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 : **Ingénierie des Systèmes d'Information**

THEME :

Optimisation multi-requêtes dans l'entrepôt de données

Etudiant(e) : Benyagoub nor el houda

Encadrant(e) : Mme Betouati fatiha

Année Universitaire 2015/2016

Résumé

Les entrepôts de données ont pris une place importante dans les préoccupations des utilisateurs des bases de données, L'idée pour la mise en œuvre d'un entrepôt de données est de fournir un accès permanent aux données même lorsque les bases de données individuelles sont inaccessibles, et de réduire les accès distants aux systèmes gérant les données d'origine.

Les entrepôts de données sont dédiés aux applications d'analyse et de prise de décision. Le processus d'analyse est réalisé à l'aide de requêtes complexes comportant de multiples jointures et des opérations d'agrégation sur des tables volumineuses, L'administrateur, dans le but de minimiser le coût d'exécution de ces requêtes, sélectionne un ensemble de vues matérialisées qui représentent les nœuds de jointure. Cette sélection diminue le coût des requêtes, mais entraîne le problème d'occupation de place pour les vues matérialisées, et en conséquence, ils ne peuvent pas être stockés en totalité dans la mémoire centrale.

Le travail présenté dans ce mémoire concerne l'optimisation des requêtes dans l'entrepôt de données, plus particulièrement la technique des vues matérialisées. Un important problème d'optimisation a été suggéré dans cette technique, la sélection de vues. le problème consiste à sélectionner l'ensemble des vues à matérialiser pour accélérer dans le futur l'exécution des requêtes.

Ce mémoire propose une solution à ce problème par l'utilisation des techniques du data-mining, plus précisément l'algorithme de classification k-means.

Mots Clés : Entrepôt de données, vue matérialisée, requête OLAP, optimisation multi requêtes, algorithme k-means.

Remerciement

Je remercie dieu le tout puissant qui m'a donné La volonté et la patience pour Mener à bien mon modeste travail.

Je tiens a remercié Mon encadreur Mme BETOUATI pour m'avoir guidé tout au long de mon travail. Je la remercie pour sa disponibilité, pour ses précieuses orientations, pour sa passion pour la recherche et pour son soutien.

Je tiens également à remercier tout les Membres du jury, qui ont accepté D'examiner ce travail.

Je voudrais adresser mes hommages respectueux à tous les enseignants, qui nous ont dispensé des cours et prodigués des conseils durant mes années d'études.

Je remercie tous mes amies et tous ce qui m'ont aidé.

Enfin et surtout à ma famille et particulièrement à mes parents pour leur soutien sans faille. Rien n'aurait été possible sans eux.

Dédicace

Le présent mémoire est dédié

A ma mère, Que ce travail soit pour toi pour ton aide précieuse pendant toutes ces années

Je dédie ce travail A mon père Youcef A mes frères Mohamed et Abdel kader et A ma chère sœur fatima

A toute ma famille Benyagoub, larbaoui

A mes chers amis et mes proches

A tous ceux qui ont été à mes côtés durant ces années d'étude.

Sommaire

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Liste des tableaux

Liste des abréviations

ED : Entrepôt de Données.

SGBD: Système de Gestion de Base de données.

MOLAP: Multidimensional On-Line Analytical Processing.

ROLAP: Relational On- Line Analytical Processing.

PSV : Problème de Sélection des Vues matérialisées.

MVPP: Multi View Processing Plan.

INTRODUCTION

GENERALE

Introduction Générale

Pour faire face aux nouveaux enjeux, l'entreprise doit collecter, traiter, analyser les informations de son environnement pour anticiper et rester concurrente.

L'entreprise manipule et stocke une multitude de données au sein de plusieurs applications. Ces données se trouvent dans différentes sources hétérogènes, distribuées et autonomes. Elles peuvent aussi être structurées, semi-structurées ou non structurées et stockées sous plusieurs formes : relationnelles, objets, texte, XML, HTML, fichiers, etc... Ces données sont utilisées quotidiennement et sont souvent inappropriées pour des besoins de prise de décision. Il devient alors fondamental de rassembler et d'homogénéiser les données afin de permettre l'analyse des indicateurs pertinents pour faciliter la prise de décision [2].

Les entrepôts de données ont émergé comme étant une solution potentielle répondant aux besoins du stockage et de l'analyse de grands volumes de données, et de pallier les insuffisances des systèmes transactionnels.

Les entrepôts de données permettent, à travers l'analyse de l'activité de l'entreprise, de produire des connaissances rigoureuses et pertinentes qui seront ensuite exploitées par les décideurs ou les scientifiques en vue d'améliorer les performances [3]. L'entrepôt de données supporte un processus d'aide à la décision. Typiquement, ce processus est mené par l'intermédiaire de requêtes. Ces requêtes sont très complexes et demandent un temps d'exécution trop élevé (des heures voire des jours) qui nécessite une optimisation de ces requêtes.

L'optimisation des requêtes est primordiale. Dans la première génération des bases de données, les optimiseurs étaient conçus pour optimiser des requêtes individuelles. Après l'identification des interactions entre les requêtes, des optimiseurs sont proposés pour offrir une optimisation multiple. La difficulté de cette optimisation réside dans l'identification des expressions communes entre les requêtes. Cette interaction a été largement exploitée par les algorithmes d'optimisation des requêtes, connue sous le nom d'optimisation multi-requêtes.

Cette interaction a été utilisée pour définir des méthodes de sélection des vues matérialisées. Ces vues améliorent l'exécution des requêtes, en pré-calculant les opérations les plus coûteuses comme la jointure et l'agrégation, et en stockant leurs résultats dans la base.

Ainsi, l'objectif de notre travail est d'étudier l'utilisation des techniques du datamining comme solution proposée au problème de sélection des vues matérialisées. Nous nous sommes basés sur l'application de la technique du k-means pour partitionner la charge de requête qui est en interaction suivant le nœud de jointure.

Ce mémoire est organisé selon trois chapitres, comme suit :

Le premier chapitre est consacré aux définitions des concepts fondamentaux de l'entrepôt de données, l'historique, l'architecture et le cycle de vie avec les différentes phases de conception d'un entrepôt de données, et aussi aux techniques d'optimisations des requêtes : les techniques redondantes comme : les index, les vues matérialisées et la fragmentation verticale, et les techniques non redondantes comme : la fragmentation horizontale et le traitement parallèle.

Dans notre étude, on s'est focalisé sur les techniques d'optimisations redondantes qui sont les vues matérialisées et spécialement leurs problèmes de sélections.

Le deuxième chapitre expose, en détail, le contexte du problème de sélection des vues matérialisées et nous donnons un aperçu sur les algorithmes et les travaux développés dans ce contexte.

Dans le troisième chapitre nous présentons notre approche qui se base sur l'utilisation de la technique k-means pour identifier les expressions communes entre les requêtes afin de sélectionner des vues à matérialiser, et finalement nous présentons l'implémentation de notre approche.

I. Introduction

Un concept clé dans le monde du décisionnel est l'entrepôt de données. En effet, les analystes du monde de l'informationnel n'ont pas trouvé de mieux pour modéliser de façon unifiée et simple toutes les données de l'entreprise, ainsi actuellement c'est la seule solution.

Un entrepôt est une structure capable de stocker, sous un format précis, toutes les données de l'entreprise en vue d'une utilisation informationnelle.

Nous présentons dans ce chapitre le concept d'entrepôt de données, l'historique, l'architecture et nous détaillons l'ensemble des phases de cycle de vie de conception d'un entrepôt de données et les techniques d'optimisation des requêtes.

II. Entrepôt de données

Un entrepôt de données est un dépôt de multiples sources de données hétérogènes, organisées sous un schéma unifié pour faciliter la gestion de la prise de décision [3].

Le concept de l'entrepôt de données est défini en 1990 par Bill Inmon, père des ED, comme "une collection de données orientées sujet, intégrées, non volatiles et historiées, organisées pour supporter un processus d'aide à la décision"[3].

- **Orientées sujet :** Les données sont structurées par thèmes. L'intérêt d'une telle représentation est de faciliter la réalisation des analyses sur ces différentes activités [2]. L'avantage de cette représentation demeure dans le fait qu'il devient possible de réaliser des analyses sur des sujets transversaux aux structures fonctionnelles et organisationnelles de l'entreprise. Et ainsi, de pouvoir analyser un processus dans le temps à différentes étapes de sa conception au sein du système d'information. Cette orientation permet également de faire des analyses par itération, sujet après sujet [3].
- **Intégrées :** les données proviennent de différentes sources. Pour assurer la cohérence, les données doivent être mise en forme et unifiées avant d'être intégrées au sein de l'entrepôt de données [2].
- **Non volatiles :** L'entrepôt de données veut conserver la traçabilité des informations et des décisions prises. Les données ne peuvent pas être modifiées par l'utilisateur [2].
- **Historisées :** L'entrepôt de données contient des données archivées afin de les utiliser pour les comparaisons, la prévision [3].
- **Organisées :** Les données doivent être agrégées et réorganisées afin de faciliter le processus de prise de décision [3].

Cette définition reflète l'objectif principal d'un entrepôt de données. Il contient des données et les remet aux dirigeants comme connaissances par lesquelles ils peuvent prendre leurs décisions.

III. Historique des entrepôts de données

L'origine du concept entrepôt de données remonte aux années 80, durant lesquelles un intérêt croissant au système décisionnel a vu le jour, dû essentiellement à l'émergence des SGBD relationnel et la simplicité du modèle relationnel et la puissance offerte par le langage SQL [4].

Au début, l'entrepôt de données n'était rien d'autre qu'une copie des données du système opérationnel prise de façon périodique, dédié à un environnement de support à la prise de décision. Ainsi, les données étaient extraites du système opérationnel, stockées dans une nouvelle base de données «concept d'infocentre », le motif principal étant de répondre aux requêtes des décideurs sans pour autant altérer les performances des systèmes opérationnels [4].

L'entrepôt de données, tel qu'on le connaît actuellement, n'est plus vu comme une copie des données du système, il est devenu une nouvelle source d'information, alimentée avec des données recueillies et consolidées des différentes sources internes et externes.

Les principales dates à retenir construisant l'histoire de l'entrepôt de données sont les suivantes :

- Années 1960 - General Mills et l'Université Dartmouth, dans un projet conjoint, créent les termes *faits* et *dimensions*.
- 1983 - Teradata introduit dans sa base de données managériale un système exclusivement destiné à la prise de décision.
- 1988 - Barry Devlin et Paul Murphy publient l'article *Une architecture pour les systèmes d'information financiers* (*An architecture for a business and information systems*) où ils utilisent pour la première fois le terme *Datawarehouse*.
- 1990 - Red Brick Systems crée Red Brick Warehouse, un système spécifiquement dédié à la construction de l'entrepôt de données.
- 1991 - Bill Inmon publie *Building the Data Warehouse* (*Construire l'entrepôt de données*).
- 1995 - Le *Data Warehousing Institute*, une organisation à but lucratif destinée à promouvoir le data warehousing, est fondé.
- 1996 - Ralph Kimball publie *The Data Warehouse Toolkit* (*La boîte à outils de l'entrepôt de données*).

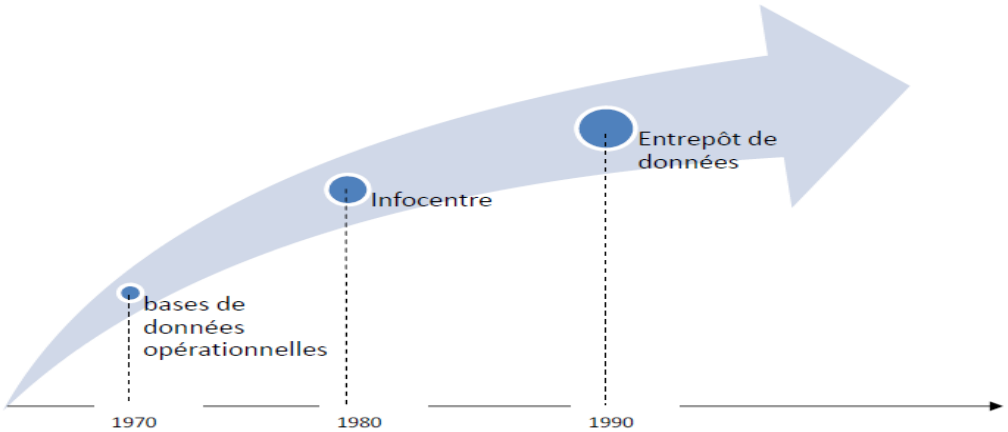

Figure I.1évolution des bases de données décisionnelles [4].

