MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière : Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : **Systèmes d'Information Géographique**

THEME :

L'optimisation des requêtes dans l'entrepôt de donnée en utilisant la fragmentation horizontale

Etudiantes : « **benmati Abderrahmane zahir**»

« **bendani Mohammed al amine** »

Encadrante : « **Betouati Fatiha** »

Année Universitaire 2016/2017

Résumé

La conception d'un entrepôt de données parallèle consiste à choisir l'architecture matérielle, à fragmenter le schéma d'entrepôt de données, à allouer les fragments générés, à répliquer les fragments pour assurer une haute performance du système et à définir la stratégie de traitement.

La fragmentation de données est une des techniques d'optimisation utilisée dans la conception physique des entrepôts de données, elle permet d'accélérer l'exécution des requêtes et de faciliter la gestion des données de l'entrepôt. La meilleure manière de fragmenter un entrepôt de données relationnel consiste d'abord à décomposer les tables de dimension ensuite à utiliser des schémas de fragmentation pour partitionner la table de faits. L'espace de recherche pour sélectionner le schéma de fragmentation optimal peut être très important. Nous proposons de formaliser d'abord le problème de sélection d'un schéma de fragmentation pour un entrepôt de données relationnel comme problème d'optimisation avec une contrainte de maintenance.

Mots-clés

Entrepôt de données, Fragmentation, les tables de dimension, la table de faits Schéma optimal.

Abstract

The design of a parallel data warehouse consists of choosing the hardware architecture, fragmenting the data warehouse schema, allocating the generated fragments, replicating the fragments to ensure high system performance and to define the processing strategy.

The fragmentation of data is one of the optimization techniques used in the physical design of data warehouses, it helps accelerate the execution of requests and facilitate management of data warehouse. The best way to fragment a relational data warehouse is first to break down tables dimension then use patterns of fragmentation to partition the table of facts. The space research to select the optimal pattern of fragmentation can be very important.

We propose to formalize the first problem of selecting a pattern of fragmentation for a relational data warehouse as optimization problem with constraint maintenance.

Key words

Data warehouse, Fragmentation, optimal Diagram, tables dimension, table of facts.

Dédicaces

Je dédie ce travail

A ceux qui sont dans mon cœur, qui ont veillés pour notre

Confort et sacrifié beaucoup pour notre réussite,

Ma chère mère (que dieu me la garde)

A celui qui m'a toujours apprit comment réfléchir

Avant d'agir, à

Celui qui m'a soutenu tout au long de ma vie scolaire, à celui qui

N'a jamais épargné un effort pour mon bien,

Mon cher père (Que dieu me le garde)

A mes chers frères, mes chères sœurs, toutes ma familles, mes chères amis

A tous les membres de la promotion

Master informatique de l'université ABDELHAMID IBN BADISS de MOSTAGANEM.

Et à tous ceux qui me connaissent

Remerciements

*Je remercie **Dieu tout Puissant** de m'avoir permis de mener à terme ce projet qui est pour nous le point de départ d'une merveilleuse aventure, celle de la recherche, source de remise en cause permanent et de perfectionnement perpétuelle. Qu'il me soit permis de rendre un vibrant hommage à notre encadrant, Madame **Betouati Fatiha** pour avoir bien voulu superviser ce modeste travail et donné de son temps et de son intelligence à la réussite de ce projet qui pour moi représente un modèle de réussite et une source de motivation permanente, pour sa disponibilité, et son sens aigu de l'humanisme pédagogique.*

Je profite de cette tribune pour remercier les personnes qui de passage, ont pu m'apporter leur contribution, que ce soit au niveau des idées qu'à celui des conceptions. Qu'elles trouvent ici l'expression de ma sincère reconnaissance.

Enfin je remercie les membres du jury qui ont bien voulu accepter, et ce nonobstant, leur lourdes et exaltantes responsabilités pour procéder à l'évaluation de ce modeste

Résumé
Introduction général

Sommaire

Chapitre I : Les entrepôts de données relationnels

Chapitre II : Optimisation par la fragmentation horizontale

Chapitre III : Conception du schéma de fragmentation

Liste des figures

Liste des tableaux

Liste des abréviations

ED: Entrepôt de Données.

ETL: Extract, Transform, Load.

MOLAP: Multidimensional On-Line Analytical Processing.

OLAP: On-Line Analytical Processing.

ROLAP: Relational On-Line Analytical Processing

SGBD : Système de Gestion de la Base de Données.

W : Nombre de fragments que l'administrateur souhaite.

F&A : sélection conjointe de la fragmentation et d'allocation.

F&A&R: Sélection conjointe de la fragmentation et d'Allocation et la Réplication.

AG: Algorithmes Génétiques.

Introduction général

Le « Data Warehouse » ou « Entrepôt de données » en français, est une vision centralisée et universelle de toutes les informations de l'entreprise, qui regroupe les données de cette entreprise pour des fins analytiques et pour aider à la décision stratégique et la décision stratégique étant une action entreprise par les décideurs de l'entreprise et qui vise à améliorer, quantitativement ou qualitativement, la performance de l'entreprise. En gros, c'est un gigantesque tas d'informations épurées, organisées, historiées et provenant de plusieurs sources de données

Les performances d'un Data Warehouse consistent en l'orchestration de toutes ces composantes en un temps record. En effet, qui dit Data Warehouse dit manipulation de données colossales. L'attente des usagers face à l'interrogation de toutes ces données est avant tout une réponse fiable mais aussi très rapide ou du moins dans un délai acceptable.

Les entrepôts de données sont très souvent modélisés par un schéma en étoile. Ce schéma est caractérisé par une table de faits de très grande taille liée à un ensemble de tables de dimension de plus petite taille. La table des faits contient les clés étrangères des tables de dimension ainsi qu'un ensemble de mesures collectées durant l'activité de l'organisation. Les tables de dimension contiennent des données qualitatives qui représentent des axes sur lesquels les mesures ont été collectées. Les requêtes définies sur un schéma en étoile (connues par requêtes de jointure en étoile) sont caractérisées par des opérations de sélection sur les tables de dimension, suivies de jointures avec la table des faits. Aucune jointure n'existe entre les tables de dimension. Toute jointure doit passer par la table des faits, ce qui rend le coût d'exécution de ces requêtes très important. Sans technique d'optimisation, leurs exécutions peuvent prendre des heures, voire des jours.

Pour optimiser ces requêtes, l'administrateur est amené à effectuer une tache importante: la conception physique. Durant cette phase l'administrateur choisit un ensemble de techniques d'optimisation à sélectionner. Ces techniques appartiennent à deux catégories : redondantes comme les index et les vues matérialisées et non redondantes comme la fragmentation horizontale et le traitement parallèle. Dans le cadre de ce projet, nous nous intéressons à une technique d'optimisation non redondante (la fragmentation horizontale).

La fragmentation est une technique de conception logique introduite dans les bases de données réparties. La fragmentation consiste à partitionner une table horizontalement ou verticalement de façon à réduire le nombre des accès nécessaires pour le traitement de certaines requêtes. Dans notre étude, nous nous intéressons à la fragmentation horizontale qui semble être une réponse au problème de réduction du temps d'exécution des requêtes décisionnelles. En effet, elle a été introduite dans les bases de données réparties dans le but de minimiser le nombre d'entrées-sorties (ou le coût de transfert de données) pendant l'exécution des requêtes.

Notre travail vise à réviser le problème de sélection de schéma de la fragmentation horizontal optimal pour les entrepôts de données, en prenant en considération les interactions existantes entre les requêtes, afin d'optimiser les performances des requêtes

Notre projet est organisé comme suit :

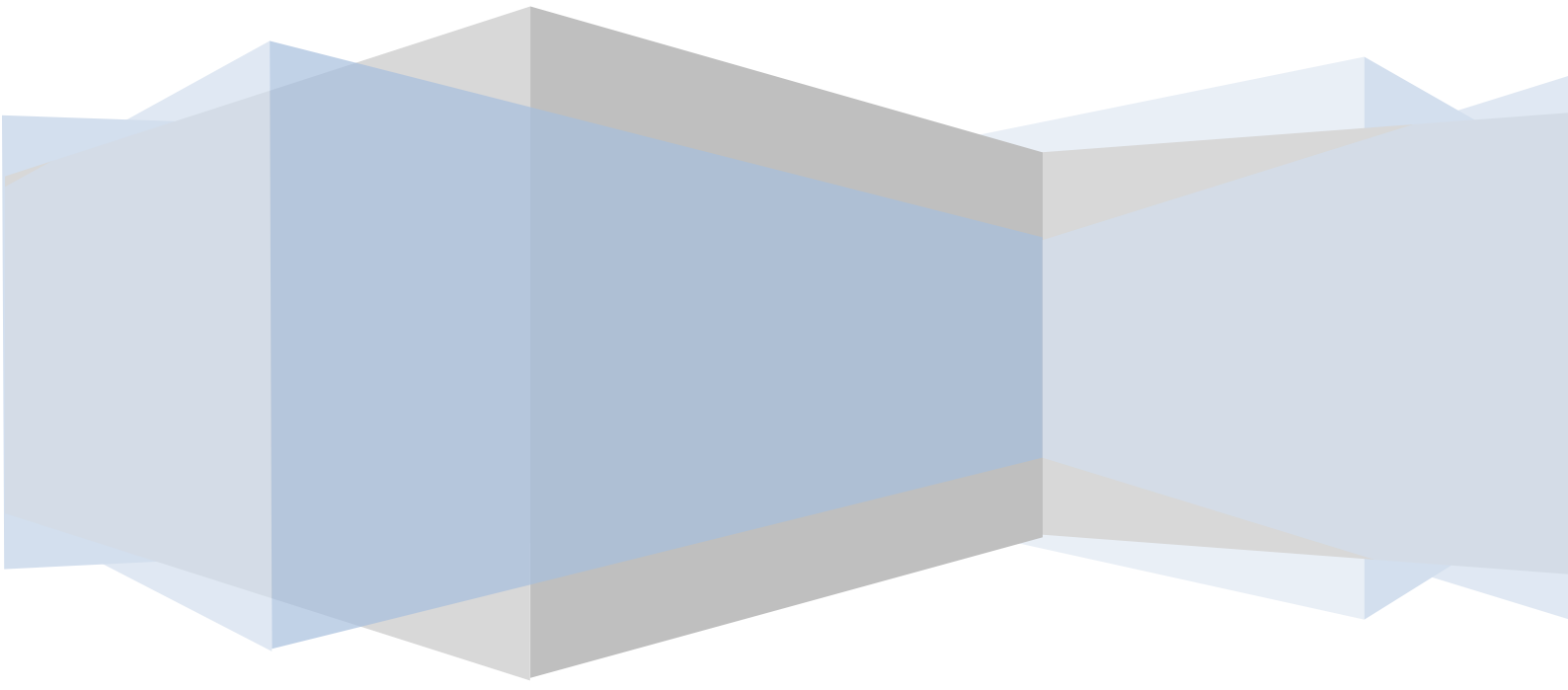
Dans le premier chapitre nous donnons premièrement une définition générale du l'entrepôt de données et cycle de vie de la conception de cet entrepôt de données qui contient trois phases : 1) planification, 2) conception-implémentation, 3) maintenance-évolution , on se basant sur la deuxième phase, nous mettons l'accent sur la classification des différentes techniques d'optimisation, à savoir les vues matérialisées, les index et la fragmentation et on exposant les problèmes liées a chacun une des techniques.

Dans le deuxième chapitre, nous exposerons la fragmentation avec ces deux types (horizontale, verticale), aussi les méthodologies de la fragmentation horizontale dans les entrepôts de données, et nous détaillons l'ensemble des travaux existants sur la résolution de ce problème

Le troisième chapitre sera consacré à la présentation de notre approche qui permet de résoudre le problème suivant : l'optimisation de requête dans l'entrepôt de donnée parallèle en utilisant la fragmentation horizontal

Chapitre I

Etat de l'art



I. Introduction

Un entrepôt de données ou *Data Warehouse*, est considéré beaucoup plus comme un environnement que comme une application ou une base de données. Plusieurs composantes définissent et caractérisant un entrepôt de donné : les systèmes sources, les différentes connexions, le réseau, les traitements d'extraction - chargement, le calcul des agrégations, l'interrogation des données comme nous allons détailler en cour de ce chapitre, et nous donnons premièrement les concepts de base et cycle de vie de la conception de cet entrepôt de données.

II. L'entrepôt de données

Un entrepôt de données d'après Bill Inmon est une collection de données orientées sujet, non volatiles, intégrées, historiées et utilisées pour supporter un processus d'aide à la décision. Les entrepôts de données sont utilisés dans les systèmes décisionnels. Ils sont le cœur de l'exploitation de l'information de l'entreprise. L'objectif principal d'un entrepôt de données est de fournir rapidement et de façon fiable les informations utiles à la prise de décision, cela nécessite la reconstitution des informations afin de les rendre plus facilement compréhensibles et utilisables.

II.1. Les caractéristiques des données de l'entrepôt de donnée

Nous détaillons les cinq caractéristiques citées dans de l'entrepôt de données [reff_1] :

- **Orientée sujet :** Organisée par thème afin de permettre la réalisation d'analyses autour des sujets primordiaux et des métiers de l'entreprise. Par cette organisation on peut ainsi passer d'une vision verticale de l'entreprise à une vision transversale beaucoup plus riche en information.

- **Intégrée :** Construit en intégrant des sources de données multiples et hétérogènes (BD relationnelles, fichiers, enregistrements de transactions), nettoyage et intégration des données. Consistances dans les noms des champs, le codage des données issues de plusieurs sources (La conversion se fait quand les donnés sont transférées dans le DW)

- **Non volatile :** Non volatile pour être stable, en lecture seule, non modifiable afin de conserver la traçabilité des informations et des décisions prises

- **Historiées :** Contrairement au système de production les données ne sont jamais mises à jour. Chaque nouvelle donnée est insérée, Un référentiel de temps doit être mis en place afin de pouvoir identifier chaque donnée dans le temps.

- **Organisées :** Les informations issues des sources de données doivent être agrégées et réorganisées afin de faciliter le processus de prise de décision.

II.2. Architecture d'un entrepôt

L'intégration des sources de données hétérogènes ayant des structures différentes est le principal objectif de la naissance des entrepôts de données. Cependant, l'entrepôt de données joue un rôle stratégique dans la vie d'une entreprise. Il stocke des données pertinentes aux besoins de prise de décision en provenance des systèmes opérationnels de l'entreprise et d'autres sources externes [reff_10].

Un entrepôt de données est généralement construit selon une architecture en trois parties :

1) Extraction des données à partir de différentes sources.
2) Organisation et intégration des données dans l'entrepôt.
3) Accès aux données intégrées et analyse de ces dernières dans une forme efficace et flexible (voir figure 1)

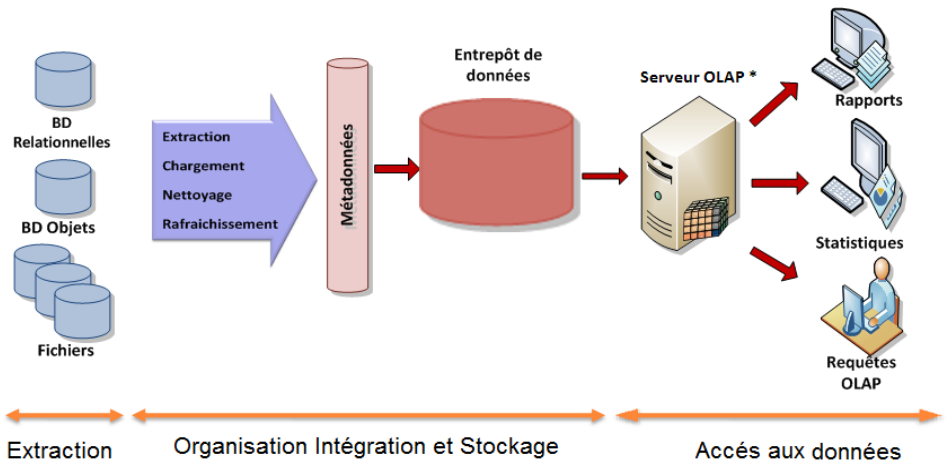


Figure 1 - l'architecture conceptuelle d'un entrepôt de données [reff_2]

Dans la première et la deuxième phase, les données issues de différentes sources de données sont extraites, nettoyées et intégrées dans l'entrepôt de données. Durant la troisième phase, un serveur OLAP se charge de présenter les informations demandées par les utilisateurs sous plusieurs formes : tableaux, rapports, statistiques.

III. Cycle de vie de conception d'un entrepôt de données :

Le cycle de vie de conception d'un entrepôt de données regroupe les phases suivantes: La planification, la conception et l'implémentation, la maintenance et la gestion de l'évolution et le test [reff_1].

- **La planification :** Cette phase permet de préparer le terrain pour le développement des l'entrepôt de données. Elle consiste à déterminer les objectifs de l'entrepôt à développer et identification des futurs utilisateurs de l'entrepôt.

