBD commerciaux, d'ailleurs, nous pouvons remarquer que rare sont les SGBD qui ont mis au point un dispositif pour intégrer et supporter la fragmentation verticale, et aucun des outils d'assistance n'intègre des recommandations sur la fragmentation verticale.

Dans le prochain chapitre nous vous proposons l'outil FragMan, un outil qui offre des recommandations sur la fragmentation verticale, se basant sur les techniques des algorithmes génétiques, le datamining, et l'approche Affiner que fait l'objet de notre contribution.

III. Algorithmes de fragmentation verticale.

III.1. INTRODUCTION :

Après avoir fait un survol sur le contexte de notre travail, et eu un aperçu général sur les travaux phares réalisés sur la fragmentation verticale, nous aborderons dans cette partie les approches et contributions de notre travail.

Ce chapitre est divisé en trois parties.

La première partie concernera l'approche fouille de données (datamining), une explication et compréhension de l'approche est présentée. Nous enchainerons avec l'algorithme Apriori, une variante optimisée basée sur les algorithmes de fouille de données, suivi d'une illustration sur le fonctionnement d'Apriori. Enfin, nous clôturons cette partie par la manière avec laquelle on va utiliser Apriori pour procéder à la fragmentation verticale.

La deuxième partie traitera le sujet des algorithmes génétiques, nous présenterons des définitions et analyses sur les notions théoriques de l'algorithme, le mode de fonctionnement des algorithmes génétiques, suivi par le procédé d'utilisation des algorithmes génétiques pour la fragmentation verticale.

La troisième partie traite l'approche "Affiner" que nous avons proposée, avec une explication détaillée de cette approche, suivie d'une illustration de l'approche pour la fragmentation verticale. Enfin, nous clôturons ce chapitre par une conclusion.

III.2. FOUILLE DE DONNÉES (DATAMINING)

Considérée comme une discipline associée au domaine de l'aide à la décision, la fouille de donnée ou « datamining » est un procédé qui nous permet d'extraire les connaissances et les informations valides et exploitables à partir de sources de données.

Le terme Datamining à été lancé par le MIT « Massachusetts Institute of Technology » vers les années 1980.

L'idée principale est de *valoriser la mémoire et les données d'une entreprise ou d'une entité en les explorant au moyen de techniques de mathématiques appliquées*. Le MIT annonçait que cette technologie de l'information préparait la prochaine révolution du siècle.

A la différence des méthodes d'analyse de données ou de statistiques, la fouille de données permet l'extraction des connaissances sans établir une hypothèse au préalable ; en fait les connaissances extraites émergent.

La fouille de données se propose donc de transformer de grands volumes données en informations ou connaissances, les données analysées proviennent souvent de sources diverses comme les entrepôts de données, ou autres comme Internet, ou source de temps réel (GAB, analyse des achats par téléphone ….).

Il est à noter qu'une réécriture des données s'impose dans le cas où la source de données n'est pas forcement une base de données, cela suppose s'être doté des outils nécessaires.

Souvent utilisé dans l'analyse du comportement des personnes : consommateur, clients et usager, la fouille de données peut aussi s'avérer utile dans la prédiction des comportements des gens ainsi que la détection des fraudes et la prédiction des zones et heures de pointes dans le transport.

III.2.1. PRINCIPE DE LA FOUILLE DE DONNÉES

Plus qu'une théorie normalisée, la fouille de données est un processus d'extraction de connaissances métiers comportant les étapes principales suivantes :

- *Formaliser* un problème que l'organisation cherche à résoudre en termes de données
- *Accéder* aux données appropriées quelles qu'elles soient
- *Préparer* ces données en vue des traitements et utilisations futurs
- *Modéliser* les données en leur appliquant des algorithmes d'analyse
- *Évaluer* et valider les connaissances ainsi extraites des analyses
- *Déployer* les analyses dans l'entreprise pour une utilisation effective

Ce processus est cyclique et permanent; la fouille de données rend dès lors plus compréhensible, "visible", l'activité de l'organisation, et permet de rationaliser le stockage de l'information et des données.

La fouille de données ne consiste pas en une succession d'études ad hoc mais a pour objectif de capitaliser des connaissances acquises sous forme de connaissances explicites.

Elle conduit donc à mieux structurer les contenus nécessaires à l'ingénierie des connaissances.

C'est sa principale raison d'être ; on peut comparer de façon lointaine cette activité à celle de conceptualisation au cours de l'apprentissage humain : « Une bonne *compréhension* est intimement liée à une bonne *compression* », l'une comme l'autre utilisant une connaissance de corrélation pour représenter - et donc manier - l'information sous forme plus concise.

Nous pouvons distinguer trois grandes familles dans les algorithmes de fouille de données:

III.2.1.1. MÉTHODES NON SUPERVISÉS :

Cette catégorie ne considère en aucun cas qu'un ensemble de données ou de variables soit plus important que les autres.

Parmi les techniques les plus connues nous citons : réseau de neurones, k-means, règles d'association, etc.…

III.2.1.2. MÉTHODES SUPERVISÉES :

Contrairement aux méthodes non supervisées, cette catégorie est adaptée plus particulièrement à un groupe de variables définis comme étant l'objectif de l'analyse.

Les méthodes supervisées ont donc pour objectif d'expliquer un comportement précis comme l'augmentation des ventes d'un produit par rapport a un autre, ou les raisons de la réussite d'une technique de marketing a une autre.

Les techniques suivantes appartiennent à cette catégorie : techniques statistiques de régressions linéaires et non linéaires, techniques à base d'arbres de décision, raisonnement par cas ….

III.2.1.3. LES MÉTHODES DE RÉDUCTION DES DONNÉES :

Permettent de compresser et de réduire et de filtrer un ensemble de données volumineux, et d'en extraire les informations les plus pertinentes et significatives.

Souvent utilisées comme approches complémentaires des techniques supervisées ou non supervisées et dans le domaine de la statistique. Parmi ces techniques nous citons : techniques d'analyse factorielle et technique de positionnement.

Dans notre étude nous nous intéressons plus aux techniques de génération des règles d'association, les règles d'association sont traditionnellement liées au secteur de la distribution car leur principale application est << l'analyse du panier de la ménagère >> qui consiste en la recherche d'associations entre produits sur les tickets de caisse.

Le but de la méthode est l'étude de ce que les clients achètent pour obtenir des informations sur qui sont les clients et pourquoi ils font certains achats. La méthode recherche **quels produits tendent à être achetés ensembles.**

La méthode peut être appliquée à tout secteur d'activité pour lequel il est intéressant de rechercher des groupements potentiels de produits ou de services : services bancaires, services de télécommunications, par exemple.

Elle peut être également utilisée dans le secteur médical pour la recherche de complications dues à des associations de médicaments ou à la recherche de fraudes en recherchant des associations inhabituelles.

Un attrait principal de la méthode est la clarté des résultats générés. En effet, le résultat de la méthode est un ensemble **de règles d'association.**

Des exemples de règles d'association sont :

- si un client achète des plantes alors il achète du terreau,

- si un client achète du poisson et du citron alors il achète un jus.
- si un client achète une télévision, il achètera un magnétoscope dans un an.

Ces règles sont intuitivement faciles à interpréter car elles montrent comment des produits ou des services se situent les uns par rapport aux autres. Ces règles sont particulièrement utiles en marketing.

Les règles d'association produites par la méthode de réduction des données peuvent être facilement utilisées dans le système d'information de l'entreprise.

Cependant, il faut noter que la méthode, si elle peut produire des règles intéressantes, peut aussi produire des règles triviales (déjà bien connues des intervenants du domaine) ou inutiles (provenant de particularités de l'ensemble d'apprentissage).

La recherche de règles d'association est une méthode non supervisée car on ne dispose en entrée que de la description des achats.

Dans le paragraphe suivant nous allons définir la terminologie exploitée dans le domaine.

III.2.2. DÉFINITIONS

Les items décrivent l'ensemble des articles ou d'éléments sujet de l'étude, par exemple dans l'application du panier ménagère la liste décrivant la liste des items corresponds à l'ensemble des articles disponibles dans le magasin (pain, lait, café, friandises). Il est à noter que les items ne sont pas dupliqués dans le même ensemble.

Les règles d'association, est une association entre plusieurs articles qui permet de découvrir à partir d'un ensemble de transactions, un ensemble de règles qui exprime une possibilité d'association entre différents items.

Une transaction est une succession d'items exprimée selon un ordre donné ; de même, l'ensemble de transactions contient des transactions de longueurs différentes.

Un exemple classique sur l'application des règles d'association est le panier de ménagère qui décrit un ensemble d'achats effectués au dans un supermarché ; les règles d'association permettent de découvrir de régularités dans l'ensemble de transactions comme par exemple : Si champoing alors gèle douche, etc.

Ces règles permettent par exemple au gérant de proposer des bons de réductions significatifs sur les achats futurs des clients.

Une ***règle d'association*** est une règle de la forme : Si **condition** alors **résultat**. Dans la pratique, on se limite, en général, à des règles où la **condition est une conjonction d'apparition d'articles et le résultat est constitué d'un seul article.**

Par exemple, une règle à trois articles sera de la forme :

Si X et Y alors Z : (X et Y) → Z ; règle dont la sémantique peu être énoncée : Si les articles X et Y apparaissent simultanément dans un achat alors l'article Z apparaît.

Pour choisir une règle d'association, il nous faut définir les quantités numériques qui vont servir à valider l'intérêt d'une telle règle.

Pour faire, les règles d'associations sont vérifiées sur la base de la mesure des deux valeurs du Support et de la Confiance définies dans la section suivante.

(3) SUPPORT ET CONFIDENCE D'UNE RÈGLE :

Le **support d'une règle** indique le pourcentage d'enregistrements qui vérifient la règle, il indique le nombre de fois, en termes de pourcentage, où la loi est applicable.

Considérons l'ensemble des paniers d'achats T ; soit R un groupe de produits et U l'ensemble des paniers d'achat contenant le sous-groupe R, on a :

$$Support \quad (R) = \left(\frac{|U|}{|T|} \right) * 100 \ \%$$

Avec $|U|$ et $|T|$ le nombre d'éléments de U et de T.

Un ensemble d'items est dit fréquent s'il correspond à un motif fréquent dans la base de transactions. Un seuil minimal de support min_Supp est fixé à partir duquel un ensemble d'items est dit fréquent.

Exemple

Si S = {café, lait}, si un magasin reçoit 210 consommateurs dans la journée et que 45 d'entre eux achètent du café et du lait, alors :

Support ({café, lait}) = (45 / 210) * 100% = 21,4 %.

Support (S1 -> S2) = support (S1 & S2).

Reprenons notre exemple : la loi « café -> lait » n'est applicable que si le consommateur qui a acheté du lait, a acheté du café, c'est-à-dire dans 24,8% des cas.

Plus le support est important, et plus la fréquence avec laquelle on a pu vérifier que la loi était vraie ou fausse l'est aussi.

La confiance est le rapport entre le nombre de transactions où tous les items figurant dans la règle apparaissent, et le nombre de transactions où les items de la partie condition apparaissent.

Soit la règle d'association R = S1 -> S2, avec S1 et S2 des groupes de produits.

$$confiance = \frac{Support(S1, S2)}{Support(S1)} * 100\%$$

Exemple

Si S1 = {café} et S2 = {lait}, et que sur les 210 consommateurs 52 achètent du café, 108 achètent du lait, parmi lesquels 45 achètent ces deux produits simultanément:

*Confiance(R) = (support ({café, lait}) / support ({café}))*100%*

Confiance(R) = 21,4% / 24,8% = 86,3%.

Cela signifie que si le consommateur achète du café, il achète également du lait dans 86,3% des cas.

A noter que la loi S1 -> S2 est différente de la loi S2 -> S1. Ainsi, dans notre exemple, si le consommateur achète du lait, il n'achète du café que dans 41,6% des cas.

Une règle est dite solide si son support est supérieur ou égal à un support minimal fixé et si sa confiance est supérieure à une confiance minimale donnée.

Une variante des plus intéressante des algorithmes d'extractions des règles d'associations et qui à fait l'objet de notre étude, c'est l'algorithme dit Apriori, développé par Agrawal et al [1], ce qui nous a intéressé dans cette approche c'est la complexité réduite des opérations pour aboutir aux même résultats que les autres algorithmes.

III.2.3. ALGORITHME APRIORI

Apriori est le fruit des travaux réalisés par Agrawal & al [1], c'est un algorithme permettant de trouver des lois d'association. L'induction des lois d'association, également appelée analyse des paniers d'achats, est une méthode qui a pour objectif de trouver des régularités dans le comportement des consommateurs.

Ces régularités se caractérisent par des groupes de produits qui sont régulièrement achetés. Ainsi, la présence d'un ou plusieurs produits dans un panier d'achat peut entraîner la présence d'un ou plusieurs autres produits.

L'Algorithme Apriori est basé sur le principe que tout ensemble d'un sous-ensemble non fréquent est non fréquent ce qui implique une limitation de l'espace de recherche.

Déroulement de l'algorithme A Priori:

- Génération des ensembles d'items
- Calcul des fréquences des ensembles d'items

- On garde les ensembles d'items fréquents : avec un support minimum: les ensembles d'items fréquents
- Générer à partir de l'ensemble des items fréquents des règles d'associations solides : ayant une confiance suffisante.

Code de l'algorithme : Nous adoptons dans la suite la terminologie suivante :

- Ck est l'ensemble des items de taille k,
- Lk est l'ensemble d'items fréquents de longueur k.

L'algorithme A priori est énoncé dans l'algorithme 1 :

Calculer L1 (l'ensemble des items fréquents)

K ← 2

Tan que $L_{k-1} \neq \emptyset$ faire

C_k ←apriori-gen (L_{k-1})

Pour chaque transaction t

Faire

C_t ←sous-ensemble (C_k, t)

Pour tout les candidats c ∈ C_t :

Faire

Count[c] ++;

Fin faire

Fin faire

L_k ← {c∈ C_k | count[c]≥ minSup};

K++;

Return $\bigcup_k L_k$;

Algorithme 1 : Algorithme Apriori.

Où count [c] représente la fréquence de l'élément « c » dans les transactions ou le panier.

L'algorithme Apriori permet de découvrir les sous-ensembles d'items fréquents en partant de ceux dont la longueur est 1 et en augmentant la longueur au fur et à mesure.

Cet algorithme est fondé sur la propriété des sous-ensembles d'items fréquents. Il fait appel à deux algorithmes :

Apriori-gen : L'algorithme Apriori-gen est constitué de deux phases la première phase nommé Joindre trouve tous les candidats possibles de longueur k à partir de l'ensemble Lk−1,

La deuxième phase efface Ck les éléments qui ne vérifient pas la propriété des sous-ensembles fréquents.