IV. Architecture d'un entrepôt de données

L'entrepôt de données joue un rôle stratégique dans la vie d'une entreprise, permettant le stockage des données essentielles aux besoins de prise de décision en provenance des systèmes opérationnels de l'entreprise et d'autres sources externes [2]. Les différentes phases de la construction d'un entrepôt de données s'organisent en trois opérations principales :

1. *Extraction des données* : Elle consiste à extraire les données provenant de différentes sources. Ces sources peuvent être des bases de données, des fichiers de données, des sources externes à l'entreprise, etc. Les données seront transformées afin d'effectuer un prétraitement pour faciliter l'analyse, ainsi le nettoyage des données est fait : l'homogénéisation, la suppression des doubles, la détection de données non conformes [3].
2. *Organisation et intégration des données dans l'entrepôt* : les données en provenance des différentes bases concernées de l'entreprise sont intégrées et stockées dans la base de données de l'entrepôt en respectant son organisation par sujets [3].
3. *accès aux données intégrées* : cette opération permet [3] :
   - L'analyse et l'exploration des données entreposées.
   - La formulation de requêtes complexes afin de trouver des faits à étudier tels l'analyse en tendance (courbes d'évolution), l'extrapolation et la découverte de connaissance (règles, contraintes, …).

Les métadonnées : C'est un annuaire spécialisé qui conserve toutes les informations utiles sur la création, l'utilisation et la gestion de l'entrepôt [2].

La figure ci-dessous représente l'architecture d'un entrepôt de données [2] :

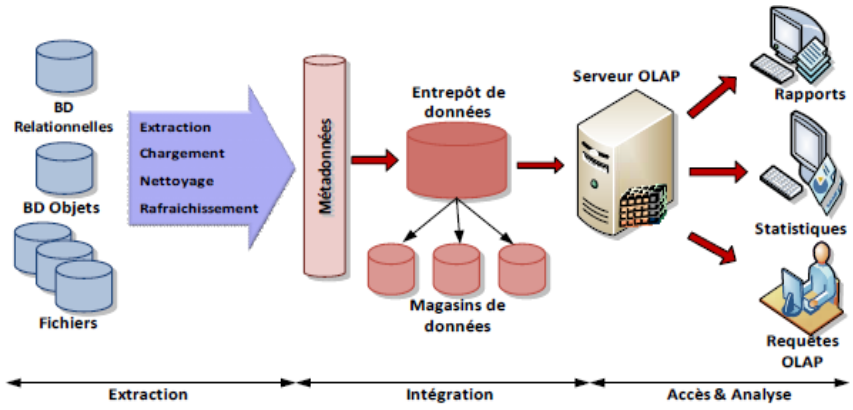

Figure I.2 Architecture d'un entrepôt de données [2].

V. Cycle de vie des entrepôts 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.

- ***La planification*** : Cette phase consiste à déterminer l'étendue du projet ainsi que les buts et objectifs de l'entrepôt à développer, évaluer la faisabilité technique et économique de l'entrepôt et identifier les futurs utilisateurs de l'entrepôt [3].

- ***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 [3].
- ***La maintenance et la gestion de l'évolution*** : Cette phase implique l'optimisation de ses performances périodiquement. L'évolution de l'entrepôt de données concerne la mise à jour de son schéma en fonction des différents changements survenant au niveau des sources ou des besoins des utilisateurs.

Le schéma ci-dessous représente la succession des tâches nécessaires à la mise en place des entrepôts de données efficaces [3].

La deuxième phase de conception comporte actuellement cinq principales phases, constituant le cycle de conception de l'entrepôt de données: l'analyse de besoins, la modélisation conceptuelle, la modélisation logique, le processus d'extraction-transformation-chargement (ETL) et une phase de modélisation physique [5].

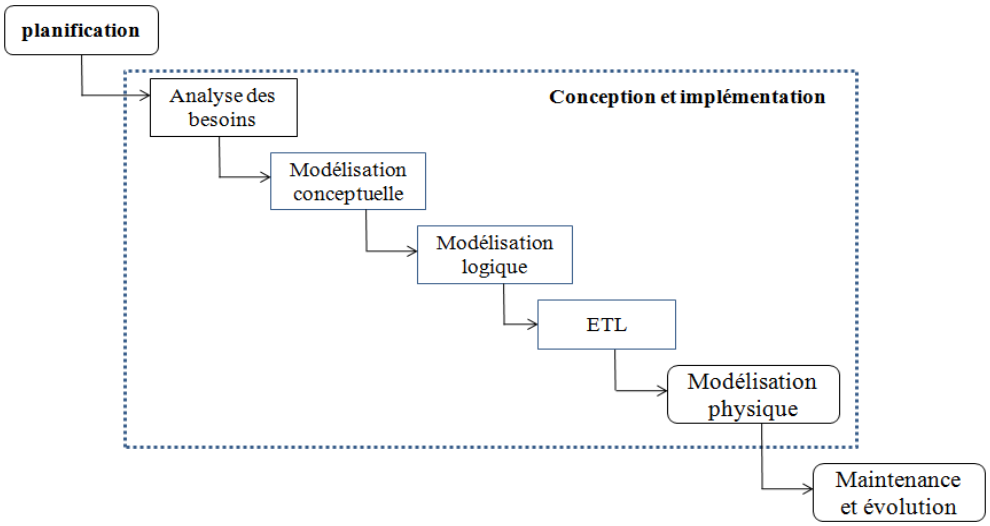

Figure I.3 Cycle de vie des entrepôts de données.

V.1 Analyse des besoins

L'analyse des besoins joue un rôle clé dans tout projet en permettant de réduire le risque d'échec du projet. L'analyse des besoins pour les applications d'ED est définie comme le processus de développement des besoins selon un processus itératif et coopératif d'analyse du problème, de documentation des observations résultantes dans divers formats de représentation et de vérification des résultats obtenus [3].

Deux types de besoins sont distingués : les besoins fonctionnels et les besoins non fonctionnels

- *les besoins fonctionnels :* sont ceux qui caractérisent le système (les besoins en matière de performance, de type de matériel ou de type de conception). Ils peuvent concerner les contraintes d'implémentation (langage de programmation, type SGBD, système d'exploitation, . . . etc) [5].
- *les besoins non fonctionnelles :* Un besoin non fonctionnel est défini comme un attribut ou une contrainte du système comme la flexibilité, la performance, la sécurité, etc.

Deux méthodes sont utilisées pour collecter les besoins : une collecte orientée source et une collecte orientée utilisateur [5].

Les approches considérant une phase de définition des besoins des utilisateurs lors de la conception de l'entrepôt de données sont appelées approches orientées besoins ou approches descendantes.

Les approches orientées sources ou ascendantes se limitent à identifier les besoins de l'entrepôt de données à partir de l'ensemble des données disponibles au niveau des sources.

L'analyse des besoins dans un projet d'entrepôt de données passe par les activités suivantes [5] :

- ✓ *Planning de gestion des besoins :* consiste à :
  - définir les objectifs du projet par les utilisateurs et les concepteurs.
  - définir les règles d'intégration des sources de données.
  - établir un planning de gestion des besoins du projet.
- ✓ *Spécification des besoins* : s'effectue par un processus itératif d'acquisition ou collecte des besoins, ensuite de représentation et spécification des besoins.
- ✓ *Validation des besoins* : consiste à valider les modèles initiaux construits lors de l'étape de spécification des besoins, avec les utilisateurs ainsi qu'avec les sources de données existantes. La validation des besoins est menée par l'équipe de développement, les utilisateurs et les experts du domaine.
- ✓ *Suivi et gestion de l'évolution des besoins* : la gestion doit être effectuée à deux niveaux :
  1. une gestion de l'évolution des besoins des utilisateurs.
  2. une gestion de l'évolution de l'architecture des sources.

V.2 la modélisation conceptuelle

La modélisation conceptuelle consiste à définir le schéma conceptuel de l'entrepôt de données annoté par les concepts multidimensionnels. Leur but est de fournir un modèle conceptuel de l'entrepôt de données fournissant une représentation abstraite de la situation en cours d'étude indépendamment de toute contrainte d'implémentation technique. Ce modèle conceptuel peut être défini à partir du modèle de besoins [5].

Un modèle conceptuel est caractérisé par le domaine concerné, le formalisme utilisé pour modéliser le schéma conceptuel, et le point de vue correspondant aux besoins des utilisateurs. Le modèle conceptuel obtenu doit se conformer à la modélisation multidimensionnelle adaptée aux modèles d'entrepôt de données permettant d'organiser les données entreposées de manière à faciliter leur analyse décisionnelle.

V.3 la Modélisation logique

La modélisation logique de l'entrepôt de données passe par la gestion des trois principaux aspects [5] :

1. *Le suivi des données* : consiste à concevoir le modèle logique cible de l'entrepôt de données répondant aux besoins des utilisateurs, tout en tenant compte des spécificités des données des sources. La conception du modèle logique passe par les tâches suivantes :
    - la traduction du modèle conceptuel multidimensionnel en un modèle logique.
    - L'identification des sources de données candidates pour alimenter le schéma de l'entrepôt de données et pour préparer la prochaine étape ETL.
    - La sélection des sources candidates pour alimenter le modèle de l'entrepôt de données.
    - le mapping entre les données des sources et le schéma logique cible de l'entrepôt de données.
    - l'estimation de la charge de l'entrepôt de données.
2. *Le suivi technologique* : l'application décisionnelle nécessite l'intégration de nombreuses technologies et de gestionnaires de stockage des données. Le suivi technologie consiste à fournir une vue abstraite de l'architecture technique de l'entrepôt de données [5].
3. *Le suivi de l'application décisionnelle* : consiste en l'identification de l'application décisionnelle supportant l'entrepôt de données et passe par le développement des rapports normalisés, des requêtes paramétrées, des tableaux de bord, des modèles analytiques, des applications d'exploration de données ainsi que les interfaces associées à la navigation [5].

En sortie, la phase de modélisation logique doit fournir le modèle logique de l'entrepôt de données, adapté aux particularités du modèle d'implémentation logique choisi, nous détaillons ce dernier dans la section VI.

V.4 Phase ETL (Extract-Transform-Load)

Cette phase consiste à extraire les données à partir des sources sélectionnées et à effectuer les transformations nécessaires pour assurer le chargement des données des sources au niveau du schéma cible de l'entrepôt.

Les données sont extraites à partir des sources de données hétérogènes. Elles sont ensuite propagées dans une zone de stockage temporaire où auront lieu leur transformation, homogénéisation et nettoyage. Enfin, les données sont chargées dans l'entrepôt de données cible [5].

V.5 Modélisation physique

Cette étape consiste à implémenter physiquement le modèle logique de l'entrepôt de données et à spécifier les techniques et les schémas d'optimisation de l'entrepôt, accompagné d'une spécification détaillée des caractéristiques physiques de l'entrepôt de données (les types de données, la segmentation des tables, les paramètres de stockage, la spécification des clés des tables, la gestion des instances). Le choix et l'application des techniques d'optimisation se fait également lors de la phase physique [5].

V.5.1 Les techniques d'optimisation

Afin d'adopter une politique d'optimisation d'un entrepôt de données, il faut choisir la structure d'optimisation, la nature de sélection et l'algorithme de sélection [6]. Ces techniques appartiennent à deux catégories [6]:

- ✓ ***Redondantes :*** Les techniques redondantes sont des techniques d'optimisation des requêtes qui nécessitent un stockage physique des données et une maintenance dont l'utilisation produit une duplication des données. On distingue : les index, les vues matérialisées et la fragmentation verticale. Dans ce projet nous nous intéressons aux structures d'optimisation redondantes.
  - **Les index :** Les techniques d'indexation ont été largement étudiées et constituent une option très importante pour la phase de conception physique des bases de données traditionnelles et avancées [1], [3].

    Les index sont utilisés pour améliorer le temps d'accès aux données. La définition des index se fait souvent pendant l'exploitation de la base de données. Certaines techniques d'indexation sont apparues dans le contexte d'entrepôts de données, vu la nature de requêtes OLAP. Ce sont par exemple les index binaires, les index de jointures binaires, les index de jointure en étoile, etc [1], [3].
  - **Les vues matérialisées :** Une vue est une requête nommée. Elle est dite matérialisée si son résultat est stocké physiquement. Les vues améliorent l'exécution des requêtes en pré-calculant les opérations les plus coûteuses comme la jointure et l'agrégation, et en stockant leurs résultats dans la base. Dans cette situation, certaines requêtes nécessitent seulement l'accès aux vues matérialisées et par conséquent sont exécutées plus rapidement [1], [3].