- **La conception et l'implémentation :** Cette phase consiste à développer le schéma de l'entrepôt et à mettre en place toutes les ressources nécessaires à son implémentation et à son déploiement.

- **La maintenance et la gestion de l'évolution :** Cette phase implique l'optimisation de ses performances périodiquement. L'évolution de l'ED concerne la mise à jour de son schéma en fonction des différents changements survenant au niveau des sources ou des besoins des utilisateurs.

- **Les tests :** Le test dans l'entrepôt de données est nécessaire, il permet de tester l'ensemble des phases de conception de l'entrepôt de données en termes de performance et de sécurité.

Le schéma illustré par la figure 2, représente la succession des tâches nécessaires à la mise en place des entrepôts de données efficaces. Chaque rectangle indique une phase.

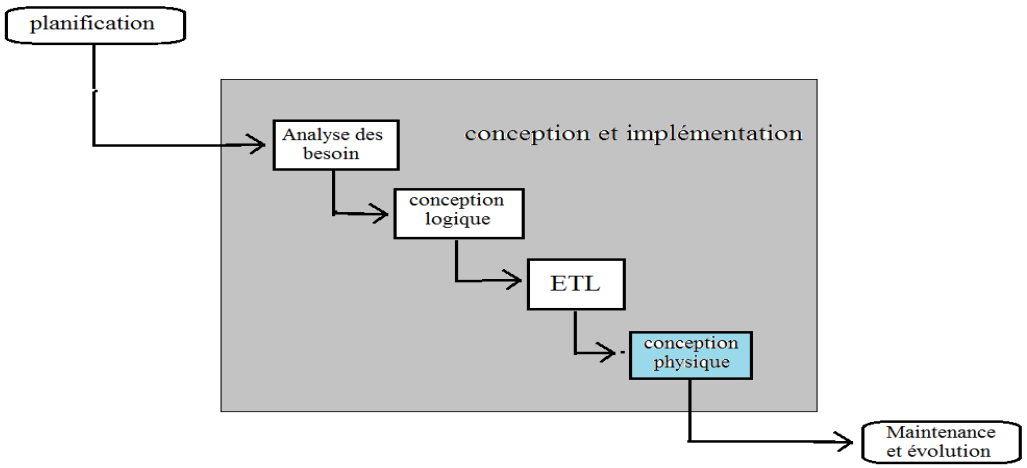


Figure 2 - Cycle de vie des entrepôts de donnée

Parmi les quatre étapes de la phase conception à savoir : analyse des besoins, conception logique, alimentation(ETL) et conception logique, nous nous intéressons à la conception physique, dans la section suivante nous détaillons ces différentes étapes.

III.1. Analyse des besoins :

La première étape de la conception consiste à analyser la situation pour tenir compte des contraintes, et tous les éléments pertinents et assurer un ouvrage ou un processus répondant aux besoins du client. Il existe des besoins fonctionnels et des besoins non fonctionnels. Les besoins fonctionnels sont ceux qui caractérisent le système (langage de programmation, type SGBD, système d'exploitation, ... etc.). Par contre, les besoins non fonctionnelles sont des besoins implicites auxquelles le système doit répondre. Citons la réutilisabilité, la fiabilité, ...etc.

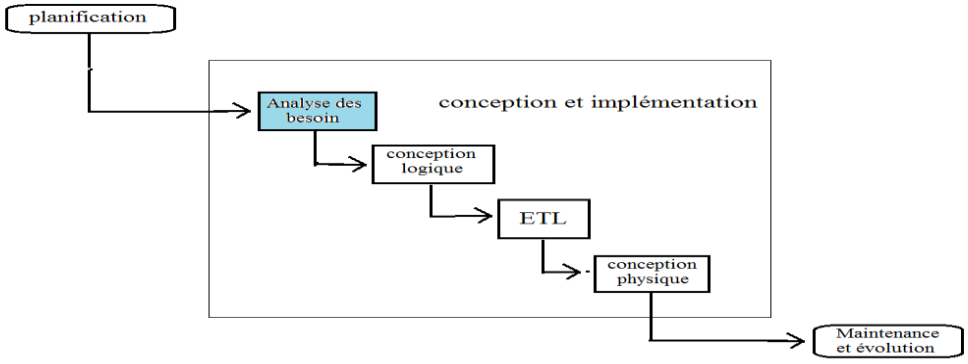


Figure 3- la phase analyse des besoins

III.2. Conception logique

Le but de la conception logique d'un entrepôt de données est de classer et historiser les données pour facilité leurs exploitations afin de faire des analyse et de prise de décisions nécessaires pour l'entreprise, pour cela le modèle le plus adaptée au processus d'analyse est la représentation multidimensionnel, Ce modèle est constitué de deux types de tables (la dimension et les faits)[reff_1]

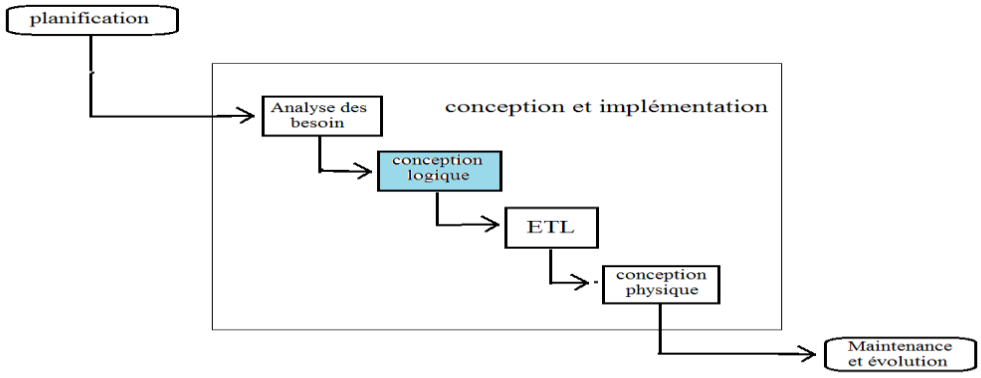


Figure 4- la phase conception logique

III.2.1. Les modèles multidimensionnels

L'objectif majeur de cette modélisation est la vision multidimensionnelle des données, ce qui est assuré via le concept de cube de données. Ce dernier organise les données en une ou plusieurs dimensions qui déterminent une mesure d'intérêt. Un entrepôt de données peut être modéliser de deux manières différentes : le modèle multidimensionnel MOLAP où un tableau multidimensionnel à n dimensions est utilisé pour représenter les données, et le modèle relationnel ROLAP qui offre plusieurs types de schémas (en étoile, en flocons, en constellation) plus adaptés aux besoins multidimensionnels[reff_1]

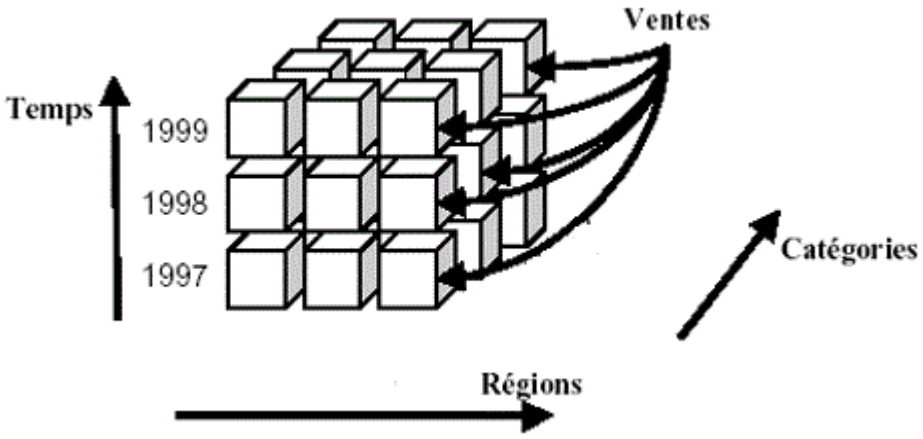


Figure 5- Le modèle multidimensionnel de données

III.2.2. Concepts dimensions et faits

– **Les tables de dimension** : c'est un ensemble de tables contenant les axes à considérer dans la phase d'analyse. Par exemple, les dimensions Temps, Produit et Client
– **La table des faits** : c'est une table contenant les différentes mesures requises dans la phase d'analyse. Par exemple, une table des faits Ventes comporte plusieurs mesures comme Quantité, Montant ou encore Coût unitaire

III.2.3. Les systèmes de type MOLAP

Les systèmes MOLAP (Multidimensional On Line Analytical Processing) stockent les données dans un SGBD multidimensionnel sous la forme d'un tableau multidimensionnel où chaque dimension est associée à une dimension du cube. L'intérêt de cette approche est l'optimisation du temps d'accès, mais on trouve des difficultés dans ce system tel que la mis a jour et la consommation de l'espace [reff_3].

III.2.4. Les systèmes de type ROLAP

Les systèmes de type ROLAP (Relational On Line Analytical Processing) utilisent une représentation relationnelle du cube de données, sous une forme d'une table de fais et des tables de dimensions. La table de faits possède comme attributs les mesures d'activités et les clés étrangères vers les tables de dimension. Ce modèle a pour d'avantage l'utilisation des systèmes de gestion de bases de données existants, ce qui réduit le coût de mise en œuvre. Plusieurs modèles ont été proposés pour la modélisation des systèmes ROLAP, les plus utilises sont : le schéma en étoile, le schéma en flocon de neige.
Nous représentons deux schémas principaux pour modéliser les systèmes **ROLAP** :

III.2.4.1. Modèle en étoile :

Une étoile est une façon de mettre en relation les dimensions et les faits dans un entrepôt de données. Il contient une table des faits normalisée au centre du schéma et des tables de dimension qui sont généralement dé-normalisées au tour du table fait et qui sont directement reliées à un fait par des clés (schématiquement, ça fait comme une étoile), afin de minimiser le nombre de jointures nécessaires pour évaluer une requête.

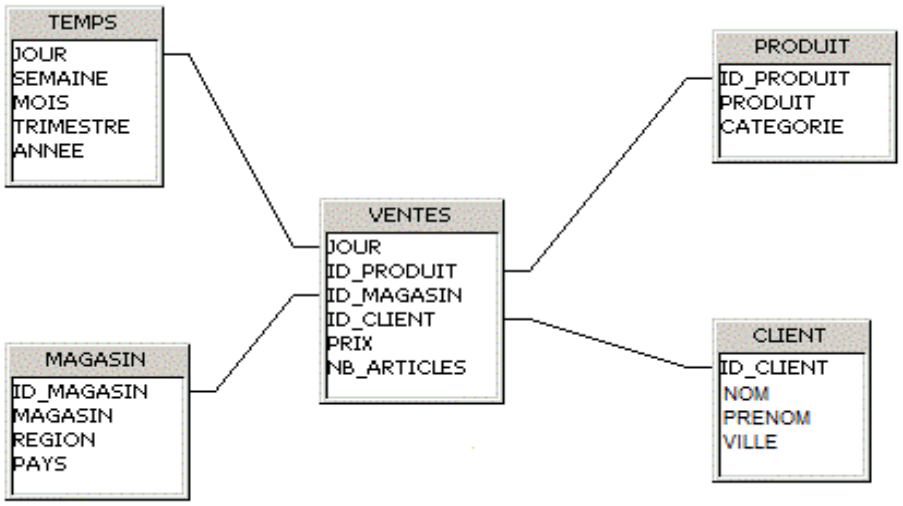


Figure 6- exemple d'un schéma en étoile

III.2.4.2. Modèle en flocons :

Un autre modèle de mise en relation des dimensions et des faits dans un entrepôt de données. Le principe étant qu'il peut exister des hiérarchies de dimensions et qu'elles sont reliées aux faits, ça fait comme un flocon. Ce modèle est une évolution du schéma en étoile avec la décomposition des dimensions du modèle en étoile en sous hiérarchies le plus bas est reliée à la table de fait. On dit qu'elle a la granularité la plus fine

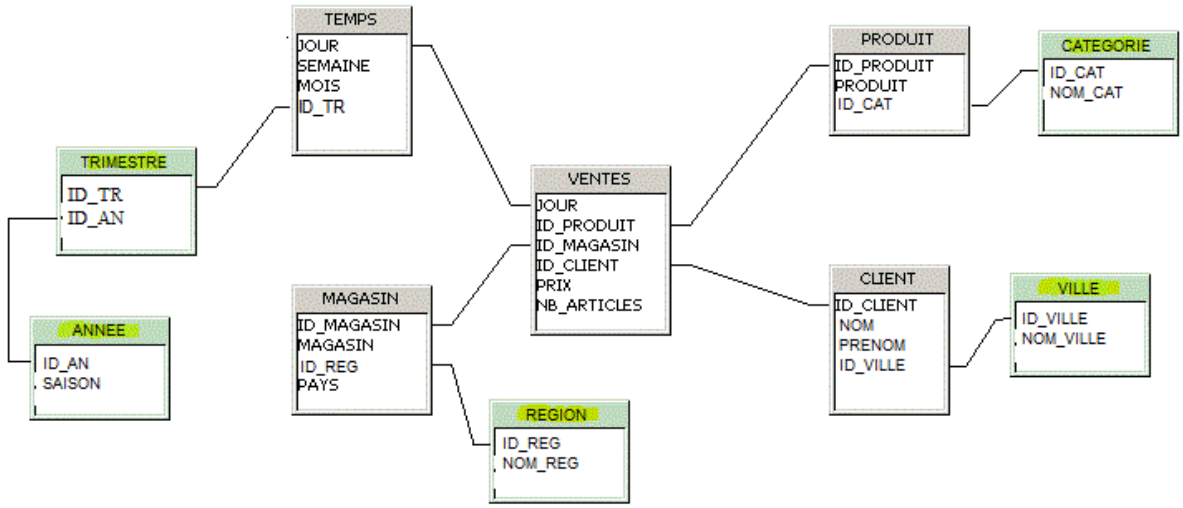


Figure 7- exemple d'un schéma en flocons

Il est à noter que la fusion de plusieurs schémas en étoile, qui ont des tables de dimension, donne lieu à un *schéma en constellation*. Ainsi, le schéma contient plusieurs tables de faits partagent des tables de dimensions. Peut-être vu comme une collection d'étoiles (schéma en galaxie ou constellation de faits).

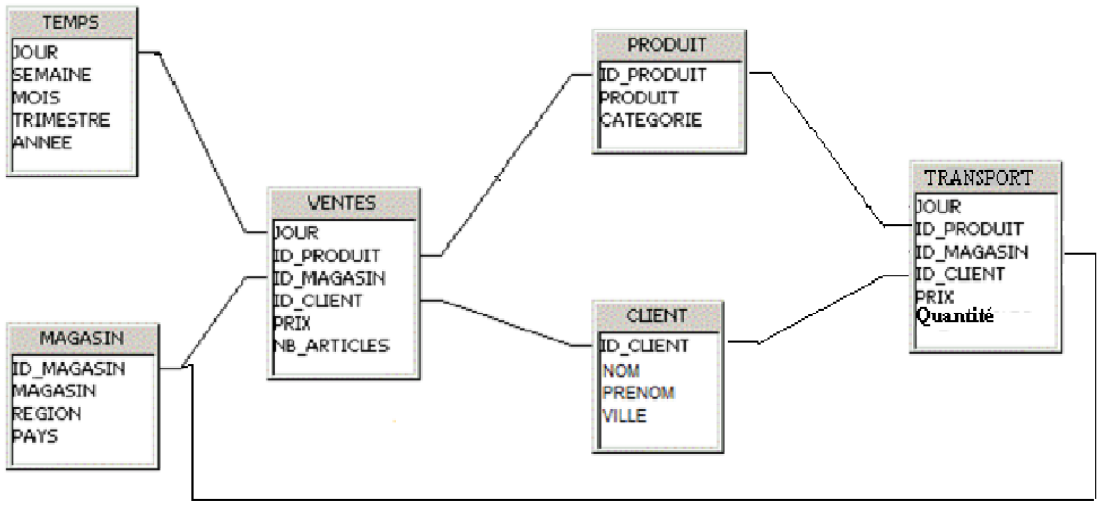


Figure 8- exemple d'un schéma en constellation de faits

III.3. La phase ETL (Alimentation d'entrepôt des données)

L'extraction de données : L'extraction est souvent effectuée à l'aide d'un outil d'ETC (Extraction, Transformation, chargement). Elle consiste à aller chercher les données la où elles se situent, à les trier, et à les transformer éventuellement afin d'effectuer un prétraitement pour faciliter l'analyse.

Nettoyage, Transformation et Chargement :

- ***Objectifs du nettoyage*** : résoudre le problème de consistance des données au sein de chaque source.
- ***Objectifs de la transformation*** : Suppression des incohérences sémantiques entre les sources pouvant survenir lors de l'intégration des schémas et des données.
- ***Objectif du chargement*** : charger les données nettoyées et préparées dans l'ED.