Sous-ensemble : L'algorithme sous-ensemble calcule le sous ensemble Ct ⊆ Ck qui correspond aux sous-ensembles présents dans les transactions contenues dans D.

Par exemple si L3 = {{123}, {124}, {134}, {135}, {234}},

La phase joindre donne comme résultat C4 = {{1234}, {1345}}, ensuite la phase effacer donne le résultat :

C4 = {1234} car l'élément {145} n'est pas dans L3 et donc {1345} est effacé.

Exemple d'application : avec minSup =2

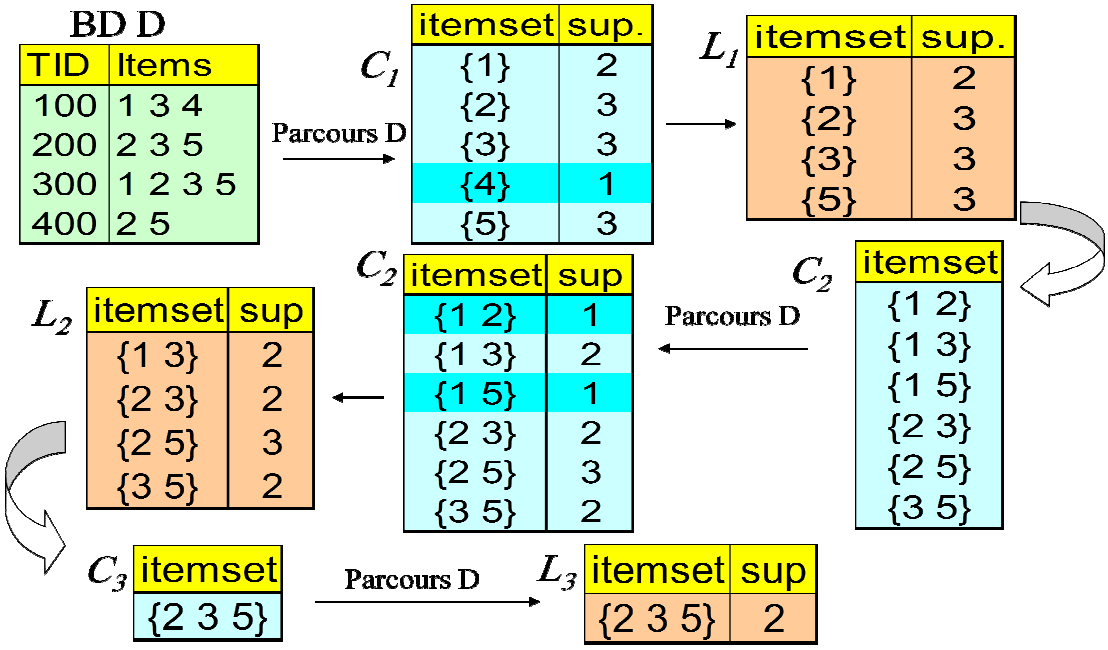


Figure 7 : Illustration de l'exécution d'Apriori

L'algorithme Apriori nous permet d'extraire les règles d'associations entre les différents items, l'intérêt majeur que nous apporte l'algorithme c'est la complexité réduite de ce dernier, apporter des solutions en un temps efficace.

Nous avons vu dans le chapitre I que la fragmentation verticale sur les entrepôts de données est un problème difficile, et les nombres de propositions possibles est en nombre de Bell, la complexité réduite de l'algorithme nous à motivé à appliquer Apriori pour apporter une réponse au problème de la fragmentation verticale des bases de données.

III.2.4. UTILISATION D'APRIORI POUR LA FRAGMENTATION VERTICALE

L'utilisation d'Apriori pour générer des fragments verticaux suit le raisonnement suivant.

Soit A et B deux attributs dans une relation. Si la valeur de la confiance de la règle A→B est plus grande que le minimum prédéfini (min_conf), alors l'association entre les attributs A et B est assez forte pour qu'ils puissent être groupés dans la même partition.

Puisque l'ordre des attributs dans la même partition n'est pas significatif, les valeurs de confiances pour A→B et pour B→A sont calculées et la plus faible valeur sera choisie pour représenter l'association entre les attributs A et B.

Ainsi, la confiance de (A, B) = freq (A, B)/max (freq(A), freq(B)).

La valeur de confiance est plus haute dans deux situations :

- Quand il existe peu de transactions requérant l'attribut A seulement ou B seulement,
- Quand les transactions requérant à la fois A et B. Ainsi, la confiance fournit davantage de justification, en ajout pour le support minimum (minSup) du stockage des attributs A et B dans la même partition.

L'algorithme proposé regroupe les étapes suivantes :

- découvrir les grands ensembles d'item,
- filtrer les ensembles d'item,
- produire les partitions de données, et choisir le meilleur schéma de fragmentation.

III.2.4.1. DÉCOUVERTE DES GRANDS ENSEMBLES D'ITEM (ITEMSET)

Les grands ensembles d'item sont les combinaisons d'attributs qui ont un support au-dessus du support minimum prédéfini minSup.

Pour une transaction de récupération, l'ensemble des attributs dans les clauses select est considéré ; pour une transaction d'INSERT/DELETE, tous les attributs dans la relation sont employés.

De grands ensembles d'items (itemsets) peuvent être découverts en adaptant l'algorithme « Apriori » (Agarwal et Srikant, 1994 [1]) et en ajoutant des fréquences de transaction plutôt que de compter les transactions.

Le support des candidats en C_k est calculé par la somme de la fréquence des requêtes qui contiennent l'ensemble des candidats. On dérive l'ensemble des grands items L_K de telle façon qu'il atteint le niveau minimum prédéfini de support (minSup).

Ainsi, les entrées pour cette étape sont un ensemble de transactions de base de données (des récupérations et des mises à jour) $T_1 \ldots T_m$, les fréquences de transaction $Freq_1 \ldots Freq_m$, et le niveau de support prédéterminé minSup.

Les sorties de ce module représentent les grands ensembles d'items $L_1 \ldots L_K$.

III.2.4.2. FILTRAGE DES GRANDS ENSEMBLES D'ITEMS

L'ensemble d'Items dont la confiance est plus petite que la confiance minimum prédéterminée (min_conf) est rejeté.

Pour chaque ensemble d'Items des grands itemsets (ensembles d'items) L_K, toutes les règles d'association possibles sont produites et les niveaux de confiance correspondants sont calculés.

Soit les règles d'association LHS→RHS, où LHS, RHS représentent des ensembles d'items, le niveau de confiance est calculé comme la fréquence totale des transactions qui requièrent à la fois LHS et RHS, divisés par la fréquence totale des transactions requérant seulement LHS.

Cette étape prend en entrées les grands Itemsets ($L_1 \ldots L_K$) et le niveau de confiance minimum prédéterminé (min_conf) et en sortie génère de grands Itemsets filtrés ($L'_1 \ldots L'_k$).

III.2.4.3. DÉRIVATION DES PARTITIONS VERTICALES

Après avoir déduit les grands itemsets filtrés $L'_1 \ldots L'_k$, nous commençons à partir des k-itemsets pour déterminer les partitions.

Créer un schéma de fragmentation en sélectionnant un itemset à partir de L'_k, et puis en sélectionnant d'autres itemset de $L'_{k-1} \ldots L'_1$ dans cet ordre de telle sorte qu'elles soient disjointes.

Réitérer la même opération de sélection des itemset, jusqu'à ce que les derniers items soient sélectionnés à partir L'_1, ceci produit un schéma de fragmentation.

Répéter le processus ci-dessus jusqu'à ce que tous les schémas de fragmentation possibles soient dérivés. Ainsi, ce module à pour entrée les grands Itemsets filtré $L'_1 \ldots L'_k$ et retourne en sortie les schémas de partitions P1…Ps.

Il convient noter que chaque schéma de fragmentation peut être constitué de plusieurs partitions.

III.2.4.4. DÉDUCTION DU SCHÉMA FINAL

Les entrées pour ce module est l'ensemble des schémas de fragmentations P1…Ps et l'ensemble des transactions $T_1 \ldots T_M$ et en sortie le schéma résultant P_{res}.

Une fois la dernière étape de l'algorithme est exécutée, nous obtenons en résultat au moins un schéma de fragmentation proposée par la méthode.

En calculant les coûts d'exécutions pour les schémas de fragmentations P (par le calcule de la sommes des coûts pour chaque transaction en utilisant le modèle de coût de YAO [33] – voir chapitre I), on peut déduire le schéma de fragmentation P_{res} avec le meilleur coût d'exécutions.

III.2.5. ILLUSTRATION :

Dans ce qui suit nous allons illustrer le fonctionnement de l'algorithme Apriori pour la fragmentation verticale.

Considérons l'ensemble de transactions T1… T5 et leurs fréquences opérant sur une relation avec six attributs R (A, B, C, D, E, F).

	Transactions	Attributs	Fréquences
T1	SELECT A, B, E FROM R ;	A, B, E	1
T2	SELECT B, E FROM R ;	B, E	3
T3	SELECT A, D, F FROM R ;	A, D, F	3
T4	INSERT FROM R ;	A, B, C, D, E, F	2
T5	DELETE FROM R;	A, B, C, D, E, F	1

Tableau 8 : ensembles de transactions

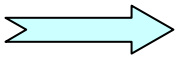
Soit le support minimum de 40% (c.-à-d., le min_supp est 4), les grands itemsets LK sont produit en adaptant l'algorithme Apriori.

D'abord, le candidat 1-itemsets C1 est dérivé en ajoutant les fréquences des transactions dans lesquelles un attribut est employé.

Pour les transactions d'insertions et de suppression, on suppose que tous les attributs sont employés puisque la rangée entière doit être accédée.

Par exemple, l'attribut A est employé dans les transactions T1, T3, T4, et T5 avec les fréquences 1, 3, 2, et 1 respectivement. Ainsi le support pour l'attribut {A} = 7. L'opération est reproduite pour chaque attribut.

Ensemble d'item	Support
A	7
B	7
C	3
D	6
E	7
F	6



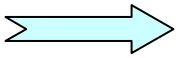
Ensemble d'item	Support
A	7
B	7
D	6
E	7
F	6

Tableau 9 : dérivation des 1- fréquent itemsets

Le candidat 1-itemset L1 est dérivé du candidat 1-itemset C1 en éliminant des les itemsets dont le support est inférieur à 4.

Le candidat 2-itemset C2 est dérivé en concaténant L1 avec L1 et en cherchant les niveaux de support en balayant l'ensemble de transaction, il faut noter que les doublons sont éliminés.

Ensemble d'item	Support
A, B	4
A, D	6
A, E	4
A, F	6
B, D	3
B, E	7
B, F	3
D, E	3
D, F	6
E, F	3



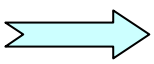
Ensemble d'item	Support
A, B	4
A, D	6
A, E	4
A, F	6
B, E	7
D, F	6

Tableau 10 : dérivation des 2- fréquent itemsets

Par exemple, à partir de L1, en concaténant {A} et {B}, nous trouvons {A B} en tant qu'un des 2-itemsets C2. Le niveau de support pour {A B} est obtenu en ajoutant les fréquences des transactions dans lesquelles A et B sont employés.

Ces transactions sont T1, T4, et T5 avec des fréquences de 1, 2, et 1, respectivement, ayant pour résultat le niveau de support 4 pour {AB}.

Ensemble d'item	Support
A, B, D	3
A, B, E	4
A, B, F	3
A, D, E	3
A, D, F	6
A, E, F	3



Ensemble d'item	Support
A, B, E	4
A, D, F	6

Tableau 11 : dérivation des 2- fréquent itemsets

Le processus ci-dessus est répété jusqu'à ce que tous les grands itemsets soient dérivés.

III.2.5.1. FILTRAGE AVEC DES NIVEAUX DE CONFIANCE PRÉDÉTERMINÉS :

Chaque ensemble d'items dans les grands ensembles d'items est maintenu seulement si l'ensemble d'items à un niveau de confiance supérieur ou égale à min_conf.

Par exemple, considérer l'association {A, D, F}. La valeur de confiance pour A→DF est calculée par le nombre de transactions dans lesquelles tous les attributs A, D, et F sont sollicités (c.-à-d., 6) divisé par le nombre de transactions dans lesquelles A est employé (c.-à-d., 7).

Ainsi, les valeurs de confiance pour A→DF, D →AF et DF→A sont 6/7, 6/6, et 6/6, respectivement. De même la confiance pour DF→A, AF→D, AD→A est 6/6, 6/6, et 6/6, respectivement. Puisque la valeur la plus basse (6/7 ou 86%) est plus haute que le niveau de confiance minimum prédéfini (disons que min_conf = 30%), l'association {A, D, F} est maintenue dans l'ensemble des grands items filtrés L3.

III.2.5.2. DÉDUCTION DES FRAGMENTS VERTICAUX :

L'ensemble d'items {A, D, F} est pris de L3 en tant qu'une partition ; alors nous allons sur L2 balayer l'ensemble 2-items et constater qu'il y a seulement {B E} qui ne recouvrent pas avec la partition existante. L'attribut restant {C} est pris de L1, ayant pour résultat le schéma de fragmentation, (C, B E, A D F).

De la même façon, en considérant l'autre ensemble 3-items {A, B, E} de L3, nous obtenons (C, D F, A B E) comme autre solution.

Hormis ces deux schémas de fragmentation, celui qui a le moindre coût d'exécution selon le modèle de coût est choisi comme le schéma de fragmentation.

III.3. ALGORITHME GÉNÉTIQUE:

C'est à la parution du livre intitulé « L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature » que Charles Darwin en 1860 à bouleverser toute les théories sur la création et l'existence des espèces.

Bien qu'elle ne soit pas réellement basée sur des fondements scientifiques, et quelle soit très controversée par la communauté scientifique, le fondement de ses révélations, a pour hypothèse que les entités appartenant à un environnement subissant l'influence de facteurs externes peuvent changer de composition et de structure et s'adapte au fur et à mesure leurs environnement.

Les êtres d'un espace donné, évoluent en perpétuelle continuité au fil du temps ; grâce à la mutation ; et transmettent leurs acquis génétiquement par le procédé de la

reproduction, créant ainsi de nouveaux individus plus fort ; le plus adapté et le plus fort des individus à plus de chance de survivre et de se reproduire, et les moins forts disparaissent ce procédé est la sélection naturelle.

Ces révélations ont inspirés plusieurs recherches, les algorithmes génétiques (AG) en sont un fruit.

Les algorithmes génétiques sont basés sur des concepts basiques de la génétique et des lois de la survie dans la nature énoncées par Darwin : croisements, mutations, sélection.

Ils représentent une variante des algorithmes évolutionniste, une variante des algorithmes métaheuristiques et des algorithmes d'optimisation stochastiques.

