MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DELA RECHERCHESCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Ingénierie des Systèmes d'Information**

Présenté par :

HAOUACH HADJER

SALHI ASMAA

THÈME :

Optimisation des requêtes par les vue matérialisées dynamique

Soutenu le : 14/09/2021

Devant le jury compose de

Mme Kenniche.A	Université de Mostaganem	Présidente
Mme Bouzebiba.H	Université de Mostaganem	Examinatrice
Mme Betouati.F	Université de Mostaganem	Encadrant

Année Universitaire 2020-2021

Dédicaces

Je dédie ce modeste travail :

A l'homme de ma vie, mon soutien moral et

source de joie et de bonheur, celui qui s'est toujours sacrifié

Pour me voir réussir de toutes ma vie, que dieu te garde dans son vaste paradis, à toi Mon père.

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur, maman que j'adore.

A mes chères sœurs

A mes chers frères

A Fatiha, Houria, Mns, Fatima, Amel

A Azzeddine, Kacem

A toutes la famille de maman HAMMOU

A toutes ma famille HAOUACH

A notre encadreur Mm.Betouati

A mon binôme SALHI ASMAA

A mes très chères amies

A tous ceux qui m'ont soutenu moralement de près ou de loin.

Enfin à toute(es) les étudiants de la promotion Master2 informatique avec lesquelles j'ai passé cette année

HAOUACH HADJER

Dédicaces

Je dédie ce modeste travail :

A l'homme de ma vie, mon soutien moral et
source de joie et de bonheur, celui qui s'est toujours sacrifié
Pour me voir réussir de toutes ma vie, que dieu te garde dans son vaste paradis, à toi Mon père.
A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur, maman que j'adore.
A mes chères sœurs
A mes chers frères
A notre encadreur Mm.Betouati
A mon binôme HAOUACH HADJER

A mes très chères amies

A tous ceux qui m'ont soutenu moralement de près ou de loin.
Enfin à toute(es) les étudiants de la promotion Master2

informatique avec lesquelles j'ai passé cette année

SALHI ASMAA

Remerciements

Nous remercions vivement Notre créateur (Allah) pour nous avoir donné de la force à accomplir ce travail.

Notre profonde gratitude à Mm.Betouati, d'avoir proposé ce sujet et qui s'est toujours montré à l'écoute et très impliqué tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Nous tenons à exprimer nos sincères remerciements aux membres du jury qui nous font honneur en jugeant ce travail.

Nous adressons également notre gratitude à tous les professeurs de l'université ***ABDELHAMID IBN BADIS*** *en particulier ceux du département De MATHEMATIQUE ET INFORMATIQUE.*

Enfin, nous remercions toutes les personnes qui, de près ou de loin, ont contribué à l'élaboration de ce mémoire.

Résumer

L’entrepôt de données occupe une place importante dans l'attention des utilisateurs de la base de données. L'idée de la mise en œuvre d'un entrepôt de données est de fournir un accès permanent aux données même lorsqu'une seule base de données n'est pas accessible, et de réduire l'accès à distance au système qui gère 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 de ce projet implique l'optimisation des requêtes de vues matérialisées dynamiques Cette technologie pose un problème d'optimisation important, à savoir le choix des vues. Le problème est de sélectionner l'ensemble de vues à implémenter pour accélérer l'exécution future des requêtes.

Mot-clé : entrepôt de données, vue matérialisée, Problème de Sélection des Vues Matérialisées (PSV).

Abstract

The data warehouse occupies an important place in the attention of the users of the database. The idea of implementing a data warehouse is to provide permanent access to data even when a single database is not accessible, and to reduce remote access to the system that manages the data original.

Data warehouses are dedicated to analysis and decision making applications. The analysis process is carried out using complex queries with multiple joins and aggregation operations on large tables, the administrator, in order to minimize the cost of executing these queries, selects a set materialized views that represent the join nodes.

This selection decreases the cost of queries, but causes the problem of occupying space for materialized views, and as a result, they cannot be stored entirely in main memory.

The work of this project involves the optimization of dynamic materialized view queries. This technology poses an important optimization problem, namely the choice of views.

The problem is to select the set of views to implement to speed up the future execution of the queries.

Keywords: Data warehouses, materialized view, View Selection Problem (VSP)

Liste des figures

Liste des tableaux

Table des matières

Introduction Générale

Systèmes décisionnels adoptent l'architecture basée sur les entrepôts de données.

Ce qui justifie aisément le choix de l'entrepôt de données pour mettre en place notre base de stockage destinée aux gestionnaires et aux décideurs des diverses structures.

Dans ses travaux sur les techniques de construction des entrepôts de données multidimensionnelles, (Kimball, 1996) propose la définition suivante : « Un entrepôt de données est un espace de stockage centralisé sur lequel repose un système décisionnel, son rôle est d'intégrer et de stocker l'information utile aux décideurs et de conserver l'historique des données pour supporter les analyses effectuées lors de la prise de décision».

Il est connu que les requêtes analytiques sont coûteuses, du fait qu'elles contiennent des jointures et des agrégations sur les relations des bases de production qui sont volumineuses, ce coût diminue les performances du système.

Un entrepôt de données doit fournir des données actualisées, si les données des bases de production sont modifiées, alors ces modifications doivent être répercutées au niveau de l'entrepôt de données afin de garantir la cohérence de ses données. Cette procédure s'appelle la maintenance de l'entrepôt de données.

Pour maintenir l'entrepôt de données, et optimiser les requêtes analytiques, l'administrateur doit effectuer la tâche de la conception physique dans laquelle il choisit un ensemble de techniques d'optimisations afin d'améliorer les performances du système ; les vues matérialisées font partie de ces techniques.

Un entrepôt de données est vu comme un ensemble de vues matérialisées pour améliorée le temps de réponse, ce qui implique que sa maintenance consiste à maintenir

l‘ensemble de ses vues matérialisées, en répercutant les modifications survenues aux sources de données sur ces vues matérialisées.

En effet, l‘ensemble des vues matérialisées de l‘entrepôt de données est un résultat d‘une sélection des différents résultats intermédiaires des différentes requêtes, dont le but est d‘accélérer ces requêtes. Mais la matérialisation de tous les résultats est une solution non pratique, à cause de l‘espace de stockage limité et le temps de leur maintenance, qui nécessite de matérialiser un sous ensemble de résultats, dans lequel le temps d‘exécution des requêtes, l‘espace de stockage et le temps de maintenance varient adéquatement ; ce problème est connu par la sélection de vues matérialisées. Par conséquent, ces vues sont multi-relations du fait qu‘elles matérialisent les jointures.

Objectif de notre projet est optimisation des requêtes par les vues matérialisé dynamiques.

Pour cela, nous avons divisé notre travail en quatre chapitres : le premier chapitre abordera les généralités sur l’entrepôt de données et l’optimisation des requêtes, dans le deuxième chapitre nous présenterons les problèmes de sélection des vues matérialisée dans les deux cas statique et dynamique, et les travaux développés dans ce contexte, au troisième chapitre nous présentons notre approche de sélection ces vues dynamique ainsi que les étapes pour implémenter notre solution, dans le dernier chapitre nous allons présenter notre projet, et décrire l'implémentation de ce problème et présentation des interfaces. Nous terminerons par une conclusion générale.

Chapitre 1

Généralité sur l’entrepôt de données

1.1 Introduction

L'entrepôt de données est un concept clé dans le domaine de l'intelligence d'affaires.

En effet, les analystes du monde de l'information n'ont pas encore trouvé de meilleur moyen de modéliser toutes les données de l'entreprise de manière unifiée et simple, c'est donc la seule solution. Un entrepôt est une structure qui peut stocker toutes les données de l'entreprise dans un format spécifique pour référence.

Dans ce chapitre, nous avons présenté le concept d'entrepôt de données, son histoire, son architecture et présenté en détail toutes les étapes du cycle de vie de la conception de l'entrepôt de données en particulier conception physique et les techniques d'optimisation des requêtes.

1.2 Entrepôt de données

L'entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision. [1]

- **Orientés sujet :** Les données de l'entrepôt sont organisées par thème (autour des sujets majeurs et des métiers de l'entreprise). L'intérêt dans cette organisation est de disposer d'un ensemble d'informations utiles sur un sujet transversal aux structures fonctionnelles et organisationnelles de l'entreprise.

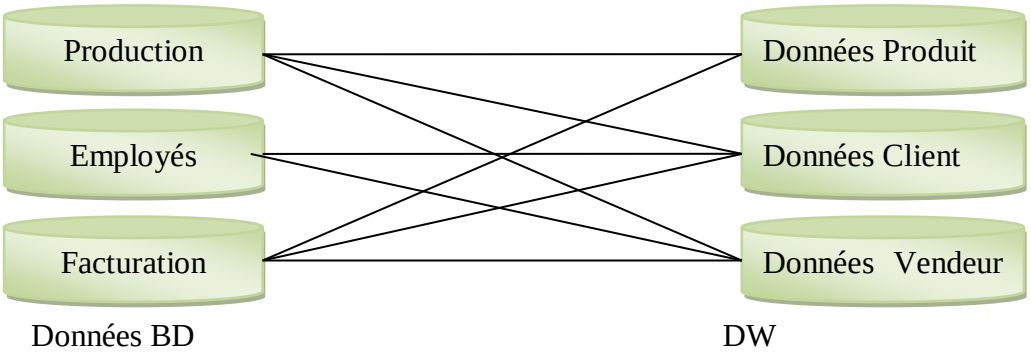

Figure 1.1 : Orienté sujet (source : C Vangenot, Laboratoire de Bases de Données)[1]

- **Données intégrées :** Les données proviennent de différentes sources souvent structurées et codées de façon différente. L'intégration assure une représentation uniforme, cohérente et transparente. Cela résout les problèmes d'hétérogénéité des systèmes de stockage, des modèles de données et de sémantique de données.

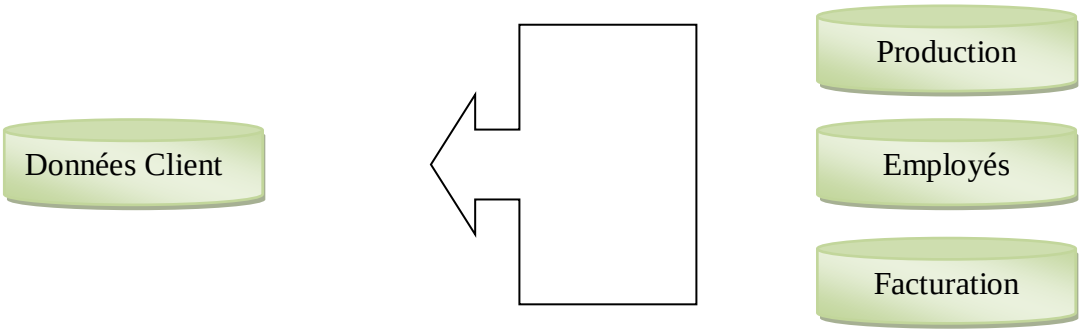

Figure 1.2 : Données intégrées (source : C Vangenot, Laboratoire de Bases de Données) [1]

- **Données historisées :** Les données proviennent de différentes sources souvent structurées et codées de façon différente. L'intégration assure une représentation uniforme, cohérente et transparente. Cela résout les problèmes d'hétérogénéité des systèmes de stockage, des modèles de données et de sémantique de données.
- **Données non volatiles** : A la différence des données opérationnelles, celles de l'entrepôt sont permanentes et ne peuvent pas être modifiées. Le rafraîchissement de l'entrepôt, consiste à ajouter de nouvelles données, sans modifier ou perdre celles qui existent.

1.3 Historique

L'origine des ED revient à 1960, ou l'entreprise General Mills et l'Université Dartmouth, dans un projet conjoint, créent les termes "faits" et "dimensions" .Les dates marquantes de l'histoire des entrepôts de données sont [2] :

- 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 destiné 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.*

1.4 Architecture d'un entrepôt de données

L'intégration de sources de données hétérogènes avec des structures différentes est l'objectif principal de la naissance des entrepôts de données. Cependant, l'entrepôt de données joue un rôle stratégique dans la vie d'une entreprise. Il stocke les données liées aux besoins de prise de décision du système d'exploitation de l'entreprise et d'autres sources externes [3].

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

- Extraction des données à partir de différentes sources.
- Organisation et intégration des données dans l'entrepôt.
- Accès aux données intégrées et analyse de ces dernières dans une forme efficace et flexible (voir figure 1.3).