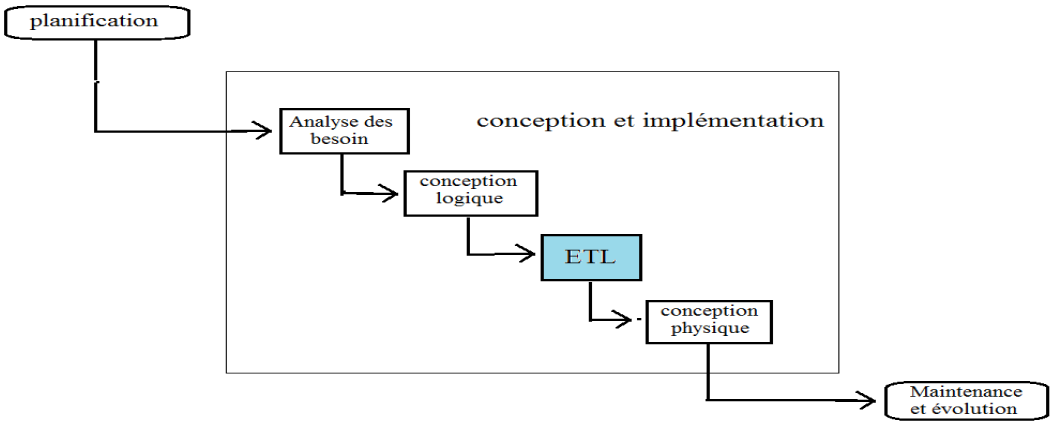


Figure 9- la phase ETL

III.4. Conception physique de l'entrepôt de données

La conception physique d'un entrepôt de données consiste à établir une configuration physique sur le support de stockage. Aujourd'hui face à la complexité des requêtes à traiter et le volume important des données, la conception physique a reçu une importance phénoménale Cela comprend la spécification détaillée des éléments de données, les types de données et la sélection des techniques d'optimisation. Cette dernière est au cœur de la conception physique, plusieurs techniques d'optimisation ont été proposées dans la littérature. Nous pouvons les classer en deux catégories principales : techniques redondantes et techniques non redondantes [reff_9].

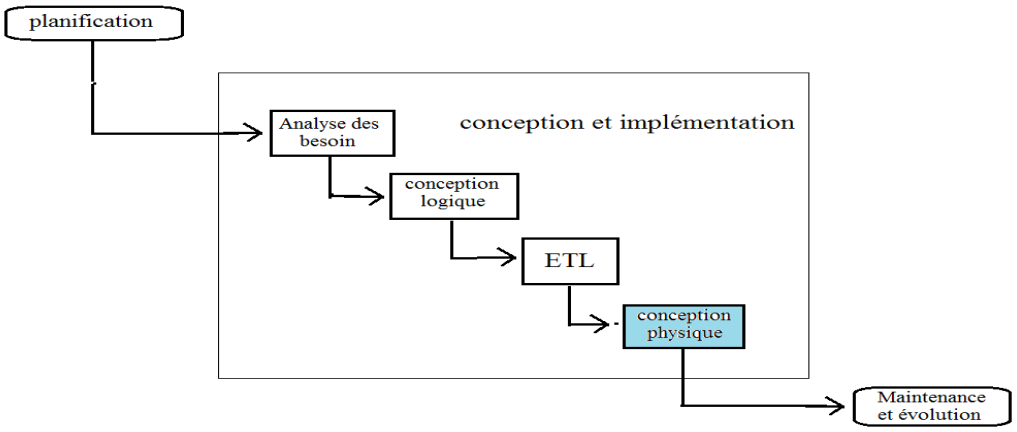


Figure 10- la phase conception physique

III.4.1. Les techniques d'optimisation

Nous présentons quatre solutions de technique d'optimisation qui sont classifié selon deux type (redondants, non redondants).

III.4.2. Classification des techniques d'optimisation

Les structures d'optimisation existantes sont classées selon la redondance. Ainsi, si une structure ne duplique pas les données et n'engendre pas un coût de stockage ou de maintenance supplémentaire, alors la structure est dite non redondante. Sinon, elle est redondante, la figure 11 montre une classification des principales techniques d'optimisation

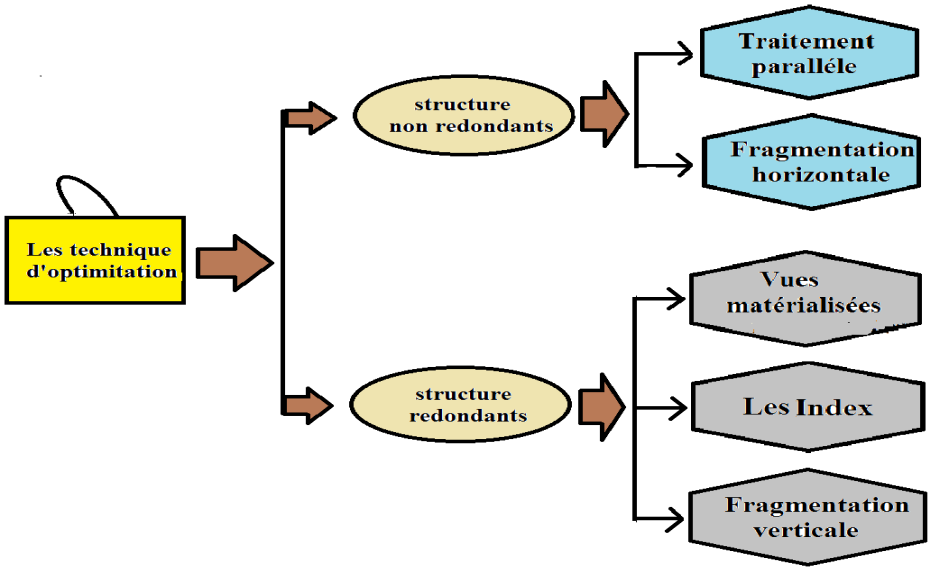


Figure 11 - classification des techniques d'optimisation

III.4.2.1 Techniques d'optimisation redondantes :

Les vues matérialisées :

Les vues matérialisées peuvent être utilisées pour satisfaire plusieurs objectifs, comme l'amélioration de la performance des requêtes ou la fourniture des données dupliquées.une vue matérialisée est une entité utilisée dans un SGBD de type relationnel. Une vue matérialisée est une table contenant les résultats d'une requête. Les vues améliorent l'exécution des requêtes en pré calculant les opérations les plus couteuses comme la jointure et l'agrégation, et en stockant leurs résultats dans la base. En conséquence, certaines requêtes nécessitent seulement l'accès aux vues matérialisées et sont ainsi exécutées plus rapidement. Les vues dans le contexte OLTP ont été largement utilisées pour répondre a` plusieurs rôles : la sécurité, la confidentialité, l'intégrité référentielle [reff_3].

Les Index :

Un index est une structure redondante ajoutée à la base de données pour permettre les accès rapides aux données. Il permet à partir d'une clé d'index de trouver l'emplacement physique des n-uplets recherchés. Parmi les techniques d'indexation proposées dans le cadre des bases de données classiques, nous pouvons citer l'index B-tree, l'index de projection, l'index de jointure. La majorité de ces index est aussi utilisée dans le cadre des entrepôts relationnels. Certaines techniques d'indexation sont apparues dans le contexte d'entrepôts de données comme les index binaires, les index de jointure binaires, les index de jointure en étoile, etc. Nous pouvons classer les index proposés en deux catégories, les index mono-table qui sont des index définis sur un ou plusieurs attributs de la même table, comme les index B-arbre, de hachage, binaires et les index multi-tables sont des index définis sur plusieurs tables comme les index de jointure standards, en étoile et binaires [reff_3].

Les fragmentations verticales :

La fragmentation verticale est obtenue par décomposition de la table en groupes de colonnes, elle consiste à diviser une relation R en sous relations appelées fragments verticaux résultant de l'application de l'opération de projection, elle favorise naturellement le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de la fragmentation et la reconstruction se fait par des jointures [reff_3].

III.4.2.2 Techniques d'optimisation non redondantes :

Les fragmentations horizontales :

La fragmentation est le processus de décomposition d'une base de données logique en un ensemble de "sous" bases de données. Cette décomposition doit être sans perte d'information. La fragmentation peut être coûteuse s'il existe des applications qui possèdent des besoins opposés. Elle a été introduite à la fin des années 70 et au début des années 80.La

fragmentation consiste en la division en plusieurs fragments (sous-ensembles de la relation) qui peuvent être non disjoints, c'est une technique qui permettant l'optimisation de performances des requêtes et d'éviter le balayage de grandes tables nous nous intéressons à cette technique [reff_4].

III.4.3 Modes de sélection des structures d'optimisation

L'optimisation se fait par une ou plusieurs structures d'optimisation |SO||, nous distinguons deux modes de sélection des structures d'optimisation [reff_5] :

– **Mode isolé** : où l'administrateur sélectionne une seule structure d'optimisation pour satisfaire sa charge de requêtes Q. Plus formellement : ||SO|| = 1

– Mode multiple : où l'administrateur sélectionne plusieurs structures d'optimisation à la fois pour satisfaire sa charge. Plus formellement : ||SO|| > 1

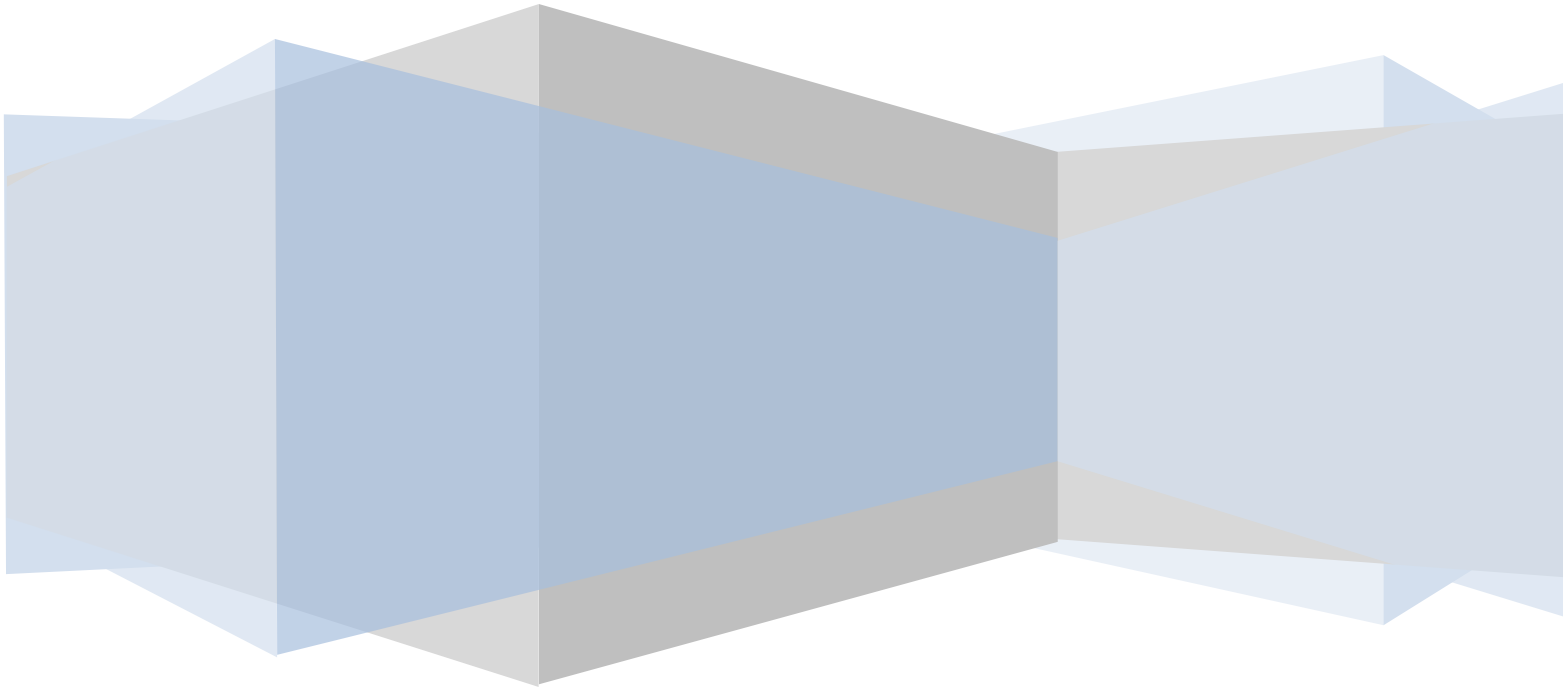
IV. Conclusion

Nous avons présente dans ce chapitre une présentation des entrepôts de données, aussi leurs caractéristique et leurs architecture générales. Nous avons aussi vue la conception logique dans le cycle de vie des entrepôts de données avec le modèle multidimensionnel qui peut être implémentée sur deux modèles : MOLAP et ROLAP**.** Deux schémas principaux les plus utilisés pour la modélisation des systèmes ROLAP : le schéma en étoile et le schéma en flocon de neige, nous avons mis l'accent sur la classification des différentes techniques d'optimisation. Ces techniques appartiennent à deux catégories : redondantes comme les index et les vues matérialisées et non redondantes comme la fragmentation horizontale et le traitement parallèle. Dans le cadre de notre projet, nous nous intéressons à une technique d'optimisation non redondante qui est la fragmentation horizontale.

Dans le chapitre suivant, nous détaillerons la technique de fragmentation horizontale, aussi les travaux existant pour cette technique dans les différents contextes parallèle, distribué, centralisé.

Chapitre II

Optimisation par la fragmentation horizontale



Chapitre II : Optimisation par la fragmentation horizontal

I. Introduction

La fragmentation est l'une des techniques d'optimisation des requêtes introduite dans les bases de données réparties, ce dernier consiste à partitionner une table horizontalement ou verticalement de façon à réduire le nombre des accès nécessaires pour le traitement de certaines requêtes. Dans notre étude, nous nous intéressons à la fragmentation horizontale qui semble être une réponse au problème de réduction du temps d'exécution des requêtes décisionnelles, nous proposons d'instancier le mode de sélection isolée en prenant le cas d'optimisation par la fragmentation horizontale, dans un premier temps, nous donnons l'ensemble des définitions et des principes liées à cette technique, suivie par la problématique posée sur cette technique, nous détaillons ensuite les travaux réalisés avec cette technique et on se termine par une synthèse de ces travaux.

II. Fragmentation des entrepôts de données

Plusieurs scénarios sont proposés pour fragmenter un entrepôt de donnée et le meilleur scénario et le plus utilisé pour l'optimisation c'est la fragmentation horizontale, elle consiste à partitionner les tables dimensions suivants une fragmentation primaire sur leurs attributs, ensuite la table de faits est fragmentée par une FH dérivée suivant la FH primaire des tables dimensions. Cela permet de répartir l'entrepôt de données en plusieurs sous schémas en étoiles où chaque sous schéma contient un fragment de la table de faits et tous les fragments des tables dimensions qui ont permet de le définir. Ce scénario de FH permet d'améliorer les opérations de sélections sur les tables dimensions et les opérations de jointures entre la table de faits et les tables dimensions contenues dans les requêtes de jointures en étoile.

II.1 Allocation de données

C'est une technique très importante pour atteindre la haute performance des systèmes parallèles de façon générale, Les objets à allouer peuvent être soit des données (tables, vues, fragments) soit des requêtes. Ces objets doivent d'une manière judicieuse allouer sur les nœuds machines pour atteindre la haute performance d'un système [reff_1].

II.2 Réplication des fragments

La réplication est une technique qui consiste à placer plusieurs copies de l'objet sur les différents nœuds. La réplication permet d'augmenter la fiabilité et la disponibilité des

données. En plus, elle fournit une grande disponibilité des données et garantit la tolérance aux pannes en débit de la défaillance des nœuds [reff_14].

II.3 Equilibrage de charge

C'est une stratégie de traitement des requêtes, elle permet d'allouer la charge de requêtes d'une manière équilibrée sur les nœuds de traitement après la fragmentation, l'allocation et la réplication des données [reff_1].