Historiquement les premiers algorithmes évolutionniste et méta heuristique ont vues le jour en 1965 à l'université technique de Berlin (Allemagne).

L'idée à été proposée par Ingo Rechenberg, quand aux algorithmes génétiques ils furent révélés grâce aux travaux de John Holland [28] sur les systèmes adaptatifs. John Holand a proposé le concept par lequel les programmes informatiques pouvaient évolués et s'adapter en conséquence de l'environnement dont elles appartiennent.

L'objectif des algorithmes génétiques (AG) est de résoudre les problèmes combinatoires complexes PCD (ou d'en obtenir une solution rapprochée) dans un temps acceptable.

Les problèmes combinatoires difficiles (PCD) sont des problèmes n'ayant pas de solutions calculables en temps raisonnable, pour lesquels on ne connaît pas de solution exacte ou meilleur, là où souvent échouent d'autres méthodes classiques.

Les PCD sont caractérisées par ensemble de solutions possibles très vastes désigné par le terme espace de recherche, et l'exploration de tout cet espace peut se révéler très coûteuse, ce qui peut consommer beaucoup de temps et de ressources.

Il est clair que les méthodes de calcul classiques ne sont pas particulièrement bien adaptées aux espaces de recherche multimodale, un espace multimodale étant défini par une espace qui se compose de plusieurs extrema locaux.

Les extrema locaux multiples présentent des difficultés pour les méthodes classiques pour atteindre l'extrême global, les méthodes classiques peuvent se stabiliser sur l'un des extrema locaux, en ignorant totalement l'extrema global qui existe ailleurs dans l'espace.

La Figure 8 illustre un espace de recherche très multimodal, cette figure reflète la forme d'un espace de recherche difficile contenant plusieurs extrema locaux. L'optimum sur cet espace est difficile à atteindre.

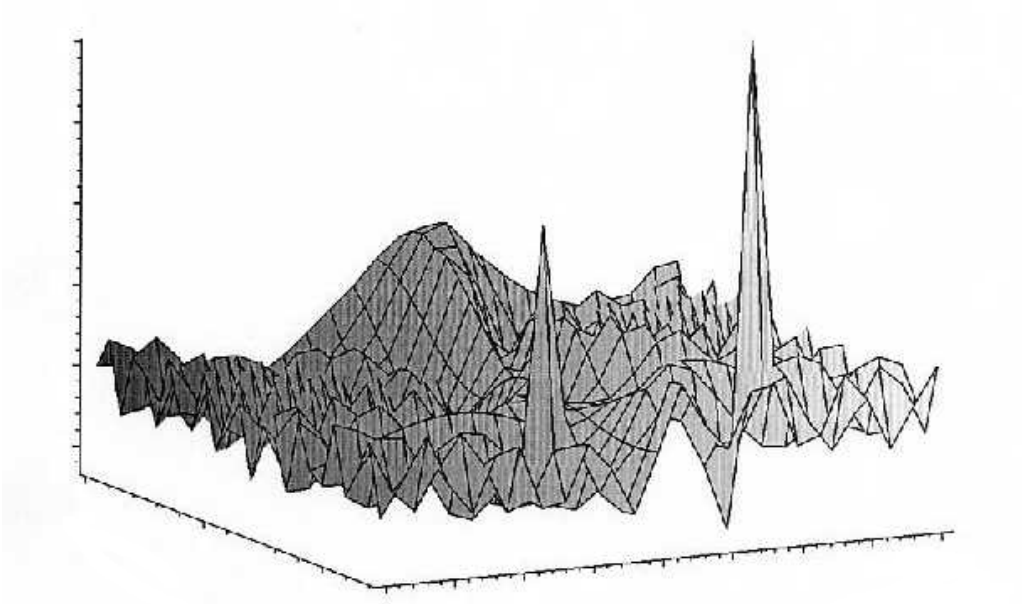


Figure 8 : Espace de recherche multimodal

Dans la partie qui suit nous présenterons le mode de fonctionnement des algorithmes génétiques.

III.3.1. FONCTIONNEMENT DES AG

Le principe fondamental de la méthode est le suivant :

1- au début l'algorithme commence par générer aléatoirement une solution initiale (population);

2- chaque individu de la population est soumit à une évaluation par la fonction fitness afin d'en mesurer la qualité de la solution fournie par ce dernier (la force) ;

3- après l'évaluation intervient l'opération de sélection des individus pour en reproduire les nouveaux éléments de la population, selon Mendel la survie du plus adapté (la loi du plus fort) permet d'obtenir une progéniture forte.

4- ensuite l'opération de reproduction à base des individus sélectionnés au préalable en utilisant les opérations de mutation et de croisement qui nous permettent d'obtenir une nouvelle génération qui serra en principe mieux

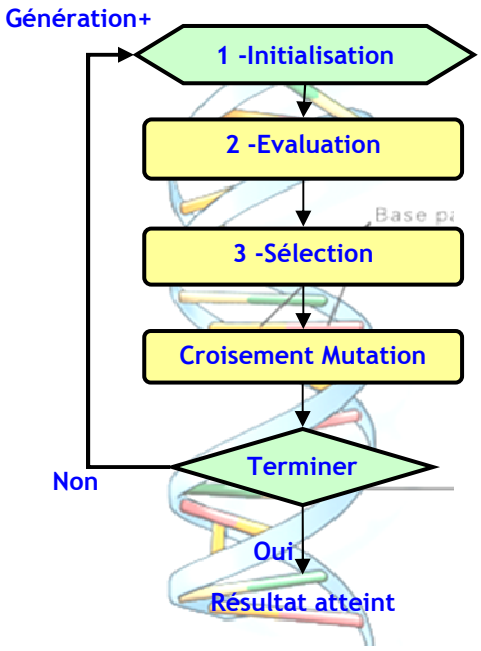


Figure 9 : Fonctionnement des AG

adaptés ou équivalente au fitness de la génération précédente.

En générale, à la fin de l'exécution de l'algorithme, nous obtenons une solution optimale ou proche de l'optimum pour le problème posé initialement.

III.3.2. OPÉRATEURS DES ALGORITHMES GÉNÉTIQUES

Dans la section suivante, une description détaillée des différents opérateurs et étapes de l'algorithme vous serra présenter :

III.3.2.1. LE CODAGE

La mise en œuvre des AG débute par le codage ; en fait c'est le langage que manipulent les AG ; et c'est l'un des facteurs les plus importants pour le bon déroulement de l'algorithme.

Il est à rappeler que dans la biologie naturelle les scientifiques procèdent à un codage des gènes et chromosomes de l'ADN afin de les identifiés ; en suivant le même raisonnement, dans les AG.

Nous commençons par adopter un codage approprié à notre situation, opération dont l'objectif est de construire un modèle numérique et permet d'obtenir une projection de notre problème sur un modèle manipulable par les algorithmes génétiques.

Pour bien expliquer le codage, nous allons essayer d'avoir une idée générale sur les entités que manipule un AG. Les AG manipulent une population, cette population sera générée à chaque itération grâce aux procédés de la mutation et croisement et sélection.

La structure de la population est la suivante : une population est un ensemble d'individus, chaque individu est un code qui représente une solution potentielle pour notre problème, un individu est formé d'un ou plusieurs chromosomes.

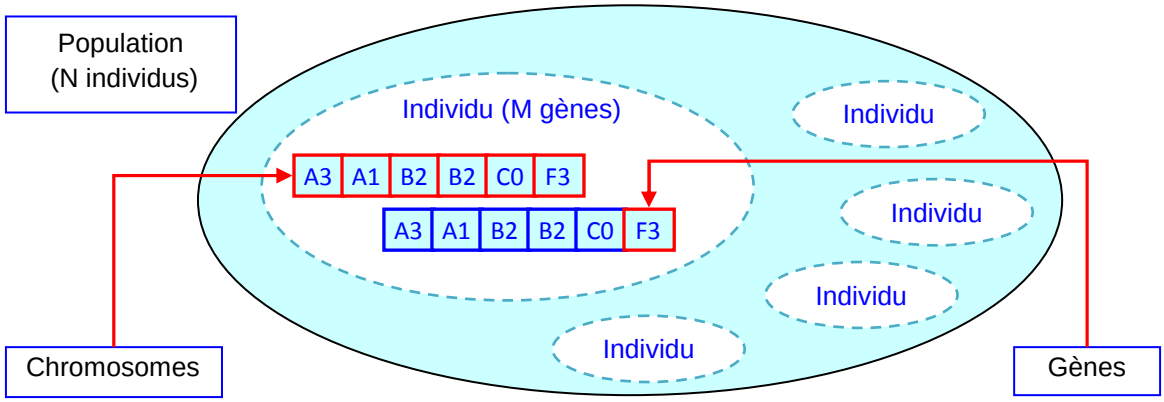


Figure 10 : le codage

Un chromosome est une suite de gènes ; les gènes sont une suite de code et peuvent prendre une des différentes valeurs des allèles ; en quelque sorte un gène est une variable qui puise ses valeurs dans les allèles.

La récapitule la notion du codage sur les algorithmes génétiques.

Par exemple le problème du déplacement d'un robot dans un labyrinthe, un individu peut être codé comme suivant, {droite, gauche, haut, gauche, droite, bas}.

Il faut prendre en considération qu'un bon codage peut aider énormément dans la réussite de l'exploration efficace de l'espace de recherche, et en atteindre les optimums plus rapidement.

Il existe plusieurs variantes de codage nous en citons :

Le codage binaire : C'est le type de codage le plus utilisé, chaque individu de la population est représenté par une suite de 1 et 0, {1,0}, exemple : [0 0 1 0 1 0 1 0].

Le codage de Gray : il ajoute le concept de distance entre deux individus, deux éléments voisins en termes de distance de Hamming peuvent ne pas coder nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient dans les autres types codage peut être évité en utilisant un codage de Gray.

La "distance de Hamming" est considérée comme mesure de la différence entre deux éléments de population, cette mesure compte les différences de bits du même rang de ces deux séquences.

III.3.2.2. LA SÉLECTION

L'étape de sélection est l'étape qui permet de choisir pour chaque génération les individus destinés à transmettre leurs gènes et à se reproduire, les individus qui seront soumis aux deux opérations de croisement et de mutation.

Il existe plusieurs algorithmes de sélection nous en citons les suivant :

III.3.2.2.1. ROUE DE LA FORTUNE (ROULETTE WHEEL).

Cet algorithme est l'une des variantes les plus intéressantes et les plus utilisées dans les applications recourant aux algorithmes génétiques.

Son principe est le suivant, un individu aura les chances de se reproduire proportionnellement par rapport à la valeur de son fitness. L'algorithme partage un camembert sur les individus, chaque individu lui sera attribué une partie de la roue, une fois la roue est tournée, l'individu sur lequel s'arrêtera la roue sera élu.

Exemple : Soit la population composée de quatre individus avec les valeurs de leurs fitness représenté comme suit: P = {(Individu A, f=0.1), (Individu B, f=2.2), (Individu C, f=7), (Individu D, f=0.7)}.

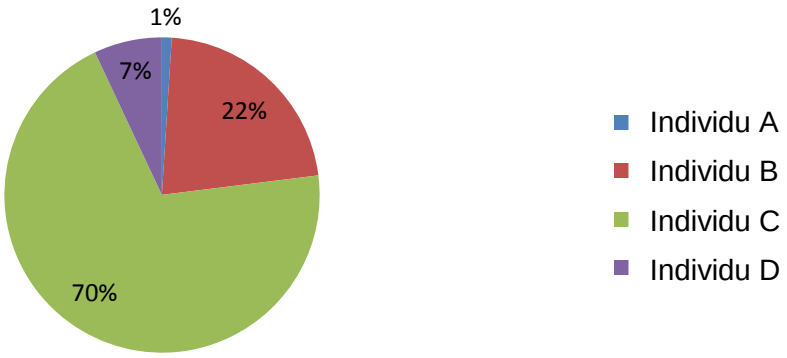


Figure 11 : Sélection par roulette de la fortune

Dans la Figure 11 on remarque que la roue est partagée entre les quatre individus qui composent la population en proportion de leur fitness, comme pour le cas de l'individu C, qui vraisemblablement occupe la plus grande partie de la roue, par ce fait, cet individu est le mieux positionné pour être sélectionner, néanmoins il faut prendre en considération que l'individu « A » a quand même une chance d'être sélectionner.

III.3.2.2.2. LA MÉTHODE ÉLITISTE.

Le principe est le plus simple qu'il soit, choisir les meilleurs individus de la population vis à vis de leurs fitness.

Il est à noter que cette méthode n'est pas efficace, car elle peut converger vers un minimum local, et aboutir sur des solutions prématurées, et par conséquent, omettre une grande partie de l'espace de recherche.

III.3.2.2.3. LA SÉLECTION PAR TOURNOIS.

Le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de la population, on les compare, celui qui a le fitness le plus élevé l'emporte avec une probabilité p comprise entre 0.5 et 1.

III.3.2.2.4. LA SÉLECTION UNIVERSELLE STOCHASTIQUE.

Dans cette méthode la population est considérée comme l'image d'un segment, l'image est découpée en nombre de sous-segments autant de fois que le nombre d'individus. Un ensemble de points équidistants sont choisis, ces points représentent les individus sélectionnés par cet algorithme.

III.3.2.2.5. SÉLECTION UNIFORME.

La sélection des individus se fait aléatoirement, uniformément et sans prise en compte du fitness. Chaque individu a donc une probabilité 1/P d'être sélectionné, où P représente le nombre total d'individus dans la population.

III.3.2.3. LE CROISEMENT

L'opération de croisement est le procédé par lequel s'effectue la production de nouveaux individus sur la base de deux autres individus, le croisement dans les algorithmes génétiques est inspiré de la reproduction dans la nature à partir d'un couple d'individu, le croisement des deux chromosomes des individus, nous permet d'en extraire deux nouvelles structures sur la base de la recombinaison des chromosomes parents, selon un schéma donné, avec une probabilité P renseignée au préalable dans l'algorithme.