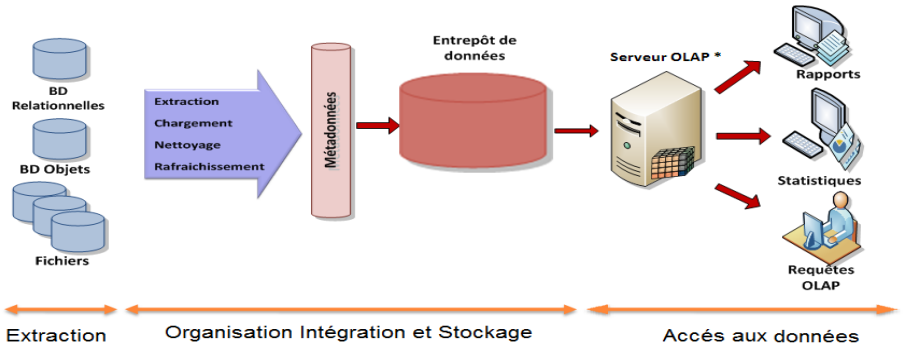

Figure 1.3 : l'architecture conceptuelle d'un entrepôt de données [4]

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.

1.5 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.

- **La planification :** Cette phase vise à préparer le terrain pour le développement de l'entrepôt de données. Elle consiste à [5]:
  - Déterminer l'étendue du projet ainsi que les buts et objectifs de l'entrepôt à développer.
  - Evaluer la faisabilité technique et économique de l'entrepôt.
  - Identifier les futurs utilisateurs de l'entrepôt.
- **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 [5].

- **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 [5].
- **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 [5].

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

1.5.1 Analyse des besoins

La phase d'analyse des besoins est une tâche cruciale sur laquelle repose le processus de la prise de décision. Les exigences sont déterminées pour produire une spécification formelle des données nécessaires au traitement de données, les relations naturelles et la plateforme logicielle pour le déploiement. Tout d'abord, il est nécessaire de distinguer les besoins fonctionnels des besoins non fonctionnels. D'une manière informelle, *les besoins fonctionnels* sont ceux qui caractérisent le système, comme par exemple 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.). Par contre, *les exigences non fonctionnelles* sont les exigences implicites auxquelles le système doit répondre. Citons la performance, la réutilisabilité, la fiabilité, . . . etc. En particulier, les exigences fonctionnelles d'un entrepôt de données sont principalement liés à l'information que l'entrepôt est censé fournir, tandis que les besoins non fonctionnels n'affectent que les informations nécessaire à une utilisation correcte.

1.5.2 Conception logique

La conception logique d'un entrepôt de données vise à organiser et classer les informations des sources de données par sujet fonctionnel. Elle est préliminaire à la modélisation dimensionnelle où chaque sujet correspond à une table gérée au sein de l'entrepôt. Ce dernier permet d'isoler les données stratégiques et de conserver les métadonnées. L'interrogation des entrepôts de données se fait par des requêtes OLAP. Ces requêtes correspondent à une structuration des données selon plusieurs axes d'analyse pouvant représenter des notions variées telles que le temps de la localisation géographique ou le code identifiant des produits. Les modèles de conception des systèmes transactionnels ne sont pas adaptés à ce type de requêtes complexes qui utilisent beaucoup de jointures, demandent beaucoup de temps de calcul et sont de nature ad hoc. Pour ce type d'environnement, une nouvelle approche de modélisation a été suggérée: *les modèles multidimensionnels*.

1.5.2.1 Les modèles multidimensionnels

La modélisation multidimensionnelle permet d'observer un sujet analysé comme un point, dans un espace à plusieurs dimensions. Les données sont organisées d'une manière qui met en évidence le sujet en cours d'analyse et ses différentes perspectives d'analyse. Ainsi, ce modèle a donné naissance aux concepts de *fait* et de *dimension*. [6]

1.5.2.1.1 Un fait

Représente un sujet d'analyse. Il est constitué de plusieurs mesures relatives au sujet traité. Ces mesures sont numériques et généralement valorisées de façon continue, en général les faits sont implicitement représentés par la combinaison des valeurs des dimensions. La Figure1.4 illustre un exemple de tables de fait. [6]

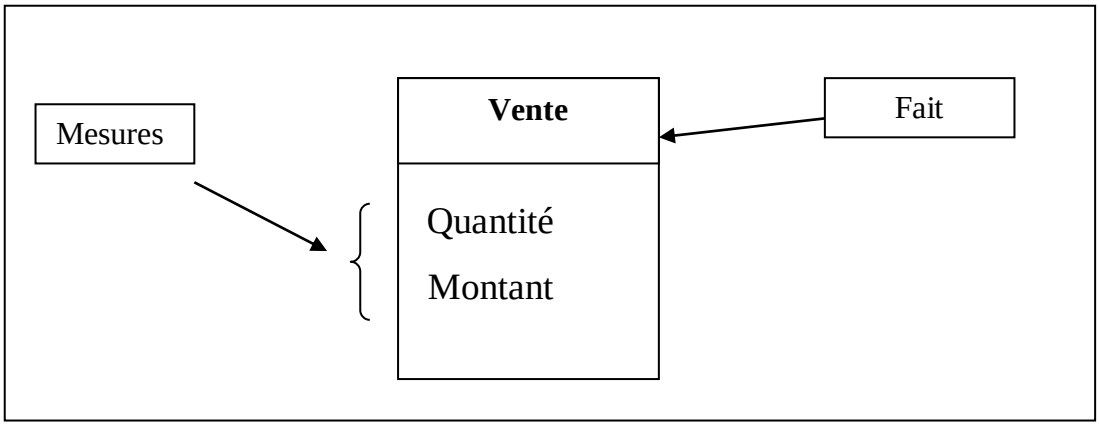

Figure 1.4 : Exemple de table de fait [6]

1.5.2.1.2 Une dimension

Concept essentiel des bases de données multidimensionnelles, la dimension est le critère suivant lequel on souhaite évaluer, quantifier et qualifier le fait, elle peut être utilisée pour la sélection de données selon le niveau de précision désiré. Aussi, elle peut être affinée, décomposée en hiérarchies, afin de permettre à l'utilisateur d'examiner ses indicateurs à différents niveaux de détail. La Figure1.5 illustre un exemple de tables de dimensions :

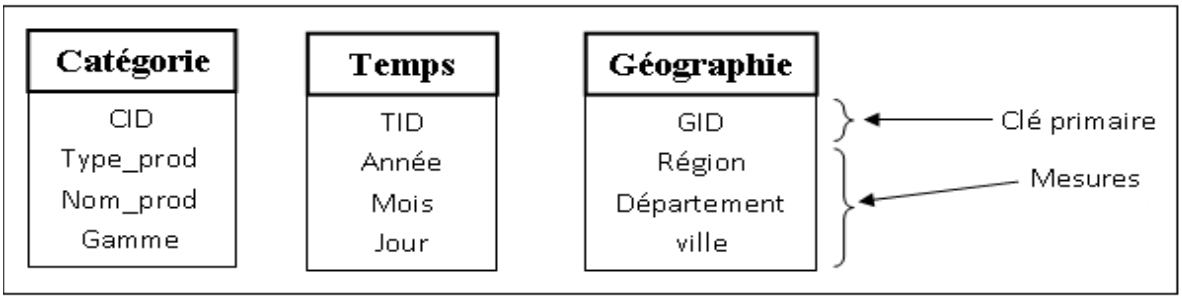

Figure 1.5 : Exemple de table dimension [6]

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.

1.5.2.2 Les systèmes MOLAP

Les systèmes de type 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 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.

1.5.2.3 Les systèmes ROLAP

Dans les systèmes de relation OLAP, les données multidimensionnelles sont mises en œuvre sous forme de tables relationnelles, tables de faits et de dimensions. La table de fait est constituée d'attributs représentant les mesures d'activité et les attributs clés étrangères de chacune des tables de dimension. Les tables de dimension contiennent les paramètres et une clé primaire permettant de réaliser des jointures avec la table de fait [6]. Ces tables sont organisées dans des structures appelées : schéma en étoile, schéma en flocon, schéma en constellation. Décrit comme suivant : [7]

1.5.2.3.1 Le schéma en étoile

Chaque dimension du cube est représentée par une table de dimension et les mesures par une table de faits qui référence les tables de dimension en utilisant une clé étrangère pour chacune d'elles. La table de faits est normalisée, les tables de dimension sont généralement dénormalisées. Cette représentation facilite l'analyse selon déférentes perspectives.

Les requêtes généralement appliquées sur ce schéma sont appelées " requêtes de jointure en étoile ", et ont les propriétés suivantes : [7]

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

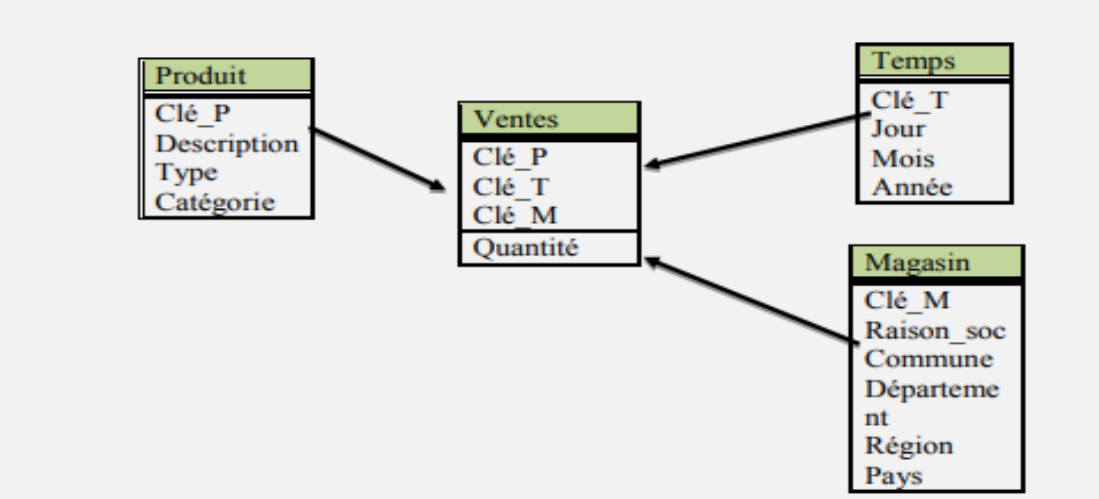

Figure 1.6 : Exemple d'un schéma en étoile [7]

La table de fait est ventes et les dimensions : produit, temps et magasin.

1.5.2.3.2 Schéma en flocon de neige

Le schéma en flocon de neige est une extension du schéma en étoile. Dans un schéma en étoile, les informations associées à une hiérarchie de dimension, sont représentées dans une seule table, même si les différents niveaux de la hiérarchie ont des propriétés différentes. Le schéma en flocon est le résultat de la décomposition d'une ou plusieurs dimensions en plusieurs niveaux formant une hiérarchie. Les tables de dimensions sont ainsi éclatées en plusieurs tables, ce qui peut être vu comme une normalisation des tables de dimensions. La table de faits reste inchangée. [7]

Les hiérarchies pour le schéma en flocon de neige de l'exemple de la figure

Dimension Temps : Jour → Mois → Année

Dimension Magasin : commune →Département → Région → Pays

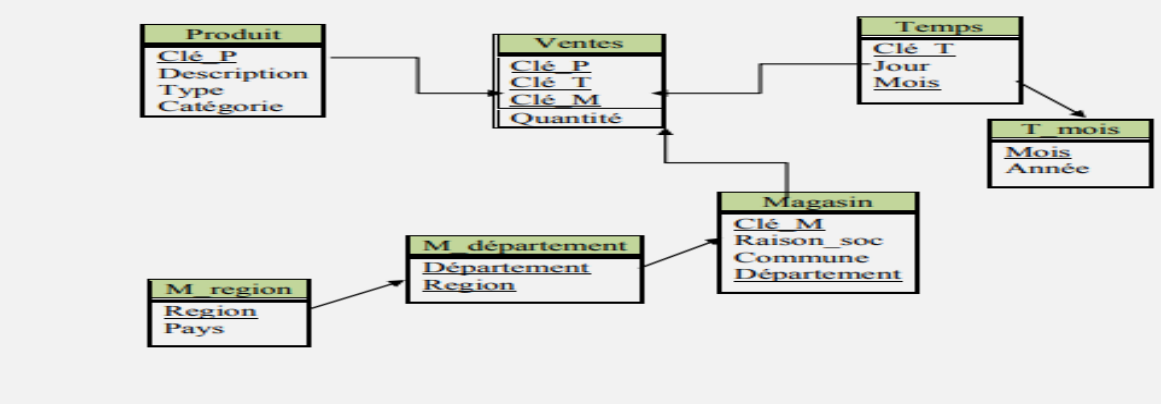

Figure 1.7 : exemple d'un schéma en flocon de neige [7]

Dans l'exemple ci-dessus, la dimension Temps a été éclatée en deux, Temps et T_Mois. La deuxième dimension Magasin, à été décomposée en trois : Magasin, M_Département et M_Région.

1.5.2.3.3 Le s

Sont des schémas où plusieurs modèles dimensionnels se partagent les mêmes dimensions, c'est-à-dire les tables de faits ont des tables de dimensions en commun. Les tables de dimensions partagées par plusieurs tables de fait doivent être exactement les mêmes.