III. Fragmentation horizontale

La fragmentation horizontale est obtenue par décomposition de la table en groupes de lignes, elle consiste à diviser une relation R en sous ensembles de n-uplets appelés fragments horizontaux, La reconstruction de n-uplets à partir de ces fragments horizontaux est obtenue par l'opération d'union de ces fragments. La fragmentation horizontale permet de réduire la complexité des requêtes décisionnelles exécutées sur un entrepôt de données relationnel et considérée comme une technique d'optimisation non redondante car elle ne duplique pas les données et ne nécessite pas un espace de stockage supplémentaire, Elle est employée dans plusieurs types d'entrepôts de données comme les entrepôts centralisés, distribués et parallèles. Deux types de fragmentation horizontale existent : « la fragmentation horizontale primaire (FHP) » et « la fragmentation horizontale dérivée (FHD) ». Dans la section suivant nous détaillons ces deux dernière et bien sur nous nous intéressons à la « fragmentation horizontale dérivée (FHD) »

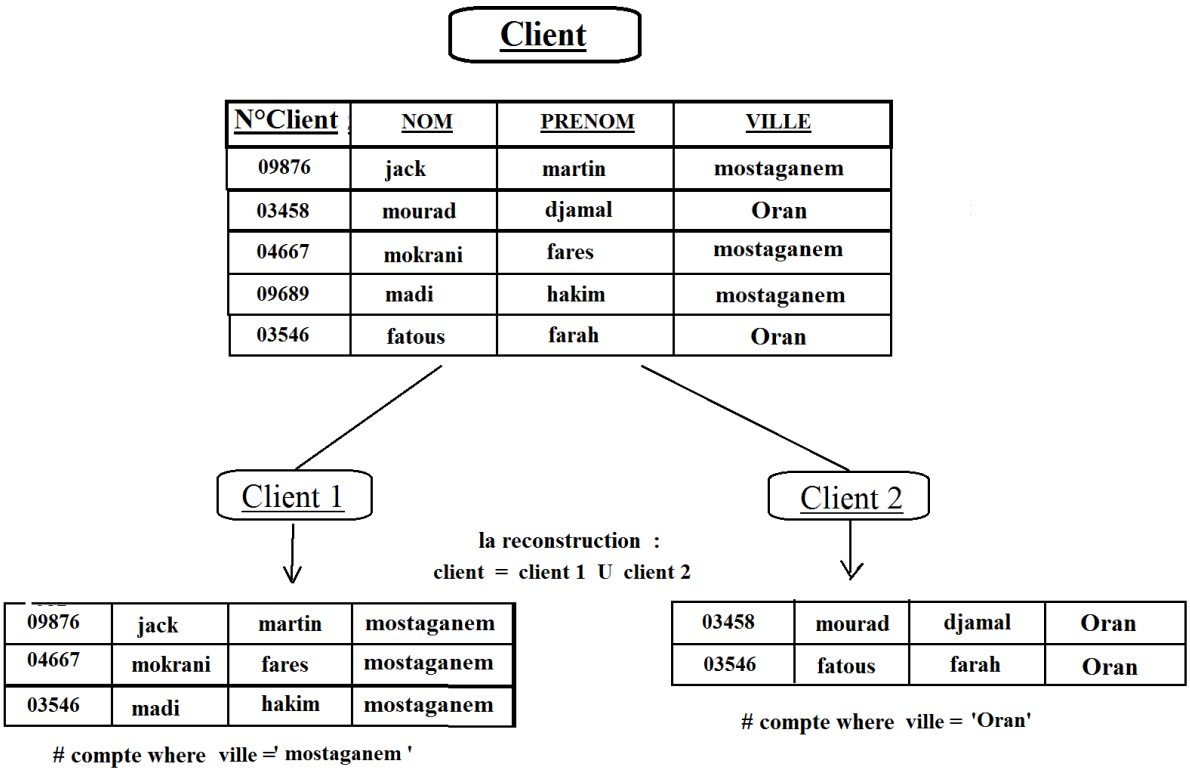


Figure 12 - exemple de fragmentation horizontal

III.1. Les types de fragmentation horizontale

Il existe deux types de fragmentation horizontale dans les entrepôts de données relationnels :

1. Fragmentation Horizontale Primaire (FHP)
2. Fragmentation Horizontale Dérivée (FHD)

III.1.1. La fragmentation horizontale primaire (FHP)

La (FHP) permet de répartir les tuples d'une table suivant la conjonction de prédicats de sélection définis sur les attributs de la même table. Elle favorise les opérations de sélection portées sur les attributs de fragmentation, à pour l'objectif d'accélérer les opérations de sélection (figure 13).

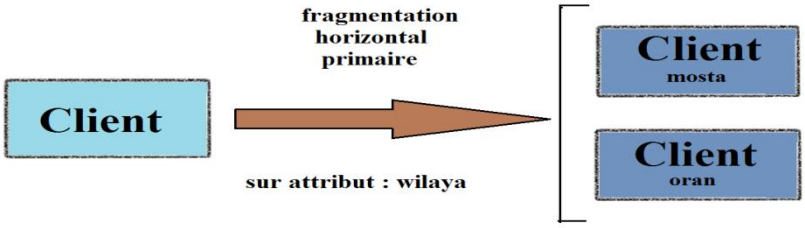


Figure 13 - principe de la fragmentation primaire sur la table de diamantions

Dans ce schéma, la table de dimension Client peut être découpée selon les fragments de la table Client pour obtenir les Client de la wilaya « Mostaganem » et les Client de la wilaya de « Oran » séparément.

III.1.2. La fragmentation horizontale dérivée (FHD)

La (FHD) permet de fragmenter une relation suivant la fragmentation horizontale primaire d'une seconde relation, à condition de l'existence de relation père-fils entre les deux tables c'est à dire (consiste à partitionner cette table selon des prédicats définis sur des données d'une autre table), fragmentation dérivée à un but pour accélérer les opérations de jointure (figure 14).

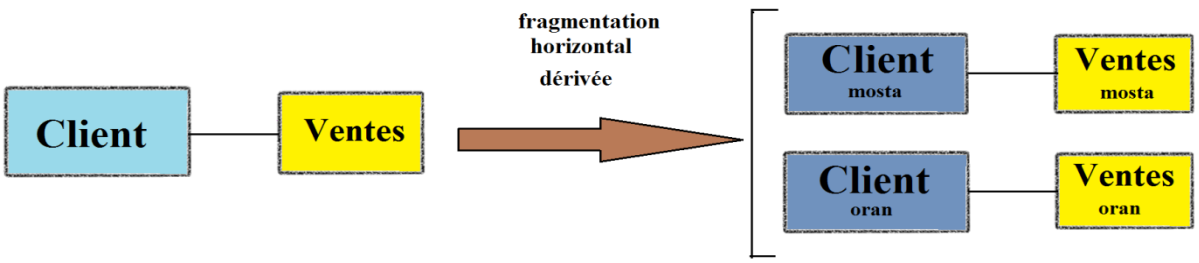


Figure 14 - principe de la fragmentation dérivée sur la table de fait

Dans le schéma de l'exemple précédent, la table des faits Ventes peut être découpée selon les fragments de la table Client pour obtenir les ventes des clients de Mostaganem et les ventes des clients d'Oran séparément.

– Ventes $_{mosta}$= (Ventes><Client $_{mosta}$)

– Ventes $_{oran}$= (Ventes><Client $_{oran}$)

Par conséquent, la recherche des ventes de clients de Mostaganem sera optimisée de la façon suivante :

SELECTCount(*)
FROM **Ventes PARTITION** (Ventes $_{mosta}$)

III.2. La validité des fragments

Pour que tout les fragments soit valide il faut vérifier les trois règle existant : la complétude, la disjonction et la reconstruction.

La complétude : toute un enregistrement dans la table initiale doit être appartenir au moins a un de ses fragments pour ne perde aucun enregistrement.

La disjonction : toute un enregistrement existe dans la table appartenir au plus d'un fragment c'est à dire on ne trouve pas un enregistrement dans deux fragments différent, l'intersection entre les fragments donne toujours l'ensemble vide, dans le but à éviter la redondance.

La reconstruction : c'est-à-dire on peut reconstruire la table initiale à partir de ses fragments, cette opération ça se fait par Union ($C = U_{j=1}^{f} Cf$), tell que C : la table initial et Cf : nombre de fragment.

III.3. Les problèmes de sélection d'un schéma de la fragmentation horizontale

Le problème de sélection d'un schéma de FH des dimensions peut être formaliser comme suit :

1) Un entrepôt de données modélisé par un schéma en étoile ayant des tables de dimension D = {D1, D2, ..., Dd} et une table des faits F.
2) une charge de requêtes Q = {Q1, Q2, ..., QT}
3) un seuil W représentant le nombre maximum de fragments de la table faits dans le schéma final de FH.

Le problème consiste à générer un schéma de FH tel que :

- Une fois la table de faits fragmentée par une FHD suivant le schéma de FHP des tables dimensions, le nombre de fragments faits ne dépasse pas le seuil W.
- Le coût total d'exécution des requêtes sur le schéma fragmenté est réduit, la figure 8 illustre ce problème

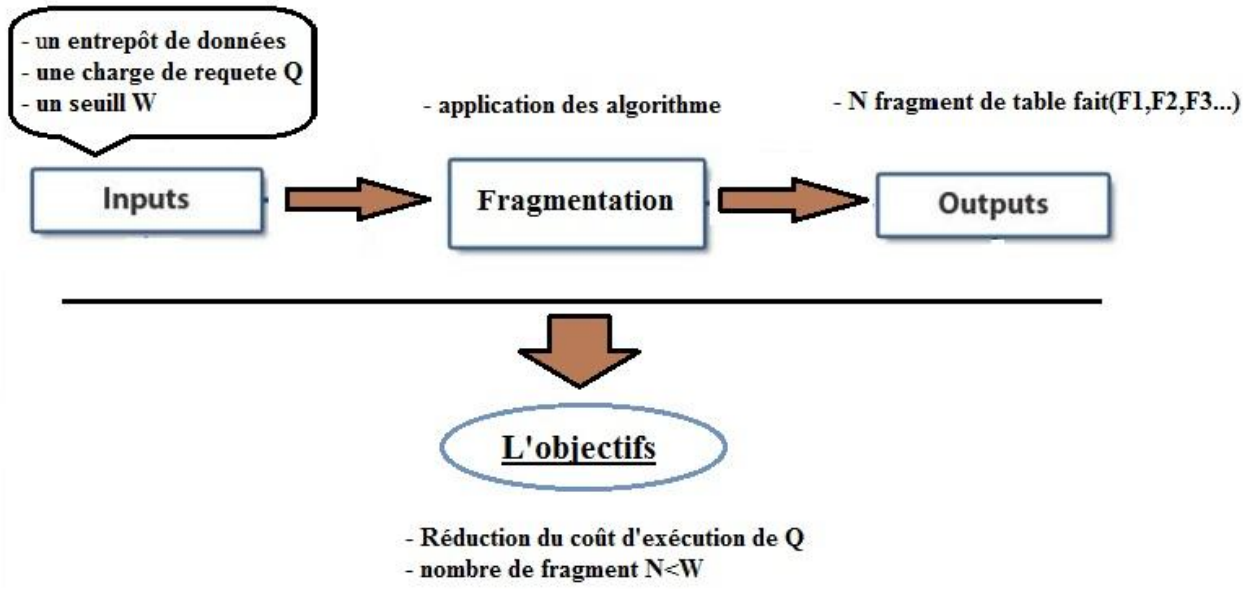


Figure 15 - illustrations de la problématique posée dans notre travaille

III.5. Travaux existants

Il existe plusieurs travaux pour la résolution de problème de la fragmentation horizontale d'entrepôt de donnée dans les différents contextes parallèle, distribué, centralisé nous citons quelques-uns ici.

III.5.1 Travaux existants dans le cotexte centralisé :

« Travaux de amira kerkad »

Dans ces travaux [reff_6], les auteurs proposent une nouvelle approche pour la FH dans les EDR basée sur l'interaction entre les requêtes. Cette interaction est intégrée dans une structure de données incrémentale. Tous d'abord, il a commençait par une représentation graphique de la charge des requêtes introduit par utilisateur qui s'appelle le MVPP (Multiple

View Processing Plan), tel que il a poussé les opérations de sélection en bas de schéma du MVPP avec application de l'algorithme Rule-Based, (voir la figure 9).

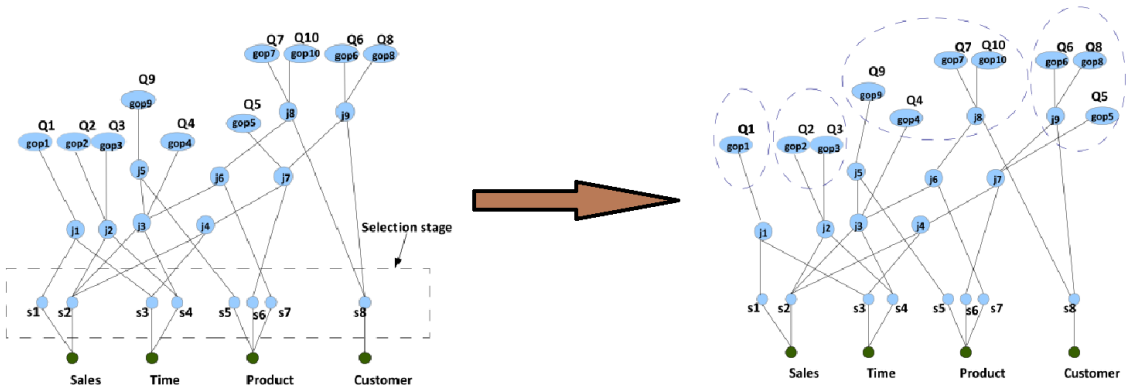


Figure 16- Exemple de MVPP avec exploitation de l'interaction [reff_6]

Après la construction du MVPP, les auteurs proposent de regrouper les requêtes obtenues selon les jointures communes (dans cet exemple (figure 10)) on obtient quatre groupe : {Q1}; {Q2; Q3}; {Q4; Q7; Q9; Q10}; {Q5; Q6; Q8} a partir de ces jointures communes : { j1; j2; j3; j4},qui sont les jointure les plus bas du graphe et les plus coûteuse pour l'exécution des requêtes, sachant que tous les requêtes dans le même groupe corrélèrent au moins dans la première jointure, ensuite elle propose un autre algorithme pour ressortir l'interaction entre les requêtes est l'Algorithme de Requête Elue (ARE) c'est une approche dite diviser pour régner, le principe de ARE est que dans chaque groupe, l'algorithme élit une requête qui permettra d'aiguiller le processus de FH. Cette requête, que l'on appelle Requête Elue (RE) est considérée comme la plus importante dans son groupe. Le choix de la requête élue du groupe est effectué selon le critère du coût d'exécution. Cet à dire que dans chaque groupe la requête ayant le coût minimal est élue. Généralement, la requête ayant un coût minimal contient moins d'opérations de jointure et de sélections

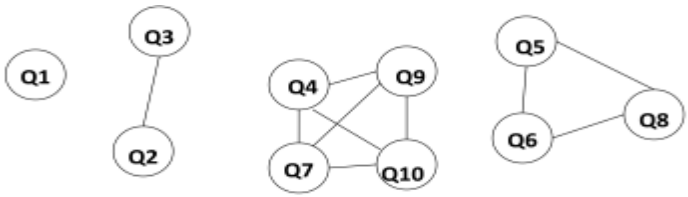


Figure 17 - Groupes de requêtes obtenus à partir du MVPP [reff_6]

Dans étape suivant, il propose un codage incrémental qui fait la décomposition d'un domaine d'attribut en sous-domaines à partir du MVPP, pour cette proposition il a utilisé deux fonction : un split horizontal ou bien un split verticale, le principe de ce codage est de

commencer par un ensemble d'attributs vide, et pour chaque requête y ajouter ses attributs de sélection en créant de nouveaux vecteurs, chacun contenant un seul champ. A chaque fois qu'une sélection est retrouvée dans la requête en cours, l'une des deux opérations est effectuée :

1. Si l'attribut n'existe pas dans le schéma, appliquer un Split Vertical et un nouvel vecteur pour l'attribut à ajouter.
2. Si l'attribut existe déjà, appliquer le Split Horizontal sur le champ else pour rajouter les nouveaux sous-domaines (voir la figure 11).