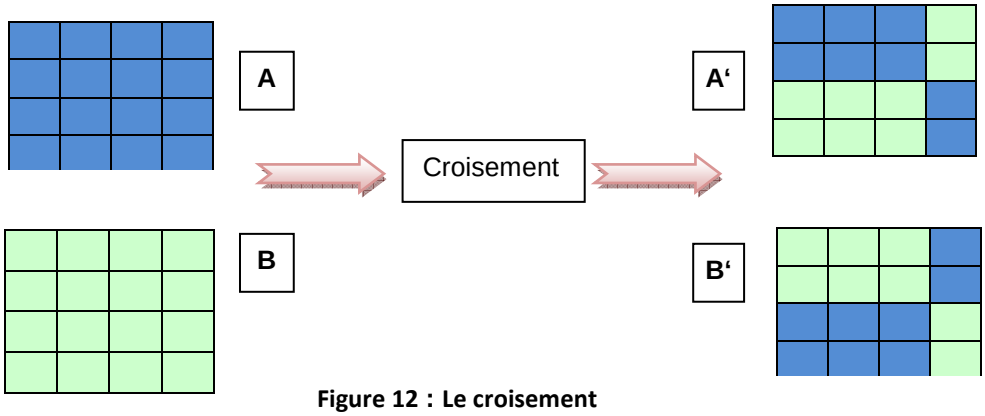


Figure 12 : Le croisement

Dans l'exemple de la Figure 12, le croisement entre deux individus permet de développer deux nouvelles combinaisons de chromosome A' et B' à partir de A et B, ces deux nouvelles combinaison sont en fait le croisement et la recombinaison des chromosomes de A et B.

III.3.2.4. LA MUTATION

Comme dans la nature, la mutation d'un chromosome est l'altération ou la modification d'un gène de la combinaison selon une probabilité donnée, (bien qu'en réalité cette altération ne peut être que néfaste dans la nature comme le cancer, effets des radiations).

Cette altération est modélisée par la modification de la valeur du gène choisi par une autre valeur prise aléatoirement parmi les valeurs possibles que peut prendre le gène.

La Figure 13 illustre brièvement l'opération de mutation, les cases colorées sont les cases mutées.

Figure 13 : La mutation

III.3.3. UTILITE DES OPERATEURS DE CROISEMENT ET DE MUTATION

Ces deux opérateurs permettent aux algorithmes d'explorer et d'atteindre les optimums locaux et généraux dans l'espace de recherche.

Le croisement permet la transmission des caractères acquis d'un individu vers les descendants, ainsi les individus développés seront soit équivalents ou soit meilleurs que leurs ascendants, concrètement atteindre le minimum local.

La mutation nous offre la possibilité de trouver des individus qui appartiennent à une autre partie de l'espace de recherche et ainsi de ne pas tomber dans le piège de se contenté du minimum local, alors qu'il pourrait exister une solution meilleur que celle du minimum local.

Grâce à l'opérateur du croisement nous pouvons atteindre l'optimum pour un extrema local, mais cette solution ne représente guère la meilleure solution. Quand à l'application de l'opérateur de mutation, il nous permet de découvrir un nouvel espace de solution qui peut être meilleur et qui n'es pas évident à atteindre.

Les algorithmes génétiques sont assez souples et flexibles pour s'adapter à n'importe quel problème, et présentent de très bonnes aptitudes à résoudre les problèmes combinatoires difficiles.

Dans notre travail nous allons expérimenter les algorithmes génétiques pour la résolution du problème de la fragmentation verticale.

III.3.4. LES ALGORITHMES GÉNÉTIQUES POUR LA FV:

Nous présentons dans cette section comment nous avons utilisé les algorithmes génétiques pour sélectionner un schéma de fragmentation verticale.

III.3.4.1. LE CODAGE

Nous avons adopté un codage assez simple, les individus représentent la manière dont laquelle la table sera fragmentée verticalement, en d'autre terme les groupes d'attributs que nous allons formés. Le chromosome est une collection d'entiers, la longueur du chromosome est égale aux nombres de d'attributs dans la table, chaque entier exprime la partition à laquelle l'attribut sera affecté.

Exemple : Table avec 10 attributs, le codage suivant : chromosome {5, 0, 5, 5, 0, 1, 1, 2, 3, 4} sera interprété comme suit :

- *Partition 0* : {1, 4} ; *Partition 1* : {5, 6} ; *Partition 2* : {7} ;
- *Partition 3* : {8} ; *Partition 4* : {9} ; *Partition 5* : {0, 2, 3}.

III.3.4.2. LA POPULATION

L'initialisation de la population s'effectue d'une manière totalement aléatoire. La taille de la population peut être ajustée par l'utilisateur, et le nombre de génération peut être aussi renseigné par l'utilisateur.

III.3.4.3. LE CROISEMENT

Un point est choisi au hasard dans le premier chromosome, avec une certaine probabilité, il sera actif pour le croisement selon une probabilité Pc de croisement (renseigné au préalable par l'utilisateur).

Un autre point est choisi au hasard sur le second chromosome en utilisant le même procédé, mais le point choisi doit être dans le même ordre que le point choisi dans le premier chromosome.

Si le point correspondant dans le second chromosome n'a pas pu être trouvé, alors les chromosomes ne seront pas croisés, voir l'exemple dans la Figure 14.

Exemple :

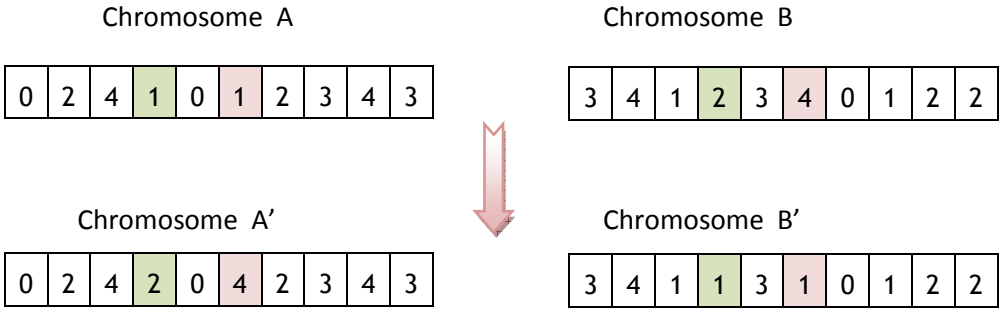


Figure 14 : Exemple de croisement

III.3.4.4. LA MUTATION

L'opérateur de mutation parcourt chaque gène des chromosomes candidat retenu par le procédé de la sélection dans la population, et les mute selon un taux de mutation donnée (renseigné au préalable par l'utilisateur).

III.3.4.5. LA SÉLECTION

Dans notre approche nous offrons la possibilité de procéder par les différents modes de sélections présentées dans les paragraphes précédents.

Le mode de sélection standard permet de sélectionner un nombre donné de chromosomes de la population pour leur permettre de continuer à survivre.

Cette sélection devrait être guidée par les valeurs de fitness, mais la fonction fitness doit être considérer comme une probabilité statistique de survie, non pas comme le seul facteur déterminant pour la survie. En d'autres termes, les chromosomes avec des valeurs de fitness plus élevées sont plus susceptibles d'être sélectionnés que ceux avec des valeurs fitness inférieure, sauf que ce n'est pas une chose toujours garantie.

III.3.4.6. LA FONCTION FITNESS

Concernant la fonction fitness, l'évaluation est basée sur le modèle de coût de Yao [32]. L'objectif de l'algorithme sera de chercher les solutions avec le moindre coût, les solutions dont le temps d'exécution sera le plus réduit.

III.3.5. ILLUSTRATION DE L'ALGORITHME GÉNÉTIQUE :

Dans notre exemple nous allons illustrer la fragmentation verticale par les algorithmes génétiques de la table personnes composé de 11 attributs dont l'attribut code est la clé primaire de la table :

Personnes : {***Code*** ; Nom ; Prénom ; Date-N ; Lieu-N ; Taille ; Adresse ; Photo ; Empreinte ; Code-ADN ; Remarque}.

L'algorithme sera chargé de fragmenter la table verticalement par rapport à une charge de travail, cette fragmentation aura pour objectif de réduire le temps d'exécution de la charge de travail.

L'algorithme génétique que nous allons appliquer aura les caractéristiques suivante : la taille de la population P est 100, l'initialisation de la population sera générés aléatoirement, la probabilité de croisement sera Pc, la probabilité de mutation sera Pm.

Sur la table personnes, la taille des chromosomes sera 10, puisque l'attribut Code ne sera pas pris en compte, pour raison de contraintes sur la clé primaire.

Dans ce qui suit nous allons simuler le déroulement de l'algorithme génétique sur cet exemple : P : {A1 ; A2 ; … ; A100} ;

Supposons que la longueur de chaque chromosome est égale à 10,

Soit : A1 = {5, 0, 5, 5, 0, 1, 1, 2, 3, 4} ; A2 = {2, 1, 1, 1, 0, 1, 1, 2, 3, 4} ; A3 = {….} ; etc.

Les chromosomes de la population seront évalués et leurs valeurs fitness correspondantes sera calculé par l'application de la formule du modèle de coût de Yao [33], supposons que les valeurs de fitness seront : *f*-A1 : 14520 ; *f*-A2 : 54620 ; *f*-A3 : 1458720 ; etc.…

Selon le mode de sélection plusieurs chromosomes seront choisis pour la reproduction, supposons que A1 et A2 seront choisies. L'algorithme procèdera au croisement et mutation, le schéma suivant illustrera le procédé de mutation et croisement sur A1 et A2 :

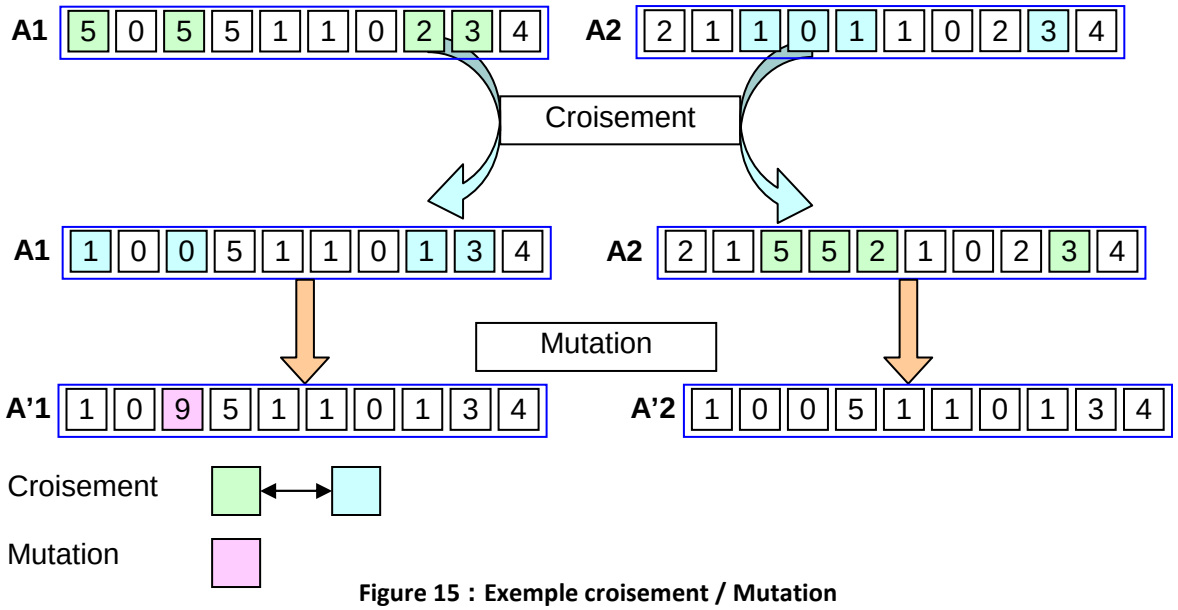


Figure 15 : Exemple croisement / Mutation

Sur la Figure 15, la procédure de croisement parcoure chaque gène du chromosome A1, et sélectionne des gènes selon la probabilité Pc (si Pc = 0.8, 80% seront sélectionnés), le même nombre de gènes sera sélectionné sur le chromosome A2, sur des points différents, une fois les emplacements du croisement définis, l'algorithme échangera les gènes marqués sur les chromosomes.

En suivant le même concept, une proportion de gènes sera sélectionnée selon la probabilité Pm. Les points marqués seront altéré avec une autre valeur aléatoire comprise entre 0 et 9. Nous remarquons que sur le chromosome n'a pas été altérer par la mutation car la probabilité Pm est assez faible.

Les chromosomes A1 et A2 ont générés deux nouveaux chromosomes A'1 et A'2, ces deux derniers formeront la nouvelle génération. Le procédé décrit si dessus sera appliqué sur tous les chromosomes choisis par l'algorithme de sélection.

III.3.6. DISCUSSION

Dans cette partie nous avons donné un aperçu global sur les algorithmes génétiques, leurs fonctionnements, et les différents modules qui les composent.

Les algorithmes génétiques ont fait leurs preuves dans la résolution d'un grand nombre de problèmes, comme le problème du voyage de commerce, ou problèmes non linéaires, comme les systèmes adaptatifs, problèmes d'optimisation multi-objectifs, comme pour les systèmes industrielles, ou de prise de décision. [11].

Les algorithmes génétiques présentent une grande souplesse et aisance dans leurs mise en œuvre et implémentation, c'est pour cette raison qu'un grand nombre de

[11] http://fr.wikipedia.org/wiki/Algorithme_génétique.

chercheurs et développeurs de systèmes industriels, informatiques, automatique et autres leurs portent un grand intérêt et les ont adopté.

Néanmoins, le bon choix du codage et des paramètres (probabilité de croisement, de mutation, nombre de générations et taille de la population) n'est pas une chose facile, car rien ne garantie que le codage soit approprié à notre problème, puisqu'un le codage représente un paramètre important et significatif pour le succès de l'algorithme et permet d'avoir de meilleurs résultats.

III.4. AFFINER

L'analyse et l'adaptation des algorithmes génétique et fouille de données, pour fragmenter verticalement une base de données par rapport à une charge de travail définie, nous a permis de déduire l'algorithme Affinité, un algorithme original conçu par nos soins, et que nous avons adapté pour la fragmentation verticale.

La motivation qui nous a poussée à proposer cette approche, est l'intuition sur le fait qu'il pouvait exister une approche plus adaptée au problème de la fragmentation verticale.

L'idée principale se base sur l'analyse et la l'évaluation de l'affinité entre chaque attribut, puis le regroupement des attributs fortement affine.

Un des atouts majeur de cet algorithme est qu'il effectue une recherche sur l'ensemble des attributs et en déduit partitions possibles sans utilisé un modèle de coût ce qui réduit énormément le temps de calcul, sachant que l'estimation du modèle reste théorique et peut changer d'un SGBD à un autre, d'une version à une autre, et d'une architecture à une autre.

En fait, après l'analyse de chaque attribut séparément, seul les partitions les plus intéressantes seront retenues pour la composition du schéma de fragmentation final.

Les attributs seront comparés un par un, par rapport à l'ensemble des autres attributs, en prenons en compte la charge de travail, en fait le calcul de l'affinité s'effectuera par rapport à chaque requête. Les attributs présentons une plus grande affinité seront regroupé dans la même partition, cette opération sera réitérée jusqu'à ce que l'ensemble des attributs soient évalué.