    Dans le deuxième chapitre on va détailler le concept de vue matérialisée, et en présente leur problème
  - **La fragmentation verticale.** La fragmentation verticale consiste à diviser une relation en sous relations appelées fragments verticaux résultant de l'application de l'opération de projection. La fragmentation verticale favorise le traitement des requêtes de projection portant sur les attributs utilisés dans le processus de la fragmentation, en limitant le nombre de fragments auxquels accéder. Son inconvénient est qu'elle requiert des jointures supplémentaires lorsqu'une requête accède à plusieurs fragments [1], [3].
- ✓ ***Non redondantes :*** Les techniques non redondantes sont des techniques d'optimisation des requêtes qui ne nécessitent pas un stockage physiques des données, ces techniques ne dupliquent pas les données mais réorganisent leur représentation physique. On distingue : la fragmentation horizontale et le traitement parallèle.
  - **La fragmentation horizontale :** La fragmentation horizontale consiste à diviser un objet de la base de données (table, index, vues) en sous-ensembles de lignes appelés fragments horizontaux résultant de l'application de l'opération de restriction. La reconstruction de l'objet à partir de ces fragments horizontaux est obtenue par l'opération d'union de ces fragments [1], [2], [3].
  - **Le traitement parallèle :** le traitement parallèle permet d'exécuter les opérations en parallèle pour un gain de temps de traitement.

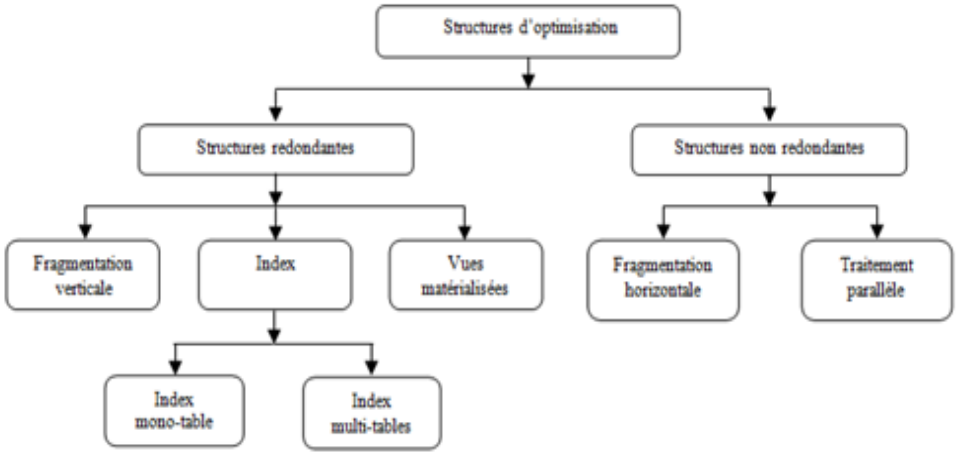

Figure I.4 Classification des structures d'optimisation.

VI. Modélisation d'un entrepôt de données

L'organisation des données dans l'entrepôt étant nécessaire voire obligatoire car facilitant leurs exploitations ainsi que leurs analyses par les décideurs. Les données manipulées dans le contexte d'entrepôts de données sont représentées sous forme multidimensionnelle qui est mieux adaptée pour le support des processus d'analyse [2]. Ce modèle permet d'observer les données selon plusieurs axes d'analyse et facilite l'accès aux données [2], [3].

L'objectif majeur de cette modélisation est la vision multidimensionnelle des données, ce qui est assuré via le concept de cube de données [1], [2], [3].

- **Le cube de données**

Le cube de données offre une abstraction très proche de la façon dont l'analyste voit et interroge les données. Il organise les données en une ou plusieurs dimensions qui déterminent une mesure d'intérêt. Une dimension spécifie la manière dont on regarde les données pour les analyser, alors qu'une mesure est un objet d'analyse.

Chaque dimension est formée par un ensemble d'attributs et chaque attribut peut prendre différentes valeurs. Les dimensions possèdent en général des hiérarchies associées qui organisent les attributs à différents niveaux pour observer les données à différentes granularités. Une dimension peut avoir plusieurs hiérarchies associées, chacune spécifiant différentes relations d'ordre entre ses attributs [1].

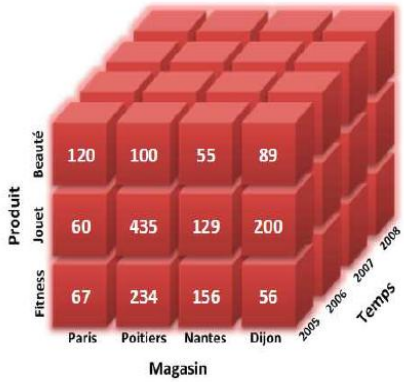

Figure I.5 Un exemple de cube de données [2].

Deux concepts fondamentaux caractérisent le modèle multidimensionnel : faits et dimensions.

- ✓ **Un fait** représente le sujet analysé, il est formé de mesures correspondantes aux informations de l'activité étudiée [1], [2]. Ces mesures sont calculables à partir d'un grand nombre d'enregistrements.

Table de faits des ventes journalières
Clé date (CE)
Clé produit (CE)
Clé magasin (CE)
Quantité vendue
Montant des ventes (€)

Figure I.6 Modèle d'une table d'une table de faits

- ✓ **Une dimension** représente l'axe d'analyse de l'activité. Elle regroupe des paramètres et des informations qui peuvent influencer sur les mesures d'activités du fait. Les dimensions possèdent des hiérarchies permettant de faire l'analyse d'un niveau faible de détail vers un niveau plus détaillé [2].

Tables de dimension "Produit"
Clé produit (CP)
Description du produit
Numéro US (clé naturelle)
Description de la marque
Description de la catégorie
Description du rayon
Description du type d'emballage
… et bien d'autre attributs

Figure I.7 Modèle d'une table de dimension

VI.1 Les implémentations des modèles multidimensionnels

L'implémentation du modèle multidimensionnel sur un SGBD réel peut se faire selon trois modèles : MOLAP (Multidimensional On-Line Analytical Processing), ROLAP (Relational On- Line Analytical Processing) et HOLAP (Hybrid On-Line Analytical Processing).

VI.1.1 Les systèmes MOLAP

Les systèmes de type MOLAP 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'avantage de cette implémentation est le gain considérable en temps d'exécution des requêtes, vu l'accès direct aux données, mais elle présente certaines limites telles que :

- le besoin de redéfinir des opérations pour manipuler les structures multidimensionnelles.
- la difficulté de la mise à jour et de la gestion du modèle.
- la consommation de l'espace lorsque les données sont éparses, ce qui nécessite l'utilisation des techniques de compression [1], [2], [3].

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

Figure I.8 Représentation sous forme de tableau multidimensionnel

VI.1.2 Les systèmes ROLAP

Les systèmes de type ROLAP utilisent une représentation relationnelle du cube de données. Chaque fait correspond à une table de faits et chaque dimension correspond à une table de dimensions. La table de faits possède comme attributs les mesures d'activités et les clés étrangères vers les tables de dimension. L'avantage de ce modèle est l'utilisation des systèmes de gestion de bases de données existants, ce qui réduit le coût de mise en œuvre [2].
Trois modèles sont utilisés pour la modélisation des systèmes ROLAP, le schéma en étoile, le schéma en flocon de neige et le schéma en constellation.

- ***Schéma en étoile :*** La table de faits référencie les tables de dimensions en utilisant une clé étrangère pour chacune d'elles et stocke les valeurs des mesures pour chaque combinaison de clés. Autour de cette table de faits figurent les tables de dimensions qui regroupent les caractéristiques des dimensions [1].

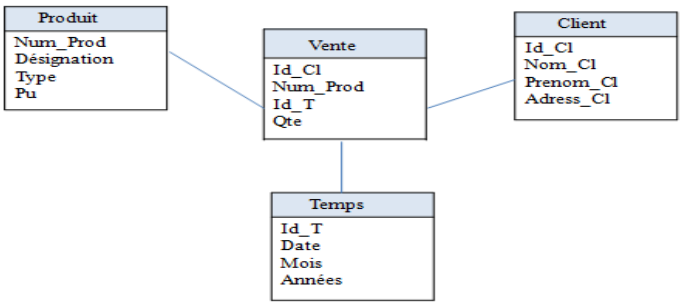

Figure I.9 Exemple d'un schéma en étoile

- ***Schéma en neige :*** est un raffinement du schéma en étoile. Certaines tables de dimensions sont normalisées selon leur hiérarchie en donnant lieu à de nouvelles tables [3].

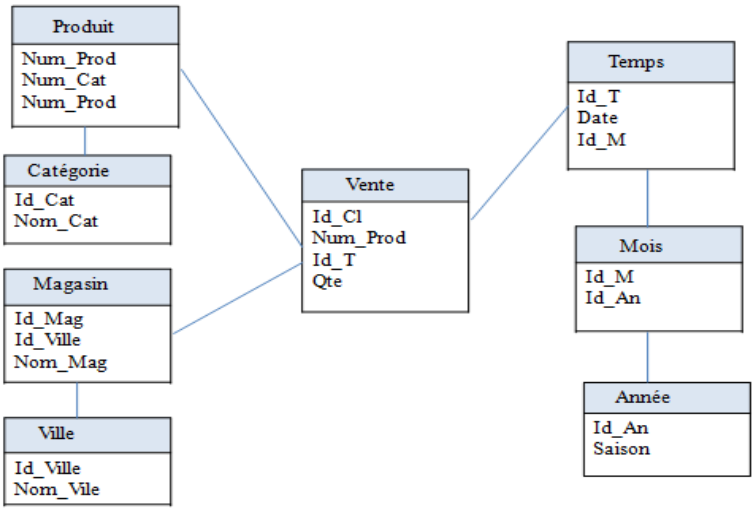

Figure I.10 Exemple d'un schéma en flocon de neige

- ***Schéma en constellation :*** est un regroupement de schémas en étoile qui partagent des dimensions. Il met en avant la relation entre les faits.

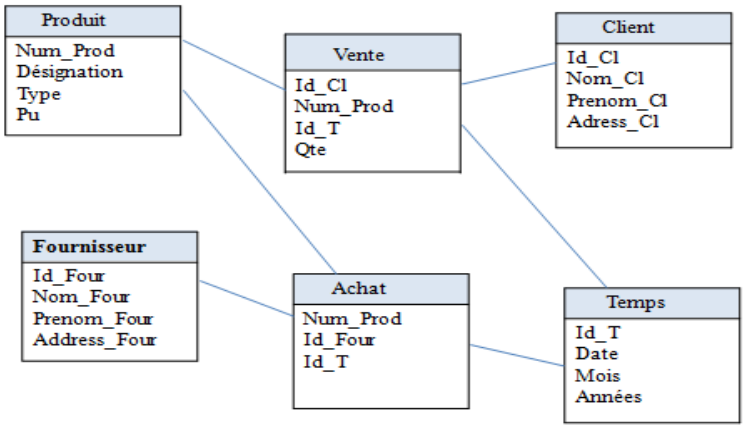

Figure I.11 Exemple d'un schéma en constellation

VI.1.3 Les système HOLAP

Le système HOLAP (Hybrid On-Line Analytical Processing) représente les données fréquemment utilisées (généralement les données agrégées) dans un tableau multidimensionnel, et représente les données non fréquemment utilisées dans un schéma relationnel. Ceci afin de bénéficier des avantages des deux systèmes cités précédemment. La séparation des données doit être transparente à l'utilisateur final [5].

Ci-dessous une liste des caractéristiques principales qu'un système HOLAP doit fournir [5]:

- ***La transparence du système :*** Pour la localisation et l'accès aux données, sans connaître si elles sont stockées dans un SGBD relationnel ou dimensionnel.
- ***Un modèle de données générales et un schéma multidimensionnel global :*** Pour aboutir à la transparence du premier point, tant le modèle de données général que le langage de requête uniforme doit être fournis. Etant donné qu'il n'existe pas un modèle standard, cette condition est difficile à réaliser.
- ***Une allocation optimale dans le système de stockage :*** Le système HOLAP doit bénéficier des stratégies d'allocation qui existent dans les systèmes distribués tels que : le profil de requêtes, le temps d'accès, l'équilibrage de chargement.

VI.1.4 Comparaison entre ROLAP, MOLAP et HOLAP

Le tableau suivant propose une comparaison entre les systèmes ROLAP, MOLAP et HOLAP.