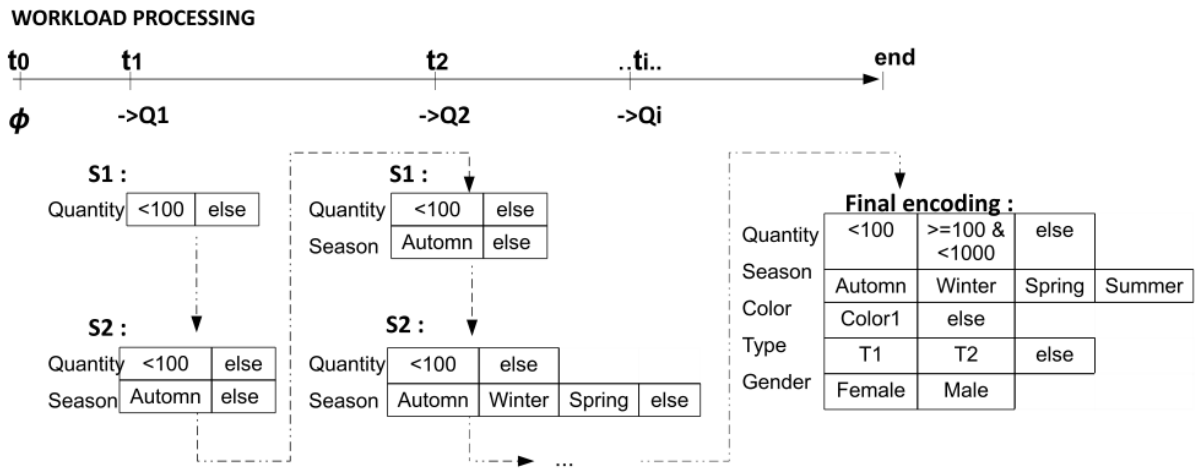


Figure 18 - Codage par Split Horizontal et Vertical sur les sous-domaines d' attributs [reff_6]

Après avoir trié tous les attributs, chaque attribut subit des opérations de fusion/éclatement (Merge/Split) de la manière suivante :

1. Les sous-domaines qui utilisés par aucune requête élue sont regroupés en une même partition
2. Les sous-domaines les plus utilisés, sont regroupés en une seule partition

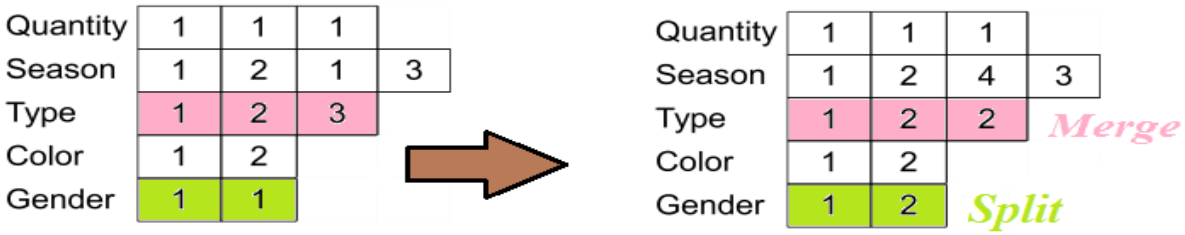


Figure 19 - Eclatement et fusion des partitions par corrélation des requêtes

Si N < W après ces opérations de fusion et d'éclatement en considérant seulement les requêtes élues, alors la fragmentation reste possible. Pour cette raison, le processus d'optimisation enchaîne avec un nouvel ensemble de requêtes élues à satisfaire. L'ensemble suivant des requêtes est celui des successeurs des RE courantes. Si au moins un groupe contient encore un successeur (parmi les requêtes triées par coût minimal), alors un nouvel ensemble de requêtes élues est généré par les successeurs retrouvés de tous les groupes. Le même processus est appliqué pour étendre le schéma de codage avec les nouveaux attributs et sous-domaines requis de façon incrémentale. Le processus réitère jusqu'à ce que N < W ou qu'aucune requête ne reste dans aucun groupe.

III.5.2. Travaux existant dans le contexte distribué

« Travaux de Noaman et al »

Pour fragmenter un entrepôt de données distribué, les auteurs [reff_4] se sont basés sur la fragmentation horizontale, et utilisent une politique descendante. Cette politique permet de partitionner le schéma conceptuel global d'un entrepôt de données pour construire les schémas conceptuels locaux. Cette répartition se fait en deux étapes essentielles : la fragmentation et l'allocation, suivies éventuellement d'une optimisation locale. Les auteurs utilisent un ensemble d'algorithmes différents pour la fragmentation.

III.5.3 Travaux existants dans le contexte parallèle :

« Travaux de Soumia_Benkrid » :

Les auteurs de ce travail [reff_1], ils ont proposé deux approches F&A (Fragmentation et allocation) et F&A&R (Fragmentation, allocation et réplication) pour la résolution de problème de la fragmentation horizontale d'entrepôt de donnée parallèle, se sont basés sur le modèle de coût, et prend en considération le mode isolée et ignorant l'interaction entre les requêtes.

Dans un premier temps : ils ont proposé l'approche F&A nommée aussi par l'approche pas à pas, Pour la fragmentation ils ont basés sur deux algorithmes « l'algorithme Hill Climbing et sur l'algorithme génétique », Cette approche consiste à fragmenter l'entrepôt de données modélisé par un schéma en étoile, pour cela ils ont identifié les tables de dimension participant dans la fragmentation de la table des faits a partir de la charge de requêtes, ça se fait par les étape suivant :

1. extraire tous les prédicats de sélection utilisés par les requêtes de Q
2. attribuer à chaque table de dimension Di son ensemble de prédicats de sélection

3. Eliminer les tables de dimension Di qui n'ayant pas des prédicats de sélection participé dans la charge des requête Q.
4. décomposer le domaine des attributs de fragmentation en plusieurs sous-domaines
5. représentation du schéma de fragmentation (chromosome) par un tableau multidimensionnel

Ensuite, pour chaque sous-domaine qui représente une partition ils ont codé par un numéro, Les sous-domaines qui possèdent le même numéro sont fusionnés en un seul ensemble si la contrainte de maintenance (nombre de fragment < seuil) n'est pas vérifié.
Après ils ont fait une allocation des fragments résultant aux divers nœuds d'une grappe de base de données pour placer chaque chromosome.
Pour le problème d'allocation, ils ont proposé une variété de l'approche du placement circulaire, F&A- ALLOC, où au lieu d'allouer un fragment par nœud, nous avons un ensemble de fragments. L'ensemble des groupes de fragments est défini selon le même principe que l'algorithme d'affinité de Navathe

Dans un second temps : une méthode de déploiement nommée F&A&R a été appliqué, combinant la fragmentation, l'allocation et la réplication. Elle suit la même démarche de F&A, formalisation et la proposition des algorithmes de sa résolution. Un algorithme basé sur La logique floue est présentée pour le processus d'allocation avec réplication. Son principal objectif est d'unifier les phases de déploiement d'un EDP.

IV. Synthèse des travaux

La fragmentation horizontale est l'une des techniques d'optimisation les plus largement adoptées dans la phase de conception physique. Cette technique a évolué parallèlement aux bases de données et aux supports de stockage, en commençant par les bases de données classiques jusqu'aux bases de données Cloud, Dans les différentes générations deux type d'approches ont été proposées :

- **sans contrainte de maintenance :** les approches basées sur les minterms et d'autres basées sur l'affinité des prédicats
- **avec contrainte de maintenance :** les approches basées sur modèle de coût et d'autres basées sur les techniques du Data Mining

Cependant, toutes les approches existantes qui traitent le problème de la fragmentation horizontale ignorent l'interaction entre les requêtes, c'est-à-dire que toutes les requêtes auront la même probabilité de participer dans le processus de partitionnement, dans la nouvelle génération autre caractéristique importante a été introduite. Cette caractéristique est l'interaction entre les requêtes, cette interaction a été utilisée pour définir des méthodes de sélection des fragmentations horizontale. La figure 20 résume la classification des travaux

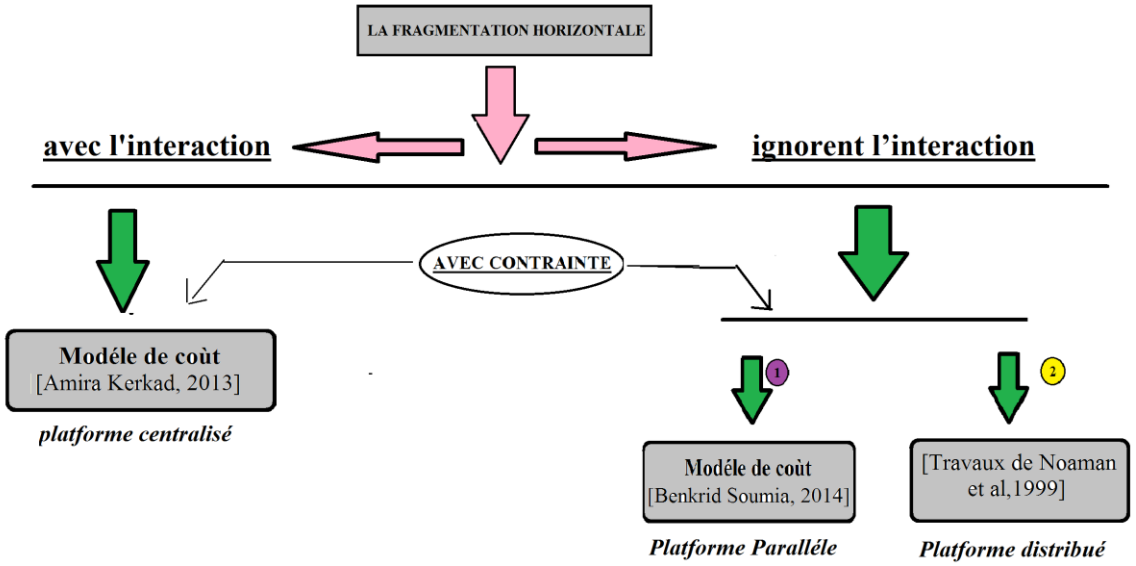


Figure 20 - Classification des travaux sur la fragmentation horizontale

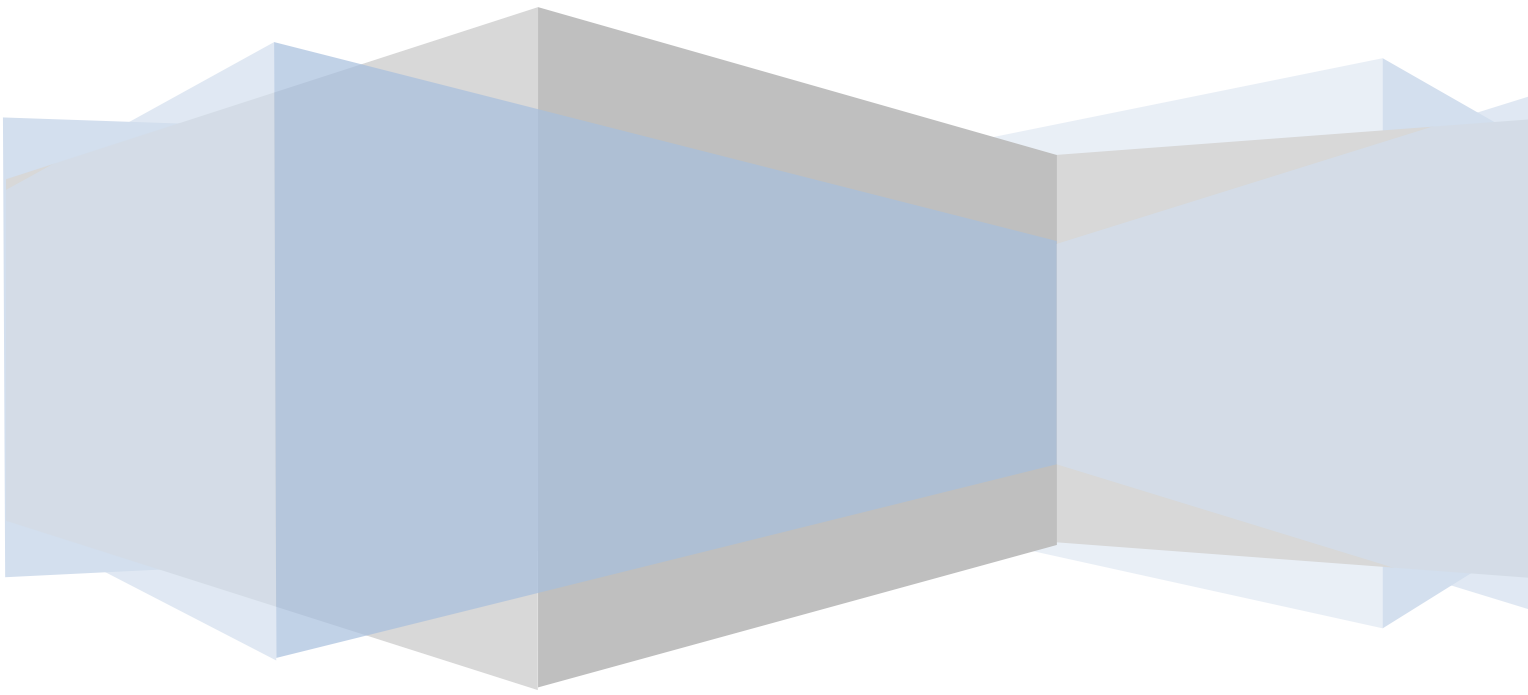
V. Conclusion

Dans ce chapitre, on a vu la technique principal qui est la fragmentation horizontal, cette dernière est utilisée pour optimisé le temps d'exécution des requêtes dans l'entrepôt de données, nous avons mis en évidence les problèmes liées à cette technique, vu que la fragmentation horizontal est une structure non redondante et ne duplique pas les données, comme nous avons présenté à la fin de ce chapitre quelques travails qui ont déjà existé dans les différent plates formes de l'entrepôt et une synthèse qui permet de faire la comparaison entre les travaux qui sont présentés.

Dans Le chapitre suivant sera consacrée à la présentation de l'approche proposé qui permet de résoudre le problème du temps d'exécution des requêtes dans l'entrepôt de donnée

Chapitre III

Conception du schéma de fragmentation



I. Introduction :

Dans ce chapitre nous présentons la démarche retenue pour la fragmentation horizontale afin d'améliorer les performances de requêtes. Notre travail c'est la continuation sur une approche basée sur l'utilisation de la technique du Data Manning, cette dernier passe par deux différentes étape a savoir la recherche des prédicats, la classification des requête ensuite, nous nous somme chargé de faire la fragmentation horizontal dérivé avec allocation des fragments obtenus sur les différents sites et en fin, nous présentons environnement de développement de notre application avec ces différentes fonctionnalités et la base de données à utiliser.

II. L'approche proposée :

Au premier temps suivant l'approche proposée dans le travail [reff_8] il faut regrouper initialement les requêtes de la base de données qui sont similaires syntaxiquement suivant les nœuds de jointure au lieu d'un partitionnement initial aléatoire comme dans le travail [reff_11], premièrement ils ont commencé par extraction des jointures pour chaque requête :

Exemple : en prenant un exemple de 8 requêtes désignées ci-après.

Q0: select sum (lo_extendedprice*lo_discount) as revenue from lineorder, dates where lo_orderdate = d_datekey and d_year = 1993 and lo_discount >= 1 and lo_discount <= 3 and lo_quantity < 25

Q1: select count (*) from lineorder, dates where lo_orderdate = d_datekey and d_year = 1993 and lo_discount >= 1 and lo_discount <= 3 and lo_quantity < 25

Q2: select sum (lo_extendedprice*lo_discount) as revenue from lineorder, dates where lo_orderdate = d_datekey and d_year = 1993

Q3: select count (*) from lineorder, dates where lo_orderdate = d_datekey and d_year = 1993

Q4: select sum (lo_revenue), d_year from lineorder, dates, part, supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_brand1 = 'mfgr#2221' and s_region = 'asia' group by d_year order by d_year

Q5: select sum(lo_revenue) from lineorder, part where lo_partkey = p_partkey and p_brand = 'MFGR#2221'

Q6: select avg(lo_revenue), d_year from lineorder, dates, part, supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_brand = 'MFGR#2221' and s_region = 'ASIA' group by d_year order by d_year

Q7: select sum(lo_revenue), d_year from lineorder, dates, part, supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_brand = 'MFGR#2221' and s_region = 'EUROPE' group by d_year, p_brand order by d_year, p_brand

II .1 La 1^{ér} partie : l'extraction des jointures

Avec cette charge de requête, ils ont extrait premièrement toutes les jointures de ces requêtes et chaque jointure est codé par un numéro (9, 10, 11, 12, 13) respectivement.

Les jointures

9: LO_ORDERDATE = D_DATEKEY
10: LO_ORDERDATE = D_DATEKEY
11: LO_ORDERDATE = D_DATEKEY
12: LO_PARTKEY = P_PARTKEY
13: LO_SUPPKEY = S_SUPPKEY

Après ils ont préparé les donnée d'apprentissage dans approche proposée qui sont les nœuds de jointures extraits au dessus pour chaque requête, tel que M[i][j] est égale à 1 si le nœud de jointure apparait dans la requête Qi,0 sinon.

	9	10	11	12	13
Q0	1	0	0	0	0
Q1	1	0	0	0	0
Q2	0	1	0	0	0
Q3	0	1	0	0	0
Q4	0	0	1	1	1
Q5	0	0	0	1	0
Q6	0	0	1	1	1
Q7	0	0	1	1	0

Tableau 1- donnée d'apprentissage.

Et ensuite, ils ont regroupé les requêtes dans des classes C, selon les jointures communes entre eux, et le résultat :

- C0= {Q0, Q1}, avec jointure Nj=9.
- C1= {Q4, Q6, Q7}, avec ensemble des jointures Nj= {12 ,13 ,11}
- C2= {Q5}, avec jointure Nj=12
- C3= {Q2, Q3}, avec jointure Nj=10

On remarque qu'il y a des jointures qui se répète dans plusieurs classe (jointure numéro 12), cela veut dire que il existe des interactions entres ces classes, donc ils ont fusionner les classe suivant une jointure qui est la plus utilisée, sachant que tous les requêtes dans la même classe corrélèrent au moins dans la première jointure on l'appel le *nœud score*, ce nœud ils permet de faire l'intersection entre une classe et une autre, pour cela ils ont proposé une fonction intersection inter-classes qui permet de trouvé le nœud score .