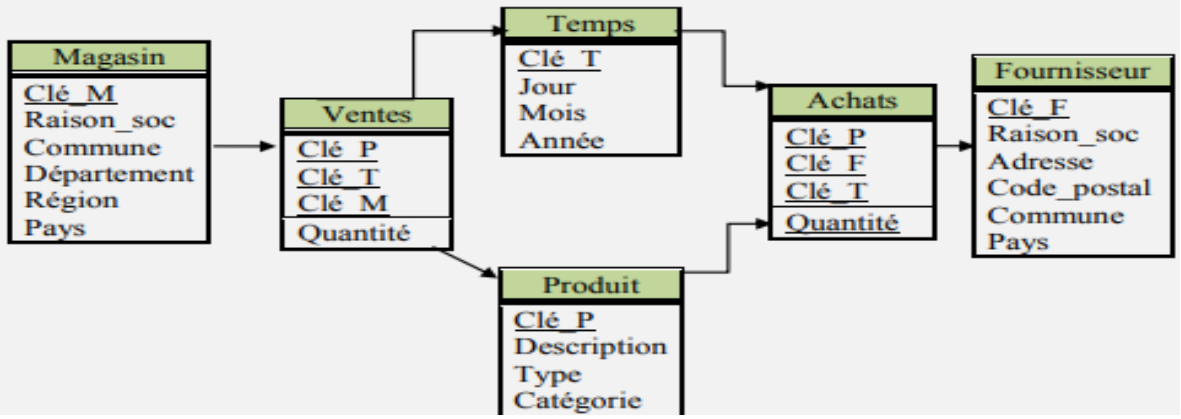

Figure 1.8 : exemple d'un schéma en constellation [7]

La figure 1.8 montre le schéma en constellation qui est composé de deux relations de faits. La première s'appelle Ventes et enregistre les quantités de produits qui ont été vendus

dans les différents magasins pendant un certain jour. La deuxième relation gère les différents produits achetés aux fournisseurs pendant un certain temps.

1.5.3 Phase ELT (Extra-Transform-Load)

Pour que des données sources soient exploitables, il est nécessaire de les agréger et de les nettoyer de tous les éléments non indispensables aux utilisateurs finaux. Cette opération d'extraction et d'homogénéisation des données est assurée par la technologie ETL (Extraction, Transformation and Loading).

L'ETL se charge de récupérer les données et de les centraliser dans l'entrepôt de données [7].
L'alimentation ETL se déroule selon les trois phases suivantes:

- **Extraction de données :** L'extraction est la première étape du processus d'apport de données à l'entrepôt de données. Extraire, cela veut dire lire et interpréter les données sources et les copier dans la zone de préparation en vue de manipulations ultérieures.
- **Transformation des données :** Il s'agit de l'action de transformer les données pour alimenter l'entrepôt de données. C'est là que le gros du processus ETL a lieu et est habituellement la partie qui prend le plus de temps. La source de données est rarement dans le format que nous voulons pour faciliter les opérations. Par conséquent, il est avantageux d'effectuer différents types de transformations de préparer la structure des données de telle sorte que les données peuvent être utilisées sans avoir besoin de ces manipulations structurelles complexes. Généralement, la partie transformation de ETL se concentre sur la consolidation des données, la correction des données, l'élimination de toute ambiguïté, l'élimination des données redondantes et le renseignement des valeurs manquantes.
- **Chargement des données :** Il s'agit de prendre la sortie de l'étape de transformation et de les placer dans l'emplacement approprié dans l'entrepôt de données. C'est une étape très délicate et exige une certaine connaissance des

structures de système de gestion de la base de données (tables et index) afin d'optimiser au mieux le processus.

1.5.4 Conception physique

La conception physique est une étape essentielle du développement des entrepôts de données et à spécifier ses caractéristiques physiques lequel est :(les types de données, la segmentation des tables, les paramètres de stockage, la spécification des clés des tables, la gestion des instances) et le choix et l'application des techniques d'optimisation (tels que la vue matérialisée, index, fragmentation, …etc.).

Où indiqué Chaudhuri dans [8]: *"The first generation of relational execution engines was relatively simple, targeted at OLTP, making index selection less of a problem. The importance of physical design was amplified as query optimizers became sophisticated to cope with complex decision support queries*". Cette amplification est due (1) le volume de données, (2) la complexité des requêtes et (3) les exigences des décideurs sur le temps de réponse des requêtes. Pour offrir une meilleure utilisation d'un entrepôt de données, donc la conception physique reste toujours une solution ouverte. Nous détaillons cette phase dans l'étape suivante.

1.5.5 Les techniques d'optimisation des entrepôts de données

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 [9]. Ces techniques appartiennent à deux catégories :

- **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.
  - la fragmentation verticale.

Dans ce projet, nous nous sommes intéressés par les structures d'optimisation redondantes, en particulier les vues matérialisées.

- **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.

A partir de là, on passé à l'optimisation des requêtes qui définir par : est une opération dans laquelle plusieurs plans d'exécutions de requêtes SQL sont examinés pour en sélectionner le meilleur.

1.6 Optimisation des requêtes

Dans le but de fournir l'algorithme d'accès à la base de données pour répondre à une requête exprimée en langage assertionnel, la requête passe par plusieurs étapes allant de l'analyse jusqu'à l'exécution. En comprenant les mécanismes internes, nous pouvons contrôler le temps qu'Oracle passe à évaluer l'ordre de jointure des tables, et améliorer les performances des requêtes en général.

Oracle permet d'analyser les requêtes et de connaître le temps et le plan d'exécution, ces informations permettent de définir ce qui ralentit l'exécution des requêtes afin de les optimiser. Une fois que les requêtes lentes détectées, lancer la commande EXPLAIN permet de comprendre l'exécution et donc connaître ou intervenir pour optimiser.

L'optimisation des requêtes consiste à :

- maximiser le parallélisme entre les entrées/sorties
- minimiser :
  - le nombre d'E/S
  - la taille mémoire nécessaire à l'exécution
  - le temps unité centrale

Un optimiseur a fondamentalement trois composants :

- **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 [13]. 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 [13].

  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 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 [13].

  - **Stratégie 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 [13].
  - **Stratégie 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 [13] :

- 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 [13].

- Si c'est les arbres de traitement touffus, les mouvements suivants sont employés : le choix de la méthode de jointure, jointure associative/commutative [13].

- **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) [13].

1.7 Optimisation multi requêtes

L'optimisation multi-requêtes est le processus de génération du meilleur plan d'évaluation combiné pour une collection de requêtes multiples

Contrairement à l'optimisation traditionnelle à une seule requête, l'optimisation à plusieurs requêtes peut tirer parti des points communs entre les requêtes, par exemple, en calculant les sous-expressions communes (c'est-à-dire, les sous-expressions partagées) une fois (via plusieurs requêtes) et en les réutilisant, ou en les partageant sur analyse des relations de disque[14].

1.8 Conclusion

Dans ce chapitre, Nous avons présente une généralité sur les 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, Aussi les 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 vu que la performance de cet entrepôt est liée à la phase de conception physique qui contribue largement dans l'optimisation physique des requêtes via l'utilisation des structures d'optimisation (par exemple notre cas d'étude vue matérialisée) qui sera détaille dans le chapitre suivant.

Chapitre 2

Optimisation des requêtes par les vues matérialisées

2.1 Introduction

Dans la première partie de ce chapitre, nous discutons sur les vues matérialisées, Comme leur nom l'indique, les vues matérialisées sont des vues dont les données sont matérialisées**,** c'est-à-dire stockées, plus précisément au problème de sélection des vues matérialisée sur les deux contextes statique et dynamique, et donnons des travaux proposées pour résoudre ces problèmes.

2.2 Comment l'optimiseur de requêtes utilise les vues matérialisées

Vous n'avez pas besoin de spécifier une vue matérialisée dans une instruction SQL pour que la vue soit utilisée. L'optimiseur de requêtes peut réécrire automatiquement les requêtes sur la table de base ou sur des vues ordinaires pour utiliser la vue matérialisée à la place [15].

Par exemple, supposons qu'une vue matérialisée contienne toutes les lignes et colonnes nécessaires à une requête sur une table de base. L'optimiseur peut décider de réécrire la requête pour utiliser la vue matérialisée, plutôt que la table de base. Cela peut considérablement accélérer une requête, en particulier si la table de base contient une grande quantité de données historiques. [15]

2.3 Les vues matérialisées

La vue est une requête nommée. Les vues matérialisées sont des tables contenant des résultats de requête. Les vues améliorent l'exécution des requêtes en pré-calculant les opérations plus coûteuses (telles que les jointures et les agrégations) et en stockant les résultats dans la base de données. Par conséquent, certaines requêtes doivent uniquement accéder à la vue matérialisée, et donc s'exécuter plus rapidement. Les vues dans le contexte OLTP ont été largement utilisées pour remplir les rôles suivants: sécurité, confidentialité, intégrité référentielle, etc. Les vues matérialisées peuvent être utilisées pour atteindre plusieurs objectifs, tels que l'amélioration des performances des requêtes ou la fourniture de données en double. [16]

Exemple : création d'une vue matérialisée pour la requête ci-dessus en SQL :

Create materialized view Somme **as**
select s_region , p_brand sum(lo_revenue) **as** sold
from lineorder, dates, part, supplier
where lo_orderdate = d_date key and lo_partkey = p_partkey and lo_suppkey = s_suppkey
group by d_year, p_brand
order by d_year, p_brand

s_region	p_brand	Sold
AMERICA	MFGR#111	1600
AFRICA	MFGR#2438	2300
EUROPE	MFGR#1240	60
ASIA	MFGR#5537	9600

Tableau 2.1 ; Exemple de vue matérialisée

Cette vue matérialisée est intéressante pour réduire le temps d'exécution de la requête suivante :

select s_region , p_brand sum(lo_revenue)
from lineorder, dates, part, supplier
where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_brand = 'MFGR#2438' and s_region = 'AFRICA'
group by d_year, p_brand
order by d_year, p_brand

Les deux principaux problèmes liés à la vue de réalisation sont:

- le problème de la sélection de la vue matérialisée.
- le problème des maintenances de la vue matérialisée.

Nous discuterons de ces deux problèmes dans les sections suivantes.

2.3.1 La sélection des vues matérialisées

Sélection d'une vue matérialisée Dans un environnement d'entrepôt de données, vous pouvez généralement isoler un ensemble de requêtes préférées. L'ensemble de vues matérialisées doit être déterminé en fonction de cet ensemble de requêtes.

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 seul de temps de maintenance à ne pas dépasser.

Deux types de PSV ont été considérés : le PSV statique et le PSV dynamique [16], [3], [17] :

- **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) [16], [3], [17].
- **Le PSV dynamique** : pensez que toutes les demandes changeront au fil du temps. Dans ce cas, l'algorithme dynamique peut devoir modifier l'ensemble des vues matérialisées en fonction de la nouvelle requête. En fait, de nouvelles vues peuvent être insérées et d'autres vues seront supprimées [16], [3], [17].

2.3.2 La maintenance des vues matérialisées

Les vues matérialisées représentent le stockage physique du résultat des requêtes fréquemment exécutées sur l'entrepôt. Quand de nouvelles données arrivent, il faut s'assurer de la cohérence du contenu de ces vues par rapport au nouvelles données en effectuant une mise à jour de la vue.

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 [16], [3] :

- **Périodiques :** les vues sont mises à jour continuellement à des périodes précises.
- **Immédiates :** les vues sont mises à jour immédiatement à la fin de chaque transaction.
- **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 [16], [3].

2.3.3 La réécriture des requêtes

La vue matérialisée de la requête réécrite est stockée sous la forme d'une table relationnelle [18]. Cela permet aux utilisateurs de les interroger, de les indexer et de les partitionner pour améliorer les performances. Après avoir sélectionné une vue matérialisée, toutes les requêtes définies sur l'entrepôt doivent être réécrites en fonction des vues disponibles. Ce processus est appelé réécriture de vue de la requête [19].

La réécriture de requêtes a attiré l'attention de nombreux chercheurs car elle est liée à plusieurs problématiques de gestion des données: optimisation des requêtes, intégration de données, conception d'entrepôt de données, etc. Le processus de réécriture des demandes a été utilisé comme technique d'optimisation pour réduire le coût d'évaluation des demandes.

Une vue matérialisée est un objet qui stocke le résultat d'une requête SELECT, c'est donc la matérialisation d'une vue.

On utilise de simples tables comme vues matérialisées, que l'on met à jour grâce à une procédure stockée ou à des triggers.

2.4 Les fonctions de coût du PSV

Pour sélectionner la vue à implémenter dans l'entrepôt de données, nous essayons rencontrez un ou plusieurs objectifs de conception et subdivisez-les en deux selon [20].

Le premier objectif est de minimiser la fonction de coût. Le second est de répondre ou de se conformer aux contraintes requises par le système ou l'utilisateur.