Tableau I.1 Comparaison ROLAP, MOLAP et HOLAP [5].

stockage	Avantages	Inconvénients
ROLAP	Standard, technologie prouvée et familière	Lenteur du traitement des requêtes
	Scalable (capacité d'expansion élevée)	Conception basée sur les diagrammes EA pas toujours appropriée pour les systèmes d'aide à la décision
MOLAP	Modèle multidimensionnel (calcul automatique des agrégations des données)	Redondance des données
	Traitement de requête spécialisé et rapide	Non scalable
	Techniques d'indexation spécialisées	Consommation de l'espace lorsque les données sont éparses
HOLAP	Accès rapide pour différents niveaux d'agrégation	Système complexe (un serveur HOLAP doit supporter des outils MOLAP et ROLAP)
	Stockage compact des agrégations	Charge importante (entre le stockage et les techniques d'optimisation ROLAP et les outils MOLAP)

VII. Conclusion

Nous avons vu que les entrepôts de données représentent le cœur des systèmes décisionnels de l'entreprise. C'est une structure qui à pour but de regrouper les données de l'entreprise à des fins analytiques et pour aider à des prises de décisions stratégiques.

Dans ce chapitre nous avons présenté les concepts de base de l'entrepôt de données, l'architecture des ED, ainsi que la modélisation multidimensionnelle qui permet de faire la conception d'un entrepôt de données. Sachant que les entrepôts de données sont des bases de données larges, la manipulation des requêtes décisionnelles est rendue pénible puisque celles ci passent par une table de fait très volumineuse. À cet effet des techniques d'optimisations ont été développées pour dépasser cet inconvénient.

Dans notre projet nous nous concentrons à la technique des vues matérialisées, Nous allons décrire celle-ci dans le chapitre suivant.

I. Introduction

Les entrepôts de données sont dédiés aux applications d'analyse et de prise de décision. Le processus d'analyse est réalisé à l'aide de requêtes complexes comportant de multiples jointures et des opérations d'agrégation sur des tables volumineuses.

Dans ce chapitre, nous allons présenter l'optimisation des requêtes, les types d'optimiseurs qui existent, le problème de sélection des vues matérialisée et à la fin quelques travaux développés pour ce problème.

II. Optimisation des requêtes

L'optimisation des requêtes est une tâche importante pour tout SGBD relationnel. Elle consiste à trouver une meilleure stratégie d'exécution d'une requête donnée se trouvant dans un ensemble de solutions.

L'optimiseur de requêtes a deux tâches principales à réaliser [7]:

- ✓ Trouver l'ensemble des plans d'exécution pour une même requête donnée.
- ✓ Choisir parmi ces alternatives des plans les plus performantes pour exécuter la requête en se basant sur certains critères.

Il existe deux types d'optimiseurs, qui sont démontrés dans le schéma si dessous :

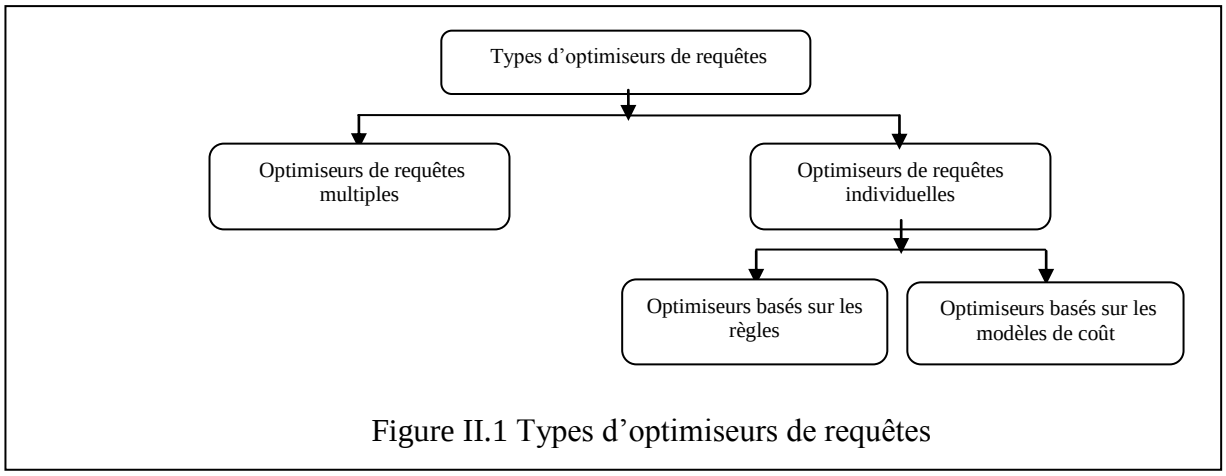

Figure II.1 Types d'optimiseurs de requêtes

Un optimiseur a fondamentalement trois composants :

1- L'espace de recherche

L'espace de recherche est définit comme étant le nombre de plans alternatifs pour une requête. Ces plans sont équivalents en ce qui concerne la production des mêmes résultats, ils diffèrent selon l'ordre d'exécution des opérations et la manière de leurs implémentations [8].

Une requête donnée est représentée par un arbre de jointure dans un espace de recherche en utilisant les règles de transformation. Si cette requête comporte plusieurs opérateurs et plusieurs relations, alors l'espace de recherche deviendra très grand car contenant un grand nombre d'arbres équivalents [8].

L'exploration d'un grand espace de recherche peut augmenter le temps et le coût d'optimisation, par conséquent l'optimisation de requête impose une certaine restriction à la taille de l'espace de recherche. La réduction de la taille de cet espace est imposée à la forme d'un arbre de requête (arbre gauche-profond, arbre droit-profond et arbre touffu) [8].

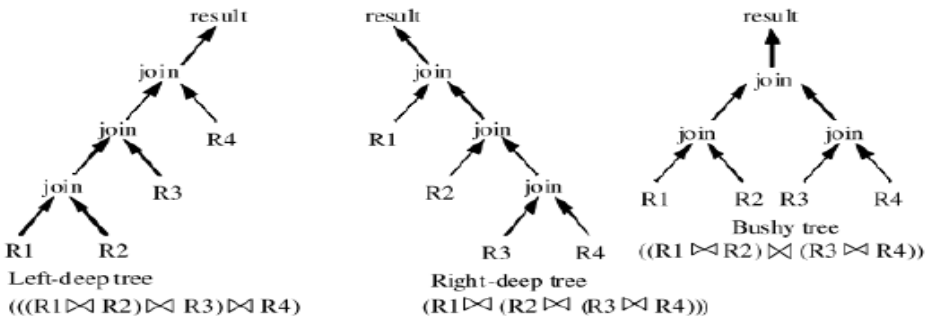

Figure II. 2 Les formes d'arbres de jointure [8].

2- La stratégie de recherche

Le problème de trouver un arbre de jointure optimal dans un ensemble alternatif de solutions est résolu à l'aide de plusieurs stratégies de recherche. La stratégie de recherche explore l'espace de recherche pour trouver le meilleur plan d'exécution, nous distinguons trois classes de stratégies d'optimisation de l'ordre de jointure qui sont les algorithmes déterministes, randomisés, et génétiques [8].

- **Stratégies déterministes :** pour une requête donnée, l'ensemble de tous les plans possibles d'exécution est énuméré, elles établissent des plans d'exécution à partir des sous-plans déjà optimisés en commençant par une partie ou l'ensemble des relations de base d'une requête. Pour réduire le coût d'optimisation, des plans partiels qui ne sont pas susceptibles de mener au plan optimal sont élagués aussitôt que possible, et les meilleurs plans sont employés pour construire le plan complet [8].
- **Stratégies randomisées :** Les stratégies déterministes sont insatisfaisantes pour optimiser des requêtes complexes, parce que le nombre des plans d'exécution devient rapidement trop grand. Pour résoudre ce problème, des stratégies aléatoires sont employées. Les algorithmes randomisés effectuent habituellement les marches aléatoires dans l'espace de solution par l'intermédiaire d'une série de mouvements. Les genres de mouvements qui sont considérés dépendent de l'espace de solution [8] :
  - ✓ Si des arbres de traitement gauche-profonds sont employés, chaque solution peut être représentée uniquement par une liste numérotée de relations participantes à la jointure [8].
  - ✓ Si c'est les arbres de traitement touffus, les mouvements suivants sont employés : le choix de la méthode de jointure, jointure associative/commutative [8].

- **Algorithme génétique :** L'Algorithme Génétique est un algorithme de recherche et d'optimisation qui suit le processus évolutionnaire normal selon lequel l'organisme s'adapte aux changements de l'environnement, l'algorithme génétique est principalement définit par : le codage de schéma (chromosome), la fonction fitness (objectif), la sélection du parent, et les opérateurs génétiques (croisement, mutation) [8].

3- le modèle de coût

Le modèle de coût est employé pour prévoir le coût de chaque plan. Ce modèle se compose de différents types d'éléments, comme le coût de stockage secondaire, le coût de stockage de mémoire et le coût de calcul [8].

III. Optimisation multi requêtes

Les optimiseurs étaient conçus pour optimiser des requêtes individuelles. Après l'identification des interactions entre les requêtes, des optimiseurs étaient proposés pour offrir une optimisation multi requêtes. Cette optimisation concerne un ensemble de requêtes en interaction, la difficulté dans ce type d'optimisation est l'identification des expressions communes entre les requêtes [7].

L'optimisation multi requêtes est souvent réalisée à l'aide des arbres algébriques, ainsi chaque arbre représente une requête. La fusion des arbres donne lieu à un graphe global appelé *plan multiple d'exécution des vues*.

III.1 Le plan multiple d'exécution des vues (MVPP : Muli-Views Processing Plan)

Le MVPP est une représentation graphique d'une charge de travail proposée dans le cadre de l'optimisation multi requêtes, c'est un plan global unifiant les plans individuels d'exécution des requêtes.

La difficulté principale liée à la génération de ce plan est l'obtention d'un plan global contenant les expressions communes entre les requêtes, surtout lorsqu'il y a une charge de requêtes très importantes à optimiser [7].

Le MVPP à plusieurs niveaux [4] :

- Le niveau 0 : les feuilles représentent les tables de base de l'entrepôt.
- Le niveau 1 : les nœuds représentent les résultats des opérations algébriques de sélection et de projection.
- Le niveau 2 : les nœuds représentent les opérations ensemblistes comme la jointure, l'union, etc.
- Le dernier niveau : représente les résultats de chaque requête.

IV. Les vues matérialisées

Les vues matérialisées sont définies dans [1], [2], [9] comme suite : "Une vue est une requête nommée. Elle est dite matérialisée si son résultat est stocké physiquement. Les vues améliorent l'exécution des requêtes en pré calculant les opérations les plus coûteuses comme la jointure et l'agrégation, et en stockant leurs résultats dans la base ".

Dans ce cas, les requêtes sont exécutées rapidement car elles nécessitent seulement l'accès aux vues matérialisées.

Les vues peuvent être utilisées pour satisfaire plusieurs objectifs, comme l'amélioration de la performance des requêtes ou la fourniture des données dupliquées.
Les vues sont associées à trois problèmes majeurs [9] :

- le problème de sélection des vues matérialisées.
- le problème de maintenance des vues matérialisées.
- la réécriture des requêtes en fonction des vues.

IV.1 la sélection des vues matérialisées

La sélection des vues matérialisées consiste à choisir un sous-ensemble de vues candidates permettant de réduire le coût d'exécution d'une charge de requêtes. La sélection des vues peut être effectuée sous certaines contraintes, généralement un quota d'espace et/ou un seuil de temps de maintenance à ne pas dépasser. Notre travail s'inscrit dans ce cadre.

Le problème de sélection de vues (PSV) consiste à trouver un ensemble de vues, qui permet de réduire le coût total d'exécution des requêtes et qui respecte des contraintes comme les contraintes de stockage, de maintenance, du temps d'exécution, etc. Deux types de PSV ont été considérés : le PSV statique et le PSV dynamique [1], [2], [9] :

- ***Le PSV statique*** consiste à sélectionner un ensemble de vues à matérialiser afin de minimiser le coût total d'évaluation de ces requêtes, le coût de maintenance ou les deux, sous la contrainte de la ressource. Le problème suppose donc que l'ensemble des requêtes n'évolue pas. Si des évolutions sont enregistrées dans des requêtes, alors il est nécessaire de reconsidérer totalement le problème (en reconstruisant les vues à matérialiser) [1], [2], [9].
- ***Le PSV dynamique*** considère que l'ensemble des requêtes évolue dans le temps. Dans ce cas, l'algorithme dynamique peut être amené à modifier l'ensemble des vues matérialisées en fonction de nouvelles requêtes. En effet, de nouvelles vues peuvent être insérer et d'autres seront supprimées [1], [2], [9].