Dans ce qui suit nous allons donner une explication plus détaillée sur l'algorithme Affiner.

III.4.1. TERMINOLOGIE

- S ensemble des éléments candidats construits à partir des attributs de la table, exemple : S = {Nom, Prénom, …},
- Ai attribut numéro i,

- Ai(k) = 1, si l'attribut i est requis pour la requête k,
- Ai(k) = 0, si l'attribut i n'est pas requis pour la requête k,
- P = est l'ensemble des partitions, générés à partir de la combinaison des éléments S,
- Aff(i,j) = Affinité entre l'attribut i et j .
- B l'ensemble des partitions retenues.

Les termes que nous avons définis seront utilisés dans l'algorithme Affiner. L'algorithme se base sur l'évaluation de l'affinité entre tous les attributs par rapport aux autres attributs, si l'indice d'affinité entre deux attributs est élevé, un regroupement entre les attributs sera effectué, chaque regroupement présentera une partition, la partition retenue sera retiré de l'ensemble S, car il serra considéré comme un élément arrangé.

Le schéma de fragmentation sera la combinaison des éléments de « F » avec l'ensemble des éléments restant en une seule partition.

L'algorithme évalue le schéma de fragmentation afin qu'il puisse être comparé aux autres algorithmes implémenté dans notre étude.

L'algorithme sera détaillé et déroulé avec un exemple dans la section suivante :

III.4.2. DÉROULEMENT DE L'ALGORITHME AFFINER

- L'ensemble des candidats est généré par la décomposition élémentaire des attributs du problème,
- Génération des partitions candidates F, en combinant les éléments de S,
- à partir de l'ensemble des éléments B déduire le schéma de fragmentation.

L'algorithme Affiner est énoncé comme suit :

```
B ← ∅(ensembles des meilleurs fragments) ;
Aff(i,j) ←0 (initialisation de la matrice des attributs) ;
D ← nombre de requêtes ;
    S (l'ensemble des attributs)
    i ←1 (Compteur)
    Pour chaque élément de S faire
        Partition (i).add(Ai );
        S'←S- Ai;
        j←i+1;
        Pour chaque élément de S' faire
            k←1 ;
                Pour chaque requête  faire
                        Si (Ai(k)= Aj(k))
                                { Aff(i,j)++ ;}
                        K++;
                Fin faire ;
            j++;
            Si (Aff(i,j)= D) {  Partition (i).add (Aj)
                                S←S- Aj;}
        Fin faire ;
        B.add (Partition(i));
        i++;
    Fin faire;
    Return B;
```

Algorithme 2 : Algorithme Affiner.

Exemple : Soit Table T = {1, 2, 3, 4, 5} ; où {1, 2, …} représentent les attributs de T, nous allons dérouler l'algorithme Affiner sur cet exemple, à chaque itération de l'algorithme nous effectuons une estimation du temps d'exécution du schéma de fragmentation correspondant à la fragmentation en cour, cette estimation est renseigné dans la colonne coût.

La Figure 18 illustre l'application de Affiner; il est à noter que l'exemple est inspirer d'un environnement totalement théorique, le but de cet exemple est de comprendre le fonctionnement de Affiner.

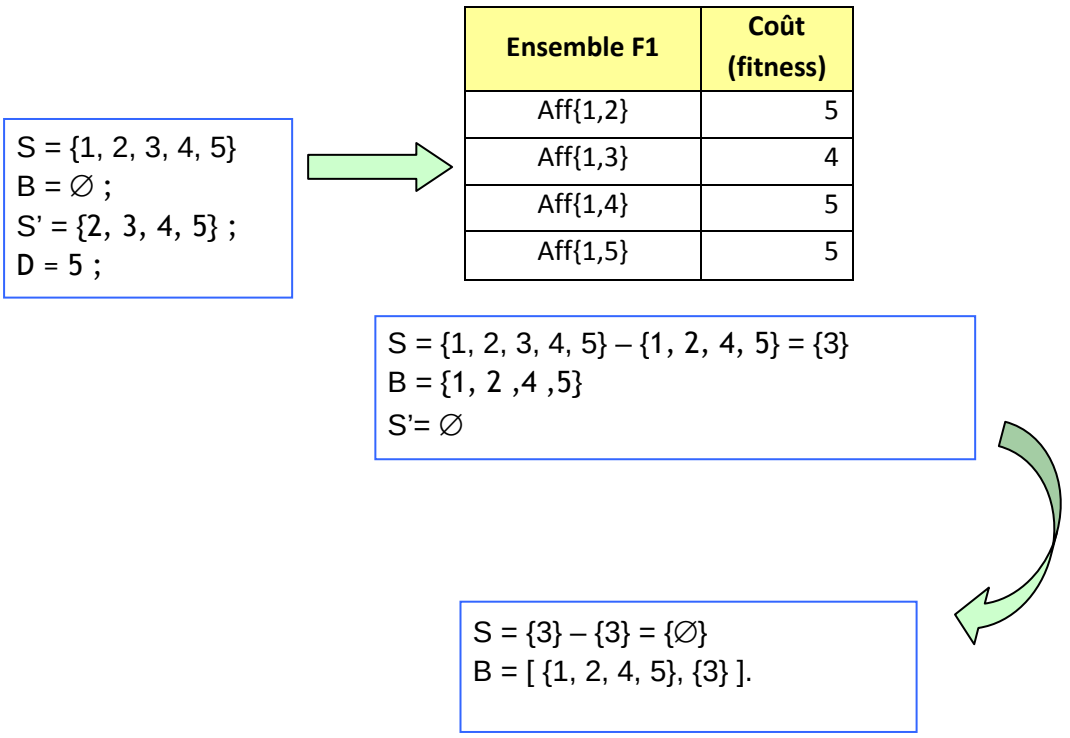


Figure 16 : illustration du déroulement de Affiner.

Le simple calcul de l'affinité entre les attributs nous permet de déduire les partitions à générer, où S représente l'ensemble des attributs potentiels pour la création des partitions, et B représente l'ensemble des attributs appartenant aux partitions retenues pour la formation du meilleur schéma.

Le critère d'arrêt de cet algorithme est l'épuisement de l'ensemble S.

Comme résultat nous obtenons le schéma de fragmentation {{1, 2, 4, 5}, {3}} déduit de l'ensemble B obtenu dans la phase finale.

III.5. CONCLUSION :

Dans ce chapitre nous avons présenté trois algorithmes Apriori, Génétique, et Affiner, cette présentation s'est accompagnée d'une étude détaillée sur ces algorithmes.

Dans le même contexte nous avons présenté l'utilisation des ces algorithmes pour donner une solution de fragmentation verticale.

L'algorithme Affiner est caractérisé par un temps d'exécution réduit, aussi nous notons que Affiner ne nécessite pratiquement aucun paramétrage par rapport aux algorithmes génétiques (Codage ; Pc ; Pm ; taille population ; etc.) et à Apriori (minSup).

Cette étude nous a fournis les éléments essentiels pour la compréhension des trois algorithmes suscités, et nous a fourni un terrain préparatoire et les éléments nécessaires pour faciliter l'implémentation et la mise en œuvre d'un outil de test.

Enfin, pour valider notre approche, nous avons intégré Affiner et les algorithmes Génétiques et Apriori dans l'outil FragMan, que nous avons développé et qui est présenté dans le chapitre suivant.

IV. FragMan.

IV.1. INTRODUCTION

Dans ce chapitre nous allons présenter l'outil *FragMan*. *FragMan* a pour fonction l'assistance des administrateurs de bases de données dans le partitionnement vertical et efficace de la base de données.

Les recommandations que nous procure l'outil sont prises sur la base d'une estimation du temps d'exécution de la charge de travail sur chaque configuration proposée, et en extrait le meilleur schéma de fragmentation qui prend en considération la charge de travail soumise sur la base.

FragMan à été programmé sur la base du langage Java. L'API Swing de java que nous avons choisis pour la programmation, nous à facilité la création d'interface graphique.

Java intègre plusieurs bibliothèques intéressantes comme JDBC : Java Data Base Connector, bibliothèque qui gère efficacement la connexion aux bases de données.

Aussi la bibliothèque JFreeChart bibliothèque permettant la création des graphiques.

Java est un langage très puissant, multiplateforme, et très robuste, car les programmes développés avec Java s'exécutent indépendamment de la machine, et du système d'exploitation, sur la machine virtuelle Java, ce qui diminue le risque du blocage du programme, et évite les exceptions.

Le langage Java à été adopté par une grande communauté de développeurs de toute catégorie confondue, en effet Java est langage assez riche.

Java est riche par les bibliothèques open source développées par la grande communauté Java, des bibliothèques très intéressantes comme JGAP, « Java Genetic Algorithms Package », pour le développement des programmes JGAP [12], et tant d'autre.

Dans les paragraphes suivants, nous allons donner une explication détaillée sur l'outil *FragMan*, les différentes fonctionnalités que nous offre cet outil, suivi des interfaces expliquant le mode de fonctionnement de *FragMan*.

[12] http://jgap.sourceforge.net

IV.2. FONCTIONNALITÉ DE FRAGMAN

FragMan présente fonctionnalités suivantes :

- supporte la connexion sur les SGBD les plus utilisés grâce à un module d'interrogation de la méta-base.
- permet de disposer de toutes les données de l'entrepôt à partir de la méta-base (tables, attributs, …).
- assure la gestion des attributs (sélection, élagage) et la gestion des requêtes (élagage manuel, élagage automatique)
- comporte un analyseur des requêtes SQL développé avec les outils GNU Flex et Bison.
- implémente trois algorithmes de fragmentation verticale : algorithme Apriori, Algorithme Génétique, Algorithme Affiner.
- permet la génération des scripts SQL qui permettent de créer les partitions directement sur l'entrepôt réel.
- fournit à l'administrateur une simulation sur l'estimation de la qualité de la configuration des partitions par des diagrammes.

IV.2.1. ARCHITECTURE ET MODE DE FONCTIONNEMENT

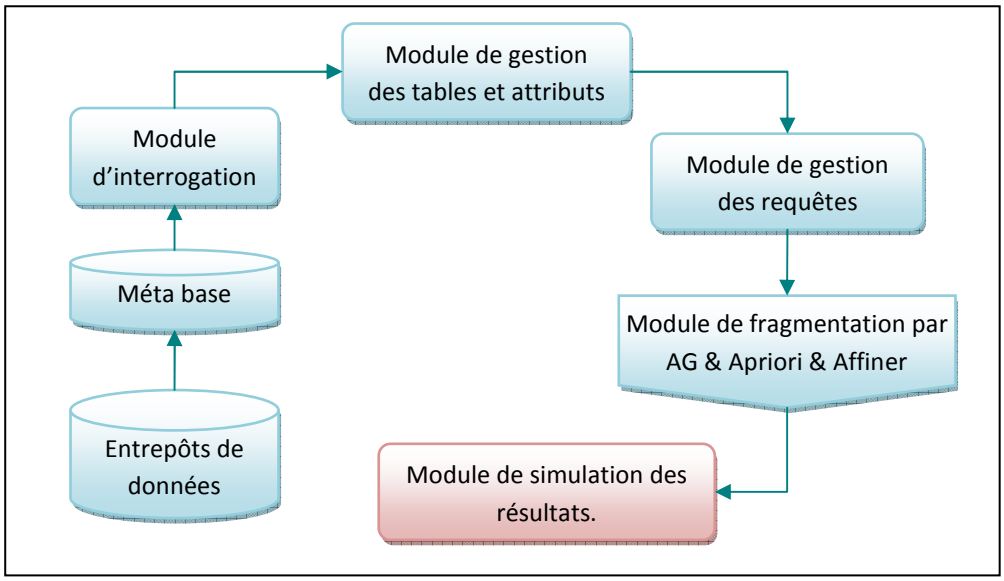


Figure 17 : Architecture de FragMan

FragMan suit la démarche suivante : en premier lieu il commence par l'exploration et la lecture du schéma et de la méta-base de la base de données, puis il effectue la lecture et l'analyse de la charge de travail, enfin il propose une configuration pour la fragmentation verticale adéquate pour le système existant.

FragMan est décomposé en cinq modules (voir Figure 17), une fois lancé, les taches s'exécuterons séquentiellement, il est possible de faire des retours en arrière afin de changer quelques paramètres.

1. **Module d'interrogation :** ce module nous permet d'interroger la méta-base du SGBD afin d'en extraire toutes les informations utiles (nom des tables, leurs attributs, les clés primaires et étrangères, cardinalité des attributs…) ainsi que les statistiques liés à l'entrepôt. Cette phase nous permet d'obtenir une simulation de la base de données dans l'outil.
2. **Module gestion des tables :** ce module a pour rôle de choisir la table et les attributs qui seront concerné par la fragmentation.
3. **Module gestion des requêtes :** ce module permet de charger les requêtes SQL à partir d'un fichier « Sql ». Une fois les requêtes chargées, l'outil lance l'analyseur «parseur.exe». L'analyseur se charge d'analyser les requêtes lexico-syntaxiquement et d'en extraire les éléments nécessaires pour les traitements.

   Une fois l'analyse achevée, les résultats sont stockés dans un fichier de sortie «out.data». L'outil récupère ce fichier et l'analyse. Si la syntaxe du code SQL est valide, l'outil affiche les attributs référencés par ces requêtes et procède un élagage automatique de ces derniers. Dans le cas contraire une boite de dialogue est affichée indiquant les requêtes erronées.
4. **Module de fragmentation :** Ce module permet de sélectionner un schéma de FV proche de la solution optimale via trois algorithmes qui sont implémentés dans ce module.
5. **Module d'évaluation :** il permet d'estimer le coût d'exécution des requêtes (charge de travail) sur le schéma fragmenté. Il se base sur le modèle de coût proposé dans [17].

   Les résultats sont affichés sous forme de graphiques en utilisant l'API JFreeChart.

IV.3. INTERFACES DE FRAGMAN :

Dans la section suivante nous présenterons quelques vues des interfaces qu'intègre l'outil FragMan, l'ordre des figures à été pris sur la base de l'acheminement et la logique que l'outil suit pour une fragmentation verticale sur un entrepôt de données réel que nous avons créé à partir du banc d'essai APB1[8].

La figure suivante montre l'interface principale où l'administrateur pourra se connecter, ajouter un serveur de BD ou une chaine de connexion.