2.4.1 Les fonctions de coût

La plupart des méthodes incluent la minimisation dans leurs objectifs de conception de la fonction de coût. Voici un exemple de minimisation de la fonction de coût :

2.4.1.1 Coût d'évaluation de requête

Les entrées de PSV (problème de sélection de vues) sont souvent des requêtes qui doivent être satisfaire par l'entrepôt de données. L'évaluation du coût de l'ensemble de requête est égale à la somme d'évaluation du coût de chaque entrée (requête) réécrite (partiellement ou complètement) au-dessus des vues matérialisées. Cette somme peut également être pondérer, chaque poids indique la fréquence ou l'importance de la requête correspondante. Plusieurs approches visent à minimiser l'évaluation de coût de requête [20].

2.4.1.2 Coût de maintenance des vues

Le coût de maintenance des vues égal à la somme du coût de propagation de chaque changement de relation source en vues matérialisées. Cette somme peut être pondérer, chaque pondération indique la fréquence de propagation des changements de la relation source correspondante. Les expressions de maintenance peuvent être évaluées plus efficacement si elles peuvent être réécrites partiellement à travers des vues qui sont déjà matérialisé dans l'entrepôt de données. De plus, l'accès aux sources de données distantes et les transmissions de données coûteuses sont réduits. [21] a proposé d'ajouter des vues auxiliaires pour réduire le coût de maintenance des vues.

Ces vues peuvent être matérialisées de manière permanente ou transitoire. Les vues matérialisées transitoires sont employées seulement pendant le processus de maintenance et elles sont détruites après [22]. Évidemment, le maintien des vues auxiliaires engendre un coût de maintenance additionnel. Ross [23] dérivent des vues auxiliaires pour matérialiser en permanence afin de réduire au minimum le coût de maintenance des vues.

2.4.1.3 Taille totale de vue

Dans un entrepôt de données distribué qui stocke à distance des vues matérialisées, le temps de transmission des données à travers le réseau représente généralement un goulot d'étranglement. Dans ce cas, les concepteurs souhaitent minimiser la taille l'une des vues matérialisées qui répondent à toutes les demandes d'entrée. [24]

Les vues matérialisées ont une incidence sur vos coûts de stockage et de ressources de calcul.

Stockage : chaque vue matérialisée stocke les résultats de la requête, ce qui s'ajoute à l'utilisation de stockage mensuelle de votre compte.

2.5 Solutions existantes pour la sélection de vues matérialisées

Dû à l'importance du problème de sélection de vues matérialisées dans la conception des entrepôts de données et l'optimisation des requêtes, plusieurs solutions ont été proposées dans la littérature pour y résoudre .il ya des travaux qui suive l'optimisation multi-requêtes alors sélection des vue matérialisée à partir les treillis, les graphes de vue AND-OR, MVPP ou des travaux qui suive analyse syntaxique des requêtes.

2.5.1 Méthodes de sélection statique

Les solutions appartenant à cette famille supposent que la charge de travail est connue apriori. Si une évolution des requêtes est enregistrée, alors il est nécessaire de reconsidérer le problème de sélection de vues en reconstruisant les vues à matérialiser.

2.5.1.1 Les travaux de « yang et al » [25]

Les auteurs ont développé un algorithme de sélection des vues dans un contexte ROLAP statique. Les auteurs partent du principe suivant :

La principale caractéristique des requêtes décisionnelles est qu'elles utilisent souvent les résultats de certaines requêtes pour répondre à d'autres requêtes [26]. On peut tirer de cette caractéristique que les requêtes décisionnelles partagent certaines expressions. L'algorithme de Yang et al. Procède de la façon suivante :

Chaque requête est représentée par un arbre algébrique. Etant donné que chaque requête peut avoir plusieurs arbres algébriques, les auteurs sélectionnent l'arbre optimal (en fonction d'un modèle de coût). Une fois les arbres optimaux identifiés, l'algorithme essaye de trouver des expressions communes entre ces arbres (ou nœud partagé).

Finalement, les arbres sont fusionnés en un seul graphe, appelé plan multiple d'exécution des vues en utilisant les nœuds partagés identifiés.

Ce graphe a plusieurs niveaux. Les feuilles sont les tables de base de l'entrepôt et représentent le niveau 0. Dans le niveau 1, nous trouvons des nœuds représentants les résultats

des opérations algébriques de sélection et de projection. Dans le niveau 2, les nœuds représentent les opérations ensemblistes comme la jointure, l'union, etc. Le dernier niveau représente les résultats de chaque requête.

Chaque nœud intermédiaire de ce graphe est étiqueté par le coût de l'opération algébrique (sélection, jointure, union, etc.) et le coût de maintenance.

Ce graphe est utilisé pour rechercher l'ensemble des vues dont la matérialisation minimise la somme des coûts d'évaluations des requêtes et de maintenance des vues. La solution prend en considération l'existence de plusieurs expressions possibles pour une requête. Chaque nœud intermédiaire est considéré comme une vue potentielle.

2.5.2 Méthodes de sélection dynamique

A cause de la nature dynamique des entrepôts de données, les méthodes de sélection statique de vues matérialisées peuvent être très coûteuses, en temps de calcul pour qu'elles soient capables de répondre aux nouvelles requêtes, le problème situe dans l'espace de stockage limité. Pour résoudre ce problème, des solutions ont été proposées de remplacer quelques vues matérialisées qui devenues non utiles par celles plus utiles, en revanche d'autres solutions ont supposé la disponibilité d'un espace de stockage supplémentaire.

2.5.2.1 SLEMAS de Boukorca et al. (2014)

Boukorca et *al.* [27] ont proposé une architecture à trois niveaux pour la sélection dynamique de vues matérialisées, en se basant sur les contraintes d'ordonnancement des requêtes (*Query scheduling*) et d'espace de stockage disponible.

Le niveau SLEMAS de cette architecture est chargé de faire trois tâches, dont la première consiste à déterminer les liens entre les requêtes utilisateurs en utilisant un hypergraphe, où les sommets sont les jointures et les arrêtes sont les requêtes. Les auteures ont partitionné l' hypergraphe en un ensemble de partitions selon les liens entre les requêtes.

Ensuite, pour chaque partition ils ont construit un DAG (Directed Acyclic Graph) tout en calculant le coût correspondant pour définir un ordre entre les sommets du graph.

La deuxième tâche consiste à la sélection de vues matérialisées dans laquelle un bénéfice est associé à chaque nœud, et au lieu d'utiliser l'espace de stockage disponible pour

toutes les vues candidates, les auteurs ont divisé l'espace de recherche (qui représente toutes les vues candidates) en plusieurs sous-espaces de recherche chacun d'eux correspond à une composante connexe associée avec un espace de stockage alloué (l'espace est divisé entre les composante d'une façon équitable), et enfin les auteurs ont traité chaque composante d'une façon indépendante, en sélectionnant les nœuds ayant le meilleur bénéfice qui seront des vues matérialisées.

La dernière tâche consiste à l'ordonnancement des requêtes par l'identification des nœuds ayant le bénéfice maximal, puis les ordonne selon leur bénéfice, et ainsi ce bénéfice est utilisé pour calculer le poids de la requête correspondante en effectuant la somme de tous les bénéfices des nœuds participant dans la requête. Enfin, les requêtes sont ordonnées selon leur poids.

La matérialisation et la dématérialisation de vues s'effectue selon des modèles de coûts disponibles au niveau SLEMAS.

2.5.2.2 Phan et al

Les vues matérialisées ou les tables de requêtes matérialisées sont des tables auxiliaires avec des données pré-calculées qui peuvent être utilisées pour améliorer considérablement les performances des bases de données suspectes. Les travaux de recherche précédents se sont concentrés sur le meilleur ensemble de candidats [28].

Une heuristique statique courante consiste à implémenter avec empressement vue avant d'exécuter la charge de travail. Bien que cette méthode soit valable lorsque la taille du vue matérialisée défini sur le disque est petite, elle ne peut pas implémenter tous les vue et index à l'avance en fonction de la limitation du coût de maintenance du disque et de la vue réelle.

Ne profitera pas des avantages potentiellement importants de ces vues qui ne sont pas sélectionnés pour être réalisés. Dans ce travail, ils ont présenté une architecture de gestion des candidats dynamique automatisée qui peut matérialiser des vues et créer des index à la demande pendant que la charge de travail est en cours d'exécution, et utiliser la mise en cache LRU pour les gérer.

Afin de maximiser l'utilisation de vue pour l'exécution des requêtes, cette architecture fait un compromis adaptatif entre l'implémentation vue, l'accès de base aux tables et les avantages des hits dans le cache. Pour trouver la permutation de la charge de travail qui

produit le revenu net global le plus élevé, ils ont utilisé un algorithme génétique pour rechercher N! Pour la solution spatiale, afin d'éviter d'implémenter des vues rarement utilisés, nous réduisons tous les candidats.

2.5.2.3 DynaMat de Kotidis et al. (1999)

Dans [29], Kotidis et al. Ont proposé un système complet de gestion des vues un système mis en œuvre pour un entrepôt de données multidimensionnel, le système est appelé DynaMat,

Il peut être intégré dans n'importe quel entrepôt de données, il combine deux problèmes connexes les vues matérialisées, à savoir la maintenance et la sélection, en fournissant une sélection dynamique de vues matérialisées en tenant en compte les deux contraintes : le coût de maintenance de vues matérialisées et l'espace de stockage disponible.

Dans DynaMat, l'auteur utilise schéma en étoile et calculez l'ensemble de vues candidat en fonction du treillis Correspondant aux données du cube afin de déterminer les liens entre les différentes vues.

Dans cette solution les auteurs ont supposé que la charge de travail est une collection des requêtes des rangs multidimensionnelles (*Multidimenséional Range queries (MR-Queries)*), chacune d'elles peut être vue comme un vecteur n-dimensionnel $\vec{q}$ contenant les rangs dans les différentes tables de dimension, où $\vec{q}$ = {R1, R2, … , Rn} n étant le nombre des tables de dimension, et est le rang dans le domaine de la clé primaire d_i de la table de dimension, ce rang peut être :

- Un rang total : $R_i = (\min_{d_i}, \max_{d_i})$ où $\min_{d_i} et \max_{d_i} sont$ la valeur minimale et maximale respectivement de la clé d_i .
- Une seule valeur ded_i.
- Vide si D ne figure pas dans la requête.

En utilisant la notation rang multidimensionnelle (MR), les auteurs ont représenté les résultats matérialisés des requêtes MR qui s'appellent fragments des rangs multidimensionnels (MRFs), cette structure représente l'unité logique de base utilisée pour stocker les données matérialisées. Et ainsi, la solution propose une sélection dynamique des MRFs et une

maintenance des MRFs. Et par conséquent chaque fragment de rangs multidimensionnel (f) est représenté en utilisant un vecteur n-dimensionnel $\vec{f}$.

L'architecture du système DynaMat est illustrée par la figure suivante :

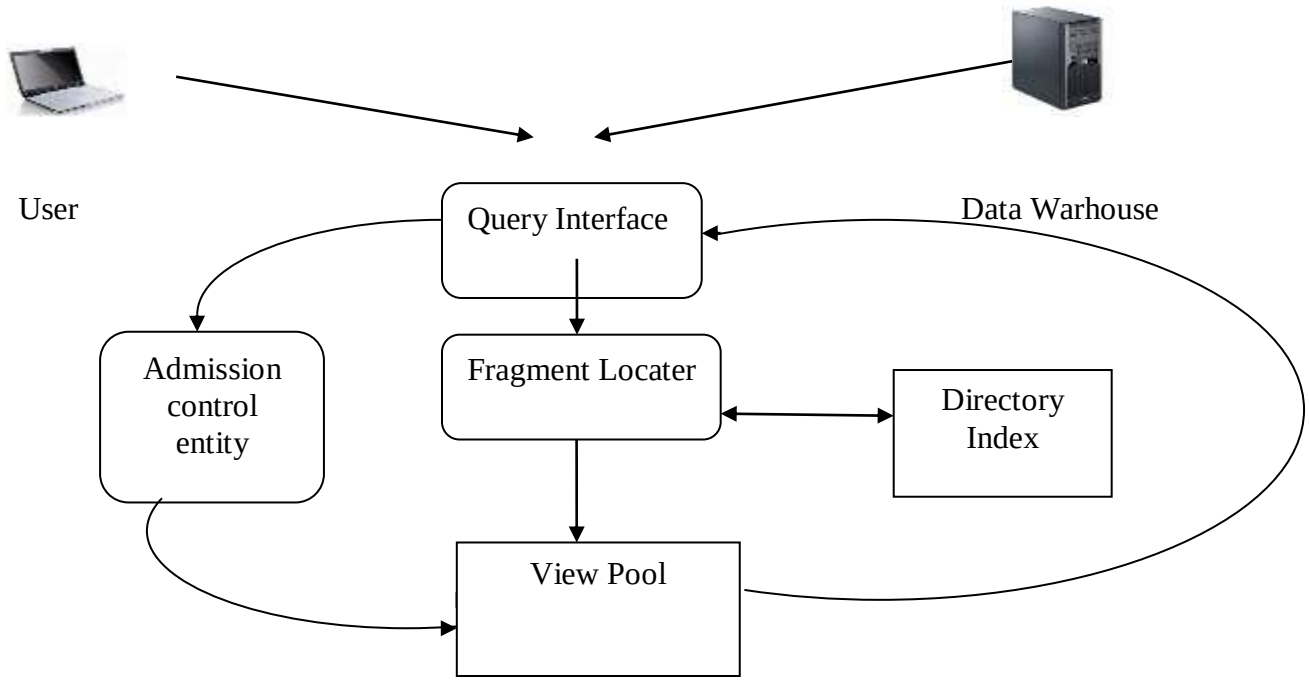

Figure 2.1: L'architecture du système DynaMat [29]

2.5.2.4 Travaux de Ravindra et al. (2013)

Ravindra et coll. [30] a proposé un Framework pour la sélection de vues matérialisées, en tenant en compte les fréquences des requêtes, le temps de traitement des requêtes et l'espace nécessaire pour chaque requête.