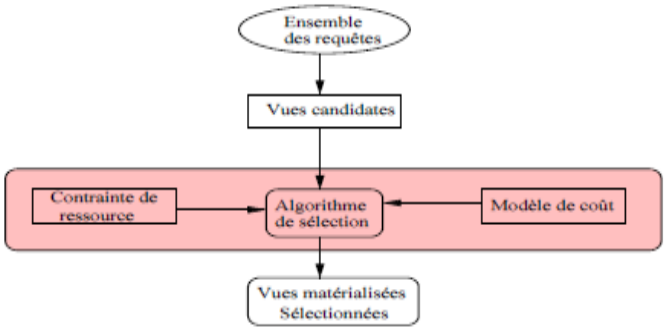

Figure II.3 Le processus de sélection des vues matérialisées [1].

On peut formaliser le problème de sélection des vues matérialisées comme suit :

Input :

- Un entrepôt de données (Data Warehouse DW).
- Une charge de requête (workload).

Out put :

- La sélection des vues matérialisées.

Objectif :

- Minimiser le coût d'exécution d'une charge de requêtes.

IV.2 La maintenance des vues matérialisées

La maintenance des vues matérialisées consiste à reporter les modifications survenues sur les tables de base à leur niveau. Cela peut se faire selon trois approches [1], [2], [9] :

1. **Périodiques** : les vues sont mises à jour continuellement à des périodes précises.
2. **Immédiates** : les vues sont mises à jour immédiatement à la fin de chaque transaction.
3. **Différée** : les modifications sont propagées d'une manière différée. Dans ce cas, une vue est mise à jour uniquement au moment où elle est utilisée par une requête d'un utilisateur.

La maintenance des vues peut être effectuée en recalculant ces vues à partir des tables de base. Cependant, cette approche est complètement inefficace car elle est très coûteuse. En effet, une bonne maintenance des vues est réalisée lorsque les changements (insertions, suppressions, modifications) effectués dans les tables sources peuvent être propagés aux vues sans être dans l'obligation de recalculer intégralement leur contenu [1], [2], [9].

Pour résoudre ce problème, trois types de maintenances ont été proposées [1], [2], [9]:

1. ***La maintenance incrémentale :*** consiste à identifier le nouvel ensemble de n-uplets à ajouter à la vue dans le cas d'une insertion, ou le sous-ensemble de n-uplets à retirer de la vue dans le cas d'une suppression, sans réévaluer intégralement la vue.
2. ***La maintenance autonome :*** assure que la maintenance d'une vue V peut être calculée uniquement à partir de V et des changements survenus sur les tables de bases sur lesquelles elle est définie.
3. ***La maintenance en batch :*** est effectuée en utilisant des transactions de mise à jour. Une transaction de maintenance est relativement longue et peut donc interrompre l'usage de l'entrepôt. Par conséquent, elle est exécutée souvent durant les périodes d'activité creuse (la nuit par exemple). Cette maintenance n'est pas souhaitable, car avec l'émergence d'internet, un entrepôt doit être opérationnel continuellement.

IV.3 La Réécriture des requêtes en fonction des vues matérialisées

Après le processus de sélection des vues, toutes les requêtes définies sur l'entrepôt doivent être réécrites en fonction des vues. La sélection de la meilleure réécriture pour une requête est une tâche difficile. Le processus de réécriture de requêtes est supporté par la plupart des SGBD multidimensionnels. Le processus de réécriture des requêtes a été utilisé comme une technique d'optimisation pour réduire le coût d'évaluation d'une requête [9].

V. Les travaux existants

Pour la résolution du problème de sélection des vues, plusieurs algorithmes sont proposés et peuvent être classés en trois catégories, en fonction du type de contrainte qu'ils utilisent [10] :

1. Algorithmes sans contrainte.
2. Algorithmes dirigés par la contrainte d'espace.
3. Algorithmes dirigés par la contrainte du temps total de maintenance des vues.

V.1 Algorithmes sans contrainte

Il existe plusieurs travaux concernant la sélection des vues sans contrainte. Parmi ces travaux ceux de **« yang et al »** et les travaux de **« Ahcène Boukorca et al ».**

V.1.1 les travaux de « yang et al »

Leurs travaux sont présentés dans le contexte des entrepôts de données sur la génération de plan unifié de requêtes afin de matérialiser les nœuds intermédiaires. Les auteurs ont développé un algorithme de sélection des vues matérialisées qui consiste à représenter chaque requête par un arbre algébrique. Chaque requête peut avoir plusieurs arbres algébriques.

Les auteurs sélectionnent l'arbre optimal en fonction d'un modèle de coût, ensuite l'algorithme essaye de trouver les expressions communes entre ces arbres. Finalement, ces arbres sont fusionnés en un seul graphe, appelé plan multiple d'exécution des vues MVPP [11].

Ce graphe est utilisé pour rechercher l'ensemble des vues dont la matérialisation minimise la somme des coûts d'évaluation des requêtes et de maintenance des vues [3]. Deux algorithmes ont été proposés dans ce contexte.

Exemple : Soit un schéma d'un entrepôt ayant cinq tables et sur lequel trois requêtes sont définies, La Figure ci-dessous montre que les requêtes Q1 et Q2 ont une expression commune. Ces nœuds sont de bons candidats pour la matérialisation.

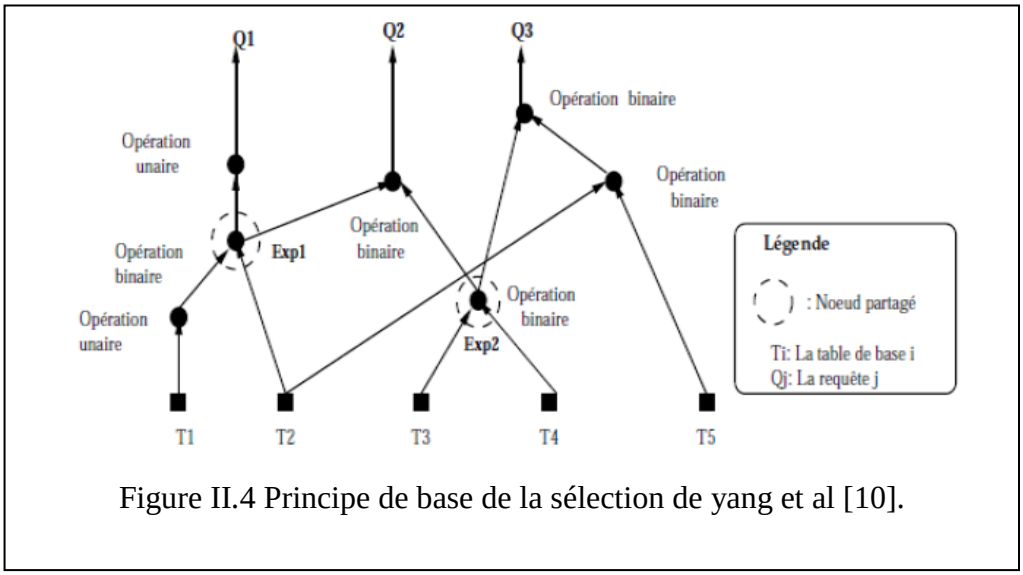

Figure II.4 Principe de base de la sélection de yang et al [10].

1. Algorithme « A feasible Solution »

Afin de réduire l'espace de recherche, l'algorithme devra commencer par les plans individuels optimaux des requêtes, ensuite les ordonner à la base de la fréquence d'accès au traitement des requêtes multiplié par le coût de requête [11].

Une fois l'ordre des plans de requête optimal est fixé, l'algorithme prend le premier plan optimal, et intègre le deuxième en utilisant les expressions communes. Ensuite les deux premiers plans sont fusionné, et ainsi de suite. Cette procédure sera répétée jusqu'à ce que tous les plans aient été pris en considération [11]. Cet algorithme est très coûteux en terme de calcul et ne donne pas forcement le MVPP optimal [7].

2. Algorithme « la programmation binaire 0-1 »

L'algorithme passe par les étapes suivantes [7] :

1. Générer tous les plans possibles *pi* pour chaque requête.
2. Identifier tous les sous arbres de jointures possibles *si* pour chaque plan généré.
3. Construction d'une matrice d'usage binaire *A,* où le coefficient *aij* représente la possibilité ou non d'exécution de la requête *qi* par le plan *pj*.
4. Construction d'une matrice binaire *B,* où le coefficient *bij* représente l'appartenance ou non du sous-arbre de jointure *sj* au plan *pi*.

Finalement, le problème de sélection de MVPP optimal est réduit à la sélection d'un ensemble de plans individuels qui minimise le coût d'exécution de la charge de requêtes *Couttotal.*

$Couttotal = \Sigma i=1 \; \Sigma j=1 \; Ecost(sj)bij$

où *Ecost*(*si*) représente le coût de construction de sous-arbre *si*.

V.1.2 Les travaux de « Ahcène Boukorca et al »

Les auteurs considèrent un MVPP comme un circuit électronique. L'idée pour la construction du MVPP consiste à réordonner les nœuds de jointure entre les différentes requêtes afin de maximiser la réutilisation des résultats intermédiaires lors de l'exécution des requêtes. La technique utilisée est basé sur la représentation des opérations de jointure par un hypergraphe de jointure, puis son partitionnement en plusieurs hypergraphes disjoints et enfin la transformation de l'hypergraphe en graphe orienté optimal [7].

La première étape est la construction de l'hypergraphe avec l'utilisation des algorithmes de la théorie de graphe qui ont prouvés leurs efficacités. Cette étape consiste à extraire tous les nœuds de sélection, de jointure, de projection et d'agrégation selon un ordre déterminé par la forme générale de la requête, et regroupe les nœuds de jointure en plusieurs composantes connexes et ensuite les représenter par un hypergraphe qui contient l'ensemble des sommets (Nœuds de jointure), et un ensemble des hyper arêtes qui représente la charge des requêtes [7].

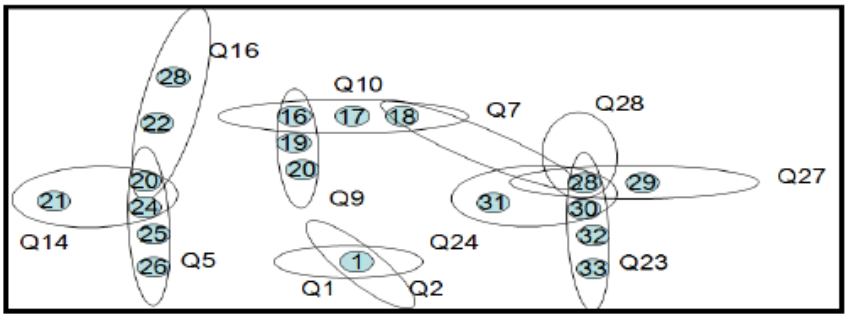

Figure II.5 Exemple d'un Hypergraphe de jointure [7].

L'étape suivante c'est le partitionnement de l'hypergraphe à l'aide des programmes de *HMETIS* pour diviser l'hypergraphe en plusieurs hypergraphes de telle sorte que le nombre d'hyper arêtes coupées soit minimal [7].

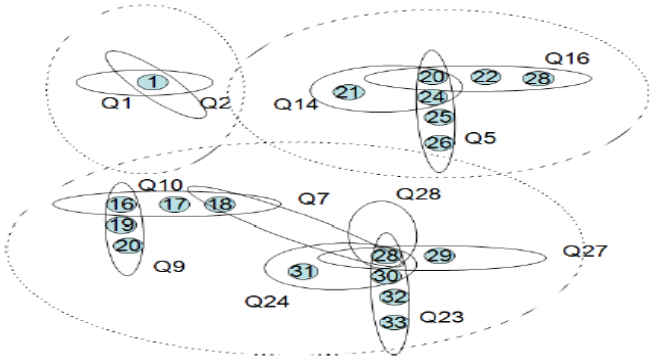

Figure II.6 Partitionnement d'un Hypergraphe de jointure [7].

À la fin, la transformation de l'hypergraphe en un graphe orienté pour construire le MVPP des requêtes. Cette étape consiste à utiliser trois algorithmes, dont chacun a une tâche précise. Le premier algorithme cherche le nœud pivot correspondant à la première opération de jointure, ce nœud va maximiser le bénéfice de réutilisation de son résultat par les différentes requêtes. Le deuxième algorithme concernant la transformation de l'hypergraphe en un graphe orienté. Le troisième algorithme permettant le fractionnement d'un hypergraphe en deux hypergraphes suivant un nœud *pivot*, et de dupliquer les sommets communs entre les deux hypergraphes [7].