II. 2 La 2ème Partie : recherche des prédicats de sélection

Après avoir trouvé les nœuds score, il nous reste que la cherche du prédécesseur de ces nœuds qui sont les prédicats de sélection qui participant à la fragmentation en utilisant le schéma MVPP. Après fusionnement des classes suivant la jointure commun entre eux, ils ont obtenu ces nouvelles classes :

- C3= {Q0, Q1}, avec jointure Nj=9
- C0= {Q2, Q3}, avec jointure Nj=10
- C4= {C1} ∩ {C2} = {Q4, Q5, Q6, Q7}, Nj={11, 13,12}.

❖ **Le schema MVPP (Multiple View Processing Plan)**

Le MVPP est une représentation graphique de la charge des requêtes introduit par utilisateur, ce plan est composé de plusieurs niveaux tel que le premier niveau représente les tables de base de l'entrepôt, le deuxième niveau représente les sélections, le troisième niveau contient les opérations du jointure et le quatrième niveau représente la charge des requêtes, on trouve que les opérations de sélection sont poussé en bas de schéma du MVPP.

En prend exemple précédent le MVPP (figure 21) correspond à les classes C0, C3, C4 tel que les nœuds de jointure dans la classe sont triés suivant le nombre d'utilisation dans la classe et à travers MVPP on capte les prédicats de sélection adéquate à la fragmentation

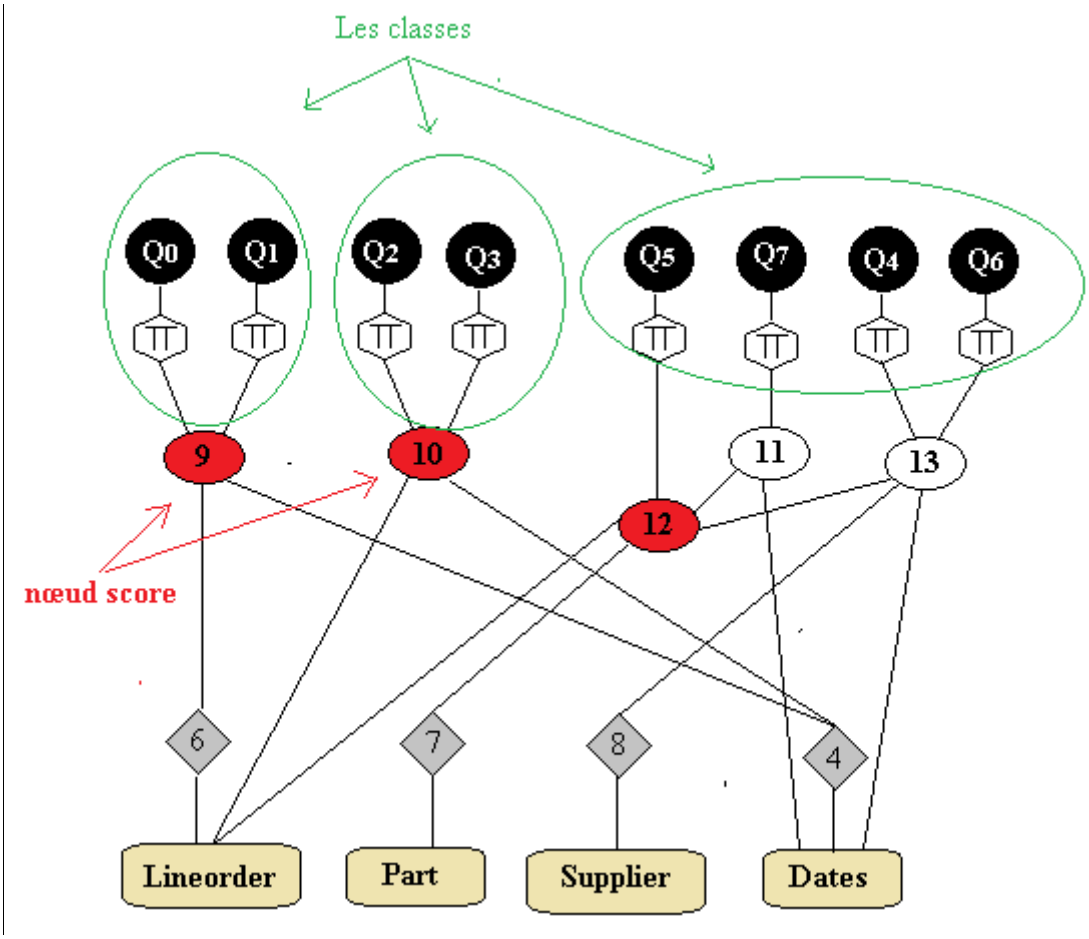


Figure 21 - Multi view processing plan (MVPP) pour la classe C7.

Nous assumons la configuration suivante de notre entrepôt parallèle:

- Un schéma en étoile composé d'une table de faits: 'Lineorder' et quatre tables de dimensions: 'fournisseur, dates, client et partie'.
- En haut de ce schéma, on travailler avec 7 requêtes en étoiles {Q0, ..., Q7}. On distingue trois types de nœuds de ce plan: une opération de sélection, une opération de jointure désignée par Ji et une opération de projection notée π.
- Notez que quatre (4) sélections et cinq (5) jointures sont identifiées

Nous constatons que les requêtes peuvent être regroupées en trois groupes principaux appelés classes: C4 = {Q4, Q5, Q6, Q7}, C0 = {Q2, Q3} et C3 = {Q0, Q1}. Chaque classe contient un ensemble de requêtes qui partagent au moins une opération de jointure. Le premier nœud de jointure partagé s'appelle le nœud score. Suivant MVPP, nous avons trois nœuds score (12 ,10 ,9) avec leurs ensembles de prédicats de sélections sont :

{S_REGION='EUROPE'},{D_YEAR=1993} et {D_YEAR=1993} respectivement.

II .3 La 3ème Partie : la fragmentation dérivée

Dans cette partie on va travailler avec l'approche de ahcéne [reff_11], dans notre cas nous avons trois nœuds scores avec leurs ensembles de prédicats de sélections sont {S_region = "EUROPE"} et {d_year = 1993}, respectivement. Notez que la notion de nœud score est assez importante, car elle guide le processus de partitionnement d'un classe au moyen de son ensemble de prédicats de référence (l'ensemble des prédicats de sélection).

Un prédicat prédécesseur de nœud du score partitionne son attribut correspondant en deux partitions, l'une avec toutes les instances satisfaisant le prédicat (par exemple, s region = "EUROPE") et une autre représentant la partition ELSE (s region <> "EUROPE"). Notez que le partitionnement des tables de dimension sera propagé pour partitionner la table des faits. Ce partitionnement s'appelle partitionnement dérivé. Ce schéma de partitionnement initial d'une classe sera raffiné en considérant d'autres prédicats de l'ensemble des prédicats de référence et d'autres prédicats n'appartenant pas au nœud score (par exemple, S7).

- F1 = {S_REGION='EUROPE' and P_BRAND='MFGR#2221' }
- F2 = {S_REGION='EUROPE' and P_BRAND='MFGR#2222' }
- F3 = {S_REGION='EUROPE' and P_BRAND='MFGR#2223' }
- F4 = {S_REGION='EUROPE' and P_BRAND='MFGR#2224' }
- F5= { S_REGION <> 'EUROPE'}

Le fragment restant de la classe initial (F5 dans notre exemple) sera concerné par le processus de partitionnement pour autre classe C0 ou bien C3. Le raisonnement de partitionnement et d'allocation ci-dessus appliqué à C0. Notez que les requêtes de la classe C0 nécessitent des fragments de la classe C4.

Un ordre de partitionnement doit être défini parmi les classes, Suivant cette proposition [reff_11], nous accordons plus d'importance à la classe qui implique des questions les plus coûteuses. Les classes avec un ensemble de prédicats de référence vide ne sont pas pris en compte pour le processus de partitionnement et bien sur que les sélections de la table de faits seront ignorées (par exemple S6: LO_QUANTITY<25).

Donc, on utilisera {**S_REGION='EUROPE', D_YEAR=1993**} pour la fragmentation, et maintenant nous appliquons l'algorithme cité dans le travail [reff_10].

- Les sélections :

S4: D_YEAR=1993
S5: P_BRAND='MFGR#2221'
S6: LO_QUANTITY<25
S7: S_REGION='ASIA'
S8: S_REGION='EUROPE'

- Les attributs de domaine :

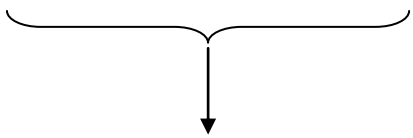
D_YEAR :{ 1992, 1993, 1994, 1995, 1996, 1997, 1998}.

S_REGION :{ ASIA, EUROPE, AFRICA, AMERICA, MIDDLE EAST}.

P_BRAND :{ MFGR#2221, MFGR#2222, MFGR#2223, MFGR#2224}.

Les fragments obtenus après l'application de l'algorithme sont:

F1 = {S_REGION='EUROPE' and P_BRAND='MFGR#2221' }
F2 = {S_REGION='EUROPE' and P_BRAND='MFGR#2222' }
F3 = {S_REGION='EUROPE' and P_BRAND='MFGR#2223' }
F4 = {S_REGION='EUROPE' and P_BRAND='MFGR#2224' }
F5= { S_REGION <> 'EUROPE'}



F5= { S_REGION <> 'EUROPE' and D_YEAR=1993}
F6= { S_REGION <> 'EUROPE' and D_YEAR<> 1993}

II .4 La 4ème Partie : l'allocation des fragments

Après on fait une allocation des fragments résultant aux divers nœuds. On supposant que on a trois machine (site), et on utilise la méthode de l'allocation **Round-robin.**
Round-robin est un algorithme d'ordonnancement courant dans les systèmes d'exploitation. Ce dernier attribue les fragments à chaque machine (site) en proportion égale, sans accorder de priorité aux processus.

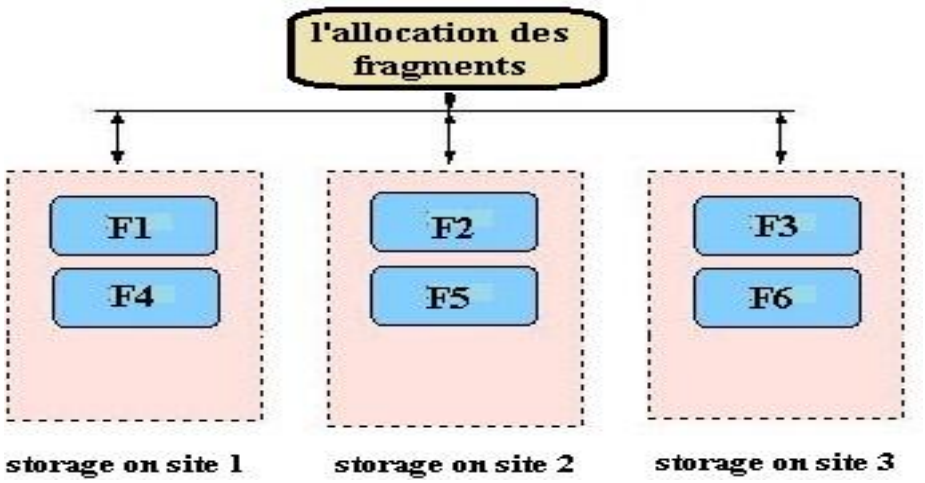


Figure 22 – l'allocation des fragments sur les sites

III. Environnement de développement

L'application est développé par le langage de programmation Java et on utilise l'Eclipse pour gérer notre entrepôt de données, on utilise Oracle SQL*Plus.

- **Java** : Java est un langage de programmation et une plate-forme informatique qui a été créé par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé, et leurs nombres ne cessent de croitre chaque jour. Java est rapide, sécurisé et fiable. Des ordinateurs portables aux centres de données, des consoles de jeux aux super ordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente sur tous les fronts.

- **Éclipse** : Eclipse est une plate-forme java open source et un environnement de développement intégré (Integrated Development Environment) développé par I.B.M, dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques. Eclipse utilise énormément le concept de modules nommés "plug-ins" dans son architecture. D'ailleurs, hormis le noyau de la plate-forme nommé "Runtime", tout le reste de la plate-forme est développé sous la forme de plug-ins. Ce concept permet de fournir un mécanisme pour l'extension de la plate-forme et ainsi fournir la possibilité à des tiers de

développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse, autres langages de programmation y compris Ada, C, C++, COBOL, Fortran, JavaScript, Perl, PHP, Python [reff_13].

- **Oracle SQL*plus** : Est un utilitaire en ligne de commande d'Oracle qui permet aux utilisateurs d'exécuter interactivement des commandes SQL et PL/SQL. Décliné en plusieurs versions (graphique et web), il est principalement distribué avec le produit Oracle Database. Oracle a été le premier SGBD commercialisé sur le marché. Pour cela, nous avons choisi *ORACLE 10g* comme SGBD. Ce choix est justifié par sa puissance et son efficacité, en termes de sécurité, volume de données traitées, ... etc. [reff_12].

IV. Notre base de données

Afin de valider l'approche proposée pour trouver les prédicats de sélection qui participe a la fragmentation, nous avons utilisé le banc d'essai Star Schéma Benchmark (SSB) qui est le schéma en étoile du banc d'essai. Le SSB est conçu pour mesurer la performance des produits de base de données à l'appui des applications d'entreposage de données classiques, et est basé sur le benchmark TPC-H [TPC-H]. SSB comporte une table des faits LINEORDERS et quatre tables de dimension PART, SUPPLIER, CUSTOMER et DATES.

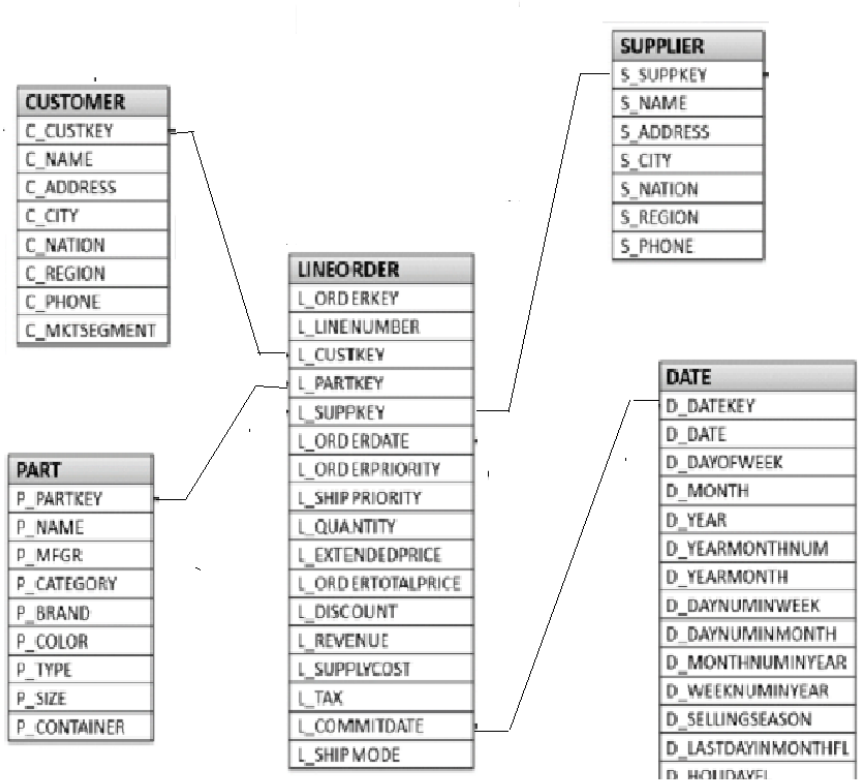


Figure 23- Schéma logique du banc d'essai SSB.

IV.1 Jeu de données

Nos expérimentations ont été réalisées sur le benchmark SSB (Star Schéma Benchmark [157]), Ce benchmark contient un entrepôt de données dont le schéma en étoile comporte quatre tables de dimensions autour d'une table des faits.
La Table 2, présente les différentes tables du schéma en étoile du SSB avec leur type et les cardinalités de chaque table. Pour notre étude, nous avons utilisé une charge de requêtes SSB contiens 100 requête de jointure en étoile au maximum.

Table	Type	Cardinalité
Lineorder	Faits	**59986052**
Supplier	Dimension	**20000**
Customer	Dimension	**300000**
Part	Dimension	**800000**
Dates	Dimension	**2556**

Tableau 2- Cardinale des tables de SBB.

V. Présentation de l'application

Nous présentons les différentes étapes d'application fournit par notre encadrante qui consiste à analyser la charge de travail pour former les groupes des requêtes qui sont en interaction grâce aux nœuds de jointure et nous nous chargé de faire la fragmentation et allocation tout d'abord en identifiant les prédicats de sélection appliqués avant l'exécution de ces nœuds de jointure (prédécesseur de nœud de jointure) suivant MVPP, ensuite en extraire tous prédicats de sélection qui participe a la fragmentation

V.1. Connexion de la base de données

Cette fenêtre permet de faire la connexion sur notre base de données.