A partir d'un ensemble de requêtes, les auteurs ont calculé pour chaque requête la fréquence d'utilisation, le temps de traitement, l'espace de stockage nécessaire, puis ils ont sélectionné un ensemble des meilleures requêtes à matérialiser en se basant sur le coût de sélection de chaque requête en fonction des coûts de fréquence d'utilisation, de traitement et de stockage, en suit un seuil S de matérialisation est calculé en divisant la somme des coûts de sélection des requêtes sur le nombre des requêtes sélectionnées. Enfin, l'ensemble approprié

de vues est celui correspond aux vues ayant des coûts de sélection inférieur ou égal à S (l seuil) selon l'espace de stockage disponible.

En raison du changement de fréquence des demandes, Calculer la quantité de stockage pour chaque vue matérialisée en fonction de la fréquence d'utilisation requête et son espace de stockage à remplacer par les visiteurs à haute fréquence ont une faible fréquence d'accès et un grand espace de stockage droits d'accès élevés et petit espace de stockage.

2.5.2.5. Travaux de Zhou et al. (2007)

Une nouvelle stratégie de matérialisation dynamique des vues a été proposée par Zhou et *al* [31], dans laquelle les auteurs ont proposé de matérialiser seulement les tuples pertinents au lieu de matérialiser toute la vue afin de réduire l'espace de stockage nécessaire, ainsi que le temps de maintenance. Zhou et *al.*

Ont proposé de créer des vues matérialisées dont leur contenu est contrôlé par des tables de contrôle en introduisant la clause EXISTS dans l'expression de définition des vues.

Le changement des tuples à matérialiser s'effectue d'une façon dynamique au moment de l'exécution des requêtes par la modification du contenu des tables de contrôle soit d'une façon manuel, ou automatique en utilisant une stratégie de cache interne afin de donner les tuples de la vue les plus fréquents.

Exemple

Soit *Q*1 une requête écrite comme suit :

*Q*1: SELECT p_partkey, p_name, p_retailprice, s_name, s_suppkey, s_acctbal, l_quantity, l_extendedprice

FROM part, lineitem, supplier

WHERE p_partkey = l_partkey and s_suppkey = l_suppkey

AND p_partkey = @pkey

Les méthodes traditionnelles ont défini une vue de jointure sur les relations de bases pour répondre à la requête *Q*1 cette vue est écrite comme suit :

CREATE VIEW *V*1 AS

SELECT p_partkey, p_name, p_retailprice, s_name, s_suppkey, s_acctbal, l_quantity, l_extendedprice

FROM part, lineitem, supplier

WHERE p_ partkey = l_partkey and s_suppkey = l_suppkey

Il est clair que la vue V1 contient des tuples additionnels qui ne sont pas utiles pour répondre à *Q*1 qui occupent un espace de stockage additionnel, pour résoudre ce problème les auteurs ont proposé de matérialiser seulement les tuples utiles, par l'utilisation de la clause EXISTS et une table de contrôle qui peut être une table régulière ou bien une autre vue matérialisée, comme suit:

CREATE TABLE HOTSPOT (hotpartkey int primary key)
CREATE VIEW *DV*1 AS
SELECT p_partkey, p_name, p_retailprice, s_name, s_suppkey, s_acctbal, l_quantity, l_extendedprice
FROM part, lineitem, supplier
WHERE p_partkey = l_partkey and s _suppkey = l_suppkey
AND **EXISTS** (select * from hotspot hs where p_partkey=hs.hotpartkey)

Et donc la vue DV1 ne contient que les tuples qui apparient avec ceux dans la table

HOTSPOT qui réduit le nombre de ces tuples d'une manière considérable.

Afin d'utiliser ce type de vue matérialisée pour répondre aux demandes des utilisateurs, l'auteur a modifié l'algorithme de correspondance des requêtes pour pouvoir vérifier lors de l'exécution si la vue matérialisée répond à ces demandes que ce soit basé sur la requête et la résolution de la vue matérialisée.

Cet algorithme est exécuté en trois étapes, les deux premières étapes et classé troisième à l'exécution. Si la vérification échoue, la demande sera évaluée Interroge la table de base.

2.6 Conclusion

Le problème de sélection des vues matérialisées a été considéré comme 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 les deux cas statique et dynamique.

Chapitre3

Conception

3.1 Introduction

A travers l'état de l'art présenté dans les chapitres précédents, nous avons constaté l'importance d'optimiser les requêtes très couteuses définit sur le entrepôt de données via d'utilisation des structures d'optimisation et parmi ceux-ci nous avons mentionné que nous nous intéressons aux vues matérialisées sélectionnées de façon dynamique pour réaliser cet objectif, nous présentons dans ce chapitre notre approche de sélection ces vues dynamique en basant sur le travail de référence réalisé par Phan et al []. Nous présentons le modèle de coût utilisé dans notre approche, pour évaluer le coût de la charge de requêtes avec et sans les vues matérialisées.

3.2 Modèle de coût utilisé dans l'évaluation

Un modèle de coût c'est une fonction mathématique pend en entrée un ensemble des paramètres recensés à partir de la charge de requêtes et les statistiques sur l'entrepôt de données, la taille du support de stockage tel que (la taille de page d'oracle PS, facteur d'échelle qui permet de déterminer la taille de base de données SF =1 c'est-à-dire taille 1go, Cardinalité de la relation, Taille de tuple dans la relation, Taille moyenne D'attribut.etc...) et d'autres paramètres comme sont présentées dans le tableau N° et il fournit en sortie une mesure telle que le cout Entrée/sortie, et le cout énergétique, coût de communication. Dans notre travail nous avons utilisé le cout Entrée/sortie.

Paramètres	Sélectivité $fs_{,p}$
PS	Taille de pages
$\|\|R\|\|$	Taille de relation en pages
$\|R\|$	Cardinalité de la relation R
$L_{R,C}$	Taille moyenne d'attribut c dans la relation R
L_R	Taille de tuple dans la relation R
$\|R,C\|$	Nombre de valeurs distincts d'attributs c dans la relation R
fs_R	Facteur de sélectivité pour la sélection fs sur la relation R
$f\pi_R$	Facteur de sélectivité pour la projection $f\pi$ sur la relation R
$fj_{R,S}$	Facteur de sélectivité pour la jointure sur les relations R et S

Tableau 3.1 : Les paramètres de modèle de coût

Le coût E/S d'une requête suivant son plan physique est la somme du coût de chaque opération (sélection, jointure, projection) telle que chacun transmet son résultat à l'opération parent. Le coût total de la charge est donné par la somme des coûts des requêtes exécutées :

$$Coût_{Total} = \sum_{i=1}^{n} Coût(Q_i)$$

Pour arriver mesurer ce $Coût(Q_i)$, on doit calculer le résultat délivré par chaque opération (sélection, jointure, projection) en utilisant des fonctions d'estimation de cardinalité qui seront détaillé par la suite.

3.2.1 Estimation de la cardinalité des résultats intermédiaires

La cardinalité de résultat intermédiaire représente le nombre l'enregistrement (tuples) produit par chaque opération algébrique .Nous présentons les formules de base utilisée par un SGBD pour estimer la cardinalité des résultats intermédiaires de chaque opération comme suit:

- **Cardinalité de la sélection :**

La cardinalité de cette opération est calculée comme suit :

$$\left|\sigma_{\mathrm{p}}(R)\right| = fs_p * |R|$$

Tel que fs_p est appelé facteur de sélectivité du prédicat *p*, qui est défini comme le nombre de tuples satisfaisant le prédicat, et sa valeur est comprise entre 0 et 1.Le tableau N° résume les différentes formules utilisées pour estimer fs_p avec A et B désignent des attributs, val,val_1 et val_2 dénotent des constantes.

Prédicat p	**Sélectivité fs_p**
$\neg(p)$	1-fs_p
$p_1 \wedge p_2$	$fs_{p_1} * fs_{p_2}$
$p_1 \vee p_2$	$fs_{p_1} + fs_{p_2} - fs_{p_1} * fs_{p_2}$
$A = val$	$1/_{\lvert A\rvert}$
$A = B$	$1/_{(\max(\lvert A\rvert, \lvert B\rvert)}$
$A > val$	$(max_A - val)/(max_A - min_A)$
$A < val$	$(val - min_A)/(max_A - min_A)$
$val_1 \leq A \leq val_2$	$(val_2 - val_1)/(max_A - min_A)$

Tableau 3.2 : Estimation de la sélectivité des prédicats

- **Cardinalité de projection**

La cardinalité du résultat d'une projection un attribut ou un ensemble d'attributs dans la relation *R*est la cardinalité de *R* elle-même :

$$\pi_A(R) = |R|$$

- **Cardinalité de jointure**

Dans notre projet, nous considérons la jointure naturelle entre deux relations $R(A,B)$et $S(B,C)$ ou B est l'attribut de jointure. Il y a plusieurs façons pour estimer la cardinalité de la jointure. Dans notre implémentation cette cardinalité est définit comme suit :

Card $(relation_1 \bowtie relation_2) = SF_j$ * Card $(relation_1)$* Card $(relation_2)$

Ou $SF_j = \frac{\text{Card}\,(relation_1 \bowtie relation_2)}{\text{Card}\,(relation_1) * \text{Card}\,(relation_2)}$

Une fois les cardinalités des opérations algébrique sont estimés. Nous calculons le cout E /S de chacun exprimé en fonction de nombre de pages en utilisant la formule suivante :

$\text{Cout}(\text{op}) = \frac{\boldsymbol{L_R} * |R|}{\boldsymbol{PS}}$ tel qu'op représente une opération algébrique (sélection, projection, jointure)

- **Coût de matérialisation**

C'est le coût de l'opération jointure, donné comme suite

$$fj_{R,S} * \text{PS} * \frac{L_{R\,S}}{L_R * L_S} * \|R\| * \|S\| + \|\boldsymbol{R}\|$$

3.3 L'approche de sélection des vues dynamique

Notre approche de sélection des vues dynamique passe par les étapes suivantes :

Etape 1 : Notre approche prend en compte l'interaction des requêtes qui représente les nœuds partagés entre les requêtes telle que la sélection, jointure, Etc., notre charge des requêtes utilisé dans l'implémentation est représentée par une structure de données sous forme MVPP, qui définit comme suit :

Multi-View Processing Plan (MVPP),c'est une structure de données Multi-View Processing Plan (MVPP) est un graphe acyclique dirigé dans lequel les nœuds racine représentent les requêtes, les nœuds feuille sont les tables de base, et les nœuds intermédiaires sont les opérateurs algébriques tels que la sélection, la jointure, la projection, etc. Cette structure a été définie par yang et al [ ].Un simple MVPP est montré dans la figure suivante :

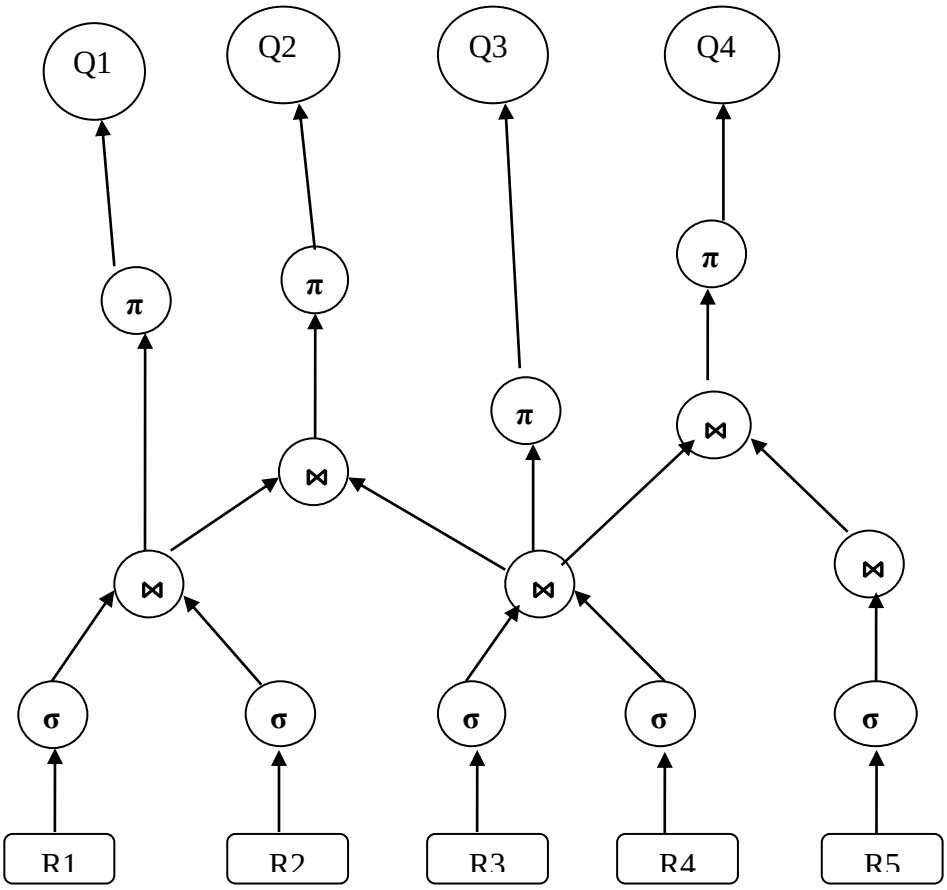

Figure 3.1 : Exemple de MVPP

Ce MVPP est utilisé comme une structure d'entrée de notre approche.