V.2 Algorithmes dirigés par la contrainte d'espace

Dans les travaux de Harinarayan et al [10], les auteurs proposent un algorithme glouton pour résoudre le problème de sélection des vues matérialisées. L'algorithme est utilisé pour sélectionner un ensemble de cellules dans un cube de données.

L'objectif de cet algorithme est de minimiser le temps de réponse des requêtes sous la contrainte pour que la taille des vues sélectionnées ne dépasse pas la capacité d'espace S. Ils se sont intéressés seulement aux requêtes ayant des fonctions d'agrégation.

Les auteurs ont modélisé le problème sous la forme d'un treillis de vues. Ils ont développé un modèle de coût afin d'estimer le coût de chaque vue du treillis.

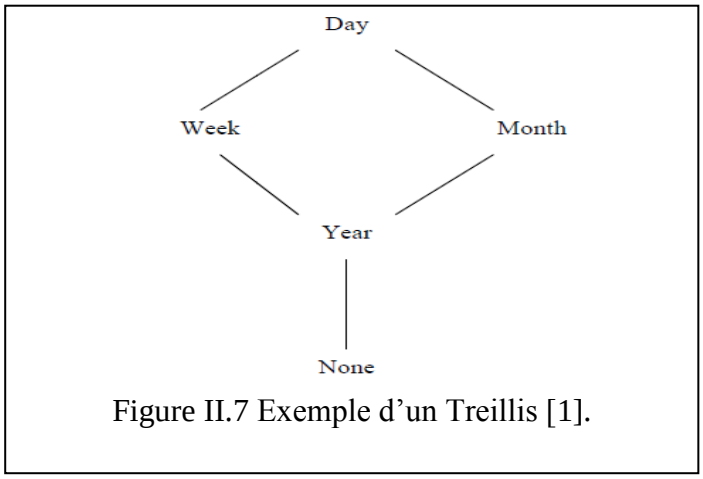

Figure II.7 Exemple d'un Treillis [1].

Ce coût est basé sur le principe suivant [10]:

- Pour évaluer une requête Q (vue), ils ont choisi une vue ancêtre de Q (QA), qui a été matérialisée.
- Le coût d'évaluation de la requête Q est le nombre de n-uplets présent dans la table correspondante à QA.
- Chaque vue est associée à un coût de stockage correspondant au nombre de n-uplets de cette vue.

L'algorithme de sélection des vues est décrit de la façon suivante :

Soit S l'ensemble des vues matérialisées à l'étape j de l'algorithme. Pour chaque vue v, un bénéfice dénoté par B (v; S) et défini comme suit :

- Pour chaque relation de dépendance $w \leq v$, définir une quantité Bw par :
  1. Soit u la vue ayant le plus petit coût dans S tel que $w \leq u$
  2. Si $C(v) < C(u)$, alors $Bw = C(u) - C(v)$, sinon, $Bw = 0$
- $B(v, S) = \Sigma\, w \leq v\, Bw$

En d'autres termes, le bénéfice de V est calculé en considérant sa contribution dans la réduction du coût d'évaluation des requêtes. Pour chaque vue w couvrant v, ils ont calculé le coût d'évaluation w en utilisant v et d'autres vues dans S offrant un coût moins élevée pour évaluer w. Si la présence de la vue v est inférieure au coût de w, alors la différence représente une partie du bénéfice de la matérialisation de la vue v.

Le bénéfice total B(v; S) est la somme de tous les bénéfices en utilisant v pour évaluer w, et fournissent un bénéfice positif.

Les auteurs montrent que cet algorithme présente des performances très proches de l'optimum. Cependant, il parcourt l'espace des solutions possibles à un niveau élevé de granularité et peut éventuellement laisser échapper de bonnes solutions [13].

V.3 Algorithmes dirigés par le temps de maintenance

Ce type d'algorithme a été étudié par Gupta et al [14], les auteurs ont abordé le problème de la sélection de vues matérialisées sous la contrainte du temps de maintenance.

Les auteurs définissent deux notions : un graphe de vue de type ET, et un graphe de vue de type OU.

- Un graphe de vue de type ET est un graphe de vue dans lequel cette dernière possède un plan d'exécution unique.
- Un graphe de vue de type OU est un graphe de vue dans lequel cette dernière possède plusieurs plans d'exécution.

Etant donné un graphe de vues de type ET-OU et une quantité S (temps de maintenance disponible), le PSV consiste à sélectionner un ensemble de vues minimisant le temps de réponse total tel que le temps total de maintenance des vues soit inférieur à S. Les auteurs ont présenté deux heuristiques pour résoudre ce problème :

1. Un algorithme "glouton" polynômial fournissant une solution quasi-optimale pour les graphes de vues de type ET et pour les graphes de vues de type OU.
2. Un algorithme de type A* pour les graphes de vues de type ET et OU. Notons que tout algorithme de type A* cherche la solution optimale dans un graphe ayant un petit nombre de nœuds, où chacune représente une solution potentielle.

VI. La synthèse

De nombreux algorithmes ont été développés pour élaborer une solution optimale ou quasi-optimale pour le problème de sélection des vues matérialisées. Ces algorithmes peuvent être classés en fonction du type de contrainte qu'ils utilisent.

Nous avons abordé le problème de sélection des vues matérialisées et présenté quelques travaux qui existent dans la littérature traitant ce problème (tableau II.1). Nous allons résumer ces travaux en les classant suivant plusieurs critères:

- Algorithme de sélection utilisé.
- Type de contrainte.
- Base de données.

Tableau II.1 Comparaison des travaux effectués sur la sélection des vues matérialisées.

Travaux	**Algorithme de sélection**	**Type de contrainte**	**Base de données**
Yang et al « a feasible solution »	Algorithme heuristique	Sans contrainte	/
Yang et al « la programmation binaire 0-1 »	/	Sans contrainte	SSB
Ahcéne et al	Algorithmes de la théorie des graphes	Sans contrainte	SSB
Harinarayan et al	Glouton	Contrainte d'espace	TPC-D
Gupta et al	- Glouton - A*	Temps de maintenance	/

VII. Conclusion

Le problème de sélection des vues matérialisées est considéré comme étant un problème d'optimisation dans lequel on définit une fonction de coût, que l'on cherche à minimiser.

Nous avons décrit dans ce chapitre le contexte du problème de sélection des vues matérialisées et on a cité quelques travaux qui ont été proposés pour résoudre ce problème.

Dans le chapitre suivant nous détaillerons notre approche pour la sélection des vues matérialisées.

I. Introduction

Dans le contexte d'entrepôt de données, les requêtes exécutées renferment des opérations de jointures en étoile entre table de faits et plusieurs tables de dimensions avec des opérations de sélection sur les dimensions sur plusieurs attributs. Ces opérations sont très couteuses et nécessitent une optimisation des requêtes.

Nous proposons ainsi, une nouvelle approche de sélection des vues matérialisées. Cette démarche se base sur l'utilisation des techniques de data-mining, plus précisément l'algorithme de classification k-means. Nous présentons, dans ce chapitre, le principe de cette nouvelle démarche.

II. Approche proposée

Notre approche est basée sur l'utilisation des techniques du data-mining, et spécialement le K-means, pour optimiser une charge de requêtes.

On utilise l'algorithme de classification K-means pour classer l'ensemble des requêtes suivant les noeuds de jointure. Ces requêtes ont des expressions communes entre eux.

II.1 L'algorithme de classification K-means

K-means est un algorithme de quantification vectorielle (clustering en anglais). K-means est un algorithme de minimisation alternée qui, étant donné un entier K, va chercher à séparer un ensemble de points en K clusters [15].

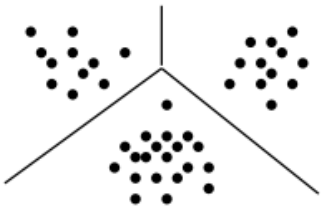

Figure III.1 la classification d'un ensemble de points.

II.2 Principe de l'algorithme K-means

Le principe de cet algorithme est de partitionner *M* données $d_1, \ldots, d_m$ en k classes $C_1, \ldots, C_k$ en positionnant dans l'espace des données, R_n, k centroïdes $ct_1, \ldots, ct_k$. Ces centroïdes sont généralement choisis de manière aléatoire parmi les données elles même. Chaque donnée est alors affectée à une classe dont le centroïde est le plus proche de cette donnée [16].

La notion de données proches peut être exprimée par la distance euclidienne entre les données représentées dans l'espace $\mathbf{R_n}$ et chaque centroïde. La distance euclidienne entre une donnée **d** dont les coordonnées dans R_n sont $(d_1,\ldots, d_n)$ et un centroïde ct avec les coordonnées $(ct_1,\ldots, ct_n)$ est donnée par la formule suivante :

$$\sqrt{\sum_{i=1}^{n}(d_i - ct_i)^2}$$

Afin de vérifier la stabilité des classes, k-means effectue la réaffectation des données à chaque classe sur plusieurs itérations. A chaque itération, k nouveaux centroïdes sont déterminés et les distances entre les données et ces centroïdes sont calculées afin de réaffecter les données à la classe la plus approprié.

II.3 Exemple de problème de sélection des vues matérialisées

Voila un exemple de 5 requêtes:

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

On fait une analyse sur ce groupe de requêtes définie comme suit : on extrait les tables de base et les opérations de jointure pour chaque requête qui seront ensuite identifiées chacune séparément dans notre exemple.

- ✓ **Tables de base**

0: lineorder
1: dates
2 : part
3 : supplier

- ✓ **Les jointures**

9 : lo_orderdate = d_datekey pfact=6 pdim=4
10 : lo_orderdate = d_datekey pfact=0 pdim=4
11 : lo_orderdate = d_datekey pfact=0 pdim=1
12 : lo_partkey = p_partkey pfact=0 pdim=7
13 :lo_suppkey = s_suppkey pfact=0 pdim=8

A partir de cet analyse, on définie nos données d'apprentissage qui représentent les nœuds de jointure avec leurs identificateurs qui concernent chaque requête tel que:

1 : si la requête Qi contient le nœud nj.

0 : sinon

Ainsi, on obtient le tableau ci-dessous :

Figure III. 2 Extraction des nœuds de jointure pour chaque requête.

	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

Ensuite, on applique le k-means qui nous permet d'obtenir une meilleure classification qui est de trois classes selon notre exemple.

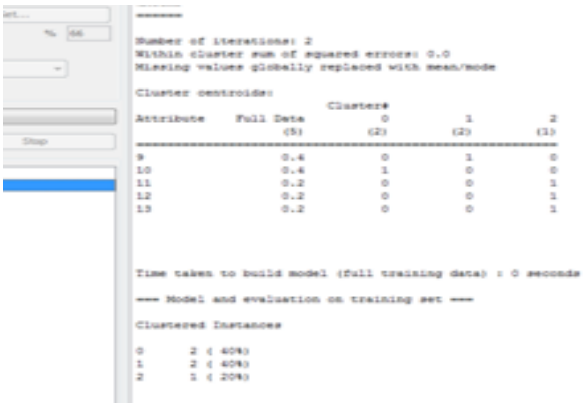

Figure III. 3 Classification aves K_means.

L'étape suivante est l'obtention des classes extraites par l'application du K-means avec les nœuds de jointure pour chaque requête.

C1= {Q0 ,Q1}, {Nj 9}
C2 = {Q2,Q3} , {Nj10}
C3 = {Q5}, {Nj11, Nj12, Nj13}

Ensuite, on applique l'opération d'intersection entre les classes pour extraire les nœuds communs entre les classes afin de capter au maximum l'interaction qui existe. Ces nœuds sont appelés nœuds fusion qui seront ensuite matérialisés.

Cette opération permet de minimiser le coût d'exécution des requêtes, ce qui est notre objectif principal.

III. Implémentation et conception

III.1 Environnement de développement

L'application est développée par le langage de programmation Java, et on utilise l'Eclipse, et 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 [17].
- ✓ **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
- ✓ **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.

III.2 Notre base de données

Le Star Schéma Benchmark (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. SSB comporte une table des faits LINEORDERS qui résulte de la fusion des tables LINEITEM and ORDERS et quatre tables de dimension PART, SUPPLIER, CUSTOMER et DATE. Son schéma logique est illustré dans la figure ci-dessous [18].