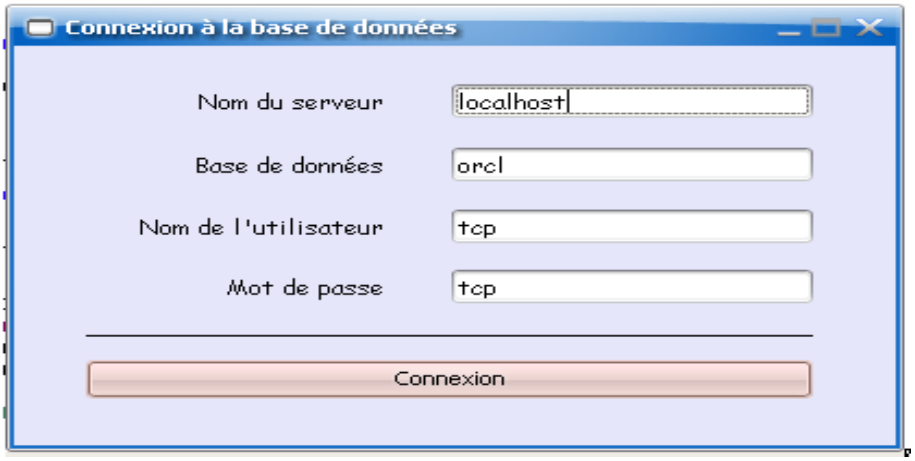


Figure 24 - Connexion de la base de données.

V.2. Notre interface générale

Interface générale contient 8 boutons et une zone pour afficher les résultats :

- Bouton « Charger requête» (utilisateur qui introduit le nombre de requête)
- Bouton « extraire les table »
- Bouton « extraire les sélections »
- Bouton « extraire les jointures »
- Bouton « intersection des classe »
- Bouton « affiché les nœuds score »
- Bouton « affiché les fragments » (utilisateur peut contrôler le nombre de fragment affiché)
- Bouton « Allocation de fragment »
- Une zone pour afficher tous les résultats obtenus.

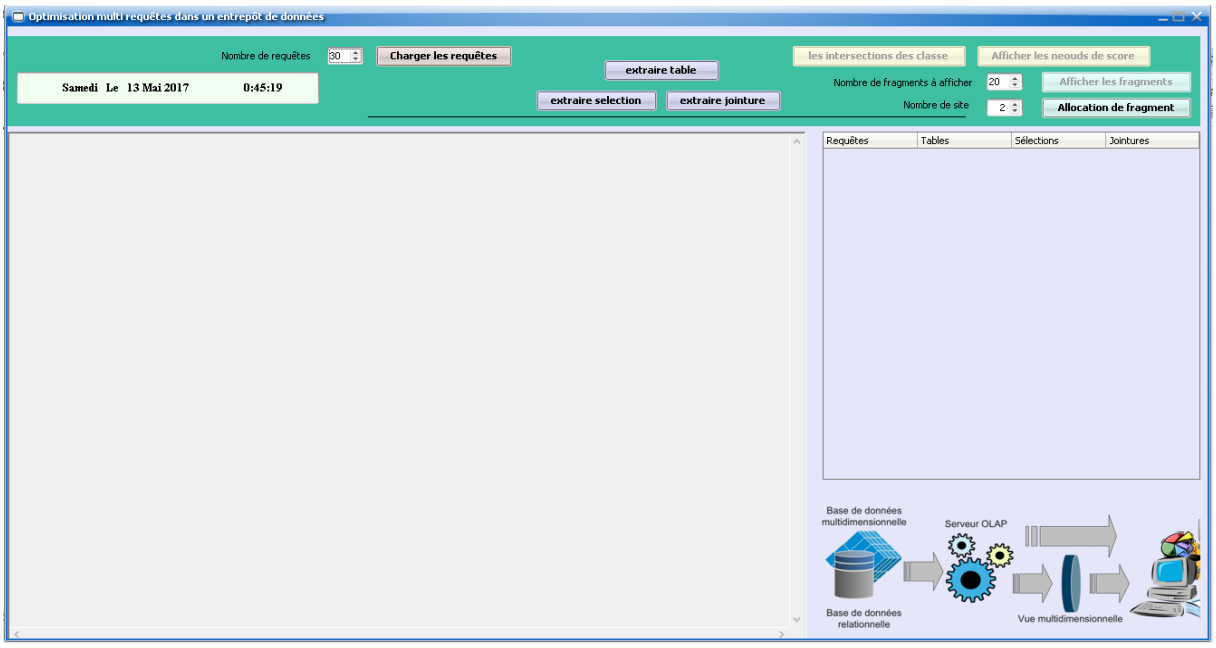


Figure 25 - Interface générale.

V.3. Charger les requêtes

Cette fenêtre permet de charger toutes les requêtes de notre travail, si on clique sur le bouton charger Q, nous avons travaillé sur 30 requêtes.

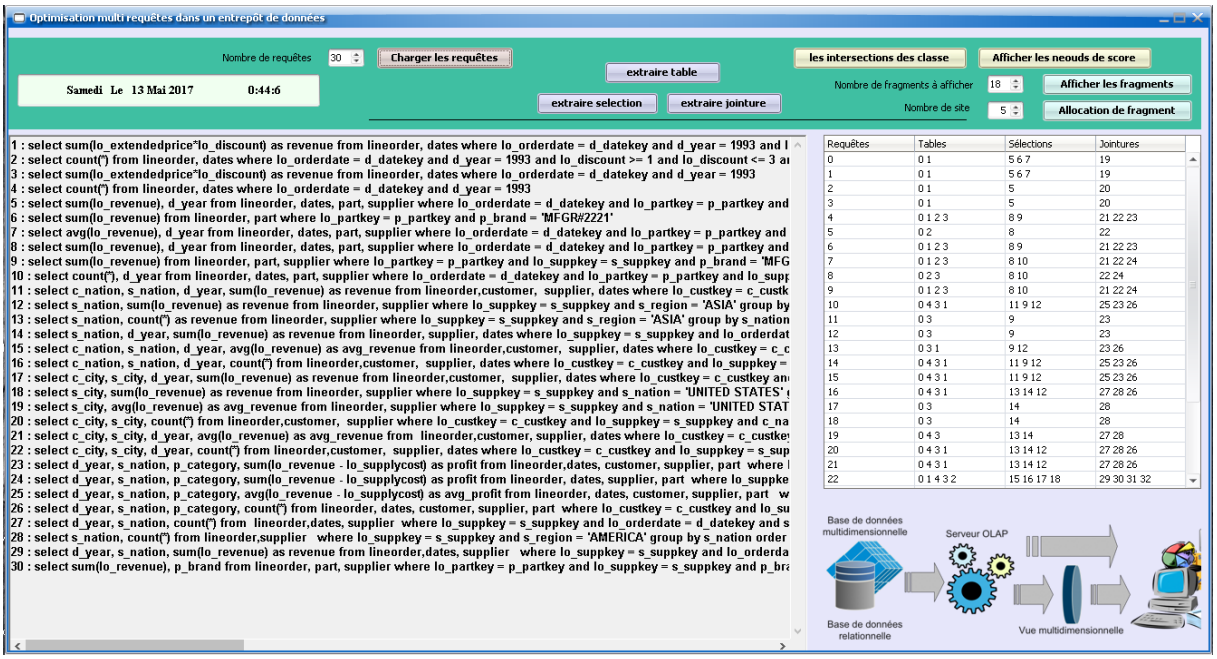


Figure 26 - Charger les requêtes.

V.4. extraire les table

Pour trouver les tables de chaque requête, tout d'abord on spécifie le nombre de requêtes, dans notre exemple, nous avons choisit 30 requêtes. Le bouton « extraire les table » permet d'afficher les requêtes et chaque requête avec leurs tables,

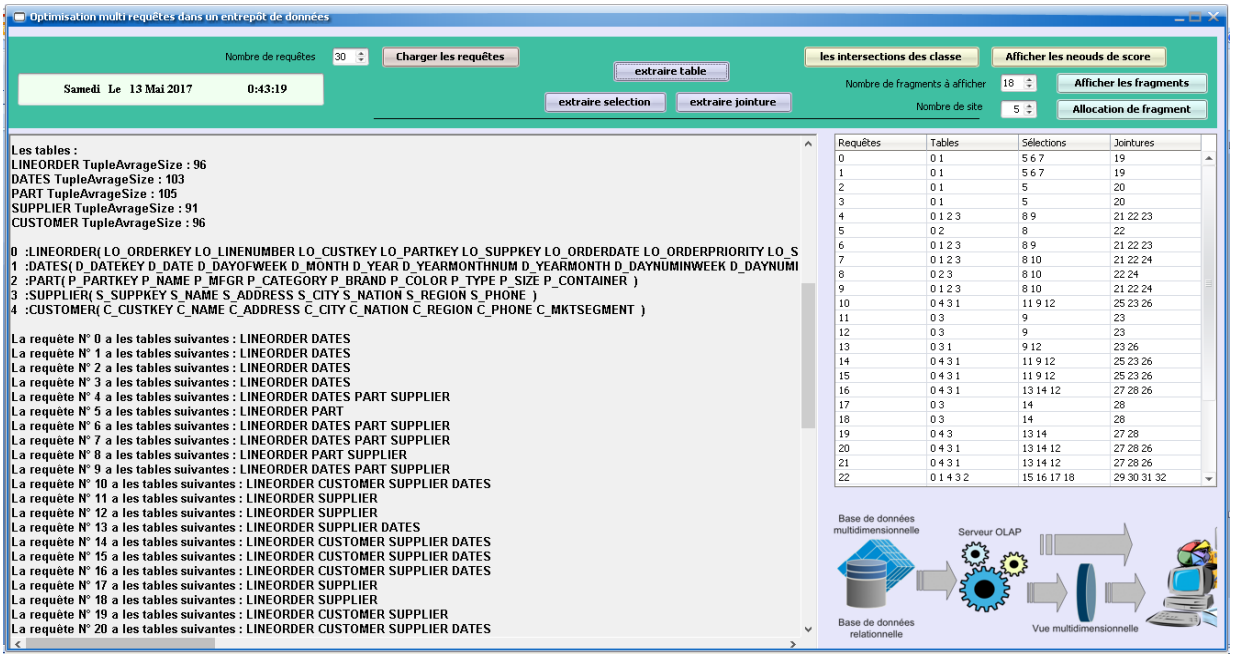


Figure 27 - extrait les tables.

V.5. extraire les sélections

Le bouton « extraire les sélections » permet d'afficher les sélections et chaque prédicat de sélections sera codé par un numéro

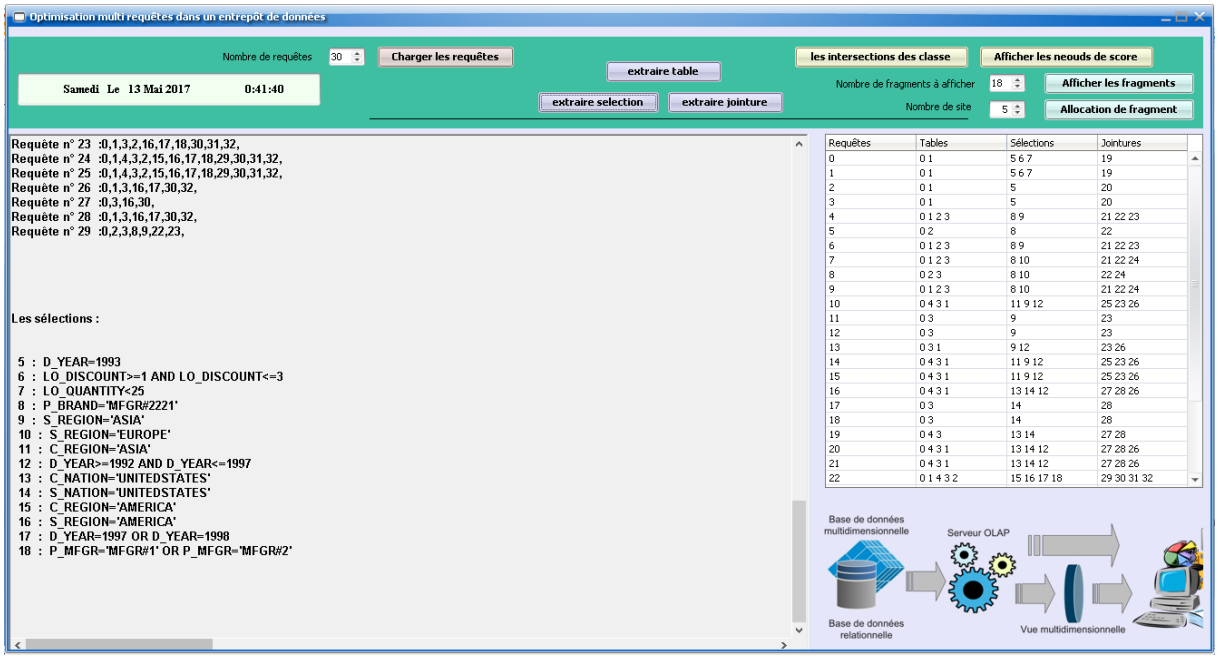


Figure 28 - extrait les sélections.

V.6. extraire les jointures

Le bouton « extraire les sélections » permet d'afficher tous les jointures de la charge des requêtes et chaque jointure sera codée aussi par un numéro

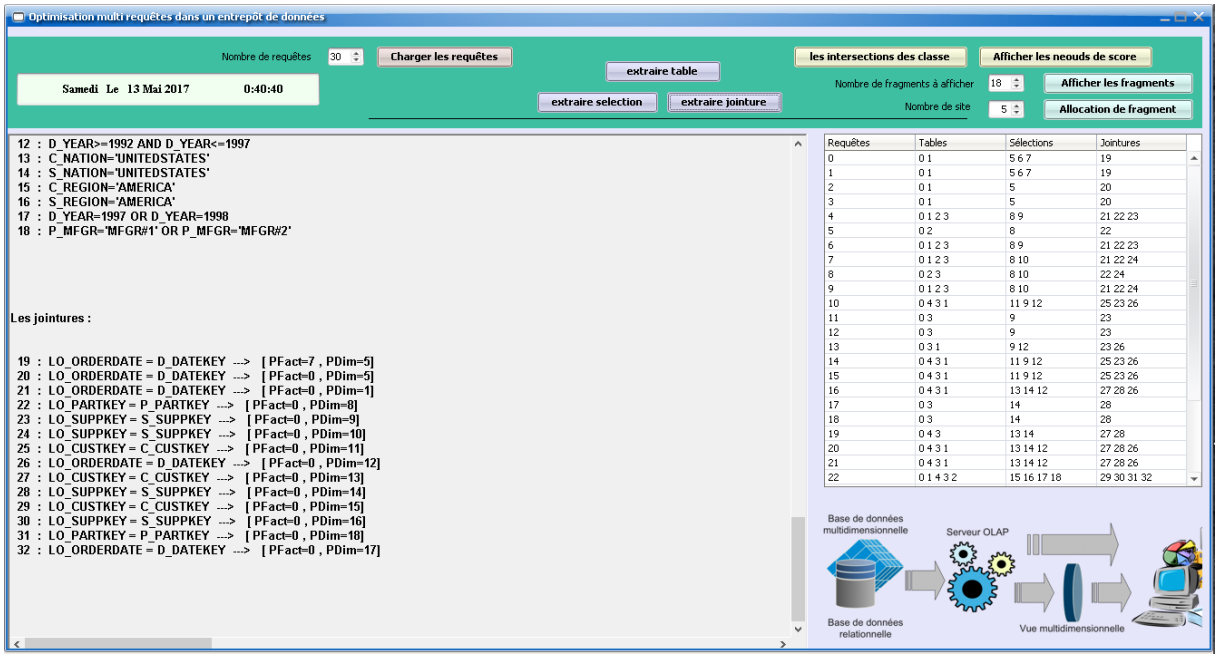


Figure 29 - extrait les jointures.

V.7. intersection des classes

Le bouton « intersection des classes » permet de capter les interactions entre les requêtes et d'afficher le nombre de cluster possible.