Etape2 : Extraction des nœuds de jointures et de sélection à partir MVPP, et calcule le bénéfice de chaque nœuds de jointure, en utilisant la formule :

Bénéfice c'est une fonction ou valeur qui permet de mesurée le gain de performance qui peut être réalisé si la vue est matérialisée.

Etape 3 : une fois les bénéfices des nœuds de jointure sont calculés, nous appliquons un algorithme déterministe de sélection des vues matérialisées, qui permet de créer la liste des vues qui ont un bénéfice positif, et les vues avec un bénéfice négatif sont élaguées, puis il les ordonne de façon décroissante suivant leurs bénéfices.

Voir le pseudo code de l'algorithme :

Algorithme 1 ensemble réduit

1: FUNCTION Prune

2: IN: W, MVPP

3: IN: M, les vues matérialisée candidat

4: OUT : réduit, ensemble réduit

5: BEGIN

6:

7:réduit:= M

8: Int bénéfice := GeneticAlgorithm (W,réduit)

9: while (benefice < 0) and (réduit.size> 1) do

10: Sort trié élagué par bénéfice décroissant

11:réduit := moitié supérieure de réduit

12:bénéfice := GeneticAlgorithm (W, réduit)

13: end while

14: return réduit

15: END

Etapes 4 : notre approche est guidée par la contrainte de coût de stockage donc il ne sera pas possible de matérialiser toutes les vues matérialisées alors la solution est construire l'ensemble des vues matérialisée réduit à partir de l'ensemble initial trouvé dans l'étape précédente, dans cette étape nous avons adopté le travail de Phan et al [28] qui proposée le modèle dynamique réduit, en utilisant un algorithme de recherche (voir l'algorithme1) qui exploite un algorithme génétique.

Sachant que la différence majore entre notre approche et celle de Phan et al, que nous prenons en considération l'interaction de requêtes via le MVPP.

Ce modèle dynamique réduit consiste à créer un sous ensemble des vues réduit à partir un ensemble des vues candidats initial V, Si une requête demande une vue alors on va vérifier s'il existe dans ensemble élague alors sera matérialisée c'est t-a dire sauvegarder dans la cache si existe d'espace, s'il n'existe pas d'espace on utilise le principe LRU pour libérer l'espace .

Si la vue demandée par la requête n'existe pas dans l'ensemble réduit donc la requête sera exécutée sans la vue demandée. Le schéma suivant montré le principe de ce modèle

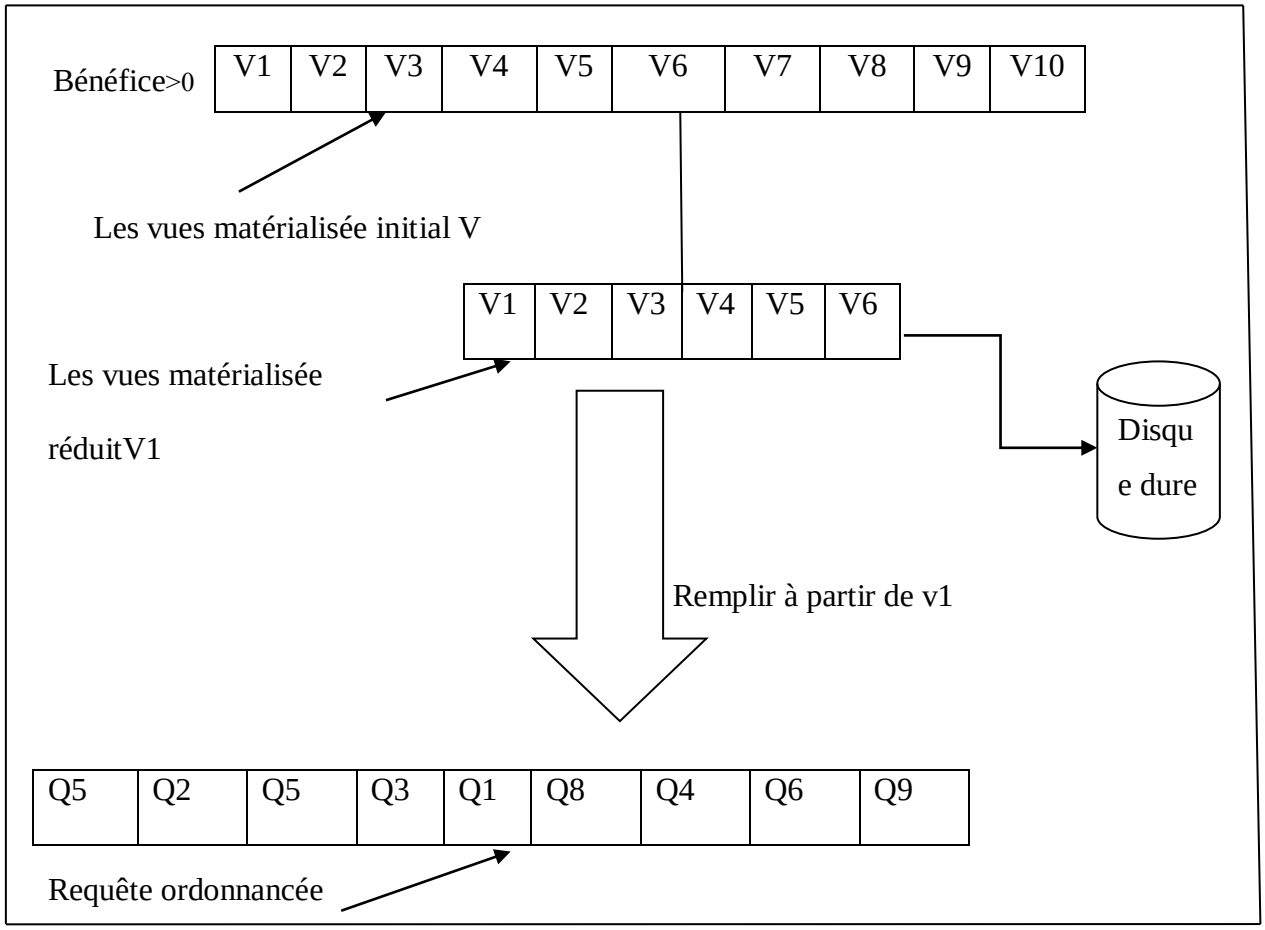

Figure 3.2 : modèle dynamique réduit

Alors que cette approche soit valable, lorsque la taille du vue matérialisée défini sur le disque est petite, il ne sera pas être capable de matérialiser toutes les vues matérialisées vu les contraintes (telles que les limites d'espace disque, afficher les coûts de maintenance, ou le temps passé sur la correspondance lors de la compilation des requêtes) dans [28], ils ont représenté la charge de requêtes par une file d'attente de N requêtes ordonnancées qui sera utilisé dans l'algorithme génétique , Nous présentons par la suite le principe de cet algorithme génétique.

3.4 Algorithme génétique

Est un algorithme d'optimisation basé sur les principes d'évolution des organismes dans la nature : l'algorithme cherche à générer les meilleurs éléments d'une population en combinant les meilleurs gènes ensemble.

Étant donné une permutation de charge de travail de l'espace de recherche N! Requête, le problème est de trouver l'ordre de requête optimale qui produit le bénéfice des vues le plus élevé. Pour rechercher cette solution d'espace, les auteurs utilisés des heuristiques de recherche d'algorithme génétique (GA), qui ont [28] trouvé un compromis entre maximiser le cache, minimiser l'implémentation des vues et minimiser l'accès de base aux tables.

La sortie de GA est une instruction de requête, qui contrôle également indirectement la mise en œuvre et le déclin des instructions vue/index (déterminé par le comportement de LRU).

Une AG émule la sélection naturelle darwinienne en faisant concurrencer les membres de la population au cours des générations successives afin de converger vers la meilleure solution, les chromosomes évoluent à travers plusieurs générations d'adaptation et de sélection.

3.5 Fonction de fitness

Le but de ces algorithmes génétiques est d'optimiser une fonction prédéfinie, appelée Fonction objective, ou fitness ; ils travaillent sur un ensemble de solutions candidates, appelées "Population" d'individus ou chromosomes, Ces derniers sont constitués d'un ensemble d'éléments, appelés "gènes", qui Peuvent prendre plusieurs valeurs, appelées "allèles".

Une composante GA importante est la fonction d'évaluation [28].Étant donné un chromosome particulier représentant une permutation de charge de travail, la fonction calcule de manière déterministe le bénéfice net de l'utilisation des candidats gérés par un cache LRU pendant la charge de travail. Il y a trois clés branches dans la fonction d'évaluation, à savoir

si un vue est dans l' ensemble des vues candidat qui a été réduits par la taille, s'il y avait une cache Hit, et s'il y avait un manque de cache. Chaque de ces événements entraîne la sanction appropriée.

3.6 Codage de chromosome

Le premier pas dans l'implantation des algorithmes génétiques est création d'une population d'individus initiaux. Chaque individu de la population est codé par un chromosome.

C'est-à-dire Une population est un ensemble de chromosomes. Chaque chromosome code un point de l'espace de recherche .donc, L'efficacité de l'algorithme génétique dépendre du choix du codage d'un chromosome.

Premièrement, création d'un ensemble aléatoire de chromosomes pour la population. Les chromosomes sont évalués (hachés) selon un certain métrique, et choisis les meilleurs pour être parent.

Dans notre cas, l'évaluation produit le bénéfice net de l'exécution de la charge de travail, accéder aux vue et matérialiser les vues dans le cache. Les parents se recombinent pour produire des enfants, simulant croisement, et parfois une mutation peut survenir qui produit de nouvelles caractéristiques qui n'étaient disponibles chez aucun des parents ,Les enfants sont classés par fonction d'évaluation, et Le meilleur sous-ensemble d'enfants est choisi pour être les parents de la nouvelle génération, pour simuler la sélection naturelle. La boucle de génération se termine une fois la condition d'arrêt remplie, ils ont choisi de finir après 1000 générations, car cette valeur était Expérimentalement le meilleur compromis entre temps d'exécution et rigueur de recherche.

3.7 Les opérations génétiques

Les opérateurs jouent un rôle prépondérant dans la possible réussite d'un AG. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement et de mutation.

L'algorithme génétique tient pour partie au fait que chacun de ces opérateurs agit selon divers critères qui lui sont propres (valeur sélective des individus, probabilité d'activation de l'opérateur, etc.).

Voici les fonctions principales d'un algorithme génétique :

3.7.1 Croisement

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus). Tout d'abord, deux individus, qui forment alors un couple, sont tirés au sein de la nouvelle population issue de la reproduction. Puis L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus).

Tout d'abord, deux individus, qui forment alors un couple, sont tirés au sein de la nouvelle population issue de la reproduction. Puis une sous-section contiguë choisie au hasard de le premier parent est copiée sur l'enfant (choisi comme la pièce entre deux au hasard tranches choisies), place cette sous-section au début de l'enfant, puis tous les autres éléments du deuxième parent (qui n'ont pas déjà été pris, de la sous-section du premier parent) sont ensuite copiées sur l'enfant dans l'ordre d'apparition. Figure suivant montré le principe de l'opération.