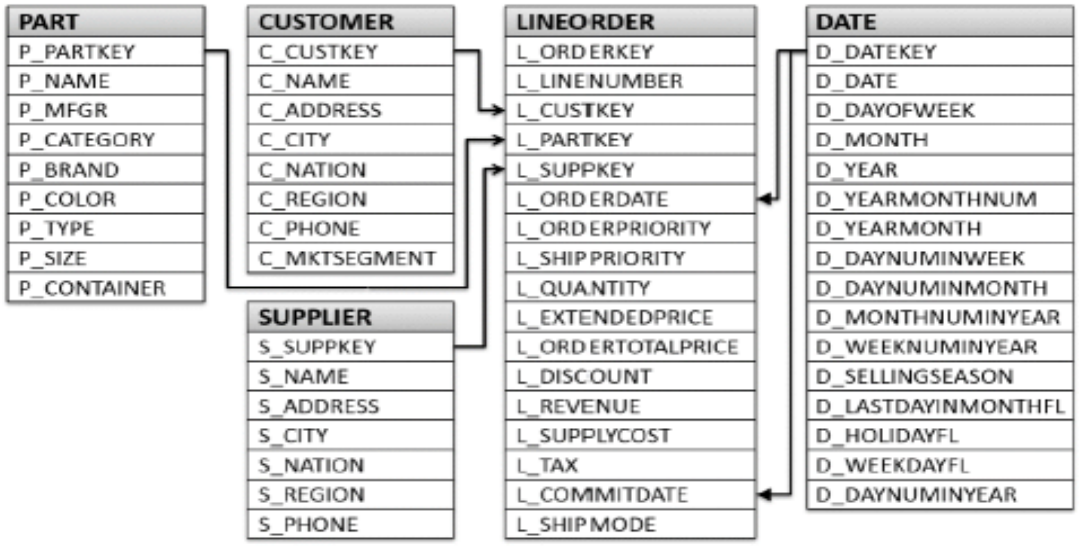

Figure III. 4 Schéma logique du SSB.

IV. Notre application

IV.1 Interface de connexion à la base de données

La figure ci-dessous représente notre première interface qui est celle de connexion à notre base de données.

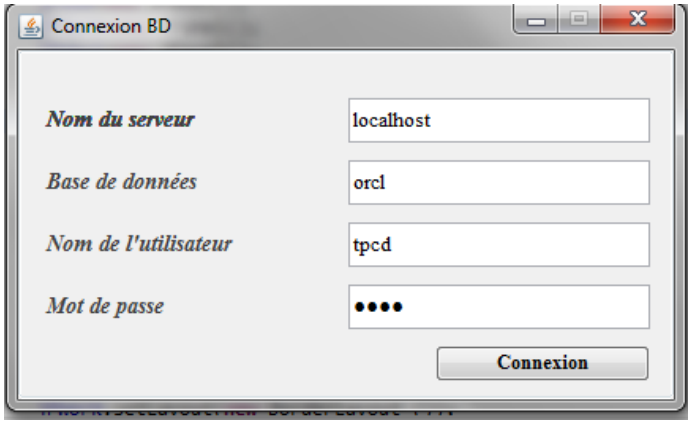

Figure III. 5 Interface de connexion à la base de données

IV.2 Interface principale

C'est notre interface principale qui contient un bouton de chargement de requêtes, un bouton d'analyse du workload et un bouton pour l'application du k-means avec un choix du nombre de classes k

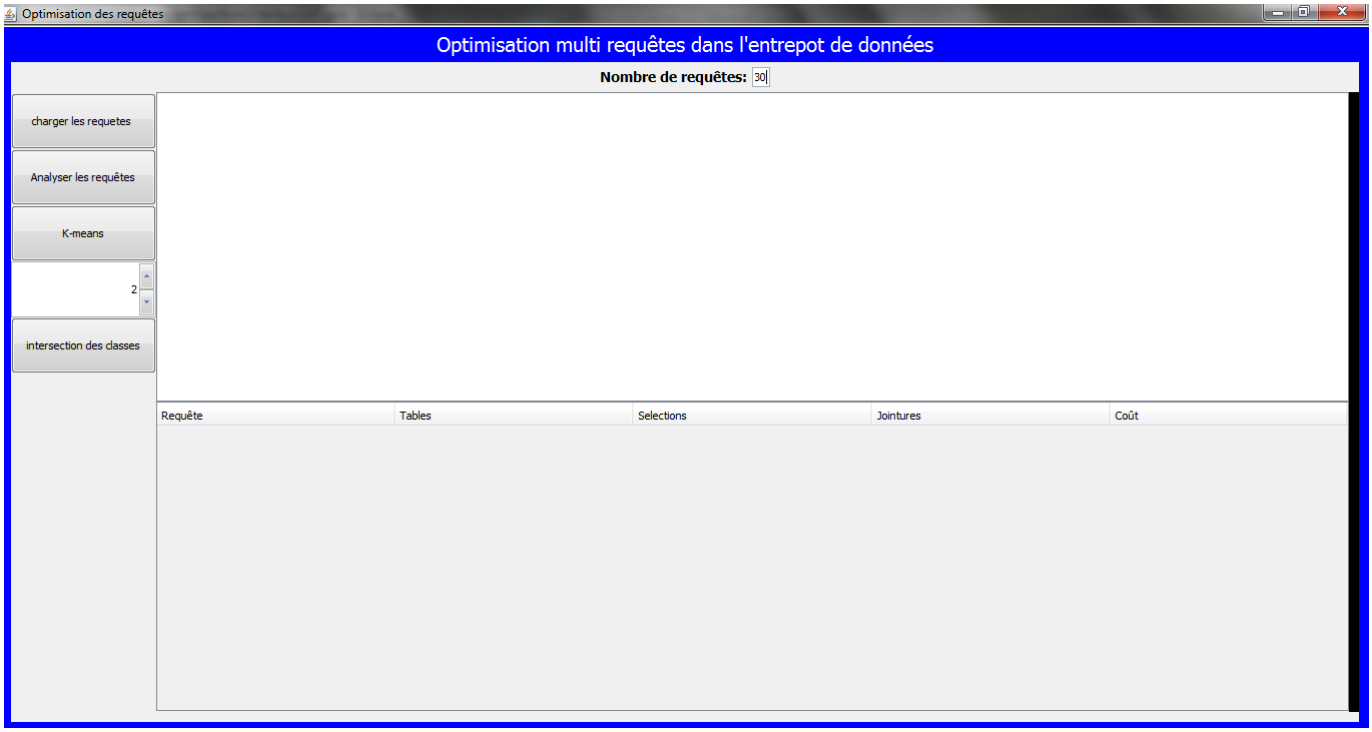

Figure III. 6 Interface principale

IV.3 Chargement des requêtes

Le chargement des requêtes permet d'afficher la charge des requêtes de notre base de données.

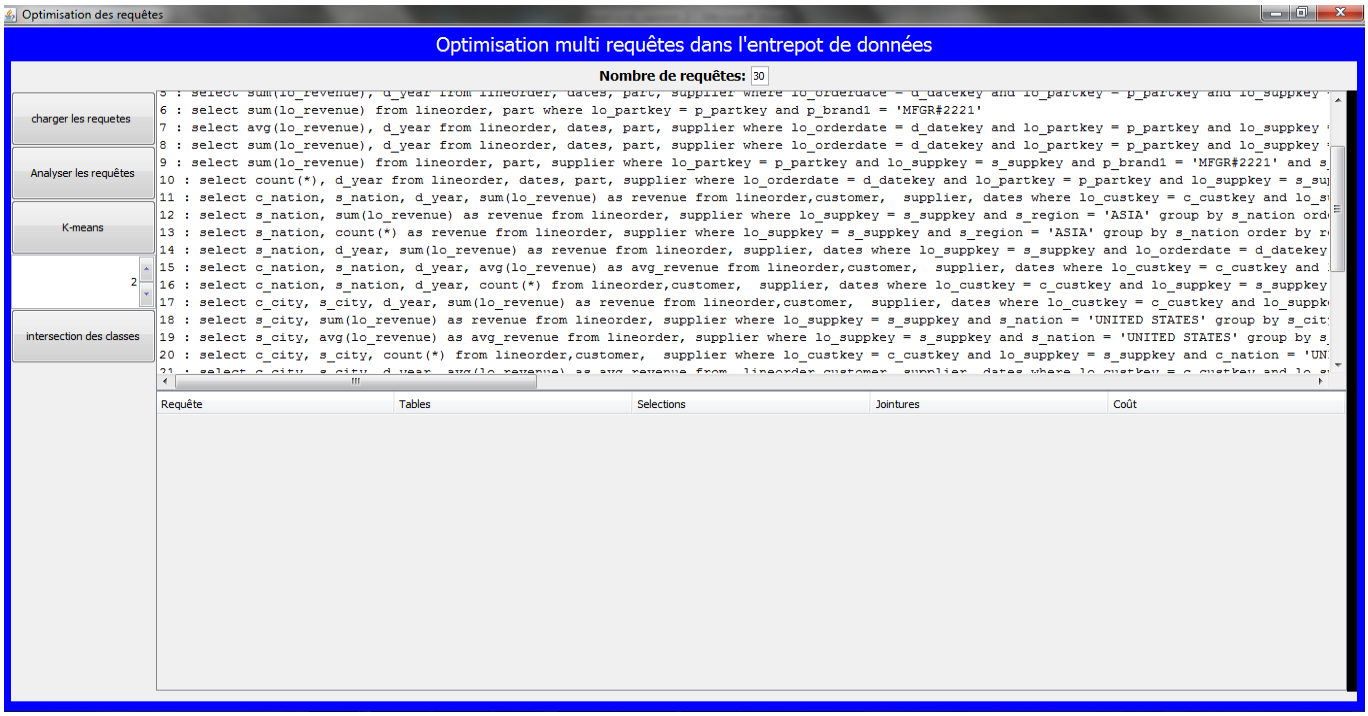

Figure III. 7 Chargement des requêtes.

IV.4 L'analyse des requêtes

Dans cette partie, on va analyser notre charge de requêtes. Tout d'abord on va extraire, les tables de base, les sélections, les nœuds de jointure et les projections pour chaque requête, et les identifier.

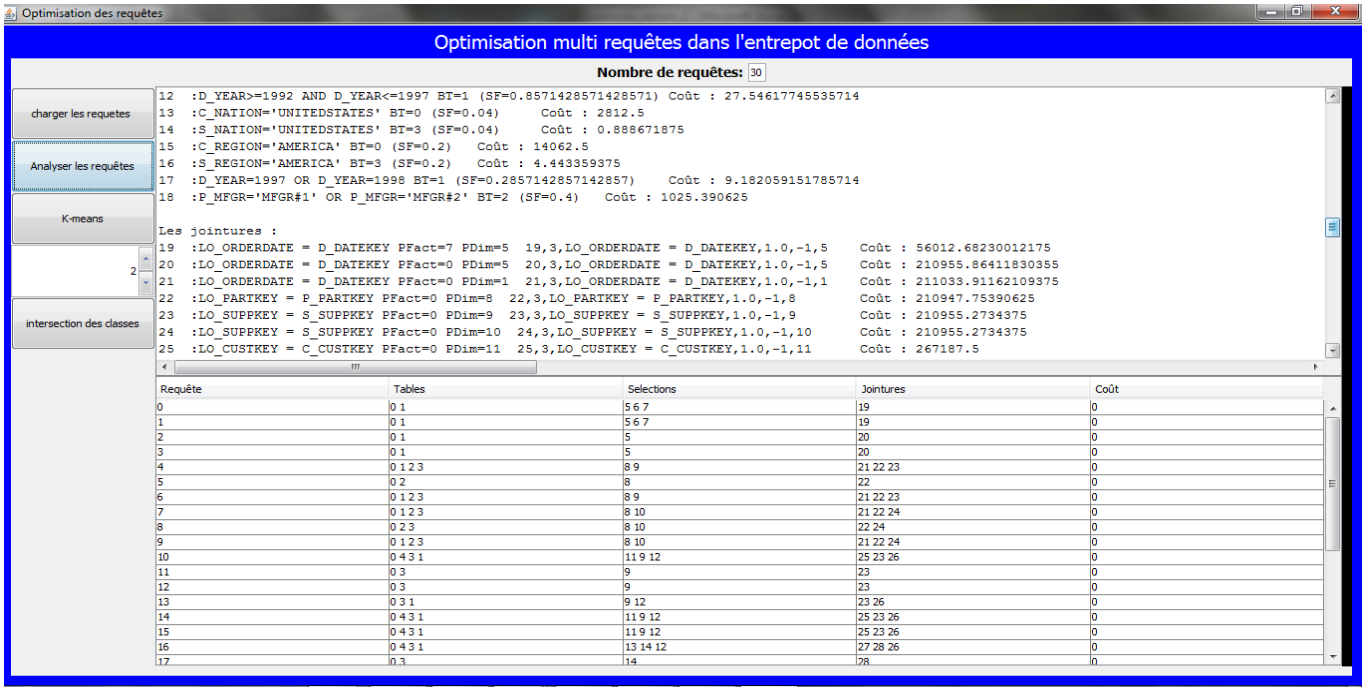

Figure III. 8 Analyse des requêtes.