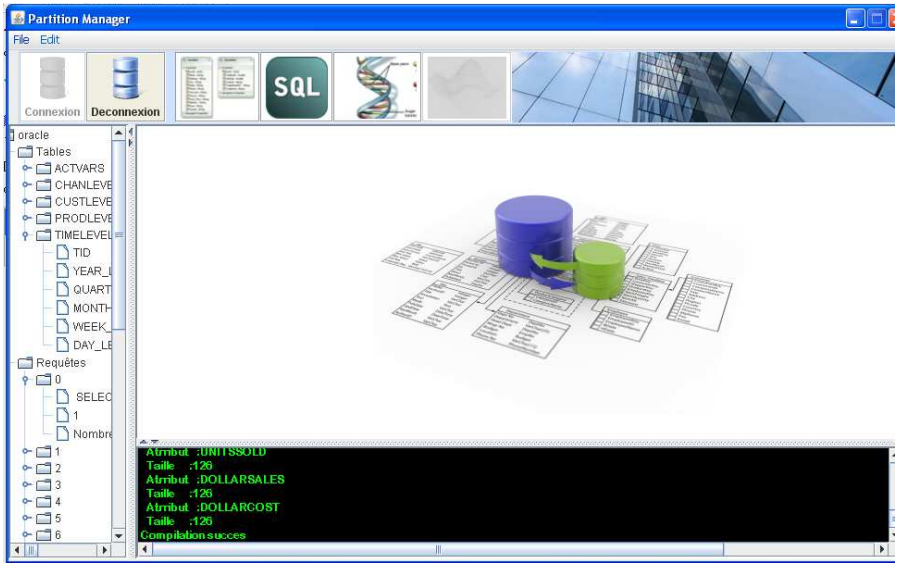


Figure 18 : Fenêtre principale

Le module de connexion est montré dans la figure suivante, il permet de choisir la base de données et de saisir la chaine de connexion (utilisateur, mot de passe, base de données, adresse IP du serveur base de données, etc.).

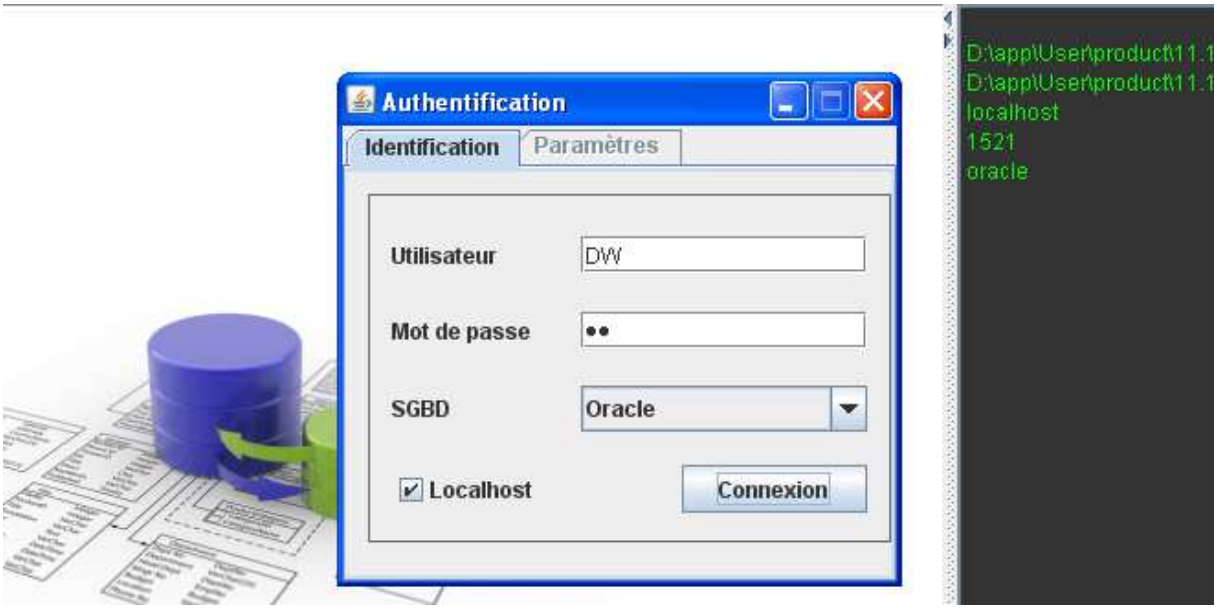


Figure 19 : Console de connexion

Une fois la connexion établie, la console FragMan effectue un scan de la base de données, et en collecte les différentes informations utiles pour le déroulement de la fragmentation verticale, ensuite, l'outil enchainera par le déroulement automatique

des taches à effectuer. Les figures suivantes nous donnent un aperçu sur la procédure.

La Figure 16 représente l'interface qui permet à l'utilisateur de choisir une table pour la fragmentation dans une liste contenant toute les tables de l'entrepôt.

Une fois que la table sélectionnée, le bouton (2) permet de transposer la table choisie vers la liste (3), et le bouton (4) de validation pour confirmer la table choisie. Le bouton (5) est le bouton de fin et de fermeture de cette fenêtre.

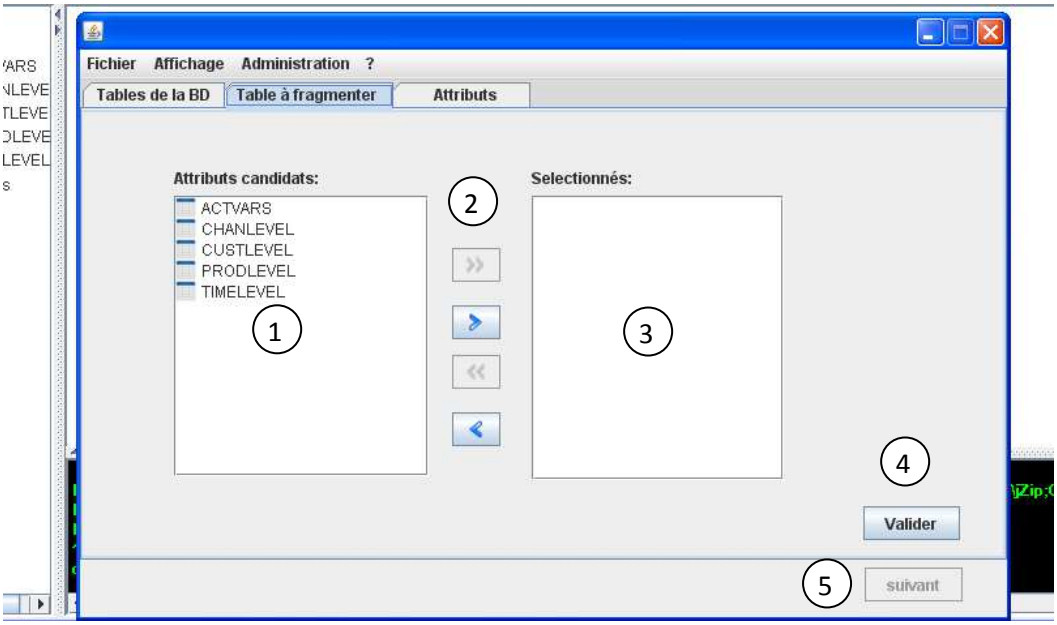


Figure 20 : Interface d'affichage des relations de l'entrepôt.

Une fois le bouton suivant activé, la fenêtre de gestion des requêtes apparaitra (voir Figure 2120).

Cette fenêtre aura pour rôle de choisir un fichier plat contenant l'ensemble des requêtes qui s'exécuterons sur la base, une fois définis et validés, un interpréteur syntaxique et grammatical se chargera d'analyser les requêtes et d'en extraire les informations utiles.

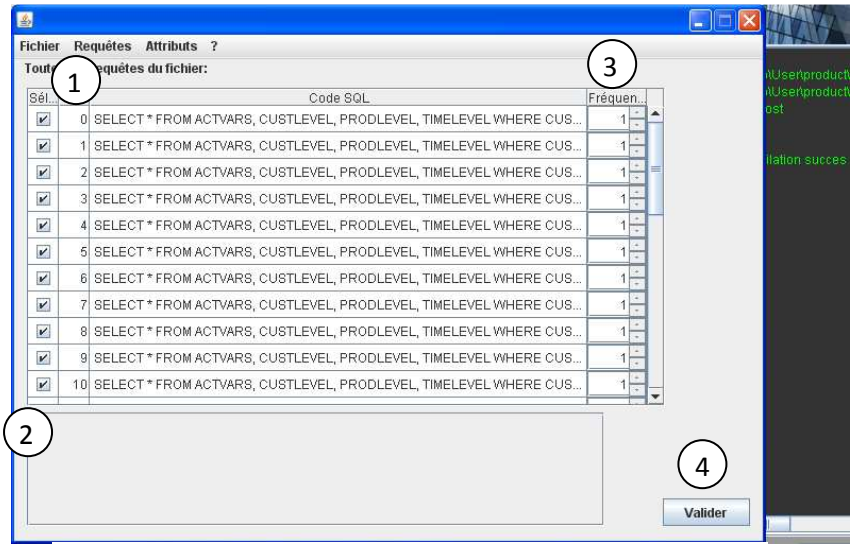


Figure 21 : Module gestion des requêtes

La Figure 21 donne un aperçu sur la fenêtre de sélection des requêtes et leurs analyses, dans cette partie l'utilisateur pourra choisir dans (1) un fichier d'extension SQL, ce fichier en principe est un fichier plat qui contient les différente requête SQL représentant la charge de travail.

L'utilisateur a la possibilité d'élaguer des requêtes par le biais des cases à cocher (2), et de spécifier la fréquence de chaque requête avec (3), par défaut de chaque requête est 1, et enfin le bouton de validation (4) pour confirmer la saisie.

Derrière cette interface il se cache le module d'analyse syntaxique et grammatical basé sur les outils LEX et Yacc.

LEX et YACC sont des outils très populaires de génération d'analyseurs lexicaux (Lex) et syntaxiques (Yacc) en langage C, Yacc est l'acronyme de « Yet anothet compiler-compiler ».

L'outil LEX utilise un ensemble de règles pour générer un programme, appelé un analyseur lexical, qui analyse les entrées et les interprètes en plusieurs catégories, numéros, lettres et opérateurs. FLEX[13] est La distribution open source disponible.

L'outil YACC utilise un ensemble de règles pour générer un programme, appelé analyseur grammatical, qui analyse les entrées en utilisant des catégories définies par l'analyseur lexical et détermine ce qu'il faut faire à l'entrée. BISON[14] est la distribution open source disponible.

[13] Flex : http://flex.sourceforge.net/

[14] Bison : http://www.gnu.org/software/bison/

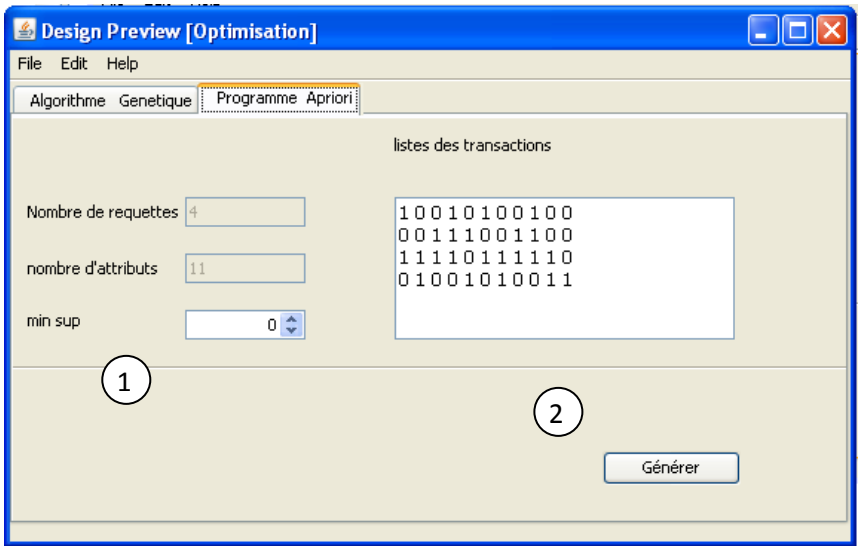


Figure 22 : Module Optimisation Apriori

La Figure 22 : Module Optimisation Apriori est l'interface qui permet de lancer la procédure de calcul de la fragmentation verticale la plus bénéfique.

Dans le programme Apriori le seul paramètre à instruire est la valeur Min Sup (1), sur laquelle l'algorithme se base pour effectuer ces calculs.

Le lancement de l'opération s'effectue par le bouton (2). Les résultats ressortent dans la fenêtre principale.

L'optimisation par algorithme génétique est incluse dans l'interface représentée dans la figure 93.

L'utilisateur peut renseigner plusieurs paramètres dont la probabilité de croisement des individus de la population (1), probabilité de mutation (2) de chaque individu qui doit être faible, taille des individus de chaque population (3), nombre de générations à produire (4), et enfin la méthode de sélection des individus pour construire les nouvelles générations (5).

Le bouton (6) est le bouton qui sert à déclencher l'algorithme, les résultats apparenteront dans la fenêtre principale.

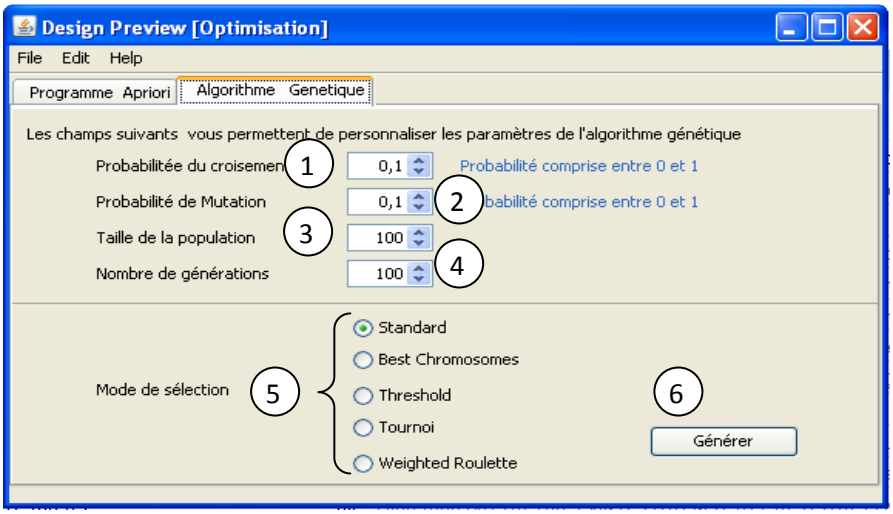


Figure 23 : Optimisation par Algorithme Génétique.

Le troisième volet que nous avons intégré est celui de l'algorithme Affiner, comme indiqué dans la Figure 23 , aucun paramètre n'est à renseigné il suffit juste de presser le bouton lancement, et les résultats s'afficherons sur la fenêtre principale.