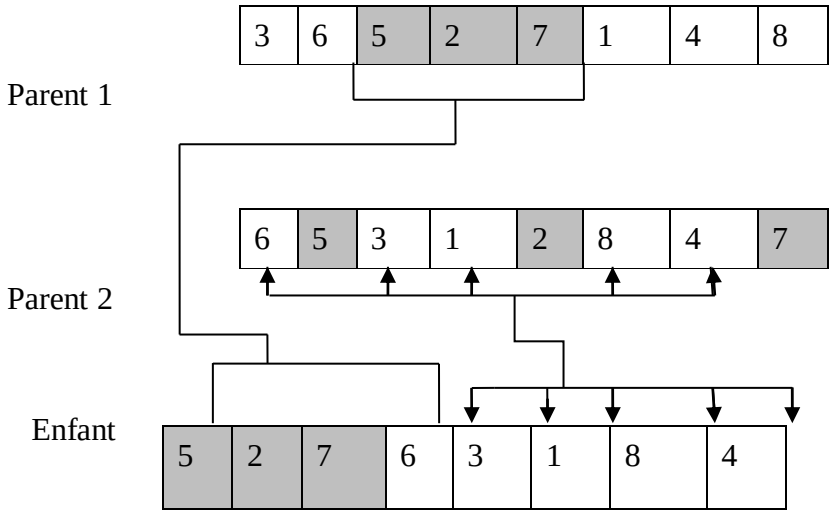

Figure 3.3 : Opération de Croisement

3.7.2 Mutation

C'est un opérateur qui permet de sélectionner deux gènes et de les changer la position entre eux, Figure suivant montré le principe de l'opération.

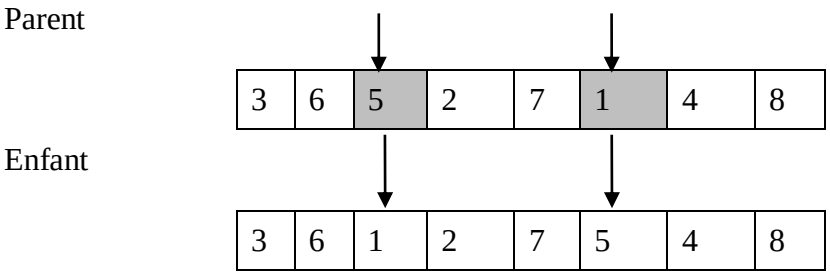

Figure 3.4 : opération de mutation

3.8 Conclusion

Dans ce chapitre nous avons présenté le modèle de coût et notre approche de sélection ces vues dynamique, et basée sur la présentation de l'algorithme génétique et ces opérations. Tous ça seront implémentés au chapitre suivant pour résoudre notre problème traité.

Chapitre 4

Implémentation

4.1 Introduction

Dans le contexte d'entrepôt de données, les requêtes contient un ensemble des opérations, sont très couteuse et nécessitent d'une optimisation.

Dans ce chapitre, nous avons implémenté notre problème de sélection des vues matérialisées Nous présentons notre environnement de développement, et notre application avec des interfaces, puis nous mettons des expérimentations pour montre notre travail.

4.2 Environnement de développement

L'application est développée par le langage de programmation Java, et on utilise Netbeans, 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.
- **NetBeans** : est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web).
- **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.

4.3 Notre base de données

Afin de valider l'approche proposée pour trouver solution pour la sélection de vues matérialisées, nous avons utilisé le banc d'essai Star Schéma Benchmark (SSB) qui est le schéma en étoile du banc d'essai. Le SSB 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, Son schéma logique est illustré dans la figure ci-dessous :

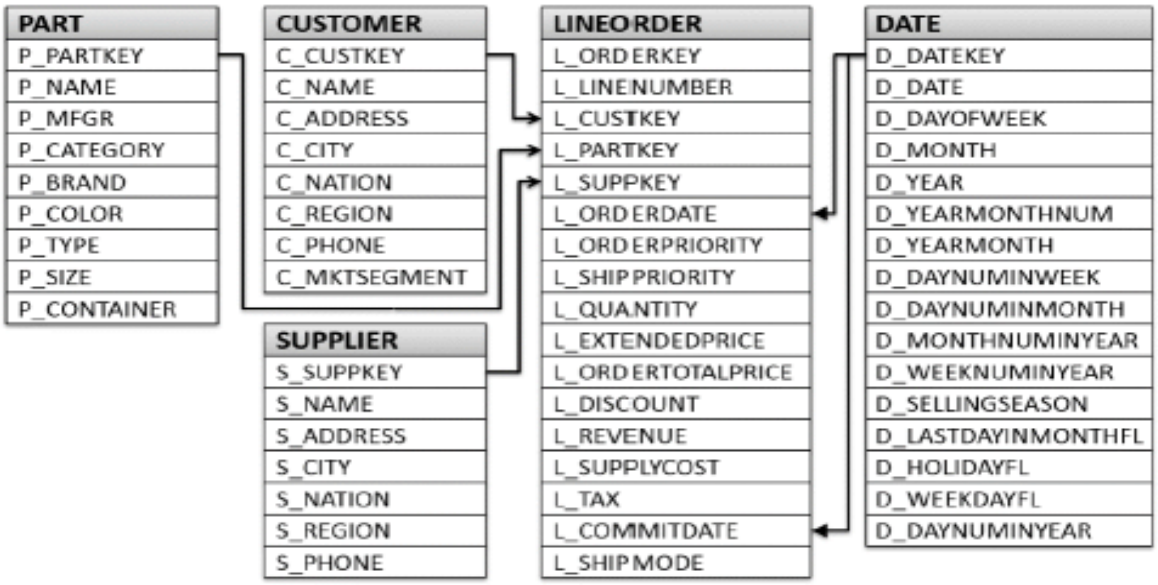

Figure 4.1 : Schéma logique du SSB [32].

4.4 Jeu de données

Nos expérimentations ont été réalisées sur le benchmark SSB (Star Schéma Benchmark).

La table 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 requêtes.

Table	Type	Cardinalité
Lineorder	Faits	6000000
Dates	Dimension	2556
Customer	Dimension	30000
Supplier	Dimension	2000
Part	Dimension	200000

Tableau 4.1 : Cardinalité des tables de SBB [32].

4.5 Présentation de l'application

Nous présentons premièrement l'application de sélection des vues matérialisées que nous avons développée en Java sous NetBeans.

Notre interface contient 4 boutons et zone pour afficher les résultats :

- Bouton «Sélectionner MVPP»
- Bouton « Générer vues initiales»
- Bouton «Elaguer»
- Bouton «Exécution»

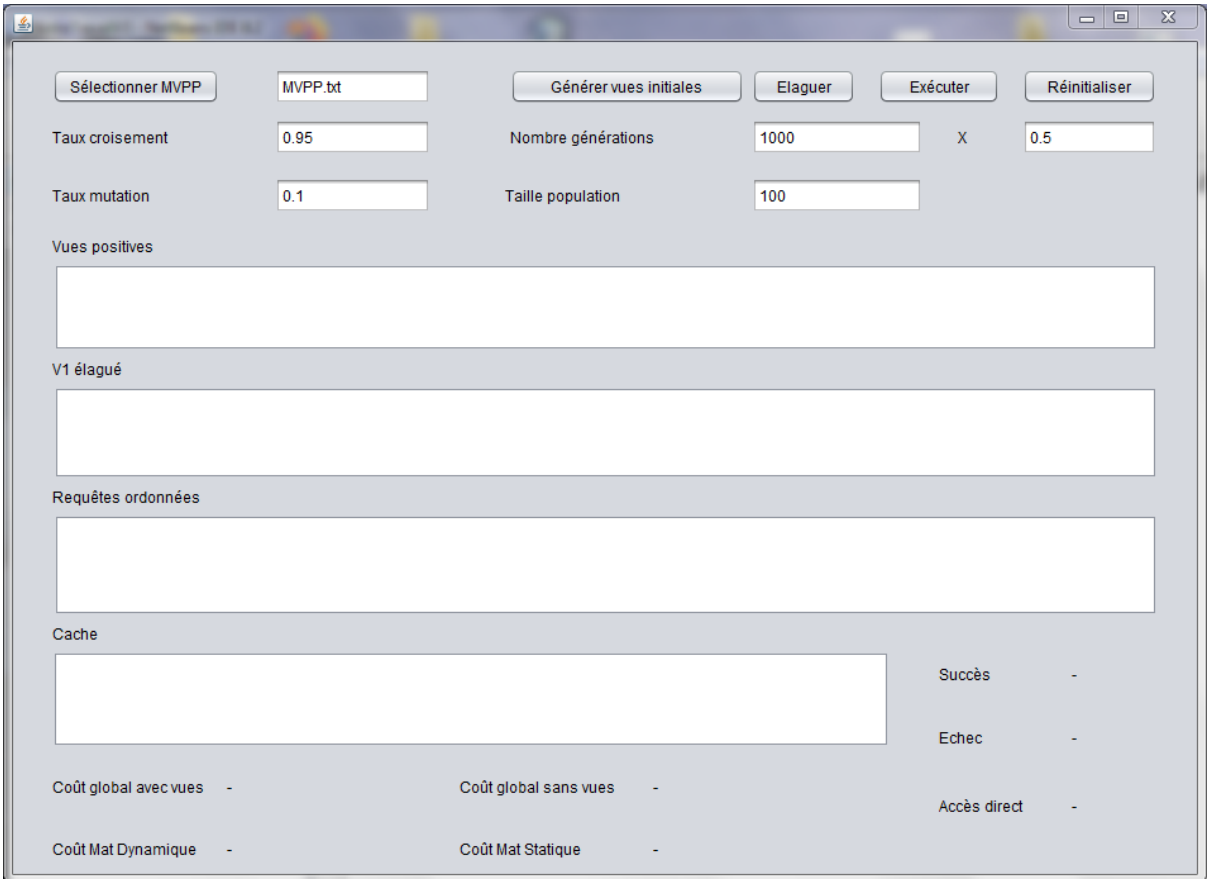

Figure 4.2 : interface générale.

On prend des probabilités pour chaque opération (mutation, croisement) qui ont permet de réussite d'un algorithme génétique qui sera appliqué sur l'ensemble des requêtes pour bien ordonnancé, et aussi un nombre de générations =1000 et taille de population =100, on prend aussi une valeur X pour initialiser la taille de cache en fonction de nombre de pages telle que cette valeur représente un pourcentage qui peut être changé selon la taille de charge MVPP c'est -à-dire selon la somme globale des tailles des vues dans l'ensemble initiale.

4.5.1 Les étapes d'exécutions

Etape 1 : cliquer sur bouton «sélectionner MVPP » permet de choisir le fichier MVPP qui contient ensemble des requêtes de différente taille pour l'exécutée.

Etape 2 : cliquer sur bouton « Générer vues initiales » permet d'afficher l'ensemble des vues candidates initiales qui ont un bénéfice positif ordonné de façons décroissante selon ces bénéfices, tel que chaque vue représenté par un identificateur.

Etape 3 : cliquer sur bouton « Elaguer » permet d'afficher un sous ensemble à partir l'ensemble précèdent avec l'algorithme réduit.

Etape 4 : cliquer sur bouton «exécute » permet de lancer l'exécution et afficher l'ensemble des requêtes de façon ordonnancée selon l'ensemble précédent, puis remplir la cache par les vues réduit utilisé par chaque requête, une fois les vues insérées dans le cache, les résultats suivant sera affichées :

- Succès : signifie que la vue demandé par requête existe dans la cache.
- Echec : signifie que la vue demandé par requête n'existe pas en cache, vérifiée sa existante dans l'ensemble réduit puis insérer dans le cache pour afficher « Succès ».
- Accès direct : signifie que la vue demandé n'existe pas en ensemble réduit, donc la requête doit accéder à table de base.
- Coût globale sans vue permet d'afficher la somme des coûts d'exécutions des requêtes sans les vues matérialisées.
- Coût globale avec vue permet afficher somme des coûts d'exécutions des requêtes avec les vues matérialisées insérée dans la cache.