IV.5 Application du k-means

Après l'analyse des requêtes, on applique le k-means avec un nombre de cluster k déterminé par l'utilisateur. Cette étape permet de classer les requêtes en interaction (nœud de jointure en commun).

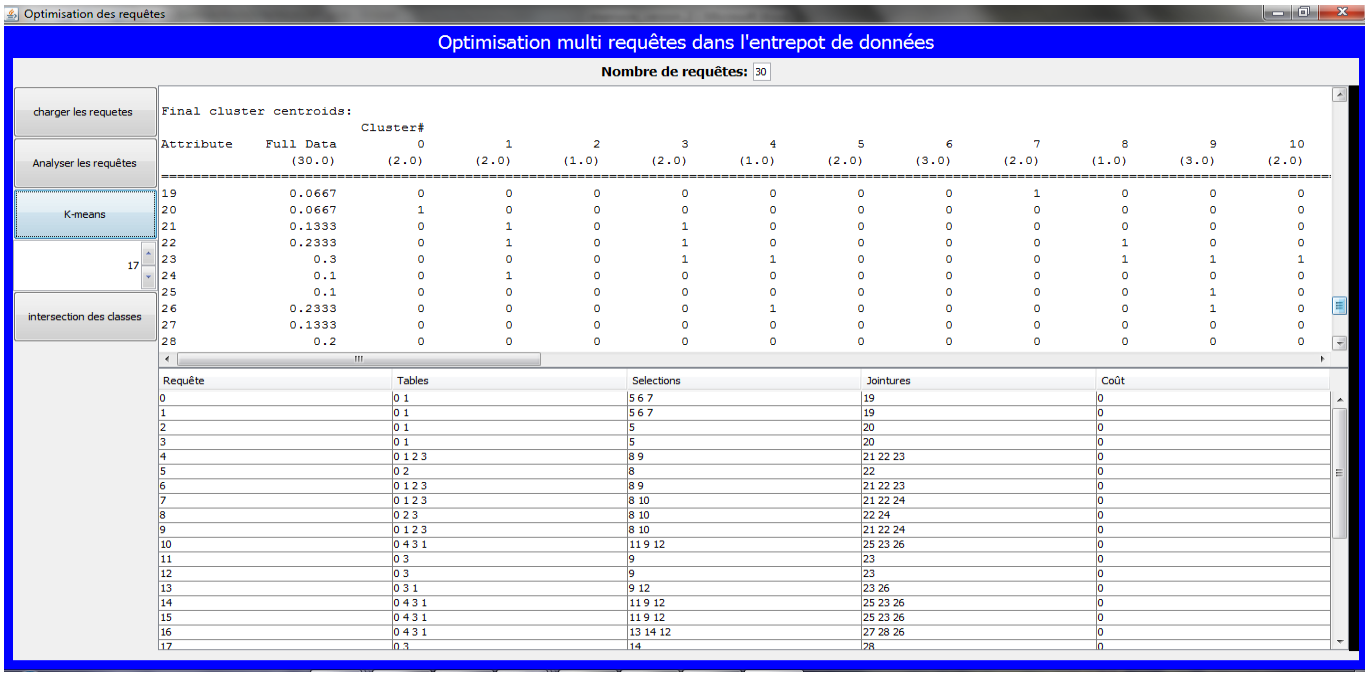

Figure III. 9 Application du k-means.

IV.6 intersection des classes

Après l'application du k-means, on analyse les classes obtenues. Pour chaque classe, on extrait toutes les informations nécessaires à l'analyse qui sont les requêtes et les nœuds de jointure, et on applique l'intersection.

Les résultats obtenus après l'opération d'intersection sont des nouvelles classes qui ont un nœud de jointure en commun (nœud fusion).

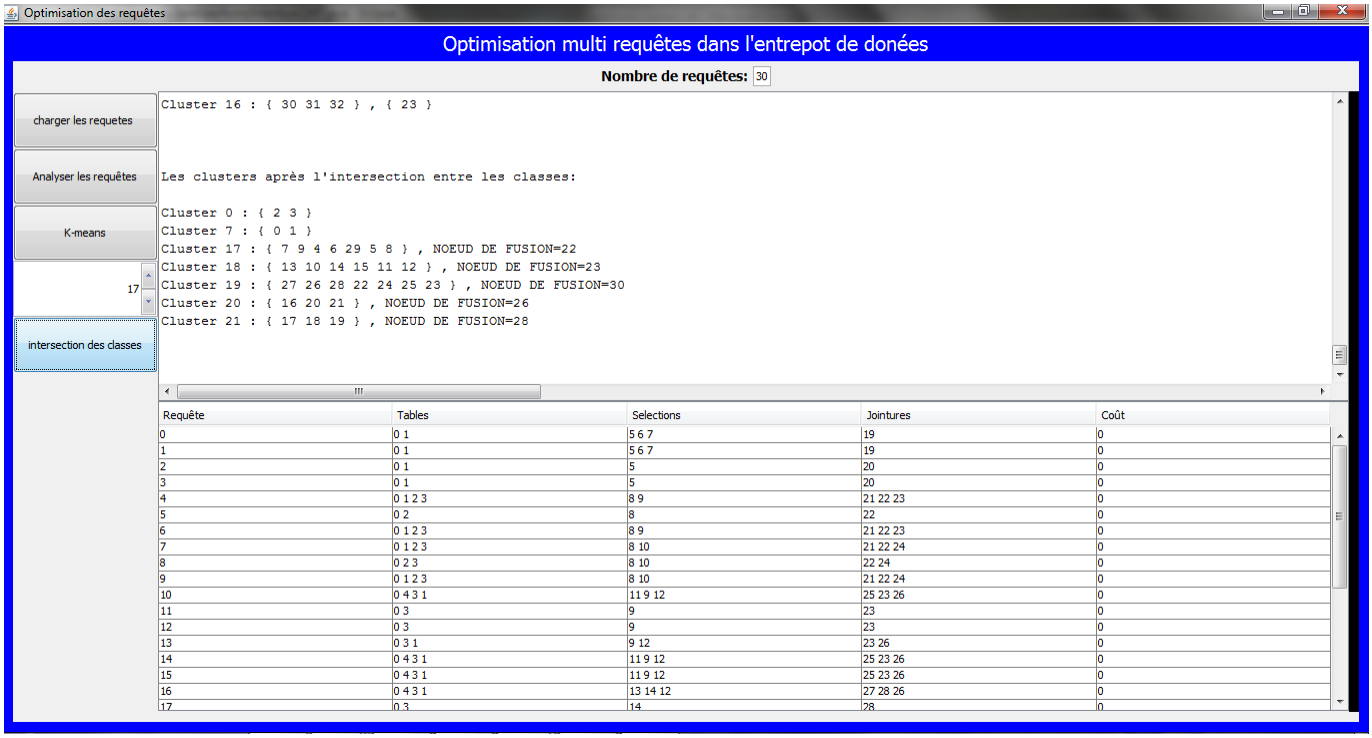

Figure III. 10 Intersection des classes

V. Conclusion

Nous avons appliqué dans ce chapitre une nouvelle approche pour résoudre le problème de sélection des vues matérialisées.

Cette approche se base sur l'application de l'algorithme de classification k-means sur une charge de requêtes pour détecter l'interaction entre elles.

Au début, on a présenté notre approche pour résoudre ce problème et on a choisi un petit exemple pour notre application, et à la fin, on a présenté notre application et ses interfaces principales.

CONCLUSION GENERALE

Conclusion générale

La croissance contenue de la taille des entrepôts de données rend le champ de recherche des techniques d'optimisation des requêtes très délicat. Parmi ces techniques, on s'est intéressé par les vues matérialisées. La problématique suggérée dans ce domaine est qu'elles sont les vues adéquates pour la matérialisation afin d'optimiser la charge de requête.

Nous avons dans ce document répondu à la problématique de sélection des vues matérialisées, à savoir, s'il était possible de définir une méthode d'exploration de l'ensemble représentant les vues candidates sans passer par des simulations. La méthode que nous proposons s'appuie essentiellement sur la technique du data-mining en utilisant l'algorithme de classification k-means pour partitionner initialement l'ensemble des requêtes qui sont similaires syntaxiquement suivant le nœud de jointure, puis on fusionne les classes qui sont en interaction. Pour faire ce travail, nous avons défini une fonction intra-classe.

L'intérêt de l'optimisation par les vues matérialisées est de rendre disponible les résultats intermédiaires pouvant être utilisés par des requêtes complexes, réduisant ainsi la complexité et le coût d'exécution de ces requêtes, donc une meilleure performance.

Dans notre application, il reste à évaluer notre approche, à l'aide d'un modèle de coût, et comparer les résultats aux travaux présentés dans le contexte de sélection des vues matérialisées.

Perspectifs

Nous avons développé une méthodologie statique pour sélectionner les vues adéquates à matérialiser. Dans ce travail, nous n'avons pas abordé le modèle de coût pour les requêtes.

Ce travail est basé sur une technique unique d'optimisation qui n'est pas corrélée avec les caractéristiques de complexité du problème. Intuitivement, il serait souhaitable d'utiliser des techniques hybrides permettant une meilleure adéquation au problème à traiter soit à priori de manière statique ou d'une manière dynamique

Liste des références

[1] Bellatreche Ladjel, « Techniques d'optimisation des requêtes dans les data warehouses », « 6th International Symposium on Programming and Systems (ISPS 03) », Algérie, 2003, p. 81-98.

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

[3] 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 de doctorat, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechnique-Poitiers France, 2014.

[4] Filali abderrahmane, kedjnane sofiane et al, « Conception et réalisation d'un Data Warehouse pour la mise en place d'un système décisionnel », Mémoire de fin d'études diplôme d'Ingénieur d'Etat en Informatique, Ecole national supérieur d'informatique ESI, 2010.

[5] Selma khouri et al, « cycle de vie sémantique de conception de systèmes de stockage et manipulation de données », Thése de doctorat, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechnique-Poitiers France, 2013.

[6] Bouchakri rym, Ladjel Bellatreche et al, « Administration et Tuning des Entrepôts de Données : Optimisation par Index de Jointure Binaires et Fragmentation Horizontale », « Doctoriales STIC'09 », Msila Algérie, 2009.

[7] Boukorca Ahcène et al, « Sonic: Scalable multi-query optimization through integrated circuits », « Database and Expert Systems Applications », Springer Berlin Heidelberg, 2013, p. 278-292.

[8] Aljanaby Alaa, Emad Abuelrub et al, « A Survey of Distributed Query Optimization », « The International Arab Journal of Information Technology », la Jordanie, 2005, p. 48-57.

[9] Mbaiossoum Bery Leouro et al, « Conception physique des bases de données à base ontologique: le cas des vues matérialisées », Thèse de doctorat, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechnique-Poitiers France, 2014.

[10] Bouchakri Rima et al, « Conception physique statique et dynamique des entrepôts de données », Thèse de doctorat, Ecole nationale supérieure de mécanique et d'aérotechnique-Poitiers France, 2015.

[11] Yang Jian et al, « Algorithms for materialized view design in data warehousing environment », «Proceedings of the 23 rd International Conference on Very Large Data bases (VLDB) », Athènes Grèce, 1997, p. 136-145.

[12] Harinarayan Venky, Anand Rajaraman et al, « Implementing data cubes efficiently »,« Proceedings of the ACM SIGMOD International Conference on Management of Data», Montreal Canada, 1996, p. 205-216.

[13] Shukla Amit, Deshpande Prasad et al, « Materialized view selection for multidimensional datasets », «Proceedings of the 24th VLDB Conference, New York USA», 1998, p. 488-499.

[14] Gupta Himanshu, « Selection and maintenance of views in a data warehouse », Thèse de doctorat, université de Stanford San Francisco, 1999.

[15] Wagstaff, Cardie et al, « Constrained k-means clustering with background knowledge », ICML, 2001, p. 577-584.

[16] LELU Alain et al, « La méthode de classification non-supervisée K-means », Document LORIA, 2008, université de comté, p. 14-25.

[17] kathy sierra, bert bates, « head first java », livre, 2005.

[18] O'Neil Patrick, Elizabeth O'Neil et al, « The star schema benchmark (*SSB)* », article, boston, 2007.

[19] Zukowski John, « The definitive guide to Java Swing », Apress, 2005.

[20] Louis SWINNEN, « Laboratoire de base de données Introduction à JDBC »,saint-laurent , 2009.