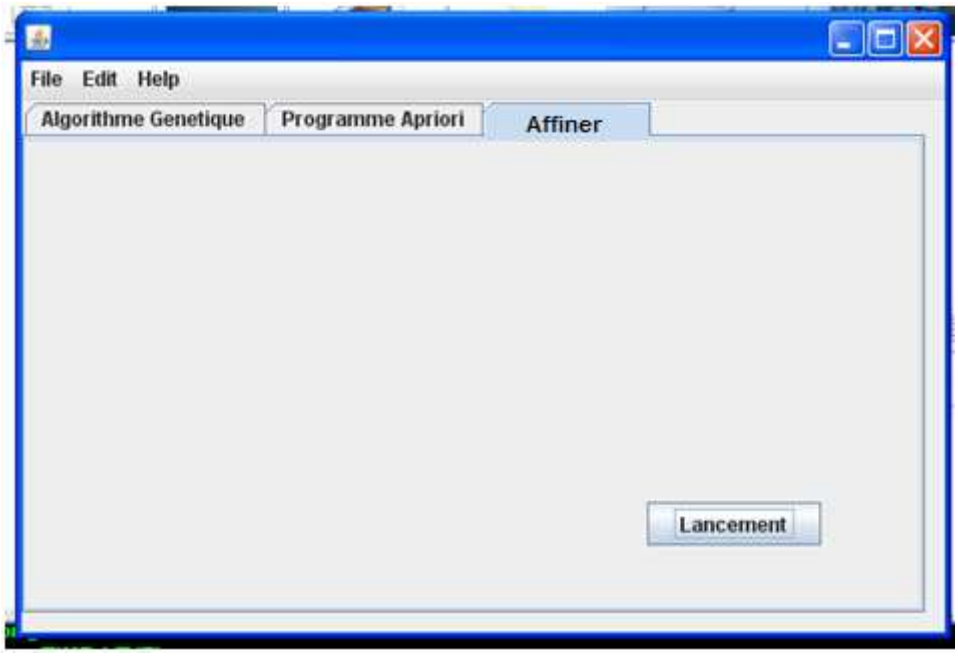


Figure 24 : Volet Algorithme Affiner

La Figure 24 représente l'interface des résultats. A partir de la configuration des partitions verticaux générés, et sur la base d'un modèle de coût FragMan prononce un diagnostic sur l'état du changement est estime le coût de l'exécution de la charge de travail pour la configuration proposé.

L'axe des abscisses représente les différents algorithmes appliqués, la colonne rouge estime le coût d'une table non partitionnée, et l'axe des ordonnées représente l'estimation du coût des solutions proposées basé sur une simulation.

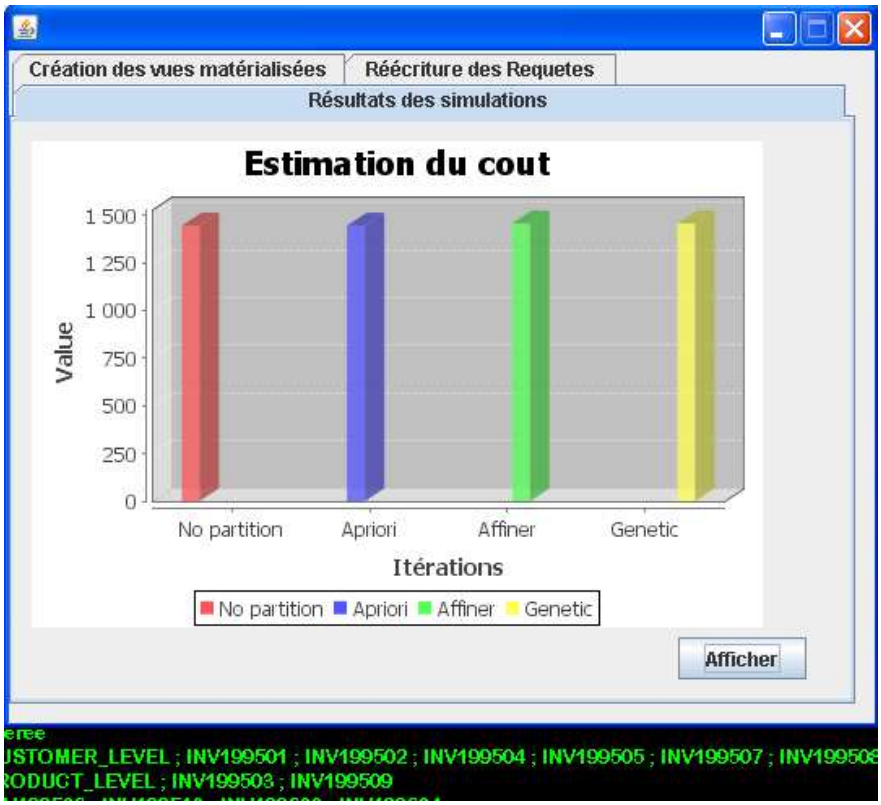


Figure 25 : Résultat de la simulation.

Après avoir expliqué le fonctionnement de l'application, dans la section suivante nous allons entamer la phase de test.

IV.4. TEST ET RÉSULTATS

Les algorithmes implémentés dans FragMan se basent sur une estimation théorique et offre la possibilité de fragmenter un schéma virtuelle, afin de valider cette étude et valoriser nos algorithmes nous avons été amenés à procéder à des tests dont nous allons présenter les différents résultats obtenus par l'application des recommandations de FragMan sur une base de données réelle.

Sur les différents bancs de tests ou benchmarks existants nous avons opté pour l'utilisation du Benchmark OLAP APB1 Release II [8], un benchmark offrant la possibilité de créer des schémas de bases de données et des table assez volumineuse ce qui nous permet de mener à bien notre recherche. Aussi APB1 Release II [8] est très facile à implémenter, aussi le mode de génération de la base de données et du script ne nécessite pas de très grandes connaissances dans le domaine des bases de

données, en plus nous avons noté que le benchmark APB1 est universelle par rapport à tous les SGBD (Oracle, IBM, …).

Pour notre application nous avons générer le schéma APB1 RII [8], et nous avons choisi de fragmenter la table des faits de HistInventory composé de 19 attributs contenant 27.700.646 enregistrements :

Hist_Inventory { customer_level char(12), product_level char(12),
inv199501 Integer, inv199502 Integer,
inv199503 Integer, inv199504 Integer,
inv199505 Integer, inv199506 Integer,
inv199507 Integer, inv199508 Integer,
inv199509 Integer, inv199510 Integer,
inv199511 Integer, inv199512 Integer,
inv199601 Integer, inv199602 Integer,
inv199603 Integer, inv199604 Integer,
inv199605 Integer}.

Le SGBD ou moteur de base de données sur lequel nous avons effectué nos tests est Oracle 11G, une version très stable de SGBD, largement utilisé dans les entreprises manipulant les bases de données de type OLAP.

Sachant que le SGBD Oracle ne supportant pas la fragmentation verticale nous avons été contraint de dupliquer la clé primaire à la table Hist_Inventory, afin que l'on puisse effectuer une jointure entre les différents partitions générées.

Une fois les recommandations énoncées par FragMan, on procède à la génération des vues matérialisées (snap shot) représentant les partitions recommandées par FrgaMan, puis on effectue une réécriture de toutes les requêtes composant la charge de travail pour qu'elles puissent s'exécutées selon les vues matérialisées générées.

Dans cette partie on a effectué plusieurs tests sur la base de données qui sont illustrés ci-dessous :

IV.4.1. PREMIER TEST :

Dans ce premier test, on a choisi une charge de travail contenant quatre requêtes s'exécutant spécialement sur la table HistInventory, après le lancement de FragMan, on obtient les recommandations qu'on implémente manuellement sur la base de données, et on obtient les temps d'exécution réels de la charge de travail sur le nouveau schéma partitionné.

Charge de travail:

```
Q1 :    select  product_level, inv199503, inv199509
        from histinventory
        where inv199503> 10 and inv199509 < 15;
Q2 :     select  product_level, inv199503, inv199505, inv199509, inv199602
        from histinventory
        where inv199506> 10 and inv199511< 12;
Q3 :     select  inv199503, inv199509
        from histinventory
        where product_level like '%M';
Q4 :     select  inv199503, inv199505, inv199506, inv199511
        from histinventory
        where product_level like '%K' and inv199509> 10 and inv199602< 12;
```

Avec les fréquences suivantes : fréquence (Q1) = 3, fréquence (Q2) = 3, fréquence (Q3)=3, fréquence (Q4)=3.

Une fois FragMan exécuté, on obtient les résultats suivants :

```
Genetic Algorithm:

  Total evolution time: 1234 ms

  Partition : CUSTOMER_LEVEL ; INV199501 ; INV199502 ; INV199504 ; INV199505
  Partition : PRODUCT_LEVEL ; INV199503 ; INV199509
  Partition : INV199506 ; INV199510 ; INV199603 ; INV199604
  Partition : INV199507
  Partition : INV199508
  Partition : INV199511
  Partition : INV199512 ; INV199601
  fitness : 1458.0 ;

Apriori Algorithm:

  Execution time is: 0.002 seconds.

  Partition : INV199506 ; INV199510 ; INV199603 ; INV199604
  Partition : PRODUCT_LEVEL ; INV199503 ; INV199509
  Partition : CUSTOMER_LEVEL ; INV199501 ; INV199502 ; INV199504 ; INV199505 ; INV199507 ;
              INV199508 ; INV199511 ; INV199512 ; INV199601 ; INV199602 ; INV199605
  fitness : 1458.0 ;

Affiner Algorithm

  Execution time is: 0.0 seconds.

  Partition : CUSTOMER_LEVEL ; INV199501 ; INV199502 ; INV199504 ; INV199505 ; INV199507 ;
              INV199508 ; INV199511 ; INV199512 ; INV199601 ; INV199602 ; INV199605
  Partition : PRODUCT_LEVEL ; INV199503 ; INV199509
  Partition : INV199506 ; INV199510 ; INV199603 ; INV199604
  fitness : 1458.0
```

Les temps d'exécutions des requêtes sont représentés sur la figure suivante :

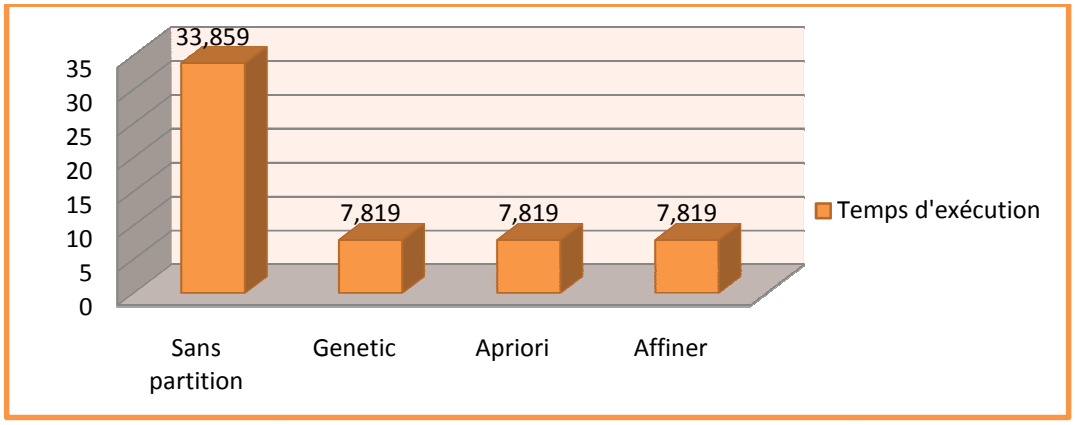


Figure 26 : Temps d'exécution Test 1.

Le gain en performance en temps d'exécution de la charge de travail à diminuer d'une manière significative avec un taux de 80%.

IV.4.2. DEUXIÈME TEST :

Charge de travail :

```
Q1 :    select a.inv199603
        from histinventory a
        where inv199604>20
Q2 :    select INV199603 , INV199604
        From histinventory a
        Where INV199603 >5;
```

Avec les fréquences suivantes :

Fréquence (Q1) = 1, fréquence (Q2) = 2,

Une fois FragMan exécuté, et les recommandations appliquées sur la base de données, les temps d'exécutions des requêtes sur le nouveau schéma sont représentés sur la figure suivante :

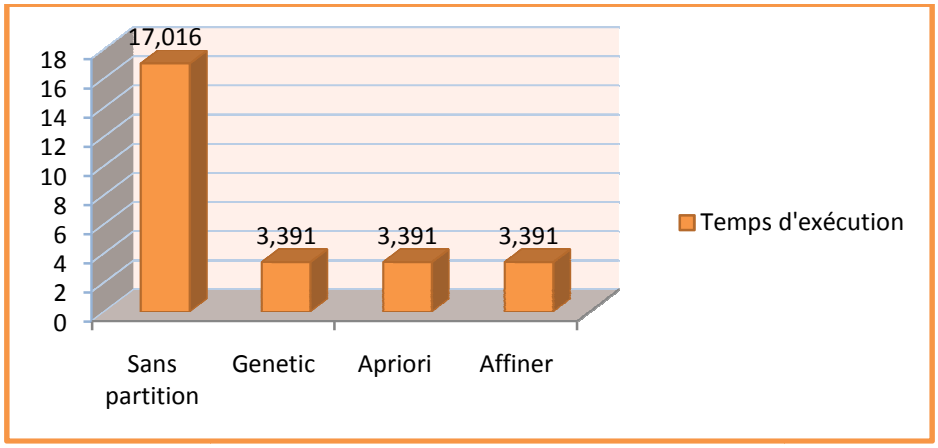


Figure 27 : temps d'exécution Test 2

IV.4.3. TROISIÈME TEST :

Charge de travail :

```
Q1 :    select  product_level,inv199503,inv199505,inv199506
        from histinventory
        where inv199509 > 2 and inv199511 <2 and inv199602 >2;
Q2 :    select  product_level,inv199503,inv199511,inv199602
        from histinventory
        where Inv199509 > 2 and inv199603>15 and inv199604 >10;
Q3 :    select  product_level,inv199503,inv199505, inv199603, inv199602
        from aahistinventory
        where inv199509 > 2 and inv199511 <2 and inv199604 >2 and inv199506=5;
Q4 :    select  product_level, Inv199509 ,inv199503,inv199511,inv199602
        from histinventory
        where product_level Like '%T'and inv199603>15 and inv199604 >10;
Q5 :    select  inv199509,inv199503,inv199505,inv199506
        from histinventory
        where product_level Like '%T' and inv199511 <2 and inv199602 >2;
Q6 :    select  product_level,inv199503,inv199505,inv199506, inv199603,inv199604
        from aahistinventory
        where inv199509 > 2 and inv199511 <2 and inv199602 >2;z
```

Avec les fréquences suivantes : fréquence (Q1) = 1, fréquence (Q2) = 1, fréquence (Q3) = 1, fréquence (Q4) = 1, fréquence (Q5) = 1, fréquence (Q6) = 1,