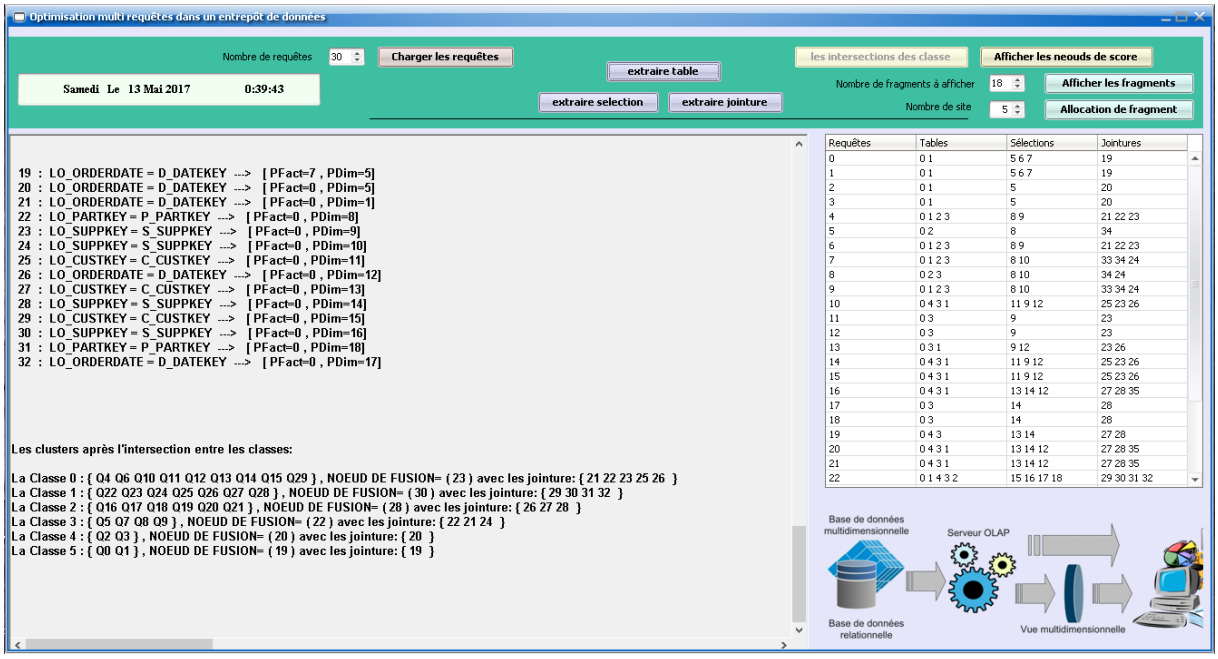


Figure 30 - fenêtre intersection des classes.-

V.8. affichage les nœuds score

Le bouton « afficher nœud score » permet d'extraire tous les nœuds de score de chaque cluster avec leurs prédécesseurs

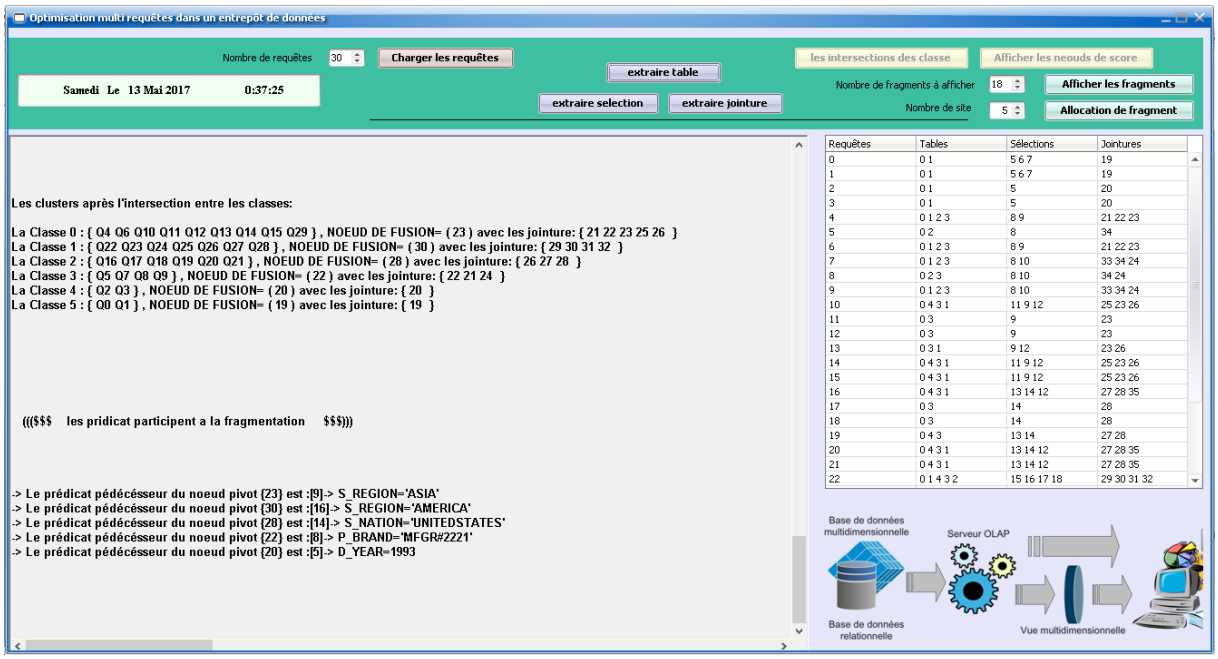


Figure 31 - fenêtre affiche les nœuds score.

V.9. affichage les fragments

Le bouton « afficher les fragments » permet d'afficher tous les fragments possible suivant le seuil, dans notre exemple, nous avons choisit le seuil égal à 18 et les résultats ne dépasse pas 18 fragments.

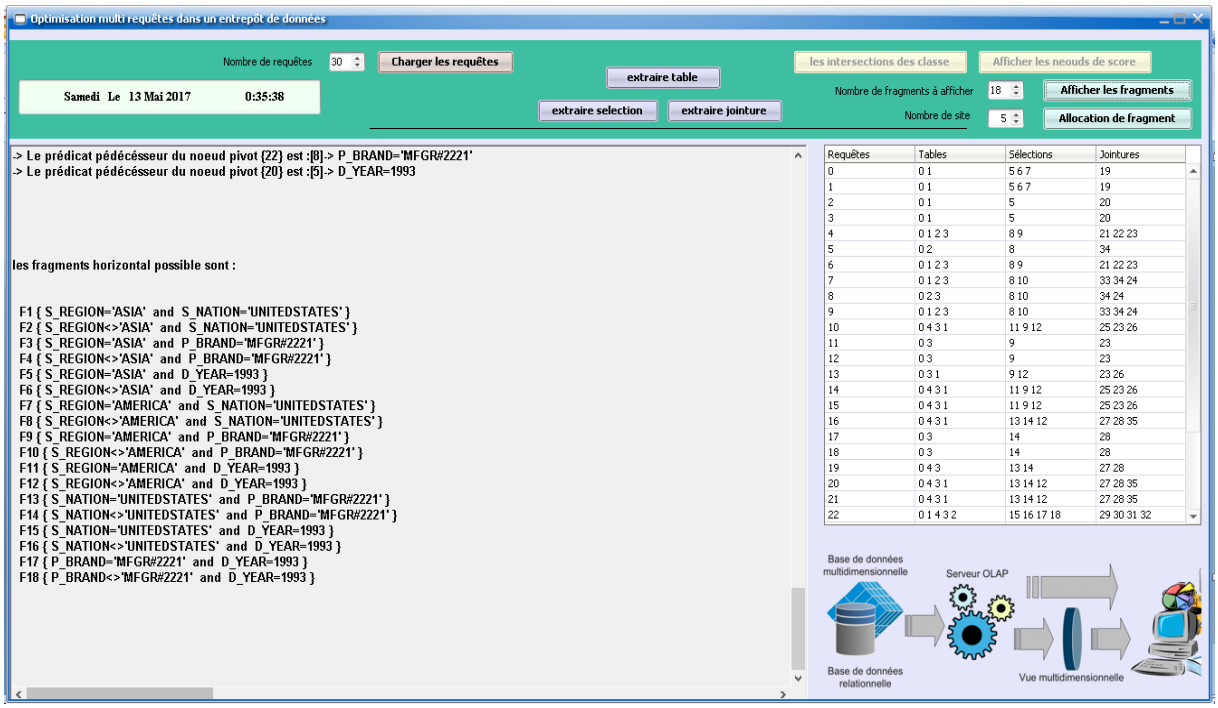


Figure 32 - fenêtre affiche les fragments.

V.10. l'allocation des fragments

Le bouton « allocation des fragments » permet de regroupé les fragments obtenus, en plusieurs sous ensemble de fragment suivant le nombre des sites dans notre exemple, nous avons travaillé avec une machine contient cinq sites

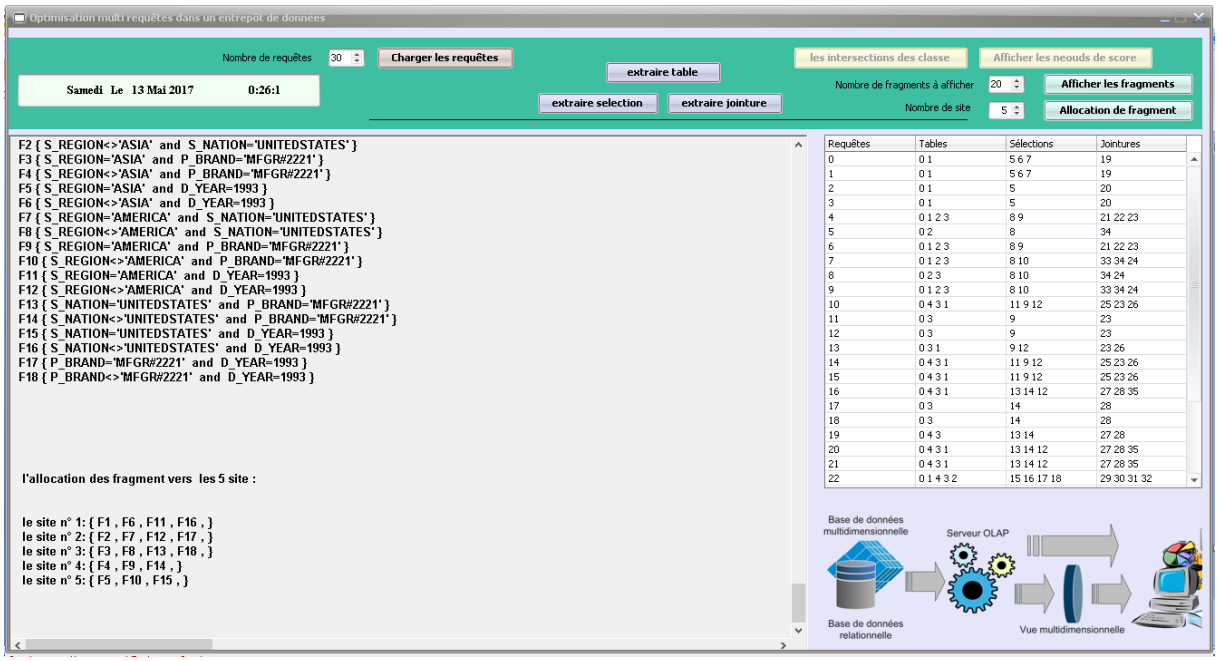


Figure 33 – L'allocation des fragments

VI. Conclusion :

Dans ce chapitre, nous avons décrit l'outil que nous avons implémenté pour effectuer l'étude de l'approche proposée. Cet outil permet de capter l'interaction entre les requêtes afin de trouver les prédicats de sélection qui participent à la fragmentation et nous nous avons implémenté seulement la deuxième partie de l'approche proposée qui consiste à charger de faire la fragmentation et l'allocation sur les site, en utilisant Star Schéma Benchmark (SSB) avec une charge de requête composée de 30 requêtes, avec la possibilité d'augmenter la charge de requête. Ensuite Nous avons décrit les différentes phases de développement de notre application.

Conclusion Générale

Conclusion générale

Les entrepôts de données sont devenus maintenant non pas un phénomène de mode mais un instrument indispensable à la bonne marche de l'organisation. Ils sont en effet à la base de toute stratégie et prise de décision de l'entreprise. Il faut en extraire les informations nécessaires à la prise de décision et également leur structure .De cet entrepôt sont extraites des bases de données multidimensionnelles, car elles permettent de regarder l'organisation sous différents angles ou dimensions, ces dernières sont constituées que de données propres à la décision.

Pour assurer une bonne conception physique des entrepôts de données, il faut utiliser Des structures non redondantes comme la fragmentation des données et redondantes comme les vues matérialisées et l'indexation.

Dans ce projet, nous nous sommes intéressés à la fragmentation horizontale non redondante des entrepôts de données relationnels qui doit être l'une des techniques parmi les autres techniques principales utilisées pour optimiser les performances et le temps d'exécution des requêtes. Vu que la fragmentation horizontale est une structure non redondante et ne duplique pas les données, la plupart des algorithmes de fragmentation horizontale existants donne un nombre de fragments que l'on ne peut pas contrôler ,dans ce document nous avons répondu à ce problème par implémentation de la parti de la fragmentation et nous avons utilisé le Star Schéma Benchmark (SSB) avec une charge de requête composée de 30 requêtes avec la possibilité d'augmenter la charge de requête.

En fin, ce thème a été pour nous une grande opportunité pour développer nos connaissances théoriques par la recherche documentaire et nos connaissances pratiques par la programmation. Ce projet nous a permis de nous familiariser avec un domaine intéressant et très vaste et nous souhaitons qu'on ait résolu une grande partie du problème de la gestion des entrepôts de donnée, en désirant que nous ayons l'occasion de développer et d'enrichir notre application pour plus d'efficacité, selon les perspectives suivantes :

- ✓ Nous proposent premièrement d'implémenter le modèle de cout pour comparer avec les travaux connexe
- ✓ de combiner plusieurs structures d'optimisation telle que les vues matérialisées, les index …etc., pour optimiser les requêtes,
- ✓ Améliorer l'interface graphique.
- ✓ créer un système de monitoring pour décider si on refait ou pas la fragmentation pour une nouvelle charge de requête ou même pour l'arrivé d'une nouvelle requête.

Bibliographique

[reff_1] Benkrid. Soumia et al, « Le déploiement, une phase à part entière dans le cycle de vie des entrepôts de données: application aux plateformes parallèles », Thèse doctorat, ISAE-ENSMA, Soutenue le 24/06/2014.

[reff_2] Filali Abderrahmane, Kedjnane Sofiane, « Conception et réalisation d'un Data Warehouse pour la mise en place d'un système décisionnel », ESI, Mémoire de fin d'études Pour l'obtention du diplôme d'Ingénieur d'Etat en Informatique, Promotion ,2009/2010.

[reff_3] L . Bellatreche « Utilisation des Vues Matérialisées, des Index et de la Fragmentation dans la Conception Logique et Physique d'un Entrepôt de Données » thèse de doctorat, université Clermont-Ferrand II Décembre 2000.

[reff_4] Noaman. A et al , «A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse», In Proceedings of the eighth international conference on Information and knowledge management, (November, 1999), (pp. 154-161).

[reff_5] Bellatreche, « Techniques d'optimisation des requêtes dans les data Warehouse », Article, In 6th International Symposium on Programming and Systems (ISPS 03), Alger, Alegria, (2003), (pp. 81-98).

[reff_6] Amira Kerkad. L'interaction au service de l'optimisation à grande échelle des entrepôts de données relationnels. Other. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechnique - Poitiers, 2013. French.

[reff_7] Saichi souad. « Optimisation de requête dans l'entrepôt de donnée ». Mémoire de fin d'études Pour l'obtention du diplôme de magister en informatique, soutenu le 27/06/2009

[reff_8] Mostefa Daouadji, Betouati.« Conception physique de l'entrepôt de données parallèle sur un cluster » Mémoire de Fin d'Etudes Pour l'Obtention du Diplôme de Master en Informatique Option : Ingénierie des Systèmes d'Information ,2016

[reff_9] Mbaiossoum Bery Leouro et al, « Conception physique des bases de données à base ontologique: le cas des vues matérialisées », Thèse de doctorat, ENSMA et Université de Poitiers, Soutenue le 12 décembre 2014.

[reff_10] Amara et Benouaz. « Techniques Data Mining pour la sélection d'une configuration d'index de jointure binaire, Architecture d'un entrepôt de donnée » Mémoire Pour l'obtention du diplôme de master en informatique, années universitaire 2010/2011

[reff_11] Ahcene Boukorca, Soumia Benkri, HYPAD: Hyper-graph-driven approach for Parallel Data warehouse design, Article, LIAS/ISAE-ENSMA - Poitiers University Futuroscope, France, 2015.

[reff_12] Didier Délégalise « Guide du développeur Oracle », livre, Edition Sup info Press 23, rue de château Landon -75010 Paris décembre 2001.

[reff_13] Laribi zineb, Rebai amina « ETUDE ET ANALYSE DES GRANDS GRAPHES D'INTERACTION ET EXPLOITATION DES SYSTEMES P2P POUR LA RECHERCHE DE DONNEES », Mémoire, Faculté des Sciences Exactes et d'Informatique, UNIVERSITE ABDELHAMID IBN BADIS MOSTAGANEM, 2014/2015.

[reff_14] Boukhalfa. Kamel et al, « De la conception physique aux outils d'administration et de tuning des entrepôts de données », Thèse doctorat, ENSMA et Université de Poitiers, juillet 2009.