Tous les résultats de ces étapes sont affichés sur la figure suivant :

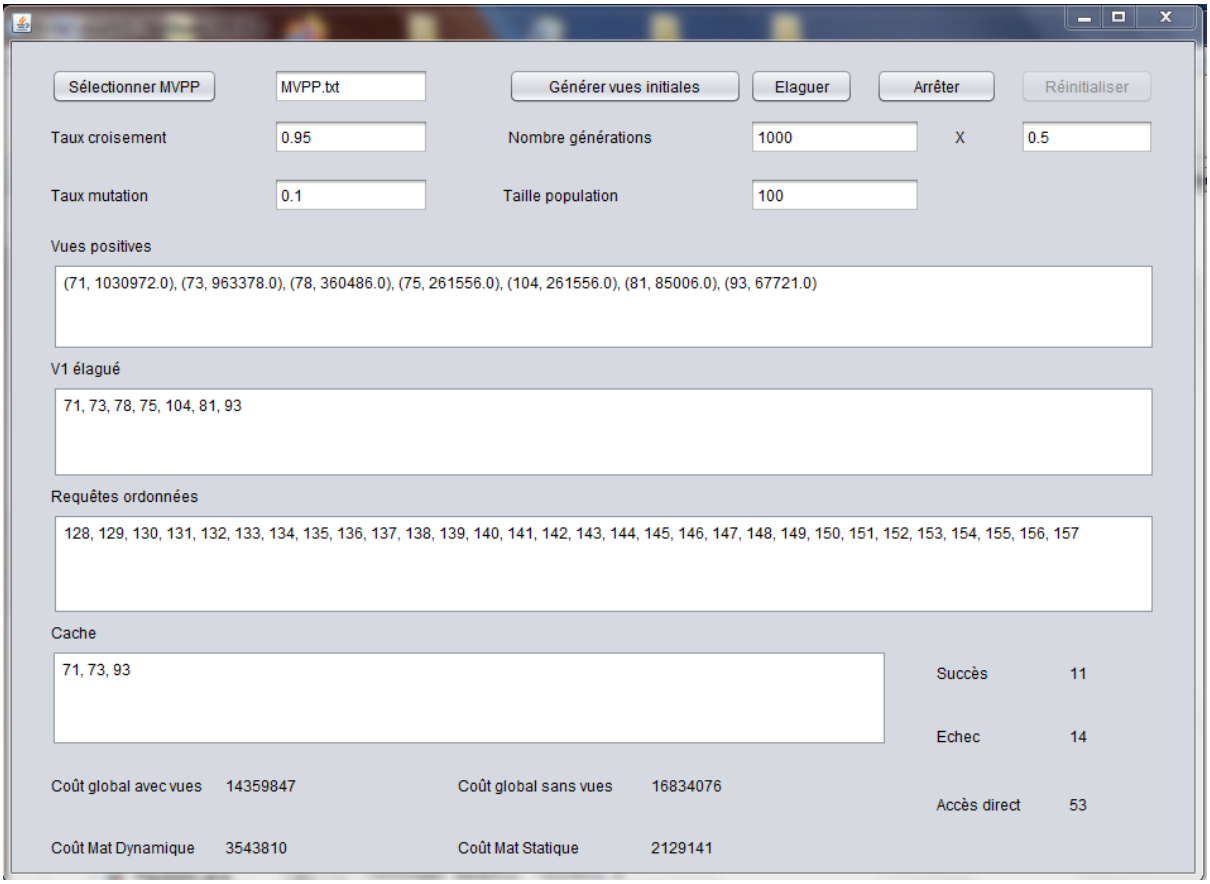

Figure 4.3 : exemple de résultat pour 30 requêtes.

Le résultat affiché dans figure confirme que l'exécution des requêtes avec vue est performante que l'exécution des charges de requêtes sans vue telle que ce dernier enregistre un coût d'exécution global plus élevé ,que l'autre enregistre un gain de performance .

4.5.2 Expérimentation

Dans cette partie, nous mettons en test certaines expérimentations afin de valider notre étude, nous testons les coûts de la charge MVPP de différentes tailles 10, 30,100, 200, avec et sans vue matérialisée. Les résultats obtenus sont donnés dans les figures 4 et 5.

Les résultats de la figure 4 comparent les coûts d'exécution globale des déférentes taille de MVPP avec et sans les vues matérialisées.tel qu'il y' une grande déférence entre les coûts, tel que le coût d'exécution avec vue est plus petit par rapport le coût sans vue, ce qui confirme l'efficacité d'algorithme pour la sélection des meilleures vues qui minimise le coût d'exécution et rapidité l'exécution des requêtes.

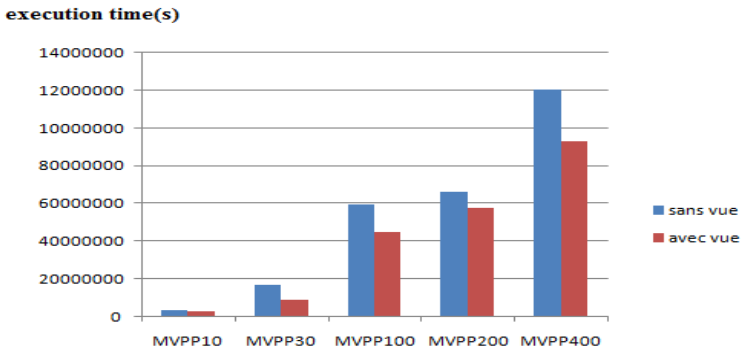

Figure 4.4 : Coût d'exécution des requêtes avec et sans vue.

Les résultats de la figure 4.4 comparent les coûts de matérialisation en cas statique et dynamique (notre cas d'étude), les résultats montrent qu'il y a une déférence entre les coûts, tel que le coût de matérialisation en cas dynamique (en utilisant les vues matérialisées) est plus élevé que la statique ,ce qui confirme l'importance et l'efficacité des vues matérialisées, donc le cas dynamique maximise le coût de matérialisation et minimise l'accès au table de base .

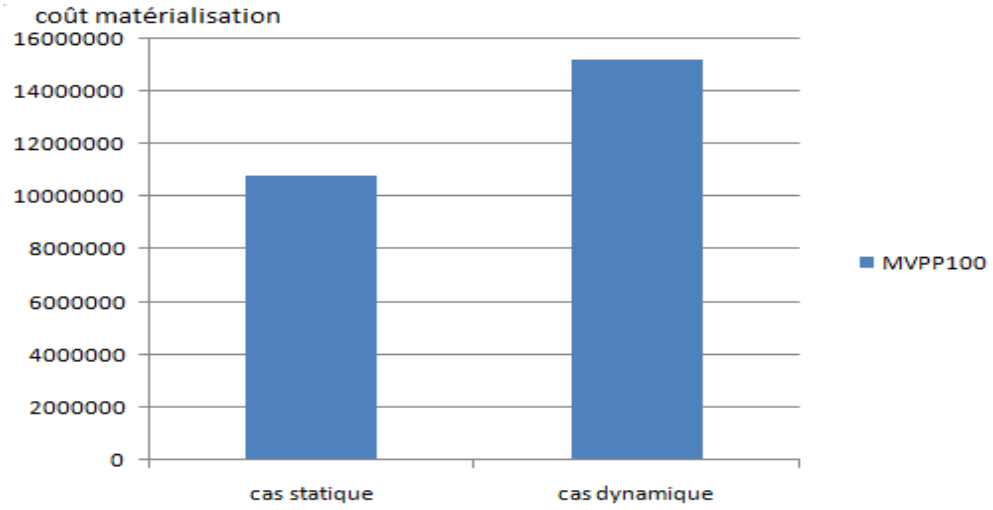

Figure 4.5: Coût de matérialisation de100 requêtes en cas statique et dynamique.

D’après notre expérimentation, nous concluons ce qui suit :

- Les modèles dynamiques offrent un net bénéfice en permettant aux vue d'être matérialisés et supprimés à la demande.
- Les vues initiale sont matérialisées et déposées via le cache(en cas dynamique) et certaines requêtes nécessitant des données doivent uniquement accéder à la vue matérialisée, et donc s'exécuter plus rapidement, et minimise l’accès à la table de base
- Le modèle dynamique augmente nombre élevé de matérialisations et moins accès à la table de base.

Conclusion

Nous avons appliqué dans ce chapitre une nouvelle approche, c’est une amélioration de l’approche proposée dans le travail [28] pour résoudre le problème de sélection des vues matérialisées dynamique.

Cette approche se base sur l’application de l’algorithme génétique pour résoudre notre problème traité, nous avons présenté notre application avec d’exemple de charge MVPP de différents tailles, puis nous testons notre application avec expérimentations qui confirme l’efficacité de notre algorithme.

Conclusion Générale

Un entrepôt de données est un ensemble de vues matérialisées, par conséquent, sa maintenance consiste à maintenir cet ensemble de vues. En effet, ce dernier est le résultat d'une sélection des différents résultats intermédiaires des différentes requêtes, dont le but est d'y accélérer. Le résultat des requêtes analytiques peut être critique dans l'aide à la décision, leur information doit être la plus récente.

Pour cette raison la sélection de l'ensemble de vues devient une propriété importante.

En fait, les solutions traditionnelles de sélection de vues matérialisées, considèrent l'ensemble de vues matérialisées connu à priori, par conséquent ces vues sont multi-relations, du fait qu'elles matérialisent des jointures entre des tables volumineuses afin d'améliorer le temps de réponse aux requêtes analytiques, ce qui rend ces solutions coûteuses soit en terme de temps de maintenance, de nombre des messages échangés entre l'entrepôt et les sources de données, ou en terme d'espace de stockage .

Dans ce travail, nous avons introduit tout d'abord, des généralités sur les entrepôts de données, puis nous nous sommes focalisés sur les problèmes de sélection de vues matérialisées. Plusieurs solutions ont été proposées dans littérature pour résoudre les deux problèmes de sélection de vues matérialisées, nous avons décrit quelques solutions qui abordent ces deux problèmes, puis nous présentons notre approche de sélection ces vues dynamique qui basant sur l'amélioration de travail Phan [28], après nous avons implémenté notre problème de sélection des vues matérialisées et mettre des expérimentations pour confirme l'efficacité de notre travail.

Perspectives : dans l'avenir nous aimerions améliorer notre approche par plusieurs fonctionnalités :

Comparer notre approche avec travail Phan [28] pour tester l'efficacité de notre algorithme

Reference Bibliographies

[2] Polraj Ponia, Data Warehousing Fundamentals for IT Professionals, The second edition John Wiley and Sons, 2011.

[3] 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.

[4] 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.

[5] 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.

[6]: An IT Mandate, Providing OLAP to User-Analysts, Edition , 19 avril 2012.

[7] SernaEncinas Marıa Trinidad. Entrepôts de données pour l'aide `a la décision médicale: conception et expérimentation. Networking and Internet Architecture.Université Joseph-Fourier - Grenoble I, 2005.

[8] S. Chaudhuri and V. Narasayya. Self-tuning database systems: A decade of progress. *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 3–14, September 2007.

[9] 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.

[10] Donald Burleson, ORACLE High-Performance SQL-Tuning , Mars 2003.

[11] Laurent Navarro, Optimisation base de données, Pearson Education France ISBN: 978-2-7440-4156-3.

[12] Oracle France - Support OR, Support de cours Référence : ntsql001.

[13] 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.

[16] 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.

[17] 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.

[18] R. G. Bello, K. Dias, A. Downing, Feenan Jr. J., W. D. Norcott, H. Sun, A. Witkowski, and M. Ziauddin. Materialized views in oracle. Proceedings of the International Conference on Very Large Databases, pages 659- 664, August 1998.

[19] D. Srivastava, S. Dar, H. Jagadish, and A. Y. Levy. Answering queries with aggregation using views. Proceedings of the International Conference on Very Large Databases, pages 318{329, September 1996.

[20] Harinarayan, V., Rajaraman, A., & Ullman, J.Implementing Data Cubes Efficiently**.** ACM SIGMOD International Conference on Management of Data (SIGMOD). Montreal, Canada,pages 205-216. 1996.

[21] Lee, M. and J.Hammer. speeding up warehouse physical design using a randomized algorithm. Int J. Cooperative inform. Syst, 10: 327-353, 2001.

[22] Mistry H., Roy P., Sudarshan S. & Ramamritham K. (2001). Materialized View Selection and Maintenance Using Multi-Query Optimization. ACM SIGMOD

International Conference on Management of Data (SIGMOD). Santa Barbara, California,
307-318, 2001.

[23] Ross, K., Srivastava, D., & Sudarshan, S. Materialized View Maintenance and Integrity Constraint Checking: Trading Space for Time. ACM SIGMOD International Conference on Management of Data (SIGMOD). Montreal, Canada, pp 447-458, 1996.

[24] Chirkova, R. & Li, C. & Li, J. Answering queries using materialized views with minimum size. The International Journal on Very Large Data Bases(VLDB J), pages 191-210. 2006.

[25] J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. Proceedings of the International Conference on Very Large Databases, pages 136–145, August 1997.

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

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

[28] T. Phan and W.-S. Li. Load distribution of analytical query workloads for database cluster architectures. In EDBT, pages 169–180, 2008

[29] : Yannis Kotidis, Nick Roussopoulos, ―DynaMat: A Dynamic View Management System for Data Warehouses‖, SIGMOD rec., 28(2):371-382, June 1999

[30] Ravindra N. Jogekar, Ashish Mohod, ―Design and Implementation of Algorithms for Materialized View Selection and Maintenance in Data Warehousing Environment‖, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 9, September 2013.

[31] Jingren Zhou , Per-Ake Larson, Jonathan Goldstein et Luping Ding, ―Dynamic Materialized Views‖, In : Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, 2007. p. 526-535.

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

Reference web graphic

[1]: http://www-igm.univ-mlv.fr/~dr/XPOSE2005/entrepot/intro.html#utilite. [Consulter le 25/02/2021].

[14] : https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_239. [Consulter le 28/02/2021].

[15]: https://docs.snowflake.com/fr/user-guide/views-materialized.html. [Consulter le 14/03/2021].