Une fois FragMan exécuté, on obtient les recommandations suivantes :

```
Genetic Algorithm:

  Total evolution time: 1454 ms

  Partition : CUSTOMER_LEVEL ; INV199501 ; INV199502 ; INV199504
  Partition : PRODUCT_LEVEL ; INV199503 ; INV199509 ; INV199511 ; INV199602
  Partition : INV199505 ; INV199506
  Partition : INV199507 ; INV199508 ; INV199510 ; INV199512 ; INV199601
  Partition : INV199603 ; INV199604
  Partition : INV199605
  fitness : 3904.0

Apriori Algorithm:

  Execution time is: 0.047 seconds.
  Solution générée
  Partition : PRODUCT_LEVEL ; INV199503 ; INV199509 ; INV199511 ; INV199602 ; INV199603 ;
              INV199604
  Partition : INV199505 ; INV199506
  Partition : CUSTOMER_LEVEL ; INV199501 ; INV199502 ; INV199504 ; INV199507 ; INV199508 ;
              INV199510 ; INV199512 ; INV199601 ; INV199605
  fitness : 4456.0

Affiner Algorithm

  Execution time is: 0.016 seconds.
  Partition : CUSTOMER_LEVEL ; INV199501 ; INV199502 ; INV199504 ; INV199507 ;
              INV199508 ; INV199510 ; INV199512 ; INV199601 ; INV199605
  Partition : PRODUCT_LEVEL ; INV199503 ; INV199509 ; INV199511 ; INV199602
  Partition : INV199505 ; INV199506
  Partition : INV199603 ; INV199604
  fitness : 3896.0
```

Les recommandations appliquées sur la base de données, on obtient les temps d'exécutions des requêtes sur le nouveau schéma qui sont représentés sur la figure suivante :

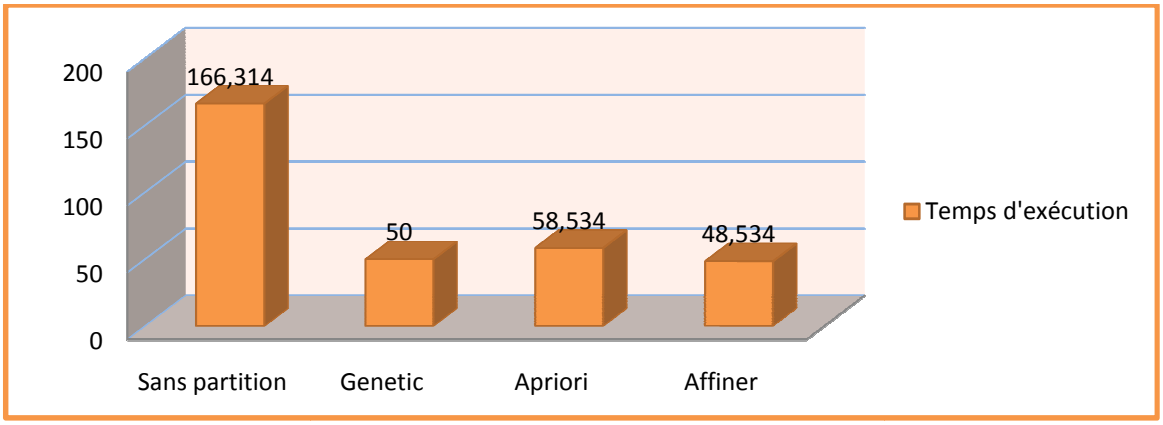


Figure 28 : temps d'exécution Test 3

Sur la Figure 28, on remarque que l'algorithme Affiner nous a procuré les meilleures recommandations, en deuxième position viens l'algorithme Génétique qui a donné des résultats presque aussi similaires, et en troisième position l'algorithme Apriori a donné des solutions plus couteuses en temps d'exécution par rapport à Affiner et Génétique.

IV.5. CONCLUSION

L'outil FragMan est un outil qui permet de procurer des recommandations de schéma pour la fragmentation verticale, nous avons intégré trois programmes Apriori, basé sur Datamining, Affiner, basé sur l'approche que nous avons développée, et Génétique, basé sur les algorithmes génétiques.

Nous avons essayé d'intégrer plusieurs fonctionnalités utiles à cet outil. L'outil pourrait bien contribuer au le développement d'une application riche et complète dans l'assistance des administrateurs de bases de données à la conception physique, et ainsi permettre d'améliorer les performances des bases de données.

Enfin nous avons effectué quelques tests sur une base de données APB1 RII [8], sur une machine HP 7800 convertible tower avec un processeur Core 2 duo 2.66 GHz et une mémoire cache de 4 MO, et 4 GO de RAM. La table fragmentée (histinventory) contenait plus de 27 millions d'enregistrements.

Les résultats obtenus été très satisfaisant et très concluant, les meilleurs résultats on été enregistré du coté de Affiner, les performances de la base de données en temps de réponse ont augmentées avec une moyenne de 70%.

V. Conclusions & Perspectives.

Conclusions & perspectives

Dans cette étude nous nous somme intéressés au sujet de la fragmentation verticale sur les bases de données, notre choix s'est porté sur l'utilisation de deux variantes algorithmiques basées sur l'évaluation par le modèle de coût, l'algorithme génétique et l'algorithme Apriori, et comme contribution nous en avons proposé un troisième algorithme que nous l'avions baptisé Affiner.

Cette étude nous à permis de simuler l'application de la fragmentation verticale sur les bases et entrepôts de données. Nous somme parvenus à proposer dans ce travail, un algorithme appelé Affiner. Affiner, est une approche qui joue le rôle d'éclaireur pour trouver le meilleur schéma pour la fragmentation verticale.

L'outil FragMan pourrait servir pour assister les administrateurs de bases de données dans l'optimisation des performances de la base de données, et pourrait servir comme un banc d'essais offrant aux gens intéressés par la fragmentation verticale la possibilité de tester et mettre en pratique cette technique de conception physique.

Les tests ont démontré l'efficacité et le gain apporté quand à l'utilisation de la fragmentation verticale sur une base de données, bien que ces tests ne représentent que quelques formes de requêtes, néanmoins la fragmentation verticale semble bien répondre à l'optimisation des performances d'une base de données assez volumineuse (entrepôts de données) traitant des requêtes de type OLAP.

Plusieurs améliorations peuvent être ajoutées à FragMan. Nous commençons par l'ajout de plusieurs choix sur les modèles de coût. Aussi, nous pouvons ajouter un module qui permet d'adapter et d'orienter l'utilisateur à choisir l'algorithme et les paramètres les plus adéquats à la base de données en étude.

L'algorithme Affiner a bien répondu à la résolution du problème de la Fragmentation Verticale, l'amélioration qu'a apportée Affiner par rapport aux autres algorithmes est que Affiner ne nécessite aucun paramétrage, pas de modèle de coût, et un temps de réponse très réduit pour proposer un schéma optimisé, et les résultats qu'a fourni Affiner ont été les meilleur du coté des performances de la base de données. Affiner peut être généralisé sur d'autres techniques de conception physique, comme la fragmentation horizontale.

La forme la plus connue pour implémenter la Fragmentation Verticale sur les BD consiste à créer des vues matérialisées représentant chacune une partition. Nous estimons que cette forme ne répond pas aux attentes des gestionnaires de bases de données, et que la fragmentation verticale trouvera un sens seulement si nous trouverons la solution pour l'appliquer directement et efficacement sur la base de données, nous voulons continuer dans cette voie et contribuer et proposer une architecture adéquate pour l'application de la FV sur les bases de données.

BIBLIOGRAPHIE

1. Abdalla HI and Marir F. "Vertical Partitioning Impact", in 18th National Computer Conference 2006, Saudi Computer Society.
2. Agrawal, R and Srikant, R. "Fast algorithms for mining association rules in large databases". in 20th International VLDB, pages 487-499, Santiago, Chile, September 1994.
3. Agrawal S, Surajit Chaudhuri and Vivek R. Narasayya, "Automated Selection of Materialized Views and Indexes in SQL Databases", Proceedings of 26th International Conference on Very Large Data Bases, 2000: 496-505.
4. Agrawal S, Vivek R. Narasayya, Beverly Yang, "Integrating Vertical and Horizontal Partitioning Into Automated Physical Database Design". SIGMOD Conference 2004.
5. Agrawal, S. et al. "Database Tuning Advisor for Microsoft SQL Server 2005". In Proceedings of the 30th VLDB, Toronto, Canada, 2004.
6. Agrawal, S. et al. "Database Tuning Advisor in SQL Server 2005". White paper. http://www.microsoft.com/technet/prodtechnol/sql/2005/sql2005dta.mspx
7. Angel, F. & al. Taddei-Zavala "Simultaneous Vertical Fragmentation and Segment Assignment in Distributed Data Bases using Genetic Algorithms".
8. APB1 OLAP Council, "APB-1 olap benchmark, release II," http://www.olapcouncil.org/research/bmarkly.htm.
9. Avnur, R., and Hellerstein, J. Eddies: "Continuously Adaptive Query Processing". In Proceedings of ACM SIGMOD Conference 2000.
10. Cardenas, A. F. "Analysis and performance of inverted data base structures". Comm. ACM 18, 5 (Mai 1975); 253 -263.
11. Ceri, S. & Pernici,S. and Weiderhold,G. "Optimization Problems and Solution Methods in the Design of Data distribution". Information Sciences Vol 14, No. 3, p 261-272, 1989.
12. Chakravarthy, S., Muthuraj, J., Varadarajan, R., and Navathe, S. B. "An objective function for vertically partitioning relations in distributed databases and its analysis". Distributed and Parallel Databases 2, 2 (1994), 183–207.
13. Chaudhuri, S. and Narasayya, V. An Efficient, "Cost-Driven Index Selection Tool for Microsoft SQL Server". In Proceedings of the VLDB, Athens, Greece, 1997.
14. Chaudhuri, S. and Narasayya, V. AutoAdmin "What-If" Index Analysis Utility. In Proceedings of ACM SIGMOD, Seattle, WA, 1998.
15. Chaudhuri, S. And Vivek R. Narasayya, "Microsoft Index Tuning Wizard for SQL Server 7.0", Proceedings ACM SIGMOD International Conference on Management of Data, 1998: 553-554.

16. Cheng. C.H; & Lee, W-K; Wong, K-F, "A Genetic Algorithm-Based Clustering Approach for Database Partitioning" IEEE Transactions on Systems, Man, and Cybernetics, 32(3), 2002, 215-230. 33.
17. Chu, P.-C. "A transaction oriented approach to attribute partitioning". Information Systems 17, 4 (1992), 329–342.
18. Cornell, D., and Yu, P. "A vertical partitioning algorithm for relational databases". In International Conference on Data Engineering (1987), pp. 30–35.
19. Dageville, B., Das, D., Dias, K., Yagoub, K., Zaït, M., Ziauddin, M. "Automatic SQL Tuning in Oracle 10g". In Proceedings of VLDB 2004.
20. Goldberg D. "GENETIC ALGORITHMS IN SEARCH, OPTIMIZATION AND MACHINE LEARNING". Reading MA Addison Wesley, 1989.
21. Goldberg D., "Algorithmes génétiques, Adisson Wesley, France", 1994.[Gol89c]
22. Golfarelli, M. & Maio, D. & Rizzi, S. (1999). "Vertical Fragmentation of Views in Relational Data Warehouses". Settimo Convegno Nazionale su Sistemi Evoluti Per Basi Di Dati (SEBD 99), Como, Italy (pp. 19–33).
23. Gorla, N. "A Methodology for Vertically Partitioning in a Multi-Relation Database Environnement", JCS&T Vol.7 No. 3 October 2007.
24. Gorla, N. & Pang Wing "vertical fragmentation in Databases Using Data-Mining technique", IGI Global Vol.4, Issue 3. 2008.
25. Hammer, M. & Niamir, B. "A heuristic approach to attribute partitioning. In Proceedings ACM SIGMOD Int. Conf. on Management of Data", (Boston, Mass., 1979), ACM, New York.
26. Hoffer, J. & Severance,D. "The Uses of Cluster Analysis in Physical Database Design", Proc in 1st International Conference on VLDB, Framingham, MA, 1975.
27. Hoffer, J. "An integer programming formulation of computer database design problems". Inf. Sci., 11(July 1976), 29-48.
28. Holland J.H. "Adaptation in natural and artificial system", Ann Arbor, The University of Michigan Press, 1975.
29. Lightstone, S., Teorey, T., and Nadeau, T. "Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more", Morgan Kaufmann Press, 2007. ISBN: 0123693896.
30. Lin, X., Orlowska, M., and Zhang, Y. "A graph based cluster approach for vertical partitioning in database design". Data and Knowledge Engineeering 11, 2 (1993), 151–169.

31. Lin, X., and Zhang, Y. "A new graphical method of vertical partitioning in database design". In Proceedings of the 4th Australian Database Conference (ADC) (1993), M. P. P. Maria E. Orlowska, Ed., World Scientific, pp. 131–144.
32. Navathe, S. & Ceri,S. & Weiderhold,G. and Dou,J. "Vertical Partitioning Algorithms for Database Design" ACM Transactions on Database Systems, Vol. 9, No. 4, 1984.
33. Navathe, S. & Ra, M. "Vertical Partitioning for Database Design: A Graphical Algorithm". ACM SIGMOD, Portland, Juin 1989.
34. Özsu .M. & Valduriez, P "Principles of Distributed Database Systems", 2nd edition (1 st edition 1991), New Jersey, Prentice-Hall, 1999.
35. P. LYMAN, H. VARIAN, J. DUNN, A. STRYGIN & K. SWEARINGEN, "How Much Information? ", School of Information Management and Systems, University of California, Berkeley, 2000 et 2003 (http ://www2.sims.berkeley.edu/research/projects/how-much-info-2003).
36. Sam S. Lightstone, Toby J. Teorey, Tom Nadeau "Physical Database Design, The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More", Morgan Kaufmann Publishers Mars 2007, **ISBN 13:** 978-0-12-369389-1.
37. Song, S.K. & Gorla, N., "A genetic Algorithm for Vertical Fragmentation and Access Path Selection," The Computer Journal, vol. 45, no. 1, 2000, pp 81-93.
38. Son, J. H., and Kim, M. H. "An adaptable vertical partitioning method in distributed systems". Journal of Systems and Software 73, 3 (2004), 551–561.
39. Valentin, G., Zuliani, M., Zilio, D., Lohman, G. and Skelley, A. DB2 Advisor: "An Optimizer Smart Enough to Recommend its Own Indexes". In Proceedings of ICDE, San Diego, USA, 2000.
40. Yao, S. B. (1977). "Approximating block access in data-base organization. Communications of the ACM", 20(4), 260-261.